A Control-Theoretic Approach to Rate Adaption for DASH Over Multiple Content Distribution Servers

Chao Zhou, Chia-Wen Lin, *Senior Member, IEEE,* Xinggong Zhang, *Member, IEEE,* and Zongming Guo

***Abstract*—Recently, dynamic adaptive streaming over HTTP (DASH) has been widely deployed on the Internet. However, the research about DASH over multiple content distribution servers (MCDS-DASH) is limited. Compared with traditional single-server DASH, MCDS-DASH is able to offer expanded bandwidth, link diversity, and reliability. It is, however, a challenging problem to smooth video bitrate switching over multiple servers due to their diverse bandwidths. In this paper, we propose a block-based rate adaptation method considering both the diverse bandwidths and feedback buffered video time. In our method, multiple fragments are grouped into a block and the fragments are downloaded in parallel from multiple servers. We propose to adapt video bitrate at the block level rather than at the fragment level. By dynamically adjusting the block length and scheduling fragment requests to multiple servers, the requested video bitrates from the multiple servers are synchronized, making the fragments download in an orderly way. Then, we propose a control-theoretic approach to select an appropriate bitrate for each block. By modeling and linearizing the rate adaption system, we propose a novel proportional-derivative controller to adapt video bitrate with high responsiveness and stability. Theoretical analysis and extensive experiments on our network testbed and the Internet demonstrate the good efficiency of the proposed method.**

***Index Terms*—Control-theoretic approach, dynamic adaptive streaming over HTTP (DASH), multiple servers, rate adaption.**

I. Introduction

IN RECENT years, dynamic adaptive streaming over HTTP (DASH) has been widely adopted for providing uninterrupted video streaming service to users with dynamic network conditions and heterogeneous devices [1]–[3]. Contrary to the past RTP/UDP, the use of HTTP over TCP is easy to configure and, in particular, it greatly simplifies the traversal of firewalls and network address translators. Besides, the deployment cost of DASH is relatively low since it employs standard HTTP servers and, therefore, can easily be deployed within content delivery networks. In DASH, a video content is encoded into multiple versions at different bitrates. Each encoded video version is further divided into small video fragments, each of which normally contains seconds or tens of seconds worth of video. Using the HTTP protocol, by employing multiple content distribution servers (MCDS), a DASH client can send requests to download video fragments from multiple servers concurrently. Compared with a single server, MCDS can provide higher bandwidth, reliability, and scalability. Thus, when deploying DASH over multiple servers, the quality of services can be greatly improved. However, to the best of our knowledge, the research about DASH over MCDS (MCDS-DASH) is still very limited.

In DASH, a client continuously requests and receives fragments of video content. To adapt the video bitrate to a varying network bandwidth, DASH allows clients to request video fragments from different versions of a video, each of which being coded with a specific bitrate. This is known as dynamic rate adaption, which is one of the most important features since it is able to automatically throttle the video quality to match the available bandwidth so that the user receives the video at the maximum quality possible. However, a DASH client might suffer from frequent interruptions and nonoptimal visual quality without an efficient rate adaptation algorithm. A video bitrate higher than the bandwidth would cause network congestion and, on the other hand, when the video bitrate is lower than the available bandwidth, the video quality does not reach the optimum allowed by the available bandwidth. Moreover, an efficient rate adaptation algorithm should prevent frequent hopping between video bitrates, because frequent changes in the perceived video quality are likely to be annoying [4]. In MCDS-DASH, rate adaption becomes more challenging as fragments can be transmitted from different servers which have different bandwidths. In general, designing an efficient rate adaptation scheme for MCDS-DASH faces three main challenges.

1) Frequent bitrate fluctuations due to heterogeneous bandwidths of MCDS-DASH. In MCDS-DASH, multiple fragments are requested in parallel from MCDS to obtain a larger bandwidth. However, it takes different time durations to complete the downloads of fragments from different servers due to their heterogeneous bandwidths. Therefore, if the video bitrate is adjusted when a fragment is downloaded completely as traditional methods do, it is likely that the requested video bitrates are

Manuscript received May 17, 2013; revised September 10, 2013; accepted October 7, 2013. Date of publication November 12, 2013; date of current version April 2, 2014. This work was supported in part by the National Science Council, Taiwan, under Grant NSC101-2221-E-007-121-MY3, in part by the National High_Tech Technology Research and Development Program (863 Program) of China under Grant 2013AA013504, in part by the National Natural Science Foundation of China under Contract 61071082, and in part by the National Key Technology Research and Development Program of China under Grant 2012BAH18B03. This paper was recommended by Associate Editor D. O. Wu.

C. Zhou, X. Zhang, and Z. Guo are with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China.

C.-W. Lin is with the Department of Electrical Engineering and the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: cwlin@ee.nthu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2013.2290580

updated at different time instants for different servers. The asynchronous bitrate updating leads to frequent video bitrate fluctuations that would degrade user viewing experience [5].

2) Additional delay of video playback due to out-of-order video fragment downloads. Traditionally, fragments are requested in an orderly way from MCDS (i.e., one after another). But, their completion time may be out of order due to the heterogeneous bandwidths of servers as mentioned above. This would cause additional delay of video playback since video fragments must be played in the correct order, because a fragment can be played only if all its previous fragments have been played.
3) Tradeoff between bitrate smoothness and bandwidth utilization. Generally, there is a fundamental conflict between video bitrate smoothness and bandwidth utilization due to the inherent bandwidth variations. For example, exactly matching video bitrate with bandwidth achieves the highest bandwidth utilization, while leading to severe rate fluctuations. Therefore, how to balance the needs for video bitrate smoothness and bandwidth utilization is challenging for rate adaption schemes. This becomes even more challenging for MCDS-DASH since fragments can be transmitted through different links with different bandwidths.

In this paper, we address the rate adaption problem for MCDS-DASH. We take the advantage of multiple servers to offer even better viewing experience. To overcome the above challenges, we propose a block-based rate adaption scheme considering both the heterogeneous bandwidth and feedback buffered video time. Multiple fragments are grouped into a block, as shown in Fig. 1. Whenever to download a new block, a different number of fragments are requested from MCDS and the block length is the total size of the requested fragments in this round; thus, the block length is also dynamically adapted. The servers having completed their downloads keep idle until the entire block has been downloaded completely. Thus, all servers begin to download each block at the same time. The requested bitrates are updated synchronously for all servers since the video bitrate is adapted at the granularity of block. At last, combining with the predicted bandwidth and feedback buffered video time, a control-theoretic approach is designed to select the best bitrate for each block. In summary, our contribution is three-fold.

1) We show that adapting a video bitrate at the granularity of fragment would cause frequent playback video bitrate fluctuations. Thus, we propose a block-based rate adaption scheme for MCDS-DASH. By dynamically determining the block size and the number of fragments assigned to multiple servers, the requested video bitrates for all servers are updated synchronously.
2) We find that requesting fragments in order delays video playback. Instead, we propose to schedule fragment requests according to their playback deadlines. Therefore, the fragments are completely downloaded in order and continuous video playback is ensured.

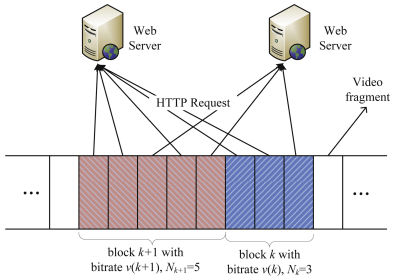


Fig. 1. Request fragments from MCDS-DASH.

3) We propose a control-theoretic approach to select the best bitrate for each block considering both the heterogeneous bandwidth and feedback buffered video time. We use two thresholds as the operating points to filter the effect of bandwidth variations on rate adaption. By theoretical analysis, we model and linearize the rate control logic. A proportional-derivative (PD) controller is further involved to improve rate adaption performance. By carefully designing the parameters for the PD controller, we balance the needs for video bitrate smoothness and bandwidth utilization.

The rest of this paper is organized as follows. Section II surveys the related works. In Section III, we briefly describe our proposed rate adaption approach. We dynamically determine the block length and schedule the fragment requests to multiple servers in Section IV. A novel control-theoretic approach for rate adaption is proposed in Section V. Some implementation details are discussed in Section VI. We show the experimental results in Section VII and conclude this paper in Section VIII.

II. Related Works

Although DASH is a relatively new application, due to its popularity, it has attracted much research effort recently. Watson [6] systematically introduced the DASH framework of Netflix, which has been the largest DASH stream provider in the world. The study in [7] further showed that Netflix always bound a user to a certain server, regardless of the available TCP throughput between the user and the server. It also suggested that the user experience can greatly be improved if a user is allowed to download a video concurrently from multiple servers.

Rate adaption is one of the most important research issues for DASH. Akhshabi *et al.* [8] compared rate adaption for three popular DASH clients: Netflix client [6], Microsoft smooth streaming [9], and Adobe OSMF [10]. The conclusion in [8] indicates that none of the DASH client-based rate adaptation is good enough, as they are either too aggressive or too conservative. Some clients even just jump between the highest bitrate and the lowest bitrate. Also, all of them have relatively long response time under network congestion level shift. There exist several research works about rate adaption for DASH, such as bandwidth-based rate adaption schemes [10]–[12] and buffer-based rate adaption schemes [13]. All these existing rate adaption algorithms aim to either adapt a video bitrate to an available bandwidth so as to achieve a high

bandwidth utilization efficiency, or ensure a buffer in the client to provide a continuous video playback. However, due to the inherent bandwidth variations, there is a fundamental conflict between video bitrate smoothness and bandwidth utilization, and existing algorithms do not balance the needs for these two aspects well. Besides, they were all designed for single-server-based DASH and cannot be directly applied to MCDS-DASH. Since our proposed rata adaption approach for MCDS-DASH jointly considers predicted bandwidth and feedback buffer occupancy, it can well balance the needs for video bitrate smoothness and bandwidth utilization.

