A Proposed Adaptive Rate Algorithm to Administrate the Video Buffer Occupancy for Smooth Video Streaming

Saba Qasim Jabbar[1,2](✉), Dheyaa Jasim Kadhim[3], and Yu Li[1]

[1] Wuhan National Laboratory for Optoelectronics, Division of Communication and Intelligent Networks, School of Electronics and Information, Huazhong University of Science and Technology, Wuhan 430074, Hubei, People's Republic of China
shura2007515@yahoo.com

[2] Computer Engineering Department, University of Baghdad, Baghdad, Iraq

[3] Electrical Engineering Department, University of Baghdad, Baghdad, Iraq

Abstract. Video streaming has become an irreplaceable technology in modern smart devices especially with the appearance of dynamic streaming over HTTP (DASH). However, the fluctuating and instability of wireless networks is still one of the greatest challenges to video streaming process. Video quality deteriorates by the amount of time that video streaming process has stopped playing because of the buffer starvation state. In this work, an adaptive rate algorithm is proposed for getting smooth video playback continuity by monitoring the available bandwidth and administrating the video buffer occupancy. Most of the conventional algorithms try to fill the video buffer quickly with available video segments which it may degrade the quality of experience (QoE) since these video segments may have low bitrate versions. So our proposed algorithm administrates the video buffer occupancy and maintains its filling with segments of high quality versions by estimating the available throughput and measuring the video buffer level. The estimation of the available throughput for the next segments is done based on the previously observed throughput of the last downloaded segment that keeps the buffer at accepted level. Through the simulation results, we checked effectiveness and robustness of our proposed algorithm under three different network bandwidth conditions namely: step-down, long term fluctuation and sudden-drop conditions. Simulation results show that the proposed algorithm has fast reaction to these cases through adapting the video bitrate accordingly and maintain the video buffer away from the risk of underflow. Also we compared the proposed algorithm with other conventional algorithms in terms of average video bitrates, number of video bitrate switches and number of playback interruptions. We found that our proposed algorithm outperforms these algorithms in achieving high video bitrates, low number of video bitrate switches and minimum number of playback interruptions.

Keywords: Video streaming · Video buffer · Quality of experience (QoE) Adaptive rate algorithm

1 Introduction

High speed broadband networks with enhancements in display technology of different devices (e.g., smart phone, laptop and personal media player) have enabled video streaming to achieve the most popular services in mobile applications. Video streaming controls Internet traffic over all network types in the world [1]. Recently, a new type of video transfer techniques have been presented to transfer video via different channels such as wireless network which is called adaptive streaming [2]. This technique is used widely for multimedia streaming service, adopted by Microsoft smooth streaming, adobe dynamic streaming, Netflix and Apple HTTP Live Streaming, etc. Also, 3GPP and MPEG encouraged standardization of adaptive streaming named dynamic adaptive streaming over HTTP (DASH) [3]. In DASH system media content is split into a series of small segments, each segment is encoded in several versions with different bit rates and qualities then streamed to user's device according to its requests. For each media streaming, the server generates media presentation description file (MPD). Before playing the video, DASH user will fetch the MPD file from the server to have the information about the media types, resolution, encoded alternatives of the media content, segment sizes, etc. At user side, a network condition will be monitored by DASH user and dynamically a suitable version for coming segment to download will be selected through the rate adaptation algorithm. Under good network conditions, the DASH user will ask the server to present a good quality version and when the network condition is bad; user's device will detect the network change and request a low quality version. At DASH server side, each segment is offered with a different bit rates and each video rate is linked with a group of encoding factors such as frame rate, resolution.

