Hindawi
Advances in Multimedia
Volume 2017, Article ID 7323681, 12 pages
https://doi.org/10.1155/2017/7323681

Research Article

SDP-Based Quality Adaptation and Performance Prediction in Adaptive Streaming of VBR Videos

Thoa Nguyen,[1] Thang Vu,[1] Nam Pham Ngoc,[1] and Truong Cong Thang[2]

[1]*Hanoi University of Science and Technology, Hanoi, Vietnam*
[2]*The University of Aizu, Aizuwakamatsu, Japan*

Correspondence should be addressed to Thoa Nguyen; thoa.nguyenthikim@hust.edu.vn

Received 27 January 2017; Revised 8 May 2017; Accepted 24 May 2017; Published 2 July 2017

Academic Editor: Mei-Ling Shyu

Recently, various adaptation methods have been proposed to cope with throughput fluctuations in HTTP adaptive streaming (HAS). However, these methods have mostly focused on constant bitrate (CBR) videos. Moreover, most of them are qualitative in the sense that performance metrics could only be obtained after a streaming session. In this paper, we propose a new adaptation method for streaming variable bitrate (VBR) videos using stochastic dynamic programming (SDP). With this approach, the system should have a probabilistic characterization along with the definition of a cost function that is minimized by a control strategy. Our solution is based on a new statistical model where the future streaming performance is directly related to the past bandwidth statistics. We develop mathematical models to predict and develop simulation models to measure the average performance of the adaptation policy. The experimental results show that the prediction models can provide accurate performance prediction which is useful in planning adaptation policy and that our proposed adaptation method outperforms the existing ones in terms of average quality and average quality switch.

1. Introduction

Nowadays, video services are increasingly popular on the Internet. According to a recent study and forecast [1], global Internet video traffic will be 80% of the entire consumer Internet traffic in 2019. Besides, HTTP protocol has become a cost-effective solution for video streaming thanks to the abundance of Web platforms and broadband connections [2, 3]. Furthermore, for interoperability of HTTP streaming in the industry, ISO/IEC MPEG has developed "Dynamic Adaptive Streaming over HTTP" (DASH) [4] as the first standard for video streaming over HTTP.

DASH requires a video to be available in multiple bitrates and split into small segments each containing a few seconds of playtime. Based on the current network conditions and terminal capacity, the client can adaptively decide a suitable data rate so that stalling is avoided and the available bandwidth is best possibly utilized. If the video is encoded in only one bitrate, either the bitrate is smaller than the available bandwidth resulting in a smooth playback but sparing resources which could be utilized for a better video quality, or the video bitrate is higher than the available bandwidth leading to video stalling. Thus, DASH enables service providers to improve resource utilization and quality of experience (QoE).

So far, existing studies have proposed simple heuristics for adapting video at the client. These heuristics can be divided into two types, buffer-based methods and throughput-based methods. The purpose of buffer-based methods is to maintain the stability of the buffer within a certain range to ensure continuous video playback. However, when the bandwidth is drastically reduced, the buffer-based methods may cause sudden change of bitrate [5–8]. Meanwhile, throughput-based methods adaptively decide version based on the estimated throughput. These methods are generally able to react quickly to the throughput variations; the streaming quality, however, may be unstable [9].

Recently, several Markov decision-based methods have been proposed to optimize decision making for the streaming client under time-varying network conditions. However, these existing methods mostly focus on constant bitrate (CBR) videos. The authors in [10] are the first to propose an adaptation algorithm in which stochastic dynamic programming (SDP) is employed to find optimal decision policies when streaming VBR videos. The segment requests are ruled by the policies which map a control parameter to every possible state of the system; however, it is limited to videos with weak bitrate fluctuations. To the extent of the authors' knowledge, in the context of adaptive streaming, there have not been any adaptive streaming methods that could (1) support variable bitrate (VBR) videos with strong bitrate fluctuations and (2) predict the streaming performance with different streaming settings in order to select the optimal one.

In this paper, we tackle these challenges by proposing an adaptation method using stochastic dynamic programming. Firstly, we discretize a system including data throughput, buffer level, and bitrate of a VBR video to form the system states. Secondly, we define a cost function that takes into account parameters that affect the subjective perceptual quality of users. In the cost function, the weights are assigned to the difference between data throughput and the bitrate of the next segment, the variance of the buffer from its optimal value, and the quality switch of the video. Finally, we construct an infinite horizon problem (IHP) and solve it to find the optimal policies for all system states. The role of a policy is mapping the control parameter (i.e., the version of the video) to every possible state of the system. This paper is an extended work of our preliminary study in [11]. The extension in this work is multifold. First, we predicted the CDF of the requested versions in a streaming session, so the maximum version could be decided for the streaming session. Second, we predicted CDF of the buffer levels to know the variance of the buffer level under the fluctuations of the network. Finally, we also evaluated the proposed method in the online context, where the statistics of bandwidth is updated periodically. Besides, we compared the performance prediction results with measurement ones in in both offline and online contexts.

A policy is optimal when it minimizes an average cost. Based on the obtained policies and the constructed system model, we develop mathematical models that could predict the streaming performance for the new streaming session including average video version, average version switch per segment, average buffer, and average underflow probability. Experiments are conducted to verify the mathematical models by comparing the predicted performance obtained from the models and the measured performance. The proposed method is evaluated in two contexts (1) offline context using statistics of a history bandwidth trace and (2) online context using bandwidth statistics of previous video segments.

The paper is organized as follows. Section 2 briefly reviews related work. Section 3 describes the system and the modeling of the system in detail. Section 4 presents the formulation of the IHP which is solved by SDP. The performance prediction is given in Section 5. Section 6 presents the experimental results and discussions. Finally, Section 7 concludes our work.

2. Related Work

In recent years, many heuristic adaptation methods for adaptive streaming have been developed (e.g., [5–9, 14]). An extensive evaluation of typical adaptation methods has been carried out in [15]. Though these methods prove to be effective in their specific settings, they cannot tell quantitatively the streaming performance with different system settings. Furthermore, most of them focus on CBR videos.

For streaming VBR video, several adaptation methods have been proposed based on the sensibility of the receiver's buffer [6, 16]. Dubin et al. [16] propose an adaptation logic that supports its bandwidth estimation decisions based on the client buffer redundancy. This method considers the fluctuations of mobile network without emphasizing the characteristics of bitrate fluctuation of VBR videos, which leads to the lack of smoothness. In [6], a partial-linear trend prediction model is developed to estimate the trend of client buffer level variation. The client will continue to have the current version for the next segment when the estimated buffer level has no significant change. The drawback of this method is the sudden version switch when the actual buffer level drops dramatically. In [14], the authors propose an adaptive logic for VBR videos based on bitrate estimation. The method demonstrates an effective adaptation behavior as it keeps the buffer at a stable and high level; however, it is still qualitative and has no mechanism to balance the streaming performance.

