Adaptive Scalable Video Streaming in Wireless Networks

Siyuan Xiang
University of Victoria
siyxiang@ece.uvic.ca

Lin Cai
University of Victoria
cai@uvic.ca

Jianping Pan
University of Victoria
pan@uvic.ca

ABSTRACT

In this paper, we investigate the optimal streaming strategy for dynamic adaptive streaming over HTTP (DASH). Specifically, we focus on the rate adaptation algorithm for streaming scalable video (H.264/SVC) in wireless networks. We model the rate adaptation problem as a Markov Decision Process (MDP), aiming to find an optimal streaming strategy in terms of user-perceived quality of experience (QoE) such as playback interruption, average playback quality and playback smoothness. We then obtain the optimal MDP solution using dynamic programming. We further define a reward parameter in our proposed streaming strategy, which can be adjusted to make a good trade-off between the average playback quality and playback smoothness. We also use a simple testbed to validate our solution. Experiment results show the feasibility of the proposed solution and its advantage over the existing work.

Categories and Subject Descriptors

C.2.5 [**Local and Wide-Area Networks**]: Internet; H.5.1 [**Multimedia Information Systems**]: Video

General Terms

Design, Performance, Experimentation

Keywords

Adaptive Video Streaming, Scalable Video Coding

1. INTRODUCTION

Progressive download is currently one of the most popular video delivery techniques on the Internet. It has several advantages over the traditional streaming techniques using RTSP/UDP. First, it is simple to deploy. At the server side, any web server can host videos and serve as a streaming server; at the client side, the user only needs a flash player or web browser supporting HTML5 for video playback. Second, the HTTP/TCP protocols used in progressive download are more firewall and NAT friendly, and the congestion control mechanism in TCP simplifies the design of the application layer. Third, for progressive download, a server can store several versions of a video to meet the requirements of heterogeneous users. Ideally, a user can select the right version of the video according to the device decoding capability, display size and available network bandwidth.

However, selecting the appropriate version of a video according to the available bandwidth may not be easy for users and their decisions might be error-prone. In addition, with progressive download, the client always downloads as much video data as possible. It is likely that when a user turns off the video player or switches to another video, a large amount of un-watched video is buffered unnecessarily, which wastes the resources of both the network and the end-systems.

Dynamic adaptive streaming over HTTP (DASH) [13] is a promising technique to overcome the aforementioned disadvantages of progressive download. Videos encoded in different versions are chopped into small segments. After the client receives one segment, it has a chance to decide which version of the video to request for the next segment, based on the current network condition. Thus, rate adaptation can be performed at the client side naturally and flexibly. Also, the client has a chance to control the client-side queue length to avoid streaming buffer overflow, e.g., when the download rate is much higher than the playback rate.

Currently, commercial adaptive streaming products such as Microsoft Smooth Streaming and Apple Live Streaming use single-layer H.264/AVC encoded videos. Multiple versions of a video with different resolution, frame rate and quality are obtained by encoding the source video multiple times with different configurations, and the different versions of the video are completely independent to each other. Thus, not only more server storage space is needed, but also the web caching hit-ratio is reduced.

Recently, scalable video coding (H.264/SVC) has been introduced to the DASH framework to improve the system performance [11]. With SVC, a video is encoded once only but can be decoded many times with different resolution, frame rate and quality. However, how to improve the rate adaptation algorithm to provide users with a satisfactory quality of experience (QoE) is still a challenging, open question. The problem is even more challenging when a user uses a handheld device and a wireless access link for video streaming, as the handheld devices typically have limited energy supply and computation capacity, and the wireless links are

MMSys '12, February 22-24, 2012, Chapel Hill, North Carolina, USA.

highly dynamic due to the time-varying fading, shadowing, interference and hand-off, all of which motivate this work.

