Adaptive Video Streaming over HTTP with Dynamic Resource Estimation

Truong Cong Thang[1], Hung T. Le[1], Hoc X. Nguyen[1], Anh T. Pham[1], Jung Won Kang[2], Yong Man Ro[3]
[1]University of Aizu, Japan
[2]Electronics and Telecommunications Research Institute, Korea
[3]Korea Advanced Institute of Science and Technology, Korea

Abstract: **Adaptive 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: **HTTP streaming, adaptivity, throughput estimation, bitrate estimation.**

I. INTRODUCTION

Thanks to the abundance of Web platforms and broadband connections, HTTP streaming has become a cost effective means for multimedia delivery [1][2][3]. Besides, due to the heterogeneity of today's communication networks, adaptivity is the most important requirement for any streaming client [3]. Especially, TCP, the underlying layer of HTTP, is notorious for its throughput fluctuations [4]. Moreover, the bitrate of a video encoded in VBR (variable bitrate) 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 decision 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 is an average value 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 mentioned 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 [13][3], 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 decision on which/when media parts are downloaded.

Recently, a new standard called Dynamic Adaptive Streaming over HTTP (DASH) has been developed by ISO/IEC 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].

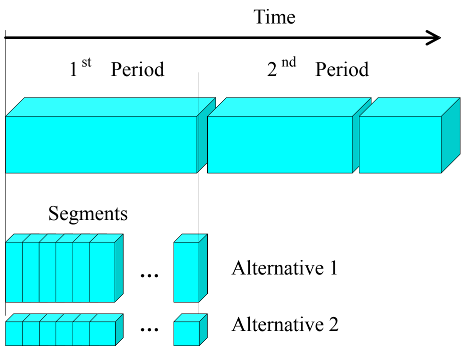


Fig. 1. Hierarchy of content division in MPEG DASH

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) 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][17][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).

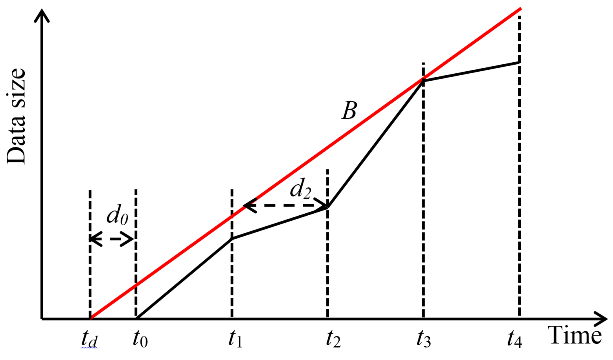


Fig. 2. Illustration of bitrate B with initial delay d_0

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 [13][2]. 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][25][26][27]. For network traffic control, if 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 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.

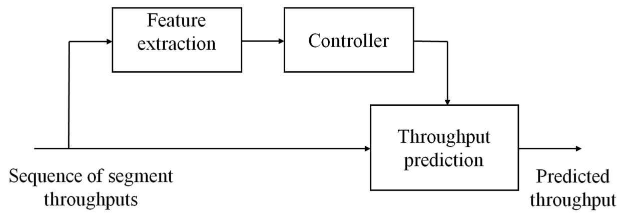


Fig. 3. Proposed framework of throughput estimation

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. 8-10s 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. 1s) during downloading a segment. Fig. 4 illustrates download throughput samples (upward arrows) and the related concepts.

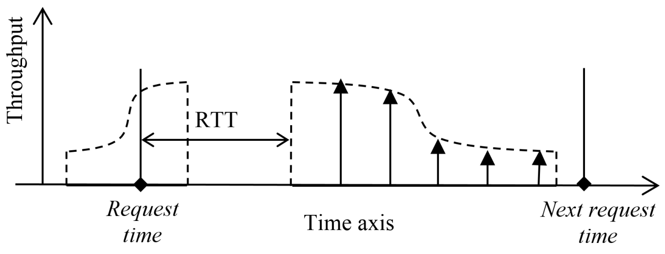


Fig. 4. Illustration of download throughput samples

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 j^{th} sample of download throughput of segment i.
– **TD**(i): the sequence of (download) throughput samples of segment i, i.e. **TD**(i) = $\{T_d(i,0), T_d(i,1), T_d(i,2), ...\}$

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

$$Th_e(i) = \frac{D^{\mathrm{e}}(i) * T_d^{\mathrm{e}}(i)}{D_p(i)}. \quad (1)$$

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

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

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

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

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:

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

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 $D^{\mathrm{e}}(i)$, 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:

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

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

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

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:

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

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

$$B^{\mathrm{c}}(i) = (1-\mu)Th_{\mathrm{e}}(i) \quad (8)$$

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. 6-10s), 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 underflow.