The adaptation rate algorithm is major in DASH service, since unsuitable bit-rate may result in bad video quality or playout interruption, further degrade the QoE. The target of adaptation rate algorithm is to rise the video quality through meeting conflicting objectives in a way enhancing the user's viewing experience, such as choosing a group of video bit rates which are the highest feasible, a voiding unnecessary bitrate switches and keeping the buffer content to avoid interruption of playback [4, 5]. To improve the user experience, the adaptive algorithms mainly determine the video rates based on existing bandwidth and the state of the playback buffer, one approach to pick a video rate relied on estimated throughput as in [6–8]. The adaptation rate algorithms of two other business solutions, namely Microsoft's seamless streaming and Netflix with OSMF are assessed in [9], and the experimental outputs decided that none of them gives smoother quality adaptation with the fixed playback buffer. However, in a very changing environment, accurate estimation of throughput becomes a challenge.

Many adaptation algorithms consider the video buffer and estimated throughput to identify the video rates as in [10, 11]. Rahman [12] proposed an algorithm to determine video bitrates based on estimated throughput and buffer occupancy where buffer levels are dynamically determined relied on the sizes of upcoming segments. In [13, 14], models have been suggested to study the video temporal quality, however they do not include change of bitrate during streaming cycle. In [15], the authors have studied the impact of the bitrate variation on QoE, presented good notes about how the alteration of frequency and amplitude will effect on user experience. In [16], an adaptation algorithm for video streaming over HTTP is proposed. This algorithm consists from two phases: fast phase for increasing the buffer level to predefined threshold value and steady phase for controlling the buffer level from going back to underflow state, so the algorithm could limit the number of video quality switches. Several studies have been performed to understand the impact of QoE on users' participation. The video playing time and user retention have been found based on QoE. Improving video quality depends on the number of bitrate switches and interrupting events. In this paper, we try to enhance the quality of experience through maintaining the interruption and switching rate at low level. An adaptive rate algorithm is suggested which determines the video bitrates based on both the estimated throughput and the content of the buffer. The proposed algorithm administrates the buffer occupancy to maintain the buffer fill with high quality segments without the situation of an empty buffer.

The work of this paper is organized as follows: Sect. 2 describes the main principles of our system design including two main units: throughput measurement unit and video buffer measurement unit. Section 3 gives a detailed description of our proposed adaptive rate algorithm, while Sect. 4 shows the simulation results that will be drawn from our experiments to describe our proposed algorithm as compared to other algorithms. Finally, Sect. 5 will give the main conclusions are gained with this work.

2 System Design

The HTTP user downloads a video streams and splits them into multiple segments. All the segments have an equivalent length of τ sec. Then, the video segments are saved in multiple separate representations at the HTTP server. The set of representations available for the video streaming is denoted by V_r where $V_r = \{V_{r1}, V_{r2}, \ldots\ldots, V_{rm}\}$. The user dynamically picks a video quality from the set $\mathrm{V_r}$ for the upcoming segment. V_{rmin} and V_{rmax} are the representations with the lowest and highest video rates in the set V_r. The video rates which they are higher and lower than the current video rate are denoted by $V_r\uparrow$ and $V_r\downarrow$ respectively. The proposed video streaming structure of the HTTP adaptation system is described in Fig. 1.

In this system, server content is represented by a media presentation description (MPD) file with group of video segments of *m* various representation qualities. The user asks the MPD from the server, and the received MPD file is interpreted by a parser in the user. The mentioned file is used for requesting video segments from the streaming server. The suggested system for the HTTP dynamic adaptive at user aspect consists of two units namely: throughput measurement (TM) unit and video buffer

