Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8078710, 16 pages
https://doi.org/10.1155/2018/8078710

Research Article

A Modification of the Fuzzy Logic Based DASH Adaptation Scheme for Performance Improvement

Hyun Jun Kim, Ye Seul Son, and Joon Tae Kim

Konkuk University, Seoul, Republic of Korea

Correspondence should be addressed to Hyun Jun Kim; khkim38@konkuk.ac.kr

Received 1 August 2017; Revised 24 October 2017; Accepted 27 November 2017; Published 5 February 2018

Academic Editor: Mauro Femminella

We propose a modification of the fuzzy logic based DASH adaptation scheme (FDASH) for seamless media service in time-varying network conditions. The proposed scheme (mFDASH) selects a more appropriate bit-rate for the next segment by modification of the Fuzzy Logic Controller (FLC) and estimates more accurate available bandwidth than FDASH scheme by using History-Based TCP Throughput Estimation. Moreover, mFDASH reduces the number of video bit-rate changes by applying Segment Bit-Rate Filtering Module (SBFM) and employs Start Mechanism for clients to provide high-quality videos in the very beginning stage of the streaming service. Lastly, Sleeping Mechanism is applied to avoid any expected buffer overflow. We then use NS-3 Network Simulator to verify the performance of mFDASH. Upon the experimental results, mFDASH shows no buffer overflow within the limited buffer size, which is not guaranteed in FDASH. Also, we confirm that mFDASH provides the highest QoE to DASH clients among the three schemes (mFDASH, FDASH, and SVAA) in Point-to-Point networks, Wi-Fi networks, and LTE networks, respectively.

1. Introduction

Recently, mobile data traffic is increasing rapidly due to the development of wireless communication network and mobile devices. According to the Global Mobile Data Traffic Forecast by Cisco [1], mobile data traffic grew 18-fold from 2011 to 2016, while mobile video traffic accounted for 60% of total mobile data traffic in 2016. Moreover, mobile video traffic is expected to increase 9-fold between 2016 and 2021 and account for up to 78% of total mobile data traffic. In this situation, dynamic adaptive streaming over HTTP (DASH) [2, 3] is adopted globally as a new standard for mobile video streaming to provide seamless video streaming services and utmost QoE to DASH clients in time-varying network conditions. By using HTTP/TCP protocol, DASH can overcome the firewall problem, in contrast with RTP/UDP in the past. It is also cost-effective due to the employment of standard HTTP servers. In DASH, media content is encoded into various versions at different bit-rates and divided into multiple segments, which are used to be played for seconds or tens of seconds. After the division, the segments are stored in the HTTP servers. To obtain utmost QoE, DASH clients act accordingly by requesting segments that are appropriate in the variable network conditions.

Recently, the fuzzy logic [4, 5] based rate adaptation scheme called FDASH [6] has been proposed. FDASH shows better performance than any other rate adaptation schemes from the aspect of the average video bit-rate and the seamlessness of video playback. However, FDASH pays no attention to buffer overflow, which is the cause of video packet loss, and shows a high number of video bit-rate changes, even though buffer overflow and number of video bit-rate changes have a decisive effect on the QoE [7–9]. Therefore, to provide optimal QoE for DASH clients, it is important to reduce the number of video bit-rate changes and to prevent buffer overflow. The more details about FDASH are described in Section 2, "Related Work" part.

In this paper, we propose a modification of FDASH for seamless media services in time-varying network conditions. The proposed scheme (mFDASH) selects a more appropriate bit-rate for the next segment by modification of the Fuzzy Logic Controller (FLC) [10, 11] and reduces the number of video bit-rate changes by applying the Segment Bit-Rate Filtering Module (SBFM). Also, we use the History-Based

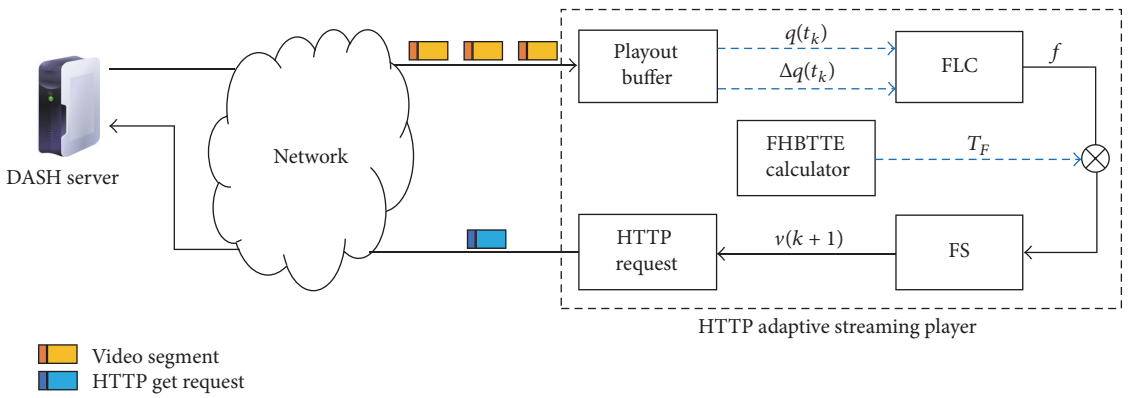


Figure 1: Architecture of FDASH scheme.

TCP Throughput Estimation (HBTTE) to more accurately predict throughputs upon smoothness of throughput and to select optimal next segment bit-rate, which is different from the throughput estimation method applied in FDASH (FHBTTE). By using this, SBFM can be operated more precisely. Also, we apply the Start Mechanism for clients to provide high-quality videos in the very beginning stage of the streaming service. Lastly, we add the Sleeping Mechanism to avoid any expected buffer overflow.

The NS-3 (network simulator) [12] is used to verify the performance of mFDASH. Upon the experimental results, mFDASH shows no buffer overflow within the limited buffer size, which is not guaranteed in FDASH. Also, we confirm that mFDASH provides the highest QoE to DASH clients among the three schemes (mFDASH, FDASH, and Smooth Video Adaptation Algorithm (SVAA)) in Point-to-Point networks, Wi-Fi networks, and LTE networks, respectively.

This paper is structured as follows. Section 2 outlines the related work. Section 3 introduces the FLC used in our scheme, while Section 4 describes the other components. Section 5 presents the simulation environment setting and results, while Section 6 concludes the paper.

2. Related Work

There have been plenty of researches on the rate adaptation scheme, mainly due to the tremendous increase of video traffic and demands for high Quality of Experience (QoE) created by DASH clients. mDASH presented in [13] selects an appropriate segment bit-rate by using the Markov theory [14, 15]. mDASH then judges whether the selected segment bit-rate is optimal or not, by using the reward values of the following factors: (1) buffer occupancy, (2) video playback quality, (3) video rate switching frequency and amplitude, and (4) buffer overflow. Moreover, mDASH considers future reward values to provide better QoE to DASH clients.

Reference [16] proposed a rate adaptation scheme using scalable video coding (SVC) [17]. The scheme uses channel state information in the physical layer together with information of the video content, video buffer, and so on in the application layer, to optimally allocate radio resources to DASH clients in Broadband Wireless Access systems, such as LTE or WiMAX. By using this scheme, more radio resources can be allocated to the closer segments to the playback deadline.

The rate adaptation scheme called SVAA [18] uses buffer occupancy and buffer trend to select the segment bit-rate. Also, SVAA uses the estimated throughput value, which is estimated using the HBTTE [19] method and the real throughput value to select a more appropriate segment bit-rate. In addition, the SVAA scheme applies a buffer cap, which limits the buffer occupancy from overflowing.

Recently, the fuzzy logic based rate adaptation scheme (FDASH) has been proposed. Figure 1 shows the architecture of FDASH scheme. At first, DASH clients receive segments from DASH server and store them in their Playback Buffers. Then, the information about buffer occupancy and differential of buffer occupancies is conveyed to the FLC of FDASH as two input variables. With those two input variables, FLC calculates the next segment bit-rate (i.e., $f \times T_F$, where T_F is the throughput estimation value calculated in FHBTTE Calculator) and then FDASH scheme (FS) decides to request the next segment with the calculated bit-rate or just to request the next segment with the previously requested bit-rate.

3. Fuzzy Logic Controller

In this section, we describe the FLC that we apply in our scheme. As a control system based on fuzzy logic, the FLC consists of a Fuzzification Interface, Knowledge Base, Decision-Making Logic, and Defuzzification Interface, as shown in Figure 2.

(1) The Knowledge Base. The Knowledge Base (KB) comprises a data base and a rule base. The data base provides the definitions that are necessary to use fuzzy values (input/output linguistic variables) and fuzzy rules. Among the definitions, membership functions of linguistic variables represent the degree of belonging of the crisp input/output to the linguistic variables. In addition, fuzzy rules in the rule base are simple if-then rules and consist of linguistic variables. The fuzzy rules decide the degree of output linguistic variables by using the input linguistic variables.

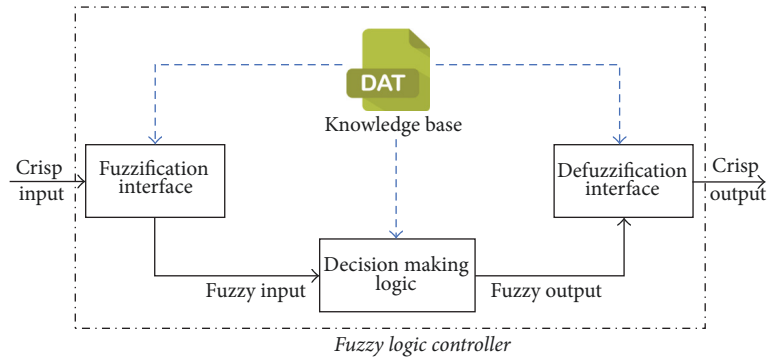


Figure 2: The fundamental architecture of FLC.

(2) The Fuzzification Interface. In the Fuzzification interface, a crisp input is mapped into the universe of discourse, and the fuzzy input values (i.e., input linguistic variables) are determined by the mapped value and the defined input membership functions in the KB.

(3) Decision-Making Logic. In the Decision-Making Logic, fuzzy output (i.e., linguistic variable) is deduced from fuzzy inputs by the fuzzy rules following a format described below:

$$\begin{aligned} &\textbf{if} \quad iLi_1 \ and/or \ iLi_2 \\ &\textbf{then} \quad oLi_i, \end{aligned} \tag{1}$$

where iLi_i is a input linguistic variable for all i and oLi_i is a output linguistic variable for all i. If the operator is *and*, then oLi_i is the minimum value between iLi_1 and iLi_2. In the case of *or*, oLi_i is the maximum value. Also, the number of input linguistic variables can be increased.

