A Proxy-assisted DASH Live Streaming Scheme

Cheng Zhao[1], Li Song[1,2] , Da Huo[1], Rong Xie[1,2] , Nam Ling[3]

[1] Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
[2] Cooperative Medianet Innovation Center, Shanghai China
[3] Department of Computer Engineering, Santa Clara University
zhaocheng100a@163.com, {song_li, rongxie}@sjtu.edu.cn, huodahaha@gmail.com, nling@scu.edu.cn

***Abstract*—Video adaptive streaming in mobile devices has always been a hard problem, and MPEG-DASH is considered to be a solution. DASH protocol requires the server to offer profiles of multiple bitrates at the same time. We can get a smoother playback as the profile number grows. However, we notice that the increase of profiles will bring in excessively frequent switches which will in turn cause a drop in QoE. Besides, growing numbers of media segments will put large burden on the backbone network. In this paper we use a proxy-assistant architecture to address above two problems. At the client side, we propose a new rate adaption method to ensure a smooth playback at the expense of a little average overall bandwidth. On the proxy server side, the smart proxy tries to aggregate similar requests with a map of importance of profiles for different bitrates. Compared with prior proxy based scheme, our scheme is proved to aggregate alike requests to reduce the cache hit ratio. The result shows that our method reduces the switching times and server requests times with the bitrate selecting by the proxy a little lower than the actual optimal one, and consequently makes our scheme both QoE-friendly and efficient under low-delay and multiple profiles scenario.**

Keywords—Dynamic Adaptive Streaming over HTTP, Low Latency, Rate Adaption Algorithm, Proxy, Lower Network Burden

I. INTRODUCTION

Rapid growth in mobile smart devices brings in prosperity in video services. Billions of data stream is transmitted through the Internet. However, mobile devices have been faced with critical network conditions compared with desktop devices. Traditional UDP-based streaming is restricted to a fixed bitrate that is not appropriate in varying network conditions. Moreover, UDP-based streaming is incompatible with firewalls and NAT. Therefore, new HTTP-based methods are proposed these years, making it possible to watch the video chips which have not been downloaded completely. In early period, dynamic adaptive streaming methods are developed by business corporations such as Apple, Microsoft and Adobe. But we must use the corresponding player towards the specified media files produced by these technologies. Since the standards are not compatible with each other, MPEG and 3GPP also released their Dynamic Adaptive Streaming over HTTP (DASH) as an international standard in late 2011. Similar to HTTP Live Streaming (HLS) presented by Apple, the workflow is to segment the original file, generating the chunks and corresponding media presentation description (MPD) that containing the information of segments.

A. Problem Statement

DASH protocol is driven by the client. The client side rate adaption method is the key component of the DASH protocol. The main challenge of rate adaption method is to accurately match the bandwidth conditions with given quality levels. It is obviously impossible for a client to access a strictly matched segment since that original server provides the video segments by given quantized quality levels, and this will cause oscillation between profiles of adjacent bitrate level at the client side. We simulate a DASH system to reveal how profile numbers affect the performance as the 1000-seconds result shown in TABLE I. We use a buffer-based rate adaption method depicted in [1]. The segment duration is set to 2s according to the requirements in the live scenario. In our experimental environment, there are six clients, a cache edge server and an original server in the network. We can measure the backbone network load by the requests received by the original server. The average variation in the table stands for the bitrate variation changing frequency during every switch time. Lower average variation means that bitrate difference between segments in the playback process is quite small, making the condition smoother. According to the experimental result, the increase of profiles will improve the performance of DASH protocol in average variation and overall average bitrate. The reason is that the video rendering will be much more accurate. Consequently, the playback will be smoother, which will increase the quality of experience (QoE) [2].

TABLE I. SIMULATION RESULT WITH DIFFERENT PROFILE NUMBERS

Profile Numbers	Average Bitrate (kbs)	Switches	Average Variation (kbs)	Server Requests
5	6010	125	2963	1196
12	6649	335	1714	1936

However the increase of profiles will bring in other problems. There is research indicating that frequent hopping from different profiles will bring in a drop in QoE [3]. In the case of Video on Demand (VOD) applications, the network fluctuations may be smoothed by the receiving buffer (usually

be 30s to 60s), thus the oscillation problem hasn't received much attention. While in the case of live streaming, the video latency has a close relationship with the buffer length. The switching frequency rises to an inacceptable level, because we need to make the buffer length shorten as far as possible. Besides, segment duration decreases sharply because of the end-to-end delay limitation in live cases. This sharp explosion of switch times will in turn cause a drop in QoE as our eyes are extremely sensitive. On the other hand, increase of profiles may put a large pressure on the backbone network as each profile serves fewer clients.

