Adaptive Streaming of Audiovisual Content using MPEG DASH

Truong Cong Thang, *Member*, IEEE, Quang-Dung Ho, Jung Won Kang,
Anh T. Pham, *Senior Member*, IEEE

**Abstract — ***HTTP streaming has become a cost effective means for multimedia delivery nowadays. In this paper, we study the use of MPEG DASH standard to stream audiovisual content. We first describe a novel estimation method for connection throughput. Then, employing the extensibility feature of DASH syntax, we present a systematic method for selecting the best audio and video alternatives given the estimated throughput. The experiments show that our solution is stable to short-term fluctuations while responding quickly to large fluctuations*[1].

Index Terms — *HTTP streaming, adaptivity, audiovisual content, DASH.*

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]. In this trend, a new standard called Dynamic Adaptive Streaming over HTTP (DASH) has been developed by MPEG and 3GPP to enable the interoperability in the industry [3][4].

Due to the heterogeneity of today's communication networks, adaptivity is the most important requirement for any streaming client [5][6]. Especially, TCP, the underlying layer of HTTP, is notorious for its throughput fluctuations [7]. In HTTP streaming, the (Web) server usually has very little knowledge about client/network status; it is the client that makes decisions on content delivery to maintain a good quality of service (QoS). For that reason, a client needs rich signaling metadata which should be prepared by the content/service providers in advance or on the fly [8].

Obviously, one of the typical applications of DASH would be audiovisual streaming service. Video component has been the main focus in previous studies due to its high bitrate. However, in some cases, the bitrate of audio is comparable to that of video. For example, when providing a music video service over mobile networks, where the actual throughput may drop to several hundreds of kilobits per second (kbps) [9], the audio and video bitrates will not be much different. Similar situations could exist for users of heavily-used networks, such as guests in a crowded hotel. Besides, to evaluate the QoS of audiovisual services, some previous studies suggested empirical quality models of audiovisual content [10][11][12]. Yet, there has been very little work on the systematic adaptation of audiovisual content so far.

Truong Cong Thang is with the University of Aizu, Aizu-Wakamatsu, Japan (e-mail: thang@u-aizu.ac.jp).
Quang-Dung Ho is with McGill University, Montreal, Quebec, Canada (e-mail: quang.ho@mcgill.ca).
Jung Won Kang is with Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: jungwon@etri.re.kr).
Anh T. Pham is with the University of Aizu, Aizu-Wakamatsu, Japan (e-mail: pham@u-aizu.ac.jp).

Manuscript received 01/16/12
Current version published 03/21/12
Electronic version published 03/21/12.

For adaptivity to networks and terminal capabilities, a provider should generate multiple alternatives of each component (video or audio) as well as the signaling metadata that contains the characteristics of the alternatives (such as bitrate, resolution, etc.). However, as explained later, it is still difficult for a client to select alternatives of audio and video using the metadata of DASH.

In this paper, based on an optimization problem formulation, we propose a solution to support adaptivity for audiovisual streaming application using DASH. Employing a novel throughput estimation method, our solution both is stable to short-term fluctuations and quickly responds to large fluctuations. Further, employing the extensibility feature of DASH syntax, we introduce quality information for each alternative and quality model for audiovisual content in the syntax, thus helping to optimally solve the formulated problem.

The paper is organized as follows. In Section 2, we first provide an overview of adaptive HTTP streaming and related work. A problem formulation is presented and discussed in Section 3, which results in the rationales of the proposed solution. Then we describe a systematic method to estimate the throughput in Section 4. In Section 5, we present a metadata-based method to optimally select audio and video alternatives given the estimated throughput. Experiments with different cases of transport are presented in Section 6. Finally, conclusions and future work are given in Section 7.

