LETTER

A Probabilistic Adaptation Method for HTTP Low-Delay Live Streaming over Mobile Networks

Hung T. LE[†], **Nam PHAM NGOC**[††], ***Nonmembers***, **Anh T. PHAM**[†], ***and*** **Truong Cong THANG**[†a)], ***Members***

SUMMARY The study focuses on the adaptation problem for HTTP low-delay live streaming over mobile networks. In this context, the client's small buffer could be easily underflown due to throughput variations. To maintain seamless streaming, we present a probabilistic approach to adaptively decide the bitrate for each video segment by taking into account the instant buffer level. The experimental results show that the proposed method can significantly reduce buffer underflows while providing high video bitrates.

key words: *HTTP streaming, adaptivity, low-delay, DASH*

1. Introduction

HTTP adaptive streaming (HAS) has become a new trend of multimedia delivery over mobile IP networks [1], [2]. Recently, the Moving Picture Experts Group (MPEG) has developed the Dynamic Adaptive Streaming over HTTP (DASH) standard to enable interoperability in the industry [1], [3]. In HAS, a video at a server is encoded at multiple video versions (with different bitrates), each of which is further divided into short segments. In practice, segment durations are typically from 2s to 10s. During a streaming session, an adaptation method deployed at a client selects an appropriate bitrate version for each segment based on the status of the network/client. Then, each segment is delivered by an HTTP request-response transaction. To cope with network variations, many adaptation methods have been proposed, which can be divided into a throughput-based group and a buffer-based group [2], [4]. It should be noted that due to bitrate adaptation, buffer size and buffer level in adaptive streaming are measured in time units [1], [2], [4].

For on-demand streaming, adaptation methods can leverage a large client buffer size to cope with throughput fluctuations. However, for live streaming, the buffer size is limited. For example, suppose at the start time of a live video service, the client will load L segments into the buffer before playing out. Because the client can download only the segments that have already been generated in real time, the client has to spend L segment durations for initial buffering. In live streaming, the total duration of segments downloaded in initial buffering is equal to the buffer size [4]. After initial buffering, if the download rate and the playout rate are equal, the buffer level will be stable at the value of buffer size; this value is also referred to as the target buffer level in this letter. More detailed explanation about the buffer size and the buffer level in live streaming can be found in [4].

Because low delay is a crucial requirement for live streaming [5], the buffer size should be reduced in this context. As analysed in [6], the smallest buffer size for live streaming is expected to be 2 segments. In [2], the buffer size for live streaming scenario is set to 8s (or 4 segments), which is much lower than the typical buffer size of 30s (or 15 segments) for on-demand scenario. In [7], a heuristic adaptation method for live streaming is proposed, but this method is only designed for buffers of three or more segments. Moreover, the method has some thresholds which are not easy to set. A learning-based adaptation method for low-delay streaming is proposed in [8]; however, the lowest supported buffer size is still 3 segments. In [4], through an extensive evaluation, it is found that throughput-based methods are more effective to support small buffer sizes than buffer-based methods. However, when the buffer size is reduced to 2 segments, no adaptation methods evaluated in [4] could guarantee a continuous session under strong variations of a mobile connection. Note that, besides the average video bitrate, interruptions strongly affect the quality of experience [9].

In this study, we propose a probabilistic adaptation method for HTTP low-delay live streaming, where the buffer size could be just two segments. We first formulate the adaptation problem for HTTP live streaming, taking into account the instant buffer level. Then, a solution to that problem formulation is presented that decides the bitrate for each segment. Through experiments, we show that the proposed method can effectively cope with throughput fluctuations and significantly reduce the chance of interruptions.

2. Adaptation Problem Formulation

Suppose that, at the server, a video is provided at a number of bitrate versions, each is chopped into segments of τ seconds. For each segment n ($n = 1, 2, \ldots$) during a streaming session, the client selects bitrate B_n from the available bitrate options.

In the initial buffering stage, the client requests the lowest bitrate for L segments to reach the target buffer level $\beta^{tar} = L \times \tau$ seconds [1]. After that, the client switches to the steady stage, where it receives and plays video segments simultaneously. In this study, our focus is on the adaptation in

Manuscript received August 19, 2016.
Manuscript publicized November 9, 2016.

