Queueing Theoretic Approach to Playout Buffer Model for HTTP Adaptive Streaming

Jiwoo Park and Kwangsue Chung
Department of Electronics and Communications Engineering, Kwangwoon University
20, Kwnagwoon-ro, Nowon-gu, Seoul, Korea
[e-mail: jwpark@cclab.kw.ac.kr, kchung@kw.ac.kr]
*Corresponding author: Kwangsue Chung

Received September 11, 2017; revised February 14, 2018; accepted April 9, 2018; published August 31, 2018

Abstract

HTTP-based adaptive streaming (HAS) has recently been widely deployed on the Internet. In the HAS system, a video content is encoded at multiple bitrates and the encoded video content is segmented into small parts of fixed durations. The HAS client requests a video segment and stores it in the playout buffer. The rate adaptation algorithm employed in HAS clients dynamically determines the video bitrate depending on the time-varying bandwidth. Many studies have shown that an efficient rate adaptation algorithm is critical to ensuring quality-of-experience in HAS systems. However, existing algorithms have problems estimating the network bandwidth because bandwidth estimation is performed on the client-side application stack. Without the help of transport layer protocols, it is difficult to achieve accurate bandwidth estimation due to the inherent segment-based transmission of the HAS.

In this paper, we propose an alternative approach that utilizes the playout buffer occupancy rather than using bandwidth estimates obtained from the application layer. We start with a queueing analysis of the playout buffer. Then, we present a buffer-aware rate adaptation algorithm that is solely based on the mean buffer occupancy. Our simulation results show that compared to conventional algorithms, the proposed algorithm achieves very smooth video quality while delivering a similar average video bitrate.

Keywords: HTTP-based adaptive streaming (HAS), bitrate adaptation, playout buffer model

1. Introduction

Global Internet video traffic has been growing rapidly with the emergence of popular video streaming services such as YouTube, Netflix, and Hulu. According to the Cisco Visual Networking Index, global video traffic accounted for 70 percent of all consumer Internet traffic in 2015, and nearly a million minutes of video content will cross IP networks every second by 2020 [1]. As video traffic has grown, many commercial video providers have employed adaptive bitrate streaming techniques to provide streaming media that results in the best experience for users. Recently, HTTP-based Adaptive Streaming (HAS) has become a popular technology due to its implementation and deployment simplicity [2]. In contrast to conventional RTP/UDP-based video streaming, HAS streams video over HTTP/TCP, which is the traditional protocol stack used for web traffic. Video streaming technologies such as Microsoft's Smooth Streaming, Apple's HTTP Live Streaming (HLS), and Adobe's HTTP Dynamic Streaming (HDS) rely on HTTP-based adaptive bitrate streaming. Not long ago, YouTube adopted the HAS approach as its default streaming method [3]. This approach encodes a video content at multiple bitrates and segments the encoded video content into small parts of fixed duration. When the HAS player starts video streaming, it sends an HTTP GET message requesting a segmented video content using the lowest bitrate. It downloads each video segment over HTTP and TCP using conventional HTTP web servers.

A HAS player uses a playout buffer, and it schedules the next video segment request based on the occupancy of the playout buffer. Typically, the HAS player begins video streaming in the buffering state, where the player successively requests the next video segment to fill up the playout buffer. However, after downloading a sufficient amount of data, the player begins to play the video from the buffer. After the playout buffer length is full, the player switches to a steady state in which the player periodically requests the next video segment to maintain a constant playout buffer length. The player also estimates the network bandwidth to adapt its video bitrate to the network conditions. As the video segments are transmitted over TCP, TCP dynamics such as slow-start and Additive Increase Multiplicative Decrease (AIMD) behavior affect the bandwidth estimation.

Many recent studies have focused on the issues of client-side streaming players. In [4], the authors showed that on-off traffic pattern at steady state is the root cause of instability, unfairness, and bandwidth underutilization. In [5] and [6], it was observed that when coexisting with TCP greedy flows, three commercial HAS players were unable to obtain a fair share. Consequently, rate adaptation based on inaccurate bandwidth estimation can lead to low-quality video streaming. Two rate adaptation schemes have been considered to tackle these issues: a rate shaping approach [7], and a control-theoretical approach [8]. Buffer-based rate adaptation algorithms exist that can improve the performance of HAS systems by avoiding these issues. However, most existing schemes still estimate the network bandwidth to measure the available network resources. Basically, a HAS player utilizes the per-segment throughput to adapt its video bitrate to the network bandwidth. However, because the HAS player is implemented in the application layer, it is difficult to accurately estimate the network conditions without the help of the transport layer protocol. Therefore, HAS players unnecessarily adjust the video quality based on inaccurate bandwidth estimation.

In this paper, we propose a buffer-aware rate adaptation scheme that does not need to estimate the network bandwidth for video quality selection; instead, it uses basic queueing theory to adapt video bitrates. To compare the proposed scheme with conventional algorithms, we implemented a HAS player on the ns-3 network simulator and performed simulations in unstable network environments.

The rest of the paper is organized as follows. Section 2 reviews related work on HTTP-based adaptive streaming. Section 3 presents the queueing model for the playout buffer in HAS. Section 4 describes the buffer-aware rate adaptation scheme and its adaptation algorithm. Section 5 presents the results of the proposed scheme, and Section 6 concludes the paper.

2. Related Work

Although HAS is a relatively new application, its popularity has resulted in many research works. In particular, the rate adaptation scheme is an interesting research topic because rate adaptation automatically adjusts the video quality to provide video to users at the maximum possible quality. We begin by classifying the rate adaptation schemes in HTTP-based adaptive video streaming and then describe the key shortcomings of today's state-of-art solutions.

We can classify the existing rate adaptation schemes into two main categories: bandwidth-based and buffer-based. Bandwidth-based rate adaptation schemes increase or reduce the video bitrate based on the estimated available bandwidth. Most of the rate adaptation schemes adopted by commercial video providers belong to this category. However, the inherent time-varying bandwidth leads to short-term rate oscillations and reduces the users' quality-of-experience of streaming services. In [9], the authors compared the rate adaptation schemes of three popular HAS clients, the Netflix client, Microsoft Smooth Streaming, and Adobe Open Source Media Framework (OSMF) and concluded that none of those schemes are good enough: they are either too aggressive or too conservative in rate adaptation. Some clients even jump between the highest video bitrate and the lowest video bitrate. In addition, they all have a relatively long response time to network congestion level changes. Due to complex network conditions, it is still challenging to accurately predict network bandwidth.