The work in [14] is among the first to study the rate adaption problem for MCDS-DASH. However, the method is originally designed for single-server DASH, and it adapts video bitrate at the fragment granularity. As a result, the requested video bitrates are updated asynchronously for multiple servers and it causes frequent fluctuations of playback video bitrate. Moreover, fragments are requested from multiple servers one after another without considering their completion time. Thus, the completion time of the fragments is generally out of order due to the channel heterogeneity of multiple servers. Instead, our proposed rate adaption approach is block based and the video bitrates are adapted synchronously. Besides, fragment requests are scheduled to multiple servers according to their playback deadline priorities so as to guarantee their completion time in order.

Control theoretical approaches have proven to be effective for rate adaption [13], [15]–[18]. Huang *et al.* [15] proposed a proportional controller to stabilize the received visual quality, as well as the bottleneck link queue. The work in [16] analyzed the TCP congestion control algorithm based on a control-theoretic approach. It is worth noticing that there is inherent feedback control logic in these systems. For example, the method proposed in [15] employed the random early detection model for bottleneck routing; therefore, the packet loss probability and queue length forms a feedback control system automatically [17]. It is shown in [16] that TCP itself is an unstable integral control system, whereas in DASH, there is no such inherent or explicit feedback control logic for rate adaption. In [13], a feedback control mechanism is designed to control the sending buffer size at the server side so that the TCP sending buffer is always full and DASH video bitrate matches the available bandwidth. These successful stories clearly demonstrate the strength of the control-theoretic approaches. Motivated by this, we propose a control-theoretic rate adaption approach for MCDS-DASH, which differs from [13] in several major ways. First, our approach is solely implemented at the client side, whereas the method in [13] is implemented at the server side which has limitation in supporting the large-scale multimedia delivery since it will dramatically increase the burden on the web server or cache. Second, the method in [13] adapts video bitrate to match bandwidth by controlling the sending buffer. It aims to achieve high bandwidth utilization while ignoring the smoothness of video bitrate. While we directly control video bitrate by the predicted bandwidth and feedback buffered video time, and balance the needs for video bitrate smoothness and bandwidth utilization. Last, the mechanism in [13] is designed for a single server and cannot be directly applied for multiple servers, whereas our proposed approach is suitable for both single and multiple servers.

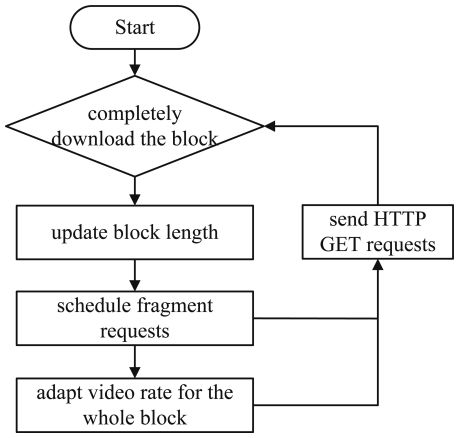


Fig. 2. Flowchart of the proposed rate adaption scheme for MCDS-DASH.

III. Proposed Rate Adaption Scheme for MCDS-DASH

In this section, we briefly describe the proposed rate adaption approach for MCDS-DASH. Instead of adapting video bitrates for individual fragments, we propose to adapt video bitrate for a block of fragments simultaneously, i.e., block based rate adaption. Fig. 2 shows the flowchart of the proposed rate adaption approach for MCDS-DASH. To overcome the challenges mentioned in Section I, the proposed rate adaption approach mainly includes the following three components.

A. Block Length Adjustment

Since fragment-based rate adaption usually causes frequent fluctuations of playback video bitrate in MCDS-DASH, we propose a block-based rate adaption approach that adapts video bitrate for a block of multiple fragments simultaneously. To determine an appropriate block size, our method dynamically determines the block sizes requested from multiple servers according to the proportions of their bandwidth estimates so that all servers complete their downloads nearly at the same time. We also prove that the bandwidth waste can be minimized by the method in Section IV-A.

B. Fragment Requests Scheduling

Different video fragments have different playback deadlines. Requesting fragments in order would cause their completion time out of order due to heterogeneous bandwidth in MCDS, thereby leading to additional delay of video playback. To address the problem, our method schedules fragment requests to multiple servers according to their playback deadlines. Thus, the fragments with earlier playback deadlines are downloaded completely no later than the ones with later playback deadlines, thereby ensuring continuous video playback.

C. Rate Adaptation

An appropriate video bitrate needs to be determined for each block. On one hand, playback interruptions happen when a too

TABLE I
MAJOR SYMBOLS USED IN THIS PAPER

S	number of servers
c_i	bandwidth of server i with $1 \leq i \leq S$
$q_{\min}$	threshold of buffer occupancy to prevent buffer under-flowing
$q_{\max}$	threshold of buffer occupancy to prevent buffer over-flowing
$q(t)$	buffer occupancy at time t
T	playback duration of each fragment (in seconds)
$block\_t_k^s$	time instant when start to download block k
$block\_t_k^e$	time instant when block k is downloaded completely
$frag\_t_{n,k}^s$	time instant when start to download the n-th fragment of block k
$frag\_t_{n,k}^e$	time instant when the n-th fragment of block k is downloaded completely
$v(k)$	video bitrate for block k
N_k	block length of block k

high video bitrate is selected; on the other hand, bandwidth cannot be effectively utilized when we select a too low video bitrate. Besides, frequent rate switchings should be avoided since it greatly deteriorates user experience of streaming services. In our rate adaption approach, two thresholds are used to mitigate the effect of short-term network bandwidth variations. In addition, they are also used as two operating points to prevent buffer from under-flowing and over-flowing, respectively. We model and linearize the rate control logic by a theoretical analysis and propose a PD controller accordingly to improve rate adaption performance and balance the needs for video bitrate smoothness and bandwidth utilization.

In the following sections, we give the details of these three components as well as some implementation details of the rate adaption approach. We list some of the important symbols used in this paper and their explanations in Table I.

IV. BLOCK LENGTH AND REQUESTS SCHEDULING

In this section, we present a method that dynamically determines the number of fragments requested from multiple servers, so as the block length. Then, a playback deadline based scheduling algorithm is presented to schedule fragment requests to multiple servers.

A. Dynamic Block Length

In MCDS-DASH, a heuristic client-based video fragment download strategy is to directly assign different fragment download requests to different servers, with the number of fragments assigned to a server to be proportional to its bandwidth. We denote all these requested fragments as a block. By this method, all servers completely download the block at about the same time. Since usually a too large block size is not robust against bandwidth variations, we always assign only one fragment to the server with the lowest bandwidth. Then, the number of fragments assigned to other servers is equal to the ratio of their bandwidth to the lowest bandwidth. However, usually, the ratios are not integer and thus need to be quantized into integer values. As a result, bandwidth waste is unavoidable. In the following, we present a dynamic-threshold-based quantification method to determine the number of fragments for each server.

Suppose there are S servers with bandwidth c_i for the ith server, and the bandwidth is sorted in the descending order so that $c_1 \geq c_2 \geq \cdots \geq c_s$. We further suppose that the fragments in block k are requested in parallel from the first $S_k (S_k \leq S)$ servers chosen by the procedure described below. The length of the kth block is denoted as N_k with n_i fragments requested from the ith server; thus, the block length $N_k = \sum_{i=1}^{S_k} n_i$. Then, we define that

$$\frac{c_i}{c_{S_k}} = \gamma_i + \eta_i \tag{1}$$

where $\gamma_i = \left\lfloor \frac{c_i}{c_{S_k}} \right\rfloor$ is the largest integer smaller than or equal to $\frac{c_i}{c_{S_k}}$. Now, the number of fragments assigned to server i is

$$n_i = \begin{cases} \gamma_i & \text{if } \eta_i < \mu_i \\ \gamma_i + 1 & \text{if } \eta_i \geq \mu_i \end{cases} \tag{2}$$

where μ_i is a dynamic quantification threshold for server i given as

$$\mu_i = \frac{-\gamma_i - 1 + \sqrt{\gamma_i^2 + 2\gamma_i + 5}}{2}. \tag{3}$$

Proposition 1: When the number of fragments assigned to server i is determined by (2) with dynamic quantification threshold in (3), the wasted bandwidth for server i or server S_k is minimized.

Proof: The proof is given in Appendix A. ■

Although the number of fragments assigned to each server is greedily determined without considering the other servers, our experiments demonstrate that our approach achieves higher bandwidth utilization than other methods. On the other hand, since a too large block size may cause rate adaption too sluggish to bandwidth variations, we set the maximum length of the block as N. When the block length is so large that $N_k > N$, no fragment will be requested from the server with the lowest bandwidth. The number of fragments assigned to the remaining servers is updated using the same method described in Proposition 1. This process is iterated until the block length does not exceed the predefined threshold. Table II summarizes to determine the length of block k and which servers will be used for downloading the fragments in block k.