[†]The authors are with University of Aizu, Aizuwakamatsu-shi, 965–8580 Japan.
[††]The author is with Hanoi University of Science and Technology, Hanoi, Vietnam.
a) E-mail: thang@u-aizu.ac.jp

DOI: 10.1587/transinf.2016EDL8172

the steady stage. During this period, if an interruption happens, the client switches back to the initial buffering state.

Denote T_{n-1} the measured throughput of the last segment $n-1 (n > L)$, which is used as the throughput estimate for the next segment n. In this study, the client uses a safety margin $\gamma_n \in [0, 1)$ to compute bitrate B_n:

$$B_n = T_{n-1} \times (1 - \gamma_n). \tag{1}$$

We denote β^r_n the buffer level measured at t^r_n, which is right after segment n is fully received. Because the buffer size in live streaming is small, the client should select the highest possible bitrate while the buffer is still not affected. For that purpose, we need to find the minimum margin γ_n so that the probability that the resulting buffer level β^r_n is lower than the target buffer level β^{tar} is smaller than a desired constraint.

Denote $\Pr\left(\beta^r_n < \beta^{tar} | \gamma_n\right)$ the probability that β^r_n is smaller than β^{tar} given γ_n. The adaptation problem is defined as:

$$\text{minimize} \quad \gamma_n \tag{2}$$

$$\text{subject to} \quad \Pr\left(\beta^r_n < \beta^{tar} | \gamma_n\right) < \epsilon, \tag{3}$$

where ϵ is a desired probability constraint. Intuitively, the smaller ϵ is, the lower the risk of buffer instability will become, and also the lower the bitrate will be.

3. Adaptation Solution

In this section, we present our proposed method in detail. Figure 1 shows an illustration of the request times and buffer behavior of the client at segments $n-1$, n, and $n+1$. Let t^s_n and β^s_n be the time and the buffer level when the client sends the request of segment n, respectively. In live streaming, the client should periodically send requests with a distance of τ seconds. So, in an ideal condition, segment n is requested at time t'_n (i.e., $t^s_n = t'_n$), which is computed by

$$t'_n = \theta + (n-1) \times \tau, \tag{4}$$

where θ is the request time of the first segment. The time t'_n is considered as the earliest time, at which the client can send the request of segment n.

At time t'_{L+1} (i.e. when the play-out is started), β^s_{L+1} is β^{tar}. Therefore, in the steady stage, if the last segment $n-1$ has been fully received before t'_n, the buffer level β^s_n at time $t^s_n = t'_n$ is also β^{tar} (Fig. 1).

Denote t^r_{n-1} the time right after the client has fully received the last segment $n-1$. If $t^r_{n-1} < t'_n$, the client has to wait for an interval, which is $\Delta t^r_{n-1} = t'_n - t^r_{n-1}$, before sending the next request at t'_n (as illustrated in Fig. 1). However, because of throughput variations, the client may complete receiving the last segment $n-1$ after time t'_n (i.e., $t^r_{n-1} \geq t'_n$). In the case where $t'_n \leq t^r_{n-1} \leq t'_n + \beta^{tar}$, the buffer level is reduced bellow the target level β^{tar} and so, the client should send the request immediately at time t^r_{n-1} to quickly increase the buffer level. Finally, if $t^r_{n-1} > t'_n + \beta^{tar}$ (i.e., the buffer has

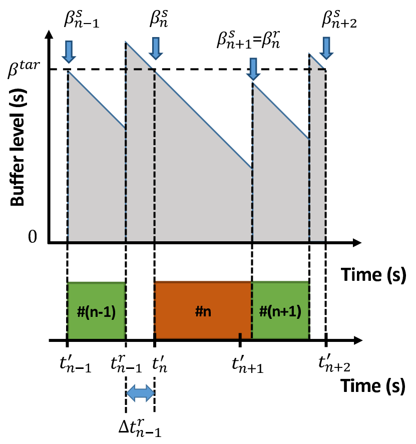

Fig. 1 Illustration of the request times and buffer behavior of the client at segments $n-1$, n, and $n+1$.

been depleted and the recently received segment $n-1$ has missed its deadline), the client ignores segment $n-1$ and switches back to the initial buffering stage.

