Adaptive Video Streaming over HTTP with Dynamic Resource Estimation

Truong Cong Thang, Hung T. Le, Hoc X. Nguyen, Anh T. Pham, Jung Won Kang, and Yong Man Ro

Abstract: **Adaptive hypertext transfer protocol (HTTP) streaming has become a new trend to support adaptivity in video delivery. An HTTP streaming client needs to estimate exactly resource availability and resource demand. In this paper, we focus on the most important resource which is bandwidth. A new and general formulation for throughput estimation is presented taking into account previous values of instant throughput and round trip time. Besides, we introduce for the first time the use of bitrate estimation in HTTP streaming. The experiments show that our approach can effectively cope with drastic changes in connection throughput and video bitrate.**

Index Terms: **Adaptivity, bitrate estimation, HTTP streaming, throughput estimation.**

I. INTRODUCTION

Thanks to the abundance of web platforms and broadband connections, hypertext transfer protocol (HTTP) streaming has become a cost effective means for multimedia delivery [1]–[3]. Besides, due to the heterogeneity of today's communication networks, adaptivity is the most important requirement for any streaming client [3]. Especially, transmission control protocol (TCP), the underlying layer of HTTP, is notorious for its throughput fluctuations [4]. Moreover, the bitrate of a video encoded in variable bitrate (VBR) mode may also vary widely according to the characteristics of the content [5]. So, the mismatch of both throughput and video bitrate is a big challenge in video streaming.

For adaptivity to networks and terminal capabilities, an HTTP streaming provider should generate multiple alternatives (or versions) of an original video as well as the signaling metadata that contains the characteristics of the alternatives (such as bitrate, resolution, etc.) [6]. Based on the metadata and status of terminal/networks, the client makes decisions on which/when media parts are downloaded. This client-based approach is fundamentally different from the conventional server-based approach (e.g., [7]), where the server plays a decisive role in streaming. In order to make good decisions, the client needs to estimate correctly 1) resource availability and 2) resource demands. In this paper, we focus on the most important resource type, which is bandwidth/bitrate.

In video streaming, if the actual throughput is lower than the estimated throughput or similarly if the actual video bitrate is higher than the specified bitrate, video data transmission will be delayed and the decoding buffer will quickly become empty. To cope with errors in both estimated throughput and specified video bitrate, a client should buffer some amount of video data before it can start playing [3], [9]. Obviously, if the amount of buffered data is large, the client can better cope with the future mismatches. However, this action results in the so-called *initial buffering delay* (sometimes up to tens of seconds), which badly affects the quality of experience, especially for live streaming [9]. So, the accuracy of throughput and video bitrate information will be crucial to maintain a low and stable buffer level for a streaming client. To this end, the main contributions of this paper are as follows.

First, we propose a new and general formulation for throughput estimation taking into account previous values of instant throughput and round trip time (RTT). Currently, throughput estimation is based on the previous segment throughputs [3], [10], [11], which are average values that may not capture the fast bandwidth fluctuations when the segment duration is long. Second, we introduce for the first time the use of bitrate estimation in HTTP streaming. With bitrate estimation, the client will be able to dynamically select the highest possible bitrate at any time. So far, previous studies have dealt with constant bitrate (CBR) video only. To the best of our knowledge, our previous standard contribution [12] is the first work that has highlighted the importance of instant bitrate information in HTTP streaming. As shown later, it is interesting that this solution may enable CBR-streaming even though the video is encoded in VBR mode.

The paper is organized as follows. In Section II, we first provide an overview of adaptive HTTP streaming, bitrate concept, and related work. A systematic method to estimate the throughput is proposed in Section III. In Section IV, we present the mechanisms for the client to estimate the instant bitrate for VBR video content. Experiments with different scenarios are presented in Section V. Finally, conclusions and future work are given in Section VI.

II. OVERVIEW OF ADAPTIVE HTTP STREAMING

A. HTTP Streaming and Bitrate Adaptation

As discussed in [3], [13], the general architecture of adaptive HTTP streaming consists of servers, delivery networks, and clients. Video versions together with their metadata are hosted at some servers and will be requested by the client. Based on the metadata and status of terminal/networks, the decision engine at the client makes decisions on which/when media parts
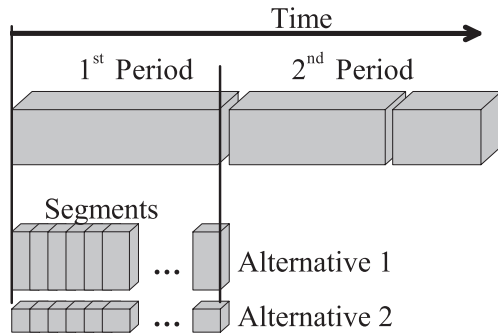
Fig. 1.   Hierarchy of content division in MPEG DASH.

Fig. 2.   Illustration of bitrate B with initial delay d_0.

are downloaded.

Recently, a new standard called dynamic adaptive streaming over HTTP (DASH) has been developed by international organization for standardization (ISO)/international electrotechnical commission (IEC) moving picture experts group (MPEG), specifying the metadata and media formats exchanged between clients and servers [14]. In MPEG DASH's terminology, the metadata is called *media presentation description* (MPD). A long content item could be divided into one or more temporal chapter (called *period*). Alternatives (called *representations*) having some common characteristics (e.g., same content component) are grouped into an *adaptation* set. Further, each representation could be divided into media *segments*. An illustration of media division hierarchy is shown in Fig. 1. More information about the structure and basic concepts of DASH could be found in [2], [13].

Media will be delivered by a sequence of HTTP request-response transactions. In most cases, for each request from the client, the server will send one segment. The term "initial buffering" in this paper means the length (in seconds) of media needed in the buffer before the playout can start.

HTTP streaming can be applied to both on-demand streaming and live streaming. The main difference between these two cases is the available time of segments. In live streaming, the time distance between the requests of two consecutive segments is approximately the duration of the first segment. So, if segments have the same duration of τ seconds, the distance between requests will be τ as well. Meanwhile, in on-demand streaming, requests could be sped up to quickly fill the buffer [15]. Note that in live streaming, the playback takes place with a short delay (typically less that 10 s), so the client should maintain a small buffer.

As we focus on the difficult problem of maintaining a low and stable buffer level, the initial buffering is also the target buffer level to be kept during a session.