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 AVC (Advanced Video Coding) [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 $B^{\mathrm{e}}(i-1,n)$ of any segment $S(i-1,n)$ as a function of $B(i-1,a)$:

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

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

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

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 Eq. (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].

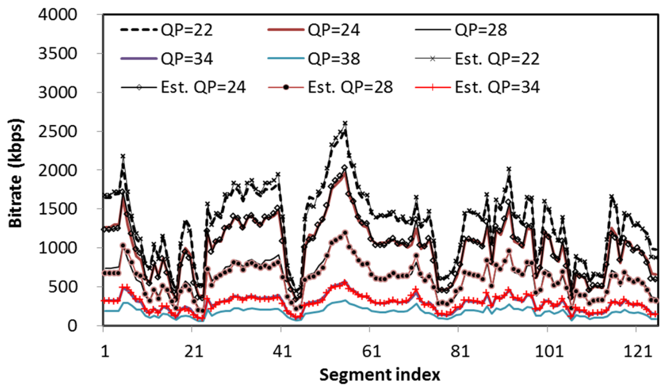


Fig. 5. Estimated bitrates across different alternatives

B. *Intra-stream bitrate estimation*

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

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

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 client.

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.

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

Due to its simplicity, which causes nearly no computation overhead, this simple model is used in our system. However, as shown in the experiments, the results provided by this solution are already very effective.

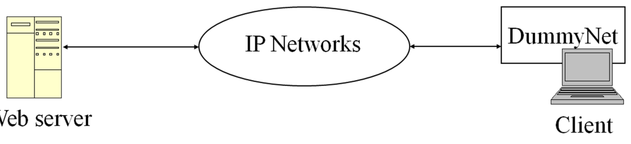


Fig. 6. Test-bed organization for the experiments

V. EXPERIMENTS

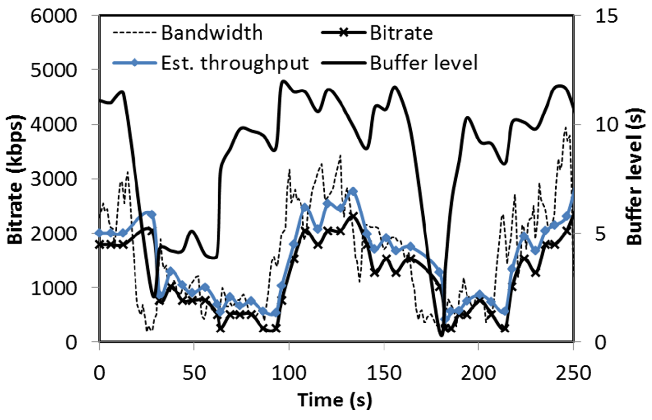
The organization of our test-bed is shown in Fig. 6. The server is an Apache of version 2.2.21 run on Ubuntu 11.10 (with default TCP CUBIC). For alive connections, the server's timeout is set to 100s and MaxRequest to 0 (i.e. unlimited). DummyNet [32] is installed at the client to simulate network characteristics. 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 40ms. 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 2.0GHz Core2Duo CPU and 2G RAM. Video is encoded by the main profile of AVC (Advanced Video Coding) [31]. Video component has the frame rate of 30fps and resolution of 320x240. 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.

