Improving Video Quality in DASH Systems by Proposing Adaptive Bitrate Scheme based on Variable Segment Size Approach

Saba Qasim Jabbar
Huazhong University of Science and Technology
China, Wuhan

Dheyaa Jasim Kadhim
University of Baghdad, Electrical Engineering Department
Iraq, Baghdad

Yu Li
Huazhong University of Science and Technology
China, Wuhan

ABSTRACT

Dynamic adaptive streaming via HTTP (DASH) has been widely disseminated over the Internet especially under the circumstances of the time varying network, which it is still the biggest challenge for providing smoothly video streaming with high quality. In DASH system, after the downloading process of a segment is completed, the player estimates the available network bandwidth by calculating the download throughput and adapting the video bitrate level based on its estimations. At client side, the DASH player uses an Adaptive Bitrate Algorithm (ABR) to choose the suitable bitrate for next segment based on current conditions. However, these adaptive algorithms discard the fact that the segment sizes greatly vary for a given video bitrate. Hence they may fail to predict the time needed for downloading the next segment. In this paper, an adaptive bitrate algorithm is proposed based on the video segment size as well as the network bandwidth estimation and the current buffer occupancy in order to accurately predict the time needed to download the next segment. Simulation results show that the proposed scheme is able to predict the download time. Also we compared the proposed scheme with other conventional schemes, we found that our proposed scheme outperforms others in achieving high video bitrates, low number of video bitrate switches, minimize the other QoE metrics such as video convergence time and the number of bitrate switching events.

General Terms

Video Streaming, Video Quality, Network Bandwidth.

Keywords

Video streaming, video buffer, adaptive bitrate algorithm, DASH, ABR.

1. INTRODUCTION

World Internet video traffic has been increasing rapidly with the advent of popular video streaming services. According to the visual networking reports, world video traffic accounted for 70% of all consumer Internet traffic in 2015, and nearly a million minutes of video contents will cross IP networks every second in 2020 [1]. Many commercial video providers have utilized adaptive bitrate streaming techniques to provide streaming media with the best quality for users. Recently, dynamic adaptive streaming via HTTP (DASH) has become a famous technology, due to its implementation and deployment simplicity [2].

In contrast to conventional RTP/UDP based video streaming, HAS streams the video over the traditional protocol stack HTTP/TCP, which is used for web traffic. Many video streaming technologies rely on HTTP-based adaptive bitrate streaming such as Microsoft's Smooth Streaming, Apple's HTTP Live Streaming (HLS), and Adobe's HTTP Dynamic Streaming (HDS) also YouTube began utilizing the HAS approach as its default streaming method [3]. In this technique, a video content is encoded into multiple bitrates level and divided the encoded video into small parts with fixed playback duration. When the video session starts, the HAS player sends an HTTP GET message requesting for a video segments to the HTTP server. At server a Media Presentation Description (MPD) file is created for each video, which is an XML format file that lists all the representations, segment information, and segment URLs for each representation. At the client side, the DASH player first downloads and parses the MPD file to retrieve the media information. The DASH player can change the representation according to the platform, user preference, and network conditions. A HAS client typically uses an adaptive bitrate selection (ABR) algorithm to select the most suitable representation. The client usually begins with the lowest representation, and then based on one of the parameters measured during the download, estimates the best representation for each subsequent segment.

Usually, a DASH player has a playback buffer and schedules the next video segment request based on the occupancy of the video buffer. After downloading a sufficient amount of data, the player starts to play out the video from the buffer. Once the length of video buffer reaches the maximum level, the player switches to steady state where the player periodically requests the next segment to maintain a constant buffer length. Through the adaptation process, the player estimates the available bandwidth according to the network conditions. Since video segments are transmitted over TCP, TCP dynamics behaviour affect the bandwidth estimation process, such as slow-start and Additive Increase Multiplicative Decrease (AIMD) approach.