In general, the process of bitrate adaptation takes into account 1) the estimated throughput and 2) the bitrates of alternatives which are specified in the metadata (MPD). For each segment interval, the bitrate can be decided as the highest value of the alternatives' specified bitrates that is smaller than the estimated throughput. Further, if the client has the ability to estimate instant bitrates of alternatives, the estimated bitrates will be used instead of specified bitrates. Throughput estimation and bitrate estimation will be tackled respectively in Sections III and IV.
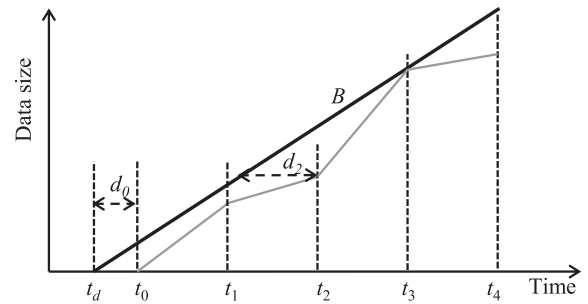
B. Bitrate Concept

Though bitrate is one of the most important concepts in video transport, its definition is actually not simple. The definition of bitrate depends on two basic factors, namely initial delay and play time instant [16]–[18]. Note that this initial delay is specific to the context of bitrate definition; it is different from the initial buffering delay which is used to cope with the fluctuations of connection throughput.

Fig. 2 shows a playout curve (piece-wise curve) of a video stream, which represents the accumulative played data size with respect to time. This playout curve consists of four intervals $\{(t_i, t_{i+1}) | 0 \leq i \leq 3\}$ corresponding to four segments of the video. The slope of the curve in each interval is the average bitrate of the corresponding media segment. Suppose that, at time t_d, the client starts receiving video data at a rate B; then the client starts playing the video data at time t_0. The value $d_0 = t_0 - t_d$ is called initial delay, which is the duration the client must wait before consuming the data. Given an initial delay d_0, *the bitrate of the whole video stream* is the minimum slope B of a tangent line that starts from point $(t_d, 0)$ and is never lower than the playout curve at any time instant.

Obviously, the larger the initial delay is, the lower the bitrate B becomes. However, this pair of bitrate-delay is just valid when the video is played continuously from the beginning to the end. If the user wants to play from t_2 (through random access/seeking), the initial delay must be d_2 to maintain the same value of bitrate B. If initial delay d_0 ($d_0 \leq d_2$) should be maintained, the defined bitrate at this point must be higher than B. So, the value of bitrate is highly dependent on the targeted play time instant (or random access point).

In the standardization process of DASH, the relationship of different initial delays and corresponding bitrates has been discussed in [18], and the notion of instant bitrate (or segment bitrate) was first presented in [12]. After several rounds of revision, the bitrate is finally derived by using the same initial delay for any random access point [14]. That means, it is *the highest value* of instant bitrates given a fixed initial delay. In DASH syntax, the initial delay and bitrate are represented by attributes *@minBufferTime* and *@bandwidth* respectively.

In IETF Internet-Draft "HTTP Live Streaming", the definition of bitrate also specifies that video bitrate is the highest segment bitrate of all segments of an alternative [19]. Because initial delay is not specified in the bitrate definition of [19], a segment bitrate can be understood as the average segment bitrate with

initial delay equal to 0. With this definition, the slope of segment (t_2, t_3), which is higher than the slopes of other segments, is the very representative bitrate of the above stream. For simplicity, in the following, segment bitrate means the average bitrate of a segment.

So, in existing HTTP streaming standards, the highest segment bitrate of an alternative is used as the representative bitrate of the alternative. This is obviously due to safety reason in adaptive streaming. However, for VBR video, this definition may result in a very poor bandwidth usage as shown later in Sections IV and V.

C. Related Work

Some recent work provides good reviews of HTTP streaming which mostly aims at delivering multimedia via the Web [1], [2]. As for DASH, overviews and basic ideas behind the development of the standard are highlighted in [2], [13]. A detailed analysis on the use of DASH for live service is presented in [9], where an initial buffering of about 2 segment durations is suggested. Detailed investigations of adaptivity in some commercial clients are carried out in [15], [20], providing some insights into the behaviors of the clients.

In [10], the measure of segment fetch time is used to determine requested video bitrates in an aggressive decrease and step-wise increase manner (like TCP congestion control [4]). Instead of using a TCP-like mechanism, a reliable estimation of throughput for city commuters using the prior-knowledge of commuting routes (e.g., metro/bus tracks) is used to determine video bitrate [21]. In [11], a Java client for HTTP streaming on Android platform is developed and different algorithms using different throughput estimation ways are compared. In [3], we presented a novel approach for throughput estimation, which is stable to short-term fluctuations while responding quickly to large fluctuations of the networks. Also, it is experimentally shown that an initial buffering delay of two segment durations could be achieved [3]. In [22], [23], the issues of stability and fairness when there are multiple clients or cross-traffic are investigated. To improve the stability, a special point of [23] is the use of a randomized scheduler for requesting media segments.

Bitrate estimation has long been an interesting research topic [24]–[27]. For network traffic control, if the output video bitrate from a live source is estimated to be higher than connection throughput, some bitrate scaling operation should be applied to avoid potential congestion [28]. Besides, a correct estimate of video bitrate would help providers to efficiently manage network resources [17]. Yet, there have been no studies on the use of bitrate estimation for HTTP streaming. To the best of our knowledge, [12] is the first work that discusses the importance of instant bitrate in HTTP streaming.

III. THROUGHPUT ESTIMATION

As mentioned, media segments are delivered by a sequence of HTTP request-response transactions. In this section, we just focus on the sequence of received segments, without considering their alternative indexes. The throughput in general is calculated by dividing the amount of data (data size) by the delivery interval. In fact, the difference between various throughput metrics
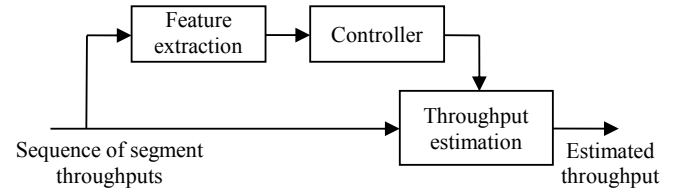

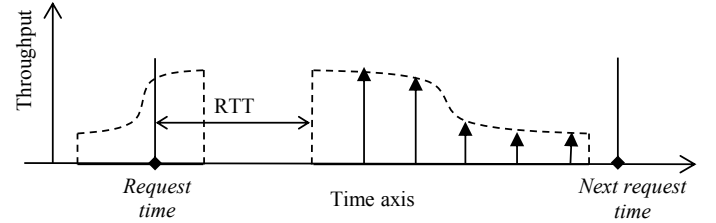Fig. 3. Proposed framework of throughput estimation.

Fig. 4. Illustration of download throughput samples.

is due to the very choice of delivery interval.

The general framework of our method is shown in Fig. 3. Here, feature extraction block provides one or more throughput related parameters of the previous segments. Based on the features, the controller block will decide to adjust the computation model in the throughput estimation block.

Using this framework, in [3], we presented a throughput estimation method that is based on previous segment throughputs. The segment throughput is computed by dividing the segment data size by the request-response duration, which is from the instant of sending the request to the instant of receiving the last byte of the response.

As discussed in [3], [11], the aggressive method, where the last segment throughput is simply used as the estimated throughput, currently is the most responsive method to capture the dynamic changes of throughput. However, when the segment duration is long (e.g., 8–10 s as in [19]), that method may not be effective to track the fast fluctuations of the connection.