B. Playback Deadline Based Requests Scheduling

Requesting fragment in order may delay video playback. For example, suppose there are two servers with bandwidth 4 Mb/s and 1 Mb/s, respectively, and the requested video bitrate is also 1 Mb/s. If we request fragments from these two servers in order, then the first, third, and fourth fragments will be requested from the first server, and the second fragment will be requested from the second server. But, the third fragment is downloaded completely earlier than the second one. This

TABLE II
BLOCK LENGTH UPDATING ALGORITHM

initialize $c_{\min} \leftarrow c_s, S_k \leftarrow S$;
while 1 **do**
  $n_{S_k} \leftarrow 1$;
  for $i \leftarrow 1$ to $S_k - 1$ **do**
    compute n_i according to (1-3);
  end for
  $N_k \leftarrow \sum_{i \leftarrow 1}^{S_k} n_i$
  if $N_k > N$ **then**
    $S_k \leftarrow S_k - 1, c_{\min} \leftarrow c_{S_k}$;
  else
    return N_k, S_k;
  end if
end while

TABLE III
FRAGMENT REQUEST SCHEDULING ALGORITHM

initialize $\mathbf{X} \leftarrow \mathbf{0}$;
for $i \leftarrow 1$ to N_k **do**
  $j^* \leftarrow \underset{1 \le j \le S_k}{\operatorname{argmin}} \left(\frac{1}{c_j} + \sum_{t=1}^{i-1} x_{tj} \frac{1}{c_j} \right)$;
  $x_{ij^*} \leftarrow 1$;
end for
return $\mathbf{X}$;

delays video playback since the third fragment cannot be played until the second one has been downloaded completely and played.

To address the problem, we propose to schedule the fragment requests according to their playback deadlines. Thus, the fragments are completely downloaded in order. Let $\mathbf{X}$ be the binary indicator matrix with size $N_k \times S_k$, its entries $x_{ij} = 1$ if the ith fragment in block k is requested from server j; otherwise, $x_{ij} = 0$. Since each fragment has the same size in a certain block (i.e., the same bitrate and playback duration), its download time is inversely proportional to a server's bandwidth. In order to provide continuous video playback, we schedule the fragment requests to multiple servers with the objective that fragment i needs to be completely downloaded no later than fragment j for any $i < j$. Then, the fragments are requested from multiple servers as summarized in Table III.

V. PROPOSED CONTROL-THEORETIC RATE ADAPTATION FOR MCDS-DASH

Having synchronized the download time for all servers by dynamically updating the block length and scheduling fragment requests to MCDS, we now move to rate adaptation for each block.

A. Buffered Video Time Model

In DASH, different video versions have different video playback bitrates. Since a video buffer contains fragments from different versions, there is no longer a direct mapping between the buffered video size (i.e., buffer occupancy) and the buffered video time. To deal with multiple video versions, we use the buffered video time to measure the length of video playback buffer.

The buffered video time process, denoted as $q(t)$, can be modeled as a queue with a constant service rate of unity, i.e., in each second, a piece of video with length of one second of playback time is played and dequeued from the buffer. The enqueue process is driven by the video download rate and the downloaded video version. All versions of the video are partitioned into equal-length fragments, each of which consumes the same playback time of T. We adapt the video bitrate when a block is downloaded completely. Without loss of generality, suppose a client starts to download block k at time instant $block\_t_k^s$ and the block is downloaded completely at $block\_t_k^e$. The video bitrate for block k is denoted as $v(k)$. Moreover, the starting and ending download time instants for the nth fragment of block k are denoted as $frag\_t_{n,k}^s$ and $frag\_t_{n,k}^e$. Then, according to Table III, we have

$$frag\_t_{n,k}^e - frag\_t_{n,k}^s = \frac{Tv(k)}{\sum_{j=1}^{S_{\max}} x_{nj} c_j} \tag{4}$$

$$frag\_t_{n,k}^e - block\_t_k^s = Tv(k)\,\alpha(n) \tag{5}$$

where $\alpha(n) = \frac{\sum_{j=1}^{S_k} \left(x_{nj} \sum_{i=1}^{n} x_{ij} \right)}{\sum_{j=1}^{S_k} x_{nj} c_j}$. Equation (4) denotes the time consumed to download fragment n, and (5) denotes the time consumed to download all previous n (from the first to the nth) fragments of block k. When $n = N_k$, (5) also denotes the time consumed to download block k. Then the buffer occupancy evolution becomes

$$q\left(frag\_t_{n,k}^e\right) = q\left(block\_t_k^s\right) + Tn - Tv(k)\,\alpha(n) \tag{6}$$

where the second term of (6) is the added video time upon the completion of the downloading of fragments 1 to n, and the third term reflects the fact that the buffer occupancy is consumed linearly at a rate of 1 during the downloading process.

Because it is difficult to obtain the accurate changing rate of video buffered time at any time instants, we propose to use the fluid approximation [14], which evenly distributes the added video time of TN_k over the download interval for the whole block $(block\_t_k^s, block\_t_k^e]$; then, we have

$$\begin{aligned} q'(t) &= \frac{dq(t)}{dt} \\ &\approx \frac{q\left(block\_t_k^e\right) - q\left(block\_t_k^s\right)}{block\_t_k^e - block\_t_k^s} \\ &= \frac{n}{v(k)\,\alpha(n)} - 1 \Big|_{n=N_k}, \quad t \in (block\_t_k^s, block\_t_k^e]. \end{aligned} \tag{7}$$

B. Rate Control Model and Linearization

From the control system point of view, there is a fundamental conflict between maintaining stable video bitrate and maintaining stable buffer size, due to the unavoidable network bandwidth variations. Nevertheless, from the end user point of view, video bitrate fluctuations are much more perceivable than buffer size oscillations. The recent work in [5] has shown that switching back-and-forth between different

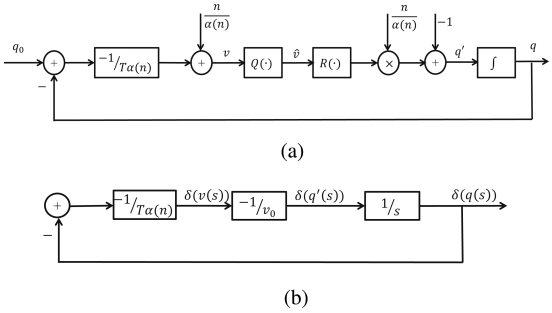


Fig. 3. Block diagram of the buffer based rate adaptation model. (a) Original model. (b) Linearized model.

bitrates will significantly degrade users' viewing experience, whereas buffer size variations do not have direct impact on video streaming quality as long as the video buffer does not deplete. Thus, we propose to use two thresholds, $q_{\min}$ and $q_{\max}$, in the buffered video time model to mitigate the effect of buffer size oscillations on video bitrate adaption. When the buffer occupancy is higher than $q_{\max}$ or lower than $q_{\min}$, $q_{\max}$ and $q_{\min}$ are used as the operating points to select appropriate video bitrates to avoid buffer overflow/underflow. Otherwise, the bitrate remains unchanged and a smooth video bitrate is provided.

When $q\left(block\_t_k^s\right) < q_{\min}$, $q_{\min}$ is used as the operating point to select an appropriate video bitrate to drag current buffer occupancy to be no smaller than $q_{\min}$. To ensure no buffer underflow happens when downloading any fragments in block k, we must have $q\left(frag\_t_{n,k}^e\right) \geq q_{\min}$ for all $n = 1, 2, ..., N_k$; therefore

$$v(k) \leq \frac{n}{\alpha(n)} + \frac{1}{T\alpha(n)}\left(q\left(block\_t_k^s\right) - q_{\min}\right). \tag{8}$$

Similarly, when $q\left(block\_t_k^s\right) > q_{\max}$, buffer overflow needs to be avoided. Therefore, we must have $q\left(frag\_t_{n,k}^e\right) \leq q_{\max}$ for all $n = 1, 2, ..., N_k$ so that

$$v(k) \geq \frac{n}{\alpha(n)} + \frac{1}{T\alpha(n)}\left(q\left(block\_t_k^s\right) - q_{\max}\right). \tag{9}$$

Equations (8) and (9) give the bounds of video bitrate to prevent buffer from either underflow or overflow. In order to avoid selecting a too low or too high bitrate, which may cause frequent video bitrate changes, we use the boundary values in (8) and (9) to control the video bitrate

$$\begin{cases} v(k) = \frac{n}{\alpha(n)} + \frac{1}{T\alpha(n)}\left(q\left(block\_t_k^s\right) - q_0\right) \\ q'(t) = \frac{N_k}{v(k)\alpha(N_k)} - 1 \\ q_0 = \begin{cases} q_{\min}, \text{ if } q(t) < q_{\min} \\ q_{\max}, \text{ if } q(t) > q_{\max}. \end{cases} \end{cases} \tag{10}$$

The block diagram of the rate control model described in (10) is shown in Fig. 3(a), where $Q(\cdot)$ denotes the quantization operation considering that the available video bitrates are finite and discrete, and function $R(\cdot)$ is defined as $R(x) = \frac{1}{x}$. Since this is a nonlinear feedback system, to facilitate the system analysis and design, we propose to linearize the model described in (10) around the operating point q_0 satisfying $q'(t) = 0$. Let the corresponding video bitrate be v_0, we have

$$q'(t) = 0 \Rightarrow v_0 = \frac{N_k}{\alpha(N_k)}. \tag{11}$$

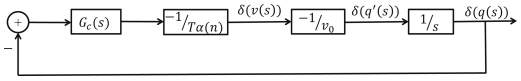


Fig. 4. Block diagram of the feedback control system for rate adjustment.

