AN ADAPTATIVE BITRATE ALGORITHM FOR DASH

Yunlong Li[1], *Yue Wang*[1], *Shanshe Wang*[1], *Siwei Ma*[1,2]
[1]Institute of Digital Media & Cooperative Medianet Innovation Center, Peking University, Beijing, China
[2]Peking University Shenzhen Graduate School, Shenzhen, China
{yl_li, cyberrush, sswang, swma}@pku.edu.cn

ABSTRACT

Dynamic adaptive streaming over HTTP (DASH) has been widely used on the Internet. However, DASH does not impose any algorithm to choose video quality. In this paper, an adaptive bitrate switch algorithm for DASH player is proposed. Firstly, the proposed algorithm takes video playback quality, video rate switching frequency and buffer status into account in order to meet the available bandwidth. Secondly, several measures are designed specially to improve user visual quality. Besides, the proposed player takes a sub-optimal algorithm to avoid video playback interruptions. Experimental results demonstrate that the proposed player can provide better performance compared with Bitdash player in several aspects, such as video quality switch frequency, bandwidth utilization and subjective visual experience.

Index Terms—Dynamic Adaptive Streaming over HTTP (DASH), Rate Adaption, Video Quality Switch

1. INTRODUCTION

Dynamic adaptive streaming over HTTP (DASH) has been recently widely adopted for providing uninterrupted video streaming services to users under dynamic network conditions and heterogeneous devices [1][2]. It separates the video into many fragments by the time. The most important feature is that the server can provide video with different bitrate for each fragment, thus the client can request suitable fragments to adapt the video bitrate to a varying network bandwidth. However, DASH does not impose any logic for selecting the quality of video fragments.

Generally, a good implementation of DASH needs to take video playback quality, video rate switching frequency, buffer overflow/underflow, buffer occupancy and visual quality into account [3]. Based on the proposed algorithm in [3], we proposed one complete solution for DASH client with all metrics above taken into consideration.

This work was supported in part by the National Science Foundation of China(61322106, 61571017), National Basic Research Program of China(973 Program, 2015CB351800),and Shenzhen Peacock Plan, which are gratefully acknowledged.

The remainder of the paper is organized as follows: Section 2 describes the proposed algorithm. Section 3 presents our innovations. Experimental results are provided in Section 4. Section 5 is the summary.

2. PROPOSED ALGORITHM

A mDASH player proposed in [3] defined a state vector describing the current system situation, including the buffer occupancy and its changing rate, the video bitrates requested for previously downloaded fragments, the bitrate consistency function, and the bandwidth conditions. Based on [3], we provided an algorithm from the following aspects.

2.1. Adaptation algorithm

A good algorithm can request suitable fragment according to current network conditions to provide best visual experience.

Several factors must be taken into consideration to realize a good adaptation algorithm as we introduced in the beginning of this chapter. Our rate decision is made at k for fragment k+1. The buffered video time and its changing rate play important roles in rate switching decision [4]. We separate the buffered video time into three parts with thresholds $q_{\max}$ and $q_{\min}$. It is hard to find the optimal values of the two thresholds due to the complex network environment [3]. We improved thresholds recommended in [3] with numerous experiments.

2.1.1. Buffer overflow algorithm

Buffer overflow means the player is requesting lower video bitrate than the available bandwidth, so the downloaded fragments accumulate. The player can bring better visual experience with higher video bitrate. Our adaptation algorithm monitors the buffer state and takes several methods to avoid buffer overflow.

If the buffer video time is larger than $q_{\max}$, it means the available bandwidth is better than the requested fragment quality. Buffer overflow needs to be avoided by increasing the requested fragment bitrate. Note, if the bandwidth is too high, a sleep mechanism is used to delay the fragment request to avoid buffer overflow [4].

2.1.2. Buffer underflow algorithm

Buffer underflow means the player is requesting higher video bitrate than the available bandwidth. When the buffered video time is at low level, the video rate decision is mainly account for avoiding playback freeze.

Our adaptation algorithm decreases the requested fragment quality to avoid buffer underflow, when the buffer video time is less than $q_{\min}$. The fragment quality requested next is decided by the estimated bandwidth.

2.1.3. Smooth rate algorithm

When the buffer video time is between $q_{\min}$ and $q_{\max}$, our algorithm does nothing to avoid frequent video quality switches. It is shown in Section 4 this works very well to decrease the video bitrate switching.

Short-term bandwidth variations are common in practice, a good rate adaptation algorithm should be able to well compensate for such spikes. Our smooth rate algorithm utilizes the buffered video to compensate such short-term variations and avoids video quality fluctuations in such situation.