To cope with this problem, we present a new and general formulation based on "download throughput" samples and RTT. Download throughput is computed by dividing the segment data size by the download duration, which is from the instant of receiving the first byte of the response to the instant of receiving the last byte of the response. A download throughput sample is an instant download throughput, computed over a short interval (e.g., 1 s) during downloading a segment. Fig. 4 illustrates download throughput samples (upward arrows) and the related concepts.

Before delving into the details, let's have the following notations:

- $D_p(i)$: The presentation duration of segment i
- $D^e(i)$: The expected download duration for segment i
- $T_d^e(i)$: The estimated download throughput for segment i
- $RTT(i)$: The duration between the instant of sending the request for segment i and the instant of receiving the first byte of the corresponding response.
- $T_d(i, j)$: The jth sample of download throughput of segment i.

- **TD**(i): The sequence of (download) throughput samples of segment i, i.e., **TD**(i) = $\{T_d(i, 0), T_d(i, 1), T_d(i, 2), \cdots\}$

If we can estimate $D^e(i)$ and $T_d^e(i)$, the expected amount of data delivered for segment i is $D^e(i) \cdot T_d^e(i)$. So, the estimated segment throughput for segment i is

$$Th_e(i) = \frac{D^e(i) \cdot T_d^e(i)}{D_p(i)}. \tag{1}$$

Denote $\{T_s(0), T_s(1), T_s(2), \cdots\}$ the sequence of throughput samples which is the combined sequence of $\{$**TD**$(0),$ **TD**$(1),$ **TD**$(2), \cdots\}$. In general, $T_d^e(i)$ is a function of throughput samples:

$$T_d^e(i) = f[T_s(0), T_s(1), T_s(2), \cdots]. \tag{2}$$

To compute $T_d^e(i)$, we apply the concept of running average T_s^{av} of $\{T_s(k)|k = 0, 1, 2, 3, \cdots\}$ as follows [3], [15]

$$T_s^{av}(k) = \begin{cases} (1 - \delta)T_s^{av}(k-1) + \delta T_s(k), & k > 1 \\ T_s(k-1), & k = 1 \end{cases} \tag{3}$$

where δ is a weighting value.

Suppose that $T_s(K_{i-1})$ is the last throughput sample of segment $(i-1)$, the average throughput $T_s^{av}(K_{i-1})$ then is used as the estimated throughput for segment i:

$$T_d^e(i) = T_s^{av}(K_{i-1}). \tag{4}$$

Usually, the higher the value of δ is, the more the estimated throughput depends on the recent throughput samples. To have a smoothed value of throughput, a small value of δ should be used. In this paper, we adopt the method in [3] to control the value of δ, so that the estimated value is stable to short-term fluctuations while responding quickly to large fluctuations of the networks.

As for $D^e(i)$, this value is strongly dependent on the request time instant. For that, we classify the problem into two cases, non-pipelined and pipelined.

A connection is non-pipelined if a request is sent only after the previous segment has been fully received. After a request is sent, it takes about an RTT delay for the client to receive the first byte of the response. So, the expected download duration is as follows:

$$D^e(i) = D_p(i) - RTT^e(i) \tag{5}$$

where $RTT^e(i)$ is the expected RTT of segment i. $RTT^e(i)$ also is computed as a running average of previous RTT values:

$$RTT^e(i) = \begin{cases} (1 - \gamma)RTT^e(i-2) + \gamma RTT(i-1), & i > 1 \\ RTT(i-1), & i = 1 \end{cases} \tag{6}$$

where γ is a weighting value which is similar to δ of (3). Usual γ takes a small value for a smooth RTT estimation. In this paper, γ is set to 0.125 as recommended by [29].

The connection is of pipelined type if a request is sent before having fully received the previous segment. In this case, the expected download duration is as follows:

$$D^e(i) = D_p(i). \tag{7}$$

The highest possible value of segment bitrate $B^c(i)$ is computed from the estimated throughput using a simple safety margin μ as follows:

$$B^c(i) = (1 - \mu)Th_e(i) \tag{8}$$

where μ usually takes a small value in the range [0, 0.5].

From (5) and (7), we can see that the use of pipelining can help increase the download time (about one RTT), leading to the increase in segment throughput. However, when segment duration is long (e.g., 6–10 s), the improvement by using pipelining is insignificant because RTT in today's networks is just several tens or hundreds of milliseconds. Moreover, pipelining leads to a lot of complexity in handling and detecting timing of responses. So, in this study, we just focus on non-pipelined case.

IV. VIDEO BITRATE ESTIMATION

The previous section provides the estimated throughput which is used to decide the bitrate of the next segment. If video is created with constant bitrate, the selection of appropriate alternative would be straightforward. However, when the bitrate is not constant, there may be two cases. First, the instant video bitrate is much lower than the actual throughput, thus resulting in poor bandwidth usage. Second, the instant video bitrate is higher than the actual throughput, resulting in transmission delay and then buffer underflows.

Our objective in this section is that, through bitrate estimation, the client will be able to dynamically select the highest possible bitrate in both cases. Hereafter, the notation $S(i, n)$ means the segment of time (or segment) index i and alternative n; and $B(i, n)$ means the bitrate of segment $S(i, n)$.

We suppose that each video alternative now is encoded by a quantization parameter (QP) value [30]. At time or segment index $i - 1$, suppose that the client has already received a segment $S(i - 1, a)$ and computed the actual bitrate $B(i - 1, a)$ of that segment. Then, before downloading a segment of time index i, the client should estimate bitrates $\{B(i, n)|1 \leq n \leq N\}$, where N is the number of alternatives. For that, the client should estimate first the bitrates $\{B(i - 1, n)|1 \leq n \leq N$ and $n \neq a\}$.

So, in our approach, bitrate estimation is divided into two parts: 1) Inter-stream estimation and 2) intra-stream estimation. The former means estimating the bitrates of segments (in the same interval) across different alternatives, while the latter implies estimating the bitrate of a future segment within an alternative. The estimation models here are specific for advanced video coding (AVC) [31], which is the most popular video format and is used in our streaming system.

