Client Driven Rate Adaptation Algorithm for Streaming over HTTP

Waqas ur Rahman and Kwangsue Chung
Department of Electronics and Communications Engineering
Kwangwoon University
Seoul, Korea
Email: waqas@cclab.kw.ac.kr, kchung@kw.ac.kr

***Abstract*— Video streaming services make up a large proportion of Internet traffic throughout the world. Adaptive streaming allows for dynamical adaptation of the video bitrate with varying network conditions, to guarantee the best user experience. We propose an adaptive bitrate scheme that intelligently selects the video bitrates based on the estimated throughput and buffer occupancy. We show that the proposed algorithm selects a high playback video rate and avoids unnecessary rebuffering while keeping a low frequency of video rate changes.**

Keywords- Rate adaptation; Quality adaptation; Quality of Experience; HTTP Streaming; Multimedia

I. INTRODUCTION

High speed broadband networks and improvements in display technology of various devices have enabled video streaming to become one of the most popular applications. Video traffic dominates Internet traffic on both fixed and mobile access networks all over the world.

Initially, the video clients completely downloaded the video before the streaming could start. This was followed by the progressive download with which the clients begin the playback at a defined video rate before the download is complete. Recently, video streaming services are based on Hypertext Transport Protocol (HTTP) over TCP for streaming multimedia over computer networks. Network conditions and video clients' capabilities vary with time and place; therefore, adaptive streaming over HTTP allows the adaption of video quality based on the available resources on the path between the server and client. Multiple versions of the multimedia content are stored at the server. The server shares the information about the characteristics of the stored multimedia content with the client. The adaptive bitrate (ABR) algorithm at the video client is responsible for selecting a suitable bitrate depending on the system conditions such as throughput and the occupancy of the playback buffer.

ABR algorithms strive to maximize the user experience by meeting conflicting video quality objectives in different environments. Some of the potential objectives include selecting a set of video bitrates that are the highest feasible, avoiding needless video bitrate switches and preserving the buffer level to avoid interruption of playback [1][2]. Maximizing the video rate increases the risk of playback interruption whereas mitigating the frequency of video rate switches results in lower average video rate.

One way to pick video bitrates is to make an estimate of the future throughput from past observations. An inaccurate estimation may lead to selecting the video bitrate that results in extensive rebuffering. If the selected video rate is higher than the available throughput, the client's playback buffer drains which may result in interrupted playback. To avoid interrupting playback, ABR algorithms add playback buffer occupancy as an adjustment parameter on top of throughput estimation to select video bitrates.

In this paper, we show that the proposed algorithm selects the video rates based on the buffer occupancy by exploiting the variation of the sizes of the upcoming segments. The results show that our approach provides better viewing experience by delivering higher average video rate without unnecessary rebuffering while maintaining a low frequency of video rate changes. The rest of the paper is organized as follows. The related works are presented in Section II. The proposed scheme is presented in Section III. The experimental results are provided in Section IV, and finally the concluding remarks are given in Section V.

II. RELATED WORK

The main objective of all adaptive video rate algorithms is to improve the user's viewing experience. Adaptation algorithms mainly select video rates based on the estimated throughput and the state of the playback buffer. Segment throughput is calculated as the ratio of the segment size to the time that it takes to download the segment [3]. In many commercial clients, the moving average of the throughput of previous segments is used to estimate the throughput [4]. Once the throughput has been estimated, clients pick the video rate of the next segment based on the throughput [5-7].

Many ABR algorithms consider playback buffer along with the throughput to select the video rate of the next segment. The buffer is divided into predefined ranges and different decisions are taken to select the video rates when the buffer level stays in different ranges [8][9]. The method in [9] is more stable as compared to the method in [8] but it is late to react to the changes in the throughput as it waits for the playback buffer to reach a threshold before selecting a higher video rate. We propose an adaptive bitrate scheme [10] that intelligently selects the video bitrates based on the estimated throughput and buffer occupancy. The scheme

 ISBN: 978-1-61208-482-4

improves viewing experience by achieving a high video rate without taking unnecessary risks and by minimizing the frequency of changes in the video quality. Huang *et al* [11] propose a video rate adaption algorithm that selects the video rate by observing only the client's playback buffer. The video rate is increased and decreased as the playback buffer builds up and drains respectively. Furthermore, the algorithm selects the video rates considering the sizes of the upcoming segments. In this paper, we propose a scheme that is similar to the schemes proposed in [8][9][13] as it selects the video rates based on both the estimated throughput and the buffer occupancy. The current schemes in the literature pick the video rates based on the predefined buffer ranges whereas the proposed scheme dynamically selects the buffer ranges to optimally pick the video rates based on the upcoming segment sizes to optimize the QoE.