Several recent studies have proposed mathematical model based adaptation methods in streaming video [17–21]. A mathematical model is proposed in [17] to calculate the underflow probability of VBR streaming under VBR channel based on initial delay and maximum buffer. However, this work only considered VBR videos with one version and constant play-out curve without developing any adaptive logic. Meanwhile, Kang et al. [18] present a no-reference, content-based QoE estimation model for video streaming service over wireless networks using neural networks. Nonetheless, neural networks are computationally complex and require large training data and long training time. Besides, Xiang et al. [19] propose a rate adaptation method using the Markov Decision Process to obtain an optimal streaming strategy for VBR videos. Nevertheless, their proposal is not able to adapt to the real bandwidth changes and has no performance prediction. The prediction of streaming performance was first proposed for streaming CBR videos in [20] and VBR videos in [21] by Liu et al. In their studies, a video session is divided into subsessions. With a given target rebuffering probability, the video bitrate/the average video bitrate for each streaming subsession is predicted when streaming CBR videos/VBR videos, respectively. The results show that the average actual rebuffering probability achieved by these methods is reasonably close to the target. However, they have not done any assessments in terms of video quality and video quality switch, so the QoE can be affected.

Recently, SDP has been known as an effective technique for solving optimization problems in video streaming [10, 22, 23]. For instance, the authors in [22] apply SDP to find the optimal policy for choosing sending rates when streaming

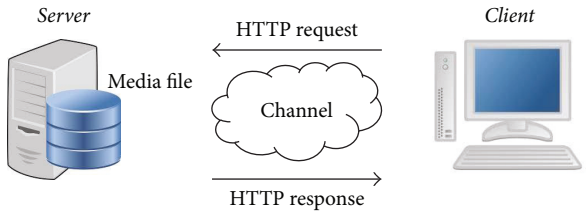

Figure 1: General DASH system.

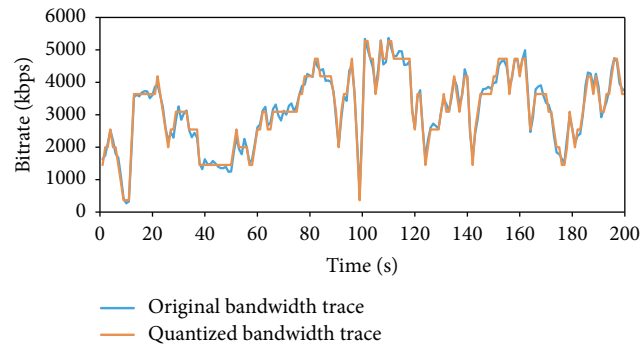

Figure 2: A bandwidth trace obtained from a mobile network [12].

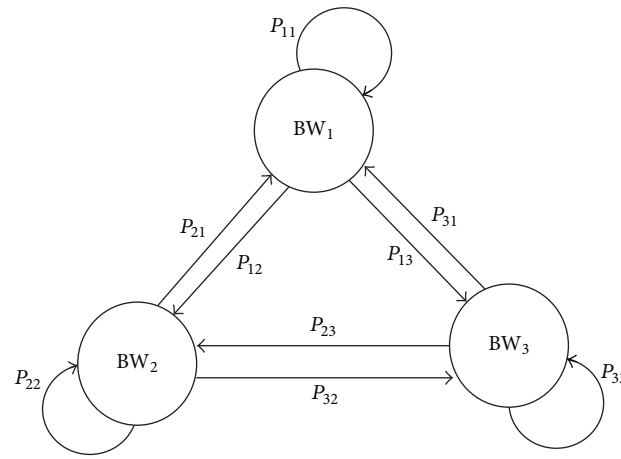

Figure 3: General 3-state Markov-chain bandwidth model.

on-demand scalable VBR videos over wireless network. Nevertheless, they have not considered the effect of channel-state aware adaptation. Meanwhile, García et al. [10] construct an infinite horizon problem and apply SDP to solve it specifically for HTTP adaptive streaming. They propose a channel model in which transitions are only possible to adjacent states, with equal probabilities, which is only suitable for the stable bandwidth, with little fluctuation. In addition, by observing the histograms of segment size encoded with VBR, the authors assume that segment size (which is proportional to segment bitrate) can be modeled through a discrete Gaussian distribution. Then, the probability distribution of segment sizes is used to calculate transition probabilities. However, the probability distribution of segment sizes is not taken into account in the cost function. Instead, the average bitrate representing the bitrate for each version is used. This is only reasonable when the deviations of segment sizes (i.e., segment bitrates) are small. In another study, Xing et al. [23] use SDP to find the optimization for Advance Video Coding content when streaming through several wireless connections at the same time. They offer a cost function in terms of QoE, but the computational complexity of their model is significantly caused by eight system variables in each state.

The SDP-based method proposed for HTTP streaming in this paper is *different from the previous studies in several points*. First, our method considers an actual time-varying bandwidth of mobile networks. Second, the drastic bitrate fluctuations of actual VBR videos are effectively supported. Third, we develop mathematic models to predict the performance of a streaming subsession, which helps to select the maximum allowed version parameter in advance.

3. System Modeling

3.1. System Overview. Figure 1 shows the functional diagram of a general DASH system consisting of a server and a client. The server holds the media files with different quality versions. Each version is further divided into small segments. The client has the information about the characteristics and locations of media segments and can request any of them during a streaming session. For the next segment, the client makes a decision of what video version to request based on current status of client buffer and data throughput to provide the best streaming experience possible. In this paper, the segment selection policy for the client aims at maintaining the client buffer at a reasonable level while balancing between average version (i.e., average quality) and average version switch (i.e., quality variations).

In order to apply SDP, our system is modeled as a discrete-time stochastic one. Specifically, the timeline is divided into time stages. At each stage k, the system is represented by a state variable s_k. When the next segment is completely downloaded at stage $k+1$, the system moves to the state s_{k+1}. As the system transits to the next state, a certain cost occurs. Besides, the channel, the buffer, and the media are discretized as explained in the following subsection.

3.2. Channel, Buffer, and Media Model. In our work, we discretize the bandwidth range into W levels. The bandwidth trace before and after discretization is shown in Figure 2 with $W = 10$. With this level of quantization, the quantized bandwidth covers the original bandwidth well. We then create W different bandwidth states BW_i $(1 \le i \le W)$ from these W bandwidth levels. The value of each bandwidth state is the average of the maximum value and minimum value of the corresponding bandwidth level. To represent the transition from one bandwidth state to another on a bandwidth state space, we use the Markov chain model which has been used widely in previous studies [10, 24, 25].

Figure 3 presents a general Markov-chain model which consists of three bandwidth states. Each state is represented by one data throughput value. There is a transition probability when the bandwidth moves from one state to another after

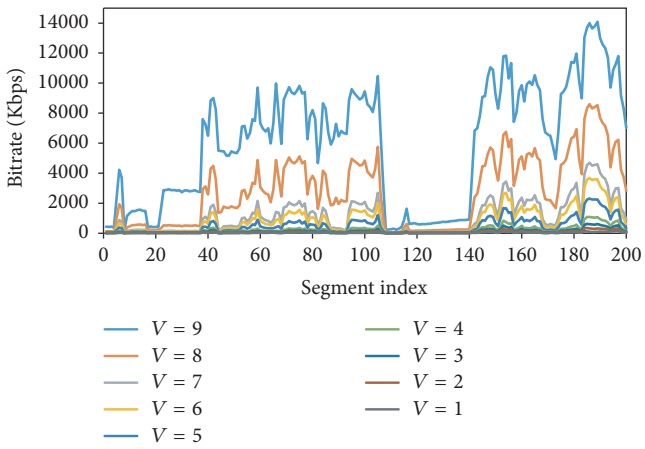

FIGURE 4: Bitrates of different versions of Tokyo Olympic video [13].

each time step. Thus, by simply extracting the statistics from the bandwidth history, the transition probability between all bandwidth states is generated.