3. INNOVATIONS

Based on the mDASH proposed in [3], our proposed player improves performance in two aspects, including shorter start-up time and a better video rate switch performance.

3.1. Start-up algorithm

Start-up is an important aspect for improving player performance. Long start-up time means long time waiting for the video, which is quite annoying. Our proposed player decreases the start-up time with a novel Start-up algorithm.

It is obvious that high start-up bitrate will lead to long start-up time. This is not what we want. So our start-up algorithm starts with a low bitrate fragment, and then tries to adapt to the available bandwidth. It was shown that users generally prefer gradual quality change to an abrupt switching [5]. Accordingly, our start-up algorithm gradually improves the fragment bitrate. Also，significant quality step-up may cause buffer underflow.

For there are fragments with different segment duration, our start-up algorithm decides the start time by the downloaded video time rather than by the downloaded bits. While the downloaded video time exceeds the threshold value, our player will start to render the video.

3.2. A better smooth video rate

In general, a better performance can be achieved with larger buffer size if the buffer delay is allowed. This is because with a large buffer size, the bandwidth variations can be compensated with the buffered video and a smooth video rate is guaranteed [3]. Our proposed player allowed a longer buffer delay than mDASH introduced in [3] and achieved a much better performance in video rate smoothing.

The maximal buffer size of our proposed player is 90s compared with 30s in the mDASH player. The experiments in the next section reflect that video rate switch times decrease compared with the bitdash palyer.

4. PERFORMANCE EVALUATION.

In this section, we evaluate our player in lab environment on a network test-bed. The proposed player and the bitdash player are compared in terms of several quality metrics, including the start-up time, the average video bitrate, bandwidth utilization and stability.

4.1. Experiment setup

The proposed player is based on the open source library (Libdash [8]). Libdash provided a perfect framework for dash player implementation, with which we add our adaptation algorithm quickly. Our development environment is Visual Studio 2010 and Qt5.5.1 (32). The test-bed consists of three nodes: one web server used for media delivery, one router, and the proposed player. The proposed player and web server both runs on Win 10. The server is installed with the Apache HTTP server of version 2.4.12. Network Emulator is used in the server to control upload bandwidth, therefore the bottleneck is the bandwidth between the server and the router.

For comparison, we implemented bitdash player in our web server. The requested fragment bitrate is got from the web server log. So we can compare the fragments bitrate sequence between the proposed and the bitdash player.

We define the duration between the start button pressed and the video rendered as the start-up time. As the exact bitdash start-up time could not be obtained, we can only compare it subjectively.

The Host organization provides dataset with segment duration of 1, 2, 4, 6, 10, 15 seconds. It takes more time to download long duration segments, this drops start-up performance and quality switches. So we focus our effort on the dataset with segment duration 1, 2, 4. Our player also supports other types, but they are not well-tested.

4.2 Constant bandwidth

Here we present the results in the case that the available bandwidth is constant, the bandwidth is 3Mbps for **Fig.1** and 2Mbps for **Fig.2** respectively.

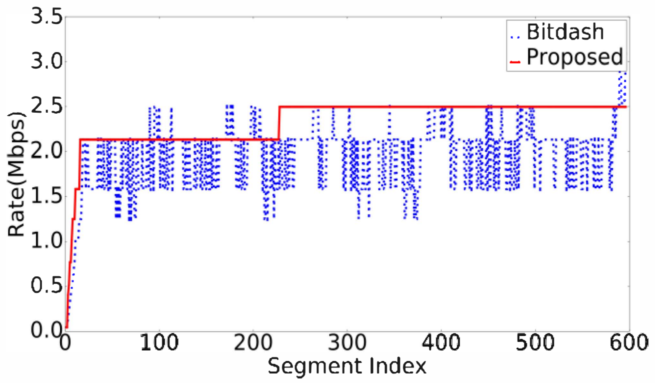

Fig. 1. The performance between proposed player and bitdash player on constant network on dataset BigBuckBunny 1s.