Buffer-based rate adaptation schemes select the video bitrate to provide continuous video playback by preventing buffer underflow (rebuffering) and overflow. A few research works have addressed the buffer-based approach. In [10], a cost-aware buffer management strategy was proposed to minimize cost induced by unconsumed video segments. In [11], the authors modeled the playback buffer as an M/M/1 queue to characterize buffer starvations. Akamai adopted the buffer-based rate adaptation technique but adjusted the video bitrate on the server side [12]. Because the server-side approach dramatically increases the burden on the web server or cache, its support for large-scale multimedia delivery is limited. Although preventing buffer underflow can ensure video playback continuity, it may also cause frequent video bitrate fluctuations and lead to an inferior quality-of-experience for users [13].

Several novel rate adaptation schemes have been proposed to strike a balance between the competing performance goals for HAS including higher average video quality, reduced rate oscillations, and minimal client rebuffering. In [14], a sophisticated Markov Decision Process (MDP) was employed to compute a set of optimal client strategies to maximize viewing quality. The authors of [15] showed that MDP-based adaptation can achieve better video streaming performance when more historical bandwidth samples are used to create a bandwidth model. However, the MDP requires knowledge of network conditions and video content statistics that may not be readily available. In [16], a control-theoretical approach was employed that included a Proportional-Integral-Derivative (PID) controller. Given an appropriate parameter choice, a PID controller can significantly improve streaming performance. In [17], a rate adaptation algorithm was designed to implement a combination of randomization for video requests, stateful video bitrate selection, and harmonic mean based averaging. In [18], machine learning based rate adaptation scheme is presented to optimize its policy for different network characteristics and Quality of Experience (QoE) metrics directly from experience. Some of the recent QoE prediction models provide dynamic adaptation of media characteristics to the varying network conditions to ensure a high QoE [19]. In [20], an efficient cloud-assisted Scalable Video Coding (SVC) video streaming with Quality of Service (QoS) aware multi-path provisioning strategies is presented to improve the performance in heterogeneous networks.

Many related works on adaptive streaming have investigated the HAS system with various rate adaptation strategies. However, the playout buffer in HAS system is usually treated as a part of the scheduling process, and it is not well-studied. In the conventional approaches, the playout buffer model is presented but its queueing dynamics is not analyzed. By using the application of queueing theory, the rate adaptation can be more robust. In this paper, we further explore an analytic model of playout buffer with queueing theory and provide a novel rate adaptation algorithm based solely on buffer information.

3. A Queueing Model for Playout Buffer in HTTP-based Streaming

This section first provides an overview of the HAS system. We then formalize a playout buffer model for the HAS player using basic queueing theory. This approach allows HAS systems to adapt video bitrates based on buffer information. **Table 1** lists the main notations used in this paper.

Table 1. Notations used in this paper

Notation	Definition
r	Video bitrate
$\mathcal{R}$	Set of video bitrates
x	Per-segment throughput
τ	Video segment duration
d	Download duration
w	Waiting time for the next request
λ	Arrival rate of video segments
μ	Service rate of video segments
b	Number of video segments in buffer
B	Playout buffer occupancy
A	Interarrival time
S	Service time
a_A, v_A	Average and variation of the interarrival time
a_S, v_S	Average and variation of the service time
c	Coefficient of variation
W	Waiting time of a queue
ρ	Traffic intensity
γ	Buffer threshold

3.1 Overview of HAS System

Fig. 1 shows a block diagram for a HAS system. Typically, the HAS system segments video content into multiple video clips of τ seconds. All the segments are pre-encoded at L video bitrates. The n-th segment from one bit-rate stream is aligned in the video time line to the n-th segment from another bit-rate stream so that a video player can smoothly switch to a different bitrate at each segment boundary. The HAS player selects the video bitrate $r \in \mathcal{R} \triangleq \{R_1, R_2, \ldots, R_L\}$, and then, depending on the playout buffer occupancy B, schedules the next download request. The player then sends an HTTP GET request to the server for the n-th segment with the video bitrate $r[n]$. In the conventional scheme, a rate adaptation algorithm measures the per-segment throughput x after downloading a segment as follows:

$$x[n] = \frac{r[n] \cdot \tau}{d[n]} \tag{1}$$

where $d[n]$ is the download duration of the n-th segment. The HAS system stores the downloaded video segment in the playout buffer.

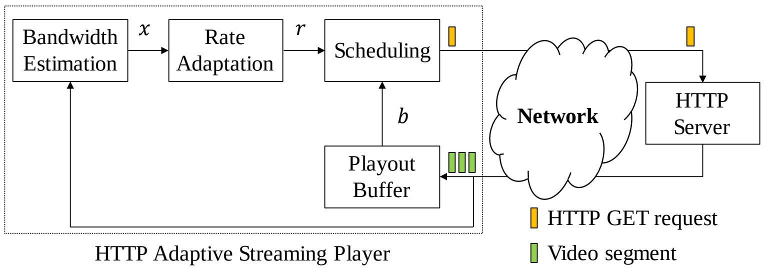


Fig 1. Block diagram for the HAS system

Because the HAS player estimates the network bandwidth in the application layer, it cannot obtain accurate information without the help of transport layer protocols. Inaccurate bandwidth estimation causes many performance issues in HAS, in particular, bitrate oscillation. Most of these problems can be solved by modifying the bandwidth estimation scheme, or by using the buffer information for rate adaptation. However, the former method has the same architectural limitation and some existing buffer-based schemes still use bandwidth information to adapt to network changes.

To solve these problems, the HAS player needs to take a more buffer-centric approach to adapting the video bitrate. By assuming that the HAS player can perfectly predict the future buffer occupancy, it can adapt its video bitrate to changing network conditions without requiring bandwidth estimation. Queueing analysis is a way to predict buffer conditions. We describe one algorithm that follows such a buffer-aware approach in Section 4.