Similar to the bandwidth trace, we divide the buffer into B levels from 0 to B_s, with B_s being the buffer size. In addition, we denote the video version by V and represent the version with lowest quality as $V = 1$ and the highest quality as $V = V\max$ assuming that the video has $V\max$ different versions. The bitrates of different versions of a VBR video are shown in Figure 4.

Because the segment bitrate of a VBR video version fluctuates very strongly, we divide the bitrate of each version into I intervals (from interval 1 to interval I), each of which is represented by its average bitrate value. For example, if a version that has the highest bitrate of 5000 kbps is divided into 10 bitrate intervals, the interval 1 will range from 0 to 500 kbps and its representative bitrate will be 250 kbps.

We assume that all versions at a bitrate interval represent a separate video flow. If the current segment bitrate belongs to one interval, the next segment bitrate will also belong to that interval regardless of segment version. With this assumption, we generate I different policy sets for I bitrate intervals. When a segment is completely downloaded, the client measures its bitrate to find the bitrate interval it belongs to. Then, the client will determine the policy set corresponding to that bitrate interval.

4. Problem Formulation and Solution

4.1. System State. With the system being discretized above, we observe the system state variable $s_k(b_k, \mathrm{bw}_k, v_k)$ when a video segment is completely downloaded at stage k. Here, b_k is the buffer level representing the number of segments available in the buffer, bw_k is the bandwidth whose value belongs to $\{\mathrm{BW}_i\}$, and v_k is the version index of the downloaded segment. The case where $b_k = 1$ corresponds to the buffer underflow event. In each state s_k, the system may choose any action a. For our system, an action is basically a decision about the version for the next segment. As there are $V_{\max}$ versions to choose from, we have totally $V_{\max}$ possible actions.

The system then randomly moves into a new state s_{k+1} at the next time step, resulting in a corresponding cost $C(s_k, s_{k+1}, a)$. With each bitrate interval i $(1 \le i \le I)$, we have a system state set $\mathbf{s}_i$ and a policy set $\boldsymbol{\mu}_i$. Let N be the number of states in $\mathbf{s}_i$, and we have

$$N = B * W * V_{\max}. \tag{1}$$

4.2. Transition Probabilities. Since the system is stochastic, which means the system outcome of each action a is not deterministic, the state transition probability between every two states that depends on action a must be constructed. We denote the probability that state s_k will lead to state s_{k+1} given action a as follows:

$$P(s_k, s_{k+1}, a) = \Pr\{s_{k+1} \mid s_k, a\}. \tag{2}$$

Due to the independence among $(b_k, \mathrm{bw}_k, v_k)$, we have

$$\begin{aligned}\Pr\{s_{k+1} \mid s_k, a\} = {} & \Pr\{b_{k+1} \mid b_k, \mathrm{bw}_k, v_k, a\} \\ & \cdot \Pr\{\mathrm{bw}_{k+1} \mid \mathrm{bw}_k\} \Pr\{v_{k+1} \mid a\}.\end{aligned} \tag{3}$$

In the right hand side of (3), the first term can be calculated as follows:

$$\Pr\{b_{k+1} \mid b_k, \mathrm{bw}_k, v_k, a\} = \begin{cases} 1, & b_{k+1} = b_a \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where b_a is the next buffer level estimated based on the current system status and action a. We calculate b_a as follows:

$$b_a = b_k + 1 - \left\lceil \frac{r_a}{\mathrm{bw}_{k+1}} \right\rceil, \tag{5}$$

where r_a is the bitrate of the target version.

When the throughput significantly drops, meaning a very low value of bw_{k+1}, b_a could be lower than zero. However, at the beginning of a new stage, there is one segment being downloaded resulting in at least one segment being always in the buffer. Therefore, (5) can be modified as follows:

$$b_a = \max\left\{ b_k + 1 - \left\lceil \frac{r_a}{\mathrm{bw}_{k+1}} \right\rceil, 1 \right\}. \tag{6}$$

The second term is easily obtained from the bandwidth model. And the third term can be simply calculated by

$$\Pr\{v_{k+1} \mid a\} = \begin{cases} 1, & v_{k+1} = a \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

Thus, expression (3) can be simplified as follows:

$$\begin{aligned}&\Pr\{s_{k+1} \mid s_k, a\} \\ &= \begin{cases} \Pr\{\mathrm{bw}_{k+1} \mid \mathrm{bw}_k\}, & b_{k+1} = b_a,\ v_{k+1} = a \\ 0, & \text{otherwise}. \end{cases}\end{aligned} \tag{8}$$

Input: number of states N,
probability function P,
cost function C
Output: optimal policy for each state μ_{s_k}
$j = 0$ // j is the iteration index
$\mu_{s_k}^j = 0,\ \forall s_k \in \mathbf{s}$ // $\mu_{s_k}^j$ is policy of state s_k at jth iteration
repeat
$j = j + 1$
Policy evaluation: compute $\lambda^j, h^j(s_k)$ using the below $N + 1$ equations:

$$h^j(s_N) = 0$$

$$\lambda^j + h^j(s_k) = C\left(s_k, s_{k+1}, \mu_{s_k}^j\right) + \sum_{r=1}^{N} P\left(s_k, s_r, \mu_{s_r}^j\right) h^j(s_r), \quad \forall s_k \in \mathbf{s}$$

Policy improvement: find for all s_k

$$\mu_{s_k}^{j+1} = \arg\min_a [C(s_k, s_{k+1}, a) + \sum_{r=1}^{N} P(s_k, s_r, a) h^j(s_r)],$$

until $\lambda^{j+1} = \lambda^j$ and $h^{j+1}(s_k) = h^j(s_k),\ \forall s_k \in \mathbf{s}$

Algorithm 1: Finding the policy set for one interval.