II. OVERVIEW OF ADAPTIVE HTTP STREAMING

A. HTTP Streaming and DASH Standard

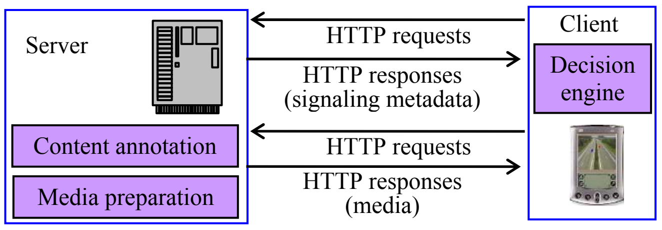


Fig. 1. HTTP streaming architecture

The basic architecture of adaptive HTTP streaming is shown in Fig. 1. Here, the content annotation module provides knowledge (or metadata) about the content, ranging from semantic level (e.g. genres) to physical level (e.g. bitstream structure/characteristics). The media preparation module provides tools for media adaptation/transcoding, packetization, encapsulation, etc., so that the media could be efficiently delivered to the client. The metadata provided by content annotation module 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.

In essence, MPEG DASH specifies the metadata and media formats exchanged between clients and servers. The metadata is called Media Presentation Description (MPD) in MPEG DASH. With conventional streaming, signaling metadata and media are usually delivered by different protocols (e.g. by RTSP and RTP respectively) [13]. Meanwhile, in HTTP streaming, both signaling metadata and media are usually delivered by HTTP protocol.

In MPEG DASH, content may be composed of one or more content *component* (e.g. video, audio). A long content item could be divided into one or more temporal chapter (or *periods*). Further, each alternative (also called *representation*) of a content component could be divided into media *segments*. The client will process the metadata for a period, and request appropriate segments of that period. It then gets metadata for the next period, and the whole process is started over. In most cases, for each request from the client, the server will send one segment (or sub-segment). So, media will be delivered by a sequence of HTTP request-response transactions.

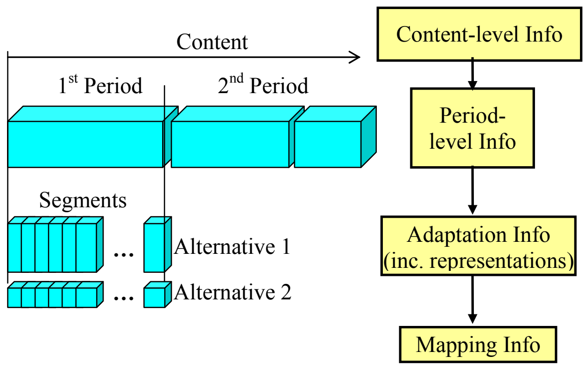


Fig. 2: Hierarchy of content division and levels of signaling metadata

An illustration of media division hierarchy together with the corresponding parts/levels of signaling metadata is shown in Fig. 2. In general, the signaling metadata of DASH could be divided into the following categories.

- Content-level information: includes the common description of content, such as available time, duration, minimum initial buffering, etc.
- Period-level information: provides the common description of a period, such as start time, duration, unique identifier, etc. Some hints on typical combinations of representations could be included in this level.
- Adaptation information: describes the characteristics of representations and representation groups (called *adaptation sets* in DASH), such as bitrate, resolution, quality. An alternative could be physical, i.e. created in advance, or virtual, i.e. to be created on the fly. Based on the information of alternatives, the client will select a segment of appropriate alternative, and thus supporting the adaptivity to terminal and networks.
- Mapping information: describes the locations to retrieve actual media data. In DASH, location information is available not only in MPD but also in some boxes of media segment format.

For efficiency in managing/processing the metadata, a metadata document may be fragmented into different parts. The separation of metadata parts also enables efficiency in storage and delivery. For example, during a session, content-level metadata could be sent once, and only period information is periodically loaded. More information about the structure and basic concepts of DASH could be found in [2][3][4].