3.2 Playout Buffer Model

In this section, we present the playout buffer model to obtain buffer information. We make some assumptions before modeling the playout buffer. First, we assume that the bottleneck bandwidth is always higher than the minimum bitrate of a video segment. Consequently, buffer underrun does not occur unless the player streams for an extended period at a bitrate higher than the bottleneck bandwidth. Second, the playout buffer is expressed in seconds: while playing, the video player consumes the playout buffer at the unit rate of one video second per real second. Third, it is assumed that video contents are encoded at a constant bitrate (CBR). Video segments will have the same amount of data. After downloading a video segment, it is stored in the playout buffer and the duration of the video segment adds to the buffer time.

With these assumptions, we can simplify the playout buffer model for HAS systems. The HAS system stores video segments in the playout buffer, which has finite waiting positions. These video segments are consumed by the decoder at a media-dependent rate in a sequential manner. The arrival process is dependent upon network conditions because the video segments are downloaded over TCP. If we assume that the video is divided into video segments that have the same duration and that users do not pause the video during playback, the departure rate will be constant.

Based on queueing theory, we adopt a G/D/1/K queue—a finite-capacity queue in which G stands for interarrival times that have a general distribution, D stands for service times that are deterministic, and K represents the queue size using Kendall's notation as the playout buffer model. Although the G/D/1/K queue has not been well-studied, we can estimate the trend of the playout buffer by measuring the basic queue characteristics. The interarrival time of adjacent video segments represents how fast the playout buffer occupancy increases. This increase also depends on the download speed and the scheduling policy. We can calculate the arrival rate λ as:

$$\lambda[n] = \frac{1}{d[n] + w[n]} \tag{2}$$

where w is the waiting time for the next segment request. The conventional scheduler in a HAS player controls playout buffer occupancy by adjusting w to avoid buffer underrun. Let $B \in [0, B_{max}]$ be the playout buffer occupancy, which is expressed in video time; the rate adaptation scheme then schedules the next download request based on the buffer occupancy as follows:

$$w[n] = \begin{cases} 0, & \text{if } B[n] < B_{max} \\ (\tau - d[n])^{+}, & \text{otherwise} \end{cases} \tag{3}$$

Unless paused, the HAS player consumes one video segment during each segment duration. Therefore, the service time is equal to the segment duration, and we can express the service rate μ as follows:

$$\mu[n] = \begin{cases} 0, & \text{paused} \\ 1/\tau, & \text{playing} \end{cases} \tag{4}$$

It is also possible to show the buffer dynamics in terms of the number of video segments. Let $b \in \{0,1,\dots,K\}$ be the number of video segments in the playout buffer. Then, we can characterize the playout buffer dynamics by the following equations:

$$b[n] = \left\lceil \frac{B[n]}{\tau} \right\rceil \tag{5}$$

$$B[n] - B[n-1] = (\lambda[n] - \mu[n]) \cdot \tau \tag{6}$$

The playout buffer increases when $\lambda > \mu$ and *vice versa*.

4. Playout Buffer-Aware Bitrate Adaptation

This section introduces the proposed rate adaptation scheme, which measures the buffer information rather than the network bandwidth for rate adaptation. We can divide the procedure of the proposed rate adaptation into four steps: estimation, expectation, adaptation, and scheduling.

4.1 Estimation

In the estimation step, the proposed scheme measures the interarrival time and service time to estimate future buffer occupancy. After a video segment has been downloaded, the algorithm estimates the average a and variation v of the interarrival time A:

$$a_A \leftarrow a_A + \alpha \cdot (A_{measure} - a_A) \tag{7}$$

$$v_A \leftarrow v_A + \beta \cdot (|A_{measure} - a_A| - v_A) \tag{8}$$

respectively, where α and β are constant smoothing factors between 0 and 1. We choose high values for the smoothing factors to reflect recent changes quickly. In the same manner, we can easily obtain the average and variation of the service time S. Because the service time is fixed, its average is identical to the segment duration, and the variation of the service time is obviously zero:

$$a_S = \tau \tag{9}$$

$$v_S = 0 \tag{10}$$

We also calculate the coefficient of variation c, which is defined as the ratio of the standard deviation (std) to the mean. Here, however, we use the mean deviation v instead of the standard deviation because v is a more conservative estimate of variation than the standard deviation. v can be a good approximation of the standard deviation and is much easier to compute.

$$c \triangleq v/a \approx std/mean \tag{11}$$

4.2 Expectation

Theoretically, any queueing system can be estimated with a given distribution. However, exact analytical solutions become very difficult to achieve. Therefore, many accurate approximations such as Kingman's G/G/1 approximation exist. Kingman's formula is the most widely used approximation for the mean waiting time in a G/G/1 queue. We treat the mean waiting time as the mean buffer occupancy. The formula is known as the VUT equation, and it is literally the product of variability, utilization and service time. The general form of VUT equation is as follows:

$$E\left[W_{G/G/1}\right] \approx \left(\frac{c_A^2 + c_S^2}{2}\right) \cdot \left(\frac{\rho}{1-\rho}\right) \cdot \frac{1}{\mu} \tag{12}$$

where $E\left[W_{G/G/1}\right]$ is the expected waiting time spent in a G/G/1 queue and $\rho = \lambda/\mu$ is the traffic intensity that represents how busy a queueing system is.