The existing bitrate adaptive algorithms for HTTP system are aiming to either achieve the efficiency of high bandwidth usage for video adaptation bitrate to match available bandwidth, or to maintain continuous video playback by homogeneity the video bitrate in order to avoid the issue of video buffer overflow/underflow such as bandwidth-based approaches discussed in [4][5] and buffer-based approaches discussed in [6][7]. While authors of [8] proposed a rate adaption algorithm which it is based on calculating the average segment downloaded rate, then switching up or down the bitrate with the network bandwidth in aggressive way to maintain the video quality in acceptable level. Authors in [9] proposed a stream switching algorithm based on encoding the raw video chain for multiple video formats with different qualities and bit rates, so that the network bandwidth fluctuations can change to other video formats to ensure video streaming continuity at client side.

In [10], an adaptation algorithm for adaptive streaming over HTTP (AAASH) is proposed for rate adaptation with multiple parameters and conditions. 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. In [11] and [12], the authors studied problems of bitrate adaptation and determined the causes of many unwanted interactions that arise as a result of modifying the video bitrate over HTTP.

Authors of [13] proposed TCP downloading throughput as input only when it is an accurate indicator of fair share bandwidth. This usually occurs when the number of subscribers and the periods of absence are exceeded. In the presence of interruptions, the proposed algorithm constantly achieves network bandwidth by increasing the transmission rate, and prepares to roll back once it experiences congestion. This algorithm dynamically selects the level of video that matches the user's available bandwidth; thus, this algorithm reduces processing costs since the video is encoded, no further processing is needed to adapt the video with variable bandwidth. While the disadvantage of this approach is the increased storage requirements and the fact that adaptation is characterized by a more rough division because video bitrates can only belong to a separate set of levels. While in [14], a Markov Decision Process (MDP) is used to deal with the stochastic decision problem, which reduces both the number of playback interruptions and the number of quality level switches while increases the quality of experience. In [15], the modified adaptive algorithm is designed to implement a combination of randomness for video requests, video bitrate selection, and average harmonic based on average.

In this paper, we introduce a scheme for improving video quality based on a video segments awareness in HTTP system, that not only uses the estimated network bandwidth and current buffer occupancy level but also uses the segment sizes to predict the time needed to download the next segment. This ensures that the best possible representation of the video is downloaded while avoiding video buffer underflow/overflow. The work of this algorithm depends on current video buffer level, video bitrate of previous segment and iterative throughput measurements, so that it can predict best video bitrate for next segment. Through simulation results, our approach outperforms the existing algorithms in measuring the fair share bandwidth, achieving fairness, buffer stability and reducing the number of video bitrates switching.

The work of this paper is organized as follows: section II describes the video streaming model that it be considered in this work. Section III gives a detailed description our proposed scheme which includes two main parts: throughput estimation and bitrate adaptation. Section IV shows the simulation results that will be drawn from our experiments to describe our proposed algorithm as compared to other algorithms. Finally, section V will give the main conclusions are gained with this work.

2. VIDEO STREAMING MODEL

Generally DASH clients play the media contents that transferred from streaming server via available network bandwidth after sending an HTTP GET request to the server. The Server-Client model for streaming system is shown in Figure 1 below, where the network bandwidth represents the data producing rate and video content's bitrate represents the data consuming rate to the video buffer of the client. At the beginning of each download stage, an adaptation algorithm at client side will select the suitable bitrate of the next segment to be downloaded. The rates of the different video representation levels are transferred to the client by the server in the Media Presentation Data (MPD) file. The encoding of a video segment is done into different versions of L, with different video bitrates $V_{Ri+1} \in V,\ V = \{V_{R1}, ..., V_{RL}\}$ and $V_{R1} < V_{R2} ... < V_{RL}$. Assuming at the video playing time, the current duration being play backed is fully loaded, the next duration's playback time is from the start time (t_s) to the end time (t_e). All video versions are divided into segments of equal length, each of which consumes the same playing time τ. Where τ is the inverse of the video frame rate F_r i.e., $\tau = 1/F_r$.