(4) The Defuzzification Interface. In the Defuzzification Interface, output linguistic variables are mapped into the universe of discourse, and crisp output is calculated with the mapped value. Several methods exist in Defuzzification method: (1) centroid; (2) bisector; (3) middle of maximum; (4) largest of maximum; and (5) smallest of maximum [20]. In our FLC, the centroid method is applied.

Like the FLC of FDASH, FLC in mFDASH has two input variables and one output variable f. The variable f stands for the amount of decrease or increase of the HBTTE value and is used as follows:

$$v(k+1) = Q(f \times T_k), \tag{2}$$

where T_k is the HBTTE value and $v(k+1)$ is the next segment bit-rate. Since the available segment bit-rates of a media content in a DASH server are discrete, $f \times T_k$ is quantized by $Q(\cdot)$ to the highest value that is lower than $f \times T_k$.

Figures 3 and 4 show the membership functions of the input and output variables in FLC of FDASH and mFDASH, respectively. Buffer occupancy means the amount of data in a buffer when the latest requested kth segment is completely received and is denoted by $q(t_k)$. Three linguistic variables [short (S), close (C), and long (L)] are selected to depict the state of the buffer occupancy in both FLC. In our scheme, the ranges of membership functions of $q(t_k)$ in FDASH scheme are modified as follows.

(1) As shown in Figure 3(a), the range of $q(t_k)$ (i.e., *short* becomes 1) in FDASH scheme is $[0, 2T/3]$, where T denotes the ideal buffer occupancy. Considering $2T/3$ is large enough (e.g., if $T = 70$ s, $2T/3 = 46.67$ s), the segment bit-rate can be reduced before the buffer is fully used since the FLC judges that the buffer is in the risk of underflow even though 46.67 s does not indicate the shortage of buffer. Therefore, in mFDASH, the range is modified to $[0, T/3]$ to fully use the buffer.

(2) Also, we changed the range of $q(t_k)$ (i.e., *long* becomes 1) from $[4T, \infty]$ to $[2T, \infty]$, since $4T$ is too large value as the reference point of buffer overflow. With this modification, FLC decides more appropriate bit-rate of the next segment when the buffer occupancy is close to the buffer limit.

Figures 3(b) and 4(b) show the membership functions of differential of the buffer occupancies. The differential of the buffer occupancies is denoted as $\Delta q(t_k) = q(t_k) - q(t_{k-1})$, which is the difference of the buffer occupancies between the case of the kth segment completely received and the case of the $(k-1)$th segment completely received. Similar to $q(t_k)$, the status of the second input ($\Delta q(t_k)$) is depicted by three linguistic variables [falling (F), steady (S), and rising (R)]. The following modifications are applied to the ranges of the membership functions of $\Delta q(t_k)$ in Figure 3(b).

(1) Our analysis of the simulation results confirmed that even if the network condition is congested, $\Delta q(t_k)$ never drops below $-T/3$ s. For that reason, we modify the range (i.e., *falling* becomes 1) from $[-\infty, -2T/3]$ to $[-\infty, -T/3]$, in order for $\Delta q(t_k)$ to be reflected in the FLC more accurately.

(2) Also, we modify the range (i.e., *rising* becomes 1) from $[4T, \infty]$ to $[\tau, \infty]$. It is clear that $\Delta q(t_k)$ cannot be higher than τ, even if the network condition is extremely good. By this modification, the FLC can more accurately judge $v(k+1)$ in the good network condition.

As shown in Figure 3(c), five linguistic variables (reduce (R), small reduce (SR), no change (NC), small increase (SI), and increase (I)) are used in FLC of FDASH. But for the output variable (f) in mFDASH, only three linguistic variables [reduce (R), no change (NC), and increase (I)] are used with the expectation that the less bit-rate changes will occur if the less linguistic variables are used. Figure 4(c) shows that the

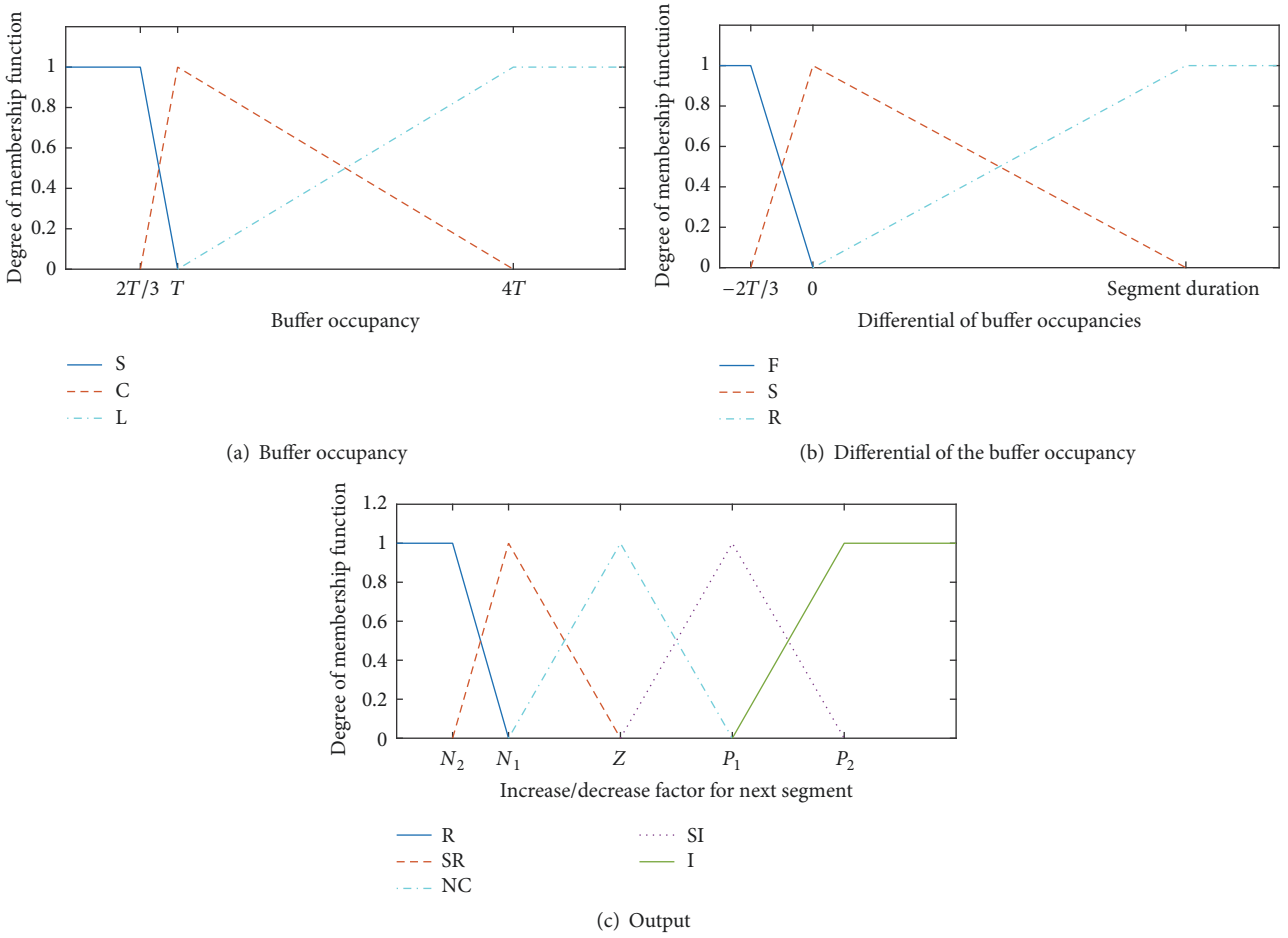


FIGURE 3: Membership functions of the input and the output of in FDASH.

ranges of the output membership functions are divided into the fractionation of f into N, Z, and P. N is a factor that decreases $v(k+1)$ compared to the HBTTE value, and the range is $N \in [0, 1]$. The second factor Z with the value of 1 retains $v(k+1)$ as $v(k)$. The last factor P with the range of $P \in [1, \infty]$ increases $v(k+1)$ compared to the HBTTE value.

The fuzzy rules are shown below.

Fuzzy Rules

(1) **if** short **and** falling **then** R: $r_1 = \min(S, F)$

(2) **if** close **and** falling **then** R: $r_2 = \min(C, F)$

(3) **if** short **and** steady **then** R: $r_3 = \min(S, S)$

(4) **if** long **and** falling **then** NC: $r_4 = \min(L, F)$

(5) **if** close **and** steady **then** NC: $r_5 = \min(C, S)$

(6) **if** short **and** rising **then** NC: $r_6 = \min(S, R)$

(7) **if** long **and** steady **then** I: $r_7 = \min(L, S)$

(8) **if** close **and** rising **then** I: $r_8 = \min(C, R)$

(9) **if** long **and** rising **then** I: $r_9 = \min(L, R)$

The values of R, NC, and I can be calculated as (3), (4), and (5), while f is represented as (6) using the centroid method:

$$\mathrm{R} = \sqrt{r_1^2 + r_2^2 + r_3^2} \tag{3}$$

$$\mathrm{NC} = \sqrt{r_4^2 + r_5^2 + r_6^2} \tag{4}$$

$$\mathrm{I} = \sqrt{r_7^2 + r_8^2 + r_9^2} \tag{5}$$

$$f = \frac{N \times \mathrm{R} + Z \times \mathrm{NC} + P \times \mathrm{I}}{\mathrm{R} + \mathrm{NC} + \mathrm{I}} \tag{6}$$

4. mFDASH Scheme

In connection with the architecture of Figure 1, Figure 5 shows the fundamental architecture of mFDASH. As shown in Figure 5, mFDASH consists of previously mentioned components (SBFM, Sleeping Mechanism, HBTTE Calculator, and Start Mechanism) as well as the components (FLC, Play-out Buffer, and HTTP Request) already existing in FDASH. Although the FLCs in both schemes use the same inputs, the detail operations are modified as described in Section 3. In this section all the components other than the FLC are described.