4.3. Cost Function. In this section, a cost function is defined to punish the situations that may cause a decrease in users' QoE. We focus on three objective parameters that affect strongly the subjective perception of the users which are quality level, video stalling, and quality switch. First, the cost function should favor the selected bitrate to be close to the current bandwidth, so it punishes the difference Δr between the current bandwidth and the bitrate of the next segment selected by action a, with

$$\Delta r = \left|\mathrm{bw}_k - r_a\right|. \tag{9}$$

Second, to prevent video stalling, the buffer level should never be underflow. We define an optimal buffer level b_{opt} that is a desired value the client should try to keep during a streaming session. When the buffer level is close to b_{opt}, the buffer underflow is avoided. Therefore, the cost function penalizes the deviation Δb of the current buffer level from the optimal buffer level, where

$$\Delta b = \left|b_k - b_{\mathrm{opt}}\right|. \tag{10}$$

Third, in order to reduce the quality switches, the cost function should contain the difference Δq between the selected quality and the last one. To punish a QoE reduction because of high quality variations, we define Δq as follows:

$$\Delta q = \left(a - v_k\right)^2. \tag{11}$$

Let α, β, γ be the trade-off parameters of three objects, namely, quality level, video stalling, and quality switch, respectively. The cost incurred when the system changes from state s_k to state s_{k+1} given action a can be calculated by

$$C\left(s_k, s_{k+1}, a\right) = \alpha\Delta r + \beta\Delta b + \gamma\Delta q. \tag{12}$$

4.4. Optimization Solution. As the system is discrete and the number of states is large, we can formulate an infinite horizon problem. For every state s_k, the most appropriate action a, called policy for state s_k, has to be decided so that the mean cost per state is minimum. As mentioned above, our system has I system state sets corresponding to I bitrate intervals, so we have to find I corresponding policy sets. For simplicity, we only present the optimization solution for a general bitrate interval with the system state set $\mathbf{s}$ and a corresponding policy set $\boldsymbol{\mu}$. Finding optimal solutions for all bitrate intervals will be done similarly. Mathematically, we have to minimize C_A which is the average cost per state obtained after downloading L video segments. C_A is calculated as follows:

$$C_A = \lim_{L\to\infty} \frac{1}{L} \sum_{l=1}^{L} P\left(s_k, s_{k+1}, a\right) C_l\left(s_k, s_{k+1}, a\right), \tag{13}$$

with $C_l(s_k, s_{k+1}, a)$ being the cost incurred after downloading the lth segment. Here, L is the number of state transitions and is also the number of video segments in the session. Based on the standard policy iteration algorithm (PIA) [26], we solve the IHP problem by using an algorithm as presented in Algorithm 1.

Applying PIA of SDP for I bitrate intervals, we would generate I policy sets which serve like a look-up table mapping each state to an optimal action. Thus, the client is able to decide an appropriate version for the next segment based on the current system condition.

Let $C_{\mathrm{prob}}, C_{\mathrm{cost}}$ be the computational complexity of the calculation of transition probability and the cost from one state to the remaining states, respectively. Let C_{PIA} be the computational complexity of Algorithm 1. The complexity of our model is $C = C_{\mathrm{prob}} + C_{\mathrm{cost}} + C_{\mathrm{PIA}}$. For each interval, each action, and each state, we consider the cost and the transition probability to $N-1$ remaining states. So the complexity of the calculation of transition probability and the cost is described as follows:

$$C_{\mathrm{prob}} = C_{\mathrm{cost}} = O\left(IV_{\max}N^2\right). \tag{14}$$

Based on [27], we have

$$C_{\text{PIA}} = O\left(IV_{\max}^{N}\right). \tag{15}$$

Therefore, the complexity of our model is

$$C = O\left(2IV_{\max}N^{2} + IV_{\max}^{N}\right). \tag{16}$$

5. Performance Prediction

After Section 4, we achieve I policy sets corresponding to I intervals of a video bitrate. In this section, we use the Markov chain model to predict the streaming performance for a session. Similar to Section 4.4, this section only presents the performance prediction for a general bitrate interval with a system state set $\mathbf{s}$ and a corresponding policy set $\boldsymbol{\mu}$. Predicting performance for all bitrate intervals will be done similarly. After carrying out the calculation for all I bitrate intervals, we take the average values as the final results.

The key is to determine the average state probability $\mathbf{p}_a = [p_{a1}, p_{a2}, \ldots, p_{aN}]$ with p_{an} $(1 \le n \le N)$ being the average probability that the system is at the nth state throughout the streaming session. The probability $\mathbf{p}_a$ is the average value of state probabilities after downloading l segments $\mathbf{p}_l = [p_{l1}, p_{l2}, \ldots, p_{lN}]$, $(1 \le l \le L)$. Here, p_{ln} $(1 \le n \le N)$ is the probability that the system is at the nth state after downloading l segments. From the Markov chain theory, the state probability after downloading $l+1$ segments can be computed as the product of the state probability after downloading l segments and a transition matrix $\mathbf{p}_{l+1} = \mathbf{p}_l\mathbf{P}_{\text{TS}}$. Assuming that the initial probability p_0 is known. The transition matrix $\mathbf{P}_{\text{TS}}$ is a $N \times N$ matrix which represents the transition probability from state s_k to state s_{k+1}. $\mathbf{P}_{\text{TS}}$ is defined as follows:

$$\mathbf{P}_{\text{TS}} = P\left(s_k, s_{k+1}\right) = \Pr\left\{s_{k+1} \mid s_k, a \in \boldsymbol{\mu}\right\}. \tag{17}$$

The average state probability $\mathbf{p}_a$ is calculated as follows.

$$\mathbf{p}_a = \frac{\sum_{l=1}^{L} \mathbf{p}_l\mathbf{P}_{\text{TS}}}{L}. \tag{18}$$

Currently, most of the adaptation algorithms developed for HTTP streaming are qualitative in the sense that the performance metrics could only be obtained after the streaming session. In this study, the predicted streaming performance could be calculated based on the average state probability and the information inside every state. Specifically, we mainly focus on the following aspects: bitrate prediction, quality switch prediction, and buffer prediction.

5.1. Quality Prediction. The video quality that the users perceive is presented through the selected version. The higher the version is selected, the better the video quality is perceived by the users. Furthermore, setting the maximum version for the streaming session also affects the perceptual quality of the users. Obviously, a very low value of the maximum version may result in a poor perceptual quality while a very high value may increase the chance of buffer underflow. In this section, we predict the quality performance of the streaming session based on the average version A_v that is calculated using (19) and the cumulative distribution function (CDF) of the versions $f(v)$ $(v \in [1; V\max])$ that is shown in (20). Based on the predicted probability of the versions throughout a streaming session, the maximum version could be decided for the session

$$A_v = \frac{\sum_{n=1}^{N} v_n p_{an}}{N}, \tag{19}$$

$$f(v) = \frac{\sum_{n=1}^{N} v_n p_{an}}{N} \mid v = v_n. \tag{20}$$

5.2. Quality Switch Prediction. Quality switch is an important factor affecting the perception of the users. The users often expect a smooth playback with the minimum number of quality switches and small switch amplitude from one segment to the next. We can predict the average version switch per segment A_{sw} as follows:

$$A_{\text{sw}} = \frac{\sum_{n=1}^{N} \left|a_n - v_n\right| p_{an}}{N}, \quad a_n \in \boldsymbol{\mu}. \tag{21}$$

5.3. Buffer Prediction. Video stalling is one of the important objective parameters that affect the subjective perception of the users. Stalling occurs when the play-out buffer underruns. To prevent this event, the buffer must be maintained within a safe range. In this session, we evaluate the buffer performance through the average buffer level A_b, the CDF of the buffer level $\widehat{f}(b)$ $(b \in [1; B\max])$, and the buffer underflow probability $\Pr_{\text{und}}$ (i.e., when the system stays at buffer level 1) which are described as follows:

$$\begin{aligned} A_b &= \frac{\sum_{n=1}^{N} b_n p_{an}}{N}, \\ \widehat{f}(b) &= \frac{\sum_{n=1}^{N} b_n p_{an}}{N} \mid b_n < b, \\ \Pr_{\text{und}} &= \frac{\sum_{n=1}^{N} p_{an}}{N} \mid b_n = 1. \end{aligned} \tag{22}$$

A_b represents the safety of the buffer. If A_b is small, the buffer level is often in low levels, which may cause playback interruption when the current bandwidth drops dramatically. $\widehat{f}(b)$ reflects the variance of the buffer level under the fluctuations of the network. $\Pr_{\text{und}}$ shows the probability that the playback would be interrupted in the streaming session.