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 two requests is approximately the duration of the first request's segment. So, if segments have the same duration of t seconds, the distance between requests will be t as well. Meanwhile, in on-demand streaming, requests could be sped up to quickly fill the buffer [14]. Note that, the term "initial buffering" in the following means the length (in seconds) of media needed in the buffer before the playout can start. Also, as a client can decide the receiving bitrate, buffer overflow will not be a problem in HTTP streaming.

B. *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 [4][2]. A detailed analysis on the use of DASH for live service is presented in [15], where an initial buffering of about 2 segment durations is suggested.

Detailed investigations of adaptivity in some commercial clients are carried out in [14][16], providing some insights into the behaviors (but not exactly algorithms) of the clients. For example, when there is a large change in connection throughput, Netflix's client is found to be more aggressive than Microsoft's client [14].

In [17], 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 [7]). Of course, other measures such as buffer level could be taken into account [18]. However, these solutions would require many empirical rules and thresholds to change media bitrate. 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 [9]. Further, the overall throughput could be improved by using of multiple connections for HTTP streaming [18][19]. In [20], a Java client for HTTP streaming on Android platform is developed and different algorithms using different throughput estimation ways are compared.

The use of Scalable Video Coding (SVC), a new video format for ease of adaptivity, has been shown to improve the storage and caching efficiency for HTTP streaming [23]. Also, the advantages and disadvantages of different strategies to schedule HTTP requests are investigated in [21][22].

Though HTTP audiovisual streaming is investigated in [14], the audio bitrate is fixed and only the adaptive behavior of video stream is studied. In [11], we have proposed a graph-based quality model that systematically quantifies different components of audiovisual quality. In [12], subjective evaluation of audiovisual quality at very low bitrates shows that the optimal ratio of audio bitrate and video bitrate widely depends on the content and the bitrate budget.

III. General Problem Formulation

In this Section a general problem formulation, together with its important aspects, is presented. Rather than following a TCP-like approach, we will tackle the adaptivity for HTTP audiovisual streaming as an optimization problem.

In practice, media adaptation is sequentially applied to consecutive temporal intervals, where the resource characteristics (e.g. bandwidth) in each interval could be considered as stable. Suppose that, in a given interval, we need to adapt N components $\{E_n, 1\leq n\leq N\}$ to meet a bitrate constraint R^c. Each alternative E_n^* of component n has bitrate R_n and quality Q_n. The adaptation problem can be defined as an optimization problem as follows.

Find E_n^* for each E_n so as to maximize the overall quality OQ which is a function of $\{Q_n\}$

$$OQ = f(Q_1, Q_2, \ldots, Q_N) \quad (1)$$

and satisfy $$\sum_{n=1}^{N} R_n \leq R^c \quad . \quad (2)$$

Of course, more constraints (e.g. limited display size) could be added into this problem. To solve the above problem, our solution consists of two important aspects:

- Estimation of connection throughput, which is then used to find the bitrate constraint R^c.
- Provision of additional metadata related to audiovisual quality to enable optimal selection of alternatives.

In the next two Sections, we will consider these two aspects respectively.

IV. Throughput Estimation

In this paper, media segments are also delivered by a sequence of consecutive HTTP request-response transactions as in most previous work. The throughput in general is obtained by dividing the amount of data (data size) by the delivery duration. As it is important to know the actual amount of data delivered by a pair of request-response, the segment throughput $T_s(i)$ for each delivered segment i is computed using the request-response duration, which is from the instant of sending the request to the instant of receiving the last byte of the response.

Obviously, before delivering a segment i, it is important to have the estimated throughput $T_e(i)$. In [20], the throughput of the last segment (or instant throughput) is used as the estimated throughput. However, due to short-term fluctuations, the estimated throughput obtained in this way will cause frequent fluctuations in media quality.