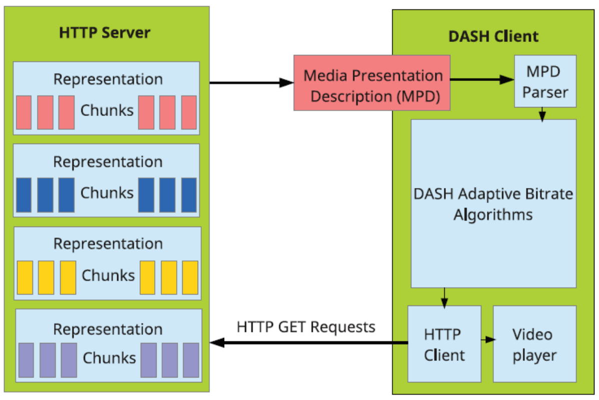


Fig 1: HTTP Server-Client model for streaming system

In conventional scheme, the download throughput for each segment is measured as:

$$T_i = \frac{\tau V_{Ri}}{t_e - t_s} \qquad (1)$$

where T and t_e-t_s is the throughput and the download duration for ith segment respectively. Then all downloaded segments are store in video buffer. According to expression (1), the throughput estimation is corrected under the assumption that video segments are constant bitrate (CBR) encoded not variable bitrate (VBR) encoded because the file size of each video segment is not identical under VBR coding, so the bandwidth estimation is affected by VBR properties.

In the other side, Video bitrates fluctuations occur more frequently when the content of video is a VBR encoding. Also, the dynamic of video buffer is impacted because the time required for download the next segment is determined by segment size. Figure 2 below shows the impact of VBR on video bitrates and buffer occupancy when the DASH player streams a video content coded via VBR. The HTTP server hosts a VBR video data-set that includes a video coded at ten bitrates ranging from 350 kbps to 7.5 Mbps with duration of 2s per segment. Most of these problems can be solved either by adjusting the bandwidth estimation scheme, or by using the buffer occupancy for rate adaptation. However, these solutions also have the same limitations when variable bitrate is used. To deal with this problem, the DASH player needs to take a video content approach into account when adapts the video bitrate. Assuming the DASH player knows the size of next segments, it can adapt their video bitrates to the variations in network conditions without any side effects of VBR properties. In next section, we will describe our proposed scheme that takes the video content into account.

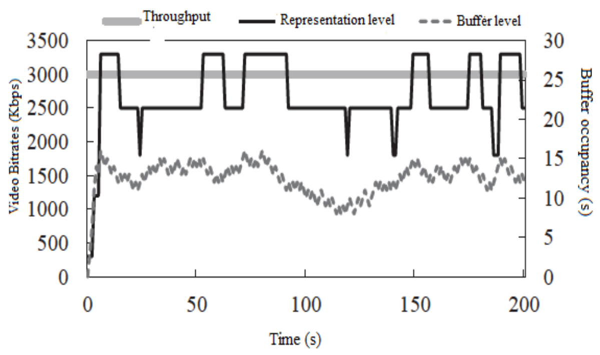


Fig 2: Video streaming under VBR

3. VIDEO BITRATE ADAPTIATION

The main goal of video bitrate adaptation scheme is to enhance the video quality at user side. In general, any video bitrate adaptation algorithm could simply prevent buffer underflow/overflow and ensure continuous video playback but it may cause video bitrate oscillation. In this section, we will present our proposed scheme which includes two main parts: throughput estimation and bitrate adaptation.