The following linearized state and output equations are obtained for the feedback system:

$$\begin{cases} \delta(v(k)) = \frac{1}{T\alpha(n)}\delta(q(t)) \\ \delta\left(q'(t)\right) = -\frac{1}{v_0}\delta(v(k)) \end{cases} \tag{12}$$

where $\delta(v(k)) = v(k) - v_0$ and $\delta\left(q'(t)\right) = q'(t) - q'_0$. Applying the Laplace transform to the differential equations, we derive the linearized system block as shown in Fig. 3(b). With some manipulations of the block diagram, we obtain the transfer function of the plant as

$$H(s) = \frac{\alpha(N_k)}{TN_k\alpha(n)s + \alpha(N_k)}. \tag{13}$$

Remark 1 (Stability): The pole of the linearized system is $s_p = -(\alpha(N_k))\backslash(TN_k\alpha(n))$ and it is located in the left phase plane. This indicates that the equilibrium state of the dynamics of the system is stable.

C. PD Controller for Rate Adaptation

Generally, a pure proportional controller would cause system oscillations and may not converge [19]. In this subsection, we design an effective controller for the linearized feedback system in (12). We redraw the block diagram in Fig. 4, where, from the control system prospective, the rate control module $G_c(s)$ is the controller to be designed, while the combination of the other blocks is the plant to be controlled.

We choose a proportional derivative (*PD*) controller since the proportional component can accelerate the response time, and the derivative component can improve system dynamics performance. It is also able to reject step-like disturbances and is very simple for software implementation. The classic *PD* controller can be expressed by the following transfer function:

$$G_c(s) = K_p + K_d s. \tag{14}$$

This controller is implemented at the client side, so it is a distributed controller. It uses the difference between the current buffer occupancy and operating point q_0, and the changing rate of the buffer occupancy to compute the adjustment of video bitrate. In the time domain, the bitrate adjustment for block k can be obtained by

$$\delta(v(k)) = \frac{1}{T\alpha(n)}K_p\left(q\left(block\_t_k^s\right) - q_0\right) + \frac{1}{T\alpha(n)}K_d\frac{q\left(frag\_t_{n,k-1}^e\right) - q\left(block\_t_{k-1}^s\right)}{frag\_t_{n,k-1}^e - block\_t_{k-1}^s}. \tag{15}$$

Generally, $\delta(v(k))$ in (15) has different values for $n = 1, 2, ..., N_k$. Since the video bitrate is adapted in block granularity, when the buffer occupancy is longer than the overflow threshold, the largest adjustment (generally with positive value) should be added to increase the video bitrate. On the other hand, when the buffer occupancy is shorter than the underflow threshold, the smallest adjustment (generally with

negative value) should be used to decrease the video bitrate. Then the video bitrate for block k is obtained as

$$\tilde{v}(k)=\begin{cases} v_0+\min(\delta(v(k))) & \text{if } q\left(block\_t_k^s\right)<q_{\min} \\ v_0+\max(\delta(v(k))) & \text{if } q\left(block\_t_k^s\right)>q_{\max} \\ v(k-1) & \text{else.} \end{cases} \quad (16)$$

Taking into account the finite discrete video bitrates, the actual requested video bitrate for block k is $v(k) = Q(\tilde{v}(k))$, where $Q(\cdot)$ denotes the quantization function.

In summary, the proposed rate adaption scheme is described as: whenever a block is downloaded completely: 1) the block length is updated for the next block by Table II; 2) the fragments are scheduled to multiple servers by Table III; and 3) all fragments belonging to the new block are set with the same video rate according to (16).

D. Parameter Analysis

In this subsection, we analyze parameters K_p and K_d to stabilize the close-loop system described in Fig. 4. The settling time of the control system is also analyzed. The overall system transfer function is expressed by

$$H_c(s)=\frac{K_d s+K_p}{(T\alpha(n)v_0+K_d)s+K_p}. \quad (17)$$

Suppose that it takes no more than mT seconds to switch to the suitable bitrate, where $m > 0$. Now, we have the following propositions.

Proposition 2: The feedback system in Fig. 4 is stabilized by the PD controller, if the parameters satisfy the following conditions:

$$\begin{cases} w_c \geq \frac{1}{mT}\sqrt{\frac{TN_k+K_d}{TN_k-K_d}}\ln\frac{20TN_k}{TN_k+K_d} \\ K_p=\sqrt{(TN_k)^2-K_d^2}w_c \\ 0<K_d<TN_k. \end{cases} \quad (18)$$

Proof: The proof is given in Appendix B. ■

Proposition 2 guarantees the convergence of our proposed rate adaption approach, i.e., it is guaranteed that a suitable video bitrate will be selected, and bitrate oscillations will be avoided.

Proposition 3: For a PD controller satisfying Proposition 2, the 5% step response settling time (T_s) of the feedback control system satisfies $T_s \leq mT$.

Proof: The proof is given in Appendix C. ■

In this proposition, the term of 5% step response settling time means the time needed for the output converge to at least 95% of the target value, i.e., the gap between the practical output and the target value is within 5% of the target value. The proposition shows that the time needed to switch to the suitable bitrate is no more than mT. Thus, by reserving appropriate amount of buffered video before adapting video bitrate (i.e. by setting appropriate thresholds $q_{\min}$ and $q_{\max}$), buffer underflow and overflow can both be avoided. As will be shown in Section VII, the experimental results justify these characteristics of the proposed rate adaptation approach.

VI. Implementation Details

In this section, we discuss some implementation details about our proposed rate adaption approach.

A. TCP Receiving Bandwidth Estimation

TCP receiving bandwidth estimation is indispensable in our proposed rate adaptation approach, by which the receiving bandwidth is estimated in two cases. For a server that is continuously selected by Table II, we use simple history-based TCP throughput estimation [20]. The client measures the time it takes to download each fragment and calculates the average TCP throughput for each fragment. To estimate TCP throughput for the next block, we simply take an average of the TCP throughput estimated for the previous W fragments, discarding the largest one and the smallest one. If W is too large, the estimation reacts slowly to sudden congestion level shift, whereas if W is too small, the estimation tends to be noisy. In our system, we set $W = 8$ which usually leads to fairly accurate estimation results.

In the case that a server is not selected by Table II, there is no TCP throughput history to extrapolate on. In this case, we use support vector regress (SVR) TCP throughput estimation [21] with initial offline training and light-weight online measurement. Mirza *et al.* [21] showed that TCP throughput is mainly determined by packet loss, delay, and the size of the file to be downloaded. They propose to use SVR algorithm to train a TCP throughput model out of training data consisting of samples of packet loss rate, packet delay, file size, and the corresponding actual TCP throughput. Since in DASH, we download each fragment as a separate file and fragments with different video bitrates have different file size, we replace the file size by the video bitrate during the SVR TCP throughput estimation.

In fact, any existing receiving bandwidth estimation methods can be used in our proposed rate adaption framework. We choose the history-based and SVR estimation methods due to their excellent performance, as reported in [14].

B. Sleeping Mechanism

If the requested video bitrate is consistently lower than the TCP throughput, the buffer occupancy quickly ramps up, leading to buffer overflow. The easiest method is to select a higher video bitrate. But, if the TCP throughput is higher than the highest video bitrate, buffer overflow is unavoidable. Therefore, a sleeping mechanism is necessary to prevent buffer overflow. In our proposed system, when the following three conditions are all satisfied: 1) the requested video bitrate is the highest available video bitrate; 2) the buffer occupancy is higher than $q_{\max}$; and 3) the buffer occupancy has been consistently increasing, the rate adaptation algorithm will wait for a period of time before sending the next request until the buffer occupancy goes down to two-thirds of the buffer size.

C. Timeout Retransmission

Although fragment requests are scheduled according to their playback deadlines, if the connection to a server suddenly incurs bandwidth deficiency, the fragment already assigned to that connection still cannot be downloaded in time, which will delay video playback. To address this problem, a timeout retransmission mechanism is introduced. Based on the bandwidth estimation, we roughly calculate the expected transmission time for each fragment. When a fragment has not

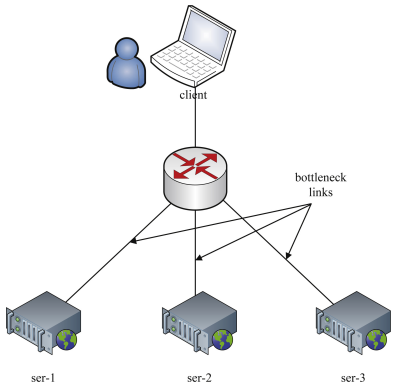


Fig. 5. Experimental topology for the test bed.

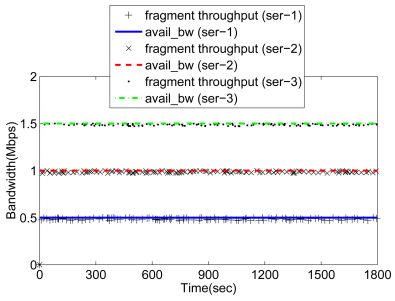


Fig. 6. Persistent bandwidth for three servers.

been downloaded completely in two times of its expected transmission time, a download request for the fragment is resent to another server.