A. Inter-Stream Bitrate Estimation

A number of models to represent the relationship between video bitrate and QP value have been proposed in the literature (e.g., [26], [27]). In general, the bitrate of a segment is well related to the bitrate of another segment with the same time index. If the client has received a segment $S(i - 1, a)$ with bitrate $B(i - 1, a)$, we can model the estimated bitrate $B^e(i - 1, n)$ of any segment $S(i - 1, n)$ as a function of $B(i - 1, a)$:

$$B^e(i - 1, n) = f_{inter}[B(i - 1, a)]. \tag{9}$$

In AVC, it is well-known that a 6-unit increase of QP would roughly halve encoded video bitrate [28], [31]. More specifically, a 1-unit increase of QP means an increase of quantization step size by approximately 12%, which results in a bitrate reduction of about 12% [31]. So, the bitrate of $S(i - 1, n)$ can be
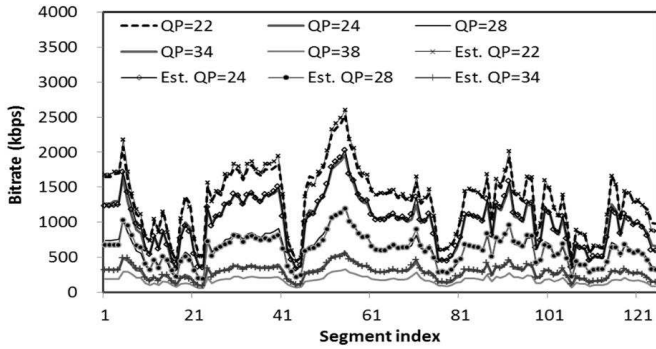
Fig. 5. Estimated bitrates across different alternatives.

estimated from the bitrate of the received segment $S(i-1,a)$ as follows.

$$B^{e}(i-1,n) = \theta B(i-1,a)2^{\frac{Q_a - Q_n}{6}} \qquad (10)$$

where Q_a and Q_n are the QP values of the alternatives and θ is an empirical factor that compensate for the approximation error of the model. It should be noted that, QP chiefly affects the amount of bits used to encode video residual data. In a coded video stream, a nontrivial amount of bits is used to convey other data such as parameter sets and slice headers [31]. Based on our experience, a reasonable value of θ is 1.05.

Fig. 5 shows the estimated bitrate together with the actual bitrate of a video clip which is taken from the "Tokyo Olympics" sequence [30]. The estimated bitrate for a given QP is computed by (10) with $n = a - 1$. We can see that the estimation model provides very good results across different bitrate ranges. In practice, the client can obtain QP value of each alternative by simply downloading and then parsing the initial part of each alternative, where the parameter sets of video are located. In addition, the QP values can easily be put into MPD using the syntax extensibility [3].

B. Intra-Stream Bitrate Estimation

In general, the bitrate of segment $S(i,n)$ is a function of some previous segment bitrates, i.e.,

$$B^{e}(i,n) = f_{intra}[B(i-1,n), B(i-2,n), B(i-3,n), \cdots]. \qquad (11)$$

A large number of bitrate estimation methods have been proposed in this area (e.g., [24], [25]). However, most of them require much knowledge of the content (e.g., shot changes) and/or high processing power, which are not appropriate for fast estimation in Web-based or lightweight streaming clients.

In practice, a video sequence is usually composed of scenes where GOPs in each scene would have similar characteristics. That means, adjacent video segments would have similar bitrates. So, for intra-stream bitrate estimation, the previous known instant bitrate of time index $i - 1$ could be used as the estimated bitrate $B^{e}(i,n)$ of the segment i, i.e.,

$$B^{e}(i,n) = \begin{cases} B^{e}(i-1,n), & n \neq a \\ B(i-1,a), & n = a. \end{cases} \qquad (12)$$

Due to its simplicity, which causes nearly no computation overhead, this simple model is used in our system. However, as
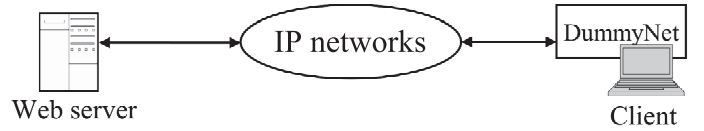

Fig. 6. Test-bed organization for the experiments.

shown in the experiments, the results provided by this solution are already very effective.

V. EXPERIMENTS

The organization of our test-bed is shown in Fig. 6. The server is an Apache of version 2.2.21 run on Ubuntu 11.10 (with default TCP CUBIC). For alive connections, the server's timeout is set to 100 s and MaxRequest to 0 (i.e., unlimited). DummyNet [32] is installed at the client to simulate network characteristics. As random fluctuations of throughput would make it difficult to compare the results of different experiment runs, the loss rate of DummyNet is set to 0% for clear results. Here, we assume that actual bandwidth trace used in the experiments already contains the fluctuations caused by packet loss. RTT value of DummyNet is set to 40 ms. The safety margin μ for deciding video bitrate is set to be 0.05.

In our test-bed, the client is implemented in Java and runs on a Windows 7 Professional notebook with 2.0 GHz Core2Duo CPU and 2 G RAM. Video is encoded by the main profile of AVC [31]. Video component has the frame rate of 30 fps and resolution of 320×240. Media segments are created with the same duration and stored in separate small files. Distance between two consecutive segment requests normally is the segment duration; however, if the previous response is delayed, the following requests will be sent right after fully receiving a response until the buffer level is stable. The initial buffering delay (and also the target buffer level) is equal to two segment durations. In the following, we present two experiments, the first is for throughput estimation, and the second is for bitrate estimation.

A. Experiment of Throughput Estimation

In the first experiment, video is encoded in CBR mode with 10 bitrates, from 2560 kbps to 256 kbps with a step size of 256 kbps. Download throughput samples are taken with a period of 1 s. To show the effectiveness of our method proposed in Section III, the aggressive method ([3], [11]) is selected to be the reference method. As mentioned before, the aggressive method is simply based on the last segment throughput and currently the most responsive to the fast fluctuations of bandwidth. A real bandwidth trace obtained from a mobile network [33] is employed in the experiment.

Figs. 7(a) and (b) show the client behavior for two cases of throughput estimation: 1) Using the aggressive method and 2) using our proposed method. Each figure has four curves, representing the bandwidth (controlled by DummyNet), estimated throughput, selected bitrate, and the resulting buffer level. In these figures, the segment duration is 6 s and the target buffer level is 12 s (2×6 s). It can be seen that, with the aggressive method, the buffer level may be reduced by about 11.5