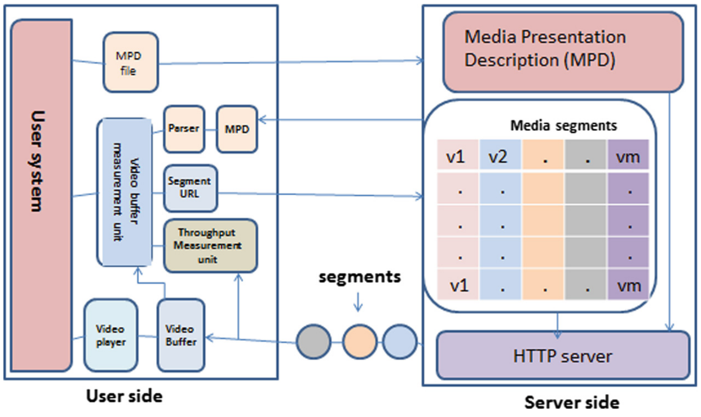

Fig. 1. Dynamic adaptive video streaming system

measurement unit (VBM). Video segments are requested by the HTTP user relied on the decision of the video buffer measurement unit (VBM). This unit dynamically selects video rate for each segment. The HTTP user sends HTTP GET requests to the HTTP server to download the segments. The HTTP server transmits the video to the user through the bottleneck link. The downloaded segments are saved in the video buffer, then sending them to the video player. The throughput measurement unit (TM) estimates the throughput for next segment, and then feeds the measuring throughput to video buffer measurement unit (VBM) for selecting the next video rate before downloading the segment.

In this work, the segments are downloaded using the serial segment fetch method. The data of τ seconds is added to the buffer once the current segment is fully downloaded and the user starts playing the video. The purpose behind the proposed algorithm is to maintain the video buffer filled with high quality versions, and to control the video buffer level when there is a throughput fluctuation. The algorithm is applied immediately after downloading $(n - 1)^{th}$ segment in order to select the suitable bitrate for next segment. Before bring the upcoming segment from the streaming server, the current network condition is estimated by throughput measurement unit (TM) which it will be discussed in details with video buffer measurement unit (VBM) in the following subsections.

A. Throughput Measurement Unit (TM)

Estimating available bandwidth considers a significant role in choosing video rate. The user is able to estimate the upcoming segments throughput based on the previous throughput notifications. So choosing the video rate for the upcoming segment is relied on the throughput T_{n-1} of the last downloaded segment that keeps the buffer stability even some oscillates in the video rate curve. As the bandwidth of the network fluctuates over time, the estimated throughput $\hat{T}_{n-1}$ is smoothed by a weighted transferring average scheme as:

$$\hat{\mathrm{T}}_{\mathrm{n}} = \rho \mathrm{T}_{\mathrm{n}-1} + (1 - \rho)\hat{\mathrm{T}}_{\mathrm{n}-1} \tag{1}$$

where ρ is a weighing factor to explain the changing in network conditions. $\hat{T}_n$ is the estimated throughput and T_n is the available throughput. A large value of ρ means the network conditions has changed abruptly and so the upcoming bandwidth is calculated by giving more weight to the last available throughput. Contrarily, bandwidth is calculated by giving more weight to the last estimated throughput. The value of ρ is updated dynamically relating to the available wireless link conditions and the throughput is estimated accordingly. Before requesting the next segment, the video buffer measurement unit will select the suitable bitrate for download the segment according to the current buffer status and the estimated throughput value which is fed from the throughput measurement unit.

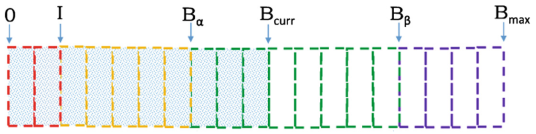

Fig. 2. Video buffer

B. Video Buffer Measurement Unit (VBM)

This unit is responsible for checking the status of video buffer and then adjusting the downloaded throughput for the upcoming segment to keep the level of the buffer close to the targeted level. Initially we define three threshold values in the level of the buffer I, B_α and B_β as shown in Fig. 2 below, which they are measured in seconds with $0 < I < B_\alpha < B_\beta < B_{max}$ where B_{max} is the maximum buffer level.