B. Related Work

In recent years, researchers have been concentrating on improving the performance of DASH in different aspects. The result in [4] shows that segment duration time accounts for most of the end-to-end delay in live cases. Thus, reducing the segment duration has been a common method in live DASH [5]. However, low latency rate adaption methods have not received much attention. Recently authors in [6] propose a low delay rate adaption algorithm, using a method to track the client receiving buffer, which may put extra pressure on the server. Except for the works focusing on low latency streaming, there are various methods to enhance the QoE performance. To deal with frequent switches, authors in [7] propose a rate adaption method considering several segments as one block. In [3] and [8], a new algorithm is to set a proxy and rewrite the HTTP requests, while clients remain unaware of the existence of proxy. The result shows that the proxy-assisted scheme promotes the QoE performance. The work mentioned above is of great assistance in putting forward our algorithm.

C. Contribution

In this paper, we assume a low delay live streaming scenario. We propose a rate adaption method to address the problem of frequent switching between chunks of different bitrate levels. Buffer-based method works well in the situation of VOD applications, which means longer segment duration time. But we really need to shorten the segmentation, since that our goal is to build a live system and low-delay attribute is especially necessary. Besides, to release the pressure of backbone network, we integrate a bandwidth probe and request aggregation strategy into edge server between public network and the numerous clients. The edge server is capable of aggregating similar requests by rewriting clients' requests. Based on the findings above, we use the proxy to neutralize the side effect on network efficiency. Compared with prior proxy based scheme in [3], our scheme is proved to aggregate alike requests to increase cache hit ratio. The result shows that our method reduces the switching times and server requests times with the bitrate selecting by the proxy a little lower than the actual measured throughput, and consequently makes our scheme both QoE friendly and efficient under scenario with low-delay and multiple profiles.

The rest of the paper is organized as follows: The proposed scheme will be discussed in Section II. In Section III, we will present the experiment result. Finally, the paper is concluded in Section IV.

II. The Proposed Scheme

A. Network Topology

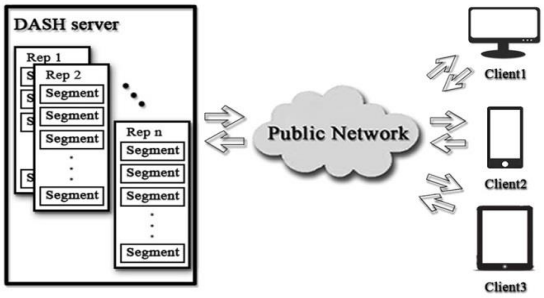

Fig. 1. Traditional DASH scenario

The traditional system architecture is shown in Fig. 1. The video content is encoded into several profiles with different bitrates, and then we divide them into segments. We put the segmented video pieces on the DASH server, which works like a regular web server. Then the client may request for segments of different quality under the corresponding bandwidth conditions. The number of bitrate levels of profiles is not specified in standard, but we need more profiles to match different bandwidth with various terminals at the same time and offer smoother playback as we have elaborated before. Thus, we add the edge server as the smart transfer station.

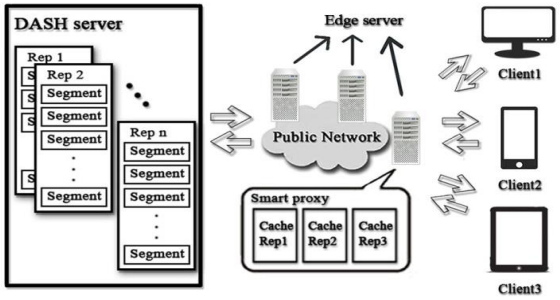

Fig. 2. Improved DASH scenario

According to our research, we find that the increase of profiles will bring in following possibilities:

1) Servers could estimate client status more accurately through specific HTTP GET requests;
2) Switches between different profiles may be annoying because of the slight distinction mentioned in [3];
3) Requires for a more accurate rate selection method;
4) It reduces the hit ratio of web caches;

Thus, we design a smart proxy application and deploy it on the existing architecture as shown in Fig. 2. The proxy aggregates similar requests by rewriting clients' requests. The client remains unaware of the existence of the proxy. Since there are more profiles in the server, the client will make more specific HTTP GET requests, through which, the proxy could probe the network condition, and thereby the proxy maintains a weight of importance map to describe each profile's importance and decides which to cache. The TABLE II. summarizes the important symbols that we use in the paper.