D. Single-Server DASH

Although the rate adaption approach proposed in this paper is designed for multiple servers, it is also suitable for single-server DASH. When it is applied to single-server DASH, the rate adaption is implemented in fragment granularity (i.e., the block length $N_k = 1$ for all blocks) and the size of binary indicator matrix $\mathbf{X}$ is 1×1 with entry $x_{11} = 1$. Since each block contains only one fragment, we have

$$frag\_t^e_{n,k-1}|_{n=1} = block\_t^e_{k-1} \tag{19}$$

and $\alpha(n) = \frac{1}{c_1}$, then the rate adjustment of block k (in the case of single-server DASH, block k also denotes fragment k) becomes

$$\delta(v(k)) = \frac{c_1}{T} K_p \left(q\left(block\_t^s_k\right) - q_0\right) + \frac{c_1}{T} K_d \frac{q\left(block\_t^e_k\right) - q\left(block\_t^s_k\right)}{block\_t^e_k - block\_t^s_k} \tag{20}$$

and $v_0 = c_1$, where c_1 is the predicted bandwidth of the single server. Last, the conditions for the system parameters are easily derived from Proposition 2 by setting $N_k = 1$. A short version of this part can refer to our previous work [22].

VII. Performance Evaluation

In this section, we evaluate our video bitrate adaption algorithms using controlled experiments on a network test bed, as well as using uncontrolled experiments in real Internet.

A. Experiments Setup

We implemented our MCDS-DASH system in Linux/Unix platforms. As shown in Fig. 5, our testbed consists of five nodes: three web servers, one router, and one DASH client. The servers and client run the standard Ubuntu of version 12.04.1. The servers were installed with Apache HTTP server of version 2.4.1. Also, Dummynet is used in the servers to control the upload bandwidth; therefore, the bottleneck is the bandwidth between the servers and the router. We also implemented the SVR-based TCP throughput estimation algorithm proposed in [21] for bandwidth prediction. However, if a server is continuously selected by Table II, we simply use history-based TCP throughput prediction [20] because it has higher accuracy than the SVR-based TCP throughput estimation method [21]. In our experiments, same with Netflix, all servers provide five different versions of video bitrates {300 Kb/s, 700 Kb/s, 1.5 Mb/s, 2.5 Mb/s, 3.5 Mb/s}. Each video is divided into small equal-length video fragments, each is of the length of 5 s.

For performance comparison, we extended the previous feedback-based rate adaption method proposed in [13] to multiple servers. By controlling the sender buffer size, the method aims to adapt the video bitrate to the available channel bandwidth. Another compared method is proposed in [14], which is among the first work to study the rate adaption problem for MCDS-DASH. For the purpose of clarity, the method in [13] is called feedback-based rate adaption (FRA), the method in [14] is called smooth video adaption (SVA), and our proposed method is called control-theoretic rate adaption (CTRA).

B. Impact of Parameter Setting

We first analyze the effect of the parameter settings ($q_{\min}, q_{\max}$, and the accuracy of bandwidth estimation) on the system performance. To eliminate the effect of bandwidth variations on system performance, for simplicity, we fixe the bandwidth of all servers as illustrated in Fig. 6, where, *avail-bw* denotes the available bandwidth controlled by Dummynet, and fragment throughput refers to the download throughput for a particular fragment. i.e., the size of that fragment divided by the corresponding download duration. The result shows that the fragment throughput tracks the available bandwidth *avail-bw* quite well.

We compare the results under different thresholds ($q_{\min}, q_{\max}$) in Fig. 7, in which we label x-axis by fragment index as it directly reflects the smoothness and quality level of playback video bitrate. From Fig. 7(a) to (f), we find that small $q_{\min}$ and high $q_{\max}$ lead to smooth video bitrate, while increasing the risk of buffer underflow, and vice versa. For example, when $q_{\min} = 5$ s and $q_{\max} = 55$ s, Fig. 7(a) shows that the video bitrate is switched most slowly, but the buffered video time approaches zero several times, as shown in Fig. 7(d), which leads to playback freeze that deteriorates the quality of user experience. On the other hand, when $q_{\min} = 20$ s and $q_{\max} = 40$ s, the buffered video time ensures continuous video playback, but the video bitrate fluctuate frequently as shown in Fig. 7(f) and (c).

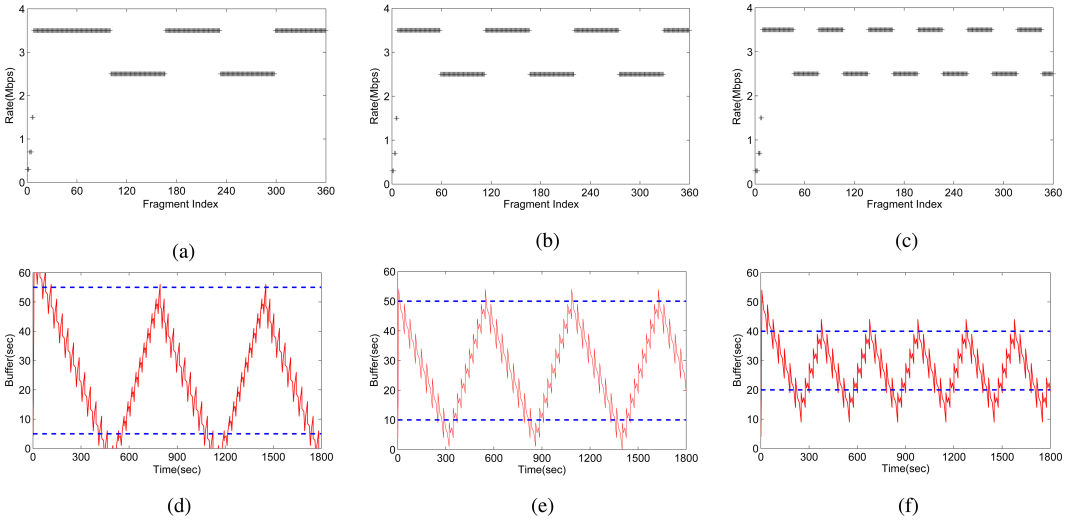


Fig. 7. Video bitrate adaption under $q_{\min}$ and $q_{\max}$. (a)–(c) Video bitrate. (d)–(f) Buffer size evolution. (a) and (d) $q_{\min}$ = 5 s, $q_{\max}$ = 55 s. (b) and (e) $q_{\min}$ = 10 s, $q_{\max}$ = 50 s. (c) and (f) $q_{\min}$ = 20 s, $q_{\max}$ = 40 s.

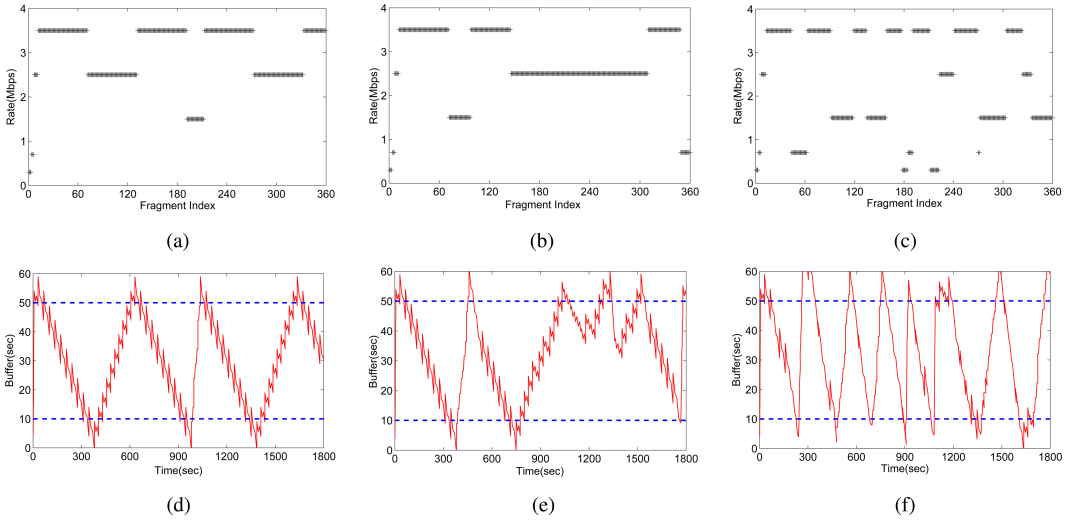


Fig. 8. Video bitrate adaption under different additional bandwidth estimation error rates. (a)–(c) Video bitrate. (d)–(f) Buffer size evolution. (a) and (d) Bandwidth estimation error rate: 10%. (b) and (e) Bandwidth estimation error rate: 20%. (c) and (f) Bandwidth estimation error rate: 40%.

Due to the complex network characteristics, it is hard to find the optimal values of the two thresholds. In the following experiments, we set $q_{\min}$ = 10 s and $q_{\max}$ = 50 s since it achieves a good tradeoff between smooth bitrate and low risk of buffer underflow. Then, we set m = 2. Thus, according to Proposition 3, the buffer occupancy varies between 0 s ($q_{\min} - 2T$ = 0 s) and 60 s ($q_{\max} + 2T$ = 60 s), thereby ensuring continuous video playback, as justified by the experiment results. The maximal buffer size of all schemes is set to 60 s.