Fig. 7.   Client behavior with segment duration of 6 s and throughput estimation using: (a) Aggressive method and (b) proposed method.
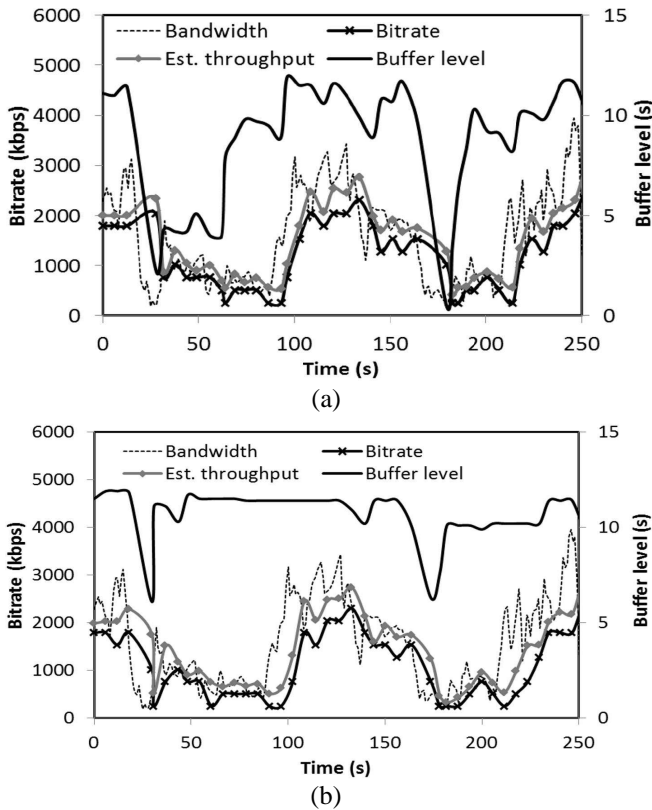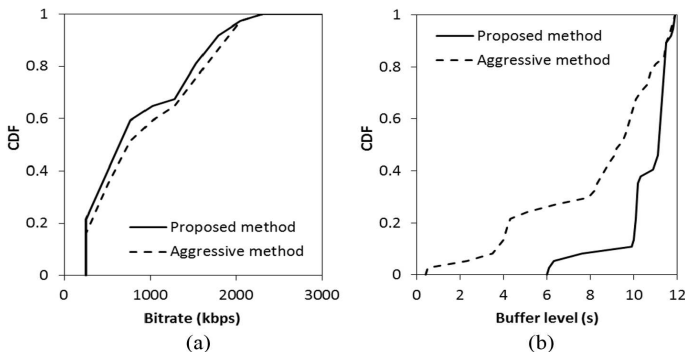

Fig. 9.   Client behavior with segment duration of 8 s and throughput estimation using: (a) Aggressive method and (b) proposed method.

Fig. 8.   Comparison of the aggressive method and proposed method with segment duration of 6 s using: (a) CDF of bitrate and (b) CDF of buffer level.

When segment duration is 8 s, the similar behaviors are shown in Figs. 9(a) and (b). Now the target buffer level is 16 s (2 × 8 s). We can see that, with our proposed method, the variations of buffer level are still within 5 s. Meanwhile, with aggressive method, the reduction of buffer level is up to 14 s. The corresponding CDFs of bitrate and buffer level (Fig. 10) also show the results similar to those of Fig. 8. Especially, in terms of buffer level, the improvement of the proposed method compared to the aggressive method is up to 9 s in Fig. 10(b) and 5.5 s in Fig. 8(b). This implies that when segment duration is longer, the responsiveness of the aggressive method would be worse. This can be explained by the fact that the aggressive method uses a throughput measure averaged over the whole segment duration. Meanwhile the use of throughput samples enables the client to track the fast fluctuations regardless of segment duration.

B. Experiment of Bitrate Estimation

In the second experiment, we prepare the video alternatives in VBR mode. Test video, which is taken from the "Tokyo Olympics" sequence [30], consists of 125 segments. The duration of each segment is set to 2 s to clearly represent the variations of instant video bitrate. Similar to [30], video alternatives are encoded with 7 different values of QPs, namely 22, 24, 28, 34, 38, 42, and 48. These alternatives are respectively denoted by #7, #6,···, #1. Fig. 11 shows the segment bitrates of each alternative. We can see that the bitrate varies widely. The average bitrate and the highest bitrate of each alternative are listed in Table 1.

s. Meanwhile, with our proposed method, the buffer is reduced by only 6 s. The comparison using cumulative distribution functions (CDFs) of bitrate and buffer level (Fig. 8) provides more insights into the client's overall behavior. The CDFs of bitrate show that the bitrates of the two methods are very similar. Meanwhile, the CDFs of buffer level show that the buffer level of the aggressive method varies widely from 12 s to 0.5 s. Especially, the buffer level of the proposed method stays mostly in the range 10 s–12 s and sometimes goes down to 6 s. That means, the accuracy of our throughput estimation method can enable a more stable buffer level, thus helping reduce the initial buffering delay.
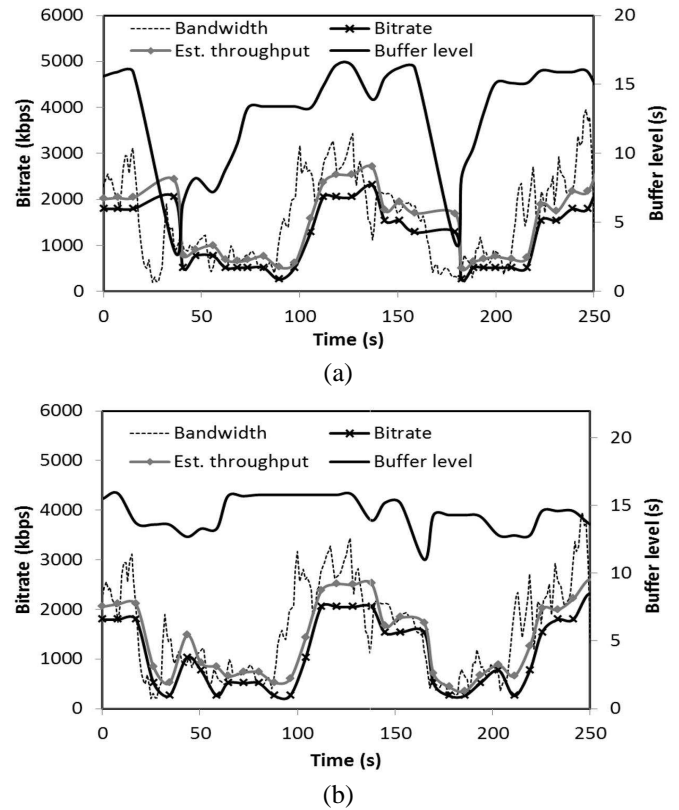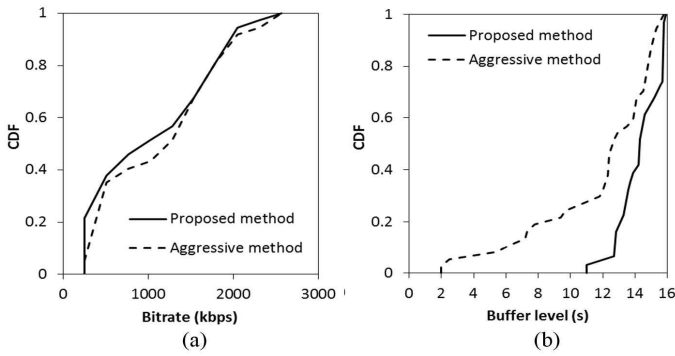
Fig. 10. Comparison of the aggressive method and the proposed method with segment duration of 8 s using: (a) CDF of bitrate and (b) CDF of buffer level.

Table 1. The average bitrate and highest bitrate of alternatives used in the second experiment.

Alternative	Average bitrate (kbps)	Highest bitrate (kbps)
#7	1320.2	2526.3
#6	1015.1	1966.7
#5	616.7	1215.7
#4	288.8	567.8
#3	173.4	329.3
#2	104.2	189.4
#1	45.4	92.7

Fig. 11. Segment bitrate of diffirent alternatives of test video.