TABLE II. IMPORTANT SYMBOLS USED IN THIS PAPER

Client side symbols	
SFT	segment fetch time
$\widetilde{SFT}$	corrected segment fetch time
$r_{i+1\,MAX}$	maximum bitrate of next segment that client could access
r_i	bitrate of current segment
bl_i	buffer level (buffer occupation ratio) calculated by (cached video length/given buffer benchmark)
T	segment duration
C_1	correction coefficient
Proxy side symbols	
n	maximum cached items
I_{req}	index of requested profile
I_{next}	index of next requested profile
φ_I	weight of importance of *I*-th profile
$\Delta\varphi$	Increment of weight of importance
C_2	correction coefficient of variance
C_3	inclination factor
C_4	clients numbers correction coefficient
N	overall profiles
m	overall clients

B. Rate Adaption Algorithm in Client

There are two main categories in the rate adaption algorithm including bandwidth-based [9] and buffer-based methods [1]. Usually we could get a larger overall bandwidth throughput from buffer-based algorithm, while bandwidth–based algorithm reacts more directly to bandwidth fluctuation.

Since that receiving buffer and segment duration account for the most of the end-to-end delay time, we need to reduce both of them in live cases, which can be simulated as the drastically varying bandwidth condition. As a result, switching times of buffer-based algorithm will explode to an unacceptably level. We simulate a typical buffer-based rate adaption algorithm described in [1]. We set the segment time to 2 seconds and 10 seconds separately. The result is shown in Fig. 3. During the simulation time of 1000s, the switch frequency grows to 20.1 switches per minute from 4.6 switches per minute. The reason is that the buffer-based algorithm works like progressive downloading mechanism, which guarantees the bandwidth usage ratio to be a high level.

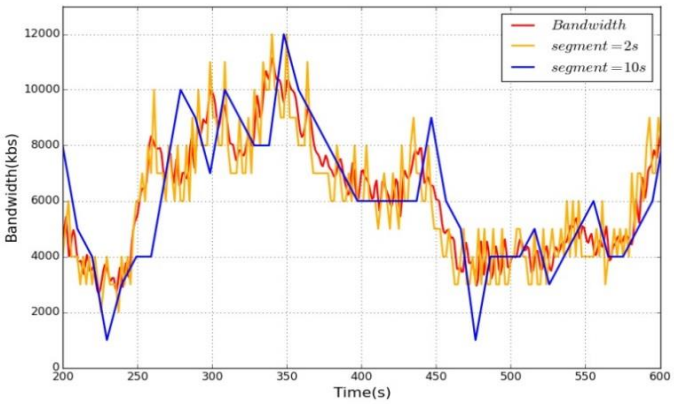

Fig. 3. The bandwidth-based method with different segment duration

While at the same time, buffer-based strategy will bring in excessive switches. The only factor that will cause the bitrate of video to raise or not is the level of buffer length. The client may choose segments with higher bitrate that may not match the bandwidth conditions well if there is enough space in the buffer. Then the quality of next requested chunk will fall down for granted, which is more evident under the condition of smaller segment time. There is a research indicating that the frequent change of video quality will get a drop in QoE evidently in [7].

In order to refine the explosion times of switches, we propose our improved rate adaption method. We use *SFT* to conduct process of switching up and buffer-based method to conduct the process of switching down. This will ensure that except for the signal that the buffer is mere empty, every event for switching up is conducted because of bandwidth improvement. Of course, behaviour of switching down is built under the guidance of buffer-based method. Additionally, to avoid getting the inaccurate bandwidth estimated under the condition of fluctuant network, we use corrected *SFT* to get a better estimation of bandwidth, where C_1 need to be regulated in different situations. The detailed procedure is presented in Algorithm 1 and the result will be discussed in the later chapter.