3.1 Throughput Estimation

The proposed scheme measures the throughput for each segment using the information about segment from MPD file. At the end of downloading a video segment, the segment throughput is computed as:

$$\hat{T}_{i+1} = \frac{(b^j_{i+1} - b^j_i + 1) \times 8}{t_e - t_s} \tag{2}$$

where j is the video quality level and b_{i+1} is the last byte of the(i+1)th segment. The video buffer length B_i can be modelled using the buffer filling rate and the buffer draining rate:

$$B''_{i+1} = f_{i+1} - d_{i+1} = \frac{\left(B_{i+1} - B_i\right)}{\tau} \tag{3}$$

where buffer filling rate (f_{i+1}) can be defined as:

$$f_{i+1} = \frac{\hat{T}_{i+1}}{V_{Ri+1}} \tag{4}$$

While the buffer draining rate (d_{i+1}) is defined as:

$$d_{i+1} = \begin{cases} 1 & \text{video is playing} \\ 0 & \text{video is stopping} \end{cases} \tag{5}$$

With assuming that video bitrate is lower than the available bandwidth, the video buffer length increases because the filling rate always more than 1.

At the client aspect, the video player uses an adaptive algorithm to determine the suitable bitrate to be selected for each segment depending on accurately calculation of the required time to download the next segment. Every segment that is downloaded placed in a buffer of maximum size B_{max}.

So the buffer is associated with three operating thresholds (B_0, B_{low}, B_{high}). Figure 3 below shows the relationship between segment throughput T(t) and video rate $V_R(t)$ with respect to a video playback buffer B(t). Where V_R / T is the drain rate at time t, and B(t) is the current buffer occupancy at time t.

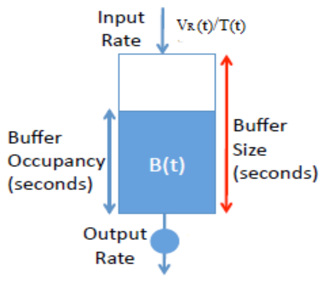


Fig 3: Video Playback Buffer

3.2 Proposed Bitrate Adaptive Algorithm

The bitrate adaptive algorithm is implemented after each segment is downloaded to determine the representation that will be chosen for the next segment. The algorithm is formatted with the available video bitrates L (representations) and thresholds for buffer B_0 (default = 2), B_{low}, B_{high}, B_{max}. These items remain constant throughout the entire video playing. The sizes of (i + 1)th segment across all the available representations ($s^{min}_{i+1}, \dots, s^{n}_{i+1}, \dots, s^{max}_{i+1}$), with the current buffer occupancy B_{curr} and estimated bandwidth for each segment. The algorithm process includes four steps:

1) Startup Step: At the beginning of the video session (or after an interruption due to buffer empty state), the number of segments in the buffer is below B_0 and the video playback begins in this step, during which the lowest bitrate is selected. The wait time is set to zero hence the next segment is downloaded immediately. Once the video buffer occupancy increases above B_0, the algorithm enters step-2. As the current buffer is less than B_0, the minimum bit rate is determined. This ensures that the startup time is maintained as small as possible. Note that reducing the startup time is important to prevent the user from giving away the video session. Also, when there are many competed clients on network resources, the share bandwidth link is in more participated case i.e. a congestion occurs, the bandwidth estimated by the clients are approximately equal, and fairness can be obtained. However when congestion occurs, the video buffer at the client side will be underflow, causing playback stop since clients may ask segments with bitrates more than the share bandwidth , which degrade video quality of experience. So the algorithm decreases the video bitrate to minimum level.

2) Increase Step: Once the video buffer exceeds B_0, the algorithm enters the bitrate increase phase. At this point, the video bitrate is increased in small steps for preventing the buffer back to the initial state in a conservative way. The time needed to download the next segment that corresponds to the current video bitrate is given by $\left(s^n_{i+1} / \hat{T}_{i+1} \right)$. At any time during the video playing, it is not feasible to download the next segment of the current bitrate before the video buffer goes below B_0, if the time needed to download the next segment is more than B_{curr}- B_0; hence, the best possible bitrate that can be downloaded in the duration B_{curr}- B_0 is chosen relied on the next segment size and the current estimated bandwidth. The algorithm attempts to keep the buffer level above B_0 and t_w

the waited time is set to zero; else, it returns back to the lowest bitrate (V_{Rmin}). In the increase step, the algorithm starts the adjustment process during which the video bitrate is gradually increased. The increase in quality takes place in one step i.e.: ↑ indicates the next level of bitrate. Using one step increments we ensure incremental quality changes as well as avoiding the unpleasant effect of switching bitrate. When the video session is in the increase step, the current buffer is less than B_{low}, the amount of time required to load the next segment is less than the B_{curr}- B_0, and then the next level is chosen. Otherwise, the current bit rate is preserved.