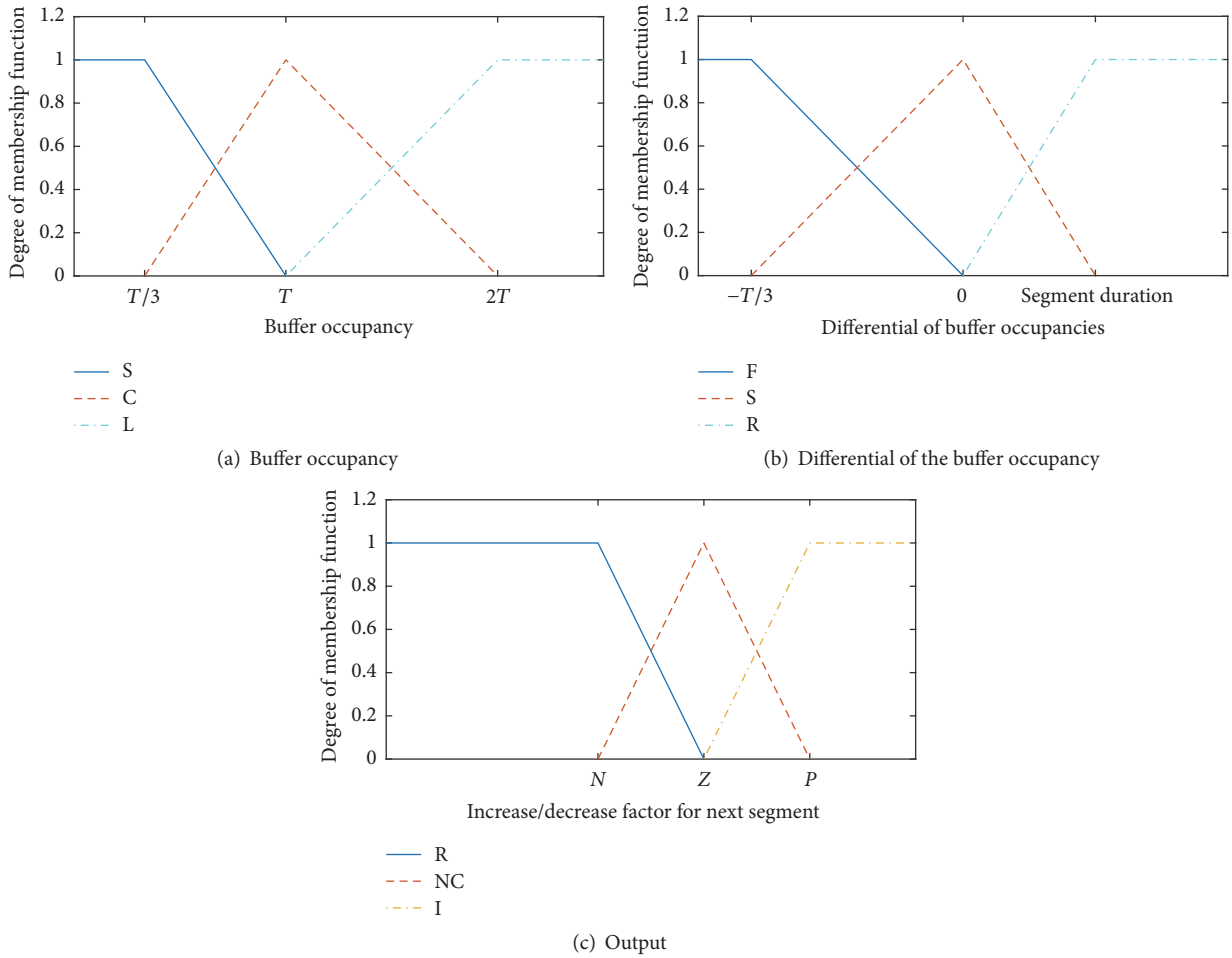


FIGURE 4: Membership functions of the input and the output in mFDASH.

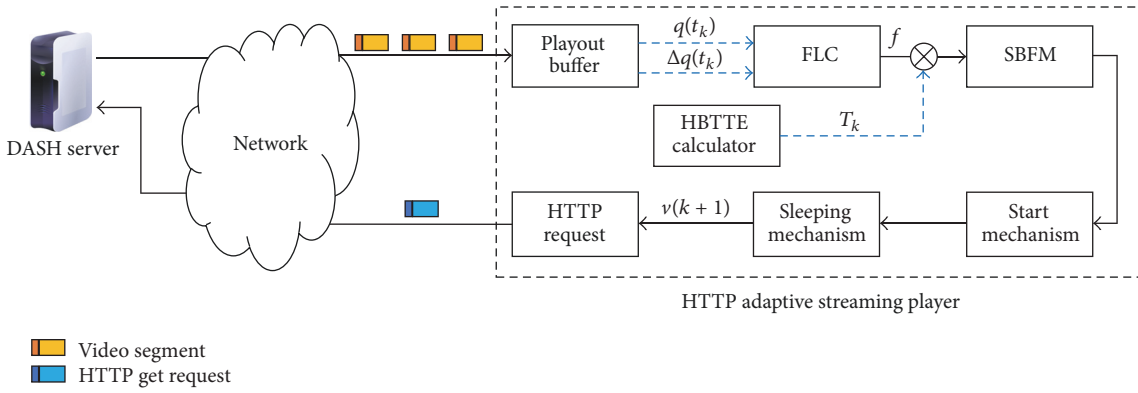


FIGURE 5: Architecture of mFDASH scheme.

4.1. The Throughput Estimation Method. FDASH scheme judges whether to request the next segment of $v(k+1)$ or to keep the segment bit-rate as the previous one ($v(k)$), by using the average value of the throughputs of the segments downloaded within the past S_{window}s. This method (FHBTTE) greatly smoothens the available bandwidth, but it cannot detect outliers and level shifts of throughputs, which lead to inaccurate throughput estimation values. Thus, mFDASH applies the HBTTE method [19], which conducts the same function as the FHBTTE and additionally detects the outliers and the level shifts of throughputs.

Figure 6 shows the difference between the two throughput estimation methods in a Wi-Fi network. The FHBTTE values cannot predict the actual TCP throughput value properly during 0~70 s, since it cannot detect the level shifts. On the other hand, because of the detections of the level shifts, the

```
(1)  v̂ = f × T;
(2)  v(k + 1) = Q(v̂);
(3)  if v(k + 1) > v(k) then
(4)     if T_k/v(k + 1) > a and q(t_k) < q_high then
(5)        v(k + 1) = v(k);
(6)     end if
(7)  else if v(k + 1) < v(k) then
(8)     if T_k/v(k + 1) < b and q(t_k) > q_low then
(9)        if q_low−flag = true then
(10)          q_low−flag = false;
(11)       end if
(12)       v(k + 1) = v(k);
(13)    end if
(14)  if q(t_k) < q_low and q(t_k) > q_min and q_low−flag = false then
(15)     q_low−flag = true;
(16)  else if q(t_k) < q_low and q(t_k) > q_min and q_low−flag = true then
(17)     v(k + 1) = v(k);
(18)  end if
(19) end if
```

Algorithm 1: The segment rate filtering module.

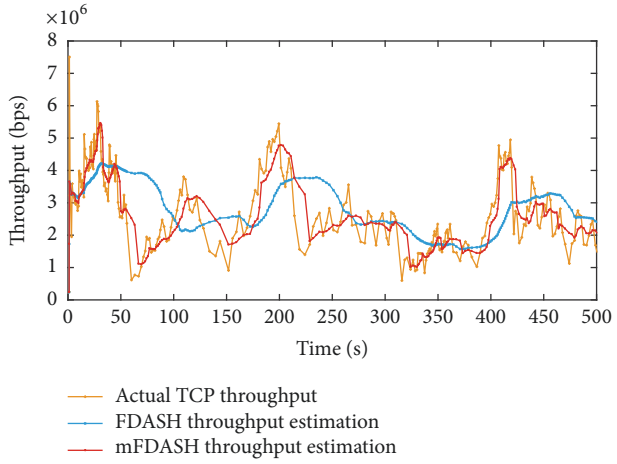


Figure 6: Comparison of throughput estimation.

HBTTE method predicts the actual TCP throughputs more accurately than the FHBTTE method does. Also, the HBTTE method shows appropriate smoothness with the detection of outliers. With this method, mFDASH can choose a proper segment bit-rate that leads the $q(t_k)$ of the client into the safe region and provides a higher quality of video.

4.2. The Segment Bit-Rate Filtering Module. The output variable f is decided by $q(t_k)$ and $\Delta q(t_k)$. For this reason, if the video quality is changed, $q(t_k)$ and $\Delta q(t_k)$ are also changed in a moment. As the preceding changes directly influence f, so the next f will be changed sequentially. Therefore, oscillation of the segment bit-rate will occur if $v(k+1)$ is decided only by using f. To solve this problem, we additionally apply the SBFM to decide whether the current segment bit-rate will be changed to $v(k+1)$ or not.

Algorithm 1 is the pseudocode of the SBFM. The three factors named q_{high}, q_{low}, and q_{min} indicate the maximum buffer occupancy, the threshold that prevents a sharp decline of the segment bit-rate, and the threshold that prevents the buffer underflow, respectively; and the size-order of these factors is $q_{\text{high}} > q_{\text{low}} > q_{\text{min}}$. After receiving $v(k+1)$ from FLC, the SBFM decides whether $v(k+1)$ is appropriate as the bit-rate of the next segment or not.

If $v(k+1)$ is larger than $v(k)$, we need to consider T_k and $q(t_k)$, to avoid oscillation of the bit-rate and buffer overflow. Thus, we only increase the segment bit-rate if T_k is larger than $v(k+1)$ above a certain level, or if $q(t_k)$ is larger than the buffer size. If $v(k+1)$ is smaller than $v(k)$, this indicates that $q(t_k)$ is on the verge of depletion. Therefore, to prevent frequent bit-rate change and depletion of buffer, we need to reduce the segment bit-rate if the ratio of $v(k+1)$ is larger than T_k over a certain level, or $q(t_k)$ is smaller than q_{low}.

Additionally, as the segment bit-rate has the possibility of being reduced continuously when $q(t_k)$ drops below q_{low}, we apply $q_{\text{low-flag}}$ to prevent the segment bit-rate from being reduced rapidly. The segment bit-rate decreases only one time when $q_{\text{low-flag}}$ is false, and $q(t_k)$ is between q_{low} and q_{min}. After the decrease of the segment bit-rate, $q_{\text{low-flag}}$ is changed into true, and the segment bit-rate is unchanged while $q_{\text{low-flag}}$ is true. Despite the precedent decrease of the segment bit-rate, if $q(t_k)$ drops below q_{min}, this indicates that the buffer will soon be depleted. So, the segment bit-rate should be decreased rapidly, until $\Delta q(t_k)$ is changed into positive. Then if $v(k+1)$ is larger than $v(k+1)$ again, $q_{\text{low-flag}}$ is reversed to false.