We also verify the robustness of our proposed rate adaption scheme against bandwidth estimation error. Although we cannot obtain the actual bandwidth, Fig. 6 shows that the estimated fragment throughput can be used as a good approximate of the actual bandwidth since it tracks the available bandwidth *avail-bw* quite well. Thus, we add some disturbances to the estimated fragment throughput to simulate different levels of the bandwidth estimation errors. The results in Fig. 8 show that our rate adaption scheme can tolerate bandwidth estimation error of up to 20% or more without severely degrading the performance. Such error tolerance is due to that the video bitrate selection decision is made based on both the estimated bandwidth and the changing rate of buffered video time, where the latter is not affected by the bandwidth estimation error.

TABLE IV
PERFORMANCE SUMMARY UNDER SHORT-TERM BANDWIDTH VARIATIONS

scheme	average video bitrate (Mbps)	bandwidth utilization (%)	average buffered time (second)
FRA	2.50	84.04	35.75
SVA	2.41	80.70	37.64
CTRA	2.84	95.41	29.17

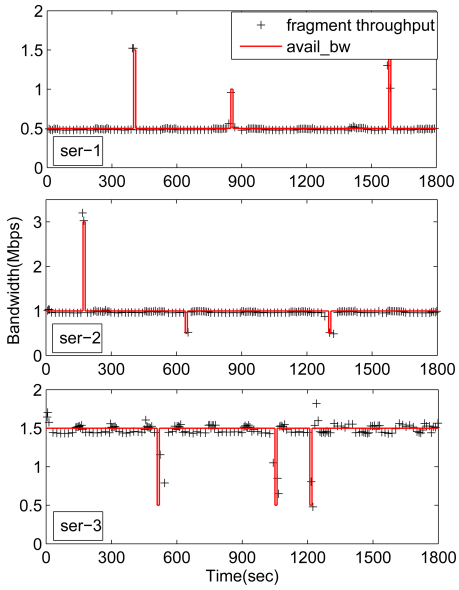


Fig. 9. Short-term bandwidth variations for three servers.

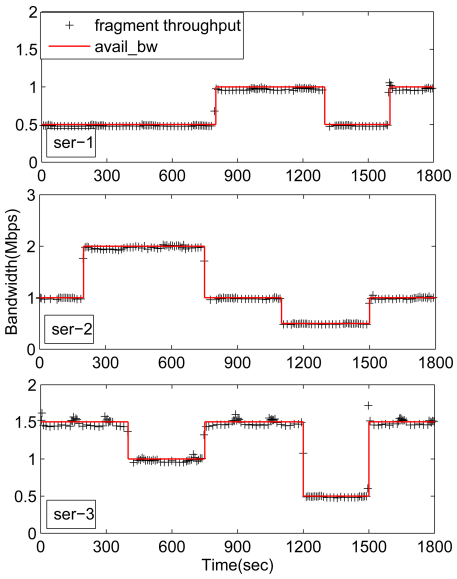


Fig. 10. Long-term bandwidth variations for three servers.

C. Impact of Short-Term Bandwidth Variations

In this subsection, we summarize the results of experiments where the available bandwidth goes through positive or negative spikes that last for a few seconds, as shown in Fig. 9. All spikes last for 10 s. Such short-term variations are common in practice and we think that a good rate adaptation scheme should be able to compensate for such spikes using its buffered video, without causing short-term rate switching.

We evaluate the performance of all three schemes under the short-term bandwidth variations, as shown in Fig. 11. Fig. 11(a)–(c) shows that both FRA and SVA react to the spikes by increasing/decreasing the requested video bitrate, whereas SVA ignores some spikes by using a counter before increasing the video bitrate. CTRA smoothes out all spikes by employing two operating points which is similar to a low-pass filter. The figures also show that both FRA and SVA keep requesting the video bitrate of 2.5 Mb/s except at some spike instances, which is the largest video bitrate smaller than the persistent bandwidth ($\approx$0.5 M/s + 1 Mb/s + 1.5 Mb/s = 3 Mb/ps). As a result, their buffered video length shows obvious periodic oscillations that the buffered video length keeps increasing to a threshold (55 s in our experiments); then, the rate adaption algorithm sleeps for a period of time before sending the next request. In CTRA, the video bitrate is switched between 2.5 Mb/s and 3.5 Mb/s. Since a video bitrate that is higher than the available bandwidth is allowed to be selected in CTRA, higher bandwidth utilization efficiency is achieved. It is also worth noticing that the video playback bitrate keeps unchanged for at least 250 s (about 50 fragments), making long-term rate switching infrequent and thereby avoiding significant degradation on user viewing experience. Last, the buffer size evolution results in Fig. 11(a)–(c) show that the buffer occupancy in FRA and SVA changes much more sharply than that in CTRA. This is mainly because both FRA and SVA request fragments in order without considering their download completion time. When a fragment is downloaded completely later than some of its succeeding fragments, the buffered video length may increase quickly. In CTRA, the fragments are requested according to their playback deadlines, the fragments are completely downloaded in order and the buffer occupancy changes gently. Besides, the result in Fig. 11(f) shows that the buffered video time varies between 0 s and 60 s, which is consistent with Proposition 3.

As shown in Table IV, CTRA achieves higher average video bitrate and bandwidth utilization efficiency compared to both FRA and SVA. Also, the average buffer occupancy in CTRA is closer to the half of the buffer size than that of FRA and SVA, making CTRA more reliable to provide continuous video playback without buffer overflow/uunderflow. Based on these results and the fragment video bitrate results in Fig. 11(a)–(c), we conclude that CTRA provides much smoother and higher video bitrate than FRA and SVA do, thereby achieving better user viewing experience.

D. Impact of Long-Term Bandwidth Variations

We then evaluate the impact of long-term bandwidth variations under the scenario depicted in Fig. 10. From Fig. 12(a) and (b), we can observe that both FRA and SVA switch video bitrate as the bandwidth level changes. According to Fig. 10, the total bandwidth approximately changes at the levels of $\{3\,\text{Mb/s} \rightarrow 4\,\text{Mb/s} \rightarrow 3$,Mb/s $\rightarrow 3.5\,\text{Mb/s} \rightarrow 3\,\text{Mb/s} \rightarrow 2\,\text{Mb/s} \rightarrow 1.5\,\text{Mb/s} \rightarrow 3\,\text{Mb/s} \rightarrow 3.5\,\text{Mb/s}\}$ are orderly, and the playback video bitrates for both FRA and SVA change the order at the levels as $\{2.5\,\text{Mb/s} \rightarrow 3.5\,\text{Mb/s} \rightarrow 2.5\,\text{Mb/s} \rightarrow 3.5\,\text{Mb/s} \rightarrow 2.5\,\text{Mb/s} \rightarrow 1.5\,\text{Mb/s} \rightarrow 2.5\,\text{Mb/s} \rightarrow 3.5\,\text{Mb/s}\}$ except some bitrate spikes. CTRA switches video bitrate in a much smoother way than both FRA and SVA, because CTRA adapts video bitrate by considering not only the bandwidth levels, but also the feedback buffer occupancy information, such as the difference between current buffer occupancy and operating points, and the changing rate of

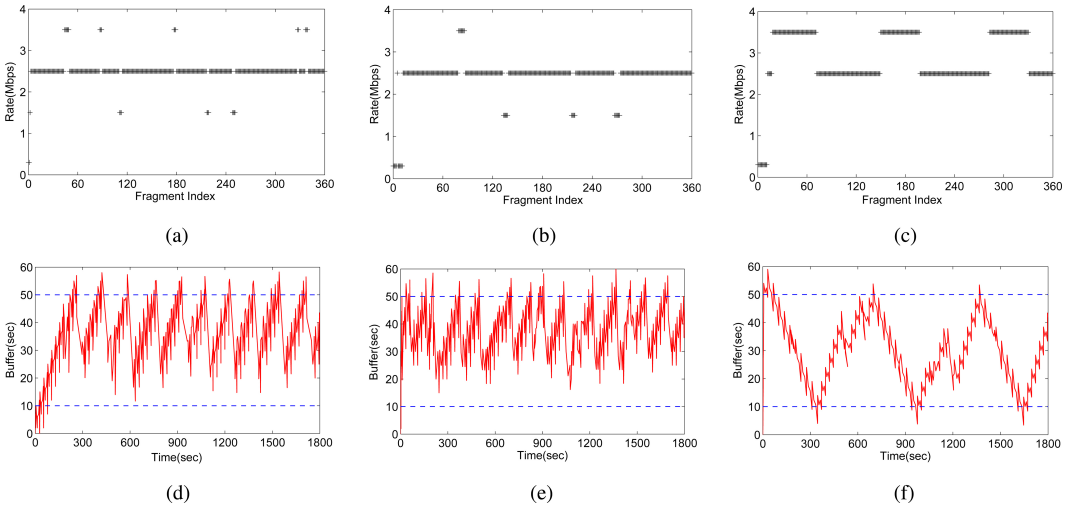


Fig. 11. Video bitrate adaption under short-term bandwidth variations. (a)–(c) Video bitrate. (d)–(f) Buffer size evolution. (a) and (d) FRA. (b) and (e) SVA. (c) and (f) CTRA.