Algorithm 1 Client Rate Adaption Algorithm

1: **while** IsPlaying

2: $\widetilde{SFT} = SFT * C_1 + (1 - C_1) * \widetilde{SFT}_{previous}$

3: $r_{i+1\,MAX} =$
$$\begin{cases} r_i \times 0.3, & \text{if } 0.00 \le bl_i < 0.15 \\ r_i \times 0.5, & \text{if } 0.15 \le bl_i < 0.35 \\ r_i, & \text{if } 0.35 \le bl_i < 0.50 \\ r_i \times T/\widetilde{SFT}, & \text{if } 0.50 \le bl_i \end{cases}$$

4: Find the highest bitrate profile less than $r_{i+1\,MAX}$

5: $\widetilde{SFT}_{previous} = \widetilde{SFT}$

6: **end while**

C. Request Aggregation Algorithm in Proxy

As we have elaborated, the increase of profiles in different levels of bitrate will result in low cache hit ratio in the local cache server. This measure will make the load of the original server heavier, as well as the pressure of backbone network. Thus, we design the proxy to aggregate alike requests. A cache server will be deployed in the local gateway, which is unaware to the client. The original DASH server will store profiles of all bitrates, while the proxy will choose some of chunks called "important ones". We assume the clients served by the identical proxy share similar bandwidth undulation characteristics, so profiles with some bitrates are redundant. We maintain a weight of importance map to decide which profiles of bitrate level to cache. As mentioned in TABLE II. , we refer *n* as upper limit numbers that proxy could cache simultaneously. The parameter *n* needs to be regulated to reach a balance between network efficiency and performance. It is worthy to note that the lowest bitrate level should always be included to prevent interruption during the playback period. When a segment with certain bitrate

is requested, it is considerable that this bitrate is pretty valuable to reflect the condition of bandwidth in clients. We suppose that the possibility of profile index of next requested segment accords with normal distribution.

$$I_{next} \sim \mathrm{N}(I_{req}, \sigma) \tag{1}$$

We further assume that:

$$I_{req} - C_2 N \le I_{next} \le I_{req} + C_2 N \tag{2}$$

According to the 3σ principle in Statistics & Probability, we can give the variance of the normal distribution:

$$\sigma = \frac{C_2 N}{3} \tag{3}$$

We use the density function as an approximation to distribution function. The function to calculate $\Delta\varphi_I$ is as follows, where we use C_3 to make the weight of lower side of the bitrate table a little higher as a cushion for bandwidth drop; $C_4 e^{1/m}$ is set to balance the increment through different numbers of clients; u means the limitation if the bitrate of video segment is about to rise. The probable model of important points map is shown as Fig. 4.

$$\Delta\varphi_I = \begin{cases} \frac{C_4 e^{\frac{1}{m}}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(I-I_{req})^2}{2\sigma^2}\right) \times {C_3}^u & \text{if } u > 0 \\ \frac{C_4 e^{\frac{1}{m}}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(I-I_{req})^2}{2\sigma^2}\right) & \text{if } u \le 0 \end{cases} \tag{4}$$

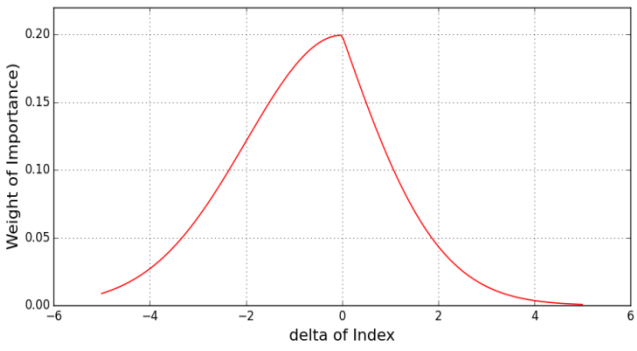

Fig. 4. Probability density for increment of importance for bitrate

Every time when a HTTP request is sent to the proxy, we add the corresponding increment to every bitrate level of profiles and then a normalization procedure is required. We mark the most *n* important profiles every duration time to be cacheable (using a priority queue). If the requested profile is marked, we handle the request as usual, otherwise we rewrite the request, and using this method we could merge approximate request, and consequently reduce the requests to original server. The detailed procedure is presented in Algorithm 2.