In order to clearly explain the client behavior, we first run the experiment with flat bandwidth. Initially, suppose that the average bitrate of each alternative is used as its representative bitrate. If the bandwidth is 1150 kbps (resulting in a throughput of about 1090 kbps), alternative #6 (with average bitrate of 1015.1 kbps) will be selected. However, due to the variations of segment bitrate, playing this alternative makes the buffer level gradually reduced from 8 s to 2 s as shown in Fig. 12. In this case, the initial buffering must be more than 6 s (i.e., 8 s – 2 s) to support a continuous playout. Obviously, it is difficult to determine in advance the necessary amount of initial buffering when average bitrate is used. This is the reason that average bitrate of a VBR video stream usually is not used as the stream's representative bitrate.

Now let's use the highest bitrate of each alternative as its representative bitrate, which is the choice of existing HTTP stream-
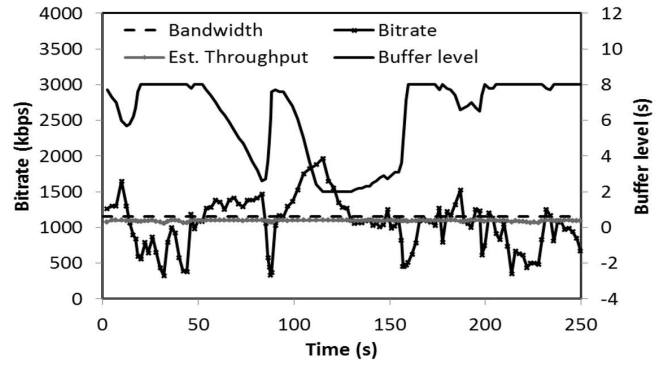

Fig. 12. Client behavior using the average segment bitrate.
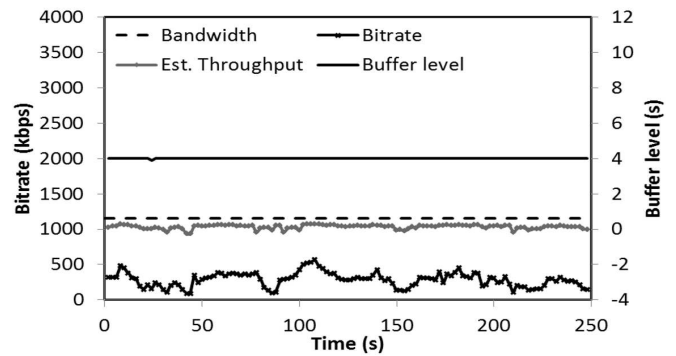

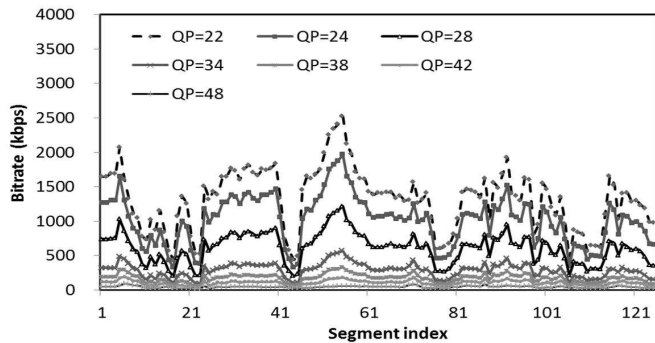Fig. 13. Client behavior when the representative bitrate of an alternative is its highest segment bitrate.

ing standards as discussed in Section II. The available bandwidth and throughput are kept as before, which are lower than the highest bitrate of alternative #5. So, with this case, the selected alternative will be #4 as shown in Fig. 13. The buffer level is obviously very stable; however, the connection bandwidth is significantly underused in this case. Especially, if the highest bitrate is much higher than the usual values of segment bitrates, the actual bandwidth usage will be very low.

If the client has the (instant) bitrate estimation capability as presented in Section IV, it can intelligently switch between alternatives #5, #6 and #7 depending on the estimated segment bitrates. As shown in Fig. 14, the client now can achieve much higher bitrate while the buffer is still stable. Note that, with this stable buffer level, the initial buffering could be reduced from two segment durations (4 s) to only one segment duration (2 s), which will significantly improve the quality of experience for users. Fig. 15 compares the CDFs of bitrate and buffer level in these three cases: 1) Using the average bitrate (ABR), 2) using the highest bitrate (HBR), and 3) using the estimated bitrate (EBR). This figure reconfirms that the case using EBR provides a good bitrate (close to the case using ABR) and stable buffer level (close to the case using HBR).

The bandwidth trace of the previous part is now employed to see how bitrate estimation helps the client in practical settings. Fig. 16 compares the bitrate and buffer level of two cases: 1) Using HBR and 2) using EBR. The case of ABR is not in-
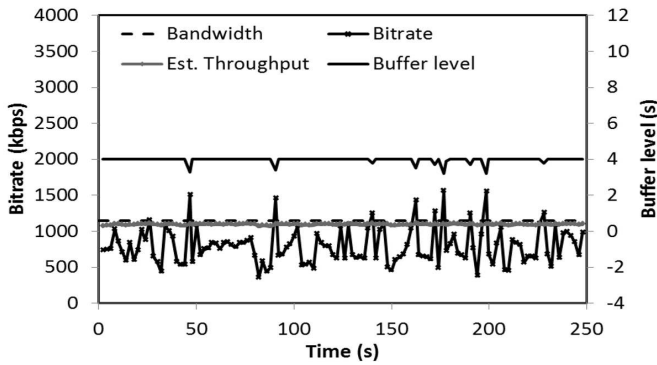
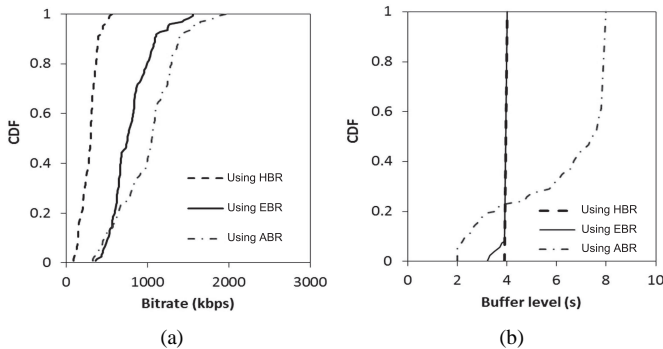Fig. 14. Client behavior when bitrate estimation is enabled.

Fig. 16. Comparison of the client behavior when the adaptation is based on the highest bitrate (HBR) or the estimated birate (EBR).

Fig. 15. When the adaptation is based on the average birate (ABR), the highest bitrate (HBR), or estimated birate (EBR): (a) CDF of bitrate and (b) CDF of buffer level.