6. Experimental Results

In order to evaluate the proposed system model and performance prediction accuracy, in this section, we perform a number of experiments in both offline and online contexts and compare the performance predicted results with the measurement ones. We also compare our proposed method with two existing ones, namely, the SDP method presented [10] which could obtain the best performance among the SDP methods and the bitrate estimation based method presented [14], which is the best among the qualitative methods.

6.1. Experiment Setup. For the simulation, our test-bed consists of a client running Java 8.0 which implements the adaptation and a server running Apache2 which holds the media segments. The client runs on a Window 7 computer with an Intel i5-1.7 GHz CPU and 4 GB memory and the server runs on Ubuntu 12.04LTS (with default TCP CUBIC) with 1 G RAM. The channel bandwidth is simulated using DummyNet [28]. We use the Tokyo Olympic video from [6]. For the video, $V\max = 9$, and, for the bandwidth, $W = 10$. Since we measure the buffer size and compute the buffer cost in the segment duration unit, we implement our adaptation method with one setting of the segment duration. In our experiments, we select a segment duration of 2 seconds, which is similar to those of [7, 8, 10]. The impacts of segment durations on adaptation performance have been considered in some recent studies [29, 30]. Further evaluation of different segment durations with a fixed buffer size (in seconds) will be reserved for our future work. Maximum buffer level is set to 5 segments (i.e., 10 seconds) and optimal buffer level is set to 4 segments (i.e., 8 seconds).

In our method, a streaming provider can adjust the balance between the requirements for high quality level, preventing video stalling, and reducing quality switches by changing the trade-off parameters α, γ, and β. Since selecting an optimal combination of trade-off parameters of the cost function involves solving a hard optimization problem, it will be investigated in our future work. In this paper, we select qualitatively the trade-off parameters of cost function as follows. Initially, we fix β and select the other two parameters. Since we want to prioritize requirement of smooth quality switch, γ is selected so that the contribution of the quality switch cost Δq is higher than that of buffer cost Δb. With parameter α, because the bitrate cost Δr can be up to thousands, parameter α should be small to reduce the contribution of the bitrate cost to the overall cost C. Based on our experience, good empirical values on parameters α, β, and γ are 0.003, 4, and 20, respectively.

6.2. Experimental Results. In the first part of the experiment, we evaluate the accuracy of performance prediction in offline context using a given bandwidth trace obtained from a mobile network [12]. In this context, the number of video segments L is 300.

Figures 5 and 6 show the predicted performance using the formulas presented in Section 5 and the measured performance obtained from the experiments when the maximum allowed version is set to 7, 8, and 9. These figures point out that the prediction results are close to the measurement ones. We can see from Figure 5 that when the maximum version increases the average version as well as the average version switch also increases. Figure 6 shows that, in both prediction and measurement cases, when the maximum version is 7, the underflow probability is almost zero and increases very slowly when the maximum allowed version increases. This analysis implies that setting the maximum version to 8 is reasonable in terms of balancing between average video quality and quality switch; meanwhile setting the maximum version to 7 ensures a very stable streaming experience.

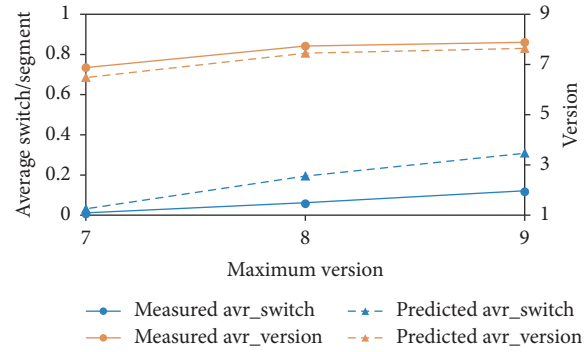

Figure 5: Predicted performance and measured performance in terms of average version and average version switch per segment in offline context using a given bandwidth trace.

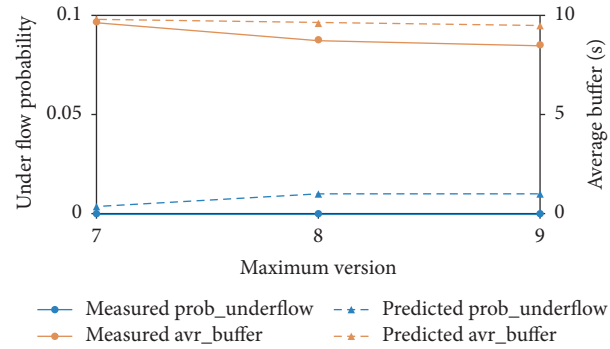

Figure 6: Predicted performance and measured performance in terms of average buffer and underflow probability in offline context using a given bandwidth trace.

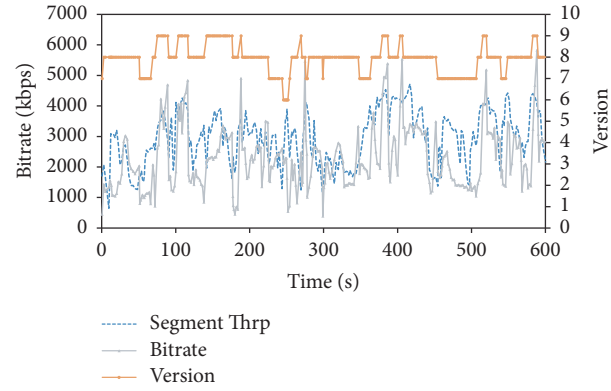

Figure 7: Experimental results of proposed method in offline context using a given bandwidth trace with $V\max = 9$.

Figures 7, 8, and 9 show the bitrate and version switch behavior of the proposed method in the three cases of maximum version. It can be seen very clearly from these figures that when the maximum version is reduced, the number of version switches (or quality changes) decreases.

Table 1 shows more detailed statistics of the experimental results in three cases of maximum version. It can be drawn

Table 1: Compare predicted performance and measured performance in offline context using a given bandwidth trace.