III. PROPOSED SCHEME

A. System Model

The HTTP client downloads a video stream divided into multiple segments. The video stream is stored at the server and the adaptive bitrate algorithm at the client decides which segment to download next. All the segments have an equal duration of τ seconds. The set of representations available for the video stream is denoted by R where $R = \{R_{min}, R_2, R_3, \ldots, R_{max}\}$. The client dynamically selects a video rate from the set R for the next segment. R_{min} and R_{max} are the representations with the highest and lowest video rates in the set R. Any video rate higher and lower than currently selected video rate is denoted by $R\uparrow$ and $R\downarrow$ respectively.

B. Adaptive Bitrate Algorithm

Available bandwidth estimation plays an important part in the selection of the video rate. The clients estimate the throughput of the next segment based on the throughput observed over the download of the previous segments. Segment throughput is calculated as the ratio of segment size divided by the time it takes to download the segment. The selection of the video rate for the next segment based on the throughput $T(i\text{-}1)$ of the last downloaded segment keeps the playback buffer stable but results in a fluctuating video rate curve. In this paper, we use the McGinely dynamic indicator for the throughput estimation measure $T^E(i)$ to overcome the fluctuating video curve which is given by [12]:

$$T^E(i+1) = T^E(i) + \frac{T(i) - T^E(i)}{N \times (\frac{T(i)}{T^E(i)})^4} \quad (1)$$

The numerator of the second term gives a sign, up or down and the power of 4 gives the calculation an adjustment factor which increases more sharply as the difference between the observed throughput of i^{th} segment and estimated throughput of segment i increases. N is the tracking factor which we set equal to 1.

The buffer dynamics are considered when the segment is completely downloaded. Let $B(i\text{-}1)$ be the buffer level at the end of the download of segment i-1, then $B(i)$ is given by:

$$B(i) = B(i-1) + \tau - [\tau \times \frac{R_k(i)}{T(i)}] \quad (2)$$

where $R_k(i)$ is the k^{th} video rate from the set R and $T(i)$ is the throughput observed during the download of segment i. (2) shows that if the selected video rate is greater than the available throughput, the playback buffer drains. As each segment contains duration of τ seconds, $C_k(i)$, the size of the i^{th} segment is $\tau \times R_k$ bits. Given the available throughput $T(i)$ and video rate $R_k(i)$, the change in buffer level during the download of i^{th} segment is equal to B^*:

$$B^* = B(i) - B(i-1) = \frac{C_k(i)}{R_k(i)} - \frac{C_k(i)}{T(i)} \quad (3)$$

where the playback buffer fills with $C_k(i) / R_k(i)$ seconds of data and the buffer drains with $C_k(i) / T(i)$ seconds of data. (2) can now be written as:

$$B(i) = B(i-1) + B^* \quad (4)$$

If $R_k(i) > T(i)$, B^* becomes negative, which means that the buffer is drained at a rate faster than the rate at which it fills, therefore, $B(i)$ will be less than $B(i\text{-}1)$. We assume that the available throughput cannot be less than $R_{min} = R_1$. We denote the change in the buffer level when $T(i) = R_{k-1}$ and the client overestimates the throughput and selects the next higher video rate R_k for the i^{th} segment as B_k^*. We denote $B_k(i)$ as the minimum buffer level occupancy to select the k^{th} video rate for the i^{th} segment.

$$B_k(i) = \tau + \left| \sum_{m=2}^{k} B_m^* \right| = \tau + \left| \sum_{m=2}^{k} \frac{C_m(i)}{R_m} - \frac{C_m(i)}{R_{m-1}} \right| =$$
$$= \tau + \left| \sum_{m=2}^{k} \frac{C_m(i) \times R_m - C_m(i) \times R_{m-1}}{R_m \times R_{m-1}} \right| \quad (5)$$

(5) ensures that if the client selects the k^{th} video rate when at least $B_k(i)$ amount of buffer is available and the throughput drops to R_{min}, there will be one segment (τ seconds) available in the buffer at the end of the segment download. Most of the streaming services encode videos in variable bitrate (VBR) where static scenes are encoded with fewer bits and active streams with more bits. In VBR, video is encoded at an average video rate and the instantaneous video rate of each segment varies around the average rate. This allows flexible and efficient use of bits. As the size of each segment is different and $B_k(i)$ depends on the segment size, the value of $B_k(i)$ will change every time a segment is downloaded. This makes the video rate change frequently. Furthermore, for a given throughput a segment of a larger size will take more time to get downloaded; hence will consume more video in the buffer than a smaller segment. To