4.3. The Start Mechanism. Due to the SBFM, there is a possibility that a DASH client watches a low-quality video for a certain period of time in the very beginning of a streaming service. Therefore, in mFDASH, we apply the Start Mechanism to guarantee higher quality video at the start of

(1) $v_{\text{BEGIN}} = \widehat{Q}(T_k/c)$
(2) **if** $Begin - flag = false$ **then**
(3) **if** $T_k > T_{k-1}$ **then**
(4) $v(k+1) = v_{\text{BEGIN}}$
(5) **else**
(6) $Begin - flag = true$
(7) **end if**
(8) **end if**

Algorithm 2: The Start Mechanism.

streaming service and to prevent buffer underflow which can occur due to the rapid increase of segment bit-rate.

Algorithm 2 is the pseudo code of the Start Mechanism. Before the start of the Start Mechanism, the first segment is requested with the lowest bit-rate (v_{lowest}). After that, the Start Mechanism begins. In every iteration, v_{BEGIN} is first set as $\widehat{Q}(T_k/c)$, where $\widehat{Q}(\cdot)$ and c denote a quantization function that outputs a lowest value that is higher than an input value and a control factor for the first segment bit-rate, respectively. The factor c prevents buffer underflow in the case that T_k suddenly decreases. Then if T_k is larger than T_{k-1}, that is, there is enough bandwidth to increase the segment bit-rate, the bit-rate is then switched to a higher bit-rate. On the other hand, if T_{k-1} becomes higher than T_k, then Begin − flag is set to true, and the Start Mechanism is halted.

4.4. The Sleeping Mechanism. In general, the buffer size of a mobile device is limited. Thus, during a streaming service, if $q(t_k)$ exceeds the maximum buffer size, that is, q_{high}, the mFDASH client remains idle for $q(t_k) - q_{\text{high}}$ s, before sending out the request for the next segment.

5. Performance Evaluation

In this section, we evaluate mFDASH scheme by analyzing the results of the experiments conducted in the NS-3. The results of mFDASH are compared with the results of FDASH and SVAA schemes. Our experiments are implemented in three Point-to-Point network conditions, ten Wi-Fi network conditions, and two LTE network conditions. In three Point-to-Point network conditions, a DASH server and a DASH client are connected by a Point-to-Point link. First link network has constant link capacity. Also, there are long-term changes in the second Point-to-Point link capacity and short-term periodic picks and falls lasting 10 s in the third Point-to-Point link capacity. In Wi-Fi network conditions, there is a DASH server, a mobile (or TV set-top box) DASH client, and five background traffic events. We implement ten different experiments in Wi-Fi network conditions by changing the background traffic. In LTE network conditions, there are two scenarios. The first scenario consists of one mobile DASH client (SVAA, FDASH, or mFDASH) with five background traffic events and the second one consists of three mobile DASH clients (one for each algorithm) and five background traffic events. In each session the average segment bit-rate (m_{avq}), the number of bit-rate changes ($\text{Num}_{\text{change}}$), the total number of interruptions (Int), and the existence of buffer overflow are described. Moreover, to evaluate the performance of the proposed schemes realistically, we use a QoE model proposed in [21]. Furthermore, in the last part of this section, the time taken by each scheme is compared with each other to show the complexity of the proposed method.

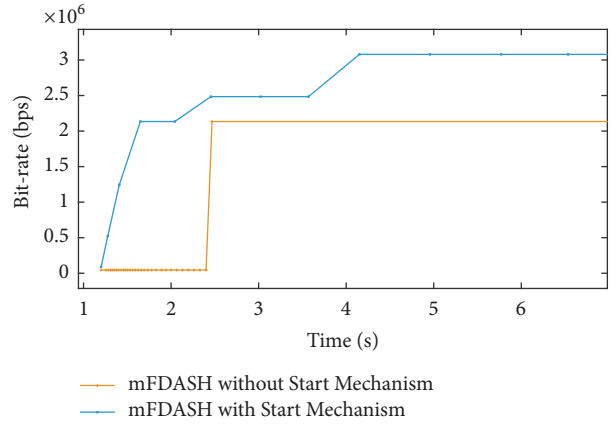
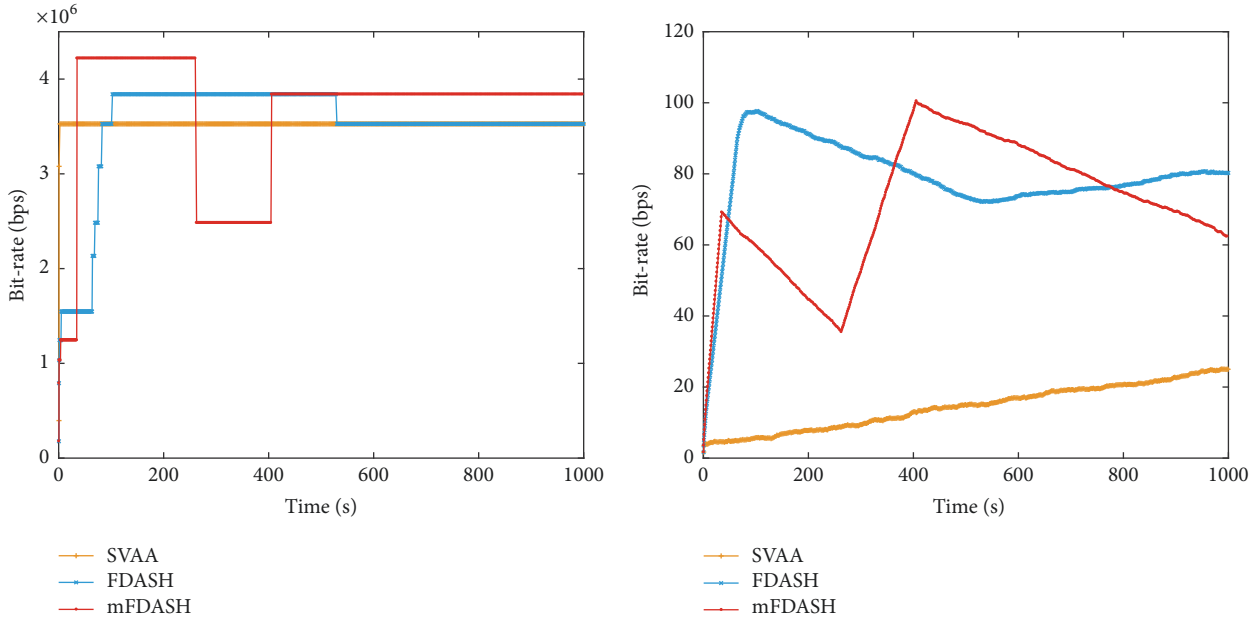


Figure 7: Segment bit-rate variations at the very beginning of streaming service.

In our experiments, we set $q_{\text{high}} = 100$ s; that is, we assume that the buffer sizes of all mobile devices in the simulation are 100 s, since this is reasonable considering the performance of current mobile devices. The ideal buffer occupancy (T) is set to 70 s, and q_{low} is set to 10 s. Also, q_{min} is set to 7 s, since it is the minimum buffer occupancy to avoid buffer underflow and to use the buffer most efficiently. The factors of output membership functions in FLC (N, Z, P) are set to 0.8, 1, and 1.3, respectively, to select optimal segment bit-rate. The two factors a and b used in SBFM are set to 0.85 and 1.3, respectively, to change the segment bit-rate properly when the network condition is extremely good or bad. Also, we set $c = 3$ to select a proper segment bit-rate when the streaming service starts. In our experiments, the DASH server provides twenty different versions of segment bit-rate: 45, 89, 131, 178, 221, 263, 334, 396, 522, 595, and 791 kbps; and 1.033, 1.245, 1.547, 2.134, 2.484, 3.079, 3.527, 3.840, and 4.220 Mbps with the segment duration of $\tau = 2$ s and all the videos are played for 1000 s. Also, to simplify the simulation, we assume that the duration between each MPEG frame is 20 ms (i.e., 50 frames per second) and each segment contains 100 MPEG frames with each frame size v where v is uniform random variable within $[0, 2 \times average\_packet\_size]$. The maximum frame size is set to $2 \times average\_packet\_size$, where $average\_packet\_size = Segment\ Bitrate/(50 \times 8)$ bytes, to make the expected value of v as $average\_packet\_size$.

5.1. Effect of the Start Mechanism. Figure 7 shows the segment bit-rate variations at the very beginning of the streaming service in a Wi-Fi network for the two cases, which include one with the Start Mechanism and another without the Start Mechanism. X-axis indicates the simulation time and Y-axis indicates the bit-rate of segments. In Figure 7, one point indicates one downloaded video segment. At first, the



(a) Segment bit-rate (b) Buffer occupancy

Figure 8: Simulation result in constant link Point-to-Point network.

Table 1: Performance in long-term bandwidth variation Point-to-Point network.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	3.51287 Mbps	3.31133 Mbps	3.35397 Mbps
Number of bit-rate changes (Num_{change})	3	11	7
Total number of interruptions (Int)	0	0	0

streaming service starts with the lowest bit-rate (45 kbps). In the case of mFDASH without Start Mechanism, 30 segments with the lowest bit-rate are fetched by the mFDASH client and stored in the Playback Buffer; that is, DASH clients should watch the lowest quality video for 60 s ($30 \times \tau$ s). On the other hand, in the case of mFDASH with Start Mechanism, the DASH client can watch higher quality of video, since the DASH client keeps increasing the segment bit-rate as much as possible by Start Mechanism.