In this paper, we investigate the optimal strategy for streaming scalable video over HTTP in wireless networks. The main contributions of this paper are twofold. First, we formulate the rate adaptation problem as a finite Markov Decision Process (MDP), aiming to find an optimal streaming strategy in terms of user-perceived QoE such as playback interruption, average playback quality and playback smoothness. We obtain the optimal streaming strategy by dynamic programing under the reinforcement learning framework [14]. We further define a reward parameter in our proposed strategy, which can be adjusted to make a good tradeoff between average playback quality and playback smoothness. Second, we evaluate the proposed streaming strategy and compare it with the existing work using a testbed with a real sample video encoded by the SVC reference codec, JSVM [10]. The experiment results show the advantage of our proposed solution.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 formulates the optimal streaming problem as an MDP and describes the proposed solution based on the reinforcement learning framework. The testbed implementation and the experiment results are described and given in Section 4, followed by concluding remarks and further research issues in Section 5.

2. BACKGROUND AND RELATED WORK

Different from the application layer multicasting [7], in a DASH system, rate adaptation is conducted at the client side, which is also called pull-based rate adaptation [3]. At the server side, a source video is encoded into different versions with different resolution, frame rate and quality. For each version, the video is divided into small segments. A web server can host these segments and send them to the clients upon HTTP requests. At the client side, after a user clicks the play button, the streaming starts. The video player first obtains the general information of the video, such as the number of versions and the corresponding resolution, frame rate and quality of each version. Then, the video player will decide the right version according to its own display size, decoding capability and network condition. Usually, the playback does not start until a sufficient number of segments are received. After the client receives a segment completely, the rate adaptation algorithm will decide which version to request for the next segment based on the current network condition and the client-side state such as the number of buffered segments. In this way, the workload of the server is reduced dramatically. Fig. 1 shows the general work flow of the video player.

There are extensive research efforts on adaptive video streaming over HTTP [13, 8, 2]. [13] introduced the 3GPP specification of dynamic adaptive streaming over HTTP, which describes the framework of the adaptive streaming system. In [2], commercial adaptive streaming products including Microsoft Smooth Streaming, Netflix player and open source media framework (OSMF) player were evaluated and compared. The results show that the performance of these products still needs to be improved substantially.

Liu et al. proposed a rate adaptation algorithm for adaptive video streaming [8]. The decision of switching to a video version of a higher or lower bit-rate is made based on the measured segment fetch time, which can be converted to

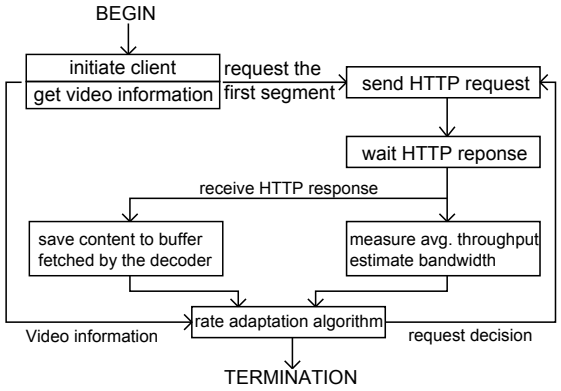


Figure 1: Video Player State Diagram

the average segment throughput and buffer state. The algorithm is evaluated using constant bit-rate (CBR), single-layer video traffic only, and the queue length may sometimes exceed the maximum buffer size. In [6], a quality adaptation controller based on the feedback control theory was proposed. The controller tries to maintain the buffer level as stable as possible to match the video bit-rate with the available bandwidth. As the server needs to maintain the information for each user to perform rate adaptation, the complexity of the server is increased.

Recently, SVC has been introduced to adaptive video streaming. With SVC, we can encode video once and decode the bitstream multiple times with different resolution, frame rate and video quality [5], so the server storage space and encoding time can be saved. In addition, thanks to the layered structure of SVC, we may even upgrade an already received segment to a higher quality [12]. [11] showed the advantage of using SVC in adaptive HTTP streaming over the single-layer advanced video coding (AVC) in terms of caching efficiency. In [12], the authors proposed a priority-based media delivery strategy using SVC with RTP and HTTP streaming. In the pre-buffering phase, the most important base layer is transmitted first, so there are more base-layer frames than enhancement-layer frames in the buffer. This scheme was designed assuming that the temporary bandwidth reduction is the only possible bandwidth variation, and the bandwidth will restore to a normal level after the temporary reduction. Thus, it cannot fully handle the random variation of network bandwidth.