Fig. 17. When the adaptation is based on the highest bitrate (HBR) or estimated birate (EBR): (a)CDF of bitrate and (b) CDF of buffer level.

cluded here because it is not usually used in practice. Also, the throughput curves are not shown for the sake of clarity. We can see that when the adaptation is based on EBR, the resulting bitrate curve is significantly higher than the case of HBR. The CDFs of bitrate in Fig. 17(a) show that the advantage of using EBR exists in the whole range of birate. Meanwhile, the buffer level of the case using EBR is stable and nearly the same as that of the case using HBR (Figs. 16 and 17(b)).

C. Discussion

The results in the first experiment show that our proposed method can capture quickly the changes of throughput, and then adjust video bitrate accordingly. In other tests with short segment duration (e.g., 2 s or 4 s), the improvement in buffer stability of our proposed method compared to the aggressive method is not significant (about 1 s). This is because the number of throughput samples in short segment case is not as many as that in long segment case.

In the second experiment, an interesting finding is that, even using the simple intra-stream bitrate estimation, which is using the last segment bitrate, we can significantly improve the quality. Sometimes the actual bitrate is higher than the throughput (as in Figs. 14 and 16); however, the errors are not big and the client still maintains a good buffer with very small variations. This suggests that it may be unnecessary to employ more complicated methods for intra-stream bitrate estimation.

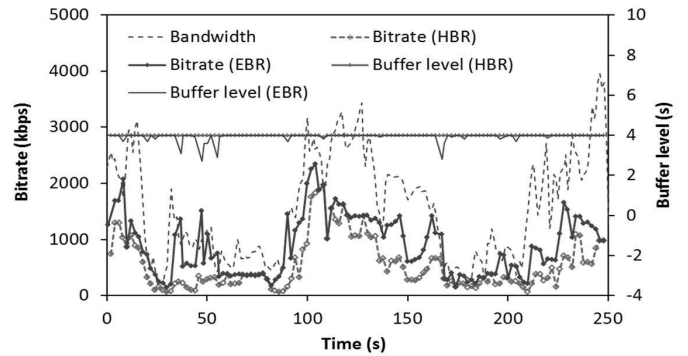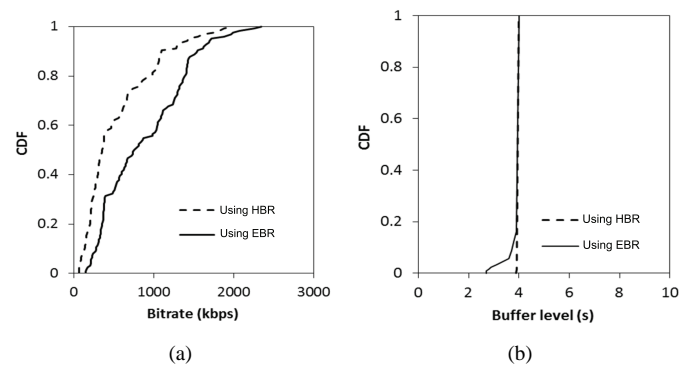It should be noted that, when a video is encoded in CBR mode, the visual quality would be changed in a similar way to switching alternatives. If the number of alternatives is higher, the switching will be smoother. So, in some sense, bitrate estimation facilitates a kind of *client-enabled CBR streaming, where the video sources are in fact encoded in VBR mode.* The advantage of this client-enabled CBR streaming is that when bandwidth is abundant, the client may turn off bitrate estimation and then run in VBR mode. Meanwhile, when bandwidth is low, the client will apply bitrate estimation and then achieve (near) CBR streaming with VBR video sources. Of course, even in case of low bandwidth, the client can still decide to use VBR mode if preferred.

The complexity of our throughput and bitrate estimation methods is very low because they are just based on analytical formulations. From our experience, the decision delay of our method, which is from the instant of receiving the last byte of a segment to the instant of obtaining the selected bitrate for the next segment, is very small (less than 1 ms). That means the estimation methods essentially do not affect the playback quality.

VI. CONCLUSION

In this paper, we have studied the estimation of connection throughput and video bitrate in adaptive HTTP streaming. A general formulation for throughput estimation was proposed, taking into account important factors such as instant download throughput and RTT. Initial mechanisms for bitrate estimation of VBR video were also presented. The experiment results

showed that the proposed solutions were effective to maintain a stable buffer under fluctuations of bandwidth and video bitrate. Our goal in the future is to improve the throughput estimation process so that the client can quickly recognize the different throughput patterns of different networks and then automatically adjust the estimation. Also, intra-stream bitrate estimation will be improved by using some simple content features such as motion activity.

ACKNOWLEDGMENT

REFERENCES

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

[2] A. C. Begen, T. Akgul, and M. Baugher, "Watching video over the web, part II: Applications, standardization, and open issues," *IEEE Internet Comput.*, vol. 15, no. 3, pp. 59–63, Apr. 2011.

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

[4] J.-W. Park, R. P. Karrer, and J. Kim, "TCP-ROME: A transport-layer parallel streaming protocol for real-time online multimedia environments," *J. Commun. Netw.*, vol. 13, no. 3, pp. 277–285, 2011.

[5] T. V. Lakshman, A. Ortega, and A. R. Reibman, "Variable bit rate (VBR) video: Tradeoffs and potentials," *Proc. IEEE*, vol. 86, no. 5, pp. 952–973, May 1998.

[6] T. C. Thang, J. Y. Lee, J. W. Kang, S. J. Bae, S. Jung, and S. T. Park, "Signaling metadata for adaptive HTTP streaming," ISO/IEC JTC1/SC29/WG11 m17771, Geneva, July 2010.

[7] D. Wu *et al.*, "Streaming video over the Internet: Approaches and directions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 282–300, 2001.

[8] F.-J. Liu and C.-S. Yang, "Proxy design for improving the efficiency of stored MPEG-4 FGS video eelivery over wireless networks," *J. Commun. Netw.*, vol. 6, no. 3, pp. 280–286, 2004.

[9] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *Proc. IEEE WoWMoM*, June 2011.

[10] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. ACM MMSys*, California, Feb. 2011.

[11] L. R. Romero, "A dynamic adaptive HTTP streaming video service for google android," M.S. Thesis, Royal Institute of Technology (KTH), Stockholm, Oct. 2011.

[12] T. C. Thang, J. Y. Lee, J. W. Kang, S. J. Bae, S. Jung, and S. T. Park, "Proposal on signaling for DASH", ISO/IEC JTC1/SC29/WG11 m18445, Guangzhou, Oct. 2010.

[13] T. Stockhammer, "Dynamic adaptive streaming over HTTP - standards and design principles," in *Proc. ACM MMSys*, California, Feb. 2011.

[14] ISO/IEC IS 23009-1: "Information technology - dynamic adaptive streaming over HTTP (DASH) - part 1: Media presentation description and segment formats," 2012.

[15] S. Akhshabi, A.C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. ACM MMSys*, California, Feb. 2011.

[16] J. R. Corbera, P. A. Chou, and S. L. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 674–687, July 2003.