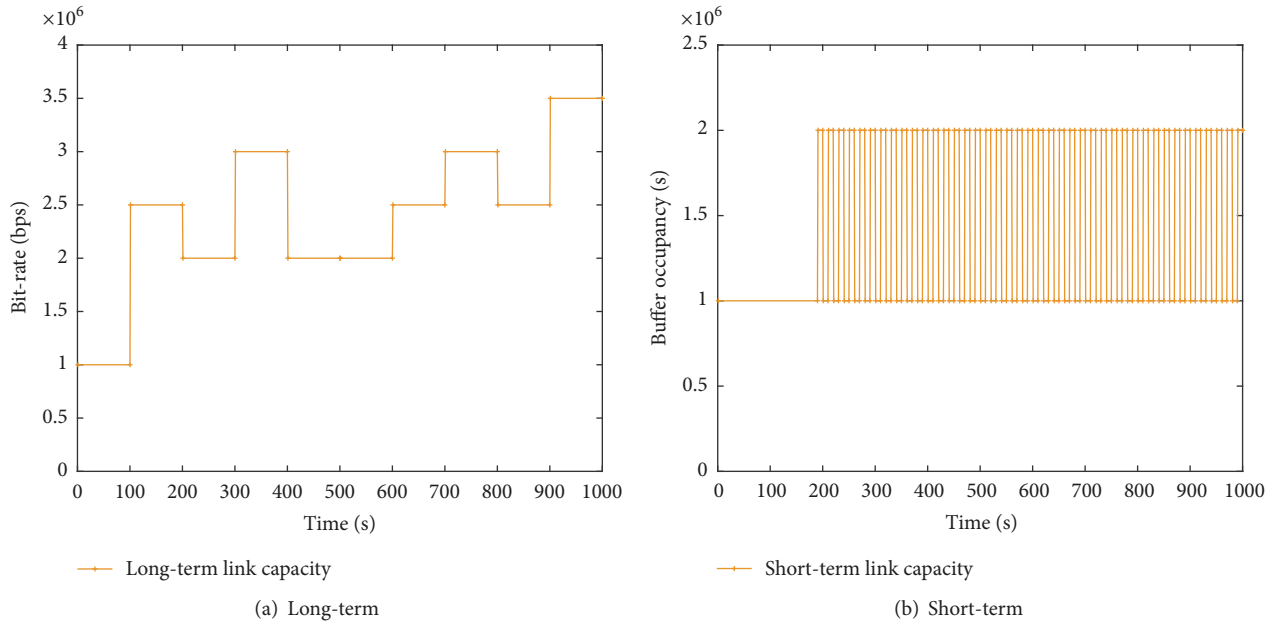
5.2. Point-to-Point Network with Constant Link Capacity (C-P2P). Figure 8 is the simulation results in Point-to-Point link with constant link capacity. The link capacity of the constant link is 4 Mbps. The segment bit-rates that are fetched by each DASH client are shown in Figure 8(a). Also, the buffer occupancy of each DASH client is described in Figure 8(b).

Table 1 shows m_{avq}, Num_{change}, and Int of DASH clients in C-P2P. First, all the schemes show zero Int. mFDASH shows almost similar m_{avq} with FDASH client and lower Num_{change} than FDASH. Particularly, SVAA shows the highest m_{avq} and the lowest Num_{change}. However, this does not mean that SVAA is the best scheme and the reason of this will be described with all the other simulation results in Section 5.7.

5.3. Point-to-Point Network with Long-Term Bandwidth Changes (L-P2P). Figure 10 shows the experiment results when the DASH server is connected with the client by a L-P2P. The link capacity of the L-P2P in Figure 10 is described in Figure 9(a), and it changes sequentially to 1, 2.5, 2, 3, 2, 2, 2.5, 3, 2.5, and 3 Mbps for every 100 s. Figure 10(a) shows the bit-rate of segments that are downloaded from SVAA, FDASH, and mFDASH clients. Also, Figure 10(b) shows each buffer occupancy ($q(t_k)$) of SVAA, FDASH, and mFDASH. In the case of FDASH, buffer overflow occurs since $q(t_k)$ of FDASH rises over 120 s with the buffer limit of 100 s. In other words, the FDASH clients lose several video segments due to the buffer overflow, which results in decrease of the clients' QoE. On the other hand, SVAA clients avoid buffer overflow due to the buffer cap, and mFDASH clients avoid buffer overflow due to the Sleeping Mechanism.

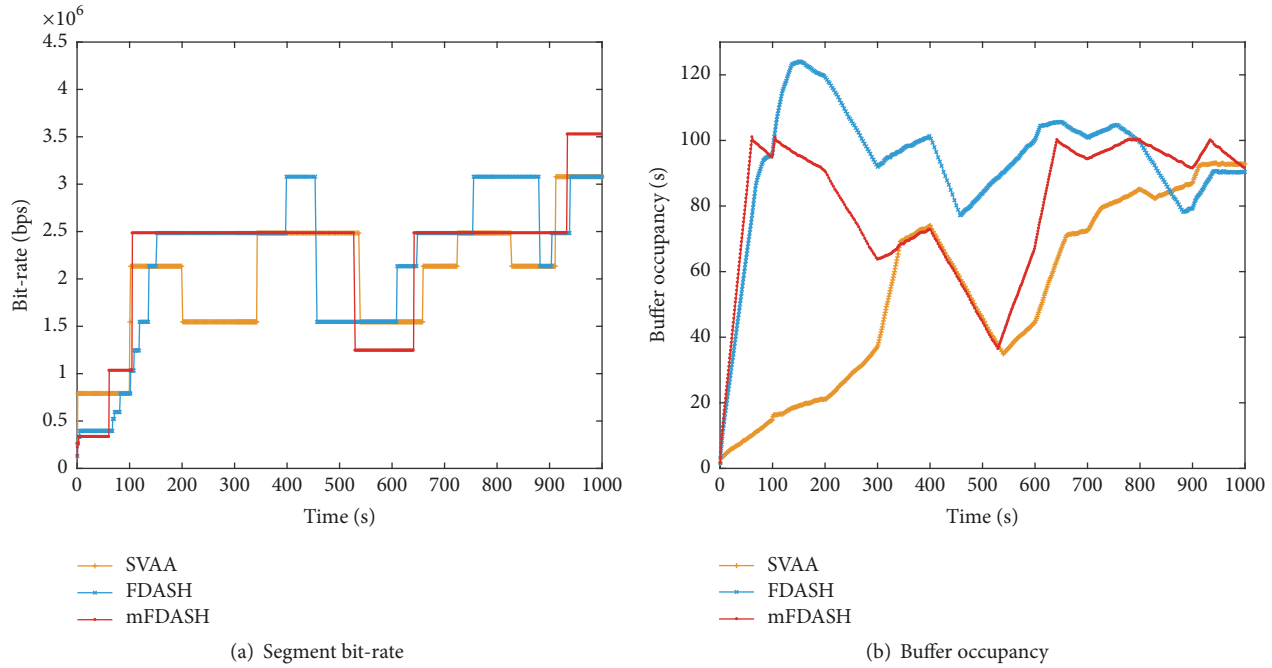
Table 2 summarizes m_{avq}, the number of bit-rate changes, and the number of video interruptions of DASH clients in L-P2P. The mFDASH client has almost the same m_{avq} compared to those of other schemes. Moreover, mFDASH shows the lowest number of bit-rate changes. Also, all the three schemes show no playback interruptions.

5.4. Point-to-Point Network with Short-Term Bandwidth Changes (S-P2P). Figure 11 shows the experimental results when the available bandwidth of the Point-to-Point link has short-term periodic picks and falls lasting 10 s. The link capacity of the S-P2P in Figure 11 is described in Figure 9(b).



(a) Long-term (b) Short-term

Figure 9: Link capacity of links with bandwidth variations.

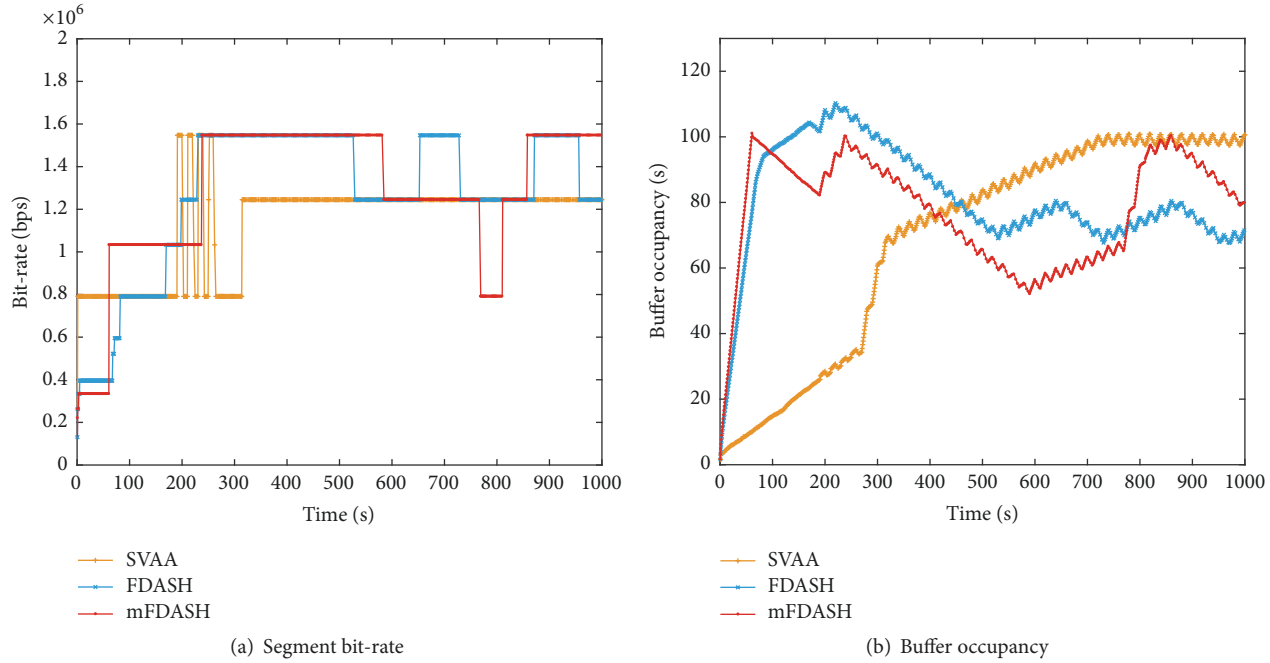


(a) Segment bit-rate (b) Buffer occupancy

Figure 10: Simulation result in long-term bandwidth variation Point-to-Point network.

Table 2: Performance in long-term bandwidth variation Point-to-Point network.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	1.865 Mbps	1.889 Mbps	1.852 Mbps
Number of bit-rate changes (Num_{change})	11	20	9
Total number of interruptions (Int)	0	0	0



(a) Segment bit-rate (b) Buffer occupancy

Figure 11: Simulation result in periodic short-term bandwidth variation Point-to-Point network.

Table 3: Performance in periodic short-term bandwidth variation Point-to-Point network.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	1.106 Mbps	1.116 Mbps	1.130 Mbps
Number of bit-rate changes (Num_{change})	14	15	10
Total number of interruptions (Int)	0	0	0

Table 4: Average performance value of ten Wi-Fi networks with a mobile dash client and five background traffic events.