Different from these existing approaches, in this paper, we focus on the rate adaptation algorithm for streaming SVC video in wireless networks, considering the random and less predictable variation of the available bandwidth. We also consider the more general case where the layered video is encoded in variable bit-rate (VBR).

3. PROBLEM FORMULATION

Considering the limited computation capacity of handheld devices and the high variation of wireless access links, we formulate the optimal rate adaptation problem as a finite Markov Decision Process, which can deal with the random network condition with a relatively simple approach that is feasible for handheld devices. For each video segment, the client uses MDP to make a decision on which action to conduct given the current client state. There are four com-

ponents for MDP, i.e., action, state, transition probability and reward. In the following, we define them one by one.

As shown in Figure 1, after a segment is obtained completely, the rate adaptation algorithm has a chance to decide the video version of the next segment to be requested and whether the client should be idle for a while to avoid buffer overflow. We define the sequential actions as $\{a_t\}(t = 0, 1, \cdots)$. a_t is the decision made at step t, where the step duration equals the time to retrieve one segment. Note that the step duration is not a constant, since the segment download time varies according to the segment size and the available bandwidth. L is the number of versions. The action set for a given state is $\mathcal{A}(s) = \{A_i, A_u, A_w\}$, where $A_i(i = -L+1, \cdots, L-1)$ means to request the next segment with i layer higher ($i \geq 0$) or lower ($i < 0$) than the current one, A_u means to "upgrade" the last received segment to a higher version, and A_w means to wait for a time duration of T_s (T_s is the constant playback time of a segment).

We define a state at step t as $s_t = (q_t, \Delta q_t, v_t, \Delta v_t, bw_t, d_t)$. Here, q_t is the queue length in terms of the number of buffered frames. Obviously, q_t is in the range of $(0, F)$, where $F = B_T \times N_s$, B_T is the target buffer size in terms of the number of segments, and N_s is the number of frames per segment. Δq_t is the queue length variation after a new segment has been retrieved, i.e., $\Delta q_t = q_t - q_{t-1}$, which indicates whether the requested video's bit-rate matches the available bandwidth. Δq_t is in the range of $[-F, N_s]$. v_t is the version index of the last received segment. Δv_t indicates the difference of video versions requested in consecutive steps. bw_t is the available bandwidth at step t. d_t is the number of received segments, which is in the range of $[0, N_T]$, where N_T is the total number of segments the client needs to request.

From the definition of the states, we can observe that the Markov property exists, since all of these states depend on their immediately previous state only, i.e.,

$$\Pr\{s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \cdots, s_0, a_0\} = \Pr\{s_{t+1}|s_t, a_t\}.$$

To obtain the state transition probability, the most challenging issue is to obtain the model for bw_t. For most wireless streaming scenarios, the bottleneck is often in the wireless access link, and the finite-state Markov chain has been widely used to model the variation of wireless channels [15, 17]. Thus, we use a discrete-time finite-state Markov model to capture the variation of the bandwidth, with the state transition probabilities obtained from the measurement or derived from the wireless channel model [15]. Given the available bandwidth for downloading the current segment, we can estimate the probability distribution of the bandwidth for the next segment using the state transition probability matrix of the Markov model.