A straightforward way to cope with short-term fluctuations is to use a "smoothed" throughput measure [19][20], resulting in a stable estimation over time. However, the disadvantage of smoothed throughput is that it causes late reaction of the client to large throughput decrease, which in turn must be handled by having a large initial buffering and continuously checking whether the buffer level is lower than a safety threshold [20]. However, the use of buffer level may not be very effective. It is because, when the actual throughput decreases significantly, the estimated throughput using a smoothed measure may be still very high. So, some segments whose bitrates are higher than actual throughput are received already when a possibility of underflow is detected.

Our goal in this Section is to have an estimated throughput measure which 1) is stable during intervals of short-term fluctuations as well as 2) quickly responds to significant fluctuations.

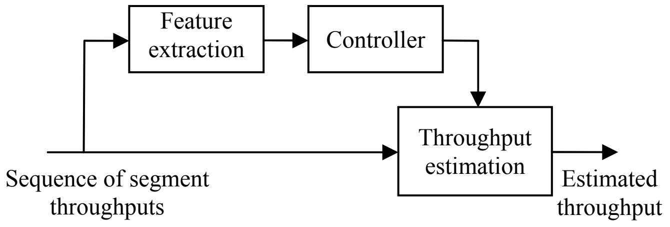


Fig. 3. Proposed framework of throughput estimation

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

To obtain the estimated throughput for segment i, we use a form of running average as follows [14][19]:

$$T_e(i) = \begin{cases} (1-\delta)T_e(i-2) + \delta T_s(i-1) & i > 0 \\ T_s(i-1) & i = 1,2 \end{cases} \quad . \quad (3)$$

Note that we do not find the estimated throughput $T_e(0)$ for segment 0; this segment is usually downloaded by simply requesting the lowest bitrate alternative.

With the above function, the higher the value of δ is, the more the estimated throughput depends on the last segment throughput. To have a smoothed value of throughput, a small value of δ should be used. In this paper, the value of δ will be adaptively controlled so that the above goal is met.

As for the feature extraction block, currently we define the following feature, called the normalized throughput deviation:

$$p = \frac{|T_s(i) - T_e(i)|}{T_e(i)} \quad . \quad (4)$$

A high value of p means that the connection throughput is likely to have a significant change. In general, a very significant change would require a quick reaction, and thus the value of δ should approach to 1. Meanwhile, a small value of p would suggest to use a strongly smoothed throughput metric, where δ approaches to a minimum value δ_{min} which is usually set to 0.05~0.2 (e.g. [14][19]). The general relationship between p and δ can be modeled by the logistic function as follows:

$$\delta = \frac{1}{1+e^{-k(p-P_0)}} \qquad (5)$$

where k and P_0 are the parameters of the logistic function. Obviously, the actual values of k and P_0 depend on the specific networks in use. From the observations with our test-bed, most short-term fluctuations correspond to p in the range of [0, 0.05] for wired networks and [0, 0.2] for wireless networks. Currently, we set $k = 21$ and $P_0 = 0.2$ which provide a specific p-δ relationship for our wireless network as shown in Fig. 4. Using this model, any changes of segment (or instant) throughput with $p > 0.4$ will result in δ equal to 1, while changes with $p < 0.1$ will result in very small δ ($\delta < 0.1$). It should be noted that this framework is different from the method of [19], where the variance of throughput is used to compute a safety margin for estimated throughput.