So, if no buffer underflow is observed, the actual request time for segment n is

$$t^s_n = \max\{t'_n, t^r_{n-1}\}, \tag{5}$$

and the buffer level at that time is

$$\beta^s_n = \begin{cases} \beta^{tar} & \text{if } t^r_{n-1} < t'_n, \\ \beta^r_{n-1} & \text{if } t'_n \leq t^r_{n-1} \leq t'_n + \beta^{tar}. \end{cases} \tag{6}$$

The next segment n, which contains τ seconds of media, will be requested with video bitrate B_n. Therefore, the delivery duration of this segment, denoted by d_n, is

$$d_n = \frac{\tau \times B_n}{T_n} = \tau \times (1 - \gamma_n) \times \frac{T_{n-1}}{T_n}, \tag{7}$$

where T_n is the throughput of segment n. Obviously, the contribution of a new segment to the buffer is τ seconds. So, the resulting buffer level β^r_n right after receiving segment n is given by

$$\beta^r_n = \beta^s_n - d_n + \tau. \tag{8}$$

Using (7) and (8), we can rewrite the constraint (3) as:

$$\Pr\left(\beta^s_n - \tau \times (1 - \gamma_n) \times \frac{T_{n-1}}{T_n} + \tau < \beta^{tar}\right) < \epsilon. \tag{9}$$

Rearranging (9), we have

$$\Pr\left(\frac{T_{n-1}}{T_n} > x_n\right) < \epsilon, \tag{10}$$

where x_n, called the *throughput ratio constraint* for segment n, is calculated by

$$x_n = \frac{\beta^s_n + \tau - \beta^{tar}}{\tau \times (1 - \gamma_n)}. \tag{11}$$

Let X be a random variable that represents the ratio $\frac{T_{n-1}}{T_n}$, and $F_X(\cdot)$ be the cumulative distribution function

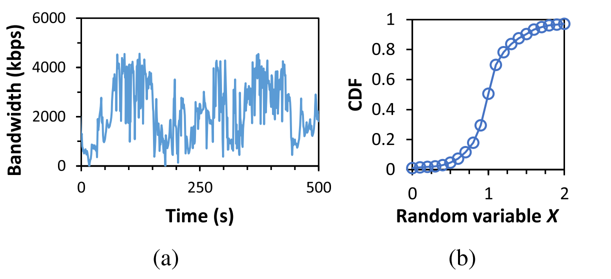

Fig. 2 Illustration of (a) the bandwidth trace and (b) the CDF of variable X, obtained by the observation process.

(CDF) of X. So, the condition (10) is converted into

$$F_X(x_n) > 1 - \epsilon. \quad (12)$$

In our method, the CDF of random variable X is obtained by using the throughput history of the client. Specifically, we use an independent process, called *observation process*, in which the CDF of random variable X is updated every $2s$. Figure 2b provides an illustration of the CDF of random variable X corresponding to a bandwidth trace (Fig. 2a).

Based on the obtained CDF, the client selects the minimum ratio x_n^* that meets condition (12), and then decides margin γ_n based on (11) as follows:

$$\gamma_n = 1 - \frac{\beta_n^s + \tau - \beta^{tar}}{\tau \times x_n^*}. \quad (13)$$

The general procedure to select the bitrate for segment n in the steady stage can be summarized as follows:

1) Measure the current buffer level. If the buffer is depleted, switch back to the initial buffering stage; otherwise, compute the throughput T_{n-1} of the last segment $n-1$ as in [1].
2) Determine the time t_n^s to send the next request using (5).
3) Given the CDF of random variable X, select the minimum x_n^* following condition (12); then compute the margin γ_n following (13).
4) Based on γ_n and T_{n-1}, decide the bitrate for the next segment following (1). If no bitrate is found, the minimum bitrate option is used.
5) Send the request to the server at time t_n^s.
6) Repeat step 1 until the end of the session.

4. Experiments

Our test-bed is based on that of [4]. At the server, we employ a popular DASH dataset, where the video content is provided in constant bitrate (CBR) mode at 17 bitrate versions, from 100kbps to 6000kbps [11]. All segments have the same duration of $\tau = 2s$. At the client side, we implement our proposed method (called probability based (PB) method) with $\epsilon = 0.25$. The target buffer level is set to $4s$ (i.e., $L = 2$). For comparison, two reference methods, the instant throughput based (ITB) method [4] and the conservative throughput based (CTB) method [12], are employed.