We modify the VUT equation for the playout buffer where service times are fixed and capacity is finite. As mentioned earlier, the playout buffer can be modeled as a G/D/1/K queue. In (12), the second term $\rho/(1-\rho)$ is the utilization, which represents the average number of video segments in the playout buffer when the buffer has an infinite number of waiting positions. To calculate the utilization of the K-size queue, we derive an equation by summation instead of infinite series. If the service time is constant, we can eliminate c_S from (12) because the variation of the service time is equal to zero. Therefore, the modified VUT equation for the G/D/1/K queue is as follows:

$$E\left[W_{G/D/1/K}\right] \approx \frac{c_A^2}{2} \cdot \left(\frac{\rho}{1-\rho} - \frac{K\rho^K}{1-\rho^K}\right) \cdot \frac{1}{\mu} \tag{13}$$

When K is infinite and service times have a general distribution, (13) converges to (12). When the buffer is in a steady state where ρ converges to 1, (13) also converges to the following equation:

$$\lim_{\rho \to 1} E\left[W_{G/D/1/K}\right] \approx \frac{c_A^2}{2} \cdot \frac{K-1}{2} \cdot \frac{1}{\mu} = \gamma \tag{14}$$

We can now anticipate future buffer occupancy using estimated values. The expectation varies according to the current coefficient of variation. The proposed scheme uses this predictive mean value to define a buffer threshold γ. Every time the player receives a video segment and buffers it, the buffer threshold γ is updated dynamically. The buffer threshold γ is a useful criterion to represent the sufficiency of buffer occupancy.

4.3 Adaptation

In the adaptation step, the proposed scheme determines the video bitrate for the next segment request. Previous studies of the HAS system have noted that many performance issues such as rebuffering and bitrate oscillation occur in this step. The rate adaptation scheme must prevent rebuffering and minimize bitrate oscillations to improve the user experience.

Every time it receives a video segment, the proposed scheme determines the video bitrate based on the current buffer occupancy $B[n]$. If the buffer has sufficient space and $\rho \geq 1$ (meaning the buffer is increasing or remaining stable), it is safe to switch to a higher bitrate. The proposed scheme decides to increase the video bitrate when the buffer occupancy exceeds its predictive mean value γ. However, in the worst-case scenario, where the network bandwidth is reduced when a player requests a segment at the highest bitrate, the buffer will decrease by at most $(R_{max} \cdot \tau)/x$ seconds. Alternatively, when the available bandwidth is higher than R_{min}, we can avoid rebuffering by switching to a lower video bitrate when the buffer occupancy is lower than $\epsilon = (R_{max} \cdot \tau)/R_{min}$ seconds and $\rho < 1$ where the buffer is decreasing. The video bitrate will remain constant except in the case of the previous situations. We can express the proposed adaptation algorithm as follows:

$$r[n+1] = \begin{cases} R_{l+1}, & \text{if } B[n] > \gamma \text{ and } \rho \geq 1 \\ R_{l-1}, & \text{if } B[n] < \epsilon \text{ and } \rho < 1 \\ r[n], & \text{otherwise} \end{cases} \tag{15}$$

where $r[n] = R_l$ is the video bitrate of the n-th segment.

To prevent oscillation in the video bitrate, we present a finite-state machine (FSM) for rate adaptation of the HAS system that has three states: switch up, switch down, and hold. **Fig. 2** shows all transitions from one state to another. It is designed to alleviate oscillations by blocking the spikes that cause the transitions to switch between up and down. The proposed scheme also tries to maintain the current video bitrate whenever it detects a variation in the playout buffer. Because bitrate switches are the important factor impacting user viewing experience, we need to avoid unnecessary bitrate switches. The proposed scheme switches bitrates conservatively based on dynamic thresholds and returns to the hold state to prevent rate oscillations.

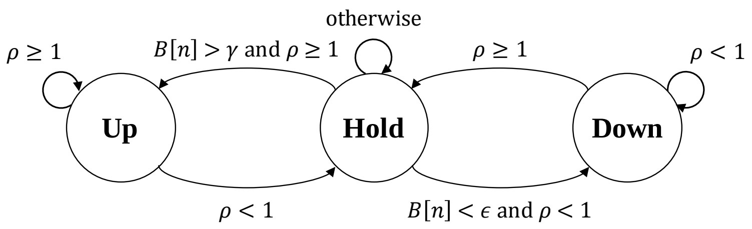

otherwise
$\rho \geq 1$
$B[n] > \gamma$ and $\rho \geq 1$
$\rho \geq 1$
$\rho < 1$
Up
Hold
Down
$\rho < 1$
$B[n] < \epsilon$ and $\rho < 1$

Fig. 2. State diagram for rate adaptation

4.4 Scheduling Policy

In the conventional approach to rate adaptation, the HAS player tries to fill up the playout buffer. To achieve this goal, the player selects a video bitrate that is lower than the estimated bandwidth and successively requests video segments until the buffer occupancy reaches its maximum. According to (3), when the playout buffer is full, the player periodically requests the next video segment every τ seconds to maintain buffer occupancy. The on-off traffic pattern occurs due to this behavior. It is widely known that the on-off traffic pattern is the key factor that causes performance degradation in HAS systems.

The problem occurs because the TCP connection is idle during the off period. When TCP has not received a packet for more than one retransmission timeout (RTO), the TCP congestion window (cwnd) is reduced to the initial value, and the TCP connection restarts slow-start after the idle period. Slow-start causes a reduction in TCP throughput and in application throughput, leading to low-quality video streaming because the network bandwidth is underestimated. Thus, we need to modify the scheduling policy to avoid potential shortcomings.

The proposed scheme simply adds a condition to (3) to prevent the influence of on-off traffic. If the network bandwidth is sufficiently high for streaming video at the highest bitrate, bandwidth underestimation will not affect the streaming quality, even when on-off traffic exits. The proposed scheme schedules the next segment request with waiting duration only if it is streaming at the highest quality, as follows:

$$w[n] = \begin{cases} 0, & \text{if } B[n] < B_{max} \text{ or } r[n] < R_{max} \\ (\tau - d[n])^{+}, & \text{otherwise} \end{cases} \tag{16}$$

5. Simulation Results

In this section, we perform a set of simulations to evaluate the performance of the proposed algorithm against other conventional algorithms.

5.1 Simulation Setup

We implemented the HAS systems in the ns-3 network simulator. All simulations use a simple dumbbell network topology as shown in **Fig. 3**. The HTTP server provides five different video bitrates: 700 kbps, 1.4 Mbps, 2.1 Mbps, 2.8 Mbps, and 3.5 Mbps. Each video is divided into small equal-length video segments, each of which is 2 seconds in length. To conduct performance comparisons, we implemented the following algorithms in the HAS player:

- The conventional bandwidth-based rate adaptation algorithm (CA) and the PANDA algorithm [21].
- The BBA algorithm with a linear rate map and a reservoir of 10 seconds [22].
- The proposed algorithm with $\alpha = 0.8$ and $\beta = 0.8$.