3) Steady Step: Once B_{curr} goes further than B_{low}, it starts to be more optimistic. By using segment sizes in the decision-making process, we ensure that even when segment sizes oscillate, the downloading time for the segment can be predicted more accurately. In step-3, the next segment request is sent immediately. The algorithm selects the highest useful video rate, making sure that the buffer is maintained at a steady level. This step represents the area between B_{low} and B_{high} (the most favoured buffer state). At this point, based on the current bandwidth link and occupancy of the buffer, the most appropriate video bitrate is determined which is greater than or equal to the current bitrate for ensuring a gradual and a stable increase in the video quality. Here, the link again be in more participated case i.e. the measured throughput starts to converge to the fair-share bandwidth so, the algorithm enter the steady phase.

4) Schedule Step: When the buffer occupancy increases above B_{high}, the most suitable video bitrate under the current bandwidth condition is selected. But, the request for next segment is sent only when the buffer occupancy falls under B_{high}. The schedule time limits the total number of segments in the video buffer, thus avoiding buffer overflow in case buffer reaches the maximum level B_{max}, i.e. the share bandwidth is in less participated case , the requested video bitrate is smaller than the available bandwidth, and OFF periods are needed to stop the transmission so as to avoid buffer overflow. Finally the algorithm returns the bit rate for the next segment.

After the buffer is filled, the client enters the periodic On-Off phase. During the phase, competing clients may perceive a biased view of the network conditions leading to instability, unfairness and under-utilization of bandwidth. This behaviour is observed in the presence of the TCP competitors and other competing HTTP clients. To deal with this situation, a random scheduler is presented based on a dynamic indicator for buffer level B_{id} which is calculated according to expression (6) .The indicator is used to request the next segment since the segments of higher video rates are greater in size they will take more time to get downloaded. To deal with this situation, a random scheduler is presented based on a dynamic indicator for buffer level B_{id} which is calculated according to expression (6) .The indicator is used to request the next segment since the segments of higher video rates are greater in size they will take more time to get downloaded. In the case when the share bandwidth link in the more participated case or in less participated case there should be a suitable segments available in the playback buffer to avoid buffer underflow or overflow. We present a time for requesting the next segment depends on the measured B_{id} as below:

$$B_{id} = B_{low} + (B_{max} - B_{low}) \times \frac{V_{Ri+1}}{V_{RL}}, \quad V_{Ri+1} > V_{R\min} \quad (6)$$

The HTTP adaptive streaming clients download the segments without any wait until the video buffer level reaches the dynamic indicator, B_{id}. The time to request the next segment is computed according to:

$$t_w = \begin{cases} 0 & B_{curr} < B_{id} \\ \tau & \text{otherwise} \end{cases} \quad (7)$$