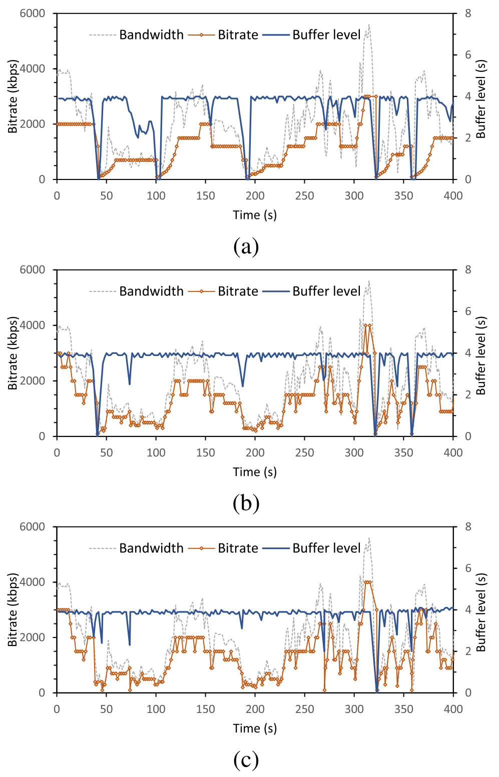

Fig. 3 The bandwidth trace and adaptation results of (a) the CTB method, (b) the ITB method, and (c) the PB method.

The former method uses a fixed margin of 0.2 while the latter method uses a mechanism like TCP congestion control, rather than a fixed margin.

In our first experiment, we use two bandwidth traces. The first trace (Fig. 2 a) is used as the bandwidth history data of a past session to compute the initial CDF of random variable X. The second trace (Fig. 3 a) that is obtained from the same mobile network as the first one, is used to evaluate the proposed method. It should be noted that, at time 325s, the bandwidth drops to nearly zero, so any experiment run will mostly have one buffer underflow at this point. Figure 3 shows the bitrate curves and buffer level curves of the three methods. We observe that the CTB method is conservative (aggressive) in increasing (decreasing) the bitrate. Despite this behavior, the CTB method results in as many as five buffer underflows (or interruptions). The bitrate curves of the ITB and PB methods are quite similar. However, the ITB method has three interruptions, whereas the PB method has only one, which is unavoidable at time 325s.

For adaptation statistics, the three adaptation methods are tested with multiple experimental runs, using a full bandwidth trace in Fig. 4 (obtained in [10]). For each adaptation method, 15 experimental runs are recorded, each of which is 400s long and has a random starting point. In this experiment, the PB method is employed with ϵ being 0.35, 0.25, and 0.15; these three options are respectively referred to as PB-35, PB-25 and PB-15. All other settings are the same as before. Table 1 provides statistics of adaptation results,

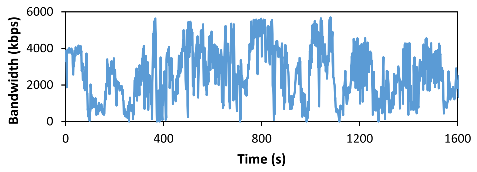

Fig. 4 The bandwidth trace of a mobile network [10].

Table 1 Average statistics of the adaptation methods.

Statistics	CBT	ITB	PB-35	PB-25	PB-15
Average bitrate (kbps)	1678	1865	2043	1895	1661
Number of interruptions	3.73	1.80	1.47	1.20	0.87
Total duration of interruptions (s)	15.1	7.6	6.3	5.1	3.7

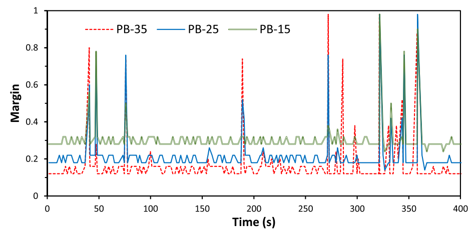

Fig. 5 Instant value of the margin for ϵ = 0.35, 0.25, 0.15.

where each item is the average value from the 15 experimental runs. The statistics show that the CTB method has a low average bitrate and the highest number of interruptions. Compared to the ITB method, the PB-35 and PB-25 options have fewer interruptions and better average bitrates. With the PB-15 option, although providing a somewhat lower average bitrate than that of the ITB method, it significantly reduces the number of interruptions (to 0.87) and the total duration of interruptions (to 3.7s).