Here, B_α and B_β are the operating thresholds for the video buffer while B_{targ} is regarded as the mid-point between the running thresholds, which it is given by:

$$\mathrm{B}_{\mathrm{targ}} = \frac{\mathrm{B}_\beta + \mathrm{B}_\alpha}{2} \tag{2}$$

If the current buffer level B_{curr} is equal to target level B_{targ} and *(B_{curr}-B_{last})* near zero then $\hat{T}_n$ is not adjusted, B_{last} is the last buffer level. Otherwise, an adjusting factor λ_n based on buffer occupancy is considered to put B_{curr} near B_{targ} and *($B_{curr} - B_{last}$)* near zero as it is cleared below:

$$\lambda_n = (1 + \lambda_1)(1 + \lambda_2) \tag{3}$$

where λ_1 is deals with the space between buffer current state and the target state i.e.: how close the buffer current state B_{curr} to the target state B_{targ} and it is calculated as shown below:

$$\lambda_1 = \frac{\mathrm{B_{curr}} - \mathrm{B_{targ}}}{\mathrm{B_{targ}}} \tag{4}$$

While λ_2 is calculated as follows:

$$\lambda_2 = \frac{\mathrm{B_{curr}} - \mathrm{B_{last}}}{\mathrm{B_{max}}} \tag{5}$$

After estimating λ_n, $\hat{T}_n$ is adjusted as:

$$\tilde{\mathrm{T}}_\mathrm{n} = \lambda_\mathrm{n} \cdot \hat{\mathrm{T}}_\mathrm{n} \tag{6}$$

3 Proposed Adaptive Rate Algorithm

In this section, we will give a detailed description of our proposed adaptive rate algorithm. This proposed algorithm is used for: (1) maintaining the video buffer filled with high-quality segments; (2) decreasing the interruption duration especially when there is a noted degradation in system throughput and (3) protecting the buffer from return to underflow level. The proposed algorithm is applied immediately after downloading the (n − 1)th segment. Then the algorithm specifies the video rate to download the next segment. The video streaming session is divided into two stages of the process: the initiation stage and normal stage. The purpose of the initiation stage is to prevent the user from giving away the video session since it starts before the buffer being empty. During the initiation stage, the user loads a little information that may be utilized to determine the video rate. Therefore, the conservative approach is initially considered, and since the content of the buffer gradually rises, the risk is taken in the selection of video rate. The algorithm sets the minimum available video bitrate V_{rmin} for the first downloaded segment. The reason behind fixing the minimum video rate for the first segment is to reduce the delay after the user demand video and before operating the video. Additionally, the user does not have any notification about the case of the network at the starting of the video streaming session.

For $B_{curr} < I$ where I is a threshold value equals to 2τ, the user changes to high video rate if $V_{r\uparrow} < \delta_1 \times T_{n-1}$. Otherwise the user changes to a higher video rate if $V_{r\uparrow} < \delta_2 \times T_{n-1}$ where δ_1 and δ_2 are two threshold margins and $(\delta_1 < \delta_2)$. The proposed algorithm continues to employ the initiate stage until a reduction in the level of the buffer is monitored. When the user monitors a reduction in the buffer level, it switches to a normal stage, four cases are considered through this stage:

1. The user chooses V_{rmin} as the next bitrate.
2. The user chooses the bitrate of the next segment to be less than the current segment bitrate.
3. The user chooses the bitrate of the next segment to be more than the current segment bitrate.
4. The user chooses the bitrate of the next segment to be the same as the current segment bitrate.