<u>The following steps for Bitrate adaptation algorithm:</u>

```
Input :
$V$ : the set of L different bitrates $\{V_{R\min},......,V_{Rn},......,V_{R\max}\}$
$B_0, B_{low}, B_{high}, B_{max}$ : predefined values(no. of segments)
$i^{th}$ : the current download segment
$V_{Ri}$ : the bitrate of current download segment
$B_{curr}$ : current buffer occupancy in seconds
$\{s_{i+1}^{min},...,s_{i+1}^{n},....,s_{i+1}^{max}\}$ : the sizes of segments according
to bitrates $\{V_{R\min},...,V_{Rn}....,V_{R\max}\}$ respectively
$\hat{T}_{i+1}$ : the estimated bandwidth for $(i+1)^{th}$ segment.
Starting:
If  $B_{curr} \le B_0$                    Step-1
then
  $V_{Ri+1} = V_{R\min}$ ;
else {
  if $\left(\frac{s_{i+1}}{\hat{T}_{i+1}}\right) > B_{curr} - B_0$  then
    $V_{Ri+1} = \max\left\{V_{Rn} \mid V_{Rn} \in V, \left(\frac{s_{i+1}^{n}}{\hat{T}_{i+1}}\right) \le B_{curr} - B_0, n \le i\right\}$;
    $t_w = 0$;

else if  $B_{curr} \le B_{low}$            Step-2
then
  if $\left(\frac{s_{i+1}}{\hat{T}_{i+1}}\right) < B_{curr} - B_{low}$
    $V_{Ri+1} = V_{Ri}$ ↑   // rise by one step
  else
    $V_{Ri+1} = V_{Ri}$;
  $t_w = 0$;
else if  $B_i \le B_{high}$                Step-3
then
  $V_{Ri+1} = \max\left\{V_{Rn} \mid V_{Rn} \in V, \left(\frac{s_{i+1}^{n}}{\hat{T}_{i+1}}\right) \le B_{curr} - B_0, n \ge i\right\}$;
  $t_w = 0$;
else if  $B_i > B_{high}$                  Step-4
then
  $V_{Ri+1} = \max\left\{V_{Rn} \mid V_{Rn} \in V, \left(\frac{s_{i+1}^{n}}{\hat{T}_{i+1}}\right) \le B_{curr} - B_{low}, n \ge i\right\}$;
  $t_w$ is calculated as (14);
else
  $V_{Ri+1} = V_{Ri}$
  $t_w = 0$;
}
Result
$V_{Ri+1}$ : the video bitrate of the next segment to be downloaded
```

4. SIMULATION RESULTS

The proposed scheme is evaluated using ns-3 network simulator. To achieve adaptive streaming, the HTTP server offers the client seven levels of representations to adapt the video rates these are V={ 356, 500, 800, 1200, 1500, 2400 and 3500Kbit/s}. The length of video segment and video buffer is 2s and 35 sec, respectively. The values of other parameters B_{low}, B_{high} and B_0 are 15s, 30s and 5s respectively. To evaluate the performance of the proposed scheme, we compare it with other schemes such as buffer based bitrate adaptation algorithm "BBA" [6] and throughput based bitrate adaptation "TBA" [8]. The segment sizes range from 90% to 195% of the average segment size.

Figure 4 and Figure 5 shows that the conventional schemes change their bitrate very frequently whereas the proposed scheme adapts the video bitrate smoothly using the actual segment size and adapts its bitrate based on the future buffer occupancy. Bandwidth-based algorithm "TBA" is extremely oscillating in this scenario, since its estimation method does not consider VBR characteristics, as we mentioned earlier. VBR characteristics also affect buffer-based algorithm, like BBA.