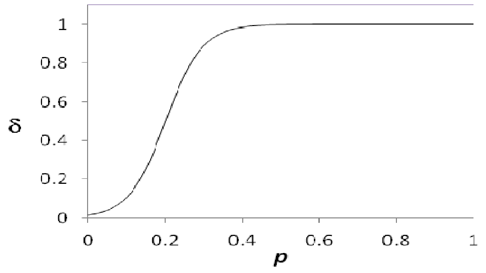


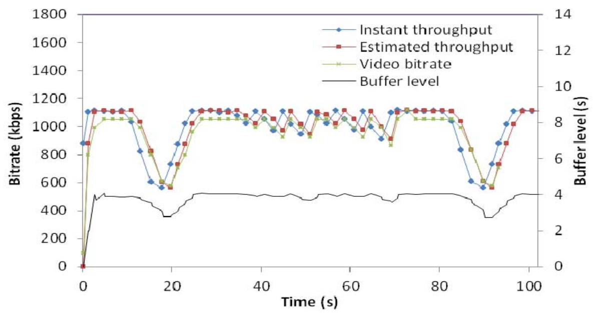
Fig. 4. Relationship of δ and p

Fig. 5 shows a comparison of three different throughput estimation methods, namely 1) using the last segment throughput only, 2) using the smoothed throughput only, and 3) using the proposed framework. For this comparison, only one video stream is delivered via a connection with both small and large fluctuations. Video is provided with 16 bitrates, from 1024kbps to 64kbps with step size of 64kbps. All video segments have the same length of 2 seconds. The initial buffering is set to 4s. Distance between two consecutive requests normally is 2s; however, if the previous response is delayed, each of the following requests will be sent right after fully receiving a response until the buffer level attains 4s.

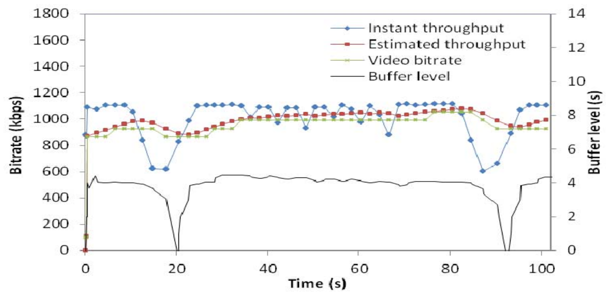
In Fig. 5, the *instant throughput* curve represents the segment throughput T_s. Video bitrate is decided by selecting the highest value that is smaller than or equal to the estimated throughput. We can see that, the estimation using the last segment throughput results in a rather stable buffer. However, the video bitrate changes frequently (Fig. 5(a)). With the estimation using smoothed throughput, video bitrate does not change much. Yet, due to the overestimated throughput at instants of 18s and 90s the buffer depletes very quickly (Fig. 5(b)). Meanwhile, the estimation using our framework provides the advantages of both previous methods, i.e. a stable buffer and less frequent changes of video bitrate (Fig. 5(c)).

In the above, our proposed estimation method is presented for a single connection. When the client uses multiple connections, the total instant throughput is first computed by adding the instant throughputs of all connections as in [18][19]. Then the total estimated throughput will be computed as the above using the obtained total instant throughput.

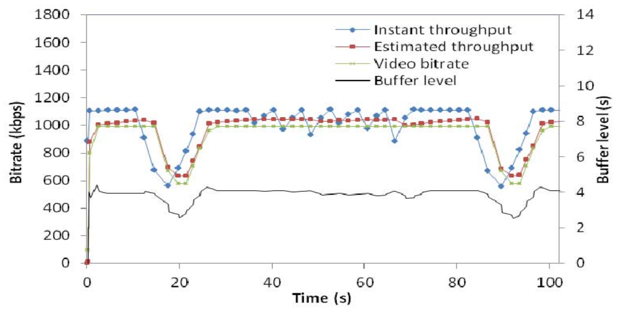
Our goal in the future is to improve the throughput estimation process so that the client can quickly recognize the different throughput patterns of different networks and then automatically adjust the estimation. For that, more throughput-related features and machine learning approach will be employed.



(a) Estimation using the last segment throughput



(b) Estimation using the smoothed throughput



(c) Estimation using the proposed framework

Fig. 5. Comparison of different throughput estimation methods

V. Optimal Selection of Audio and Video Alternatives

In this Section, given the estimated throughput computed in the above, we will investigate the selection of audio/video alternatives so as to give the users the best possible quality.