Statistics	$V_{max} = 9$		$V_{max} = 8$		$V_{max} = 7$	
	Predicted	Measured	Predicted	Measured	Predicted	Measured
$A_b(s)$	9.48	8.52	9.59	8.77	9.8	9.68
A_q	7.64	7.88	7.45	7.73	6.48	6.87
A_{sw}	0.30	0.18	0.14	0.09	0.03	0.02
$Pr_{undflow}$	0.01	0.00	0.006	0.00	0.00	0.00

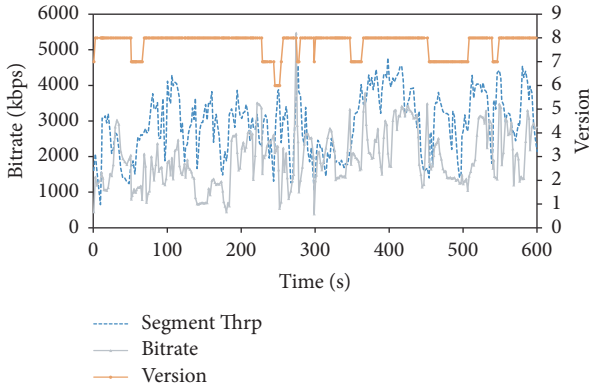

Figure 8: Experimental results of proposed method in offline context using a given bandwidth trace with Vmax = 8.

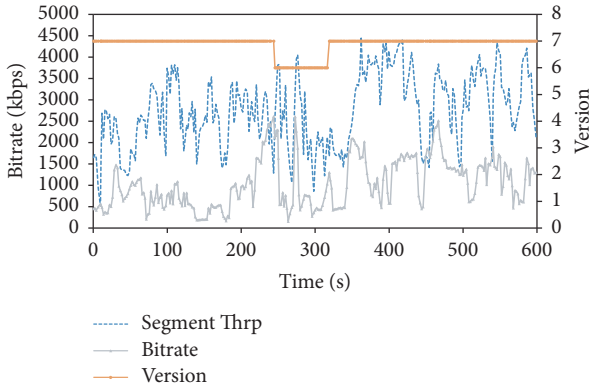

Figure 9: Experimental results of proposed method in offline context using a given bandwidth trace with Vmax = 7.

from the table that there is no significant difference between the predicted performance and the measured performance.

In the second part of the experiment, we use two history bandwidth traces recorded from two previous streaming sessions of the client. The CDFs of both bandwidths are shown in Figure 10.

Bandwidth bw_1 is used for calculating the statistical models and bw_2 is used in the simulation for measuring performance parameters. Figures 11, 12, and 13 show the bitrate and version switch behavior of the proposed method in the three cases of maximum version. The detailed results are listed in Table 2. Based on these figures and table, we affirm once again that the mathematical performance prediction model agrees well with the measurement.

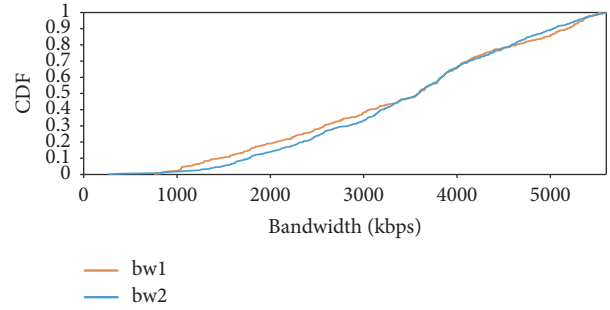

Figure 10: The cumulative distribution function of bw1 and bw2.

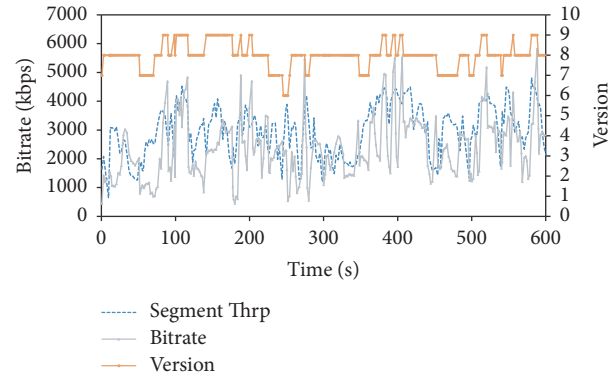

Figure 11: Experimental results of proposed method in offline context using a history bandwidth trace with Vmax = 9.

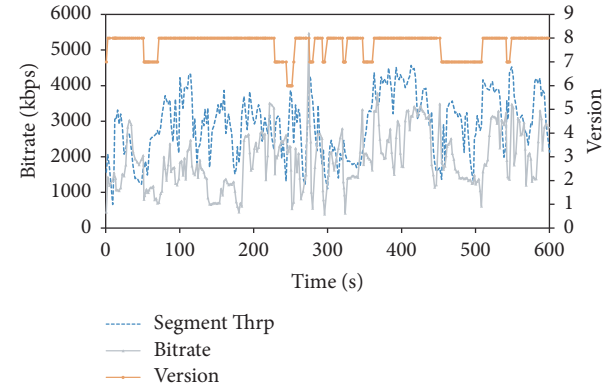

Figure 12: Experimental results of proposed method in offline context using a history bandwidth trace with Vmax = 8.

Table 2: Compare predicted performance and measured performance in offline context using a history bandwidth trace for statistics.

Statistics	$V_{max} = 9$		$V_{max} = 8$		$V_{max} = 7$	
	Predicted	Measured	Predicted	Measured	Predicted	Measured
$A_b(s)$	9.52	8.44	9.61	8.85	9.69	9.65
A_q	7.48	7.92	7.27	7.71	6.80	6.98
A_{sw}	0.31	0.19	0.18	0.11	0.08	0.05
$Pr_{undflow}$	0.00	0.00	0.00	0.00	0.00	0.00

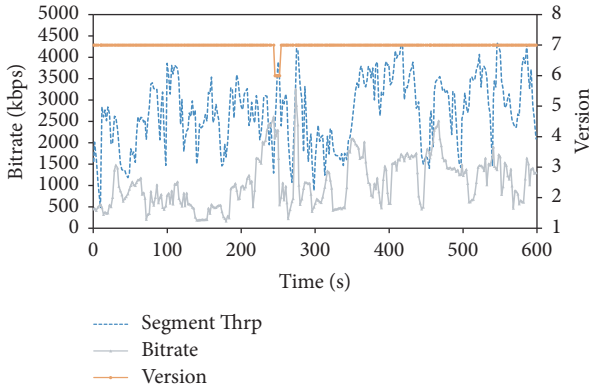

Figure 13: Experimental results of proposed method in offline context using a history bandwidth trace with Vmax = 7.

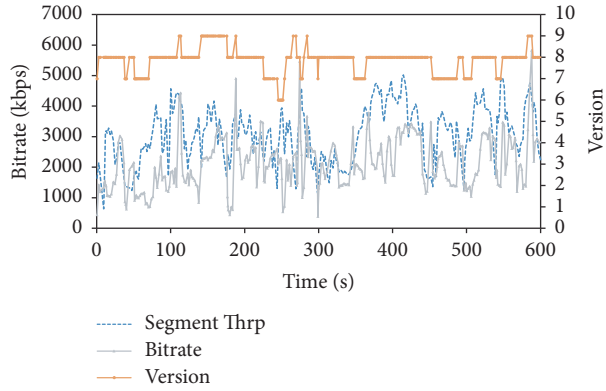

Figure 14: Experimental results of the proposed method in online context with Vmax = 9.