To choose the bitrate of next segment from the V_r set, the following conditions must be met:

a. If $B_{curr} < B_\alpha$ Then $V_{r^{next}} = V_{rmin}$
b. If $(B_n - B_{curr}) \geq 0$ Then $V_{r^{next}} = V_r(y)$

If the buffer content falls under B_α, V_{rmin} is always chosen. The reason is that it is essential matter to avoid the interruption of operation. When the buffer content lower than B_α, there is a significant risk of buffer underflow in the existence of throughput volatility. If the buffer content is above B_α, the user selects the highest video rate less than the adjusted throughput. After that, when an increasing in the network throughput is noticed, the bitrate of next segment increases as a response. So two conditions should be met:

a. $V_{r\uparrow} < \tilde{T}_n$
b. $(B_n - B_{curr}) \geq 0$

The first condition ensures that the chosen video bitrate is low than the adjusted throughput to avoid draining the buffer. A reduction in the content of the buffer when $V_{r\uparrow} < \tilde{T}_n$ suggests that the throughput was overestimated. The reason beyond the second condition is that it reduces the frequency of the video rate switching by not increasing the video rate when the user monitors a drop in the buffer content, to prevent the risk of a likely step down in the near future. If a drop in the adjusted throughput below the selected bitrate then the video rate is reduced until the following condition is satisfied $V_{r\uparrow} < \tilde{T}_n$.

The following steps show the procedure of implementing proposed algorithm:

```
Definitions :
$V_r(y) = \{V_{r1}, V_{r2}, ......., V_{rm}\}$
$I, B_\alpha, B_\beta, B_{max}$ : Buffer threshold values
$V_{r_{next}}$ : video rate chosen for next segment
$V_{r_{curr}}$ : current video rate of recent download segment
$V_{r\uparrow}$ : next higher video rate
$B_{curr}$ : current buffer status at last downloaded segment
$B_n$ : buffer status during the download of $(n)^{th}$ segment
$\tilde{T}_n$ : adjusted Throughput of $\hat{T}_n$
$\delta_1, \delta_2$ : threshold values $(\delta_1 < \delta_2)$
-------------------------------
Input : $T_{n-1}, \tilde{T}_n, B_{curr}$
Output : $V_{r_{next}}$
If the initiate stage == true then
  If $B_{curr} < I$ then
    If $V_{r\uparrow} < \delta_1 \times T_{n-1}$ then $V_{r_{next}} = V_{r\uparrow}$
  Else
  If $V_{r\uparrow} < \delta_2 \times T_{n-1}$ then $V_{r_{next}} = V_{r\uparrow}$
  End If
Else If the normal stage == true then
   If $B_{curr} < B_\alpha$ then
      $V_{r_{next}} = V_{r\,min}$
     Else if $V_{r\uparrow} < \tilde{T}_n$ then
     For y = 0 to length of $V_r(y)$
           If $V_r(y) > \tilde{T}_n$ then
           Break
           End If
            y++
          End for
    If $(B_n - B_{curr}) \geq 0$ then
       $V_{r_{next}} = V_r(y)$
    Else
      $V_{r_{next}} = V_{r_{curr}}$
  End If
```

```
Else If $V_{r_{curr}} > \tilde{T}_n$ then
        $V_{r_{next}} = \max\left\{V_r(y) : V_r(y) < \tilde{T}_n\right\}$
 Else
        $V_{r_{next}} = V_{r_{curr}}$
  End If
End If
---------------------------------
```

4 Simulation Results

In this section, we evaluate the performance of our proposed algorithm and compare its performance with other related works such as QDASH [10] and AAA [16]. Through our simulation, we consider ten representation levels which it is provided by HTTP server for adapting video bitrates as: (300, 500, 700, 1000, 1200, 1400, 1600, 1800, 2000, and 2500) kbps. The video buffer length is 60 s and the segment length is 4 s while the two margins δ_1 and δ_2 are set to 0.5 and 0.75 respectively. I, B_α, ρ and B_β are set to 2τ, 5, 0.9 and 12 respectively. In this simulation, we consider three types of bandwidth conditions: (1) step-down condition; (2) fluctuation condition and (3) sudden-drop in the available bandwidth condition.