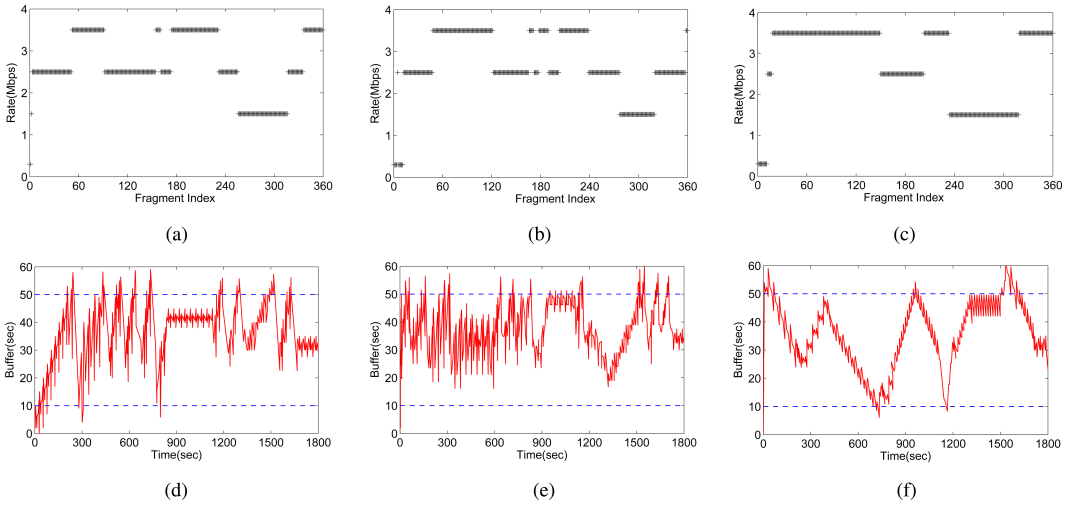


Fig. 12. Video bitrate adaption under long-term bandwidth variations. (a)–(c) Video bitrate. (d)–(f) Buffer size evolution. (a) and (d) FRA. (b) and (e) SVA. (c) and (f) CTRA.

buffer occupancy. It is worth noticing that for both FRA and SVA, there usually exist some bitrate spikes when video bitrate switches. This is mainly because both schemes adapt video bitrate at the granularity of fragment, thus the requested video bitrates are updated asynchronously for multiple servers due to their heterogeneous bandwidth. The buffer size evolution results are shown in Fig. 12(d)–(f). The buffer occupancy in CTRA changes significantly more gently than that in both FRA and SVA, as explained in the previous subsection. Table V summarizes the results of all schemes, and similar to Table IV, it also demonstrates the effectiveness of CTRA under long-term bandwidth variations.

TABLE V
PERFORMANCE SUMMARY UNDER LONG-TERM BANDWIDTH VARIATIONS

scheme	average video bitrate (Mbps)	bandwidth utilization (%)	average buffered time (second)
FRA	2.67	85.48	36.16
SVA	2.68	85.58	37.53
CTRA	2.86	91.43	35.35

E. Internet Experiments

Finally, we test our proposed scheme on the Internet. We set up three Planetlab nodes as the DASH servers; they are located in Hong Kong, China (plab1.cs.ust.hk), Japan (planet1.pnl.nitech.ac.jp), and South Korea (pl2.yonsei.ac.kr), respectively. Another Planetlab node in Beijing, China (pl1.pku.edu.cn) is set up as the DASH client. We do not inject any background traffic between the servers and the client.

Fig. 13 and Table VI show the results of various schemes. We conduct different experiments sequentially, and the bandwidth patterns are not controllable. In all experiments, we do observe long-term shift and short-term fluctuations of bandwidth along the Internet path, as illustrated in Fig. 13(g)–(i).

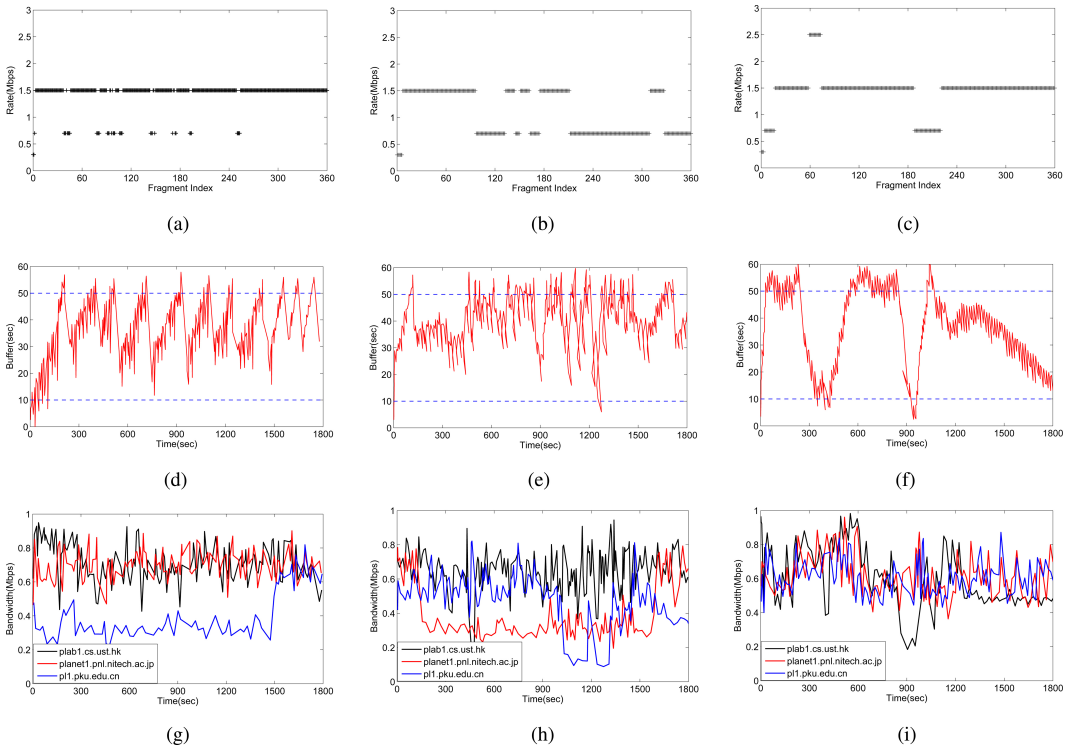


Fig. 13. Video bitrate adaption over Internet. (a)–(c) Video bitrate. (d)–(f) Buffer size evolution. (g)–(i) Fragment bandwidth variations. (a), (d), and (g) FRA. (b), (e), and (h) SVA. (c), (f), and (i) CTRA.

The video bitrate results in Fig. 13(a)–(c) demonstrate that our video bitrate adaptation algorithms can adapt well to the network conditions, which is consistent with our test-bed-based results. It is able to compensate for shot-term spikes using its buffered video, without causing short-term rate switchings. While for long-term bandwidth variations, it switches to a suitable video bitrate without causing buffer overflow or playback interruptions. In fact, the video bitrate is even smoother than our test bed-based experiments. This is because the bandwidth in the test bed experiments is changed more abruptly than the real cross-traffic along the Internet path. We can observe many short-term rate spikes with FRA and SVA, especially with FRA, where almost all the short-term rate switchings are unnecessary since there is enough buffered video data. By employing a counter, SVA outperforms FRA since the video bitrate can be switched only when several consecutive fragments have been completely downloaded. Similar to the test bed-based results, the buffer occupancy with FRA and SVA changes much more drastically than CTRA does; this is mainly because they both request video fragments sequentially without considering their playback deadlines.

Last, the results shown in Table VI clearly show that our proposed scheme achieves the highest bandwidth utilization and average video bitrate. Although the average video bitrate of CTRA is only slightly higher than FRA, CTRA achieves significantly higher bandwidth utilization compared to FRA. This is because we conduct different schemes sequentially and the bandwidth patterns are not controllable on the Internet. The low bandwidth utilization of FRA and SVA is mainly caused by their sleeping mechanisms that both schemes never allow to request a fragment whose video bitrate is higher than the available bandwidth. With the higher bandwidth utilization and smoother video rate justified by the above test results, we can confidently conclude that given any bandwidth pattern, CTRA can provide much better user viewing experience than FRA and SVA, while still keeping the average buffer occupancy of CTRA at a moderate level.

TABLE VI
PERFORMANCE SUMMARY UNDER INTERNET

scheme	average video bitrate(Mbps)	bandwidth utilization(%)	average buffered time (second)
FRA	1.41	76.42	36.82
SVA	1.07	71.38	40.77
CTRA	1.43	95.43	33.28

VIII. CONCLUSION

In this paper, we proposed a novel approach for video rate adaption for DASH over MCDS. By grouping multiple fragments into a block, we propose to adapt video bitrate at the granularity of block. Considering the instantaneous bandwidth of multiple servers, our method dynamically adapts the block length and schedules the fragments to multiple servers depending on their playback deadlines; thus, the request video bitrates for all servers are synchronized and the fragments are completely downloaded in order. Combining with the

heterogeneous bandwidth and feedback buffer occupancy information, we have proposed a control-theoretic approach to select the suitable video bitrate for each block, specifically a PD controller is adopted to improve rate adaption performance. By carefully designing the parameters for the PD controller, our method can balance the needs for video bitrate smoothing and bandwidth utilization. Theoretical analysis and extensive experiments on network testbed and the Internet both justify the high efficacy of our DASH designs.