Algorithm 2 DASH proxy substitution algorithm

```
while IsPlaying
    if HTTP GET arrives
        for I in all bitrates of profiles
            φ_I = φ_I + Δφ_I
        end
        Normalize the weight table
        if I_req is not marked
            rewrite GET to the nearest lower profile
            which is marked in the weight table
        end if
        requested chunks return from DASH server
        if new segment produced
            mark the most important n profiles
        end if
    end if
end while
```

III. Experiment Result

A. Result of Rate Adaption Algorithm

We compare the proposed rate adaption method with the standard buffer-based method described in [1]. We set that C_1 equals to 0.3, T equals to 2s, and the overall profile number equals to 12. The result is shown in Fig. 5 and TABLE III. The proposed method is rigid to switch up and sensitive to switch down. At the same time, it reduces both the switching times and average variation with only little lower in bandwidth utilization, which will promote the QoE performance.

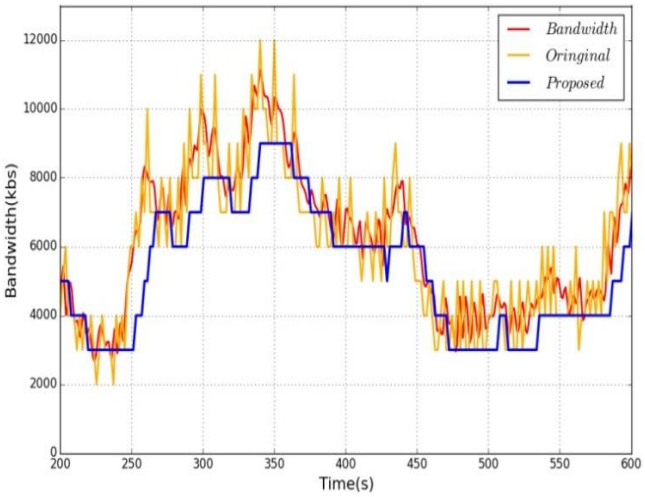

Fig. 5. Comparison between the proposed rate adaption algorithm and the typical buffer-based rate adaption algorithm

TABLE III. Comparison Between Original and Proposed Method Result

Group	Average Bitrate(kbs)	Switches	Average Variation (kbs)
Original	6542	320	1690
Proposed	6005	75	1013

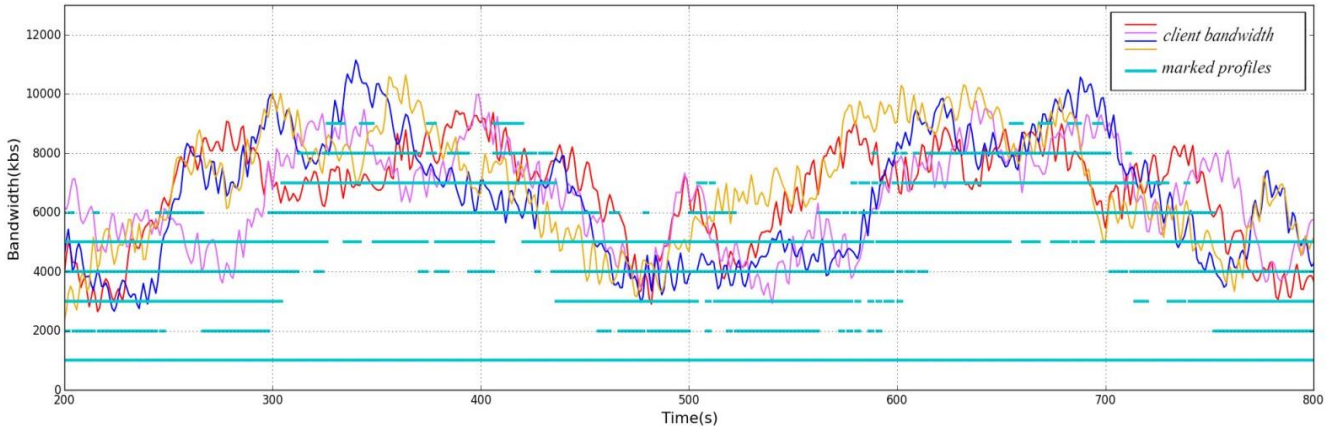

Fig. 6. The profiles of different bitrates in proxy cache for clients in different bandwidth conditions