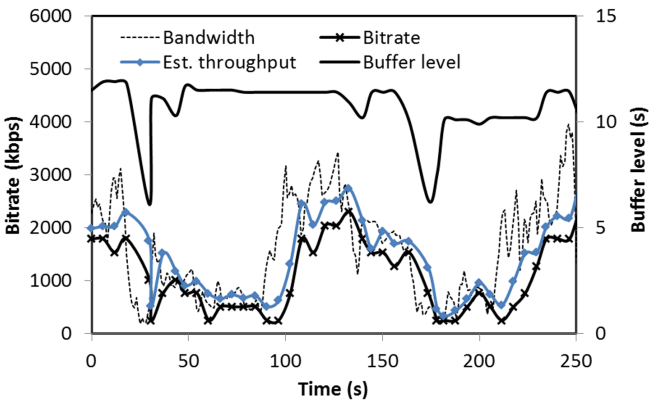
A. *Experiment of throughput estimation*

In the first experiment, video is encoded in CBR mode with 10 bitrates, from 2560kbps to 256kbps with a step size of 256kbps. Download throughput samples are taken with a period of 1s. 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.

Fig. 7a and Fig. 7b 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 6s and the target buffer level is 12s (2x6s). It can be seen that, with the aggressive method, the buffer level may be reduced by about 11.5s. Meanwhile, with our proposed method, the buffer is reduced by only 6s. 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 12s to 0.5s. Especially, the buffer level of the proposed method stays mostly in the range 10s-12s and sometimes goes down to 6s. That means, the accuracy of our throughput estimation method can enable a more stable buffer level, thus helping reduce the initial buffering delay.

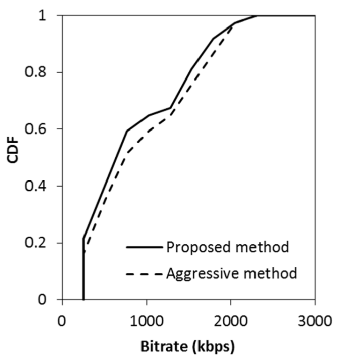


(a)

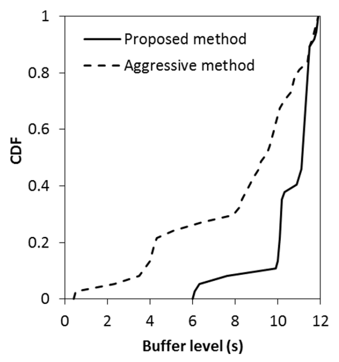


(b)

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



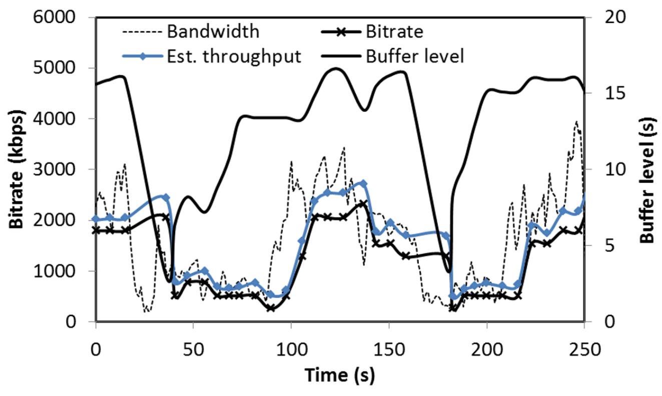
(a)



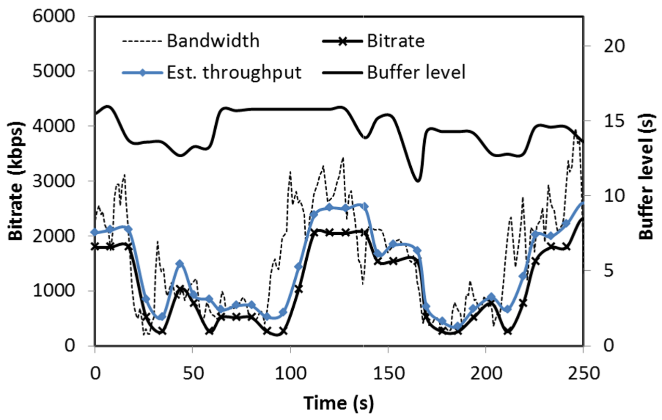
(b)

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

When segment duration is 8s, the similar behaviors are shown in Fig. 9a and Fig. 9b. Now the target buffer level is 16s (2x8s). We can see that, with our proposed method, the variations of buffer level are still within 5s. Meanwhile, with aggressive method, the reduction of buffer level is up to 14s. 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 9s in Fig. 10b and 5.5s in Fig. 8b. 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.