In the proposed method, parameter ϵ, together with the current buffer status, actually affects the margin value. Figure 5 shows the instant value of the margin when the bandwidth trace in Fig. 3a is used. It is seen that the margin value is initially decided to be 0.18 when $\epsilon = 0.25$. During the session, the margin value varies and so the bitrate is adapted accordingly. For example, at time 40s, when the buffer level is drastically reduced due to a throughput drop, the margin is increased to 0.56. When the requirement for buffer stability is increased (i.e., ϵ is reduced), the margin is accordingly increased in an automatic manner. For example, when $\epsilon = 0.15$, the starting value of the margin is decided to be 0.28. This means, parameter ϵ can be used to control the tradeoff between video bitrate and buffer stability in a live streaming session.

It can be seen that the effectiveness of the proposed method is provided by its two aspects. First, based on the recent throughput history and the probabilistic buffer constraint (ϵ), we can decide the starting value of the margin and then adjust it during a session. In the ITB method, there is no way to decide the (fixed) margin value in advance. That means we cannot know, given some bandwidth trace, which margin will be a good one. Second, our method also considers the current buffer level. When the buffer level is dropped, the margin will be increased to avoid buffer underflows. As we try to keep the buffer level as stable as possible, there are not many times the buffer level is dropped (by unexpected throughput fluctuations). This is the reason why the margin value is mostly stable and is increased only at times of buffer level drop.

5. Conclusions

In this letter, we have considered the adaptation problem of HTTP live streaming over mobile networks. A probability based method was proposed that adaptively changes the bitrate to avoid buffer underflows. The experimental results with a buffer size of only two segment durations showed that the proposed method can significantly reduce the chance of buffer underflows while providing high video bitrate.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Number 15K00134 and the competitive funds of the University of Aizu.

References

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

[2] S. Akhshabi, S. Narayanaswamy, A.C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," Signal Processing: Image Communication, vol.27, no.4, pp.271–287, Apr. 2012.

[3] ISO/IEC IS 23009-1, "Information technology - dynamic adaptive streaming over HTTP (DASH) – part 1: Media presentation description and segment formats," 2012.

[4] T.C. Thang, H.T. Le, A.T. Pham, and Y.M. Ro, "An evaluation of bitrate adaptation methods for HTTP live streaming," IEEE J. Sel. Areas Commun., vol.32, no.4, pp.693–705, Apr. 2014.

[5] R. Kooji, K. Ahmed, and K. Brunnstrom, "Perceived quality of channel zapping," Proc. 5th Int. Conf. Commun. Syst. and Netw. (CSN'06), pp.155–158, Aug. 2006.

[6] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann, "Dynamic adaptive http streaming of live content," Proc. IEEE Int. Symp. on a World of Wireless, Mobile and Multimedia Netw., pp.1–8, Jun. 2011.

[7] S. Benno, A. Beck, J.O. Esteban, L. Wu, and R. Miller, "Wilo: A rate determination algorithm for has video in wireless networks and low-delay applications," Proc. IEEE Globecom Workshop, Atlanta, GA, pp.512–518, Dec. 2013.

[8] J.V. Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," Proc. Int. Symp. on Integrated Netw. Man. (IM'15), pp.131–138, May 2015.

[9] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," IEEE Commun. Surveys Tuts., vol.17, no.1, pp.469–492, 2015.

[10] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic

adaptive streaming over HTTP in vehicular environments," Proc. 4th Workshop on Mobile Video (MoVid '12), pp.37–42, 2012.

[11] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," Proc. 3rd ACM Conf. on Multimedia Syst. (MMSys '12), pp.89–94, Feb. 2012.

[12] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," Proc. 2nd ACM Conf. on Multimedia Syst. (MMSys'11), pp.169–174, Feb. 2011.