B. *Result of Request Aggregation Algorithm*

According to the result of our experiment, a larger C_2 is required in an unstable network condition, and C_3 needs to be turned up when bandwidth drops occur frequently while C_4 is always set to be 0.65. In our simulation, we set C_2 equals to 0.5, C_3 equals to 0.6 and n equals to 5. The simulation result is shown in Fig. 6, where we show how proxy works with a group of clients. At one certain time point, the proxy server will request profiles of different levels as there are numbers of clients under different bandwidth conditions. The cyan colour line represents the marked profiles. The method will fit the scenario when the clients share similar bandwidth fluctuation. Additionally, it is possible to use the prediction result to feedback controlling DASH server encoder when the network suffers from regular long-term fluctuations.

C. *Overall Result*

Finally, we make a comparison between the proposed scheme and the original one. We simulate a simple DASH system, and the result for 1000 seconds is shown in TABLE III. We use a buffer-based rate adaption method depicted in [1]. The segment duration is set to 2s. There are total six clients, one cache edge server and an original server in the network. The system uses the refined rate adaption method and proxy-assisted architecture. The result is shown in TABLE IV. It indicates that the proposed scheme offers a smoother playback. And we could see the switching times becoming less than the baseline group. The decrease of server requests times means reducing the load of network.

TABLE IV. SIMULATION RESULT WITH SMART PROXY

Group	**Average Bitrate (kbs)**	**Switches**	**Average Variation (kbs)**	**Server Requests**
Baseline	6649	335	1714	1936
Proposed	5671	95	1192	1365

IV. CONCLUSIONS

The proxy-assisted scheme consists of rate adaption method and request aggregation strategy. The result shows that our method reduces the switching times and server requests times with the bitrate selecting by the proxy a little lower than the actual optimal one, and consequently makes our scheme to be both QoE friendly and efficient under low-delay and multiple profiles scenario.

ACKNOWLEDGMENT

This work was supported by NSFC (61521062, 61527804, 61420106008), the 111 Project (B07022 and Sheitc No.150633) and the Shanghai Key Laboratory of Digital Media Processing and Transmissions.

REFERENCES

[1] C. Müller, S. Lederer, C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in Proceedings of ACM the 4th Workshop on Mobile Video, pp. 37-42, 2012.

[2] T C. Thang, H T . Le, A T. Pham, et al. "An evaluation of bitrate adaptation methods for HTTP live streaming," IEEE Journal on, Selected Areas in Communications, vol. 32, no.4, pp. 693-705, 2014.

[3] R K P. Mok, X. Luo, E W W. Chan, et al. "QDASH: a QoE-aware DASH system," in Proceedings of ACM the 3rd Multimedia Systems Conference, pp. 11-22, 2012.

[4] T. Lohmar, T. Einarsson, P. Fröjdh, et al. "Dynamic adaptive HTTP streaming of live content," in Proceedings of World of IEEE International Symposium Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1-8, 2011.

[5] V. Swaminathan, S. Wei. "Low latency live video streaming using HTTP chunked encoding," in Proceedings of IEEE 13th International Workshop Multimedia Signal Processing (MMSP), pp. 1-6, 2011.

[6] Y. Shuai, M. Gorius, T. Herfet. "Low-latency dynamic adaptive video streaming," in Proceedings of 2014 IEEE International Symposium Broadband Multimedia Systems and Broadcasting (BMSB) pp.1-6, 2014.

[7] C. Zhou, X. Zhang, L. Huo, et al. "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," in Proceedings of IEEE Visual Communications and Image Processing (VCIP), pp. 1-6, 2012.

[8] A. El Essaili, D. Schroeder, D. Staehle, et al. "Quality-of-experience driven adaptive HTTP media delivery," in Proceedings of IEEE International Conference on Communications (ICC), pp. 2480-2485, 2013

[9] C. Liu, I. Bouazizi, M. Gabbouj. "Rate adaptation for adaptive HTTP streaming," in Proceedings of ACM the second annual ACM conference on Multimedia systems. pp. 169-174, 2011.