For the problem of our interest, we can derive the state transition probability for the MDP by

$$\mathcal{P}^a_{ss'} = \Pr\{s_{t+1} = s'|s_t = s, a_t = a\}. \quad (1)$$

The state at step t is $s = (q, \Delta q, v, \Delta v, bw, d)$. If action $a_t = A_i$ is selected, then with probability $\mathcal{P}^a_{ss'} = \Pr\{bw'|bw\}$, the new state will be $s' = (q', \Delta q', v', \Delta v', bw', d')$, i.e.,

$$\begin{aligned} v' &= v + i, \;\; \Delta v' = i, \quad (2)\\ q' &= q - \left\lceil (m^{v'}_{d+1} \times f)/bw' \right\rceil + N_s,\\ \Delta q' &= q' - q, \;\; d' = d + 1, \end{aligned}$$

where $m^{v'}_{d+1}$ is the size of version v' of segment $d+1$ and f is the playback frame rate (since we are dealing with the stored video streaming, the client can have the knowledge of the size of every segment). If $a_t = A_u$, then the new state is

$$\begin{aligned} v' &= v + 1, \;\; \Delta v' = \Delta v + 1, \quad (3)\\ q' &= q - \left\lceil [(m^{v'}_d - m^v_d) \times f]/bw' \right\rceil,\\ \Delta q' &= q' - q, \;\; d' = d. \end{aligned}$$

Similarly, we can derive other state transition probabilities.

The reward in MDP is the payoff obtained when a particular action is taken at a state,

$$r_{t+1} = R(s_t = s), \quad (4)$$

where R maps the state to a reward. Table 1 lists the rewards defined for different states. $*$ means any value for the state, F^+ means the number of buffered frames is larger than $B_T \times N_s$. The reward of a state can be looked up in the table from the top to the bottom, using the reward of the first entry in the table matching the current state. The values of rewards need to be carefully designed, since it is closely related to the control objective. The stored video has a finite length, and when the state reaches $d = N_T$, i.e., all the segments have been downloaded, the streaming task completes, which is called an episodic task. Therefore, we give state $(*, *, *, *, *, N_T)$ a reward of 0. Besides, any action taken in this state will not change the state, i.e., the terminal state will not affect the decision process. By giving the minimum reward when the buffer is empty, we can minimize playback interruption; by giving a negative reward to the state when the number of buffered frames is larger than the maximum value, we can avoid buffer overflow. When both Δq and Δv are 0, the maximum reward (0) is given, since in these states, the playback should be smooth and the selected video version matches the available bandwidth well.

Table 1: Rewards Associated with States

$s_t = s$	$R(s)$
$(*, *, *, *, *, N_T)$	0
$(0, *, *, *, *, *)$	$-F + \Delta q$
$(F^+, *, *, *, *, *)$	$-F - \Delta q$
$(*, \Delta q, *, \Delta v, *, *)$	$\min(-\alpha\lvert\Delta v\rvert, -\lvert\Delta q\rvert)$

In addition, we can associate a weight parameter α with the reward to make a trade-off between average playback quality and playback smoothness. When α is smaller, the video streaming can be more adaptive to the available bandwidth to achieve higher average playback quality; when α is larger, a higher priority is given to the playback smoothness. Note that the reward is independent of the bandwidth, since we are unable to control the varying bandwidth.

Finally, we can formulate the rate adaptation problem as an optimization problem. The objective is to find a strategy $\pi(s)$ for the action taken at a state s to maximize the reward received in the long run. The state-value function given a deterministic strategy is thus

$$V^\pi(s) = \sum_{s'} \mathcal{P}^a_{ss'} \left[R(s) + \gamma V^\pi(s') \right], \quad (5)$$

where γ is the discounting rate $0 \leq \gamma \leq 1$. Note that in our case, we can set γ to 1, since we are dealing with an episodic streaming task. An optimal strategy $\pi^*(s)$ should maximize

the state-value function in the long run, i.e.,

$$\pi^*(s) = \arg\max_{\pi} \sum_{s'} \mathcal{P}^a_{ss'} \left[R(s) + \gamma V^*(s') \right], \qquad (6)$$

where $V^*(s)$ is the optimal value function. Then, we can obtain the optimal streaming strategy using dynamic programming [14]. The solution is a table that maps each state to an optimal action. During the online decision making process, a table look-up can quickly identify the action to take, which is simple and feasible for handheld devices.

Furthermore, to reduce the number of states for MDP and the input size for dynamic programming, we divide the buffer size (in frames) into small bins and index them as q_b starting from 0 to $\lfloor B_T \times N_s / BS \rfloor$, where BS is the number of frames in each bin. Then we use $q_b \times BS$ to represent the number of buffered frames for each bin.

