Request Adaptation for Adaptive Streaming over HTTP/2

Duc V. Nguyen¹, Hung T. Le¹, Pham Ngoc Nam¹², Anh T. Pham¹ and Truong Cong Thang¹
¹University of Aizu, Aizuwakamatsu, Japan
²Hanoi University of Science and Technology, Hanoi, Vietnam

Abstract -- In this paper, we propose a client-based adaptation method for Adaptive Streaming over HTTP/2 leveraging the server push feature. A cost function that takes into account both the number of pushed segments and the playout buffer level is defined. Experiment results show that the proposed method can improve the tradeoff between the number of requests and buffer stability compared to existing solutions.

I. INTRODUCTION

HTTP Adaptive Streaming (HAS) has emerged as a dominant technology for streaming video over the Internet [1]. In the adaptation problem of HAS to throughput variations, deciding an appropriate segment duration is still a challenging problem. Existing methods use fixed-length video segments with the duration being from 2s to 10s. Obviously, the use of a long segment duration would benefit from fewer requests, and therefore less overhead. However, a long segment duration results in client’s slow reaction to changes in network condition since adaptation is usually performed at segment boundaries. Although this may work well under stable network conditions, significant fluctuations in network throughput would frequently cause playback interruptions that degrades user’s Quality of Experience (QoE). To improve client’s ability to adapt to changes in networks, one can use short segment duration. Yet, this requires more requests, and so more overhead.

The recently ratified HTTP/2 standard [2] provides a new feature called server push that can help break the tradeoff between segment duration and the number of requests. Specifically, server push feature allows a server to push a resource to the client without the client requesting it. Therefore, this enables the use of short segment duration without increasing the required number of requests by asking the server to respond a request with multiple consecutive segments of the same version. In [3], the authors have investigated a low latency live video streaming technique utilizing server push feature in which a fixed number of segments is pushed to the client upon a client’s request. However, as shown later, their method results in very poor tradeoff between request overhead and buffer stability under varying bandwidth conditions (e.g. mobile networks).

In this paper, we propose a novel adaptation method for Adaptive Streaming client over HTTP/2 leveraging the server push feature under varying bandwidth conditions. Specifically, the proposed method can dynamically adapt the number of pushed segments according to current network and buffer conditions. To find the optimal number of pushed/requested segments for a request, we define a cost function depending on two factors, which are request cost and buffer cost. Experiment results show that the proposed method is able to improve the tradeoff between number of requests and buffer stability in comparison to the use of a fixed number of pushed segments as in [3].

II. PROPOSED METHOD

The goal of our request adaptation algorithm is to decide the video bitrate and the number of pushed segments of each request of the client, so as to have the best tradeoff between the number of requests and buffer stability. Let \( R_j \) be the bitrate of version \( j \) (1≤j≤M) where \( M \) is the number of available content versions, and \( SD \) be the segment duration. Denote \( B \) the resulting playout buffer level and \( C \) the cost given a decision of selecting version \( j \) and the number of pushed segments \( N \). Given available network throughput \( R^c \), current buffer level \( B_{curr} \) and minimum buffer constraint \( B_{min} \), the adaptation problem can be formulated as follows.

Find version \( j \) (1≤j≤M) and the number of pushed segments \( N \) of the next request so as to minimize the resulting cost \( C \) which is a function of the number of pushed segment \( N \) and the current buffer level \( B_{curr} \)

\[
C = f(N, B_{curr})
\]

subject to:

\[
B \geq B_{min}.
\]

To estimate the available network throughput \( R^c \), we adopt the method proposed in our previous work [1]. In this paper, the bitrate \( R_j \) is decided as the highest bitrate that is lower than the estimated throughput as follows.

\[
R_j = \max \{ R_i | R_i \leq (1 - \delta)R^c, 1 \leq i \leq M \}
\]

where \( \delta (0 \leq \delta \leq 1) \) is a safety margin. Given the estimated throughput \( R^c \) and the selected video bitrate \( R_j \), the value of \( B \) can be estimated as follows.

\[
B = B_{curr} + SD \times N \times \left(1 - \frac{R_j}{R^c} \right).
\]
the buffer cost $C_B$ as follows.

$$C = \alpha \times C_B + (1-\alpha) \times C_R$$  \hspace{1cm} (5)$$

The buffer cost $C_B$ is computed as a linearly decreasing function of the number of pushed segments $N$ as follows.