 ISBN: 978-1-61208-482-4

this end, we take the average of the next 10 segment sizes and calculate $B_k(i)$ after every 10 segments based on their average sizes.

$$B_k(i) = \tau + \sum_{m=2}^{k} \frac{\bar{C}_m \times R_m - \bar{C}_m \times R_{m-1}}{R_m \times R_{m-1}} \quad (6)$$

where $\bar{C}_m$ is the average of every 10 segment sizes. (6) makes sure that $B_k(i)$ gets its value recalculated after every 10 segments to reduce the video rate switches. If the upcoming segments are larger, the buffer thresholds to select a given video rate will be greater than when segments are smaller to minimize the risk of buffer underflow. If we select the average segment size based on more than 10 upcoming segments, it might not correctly depict the segment size trend whereas calculating $B_k(i)$ based on fewer segments will result in a higher frequency of video rate switches.

The algorithm's pseudo-code is provided in Algorithm 1. We consider that Algorithm 1 is invoked to select the video rate of i^{th} segment. The streaming session is divided into two phases of operation: the startup phase and the steady phase. The startup phase starts when the buffer is building up from being empty, to be followed by the steady phase.

Algorithm 1: Adaptation Algorithm

if Startup phase conditions hold true
  if $B(i\text{-}1) < B_{LOW}$ ***then***
    if $R\uparrow < \alpha_1 \times T(i\text{-}1)$ ***then***
      $R_k(i) = R\uparrow$
  else
    if $R\uparrow < \alpha_2 \times T(i\text{-}1)$ ***then***
      $R_k(i) = R\uparrow$
else
  if $B(i\text{-}1) < B_{min}$ ***then***
    $R(i) = R_{min}$
  else if $R(i\text{-}1) == R_k$ && $B_{k\text{-}1}(i) < B(i\text{-}1)$ ***then***
    $R(i) = R(i\text{-}1)$
  else if $R(i\text{-}1) \neq R_{min}$ && $B(i\text{-}1) < B_{k\text{-}1}(i)$ ***then***
    $R(i) = R\downarrow$
  else if $R(i\text{-}1) \neq R_{max}$ && $B(i\text{-}1) > B_k\uparrow$ && $T^E(i) > T^E(i\text{-}1)$ ***then***
    $R(i) = R\uparrow$
  else
    $R(i) = R(i\text{-}1)$

During the startup phase, the buffer builds up from being empty. A conservative approach is considered at the start and as the buffer gradually fills up and climbs above the buffer threshold B_{LOW}, we take more risk in selecting the video rate. Minimum available video rate R_{min} is selected to download the first segment. This approach reduces the delay after the client requests the video and before the client streams the video. For $B(i\text{-}1) < B_{LOW}$, the client switches to a higher video rate if $R\uparrow < \alpha_1 \times T(i\text{-}1)$. For $B(i\text{-}1) > B_{LOW}$, a higher video rate is selected if $R\uparrow < \alpha_2 \times T(i\text{-}1)$ where α_1 and α_2 are the safety margins and $\alpha_1 < \alpha_2$. When the buffer size is small, the client will increase the video rate faster in the startup phase. When the buffer size is large, it may take time for the client to accumulate buffer up to B_{LOW} which may result in underutilization of the resource when the available throughput is high. To avoid this scenario, we set the condition that if $R_{startupphase} < R_{steadyphase}$ the algorithm switches to steady phase. $R_{startupphase}$ and $R_{steadyphase}$ are the video rates suggested by the client during the startup and steady phase respectively. The proposed scheme stays at the startup phase until any of the following conditions are not satisfied: (i) $B(i\text{-}2) < B(i\text{-}1)$; or (ii), $R_{startupphase} > R_{steadyphase}$. The motivation behind the startup phase is to quickly fill up the buffer without risking playback interruption. Afterwards, we use steady phase to select the video rate of the upcoming segments.

In the steady phase, to select the k^{th} video rate, two conditions should be satisfied:

1) The buffer level should be higher than $B_k(i)$
2) $R_k(i) < \alpha_3 \times T^E(i)$

The client will select $R_k(i)$ if the buffer level is greater than $B_k(i)$. This condition helps in avoiding the buffer underflow in case the client overestimates the throughput or there is a sudden drop in the throughput. The condition of $R_k(i) < \alpha_3 \times T^E(i)$ uses a safety margin α_3 to compute the bitrate to avoid throughput overestimation.