4. PERFORMANCE EVALUATION

In this section, we first define the QoE metrics in terms of playback interruption, average playback quality and playback smoothness. Then we evaluate the proposed solution and compare it with the existing state-of-the-art rate adaptation algorithm [8] by experiments.

4.1 QoE Metrics

Interruption ratio: Every $1/f$ second (f is video frame rate), the video player displays one frame, which is defined as one display event. If there is no decoded frame available to display, playback interruption occurs. Let n_0 be the number of occurrences that a frame to be displayed is not available. Denote by n_t the total number of display events, the interruption ratio (IR) is defined as $\text{IR} = n_0/n_t$.

Average playback quality: We define a continuous playback of Layer i video as one run and its length in terms of the number of display events as n_r for the r-th run. There are totally N runs. The layer index 0 denotes that a playback interruption happens. The weighted sum of the layer index is used to measure the average playback quality (APQ), which is defined as $\text{APQ} = \sum_{r=1}^{N}(n_r \times i) / \sum_{r=1}^{N}(n_r)$.

Playback smoothness [9]: Intuitively, a longer expected run length leads to a smoother watching experience. It also gives fair evaluation when the length of one run is much larger than the others compared with arithmetic average. Thus the expected run length is used to measure the playback smoothness (PS), and we have $\text{PS} = \sqrt{\sum_{r=1}^{N}(n_r)^2/N}$.

4.2 Experimental Settings

We have prototyped a scalable video streaming testbed [16] and used it to evaluate the proposed streaming strategy and compared it with the existing state-of-the-art solution [8]. The testbed used `Lighttpd` as the streaming server and the video player was implemented using an open-source SVC decoder [4]. The web server and the video player communicate through HTTP/TCP protocols, and a channel emulator was used to simulate the varying bandwidth in wireless networks.

We used the open-source SVC codec JSVM [10] to encode the sample video ("Big Buck Bunny" [1]) into three layers, and their configurations are listed in Table 2. Note that the Y-PSNR of Layer 3 is lower than that of Layer 2, but we still prefer Layer 3 video which has a higher resolution, and it leads to a better watching experience when displayed on a larger screen due to a higher dots per inch (DPI). Obviously,

Table 2: Layer Configuration

Resolution	Avg. bit-rate (Kbps)	std bit-rate deviation	Y-PSNR	Layer index
320x180	112.84	39.01	35.47	1
320x180	238.94	88.84	39.44	2
640x360	363.82	140.33	35.90	3

Table 3: State Prob. and Available Bandwidth

State	1	2	3	4
Bandwidth (Kbps)	50.32	180.63	260.38	550.75
Steady state prob (P_1)	0.026	0.102	0.407	0.465
Steady state prob (P_2)	0.103	0.256	0.385	0.256

PSNR is not an appropriate QoE performance index for SVC encoded videos, so it is not considered in this paper. Each layer is chopped into small segments of 17 frames. The total number of segments N_T is 200, and the frame rate is 24 frames per second. From experiments, we found that the segment size of 17 frames is small enough to react to the varying bandwidth. The playback starts when 4 segments are received.

To maximize the spectrum efficiency and limit the packet error rate, broadband wireless systems (such as 3G and WLAN) can adjust the transmission data rate according to the wireless channel quality using adaptive modulation and coding techniques. When the channel quality is good, a higher data rate is used, and vice versa. As the wireless channel condition may change randomly, the finite-state Markov model has been widely used to describe the variation of wireless channel conditions [15, 17], and thus it can also be used to describe the wireless link data rate variation for broadband wireless systems. Since the wireless access link is presumably the bottleneck, we used a discrete-time four-state Markov model to capture the variation of the available bandwidth, and the duration of the time step for the Markov model is constant (700 ms in our experiment and it is close to the segment playback duration). We used two sets of probability transition matrices for two different wireless link profiles. The two matrices, P_1 and P_2 are, respectively,

$$\begin{bmatrix} 0.5 & 0.05 & 0.05 & 0.4 \\ 0.2 & 0.25 & 0.2 & 0.35 \\ 0.2 & 0.1 & 0.2 & 0.5 \\ 0.1 & 0.1 & 0.1 & 0.7 \end{bmatrix}, \begin{bmatrix} 0.25 & 0.75 & 0 & 0 \\ 0.3 & 0.4 & 0.3 & 0 \\ 0 & 0.2 & 0.6 & 0.2 \\ 0 & 0 & 0.375 & 0.625 \end{bmatrix}.$$