Figure 3 shows the behavior of our proposed algorithm under the first condition of stepping down, so we consider the throughput is gradually decreasing and we can show below how our proposed algorithm is responding to this case in term of video bitrate and video buffer level respectively. In this figure, the proposed algorithm drops the video bitrate gradually at time 28 s when the throughput decreases. Video buffer level is below B_α so the algorithm decreases the video bitrate to V_{rmin} for avoiding any playback interruption. This feature of the proposed algorithm can offer sustainable QoE in video streaming applications. Figure 4 shows the behavior of our proposed algorithm under the network bandwidth large fluctuation condition, and this figure shows

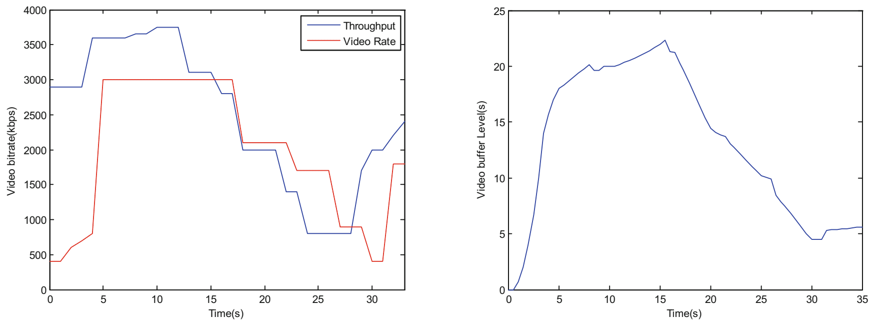

Fig. 3. System throughput and video buffer under step-down condition

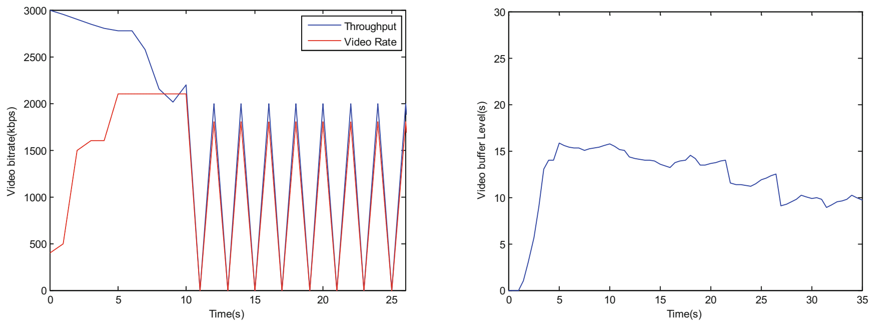

Fig. 4. System throughput and video buffer under fluctuation condition

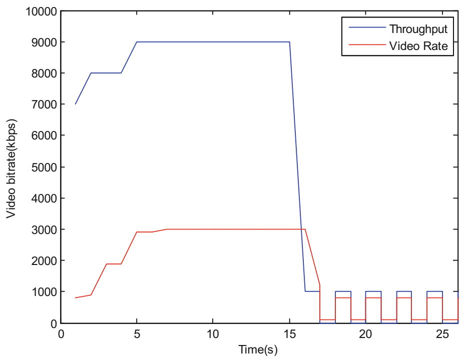

Fig. 5. System throughput vs. video rate under sudden-drop condition

how our proposed algorithm is quickly responded to the bandwidth fluctuation which is one of the main features of the adaptive rate algorithm. From the figure, when the bandwidth largely increases the proposed algorithm adapts the video bitrate by stepping up quickly. On the other hand this conservative way can improve QoE for video by allowing more segments be accumulated in video buffer so there is no interruption in video playback. Otherwise, Fig. 5 shows one of the important features of the adaptive rate algorithm under a sudden-drop condition in bandwidth. We can show how the proposed algorithm acts quickly by adapting the video bitrate to large drop in bandwidth. In this way the algorithm can avoid the risk of playback interruption and keep the QoE in acceptable level.