We picked CA as a basic rate adaptation scheme of HAS clients. PANDA is selected since it is a well-known algorithm to reduce bitrate oscillations in multi-client environments. BBA is also selected since its buffer-based rate adaptation scheme is similar to the proposed approach. CA always chooses a video bitrate that is lower than its simple estimation of network bandwidth. PANDA is also a bandwidth-based algorithm, but it estimates the network bandwidth more conservatively by taking an AIMD approach such as TCP's congestion avoidance mechanism to prevent bitrate oscillations. BBA is a buffer-based algorithm that selects the video bitrate using a rate map that is a linear function of the current buffer occupancy. All the implemented players employed a 30-second threshold for the playout buffer length.

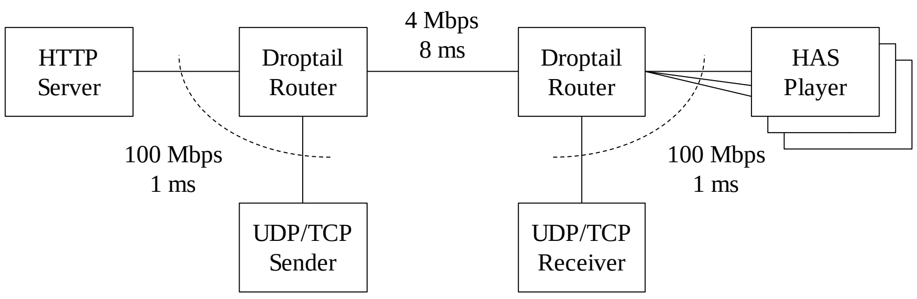


Fig. 3. Network topology configured in the simulation

5.2 Single Player over Time-Varying Link

We first evaluated the performance of all four schemes under the short-term bandwidth variations shown in **Fig. 4**. In this scenario, the available bandwidth fluctuates every 100 seconds, varying from 3 to 4 Mbps in the first interval, from 2 to 3 Mbps in the second interval, from 1 to 2 Mbps in the third interval, from 1 to 3 Mbps in the fourth interval, from 2 to 4 Mbps in the fifth interval, and from 1 to 4 Mbps in the last interval. The magnitude of the spikes in bandwidth varies from 1 to 3 Mbps. **Fig. 4** shows that conventional schemes react to the spikes by increasing/decreasing the requested video bitrate, whereas the proposed scheme ignores some spikes by employing the state machine for rate adaptation. Although the bandwidth-based rate adaptation schemes smooth estimated values to cover spikes, the constant smoothing factor is not suitable under extreme bandwidth variations. **Table 2** shows that the PANDA algorithm achieves a lower throughput than other schemes in this scenario because of the TCP slow-start restart problem. Because conventional schemes wait for the next segment request when the playout buffer reaches its maximum, the TCP connection must be re-initialized. We will further examine this phenomenon in later experiments.

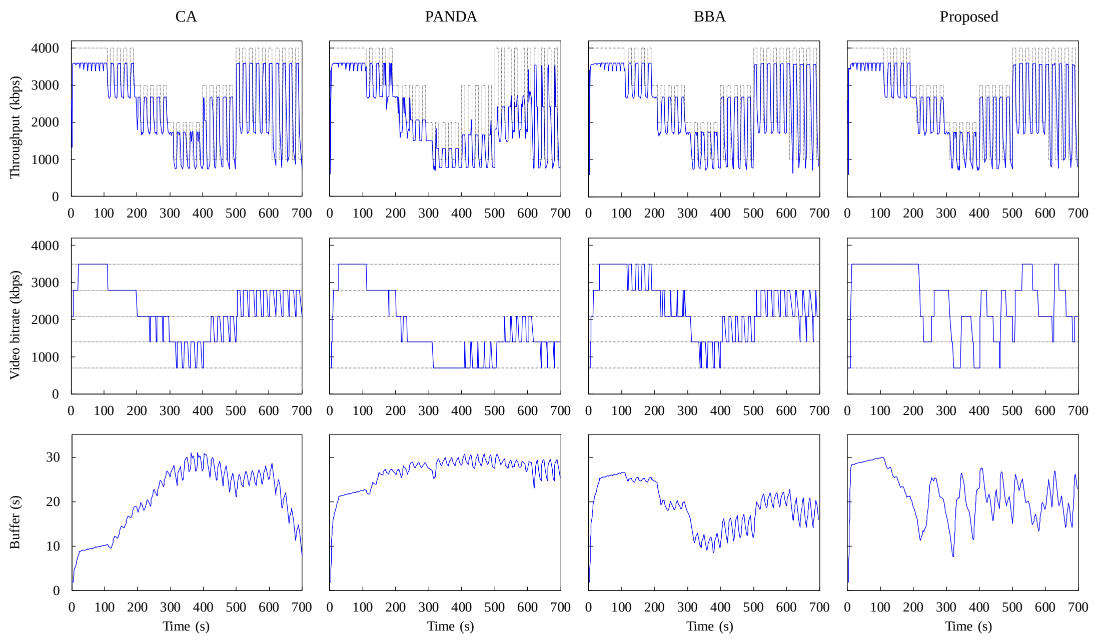


Fig. 4. Video bitrate adaptation under short-term bandwidth variations

Table 2. Performance summary under short-term bandwidth variations

Scheme	Bandwidth utilization (%)	Average video bitrate (Mbps)	Average buffer occupancy (second)	Number of switches in bitrate
CA	96.24	2.31	20.41	50
PANDA	80.03	1.73	26.21	39
BBA	97.70	2.31	18.74	72
Proposed	98.52	2.30	21.25	35

We then evaluate the impact of long-term bandwidth variations under the scenario where the network bandwidth changes every 100 seconds. In this scenario, the bottleneck link follows a bandwidth profile of {4 Mbps → 3 Mbps → 2 Mbps → 1 Mbps → 2 Mbps → 3 Mbps → 4 Mbps} without bandwidth spikes. The results of the various schemes in **Fig. 5** and **Table 3** show that CA and PANDA do not fully utilize the available bandwidth after 400 seconds when the buffer occupancy reaches the maximum. As discussed earlier, this phenomenon occurs due to their scheduling policy. The BBA algorithm uses the same scheduling policy but achieves higher throughput because it increases the video bitrate according to buffer growth before it enters the steady state. However, we see that BBA experiences severe oscillations between two adjacent video bitrates even where the network is stable. This occurs because BBA determines the video bitrate without considering the fluctuating buffer length. The proposed scheme prevents slow-start restarts using the method discussed in Section 4.4. Consequently, the proposed scheme achieves higher bandwidth utilization and average video bitrate compared to the conventional schemes.