In the third part of the experiment, we consider the online context in which the prediction of the future bandwidth is based on the statistical parameters of all previous segments. Specifically, we divide an entire session into chunks, each of which has L video segments. We treat each chunk individually as a mini streaming session. Assuming that, initially we have enough statistical data to predict the performance of the first chunk. The prediction of the subsequent chunks will be done based on the previous chunks. At the beginning of each chunk, the bandwidth statistics are updated leading to a recomputation of the policy set and the performance. In the experiment, we set L to 100 video segments. It can be observed from our experiments that it took only several seconds to (re)compute the whole model. Therefore, the computational overhead is about 3%, which is appropriate for the online context. Figures 14, 15, and 16 show the adaptation, bitrate, and version switch behavior of the proposed method in the three cases of maximum version. The predicted performance and the measured performance in the online context in the three cases are presented in detail in Tables 3, 4, and 5. It is very clear that, in the online context, the predicted performance is also very close to the measured performance.

Next, we compare our proposed method with the SDP method [10] and the bitrate estimation based method [14] using the same simulation settings as in the first part of our experiment. The experimental results obtained by simulating the SDP method [10] and bitrate estimation based method [14] are shown in Figures 17 and 18, respectively. We can see that both methods provide very fluctuating version switches. The detailed statistics of these adaptation methods with the maximum version set to 9 and our proposed method with the

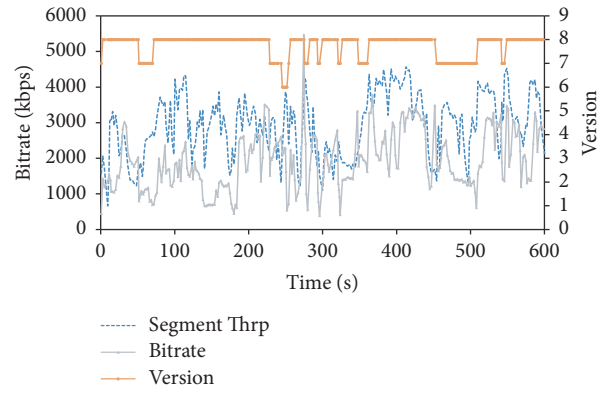

Figure 15: Experimental results of the proposed method in online context with Vmax = 8.

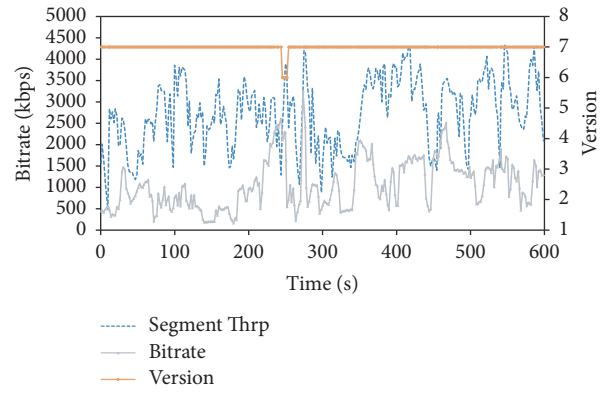

Figure 16: Experimental results of the proposed method in online context with Vmax = 7.

Table 3: Compare predicted performance and measured performance in online context with $V_{max} = 9$.

Statistics	Chunk 1		Chunk 2		Chunk 3	
	Predicted	Measured	Predicted	Measured	Predicted	Measured
$A_b(s)$	9.31	8.96	8.92	8.45	9.36	8.86
A_q	7.69	8.04	7.89	7.68	7.83	7.72
A_{sw}	0.05	0.09	0.04	0.07	0.02	0.05
$Pr_{undflow}$	0.00	0.00	0.00	0.00	0.00	0.00

Table 4: Compare predicted performance and measured performance in online context with $V_{max} = 8$.

Statistics	Chunk 1		Chunk 2		Chunk 3	
	Predicted	Measured	Predicted	Measured	Predicted	Measured
$A_b(s)$	9.53	9.01	9.42	8.63	9.42	8.96
A_q	7.56	7.86	7.65	7.63	7.65	7.67
A_{sw}	0.00	0.03	0.01	0.03	0.02	0.04
$Pr_{undflow}$	0.00	0.00	0.00	0.00	0.00	0.00

Table 5: Compare predicted performance and measured performance in online context with $V_{max} = 7$.

Statistics	Chunk 1		Chunk 2		Chunk 3	
	Predicted	Measured	Predicted	Measured	Predicted	Measured
$A_b(s)$	9.77	9.93	9.77	9.52	9.83	9.59
A_q	6.92	7.00	6.92	6.95	6.92	7.00
A_{sw}	0.00	0.00	0.00	0.02	0.00	0.00
$Pr_{undflow}$	0.00	0.00	0.00	0.00	0.00	0.00

Table 6: Statistics of different adaptation methods in offline context.

Statistics	SDP method $V_{max} = 9$	Bitrate estimation method $V_{max} = 9$	Proposed method $V_{max} = 9$	Proposed method $V_{max} = 8$	Proposed method $V_{max} = 7$
$A_b(s)$	8.62	8.85	8.52	8.77	9.68
A_q	7.40	7.85	7.88	7.73	6.87
A_{sw}	0.76	0.43	0.18	0.09	0.02
$Pr_{undflow}$	0.00	0.00	0.00	0.00	0.00

maximum version set to 7, 8, and 9 are shown in Table 6. It can be seen that the performance of the bitrate estimation method is less effective than our method in terms of average version and average version switch. Regarding the SDP method, it is evident that the performance of this method is the worst among all (with low average version and highest average switch per segment). This can be expected because this method was not originally designed for real VBR videos.

The buffer level curves of the three methods in offline context are shown in Figure 19. We can see that all buffer curves imply streaming sessions without any freezes for users. Among these, the SDP method and the proposed method with $V_{max} = 9$ provide the most unstable buffers, while the bitrate estimation based method and the proposed method with $V_{max} = 7$ provide the most stable buffers. It can be observed from Figures 17 and 18 and Table 6 that the SDP method and bitrate estimation based method result in very fluctuating version curves, and the proposed method with

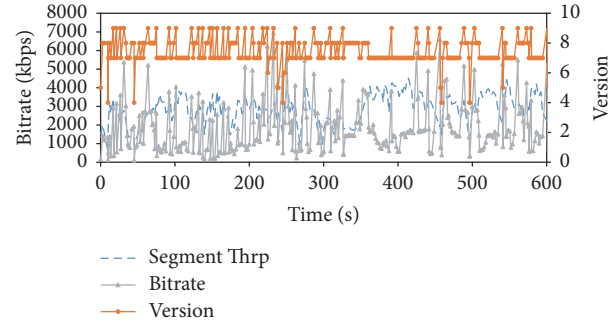

Figure 17: Experimental results of SDP method.

$V_{max} = 7$ obtains the lowest average quality. Therefore, from Table 6 and Figure 19 we can see that the proposed method in offline context with $V_{max} = 8$ or 9 can provide the best performance.