(a)

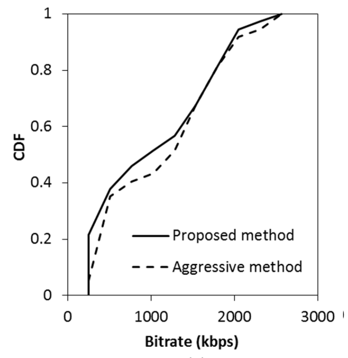


(b)

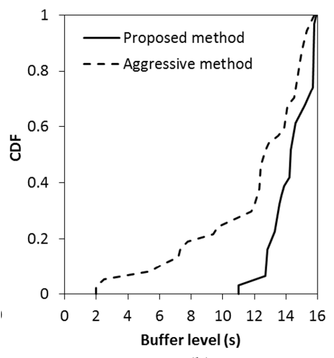
Fig. 9. Client behavior with segment duration of 8s and throughput estimation using (a) aggressive method and (b) proposed method

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 2s 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



(a)



(b)

Fig. 10. Comparison of the aggressive method and the proposed method with segment duration of 8s 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

in Table 1.

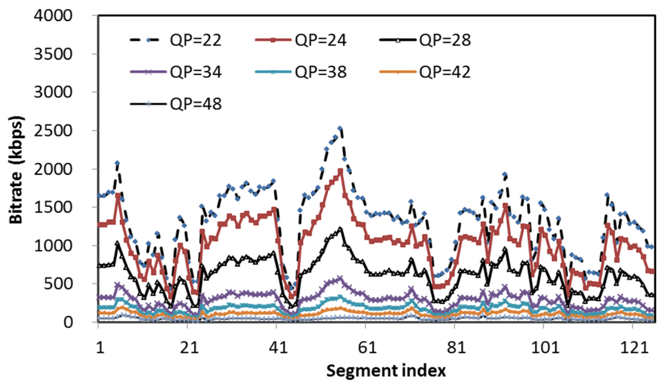


Fig. 11. Segment bitrate of diffirent alternatives of test video

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 1150kbps (resulting in a throughput of about 1090kbps), alternative #6 (with average bitrate of 1015.1kbps) will be selected. However, due to the variations of segment bitrate, playing this alternative makes the buffer level gradually reduced from 8s to 2s as shown in Fig. 12. In this case, the initial buffering must be more than 6s (i.e. 8s - 2s) 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 streaming 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 (4s) to only one segment duration (2s), 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).

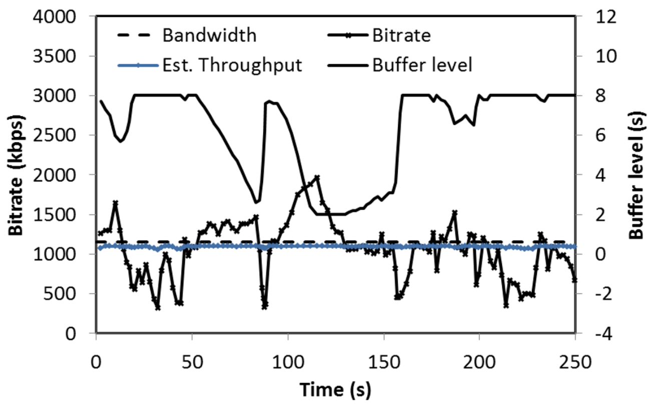


Fig. 12. Client behavior using the average segment bitrate

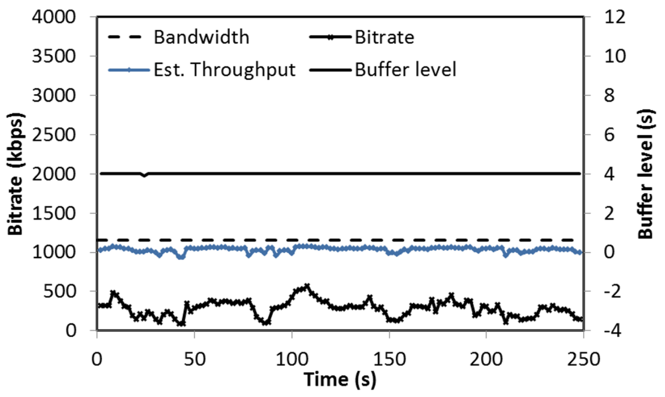


Fig. 13. Client behavior when the representative bitrate of an alternative is its highest segment bitrate