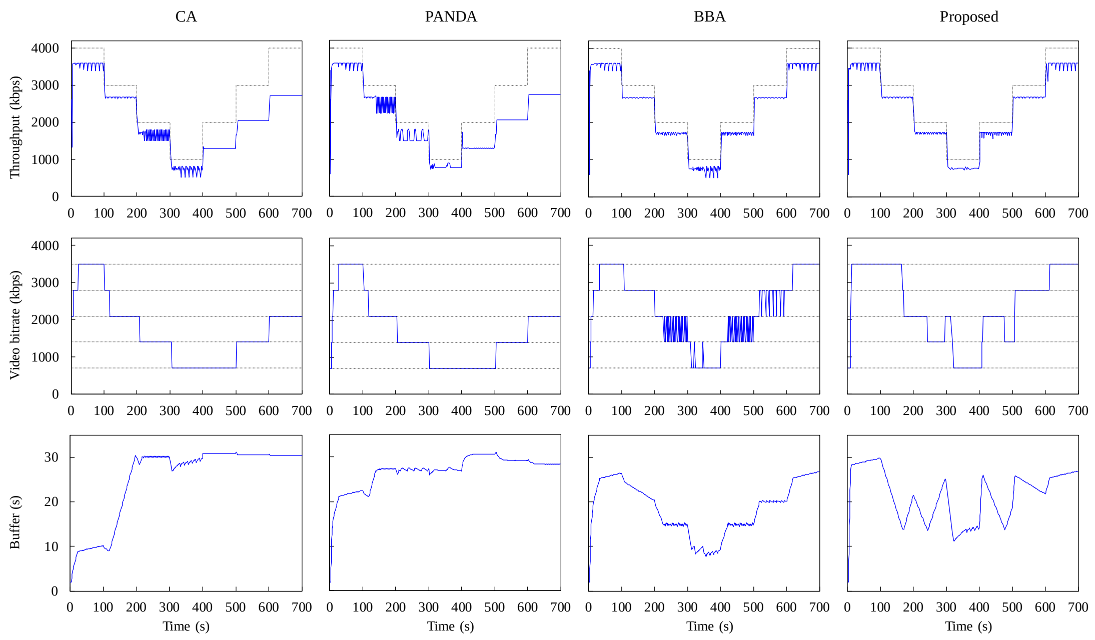


Fig. 5. Video bitrate adaptation under long-term bandwidth variations

Table 3. Performance summary under long-term bandwidth variations

Scheme	**Bandwidth utilization (%)**	**Average video bitrate (Mbps)**	**Average buffer occupancy (second)**	**Number of switches in bitrate**
CA	76.83	1.72	24.86	8
PANDA	76.80	1.67	26.45	10
BBA	86.01	2.28	18.44	96
Proposed	85.76	2.28	21.09	16

5.3 Multiple Competing Players

In this scenario, we investigated the behavior of the algorithms when multiple HAS players share a network bottleneck and compete for the available bandwidth. For each player, the video streaming duration was 400 seconds. There were four players, and one player launched every 100 seconds. **Fig. 6** shows the dynamics of video bitrate for each algorithm. Each colored line represents a different HAS player. In **Fig. 6**, CA and PANDA exhibit similar behavior: they do not achieve fairness after 400 seconds when the buffer has grown sufficiently. BBA once again shows oscillating patterns because its buffer occupancy fluctuates as the number of players increases. In contrast, the proposed scheme provides a video bitrate that oscillates around the fair share bandwidth with only a small number of switches.

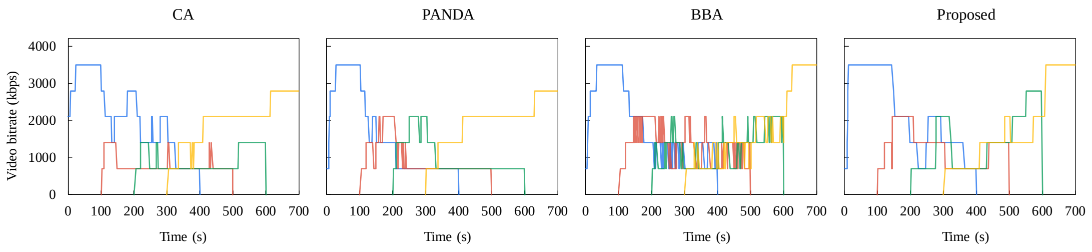


Fig. 6. Multiple HAS players sharing a 4 Mbps bottleneck link

5.4 Competing with Other Traffic

In this simulation, we tested the algorithms against three different types of cross-traffic: exponential traffic, Pareto traffic with a shape parameter of 2.0, and File Transfer Protocol (FTP). Exponential traffic represents a traditional model for circuit-switched data, Pareto traffic shows the bursty nature of packet data, and FTP is a greedy source that generates data at the maximum rate possible. In this scenario, a HAS player shares the bottleneck bandwidth with each type of cross-traffic and all the cross-traffic is generated before the player starts. The duration of all simulations is 600 seconds. We repeat each simulation 10 times and then average the results.

Fig. 7 plots the average video bitrate and the number of bitrate switches for all algorithms. CA and PANDA have similar performance; both suffer from FTP traffic, which sends data packets over TCP. BBA and the proposed scheme achieve a higher video bitrate compared to the bandwidth-based rate adaptation schemes, but BBA has severe bitrate oscillations due to its rate map function. In contrast, the proposed scheme significantly reduces the number of bitrate switches by adopting a buffer-aware algorithm for video bitrate selection.