The average bandwidth and steady state probabilities of the wireless link under different profiles are listed in Table 3. The first difference is that the average bandwidth of the two profiles are 377.6 Kbps and 292.84 Kbps, respectively. The other difference is that a link with P_1 has a smaller average fading duration than that with P_2, i.e., given the link is in a worse channel condition, the link with P_2 may stay in the worse condition for a longer duration on average.

One challenge for the MDP model used in Section 3 is that the MDP model requires the state transition probability after one segment is downloaded, but the segment download duration may not be a constant. In this paper, the segment download time is approximated by a constant to obtain the state transition matrix for the MDP. According to the experiment results, such approximation is acceptable. One reason is that the proposed rate adaptation strategy prefers to se-

Table 4: Playback Performance

P	B_T	ALG	IR	APQ	PS	Max queue
P_1	20	RA	0	2.03	117.30	23
		OS	0	2.22	189.72	20.5
	30	RA	0	1.91	124.45	33
		OS	0	2.19	237.37	30.2
		FL(3)	0.07	2.78	314.7	13.6
P_2	20	RA	0	1.68	155.27	23
		OS	0	1.88	246.54	20.4
	30	RA	0	1.60	200.5	33
		OS	0	1.87	268.32	30.1
		FL(2)	0.03	1.93	1176.74	24.7

lect the video version to match the available bandwidth, so the time to download a segment does not vary severely. Also, the proposed rate adaptation algorithm can tolerate the inaccuracy in the Markov model used, which will be discussed further near the end of Section 4.3.

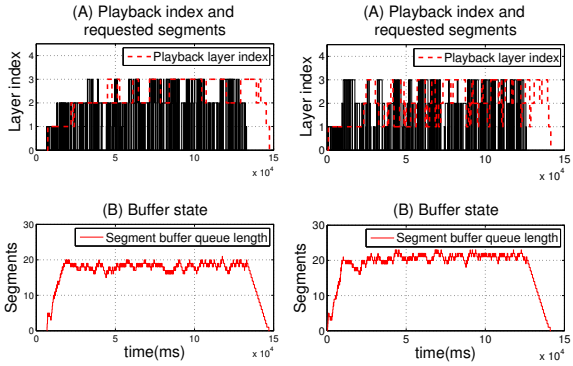
Since the video has three layers (versions), the set of actions of state s is $\mathcal{A}(s) = \{A_{-2}, A_{-1}, A_0, A_1, A_2, A_u, A_w\}$. For A_w, the client will wait for 700 ms (one time step) before sending the HTTP request of the next segment. The bin size is set to 17 to reduce the number of states. By dynamic programming, the action for every possible state can be obtained offline. During the video download process, the client only needs to look up the table to make the decision. Although the number of states is not small, since each state is unique, we can index and store each state and action at a unique location. Then looking for the action for a particular state only takes $O(1)$ time.

4.3 Performance Comparison

We compare our proposed optimal streaming (OS) strategy with the rate adaptation (RA) algorithm in [8]. With RA, a client may be idle for some time to avoid buffer overflow, but it does not explicitly set the target buffer size. To make the comparison fair, we set the minimum buffered media time in RA the same as the target buffer size (B_T) in our solution. We repeated the experiments for each algorithm 10 times under each configuration, and the results presented here are the average over 10 runs.