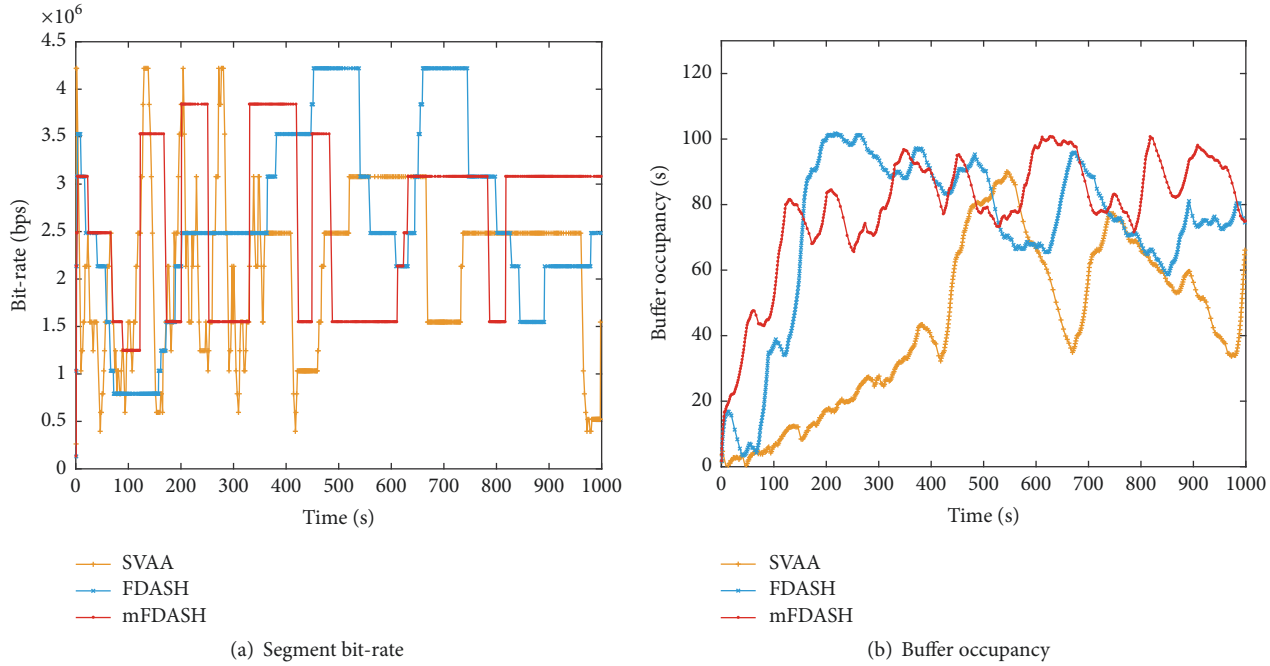
Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	2.166 Mbps	1.975 Mbps	2.205 Mbps
Number of bit-rate changes (Num_{change})	111.2	39.7	20.1
Total number of interruptions (Int)	23	0	0

The link capacity is 1 Mbps during 0–190 s and after 190 s, the short-term variation starts between 1 and 2 Mbps. Figure 11(a) describes the bit-rate of segments fetched by SVAA, FDASH, and mFDASH clients. Also, each $q(t_k)$ of SVAA, FDASH, and mFDASH is shown in Figure 11(b), and the results indicate that mFDASH and SVAA have lower possibilities of the occurrence of buffer overflow because of the Sleeping Mechanism and the buffer cap. Also, according to Figures 11(a) and 11(b), it is clearly shown that even if there are short-term periodic picks and falls during 200 s–1000 s, all the three algorithms operate normally without showing frequent bit-rate switching.

Table 3 shows m_{avq}, the number of bit-rate changes, and the number of video interruptions of DASH clients. According to Table 3, the mFDASH client shows the highest m_{avq} and the lowest number of bit-rate changes. Also, there is no video interruptions in any DASH clients.

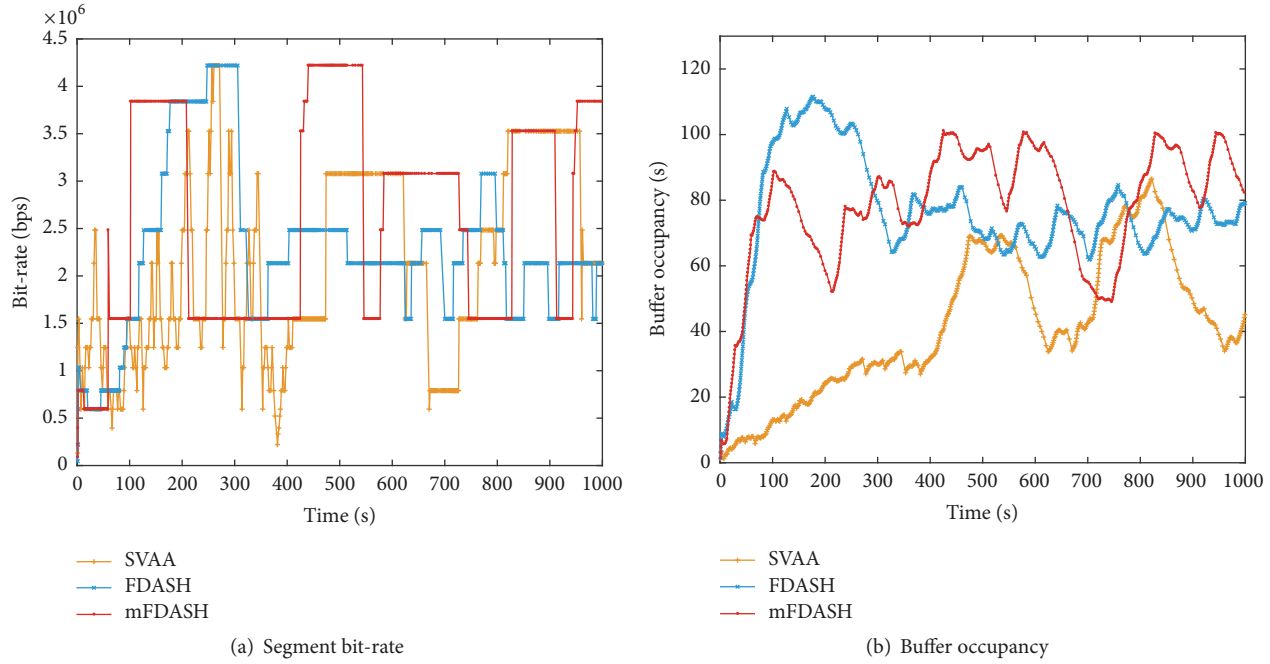
5.5. Wi-Fi Network. In our experiment, we evaluate mFDASH in ten different Wi-Fi network conditions. In this section, we only attach three cases because of the paper space consideration. However, of coarse, we reflected all the simulation results in the calculation of performance index in Table 4.

Figures 12(a), 13(a), and 14(a) show the bit-rates of the segments that are fetched by the mobile DASH clients. Also, Figures 12(b), 13(b), and 14(b) show the mobile DASH clients' $q(t_k)$. Each $q(t_k)$ of the mobile FDASH client in Figures 13(b) and 14(b) exceeds q_{high} as shown below and those facts can decrease the QoE of mobile FDASH clients. Meanwhile, each $q(t_k)$ of the mFDASH and SVAA client never exceeds q_{high}, even if the network conditions are poor and unpredictable. In addition, as shown in Figure 12(b) and Table 4, the SVAA client experiences playback interruptions while other clients experience no playback interruptions.



(a) Segment bit-rate (b) Buffer occupancy

Figure 12: First simulation result in Wi-Fi network with a mobile DASH client and five background traffic events.

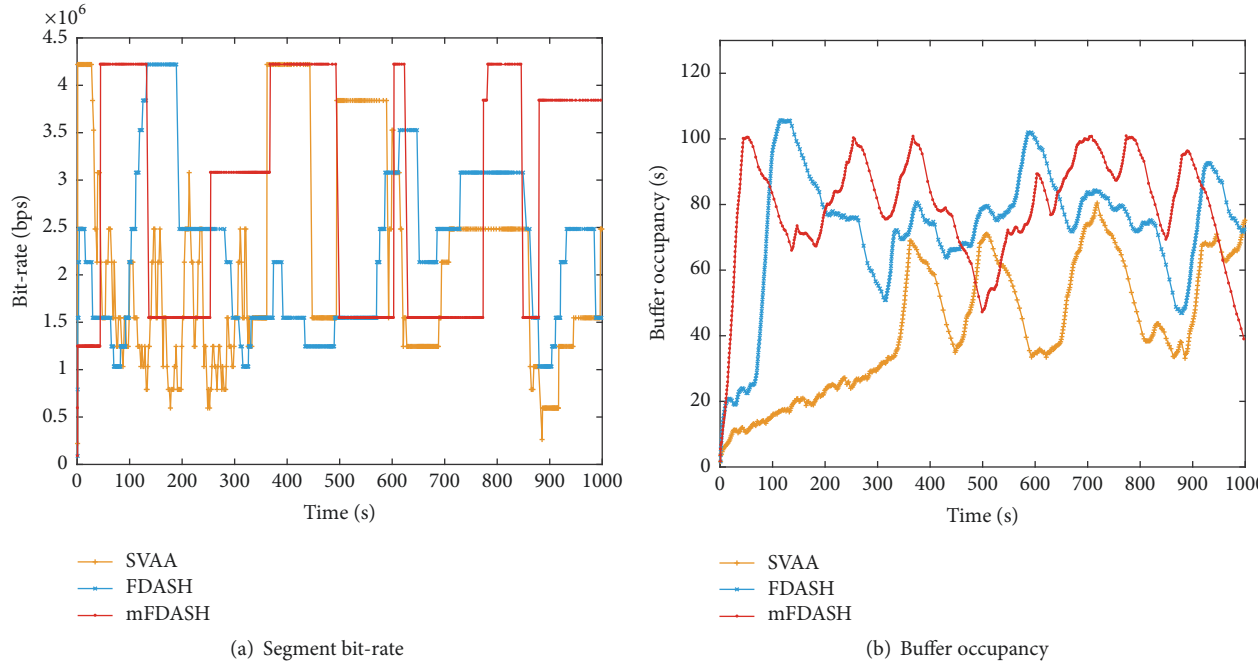


(a) Segment bit-rate (b) Buffer occupancy

Figure 13: Second simulation result in Wi-Fi network with a mobile DASH client and five background traffic events.