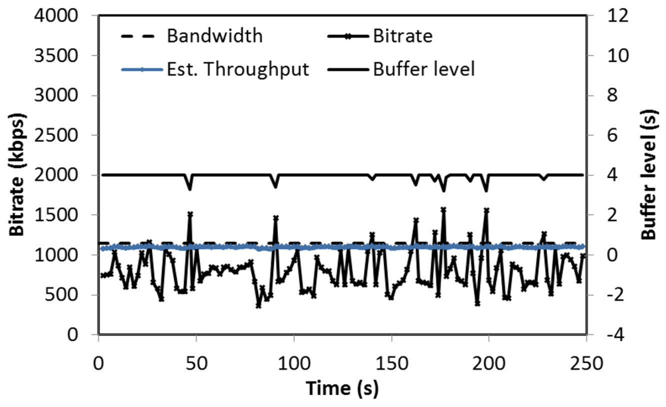


Fig. 14. Client behavior when bitrate estimation is enabled

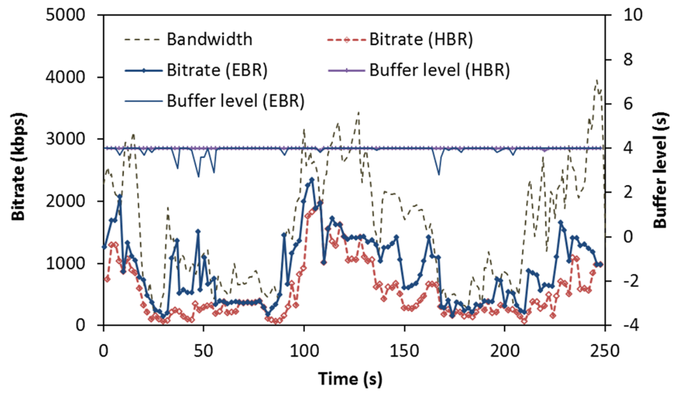


Fig. 16. Comparison of the client behavior when the adaptation is based on the highest bitrate (HBR) or the estimated birate (EBR).

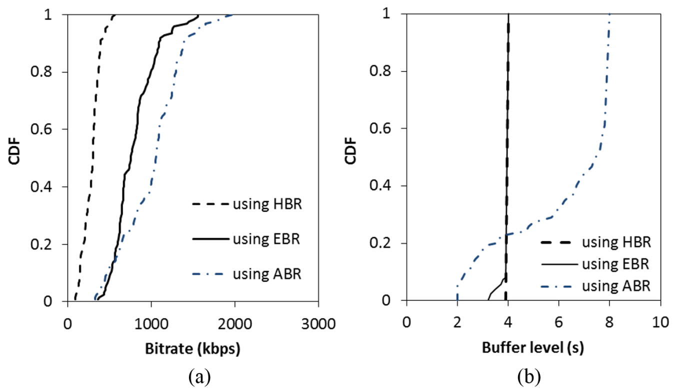


Fig. 15. CDF of bitrate (a) and CDF of buffer level (b) when the adaptation is based on the average birate (ABR), the highest bitrate (HBR), or estimated birate (EBR)

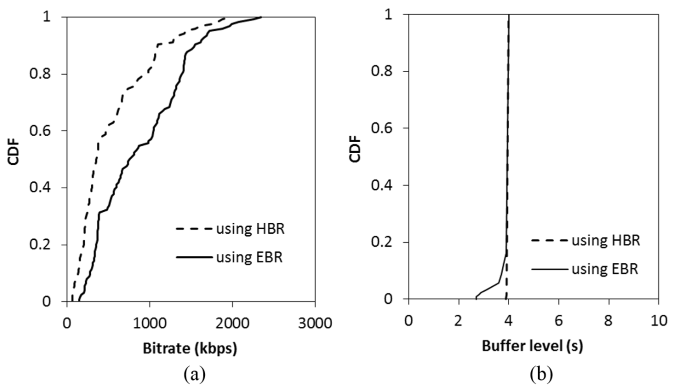


Fig. 17. CDF of bitrate (a) and CDF of buffer level (b) when the adaptation is based on the highest bitrate (HBR) or estimated birate (EBR.