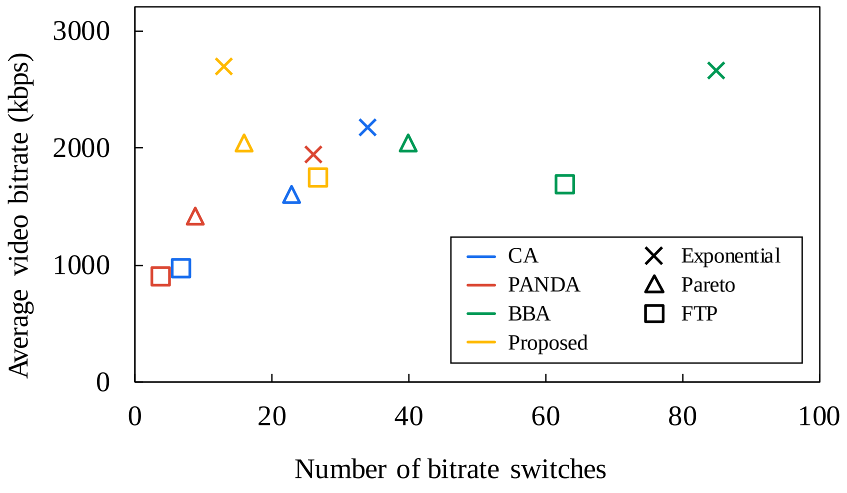
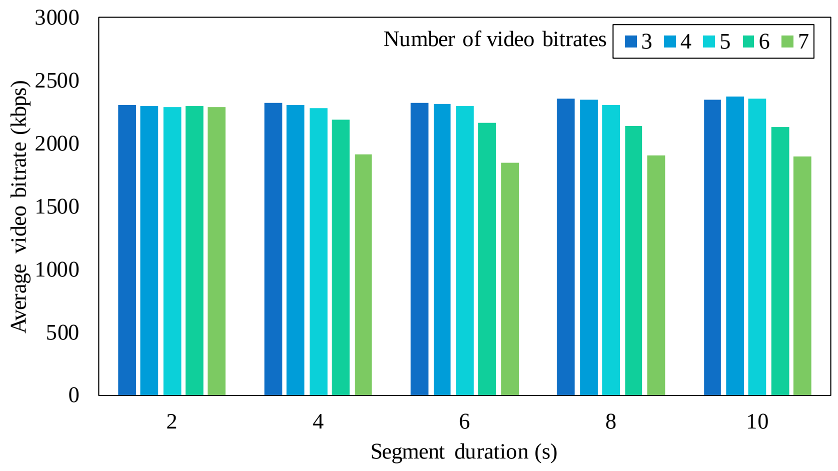


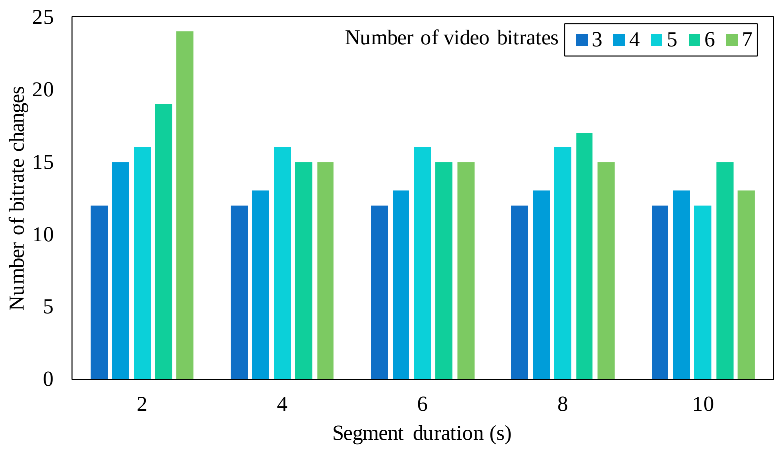
Fig. 7. The performance of all four algorithms against cross-traffic

5.5 Different Encoding Configuration

To evaluate the performance with a variety of video profiles, we change the duration of video segments and the number of video bitrates. In this simulation, video bitrates are distributed between 700 and 3500 kbps with exponentially increasing intervals. For example, if the number of video bitrates is 4, the actual video bitrates are 700, 1197, 2047, and 3500 kbps. We repeat each simulation for {2, 4, 6, 8, 10} seconds of segment durations and {3, 4, 5, 6, 7} number of video bitrates under long-term bandwidth variations as described in Section 5.2. **Fig. 8** shows the comprehensive performance of the proposed algorithm for each video setting. We calculate the average video bitrate and the number of bitrate changes to evaluate both QoS and QoE. In terms of the streaming quality, we see that the proposed algorithm chooses a lower bitrate as the segment duration increases as shown in **Fig. 8(a)**. This is because the proposed algorithm estimates its target buffer occupancy using the segment duration. Thus, long segment duration means that the proposed scheme spends more time on filling up the playout buffer before increasing the video bitrate. Also, the proposed scheme shows higher video bitrate when the number of video bitrates is small. Since the target buffer occupancy varies according to the variability of interarrival time, a large number of video bitrates increases the variability of arrival rate and will decrease the average video bitrate. For the number of bitrate changes, we expect that it may increase as the number of video bitrates increases. This is presented in **Fig. 8(b)** when the segment duration is 2 seconds. But the impact on the number of bitrate changes is reduced as the segment duration increases because the adaptation algorithm does not react quickly since its estimation period, which is the same as the request interval, is relatively long. As a result, the proposed scheme works effectively with a small number of video bitrates and short duration of the video segment.