Figure 6 shows the comparison among different adaptive rate algorithms according to the number of playback interruptions. QDASH faces three interruptions while our proposed algorithm and AAA algorithm face one interruption. The reason is that QDASH depends on download time in adapting the video bitrate and since most of video streaming providers encode their videos in VBR (Variable Bit Rate), so each segment has a different size which means different segments would have different download times and it could not estimate the available bandwidth in accurate way. On the other hand the proposed algorithm and AAA algorithm can have less number of

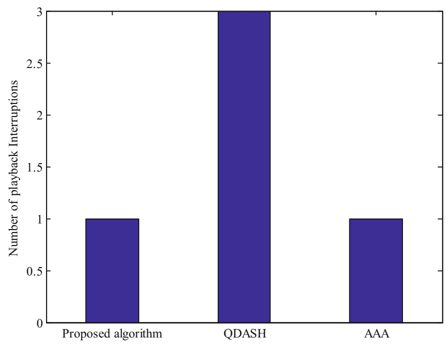

Fig. 6. Comparison of the number of playback interruptions among different adaptive rate algorithms

interruptions due to their accurate way in estimating the available throughput and administrate the video buffer. While Fig. 7 shows the comparison among different adaptive rate algorithms in term of the average video bitrates. From this figure, the proposed algorithm outperforms QDASH and AAA algorithms by 300 kbps and 200 kbps respectively due to the ability of the proposed algorithm in administrating the video buffer occupancy and keep it fill with good quality segments. The main work of the proposed algorithm is appending the video buffer with segments of high quality version (i.e. high video bitrate) and drops the downloaded segments with low quality version (i.e. low video bitrate). While AAA algorithm waits the video buffer level reaches a threshold value then stepping up or down the video bitrate for next segments.

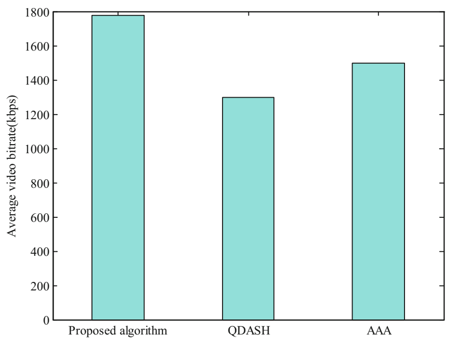

Fig. 7. Comparison of average video bitrate at different adaptive rate algorithms

Finally, Fig. 8 shows the comparison among different adaptive rate algorithms in term of the average number of video bitrates switches. From this figure the proposed algorithm outperforms the QDASH algorithm in minimizing the number of video bitrate switches because the estimated way that is used by our proposed algorithm for throughput could overcome the fluctuating video bitrate curve. Also appending the video buffer with high quality segments and dropping the low quality segments could reduce the number of video bitrate switches. For example if there is four video encoded versions: v1, v2, v3 and v4 where v1 < v2 < v3 < v4 in conventional algorithm if the available throughput decreases the algorithm would reduce the video bitrate to v3 followed by v2 and when the throughput condition is enhanced the algorithm rises the video bitrate to v4. On the other side the proposed algorithm appends the video buffer with high quality segments and drops the low quality segments (i.e. appends with v4 followed by v3) unlike the conventional algorithm it doesn't reduce the video bitrate to v2 so minimizing the number of switches.