First we consider the scenario where buffer level falls below $B_{min} = B_2(i)$. In this case, R_{min} is always selected. $B_2(i)$ is the minimum buffer occupancy to select the video rate $R_2(i)$. The reason is that it is of the primary importance to avoid interruption of the playback.

Now, we consider the scenario when the throughput and the buffer level drops. We do not immediately react to this drop in the throughput; we stay at the current video rate until the buffer level drops below $B_{k-1}(i)$. This is because we can minimize the number of video rate switches if we don't react to short-term fluctuations. Once the buffer level falls below $B_{k-1}(i)$, we continue to reduce the video rate until the condition $R_k(i) < \alpha_3 \times T^E(i)$ is satisfied.

Next, we consider the scenario of an increase in throughput and the buffer level. To increase the video rate in response to the increase in throughput and buffer level, the following conditions should be satisfied:

1) $T^E(i) > T^E(i\text{-}1)$
2) The buffer level should be greater than $B_k\uparrow$

The first condition makes sure that there isn't a recent drop in throughput while the client decides to increase the video rate. $B_k\uparrow$ is the buffer threshold to select the higher video rate $R\uparrow$. As the video rate cannot be adapted until the download of the next segment, in case of a sudden drop in throughput the second condition reduces the risk of buffer underflow. When the conditions of switching up and switching down the video rate are not satisfied, we maintain the current video rate.

IV. PERFORMANCE EVALUATION

We implement the proposed scheme in ns-3 to evaluate its performance. We compare the proposed method with the schemes proposed in [8] and [9]. We refer to the algorithms proposed in [8] and [9] as AAAS and QAAD respectively. The topology implemented in this paper is shown in Figure 1.

The topology consists of an HTTP server, HTTP client and a pair of network elements. The link between the network elements is our bottleneck link. We add the UDP traffic between the network elements to vary the throughput across the bottleneck. To achieve adaptive streaming, the HTTP server offers the client four different video rates which include 450, 850, 1500 and 2500kbps. The length of each segment and playback buffer size is 4 and 60 seconds, respectively. B_{LOW} is set to 30% of the buffer size. The safety margins are set to (α_1, α_2, α_3)=(0.5, 0.75, 0.9).

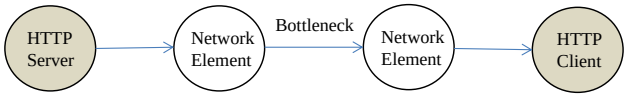

Figure 1. Network topology

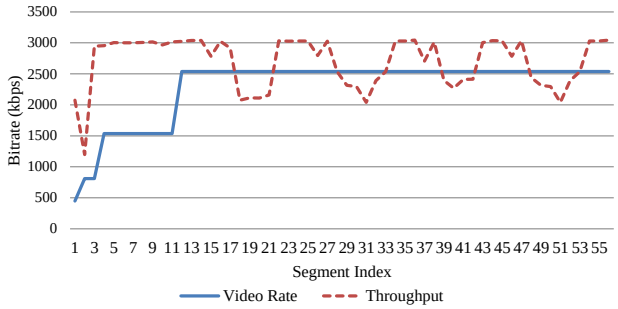

Figure 2. Response of the proposed scheme to small drop in throughput

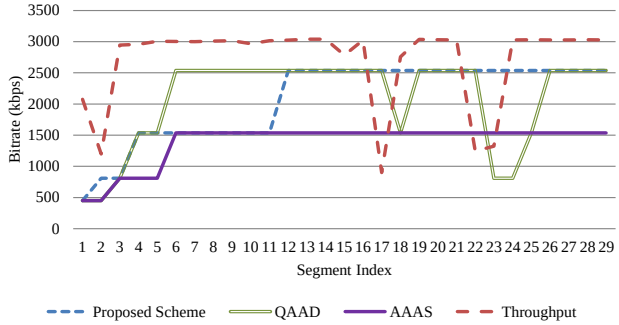

Figure 3. Comparison of the schemes in response to large throughput fluctuation

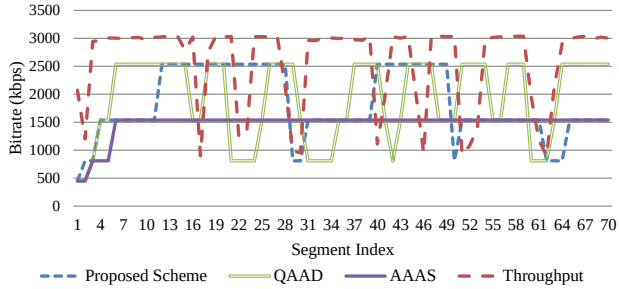

Figure 4. Comparison of the schemes in response to highly variable available bandwidth