Table 4 compares the two algorithms with different target buffer sizes and wireless conditions. We also include another algorithm FL(3) for P_1, which fixes the layer index to 3, as the average network bandwidth is larger than the average bit-rate of Layer 3. Due to the variation of segment size and network bandwidth, FL(3) suffers from "stuttering", which is very unpleasant for the watching experience. FL(2) of P_2 shows similar results. From the table, both the proposed OS and the existing RA algorithms can make the playback free of interruption. OS has the advantage over RA in terms of both APQ and PS in all cases (for P_1, OS uses $\alpha = 1$; for P_2, OS uses $\alpha = 2$). RA assumes that the size of a segment is large enough such that the average download throughput of a segment can be used to represent the average available bandwidth for the following segment. For a highly varying wireless link, even with a very large segment, the throughput of the next segment can be quite different. In addition, OS controls the queue length better than RA, because RA conservatively estimates the throughput of the next segment as the bit-rate of the lowest version of the video.

Figs. 2(a) and (b) show the playback traces during the



(a) OS ($\alpha = 1$) (b) RA

Figure 2: Performance Comparison

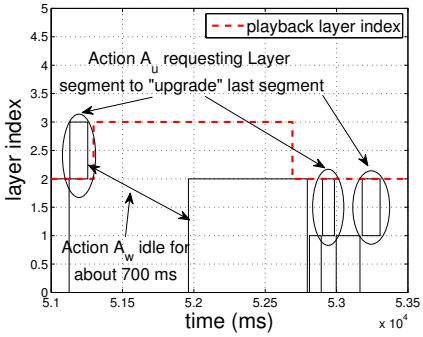


Figure 3: A Zoom-in of Playback Trace

experiments and the corresponding buffer occupancy states with OS and RA, respectively, where the transition matrix is P_1 and $B_T = 20$. From these figures, RA encounters more frequent layer switching with a lower smoothness than OS. The maximum queue length of RA is also larger.

We further zoom in the playback trace for OS. In Figure 3, the black rectangle represents a segment and the width of it denotes the download time duration (from the time instant of sending the HTTP request to that the segment is completed received). The horizontal gap between edges of rectangles is due to the waiting action to avoid buffer overflow. Figure 3 shows the advantage of the proposed video streaming framework using SVC: the rectangles rise from a non-zero layer index (circled and annotated by arrows) are the layer segments to "upgrade" the already buffered segments to improve both APQ and PS, which is not possible when using the traditional AVC streaming techniques.

With OS, another flexibility is that we can make a trade-off between APQ and PS by adjusting the reward parameter α. When we increase the value of α to 2, as shown in Table 5, the APQ is reduced slightly from 2.22 to 2.14 and the expected run length is increased from 189.72 to 333.56. When $\alpha = 3$, similar trend happens. When we set α to 10, the APQ is 1 and the run length is $3,400$, which is the extreme case that any action involving layer switching is avoided.

Last but not least, one practical issue is that the Markov model used for the varying bandwidth may not be accurate. It is important to test how sensitive the performance of OS is to the model accuracy. In the test, we used P_2 as the

Table 5: Trade-off between APQ and PS ($P = P_1$)

α	IR	APQ	PS	Max queue
1	0	2.22	189.72	20.5
2	0	2.14	333.56	20.5
3	0	2.11	410.74	20.1
10	0	1	3400	20

Table 6: Model Sensitivity Test

Env	B_T	ALG	IR	APQ	PS	Max queue
P_2	20	OS(P_1)	0	1.92	183.79	20
		OS(P_2)	0	1.88	246.54	20.4
	30	OS(P_1)	0	1.88	229.58	29.9
		OS(P_2)	0	1.87	268.32	30.1

Markov model to drive the channel emulator in the experiment, and used both P_1 and P_2 in the dynamic programming to obtain the action decisions. The results are shown in Table 6. OS(P_i) means that the streaming strategy is obtained using transition matrix P_i for $i = 1, 2$. From the table, when the matrix in the decision process does not match the real situation, the performance degrades slightly, but still in a tolerable range. Also, comparing the results in Tables 4 and 6, even with a mismatched model for the available bandwidth, the proposed OS still substantially outperforms RA in terms of both APQ and PS.