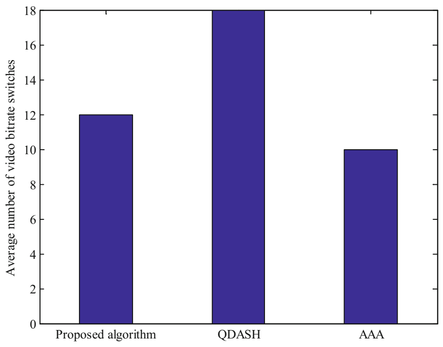

Fig. 8. Comparison of average number of video bitrate switches among

5 Conclusions

In this work, an adaptive rate algorithm is proposed to enhance the video quality through video streaming by monitoring the network conditions and administrating the video buffer occupancy. These network conditions include throughput stepping down gradually, long term throughput fluctuation and suddenly dropping of network throughput. Through the simulation results, we checked the effectiveness of the algorithm under these conditions. Our simulation results show that the proposed algorithm has a quick response to these cases through adapting the video bitrate accordingly and maintains the video buffer away from the risk of underflow. Also we compared the proposed algorithm with other algorithms suck as QDASH [10] and AAA [16] in terms of average video bitrates, number of video bitrate switches and number of playback interruptions. We found the proposed algorithm outperforms these algorithms in achieving high video bitrates and low number of video bitrate switches. This due to the main work of the algorithm based on enhancing the video quality through administrates the video buffer and maintains it fill with segments of high quality version while the conventional algorithms tend to download the segments quickly that fill the video buffer with segments of low quality version.

References

1. Petrangeli, S., Famaey, J., Claeys, M., Latré, S., De Turck, F.: QoE-driven rate adaptation heuristic for fair adaptive video streaming. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) **12**(2), 28 (2016)
2. Hossfeld, T., Seufert, M., Sieber, C., Zinner, T., Tran-Gia, P.: Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies. Comput. Netw. **81**, 320–332 (2015)
3. Stoekhammer, T.: Dynamic adaptive streaming over HTTP-design principles and standards. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems, New York, USA, pp. 2–4 (2014)
4. Dobrian, F., Sekar, V., Awan, A., Stoica, I., Joseph, D., Ganjam, A., Zhan, J., Zhang, H.: Understanding the impact of video quality on user engagement. ACM SIGCOMM Comput. Commun. Rev. **41**(4), 362–373 (2011)
5. 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, pp. 463–472 (2011)
6. Jiang, J., Sekar, V., Zhang, H.: Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, pp. 97–108 (2012)
7. Thang, T.C., Ho, Q.D., Kang, J.W., Pham, A.T.: Adaptive streaming of audiovisual content using MPEG DASH. IEEE Trans. Consum. Electron. **58**(1) (2012)
8. Liu, C., Bouazizi, I., Gabbouj, M.: Rate adaptation for adaptive HTTP streaming. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems, pp. 169–174 (2011)
9. Akhshabi, S., Begen, A.C., Dovrolis, C.: An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In: Proceedings of the Second Annual ACM Conference on Multimedia Systems, pp. 157–168 (2011)
10. Mok, R.K., Luo, X., Chan, E.W., Chang, R.K.: QDASH: a QoE-aware DASH system. In: Proceedings of the 3rd Multimedia Systems Conference, pp. 11–22 (2012)
11. Suh, D., Jang, I., Pack, S.: QoE-enhanced adaptation algorithm over DASH for multimedia streaming. In: International Conference on Information Networking (ICOIN), pp. 497–501 (2014)
12. ur Rahman, W., Chung, K.: Buffer-based adaptive bitrate algorithm for streaming over HTTP. KSII Trans. Internet Inf. Syst. (TIIS) **9**(11), 4585–4603 (2015)
13. Mok, R.K., Chan, E.W., Chang, R.K.: Measuring the quality of experience of HTTP video streaming. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 485–492 (2011)
14. Singh, K.D., Hadjadj-Aoul, Y., Rubino, G.: Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC. In: Consumer Communications and Networking Conference (CCNC), pp. 127–131 (2012)
15. Ni, P., Eg, R., Eichhorn, A., Griwodz, C., Halvorsen, P.: Spatial flicker effect in video scaling. In: Third International Workshop on Quality of Multimedia Experience (QoMEX), pp. 55–60 (2011)
16. Miller, K., Quacchio, E., Gennari, G., Wolisz, A.: Adaptation algorithm for adaptive streaming over HTTP. In: 19th International Workshop on Packet Video Workshop (PV), pp. 173–178 (2012)