The average value of m_{avq} and the number of bit-rate changes of the ten Wi-Fi experiments are also outlined in Table 4, as well as the Int. Table 4 shows that mFDASH has 11.6%, 1.8% higher average m_{avq}, 49.4%, and 81.9% lower average Num_{change} than FDASH and SVAA, respectively. As shown in the results in Table 4, SVAA does not selects optimal bit-rates in every time, since high average m_{avq} is accompanied by high Num_{change}. This means that SVAA operate inefficiently to find optimal bit-rate. FDASH, on the other hand, selects a bit lower bit-rates than SVAA with lower bit-rate changes. This means that FDASH can not find optimal bit-rate in particular network and hardware conditions. In contrast, mFDASH shows highest m_{avq} and lowest Num_{change}, which means that mFDASH can find optimal bit-rate very efficiently.

In addition to those results with mobile DASH clients, we also simulate with a fixed wireless TV set-top box DASH client to prove that mFDASH works well in clients with



(a) Segment bit-rate (b) Buffer occupancy

Figure 14: Third simulation result in Wi-Fi network with a mobile DASH client and five background traffic events.

Table 5: Performance in Wi-Fi networks with a TV set-top box DASH client and five background traffic events.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	2.235 Mbps	1.993 Mbps	2.387 Mbps
Number of bit-rate changes (Num_{change})	103	39	20
Total number of interruptions (Int)	0	0	0

Table 6: Performance in LTE network with a mobile DASH client and five background traffic events.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	0.720 Mbps	0.670 Mbps	0.667 Mbps
Number of bit-rate changes (Num_{change})	193	15	8
Total number of interruptions (Int)	0	0	0

various types of device. Wireless TV set-top box has powerful hardware, so we assume that set-top box DASH clients has no buffer limit. Also, we use the background traffic used in sixth Wi-Fi simulation. All the other configurations are the same as other simulations. Figure 15 shows the simulation results with one TV set-top box DASH clients and 5 background traffic events. Table 5 outlines the performance evaluation indicators that are the same as Tables 2, 3, and 4. As shown on Table 5, mFDASH provides the highest m_{avq} and the lowest Num_{change}.

5.6. LTE Networks. We also evaluate our scheme in two LTE networks. Briefly outlining the configuration of both LTE networks, there is only one base station that serves the mobile DASH clients and 50 Physical Resource Blocks (PRBs) are used as radio resources. Also, the Extended Vehicular A (EVA) fading model is used as a fading model and Log-Distance Propagation Loss model is used. Moreover, Proportional Fair resource allocation scheme is used as the resource allocation scheme in the base station.

Same as described earlier, Figure 16(a) represents the segment bit-rates downloaded by mobile DASH clients and Figure 16(b) represents mobile DASH clients' buffer occupancies. Also, performance results of the first LTE simulation are outlined in Table 6. FDASH and mFDASH show almost the same m_{avq} while SVAA shows the highest m_{avq}. Nevertheless, SVAA is not the best adaptation scheme since high m_{avq} is accompanied by high Num_{change}.

Figure 17 shows the simulation result in the second LTE network. In that LTE network, three rate adaptation schemes are applied in each one of the three mobile DASH clients. As shown in Table 7, mFDASH provides the highest m_{avq} to a mobile DASH client and, most importantly, shows the least Num_{change}.

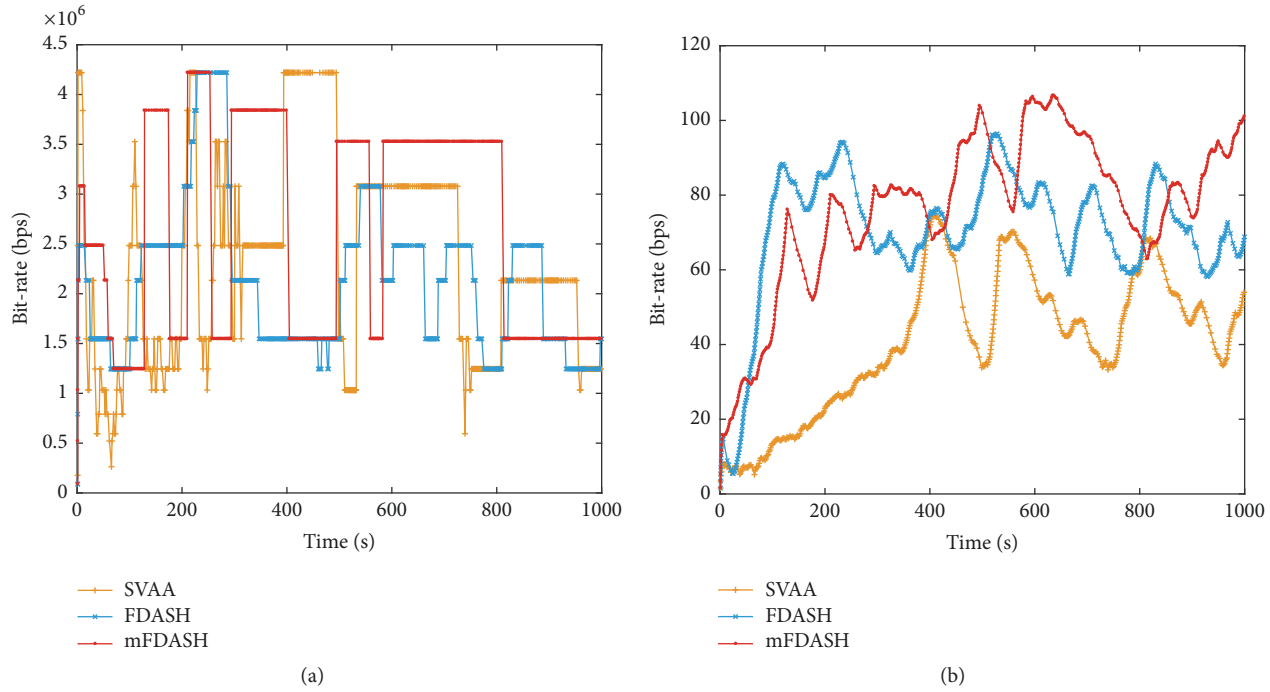


Figure 15: Simulation result in Wi-Fi network with a TV set-top Box DASH client and five background traffic events.

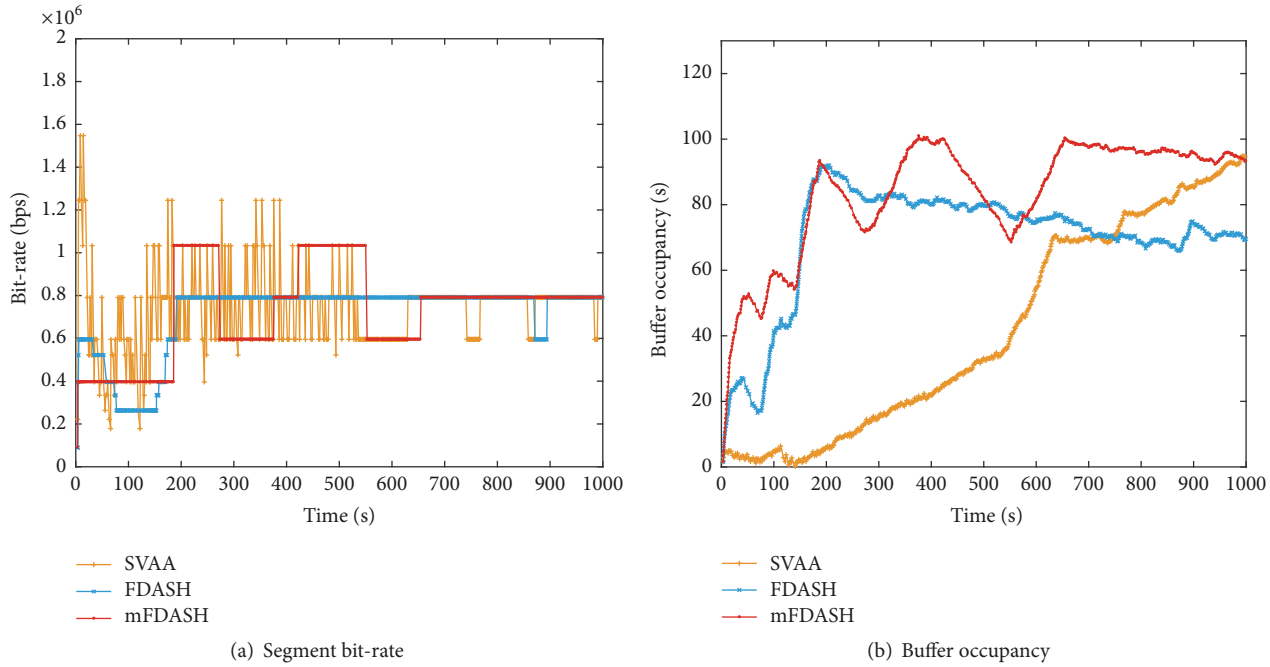
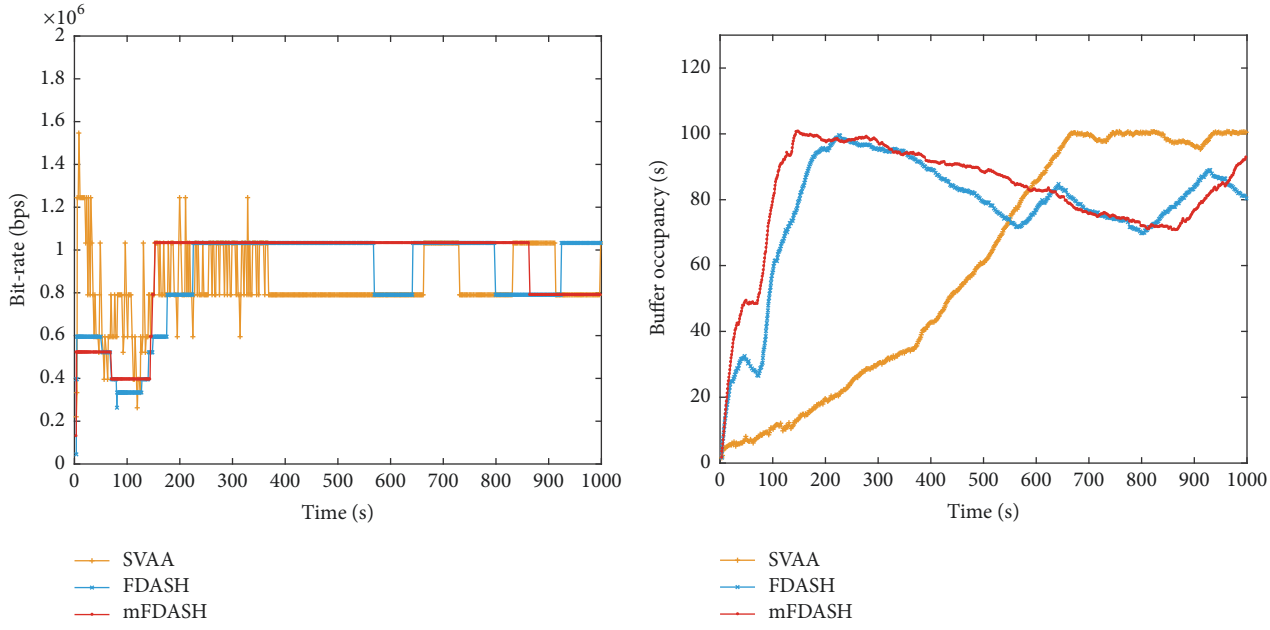


Figure 16: Simulation result in LTE network with a mobile DASH client and five background traffic events.