APPENDIX A
PROOF OF PROPOSITION 1

Proof: According to (1), if $n_i = \gamma_i$, server i will complete the downloads earlier than server S_k, and it will keep idle for a period of time. Suppose the current video bitrate is $v(k)$, the wasted bandwidth is

$$\mathcal{W}_{bw}^{i} = \left(\frac{v(k)T}{c_{S_k}} - \frac{n_i v(k)T}{c_i} \right) c_i = \mu_i v(k)T \tag{21}$$

where $v(k)T$ is the fragment size. On the other hand, if $n_i = \gamma_i + 1$, server i will complete the downloads later than server S_k, and server S_k keeps idle for a period of time. Then the wasted bandwidth is

$$\mathcal{W}_{bw}^{S_k} = \left(\frac{(n_i+1)\, v(k)T}{c_i} - \frac{v(k)T}{c_{S_k}} \right) c_{S_k} = \frac{1-\eta_i}{\gamma_i+\eta_i} v(k)T. \tag{22}$$

It is obvious that when η_i is smaller than the dynamic threshold μ_i in (3), we have $\mathcal{W}_{bw}^{i} < \mathcal{W}_{bw}^{S_k}$; otherwise, we have $\mathcal{W}_{bw}^{i} \geq \mathcal{W}_{bw}^{S_k}$. Thus, when the number of fragments assigned to server i is determined by (2) with dynamic quantification threshold in (3), we obtain the minimal bandwidth waste and Proposition 1 is proven. ■

APPENDIX B
PROOF OF PROPOSITION 2

Proof: According to Proposition 18, it is obvious that the only pole of (17) is

$$s_p = -\frac{K_p}{T\alpha(n)\, v_0 + K_d} < 0. \tag{23}$$

It is located in the left phase plane. The system is proven to be stable. ■

APPENDIX C
PROOF OF PROPOSITION 3

Proof: The step response of the system in Fig. 4 is

$$U_c(s) = \frac{K_d s + K_p}{(T\alpha(n)\, v_0 + K_d)\, s + K_p} \frac{1}{s}. \tag{24}$$

Applying the inverse Laplace transform, we have

$$u_c(t) = u(t) \left(1 - \frac{T\alpha(n)\, v_0}{T\alpha(n)\, v_0 + K_d} e^{-\frac{K_p}{T\alpha(n)v_0 + K_d} t} \right). \tag{25}$$

From the constraints in Proposition 18, the 5% step response settling time (T_s) of the feedback control system is

$$\begin{aligned} T_s &= \frac{T\alpha(n)\, v_0 + K_d}{\sqrt{(TN_k)^2 - K_d^2 w_c}} \ln \frac{20 T\alpha(n)\, v_0}{T\alpha(n)\, v_0 + K_d} \\ &\leq \frac{T\alpha(n)\, v_0 + K_d}{TN_k + K_d} \frac{\ln \frac{20T\alpha(n)v_0}{T\alpha(n)v_0 + K_d}}{\ln \frac{20TN_k}{TN_k + K_d}} mT \leq mT. \end{aligned} \tag{26}$$

The last inequality is derived by that $\alpha(n) \leq \alpha(N_k)$ for any $n = 1, 2, ..., N_k$; this is because fragment N_k would be completely downloaded at last according to Table III. ■

REFERENCES

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

[2] R. Kuschnig, I. Kofler, and H. Hellwagner, "Evaluation of HTTP-based request-response streams for Internet video streaming," in *Proc. ACM MMSys11*, Feb. 2011, pp. 245–256.

[3] T. Stockhammer, "Dynamic adaptive streaming over HTTP: Standards and design principles," in *Proc. ACM MMSys11*, Feb. 2011, pp. 133–144.

[4] N. Cranley, P. Perry, and L. Murphy, "User perception of adaption video quality," *Int. J. Human-Comput. Stud.*, vol. 64, no. 8, pp. 637–647, 2006.

[5] E. C. R. Mok, X. Luo, and R. Chang, "QDASH: A QoE-aware DASH system," in *Proc. ACM Multimedia Syst.*, Feb. 2012, pp. 11–22.

[6] M. Watson, "HTTP adaptive streaming in practice," Netflix, Tech. Rep., 2011. [Online] Available: http://web.cs.wpi.edu/~claypool/mmsys-2011/Keynote02.pdf

[7] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, *et al.*, "Unreeling Netflix: Understanding and improving multi-CDN movie delivery," in *Proc. INFOCOM*, Mar. 2012, pp. 1620–1628.

[8] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. ACM MMSys11*, Feb. 2011, pp. 169–174.

[9] A. Zambelli, *IIS Smooth Streaming Technical Overview*, Microsoft Corp., 2009. [Online] Available: http://www.microsoft.com/en-us/download/details.aspx?id=17678

[10] Adobe. *Open Source Media Framework 1.6*, (Sep. 2011) [Online]. Available: http://www.osmf.org/

[11] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. ACM MMSys11*, Feb. 2011, pp. 169–174.

[12] B. Zhou, J. Wang, Z. Zou, and J. Wen, "Bandwidth estimation and rate adaptation in HTTP streaming," in *Proc. IEEE ICNC*, Feb. 2012, pp. 734–738.

[13] L. De Cicco, S. Mascolo, and P. Vittorio, "Feedback control for adaptive live video streaming," in *Proc. ACM MMSys11*, Feb. 2011, pp. 145–156.

[14] G. Tian and Y. Liu, "Toward agile and smooth video adaption in dynamic HTTP streaming," in *Proc. ACM CoNEXT*, Dec. 2012, pp. 109–120.

[15] Y. Huang, S. Mao, and S. Midkiff, "A control-theoretic approach to rate control for streaming videos," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp. 1072–1081, Oct. 2009.

[16] S. Sanadhya and R. Sivakumar, "Adaptive flow control for TCP on mobile phones," in *Proc. INFOCOM*, Apr. 2011, pp. 2912–2920.

[17] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proc. INFOCOM*, Mar. 2001, pp. 1510–1519.

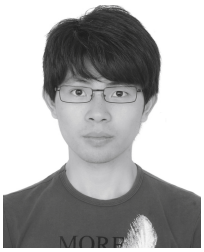
[18] J. Choi and D. Park, "A stable feedback control of the buffer state using the controlled lagrange multiplier method," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 1510–1519, Sep. 1994.

[19] K. Astrom and R. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton Univ. Press, 2008.

[20] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," in *Proc. ACM SIGCOMM*, vol. 35, no. 4. Oct. 2005, pp. 145–156.

[21] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to TCP throughput prediction," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1026–1039, Aug. 2010.

[22] C. Zhou, X. Zhang, L. Huo, and Z. Guo, "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," in *Proc. VCIP*, Nov. 2012, pp. 1–6.



Chao Zhou received the B.S. degree in communication engineering from Beijing University of Technology, Beijing, China, in 2005. He has been working toward the Ph.D. degree at the Institute of Computer Science and Technology, Peking University, Beijing, since 2009.

His research interests include video streaming, joint source channel coding, and rate adaptation.

Mr. Zhou received the Best Student Paper Awards presented by IEEE VCIP 2012.

Chia-Wen Lin (S'94–M'00–SM'04) received the Ph.D. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2000.

He is currently an Associate Professor with the Department of Electrical Engineering and the Institute of Communications Engineering, NTHU. He was with the Department of Computer Science and Information Engineering, National Chung Cheng University, Minxueng, Taiwan, from 2000 to 2007. Prior to joining academia, he was with the Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, from 1992 to 2000. His research interests include image and video processing and video networking.

Dr. Lin is an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE MULTIMEDIA, and *Journal of Visual Communication and Image Representation*. He is also an Area Editor of *EURASIP Signal Processing: Image Communication*. He serves as the Chair of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and System Society. He was the Technical Program Co-Chair of the IEEE International Conference on Multimedia and Expo (ICME) in 2010 and the Special Session Co-Chair of the IEEE ICME in 2009. His paper won the Young Investigator Award presented by VCIP 2005. He received the Young Faculty Awards presented by CCU in 2005 and the Young Investigator Awards presented by the National Science Council, Taiwan, in 2006.

Xinggong Zhang (M'11) received the B.S. degree from Harbin Institute of Technology, Harbin, China, and the M.S. degree from Zhejiang University, Hangzhou, China, in 1995 and 1998, respectively. He received the Ph.D. degree from Peking University, Beijing, China, in 2011.

He is currently an Associate Professor with the Institute of Computer Science and Technology, Peking University. He was a Senior Research Staff with Founder Research China. His general research interests include multimedia networking and video communications. His research interests include video conferencing, dynamic HTTP streaming, and content-centric networking.

Dr. Zhang received the First Prize of the Ministry of Education Science and Technology Progress Award in 2006 and the Second Prize of the National Science and Technology Award in 2007.

Zongming Guo was born in 1966. He received the B.Sc. degree in mathematics, and the M.Sc. and Ph.D. degrees in computer science, from Peking University, Beijing, China, in 1987, 1990, and 1994, respectively.

He is currently the Deputy Dean of the Institute of Computer Science and Technology, Peking University. He has published over 80 technical articles in refereed journals and proceedings in the areas of multimedia, image and video compression, image and video retrieval, and watermarking. His research interests include network video technology, including streaming media technology, IPTV, and mobile multimedia.

Dr. Guo led the Research and Developing Team, which won the First Prize of the State Administration of Radio Film and Television, the First Prize of the Ministry of Education Science and Technology Progress Award, and the Second Prize of the National Science and Technology Award in 2004, 2006, and 2007, respectively. He received a government allowance granted by the State Council in 2009.