[17] T. C. Thang and Y. M. Ro, "Practical estimation techniques of traffic specification for VBR video services," in *Proc. SPIE Electronic Imaging*, vol. 5022, Jan. 2003, pp. 323–333.

[18] T. C. Thang, J. Y. Lee, J. W. Kang, S. J. Bae, S. Jung and S. T. Park, "Bandwidth information for DASH," ISO/IEC JTC1/SC29/WG11 m19324, Daegu, Jan. 2011.

[19] R. Pantos, "HTTP live streaming," Internet Draft draft-pantos-http-live-streaming-07, 2011.

[20] L. De Cicco and S. Mascolo, "An experimental investigation of the Akamai adaptive video streaming," in *Proc. USAB*, Klagenfurt, Austria, Nov. 2010, pp. 447–464.

[21] K. Evensen *et al.*, "Mobile video streaming using location-based network prediction and transparent handover," in *Proc. NOSSDAV*, Jun. 2011, pp. 21–26.

[22] T.Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. ACM Internet Measurement Conference (IMC)*, Boston, Nov. 2012.

[23] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with FESTIVE," in *Proc. ACM CoNEXT*, Nice, Dec. 2012.

[24] A. Abdennour, "Short-term MPEG-4 video traffic prediction using ANFIS," *Int. J. Netw. Management*, vol. 15, pp. 377–392, 2005.

[25] A. Bhattacharya, A.G. Parlos, and A.F. Atiya, "Prediction of MPEG-coded video source traffic using recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2177–2190, Aug. 2003.

[26] S. Azad, W. Song, and D. Tjondronegoro, "Bitrate modelling of scalable videos using quantization parameter, frame rate, and spaial resolution," in *Proc. ICASSP*, 2010, pp. 2334–2337.

[27] Y. Wang, Z. Ma, and Y.-F. Ou, "Modeling rate and perceptual quality of scalable videos as functions of quantization and frame rate and its application in scalable video adaptation," in *Proc. Packet Video Workshop*, 2009.

[28] J. Ascenso, M. Jakubowski, and G. Pastuszak, "Constant bitrate control for a distributed video coding system," in *Proc. SIGMAP*, Porto, July 2008.

[29] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," RFC 6298, June 2011. [Online]. Available: http://tools.ietf.org/html/rfc6298

[30] G. V. Auwera, P. T. David, and M. Reisslein, "Traffic and quality characterization of single-layer video streams encoded with H.264/MPEG-4 advanced video coding standard and scalable video coding extension," *IEEE Trans. Broadcasting*, Vol. 54, no. 3, pp. 698–718, Sept. 2008.

[31] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.

[32] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," *ACM Computer Commun. Rev.*, vol. 27, no. 1, pp. 31–41, Jan. 1997.

[33] C. Muller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proc. ACM MoVid*, North Carolina, Feb. 2012.

Truong Cong Thang received the B.E. degree from Hanoi University of Technology, Vietnam, in 1997 and Ph.D. degree from KAIST, Korea, in 2006. From 1997 to 2000, he worked as an engineer in Vietnam Post & Telecom (VNPT). From 2006 to 2011, he was a Member of Research Staff at Electronics and Telecommunications Research Institute (ETRI), Korea. He was also an active member of Korean delegation to standard meetings of ISO/IEC and ITU-T from 2002 to 2011. Since 2011, he has been an Associate Professor of University of Aizu, Japan. His research interests include multimedia networking, image/video processing, content adaptation, IPTV, and MPEG/ITU standards.

Hung T. Le received the B.E. degree from Hanoi University of Technology, Vietnam, in 2012. Since 2012, he has been a research assistant at the Computer Communications Laboratory, the University of Aizu. His major research interests are multimedia networking, video adaptation, QoE support. He is a Student Member of IEEE.

Hoc X. Nguyen received the B.S. of Computer Science from The University of Aizu, Fukushima, Japan in 2013. He is currently a software engineer at FPT software JSC, Vietnam. His research interests include adaptive video streaming and subjective quality evaluation methods.

Anh T. Pham received the B.E. and M.E. degrees, both in Electronics Engineering from the Hanoi University of Technology, Vietnam in 1997 and 2000, respectively, and the Ph.D. degree in Information and Mathematical Sciences from Saitama University, Japan in 2005. From 1998 to 2002, he was with the NTT Corp. in Vietnam. Since April 2005, he has been on the faculty at the University of Aizu, where he is currently an associate professor at the Computer Communications Laboratory, the School of Computer Science & Engineering. His present research interests are in the area of computer networking, optical communications, and spread spectrum technique. He received Japanese government scholarship (MonbuKagakusho) for Ph.D. study. He also received Vietnamese government scholarship for undergraduate study. Dr. Pham is senior member of IEEE. He is also member of IEICE and OSA.

Jung Won Kang received the B.S. and M.S. degrees in Electrical Engineering in 1993 and 1995, respectively, from the Hankuk Aviation University, Seoul, South Korea. She received the Ph.D. degree in Electrical and Computer Engineering in 2003 from Georgia Institute of Technology, Atlanta, GA. Since 2003, she has been a Senior Member of Research Staff in Broadcasting & Telecommunications Convergence Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Korea. Her research interests are in the area of video signal processing, video coding, and video adaptation.

Yong Man Ro received the B.S. from Dept. of Electronics Engineering in Yonsei University (1985), Seoul and the M.S. and Ph.D. degrees from Dept. of Electrical Engineering in the KAIST (1987, 1992, respectively). He was a researcher at Columbia University and a visiting researcher in University of California at Irvine and KAIST. In 1996, he was a research fellow at Dept. of Electrical Engineering and Computer Sciences in University of California at Berkeley. In 2007, he was a visiting professor at Dept. of Electrical and Computer Engineering at the University of Toronto. He is currently a full professor of Department of Electrical Engineering in KAIST. He is also director of IVY (Image/Video sYstem) Lab. He was an active participant in MPEG standardization efforts since 1997. He has presented 78 contributions, and more than 10 technologies developed by his research lab were selected for inclusion in the MPEG-4, MPEG-7, MPEG-21, and MPEG-A standards. He is the author or co-author of more than 300 international research papers, 67 international standardization documents, more than 50 patents. His research interests include image/video processing, color face recognition, image/video indexing, and multimedia security, medical imaging, pattern recognition, and functional imaging. Dr. Yong Man Ro is a senior member of IEEE as well as member of ISMRM and SPIE. He received the Young Investigator Finalist Award of ISMRM in 1992 and the year's scientist Award (Korea) in 2003. He served as an associate editor for IEEE signal processing letters (2010–2013.3). He serves as associate editor in Transactions on Data Hiding and Multimedia Security (Springer-Verlag). He organized and served TPC in many international conferences including the program chair of IWDW 2004, he also organized special sessions on "Digital Photo Album Technology" in AIR 2005, "Social Media" in DSP 2009 and "Human 3D Perception and 3D Video Assessments" in DSP2011.