Table 7: Performance in LTE network with three mobile DASH clients and five background traffic events.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
Average segment bit-rate (m_{avq})	0.843 Mbps	0.815 Mbps	0.873 Mbps
Number of bit-rate changes (Num_{change})	122	15	7
Total number of interruptions (Int)	0	0	0



(a) Segment bit-rate (b) Buffer occupancy

Figure 17: Simulation result in LTE network with three mobile DASH clients and five background traffic events.

Table 8: QoE model configuration.

QoE model	Bitrate utility ($q(R)$)	μ
QoE_{hd}	[0.045 → 1, 0.089 → 1, 0.131 → 1, 0.178 → 1, 0.221 → 1, 0.263 → 1, 0.334 → 1, 0.396 → 1, 0.522 → 1, 0.595 → 1, 0.791 → 2, 1.033 → 5, 1.245 → 7, 1.547 → 10, 2.134 → 13, 2.484 → 14, 3.079 → 15, 3.527 → 17, 3.84 → 18, 4.22 → 20]	8

5.7. Quality of Experience Evaluation. In this paper, we use a QoE model proposed in [21]. The equation of the QoE model is shown as follows:

$$\mathrm{QoE} = \sum_{n=1}^{N} q(R_n) - \mu \times \mathrm{IntTime} - \sum_{n=1}^{N-1} \left(q(R_{n+1}) - q(R_n)\right), \tag{7}$$

where $q(\cdot)$ is the bit-rate utility function, R_n is the bit-rate of nth downloaded segments in Mbps unit, N is the number of segments downloaded, and IntTime is the total rebuffering time, respectively. The coefficient μ decides how much rebuffering time has influence on QoE and is set as Table 8 [21]. As you can see in (7), $\sum_{n=1}^{N} q(R_n)$ considers the *total bit-rate of downloaded segments*, Int considers the *rebuffering time occurred*, and $\sum_{n=1}^{N-1}(q(R_{n+1}) - q(R_n))$ considers the *number of video quality switching* and the *variance of video quality*, respectively.

The QoE model puts emphasis on High Definition (HD) video by scoring each bit-rate by referring to [22]. According to [22], video bit-rate range of 720 p (which is the start point of HD) is 1500–4000 kbps. By referring this, we scored each of the bit-rate as Table 8.

Figure 18 shows the QoE values calculated with QoE_{hd}. The QoE value in Wi-Fi network is the average value of ten simulation results. Also, the QoE values are normalized with the largest QoE value in each network environment. First of all, SVAA shows the highest QoE value in (b). Also,

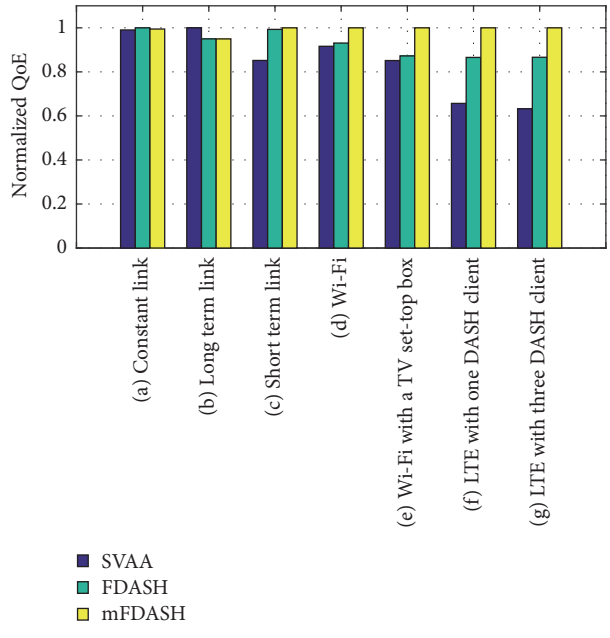


Figure 18: Normalized QoE of DASH Clients.

Table 9: Operation time of SVAA, FDASH, and mFDASH scheme.

Adaptive streaming scheme	SVAA	FDASH	mFDASH
t_O	2.71×10^{-6} s	2.84×10^{-6} s	3.24×10^{-6} s

SVAA shows quite similar QoE value in (a). However, SVAA shows the lowest QoE value in other networks. This means that SVAA scheme is only suitable in the networks with a few or zero bandwidth changes, which are not common in real world. In the case of mFDASH and FDASH, mFDASH shows almost the same QoE value compared to FDASH in (a) and (b). However, as the network has become more and more confusing (i.e., more realistic networks such as link network with short term periodic changes, Wi-Fi, and LTE), the QoE gap between mFDASH and FDASH increases. In (c), mFDASH shows 0.7% higher QoE value than FDASH while showing 7.4% higher QoE value in (d). Also, mFDASH shows 14.6% higher performance in (e), 15.5% higher performance in (f), and 15.4% higher QoE value in (g).

5.8. Complexity of Proposed Scheme. As the proposed method is a modified version of the existing method (FDASH), it is important to evaluate the complexity of the proposed method. Table 9 represents time taken by each rate adaptation scheme. The operation time is calculated as follows:

$$t_O = \frac{\sum_{k=1}^{K} t_{\text{start},k} - t_{\text{end},k}}{K} \quad (8)$$

where t_O, $t_{\text{start},k}$, $t_{\text{end},k}$, and K denote the average operation time for one segment, the time when the DASH client starts to calculate kth segment's bit-rate ($v(k)$), the time when the DASH client finishes the calculation of $v(k)$, and the total number of segments that are fetched by a DASH client, respectively. According to Table 9, mFDASH takes longer time than FDASH while SVAA takes the shortest time. Whenever more DASH clients try to fetch optimal video segment, they should conduct more complex rate adaptation scheme as a trade-off relationship. In other words, the longest operation time in mFDASH is acceptable since it provides the highest QoE to DASH clients.

6. Conclusions

This paper proposes a modified FDASH scheme for dynamic HTTP streaming. Our scheme modifies FLC of FDASH to make it choose more appropriate segment bit-rates and applies SBFM to reduce the number of bit-rate changes. Also, the HBTTE method is used to estimate throughput more accurately and to let the DASH clients choose optimum segment bit-rate, and the Start Mechanism is applied to rapidly switch to the higher bit-rate in the very beginning of a video. Lastly, the Sleeping Mechanism is applied to prevent buffer overflow. By choosing appropriate values for the factors described in Section 5, we maximize the performance of the average segment bit-rate and minimize the number of video rate changes in mFDASH. Moreover, we validate our scheme in various networks such as Point-to-Point, Wi-Fi, and LTE networks by using NS-3. The obtained results show that compared to FDASH and SVAA, the proposed scheme provides the highest QoE to a DASH client.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by ICT R&D program of MSIP/IITP [R0101-16-0189, Development of the Next Generation Convergence Broadcasting and Monitoring Systems combined with the Networks].

References

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper," 2017. [Online].

[2] I. Sodagar, "The MPEG-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[3] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceedings of the Proceeding of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, pp. 133–144, New York, NY, USA, February 2011.

[4] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic Theory and Applications*, Prentice Hall, New Jersey, NJ, USA, 1995.

[5] L. A. Zadeh, "Fuzzy logic," *The Computer Journal*, vol. 21, no. 4, pp. 83–93, 1988.

[6] D. J. Vergados, A. Michalas, A. Sgora, D. D. Vergados, and P. Chatzimisios, "FDASH: A fuzzy-based MPEG/DASH adaptation algorithm," *IEEE Systems Journal*, vol. 10, no. 2, pp. 859–868, 2016.

[7] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH system," in *Proceedings of the 3rd ACM Multimedia Systems Conference, MMSys'12*, pp. 11–22, USA, February 2012.

[8] SG12 and I. T. U. T, "Definition of quality of experience," in *TD 109rev2 (PLEN/12)*, pp. 16–26, Geneva, Switzerland, 2007.

[9] Y. Liu, Y. Shen, Y. Mao, J. Liu, Q. Lin, and D. Yang, "A study on Quality of Experience for adaptive streaming service," in *Proceedings of the 2013 IEEE International Conference on Communications Workshops, ICC 2013*, pp. 682–686, Hungary, June 2013.

[10] H. R. Berenji, "Fuzzy logic controllers," in *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, pp. 69–96, Springer, USA, 1992.

[11] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part I," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404–418, 1990.

[12] *The network simulator - ns-3*, 2017, http://www.nsnam.org/.

[13] C. Zhou, C.-W. Lin, and Z. Guo, "mDASH: A Markov decision-based rate adaptation approach for dynamic HTTP streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 738–751, 2016.

[14] D. L. Isaacson and W. M. Richard, *Markov Chains, Theory and Applications*, vol. 4, John Wiley & Sons, New York, NY, USA, 1976.

[15] R. M. Blumenthal and R. K. Getoor, *Markov processes and potential theory*, Pure and Applied Mathematics, Vol. 29, Courier Corporation, 2007.

[16] M. Zhao, X. Gong, J. Liang, W. Wang, X. Que, and S. Cheng, "Scheduling and resource allocation for wireless dynamic adaptive streaming of scalable videos over HTTP," in *Proceedings of the 2014 1st IEEE International Conference on Communications, ICC 2014*, pp. 1681–1686, Australia, June 2014.

[17] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[18] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proceedings of the 8th ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '12)*, pp. 109–120, ACM, December 2012.

[19] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, 2005.

[20] Z. Bingül and O. Karahan, "A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control," *Expert Systems with Applications*, vol. 38, no. 1, pp. 1017–1031, 2011.

[21] H. mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensive," in *Proceedings of the 2017 ACM SIGCOMM Conference*, Los Angeles, CA, USA, 2017.

[22] YouTube live encoder settings, bitrates and resolutions. http://support.google.com/youtube/answer/2853702?hl=en.