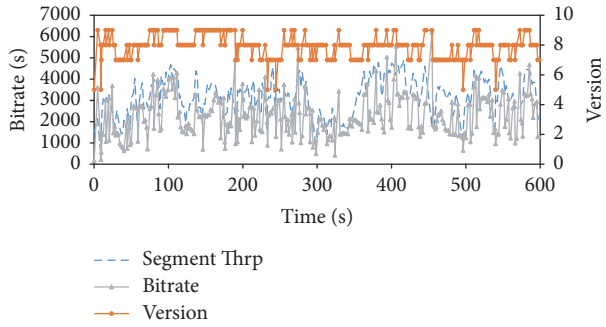

FIGURE 18: Experimental results of bitrate estimation based method.

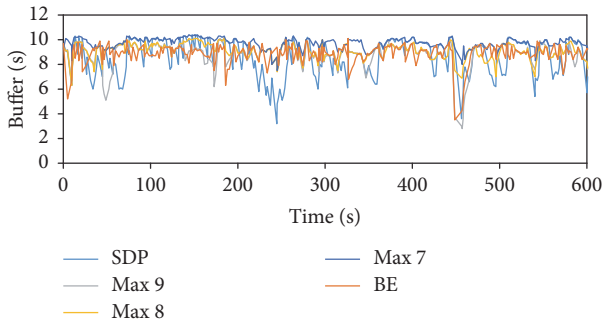

FIGURE 19: Buffer level curves of SDP method, bitrate estimation based method, and the proposed method (Max9, Max8, and Max7) in offline context.

7. Conclusion

In this paper, we have proposed an adaptation method for HTTP streaming based on stochastic dynamic programming. The system model was targeted at real bandwidth trace with strong bitrate fluctuation of VBR videos. Furthermore, we have developed a model to predict the system performance with the aim of choosing the best setting based on the performance requirements. The experimental results have shown that our method can effectively adapt VBR videos and perform accurate performance prediction which is useful in planning adaptation policy.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Cisco Systems, Inc., "Networking index: Forecast and methodology, 2012–2017," Cisco Visual, May 2013.

[2] A. C. Begen, T. Akgul, and M. Baugher, "Watching video over the web: part 1: streaming protocols," *IEEE Internet Computing*, vol. 15, no. 2, pp. 54–63, 2011.

[3] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 78–85, 2012.

[4] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceeding of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, pp. 133–144, San Jose, CA, USA, February 2011.

[5] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *Proceedings of the 19th International Packet Video Workshop (PV '12)*, pp. 173–178, IEEE, Munich, Germany, May 2012.

[6] Y. Zhou, Y. Duan, J. Sun, and Z. Guo, "Towards simple and smooth rate adaption for VBR video in DASH," in *Proceedings of the IEEE Visual Communications and Image Processing Conference (VCIP '14)*, pp. 9–12, IEEE, Valletta, Malta, December 2014.

[7] 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, 2012.

[8] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video (MoVid '12)*, pp. 37–42, New York, NY, USA, February 2012.

[9] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proceedings of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, pp. 169–174, San Jose, Calif, USA, February 2011.

[10] S. García, J. Cabrera, and N. García, "Quality-control algorithm for adaptive streaming services over wireless channels," *IEEE Journal on Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 50–59, 2015.

[11] A. H. Duong, T. Nguyen, T. Vu, T. T. Do, N. P. Ngoc, and T. C. Thang, "SDP-based adaptation for quality control in adaptive streaming," in *Proceedings of the International Conference on Computing, Management and Telecommunications, (ComManTel '15)*, pp. 194–199, IEEE, DaNang, Vietnam, December 2015.

[12] S. Lederer, C. Mueller, C. Timmerer, C. Concolato, J. Le Feuvre, and K. Fliegel, "Distributed DASH dataset," in *Proceedings of the 4th ACM Multimedia Systems Conference, (MMSys '13)*, pp. 131–135, Oslo, Norway, March 2013.

[13] "H264/AVC video trace library," http://trace.eas.asu.edu/h264/.

[14] T. C. Thang, H. T. Le, H. X. Nguyen, A. T. Pham, J. W. Kang, and Y. M. Ro, "Adaptive video streaming over HTTP with dynamic resource estimation," *IEEE Trans. On consumer electronics*, vol. 58, no. 1, pp. 78–85, 2012.

[15] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "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.

[16] R. Dubin, O. Hadar, and A. Dvir, "The effect of client buffer and MBR consideration on DASH Adaptation Logic," in *Proceedings of the 2013 IEEE Wireless Communications and Networking Conference, (WCNC '13)*, pp. 2178–2183, IEEE, Shanghai, China, April 2013.

[17] G. Liang and B. Liang, "Effect of delay and buffering on jitter-free streaming over random VBR channels," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1128–1141, 2008.

[18] Y. Kang, H. Chen, and L. Xie, "An artificial-neural-network-based QoE estimation model for Video streaming over wireless networks," in *Proceedings of the 2013 IEEE/CIC International Conference on Communications in China, (ICCC '13)*, pp. 264–269, IEEE, Xi'an, China, August 2013.

[19] S. Xiang, L. Cai, and J. Pan, "Adaptive scalable video streaming in wireless networks," in *Proceedings of the 3rd ACM Multimedia*

Systems Conference, (MMSys '12), pp. 167–172, New York, NY, USA, February 2012.

[20] Y. Liu and J. Y. B. Lee, "On adaptive video streaming with predictable streaming performance," in *Proceedings of the 2014 IEEE Global Communications Conference, (GLOBECOM '14)*, pp. 1164–1169, IEEE, Austin, TX, USA, December 2014.

[21] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proceedings of the 2nd Annual ACM Multimedia Systems Conference (MMSys '11)*, pp. 157–168, San Jose, CA, USA, February 2011.

[22] G. Ji and B. Liang, "Stochastic rate control for scalable VBR video streaming over wireless networks," in *Proceedings of the 2009 IEEE Global Telecommunications Conference, (GLOBECOM '09)*, Honolulu, HI, USA, December 2009.

[23] M. Xing, S. Xiang, and L. Cai, "Rate adaptation strategy for video streaming over multiple wireless access networks," in *Proceedings of the 2012 IEEE Global Communications Conference, (GLOBECOM '12)*, pp. 5745–5750, IEEE, Anaheim, CA, USA, December 2012.

[24] Z. Ye, R. EL-Azouzi, T. Jimenez, and Y. Xu, "Computing The Quality of Experience in Network Modeled by a Markov Modulated Fluid Model," 2014.

[25] 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, 2014.

[26] H. Thomas, E. Cormen Charles, L. Ronald, and S. Clifford, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 3rd edition, 2009.

[27] Y. Mansour and S. Singh, "On the complexity of policy iteration," *UAI*, pp. 401–408, 1999.

[28] L. Rizzo, "Dummynet: a simple approach to the evaluation of networkprotocols," *ACM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[29] D. M. Nguyen, L. B. Tran, H. T. Le, N. P. Ngoc, and T. C. Thang, "An evaluation of segment duration effects in HTTP adaptive streaming over mobile networks," in *Proceedings of the 2nd National Foundation for Science and Technology Development Confernce on Information and Computer Science, (NICS '15)*, pp. 248–253, Ho Chi Minh City, Vietnam, September 2015.

[30] L. Koskimies, T. Taleb, and M. Bagaa, "QoE estimation-based server benchmarking for virtual video delivery platform," in *Proceeding of IEEE International Conference on Communications (ICC '17)*, Paris, France, May 2017.