5. CONCLUSIONS

In this paper, for DASH-based adaptive video streaming in wireless networks, we have formulated the rate adaptation problem as an MDP and used dynamic programming to solve the problem. The trade-off between the average video quality and playback smoothness can be made by adjusting the parameter in the reward function. Experiment results have shown that the proposed solution is feasible and substantially outperforms the existing one [8]. There are several issues worth further investigation. To fully utilize the layered feature of SVC, we may consider other possible actions, such as to "upgrade" multiple previously received segments when possible. However, more actions may increase the size of the action set and require more information to describe the system state, which increases the state number and system complexity. The proposed streaming policy is an off-line solution requiring the bandwidth transition probability matrix. How to design an on-line algorithm to estimate the bandwidth transition matrix is left for future investigation. Nevertheless, the proposed solution is robust against bandwidth estimation errors, so it is promising to be used even without accurate knowledge of the channel profile. Also, the number of the states for the MDP in this paper is considerably large, which requires a large memory space. It is desirable to reduce the number of states without substantially sacrificing the performance. Besides rate adaptation, other issues such as how to organize the layer segments efficiently and optimize the segment size require further research.

6. REFERENCES

[1] Big Buck Bunny. `http://www.bigbuckbunny.org`.

[2] S. Akhshabi, A. C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *ACM MMSys'11*, pages 157–168, New York, NY, USA, 2011.

[3] A. Begen, T. Akgul, and M. Baugher. Watching video over the web: Part 1: Streaming protocols. *IEEE Internet Computing*, 15(2):54 –63, March-April 2011.

[4] M. Blestel and M. Raulet. Open SVC decoder: a flexible SVC library. ACM MM '10, pages 1463–1466, New York, NY, USA, 2010.

[5] L. Cai, S. Xiang, Y. Luo, and J. Pan. Scalable modulation for video transmission in wireless networks. *IEEE Trans. on Veh. Tech.*, 2011.

[6] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *ACM MMSys'11*, pages 145–156, New York, NY, USA, 2011.

[7] M. Kobayashi, H. Nakayama, N. Ansari, and N. Kato. Robust and efficient stream delivery for application layer multicasting in heterogeneous networks. *IEEE Transactions on Multimedia*, 11(1):166–176, Jan. 2009.

[8] C. Liu, I. Bouazizi, and M. Gabbouj. Rate adaptation for adaptive HTTP streaming. In *ACM MMSys'11*, pages 169–174, New York, NY, USA, 2011.

[9] S. Nelakuditi, R. Harinath, E. Kusmierek, and Z. Zhang. Providing smoother quality layered video stream. In *ACM NOSSDAV'00*, June 2000.

[10] J. Reichel, H. Schwarz, and M. Wien. Joint scalable video model 11 (JSVM 11). *Joint Video Team, Doc. JVT- X*, 2007.

[11] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Lelouedec. iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding. In *ACM MMSys'11*, pages 257–264, New York, NY, USA, 2011.

[12] T. Schierl, Y. Sanchez de la Fuente, R. Globisch, C. Hellge, and T. Wiegand. Priority-based media delivery using SVC with RTP and HTTP streaming. *Multimedia Tools and Applications*, 55:227–246, 2011.

[13] T. Stockhammer. Dynamic adaptive streaming over HTTP: standards and design principles. In *ACM MMSys'11*, pages 133–144, New York, NY, USA, 2011.

[14] R. Sutton and A. Barto. *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998.

[15] H. S. Wang and N. Moayeri. Finite-state Markov channel – a useful model for radio communication channels. *IEEE Trans on Veh. Tech.*, 44(1):163–171, 1995.

[16] S. Xiang. Scalable streaming. `https://sites.google.com/site/svchttpstreaming/`.

[17] R. Zhang and L. Cai. A packet-level model for uwb channel with people shadowing process based on angular spectrum analysis. *IEEE Trans. on Wireless Comm.*, 8(8):4048–55, Aug. 2009.