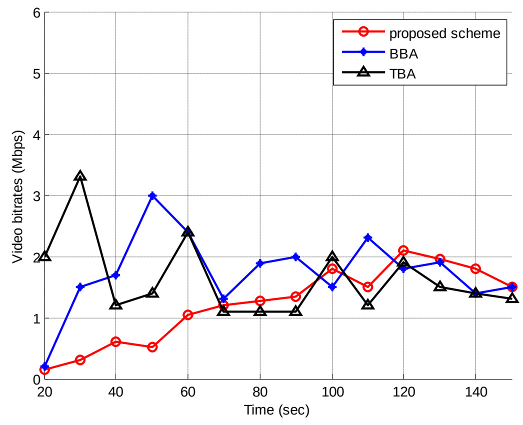


Fig 4: Video bitrates vs. time under segment size variations

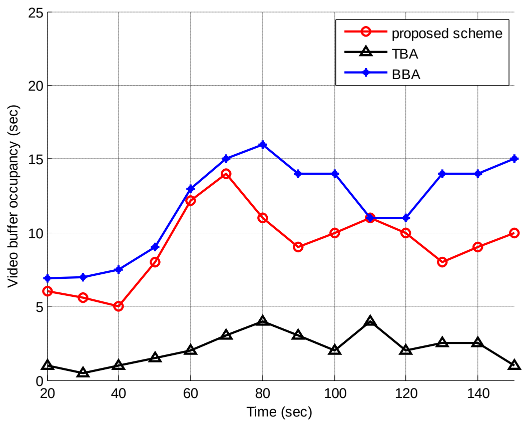
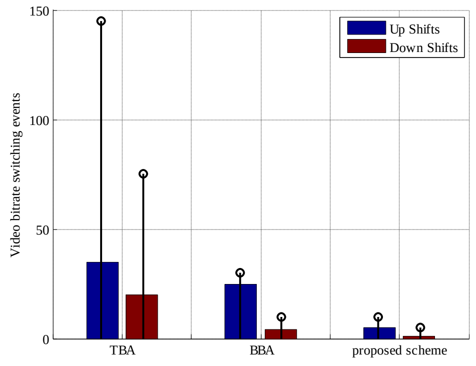
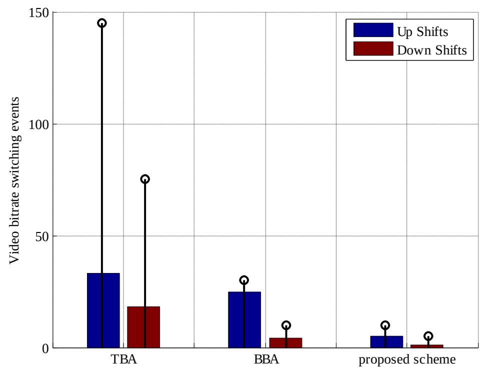


Fig 5: Video buffer occupancy vs. time under segment size variations

Figure 6 below presents the polarity of the bitrate switches where the maximum bandwidth was chosen as 1Mbps and 6Mbps. We compare the proposed scheme with BBA and TBA schemes. Since the schemes used are additive increase and aggressive switch down, we observed that most of the switching events are upward switches. With our scheme, we not only observed a reduction in positive switches, we find the negative switches to be significantly lower. While the variance in the switching events with TBA is very high due to the high switching events observed with one particular video. Because our scheme explicitly considers the segment size variations, it is able to better estimate the variation in the segment fetch times and hence, sustains a consistent QoE, irrespective of the video.



a: Max. bandwidth= 1Mbps



b: Max. bandwidth= 6Mbps

Fig 6: Polarity of video bitrate switches

One of the challenges of an adaptive bitrate scheme is to reduce the convergence time. For the BBA that depends on the buffer occupancy to determine the appropriate bitrate, the convergence time found to be higher. However, with the proposed scheme, the bandwidth measurement, along with the awareness of the segment sizes, enabled our scheme to make better estimates of the download rate of the next segments. This drove the player towards the highest bitrate much faster. In Figure 7, we see that the convergence times of our scheme were significantly better than both TBA and BBA.