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 included 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. 17a 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 (Fig. 16 and Fig. 17b).

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. 2s or 4s), the improvement in buffer stability of our proposed method compared to the aggressive method is not significant (about 1s). 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 1ms). 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.

VII. ACKNOWLEDGMENT

The authors would like to sincerely thank the anonymous reviewers for their helpful and constructive comments that greatly contributed to improving the paper. The authors are also grateful to Prof. Christian Timmerer of Klagenfurt University for providing the bandwidth trace used in this study.

REFERENCES

[1] A. C. Begen, T. Akgul, M. Baugher, "Watching video over the Web, Part I: Streaming protocols," IEEE Internet Computing, vol. 15, no. 2, pp. 54-63, Mar. 2011.

[2] A. C. Begen, T. Akgul, M. Baugher, "Watching video over the Web, Part II: Applications, Standardization, and Open Issues," IEEE Internet Computing, vol. 15, no. 3, pp. 59-63, Apr. 2011.

[3] T. C. Thang, Q-D Ho, J. W. Kang, A. T. Pham, "Adaptive Streaming of Audiovisual Content using MPEG DASH". IEEE Trans. on Consumer Electronics, vol. 58, no. 1, pp. 78-85, Feb. 2012.

[4] J.-W. Park, R. P. Karrer, J. Kim, "TCP-ROME: A Transport-Layer Parallel streaming Protocol for Real-Time Online Multimedia Environments," Journal of Communication and Networks, vol. 13, no. 3, pp. 277-285, 2011.

[5] T. V. Lakshman, A. Ortega, A. R. Reibman, "Variable bit rate (VBR) video: Tradeoffs and potentials," Proceedings of the 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, S. T. Park, "Signaling metadata for adaptive HTTP streaming", ISO/IEC JTC1/SC29/WG11 m17771, Geneva, Jul. 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, C.-S. Yang, "Proxy Design for Improving the Efficiency of Stored MPEG-4 FGS Video Delivery over Wireless Networks," Journal of Communication and Networks, vol. 6, no. 3, pp. 280-286, 2004.

[9] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in Proc. IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM), Jun. 2011.

[10] C. Liu, I. Bouazizi, M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in Proc. of ACM MMSys2011, 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, 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 MMSys2011, 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, C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in Proc. ACM MMSys2011, California, Feb. 2011.

[16] J. R. Corbera, P. A. Chou, S. L. Regunathan, "A Generalized Hypothetical Reference Decoder for H.264/AVC", IEEE Trans. CSVT, vol. 13, no. 7, pp. 674-687, Jul. 2003.

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

[18] T. C. Thang, J. Y. Lee, J. W. Kang, S. J. Bae, S. Jung, 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, S. Mascolo, "An experimental investigation of the Akamai adaptive video streaming," in Proc. of USAB2010, pp. 447-464, Klagenfurt, Austria Nov. 2010.

[21] K. Evensen et al., "Mobile Video streaming using location-based network prediction and transparent handover," in Proc. NOSSDAV2011, pp. 21-26, Jun. 2011.

[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'l J. of Network 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, D. Tjondronegoro, "Bitrate Modelling of Scalable Videos Using Quantization Parameter, Frame Rate, and Spaial Resolution," in Proc. ICASSP 2010, pp.2334-2337, 2010.

[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. 17th Packet Video Workshop, 2009.

[28] J. Ascenso, M. Jakubowski, G. Pastuszak, "Constant Bitrate Control for a Distributed Video Coding System ", in Proc SIGMAP, Porto, Jul. 2008

[29] V. Paxson, M. Allman, J. Chu, M. Sargent, "Computing TCP's Retransmission Timer," RFC 6298, Jun. 2011. http://tools.ietf.org/html/rfc6298

[30] G. Van der Auwera, P. T. David, 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, September 2008.

[31] Wiegand, T, Sullivan, G. J., Bjontegaard, G., and Luthra, A., "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 Communication Review, 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 Multimedia Systems Conference 2012 and the 4th ACM Workshop on Mobile Video, North Carolina, Feb. 2012.