$$C_B = \frac{1}{N}.$$  \hspace{1cm} (6)$$

The buffer cost $C_B$ is computed based on the following observations. First, a high value of $N$ would increase the risk that the buffer level drops below $B_{\text{min}}$ under a throughput decreasing interval. Second, a high buffer level can help the client tolerate throughput drops for long intervals. Consequently, the buffer cost $C_B$ is computed as follows.

$$C_B = \frac{N \times SD}{B_{\text{curr}} - B_{\text{min}}}.$$  \hspace{1cm} (7)$$

Note that, (7) is used only when $B_{\text{curr}}$ is larger than $B_{\text{min}}$. When $B_{\text{min}}$ becomes smaller than $B_{\text{curr}}$, the value of $N$ is set to one as long as network throughput is decreasing to avoid unnecessary reduction in playout buffer. After that, the value of $N$ is increased by one every two requests until playout buffer level attains the buffer constraint $B_{\text{min}}$.

As the problem space here is small, we apply a full-search procedure to solve the adaptation problem as follows. Cost $C$ corresponding to each of available values of $N$ is first computed using (5)(6)(7). Then, the one with minimum value of $C$ and satisfies condition (2) is selected as the number of requested/pushed segments for the next request.

III. Experiment Results

The test-bed of our experiments is similar to that of [1]. Video content is provided at 15 versions with bitrate ranging from 200kbps to 3000kbps with a step size of 200kbps. The content is composed of 1000 segments with segment duration of 1s. The content is hosted on an HTTP/2-enabled Web Server. The maximum number of pushed segments per request is set to 4. To inform the server the number of segments to be pushed, the client includes the value of $N$ into the request sent to the server using the so-called query string. The playout buffer size is set to 10s, and the buffer constraint $B_{\text{min}}$ is set to 6s. The safety margin $\delta$ is set to 0.1. DummyNet [4] is installed on the client machine to emulate characteristics of network path between the client and the server in which Round-Trip Time (RTT) is set to 50ms. Available network throughput is emulated using a real bandwidth trace obtained from a mobile network in [5] as shown in Fig. 1.

The proposed method is compared to the method of [3] in which the server pushes a fixed number of segments given a client request. Table I shows the number of requests, average video bitrate, minimum buffer level, and percentage that buffer level $B$ is lower than $B_{\text{min}}$, i.e. $p(B < B_{\text{min}})$. For the method of [3], the results of four cases, which are $N=1$, $N=2$, $N=3$, and $N=4$, are shown. For the proposed method, two results corresponding to two values of the coefficient $\alpha$, which are $\alpha=0.4$ and $\alpha=0.6$, are shown. It can be seen that the use of a fixed number of pushed segments results in very poor tradeoff between the number of requests and buffer stability. When $N=1$, it is possible to achieve a very stable buffer with the minimum buffer level of 7.06s and no buffer constraint violation. However, the client must send 1000 requests in this case. On the other hand, the number of requests can be as low as 250 (when $N=4$) but the resulting buffer is highly fluctuating in which playback interruption has been observed once.

Meanwhile, the proposed method can provide much better tradeoff between the number of requests and buffer stability. Specifically, with $\alpha=0.6$, our method is able to maintain playout buffer level as stable as $N=1$ case while the number of requests is almost halved (i.e. 518). With $\alpha=0.4$, in comparison to $N=3$ case, the proposed method can increase the minimum buffer level and reduce $p(B < B_{\text{min}})$ while the number of requests is just a little higher (i.e. 383 vs 334). It should be noted that the average bitrate of our method is always comparable to or higher than those of method [3]. These results imply that our method can smartly change the number of segments in different requests according to network fluctuations. Also, by adjusting the value of the coefficient $\alpha$, the proposed method can adjust the tradeoff between the number of requests and buffer stability.

![Fig. 1. The used bandwidth trace.](image)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Method of [3]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$N=1$</td>
<td>$N=2$</td>
</tr>
<tr>
<td>Number of request</td>
<td>1000</td>
<td>500</td>
</tr>
<tr>
<td>Avg. bitrate (kbps)</td>
<td>1609</td>
<td>1647</td>
</tr>
<tr>
<td>Min. buffer (s)</td>
<td>7.06</td>
<td>4.39</td>
</tr>
<tr>
<td>$p(B &lt; B_{\text{min}})$ (%)</td>
<td>0</td>
<td>0.4</td>
</tr>
</tbody>
</table>
REFERENCES