Table 1. Performance comparison under constant bandwidth variations

Column One	Bitdash	Proposed
Strat-up delay(ms)	-	82
Bitrate switch	266	8
Average bitrate(Mbps)	1.95	2.33
Bandwidth utilization(%)	65	77.67

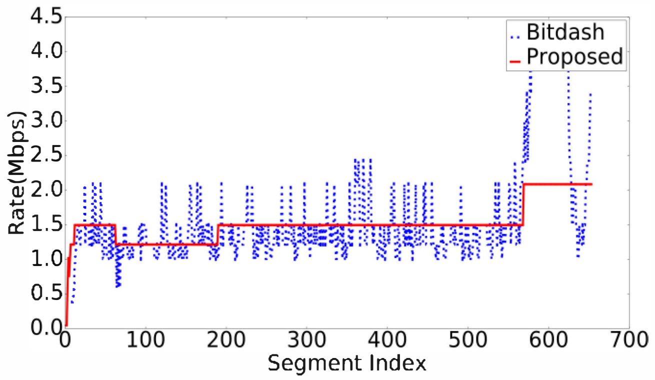

Fig. 2. The performance between proposed player and bitdash player on constant network on dataset ElephantDream 1s.

Table 2. Performance comparison under constant bandwidth variations

Column One	Bitdash	Proposed
Strat-up delay(ms)	-	78
Bitrate switch	364	8
Average bitrate(Mbps)	1.61	1.51
Bandwidth utilization(%)	80.5	75.5

It can be seen from the figures above that the bitdash player switches quickly in a larger scale when the available bandwidth is constant. Thus the bitdash player has a much worse subjective visual experience. The proposed player has much less switches than the libdash due to the proposed smooth rate control mechanism. When the bufferred video time is between thresholds, the proposed adaptation algorithm keeps the video quality stable. It can be seen that after 600s the bitdash player requests much higher video quality than the available bandwidth, and then quickly drops. This abrupt video quality swithc is very annoying. While the proposed player performs well.

As it should be, there are no video playback interruptions in this constant network condition.

4.3. Impact of short-Term bandwidth variations

Here we present the results in the case that the available bandwidth goes through some positive or negative spikes that last for 3~6s. The available bandwidth state is shown in Fig.3 and the performance is shown in **Fig.4.**

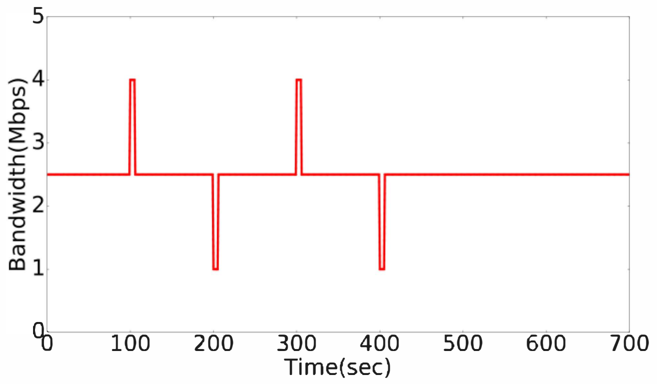

Fig.3. The available bandwidth of the short-Term variations

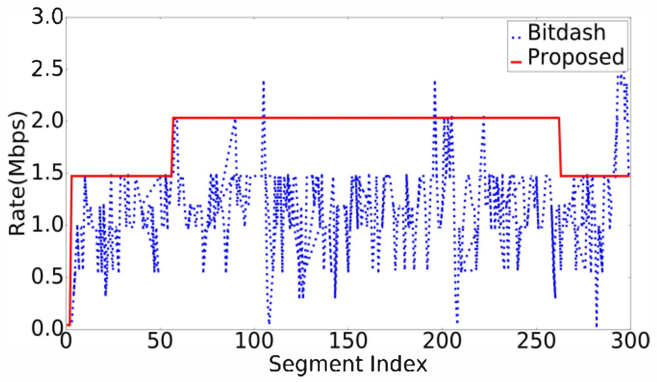

Fig. 4. The performance between proposed player and bitdash player on spike network on dataset BigBuckBunny 2s.

Table 3. Performance comparison under short-Term bandwidth variations

Column One	Bitdash	Proposed
Strat-up delay(ms)	-	94
Bitrate switch	298	3
Average bitrate(Mbps)	1.16	1.85
Bandwidth utilization(%)	46.4	74

It can be easily seen from **Fig.4** that the bitdash player changes the video rate immediately when the spikes come, while the proposed player can smooth the bandwidth spikes. The proposed player avoids bringing abrupt video quality changes to a great extent. And it can be concluded from Table.3 that the proposed player acquires much better bandwidth utilization than the bitdash player from Table.3.

4.4. Impact of long-Term bandwidth variations

Here we present the results in the case that the available bandwidth is square wave. The available bandwidth state is shown in Fig.6 and the performance is shown in **Fig.7**.