Figure 2 shows the response of the proposed scheme to short term throughput fluctuations. It shows that the proposed scheme is stable in the face of short term throughput fluctuations while maintaining a high video rate.

Figure 3 shows that the proposed scheme does not vary the video rate quickly as it maintains video rate at the expense of drop in buffer level below $B_k(i\text{-}1)$. The motivation behind maintaining the video rate at the expense of drop in buffer level is that the objective of ABR algorithm is not to keep the buffer full but to provide better user experience. AAAS scheme shows a stable response to the throughput fluctuations but stays at a lower video rate. QAAD scheme varies the video rate as the throughput fluctuates in order to avoid buffer underflow.

Figure 4 shows that the proposed scheme tries to maintain the higher video rate but reacts swiftly to large drop in throughput to avoid any playback interruption. The AAAS scheme is the most conservative of all the schemes. The reason is that it waits for the playback buffer to cross a predefined threshold before stepping up or down the video rate. The proposed scheme achieves an average of video rate of 350kbps higher than AAAS. QAAD has slightly higher video rate than the proposed scheme but at the expense of twice the number video rate switches which greatly degrades the user experience.

V. Conclusion and Future Work

Video rate adaptation techniques are used to adapt the quality of the video to the varying network resources of the computer network. In this paper, we proposed an adaptive bitrate streaming algorithm to improve the viewing experience of the multimedia streaming applications. The proposed algorithm achieves high video rate and minimizes the frequency of changes in video quality while preventing interruption in playback to guarantee QoE. In this paper, we consider a single client scenario. For the future work, we plan to extend our algorithm to a multi-user scenario where multiple clients share the bottleneck.

Acknowledgment

This work was supported by ICT R&D program of MSIP/IITP. [R0101-16-293, Development of Object-based Knowledge Convergence Service Platform using Image Recognition in Broadcasting Contents]

References

[1] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, Sep. 2011, pp. 362-373.

[2] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker effects in adaptive video streaming to handheld devices," Proc. of ACM International Conference on Multimedia, Feb. 2011, pp. 463-472.

[3] T. C. Thang, Q. D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," IEEE Transactions on Consumer Electronics, vol. 58, no. 1, Feb. 2012, pp. 78-85.

[4] T. Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video

 ISBN: 978-1-61208-482-4

streaming rate is hard," Proc. of ACM Conference on Internet Measurement, Nov. 2012, pp. 225-238.

[5] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," Proc. of ACM Conference on Multimedia Systems, Feb. 2011, pp. 157-168.

[6] T. C. Thang, Q. D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," IEEE Transactions on Consumer Electronics, vol. 58, no. 1, Feb. 2012, pp. 78-85.

[7] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," Proc. of the ACM Conference on Multimedia Systems, Feb. 2011, pp. 169-174.

[8] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," Proc. of the IEEE Packet Video Workshop, May. 2012, pp. 173-178.

[9] D. Suh, I. Jang, and S. Pack, "QoE-enhanced adaptation algorithm over DASH for multimedia streaming," Proc. of IEEE Conference on Information Networking, Feb. 2014, pp. 497-501.

[10] W. Rahman and K. Chung, " Buffer-based adaptive bitrate algorithm for streaming over HTTP," KSII Transactions on Internet and Information Systems, vol. 9, no. 11, Nov. 2015, pp. 4585-4622.

[11] T. Huang, R. Johari, and N. McKeown, "Downtown abbey without the hiccups: Buffer-based rate adaptation for http video streaming," Proc. of ACM SIGCOMM workshop on Future Human-centric Multimedia Networking, Aug. 2013 , pp. 9-14.

[12] J. R. McGinley, "McGinley Dynamics," Market Technicians Association Journal, issue 48, 1997, pp. 15-18.

[13] P. Juluri, V. Tamarapalli, and D. Medhi, "SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP," Proc. of IEEE International Conference on Communication Workshop, June. 2015, pp. 1765-1770.

 ISBN: 978-1-61208-482-4