(a) Average video bitrate



(b) The number of bitrate changes

Fig. 8. The performance of the proposed algorithm with different video settings

6. Conclusion

Typical HAS players unnecessarily adjust video quality because of poorly designed bandwidth estimation methods and the on-off traffic pattern. In this paper, we propose a buffer-aware rate adaptation scheme that avoids the effects of inaccurate bandwidth estimation. The goal of the proposed scheme is to achieve higher video bitrates and to prevent bitrate oscillations based on buffer occupation. To obtain this buffer information, we present a queueing-theoretic model for the playout buffer in which the rate adaptation is designed based on the mean buffer occupancy without considering any network conditions. To evaluate the performance of the HAS player, we implemented the HAS system with rate adaptation schemes and performed simulations using the ns-3 network simulator. Specifically, we compared the proposed algorithm with three conventional algorithms in terms of average video quality and smoothness. Based on our simulation results, the proposed algorithm achieves very smooth video quality even under unstable network conditions. The proposed scheme improves stability by employing a state machine that uses buffer information while maximizing the video bitrate. In future work, we plan to expand the proposed scheme with a more analytical buffer model for real-world commercial players.

References

[1] "Cisco Visual Networking Index: Forecast and methodology, 2015-2020," *Cisco System Inc.*, San Jose, CA, USA, Jun. 2016.

[2] T. Stockhammer, "Dynamic adaptive streaming over HTTP - Standards and design principles," in *Proc. of the 2nd Annual ACM Conf. on Multimedia Systems.*, pp. 133–144, Feb. 2011. Article (CrossRef Link).

[3] J. Roettgers, "Don't touch that dial: How YouTube is bringing adaptive streaming to mobile, TVs," Mar. 2013. [Online]. Available: Article (CrossRef Link)

[4] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?," in *Proc. of the 22nd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 9–14, Jun. 2012. Article (CrossRef Link).

[5] T. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. of the 2012 Internet Measurement Conf.*, pp. 225–238, Nov. 2012. Article (CrossRef Link).

[6] L. D. Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH)," in *Proc. of Int. Packet Video Workshop*, pp. 1–8, Dec. 2013. Article (CrossRef Link).

[7] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Began, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proc. of the 23rd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 19–24, Feb. 2013. Article (CrossRef Link).

[8] X. Zhu, Z. Li, R. Pan, J. Gahm, and H. Hu, "Fixing multi-client oscillations in HTTP-based adaptive streaming: A control theoretic approach," in *Proc. of IEEE Int. Workshop on Multimedia Signal Processing*, pp. 230–235, Oct. 2013. Article (CrossRef Link).

[9] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. of the 2nd Annual ACM Conf. on Multimedia Systems*, pp. 157–168, Feb. 2011. Article (CrossRef Link).

[10] J. He, Z. Xue, D. Wu, D. O. Wu, and Y. Wen, "CBM: Online strategies on cost-aware buffer management for mobile video streaming," *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 242–252, Jan. 2014. Article (CrossRef Link).

[11] Y. Xu, E. Altman, R. El-Azouzi, M. Haddad, S. Elayoubi, and T. Jimenez, "Analysis of buffer starvation with application to objective QoE optimization of streaming services," *IEEE Transactions on Multimedia*, vol. 16, no. 3, pp. 813–827, Apr. 2014. Article (CrossRef Link).

[12] L. D. Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proc. of the 2nd Annual ACM Conf. on Multimedia Systems*, pp. 145–156, Feb. 2011. Article (CrossRef Link).

[13] E. C. R. Mok, X. Luo, and R. Chang, "QDASH: A QoE-aware DASH system," in *Proc. of the 3rd Multimedia Systems Conf.*, pp. 11–22, Feb. 2012. Article (CrossRef Link).

[14] D. Jarnikov and T. Ozcelebi, "Client intelligence for adaptive streaming solutions," *Signal Processing: Image Communication*, vol. 26, no. 7, pp. 378–389, Aug. 2011. Article (CrossRef Link).

[15] A. Bokani, M. Hassan, S. Kanhere, and X. Zhu, "Optimizing HTTP-based adaptive streaming in vehicular environment using markov decision process," *IEEE Transactions on Multimedia*, vol. 17, no. 12, pp. 2297–2309, Dec. 2015. Article (CrossRef Link).

[16] C. Zhou, X. Zhang, L. Huo, and Z. Guo, "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," in *Proc. of IEEE Visual Communications and Image Processing*, pp. 1–6, Nov. 2012. Article (CrossRef Link).

[17] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *Proc. of the 8th Int. Conf. on Emerging Networking Experiments and Technologies*, pp. 97–108, Dec. 2012. Article (CrossRef Link).

[18] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. of the ACM Conf. on SIGCOMM*, pp. 197–210, Aug. 2017. Article (CrossRef Link).

[19] K. Miller, A. Al-Tamimi, and A. Wolisz, "QoE-based low-delay live streaming using throughput predictions," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 1, Oct. 2016. Article (CrossRef Link).

[20] Z. Zhu, S. Li, and X. Chen, "Design QoS-aware multi-path provisioning strategies for efficient cloud-assisted SVC video streaming to heterogeneous clients," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 758-768, Jun. 2013. Article (CrossRef Link).

[21] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Began, and D. Oran, "Probe and adapt: Adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, Apr. 2014. Article (CrossRef Link).

[22] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. of the 2014 ACM Conf. on SIGCOMM*, pp. 187–198, Aug. 2014. Article (CrossRef Link).

Jiwoo Park received a B.S. degree in electronics and communications engineering from Kwangwoon University, Seoul, Korea, in 2009, where he is currently working toward a Ph.D. degree in electronics and communications engineering. His research interests include network protocols, multimedia systems, and video communications—in particular, QoS support in adaptive bitrate streaming.

Kwangsue Chung received B.S. degree from Hanyang University, Seoul, Korea, M.S. degree from KAIST (Korea Advanced Institute of Science and Technology), Seoul, Korea, Ph.D. degree from University of Florida, Gainesville, Florida, USA, all from Electrical Engineering Department. Before joining the Kwangwoon University in 1993, he spent 10 years with ETRI (Electronics and Telecommunications Research Institute) as a research staff. He was also an adjunct professor of KAIST from 1991 to 1992 and a visiting scholar at the University of California, Irvine from 2003 to 2004. His research interests include communication protocols and networks, QoS mechanism, and video streaming.