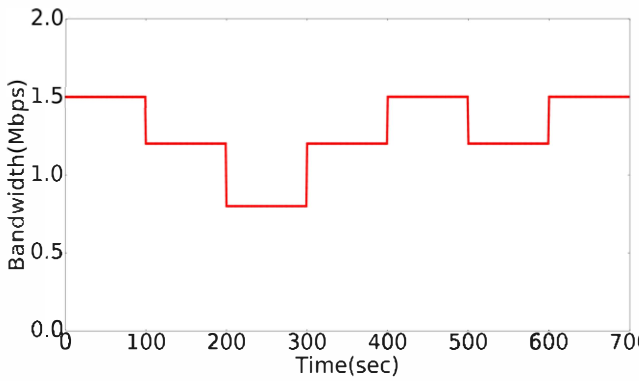

Fig.5. The available bandwidth of the long-Term variations

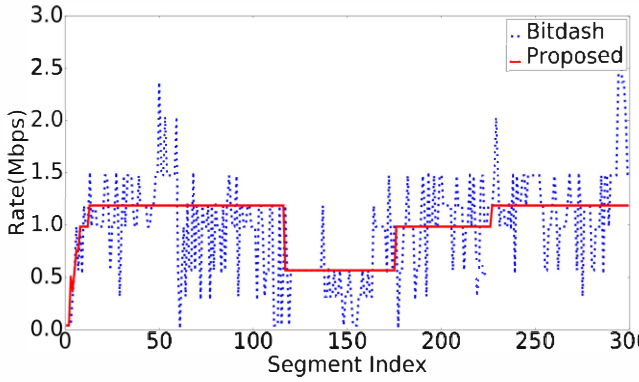

Fig. 6. The performance between proposed player and bitdash player on square ware network on dataset BigBuckBunny 2s.

Table 4. Performance comparison under long-Term bandwidth variations

Column One	Bitdash	Proposed
Strat-up delay(ms)	-	96
Bitrate switch	182	9
Average bitrate(Mbps)	0.98	1
Bandwidth utilization(%)	40.8	41.6

It can be seen in Fig.6 that the bitdash player drops the video quality to the lowest level when the available bandwidth drops. The bitdash player applies these changes to avoid video playback interruptions. Actually, for small bandwidth drops, the player should not decrease the video quality that much. It is shown in **Fig.6** that the proposed player can well adapt to the available bandwidth and decrease the video quality to a suitable level. At the same time, the proposed player has much less video quality switches. This is due to our buffer smooth rate algorithm. Less video quality switches brings less interruptions and thus achieves better visual experience.

5. SUMMARY

The proposed player with our adaptation algorithm works well in both constant and fluctuated network conditions and thus gets higher subjective score than the bitdash. We implement a sub-optimal algorithm to smooth the video playback. As a sacrifice, the proposed player does not have very high bandwidth utilization rate. In this way, our algorithm has a low complexity and can easily implemented in real-time video streaming. At the same time, the proposed player performs much better than the libdash player.

ACKNOWLEDGEMENTS

Thanks the previous work in [3] of Dr. Chao Zhou (zhouchaoyf@gmail.com) and his guide during our implementation of the proposed player. [3] is the best paper of VCIP 2015 and its extension has been published in TMM [6].

7. REFERENCES

[1] A. Begen, T. Akgul, and M. Baugher, “Watching video over the web: Part 1: Streaming protocols,” *IEEE Internet Comput.*, vol. 15, no. 2, pp. 54–63, Mar. 2011.
[2] T. Stockhammer, “Dynamic adaptive streaming over HTTP: Standards and design principles,” in *Proc. ACM Multimedia Syst.*, Feb. 2011, pp. 133–144.
[3] Zhou, Chao, C. W. Lin, and Z. Guo. "mDASH: A Markov Decision based Rate Adaptation Approach for Dynamic HTTP Streaming." *IEEE Transactions on Multimedia* (2016):1-1.
[4] Zhou, Chao, et al. "A Control-Theoretic Approach to Rate Adaption for DASH Over Multiple Content Distribution Servers." *IEEE Transactions on Circuits & Systems for Video Technology* 24.4(2014):681-694.
[5] Mok, Ricky K. P., et al. "QDASH: a QoE-aware DASH system."*Multimedia Systems Conference* ACM, 2012:11-22.
[6] Zhou, Chao, et al. " mDASH: A Markov Decision-Based Rate Adaptation Approach for Dynamic HTTP Streaming." *IEEE Transactions on Multimedia, 18.4(2016)*:738-751.
[7] Bitmovion, “bitdash.min.js,” available online: *http://bitmovion.com*
[8] Bitmovion, “libdash,” available online: *https://github.com/bitmovin/libdash*