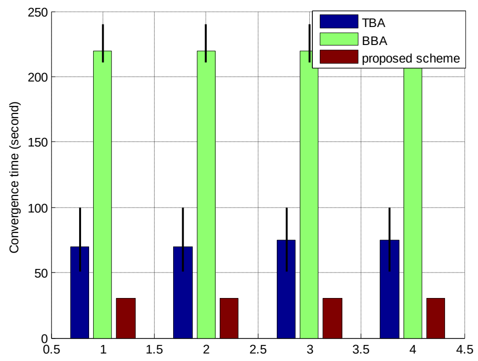


Fig 7: Convergence time with Mean Bandwidth 6Mbps

5. CONCLUION

In this work, a bitrate adaptive approach is designed on the basis of stability and fairness in DASH system under the scenario that many clients compete for network resources. In this paper, we propose a content-aware rate adaptation scheme to reduce the effects of inaccurate bandwidth estimation. The proposed scheme aims to estimate network bandwidth by considering variable segment size, and to prevent bitrate oscillations based on the buffer information. To obtain buffer information, we present a playout buffer model. Besides, the video bitrate is allowed to be higher than the measured bandwidth, thereby achieving a higher bandwidth utilization efficiency and higher average video bit-rate compared with a conventional scheme. Finally taking into account the variation of segment sizes in adapting the bitrates for next segment can improve the QoE of video streaming in term of low bitrate switching events, high video quality, and fast convergence times compare with other adaptive algorithms.

6. REFERENCES

[1] Cisco System Inc. 2016. Cisco Visual Networking Index: Forecast and methodology, 2015-2020. Cisco Company-San Jose, CA, USA.

[2] Stockhammer, T., 2011, February. Dynamic adaptive streaming over HTTP--: standards and design principles. In Proceedings of the second annual ACM conference on Multimedia systems (pp. 133-144). ACM.

[3] Roettgers, J., 2013. Don't touch that dial: How YouTube is bringing adaptive streaming to mobile, TVs.

[4] Zhou, B., Wang, J., Zou, Z. and Wen, J., 2012, January. Bandwidth estimation and rate adaptation in HTTP streaming. In Computing, Networking and Communications (ICNC), 2012 International Conference on (pp. 734-738). IEEE.

[5] Mok, R.K., Luo, X., Chan, E.W. and Chang, R.K., 2012, February. QDASH: a QoE-aware DASH system. In Proceedings of the 3rd Multimedia Systems Conference (pp. 11-22). ACM.

[6] Huang, T.Y., Johari, R., McKeown, N., Trunnell, M. and Watson, M., 2015. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. ACM SIGCOMM Computer Communication Review, 44(4), pp.187-198.

[7] Kim, S., Yun, D. and Chung, K., 2016, January. Video quality adaptation scheme for improving QoE in HTTP adaptive streaming. In Information Networking (ICOIN), 2016 International Conference on (pp. 201-205). IEEE.

[8] Liu, C., Bouazizi, I. and Gabbouj, M., 2011, February. Rate adaptation for adaptive HTTP streaming. In Proceedings of the second annual ACM conference on Multimedia systems(pp. 169-174). ACM.

[9] De Cicco, L. and Mascolo, S., 2014. An adaptive video streaming control system: Modeling, validation, and performance evaluation. IEEE/ACM Transactions on Networking (TON), 22(2), pp.526-539.

[10] Miller, K., Quacchio, E., Gennari, G. and Wolisz, A., 2012, May. Adaptation algorithm for adaptive streaming over HTTP. In Packet Video Workshop (PV), 2012 19th International (pp. 173-178). IEEE.

[11] Jiang, J., Sekar, V. and Zhang, H., 2014. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. IEEE/ACM Transactions on Networking (TON), 22(1), pp.326-340.

[12] Yin, X., Bartulović, M., Sekar, V. and Sinopoli, B., 2017, May. On the efficiency and fairness of multiplayer HTTP-based adaptive video streaming. In American Control Conference (ACC), 2017 (pp. 4236-4241). IEEE.

[13] Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A.C. and Oran, D., 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. IEEE Journal on Selected Areas in Communications, 32(4), pp.719-733.

[14] Zhou, C., Lin, C.W. and Guo, Z., 2016. mDASH: A Markov decision-based rate adaptation approach for dynamic HTTP streaming. IEEE Transactions on Multimedia, 18(4), pp.738-751.

[15] Jiang, J., Sekar, V. and Zhang, H., 2014. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. IEEE/ACM Transactions on Networking (TON), 22(1), pp.326-340.