Basically, the problem formulated in Section 3 will be solved for each segment interval i. To obtain the bitrate constraint $R^c(i)$ from the estimated throughput, a simple safety margin μ is used as follows:

$$R^c(i) = (1-\mu)T_e(i) \qquad (6)$$

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

As mentioned, our focus is on a metadata-enabled adaptivity solution for audiovisual content using MPEG DASH. In the DASH standard, a representation is optionally assigned with the *qualityRanking* attribute, which is a relative ranking value to help select the best representation from an adaptation set given some resource constraints. However, in audiovisual streaming, there are usually two adaptation sets (one for video and the other for audio). So, *qualityRanking* obviously is not relevant to select representations of components of audiovisual content.

Suppose that E_1 and E_2 are video and audio components in the above formulation, we introduce 1) quality information Q for each representation and 2) quality model OQ for audiovisual content in the syntax. These new features are implemented using the extensibility of DASH XML syntax, so that the validity of MPD is not broken.

The syntax of DASH provides various places for syntax extensibility. In our solution, the above features are provided as extensions of *Subset* element of DASH. Actually, it is Subset element that provides the hints on combinations of different adaptation sets to make a presentation within a period [3]. The advantage of extending Subset element is to have a separate place for quality information, not having to extend every Representation element. Further, it is reasonable to describe the quality model for each targeted subset.

```
    <!-- Quality information of representations -->
    <element name="RepsQuality" type="RepsQualityType"/>
    <complexType name="RepsQualityType">
        <sequence>
            <any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="repIDs" type="StringVectorType"
use="required"/>
        <attribute name="repQs" type="DoubleVectorType"
use="required"/>
    </complexType>

    <!-- Quality model of a subset with video and audio components-->
    <element name="AVQualityModel" type="AVQualityModelType"/>
    <complexType name="AVQualityModelType">
        <sequence>
            <any namespace="##other" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
        <attribute name="vi" type="xs:double" use="optional"/>
        <attribute name="au" type="xs:double" use="optional"/>
        <attribute name="av" type="xs:double" use="optional"/>
    </complexType>
```

Fig. 6. Syntax of RepsQuality element and AVQualityModel element

The quality information of representations is described by the RepsQuality element (Fig. 6). Two important attributes of this element are *repIDs* and *repQs*. The former is a vector (or list) of representation id's and the later is the list of corresponding "normalized" quality values [11], which are in the range of [0, 1]. Because a representation's id is unique within a period, a RepsQuality element may contain id's of representations from different adaptation sets. As for quality model (actually Eq. (1) above) we adopt the graph-based model of our previous work [11]. The AVQualityModel element (Fig. 6) has three attributes *vi*, *au*, and *av,* indicating the weights of video component, audio component, and the strength of the relation between the two components. When an attribute is absent, its default value is 0. The overall quality is computed by:

$$OQ = vi{\cdot}Q_1 + au{\cdot}Q_2 + av{\cdot}Q_1{\cdot}Q_2 \qquad (7)$$

where Q_1 and Q_2 are video quality and audio quality respectively.

```
<MPD  ... >
    <Period id=”P1” duration=”PT15M0S”/>
        <AdaptationSet id=”1”>
            <!-- Representations of video component -->
            <Representation id=”v0” bandwidth=”1024” ....../>
              ...
            <Representation id=”v15” bandwidth=”64”   ....../>
        </AdaptationSet>
        <AdaptationSet id=”2”>
            <!-- Representations of audio component -->
            <Representation id=”a0” bandwidth=”128”....../>
              ...
            <Representation id=”a3” bandwidth=”32”   ...../>
        </AdaptationSet>
        <SubSet contains=”1  2”/>
          <qi:RepsQuality
           repIDs=”v0 v1  v2  v3  v4  v5  v6  v7  v8  v9 v10 v11 v12 v13 v14 v15”
           repQs=”1.0 .99 .97 .96 .95 .93 .91 .88 .84 .80 .74  .68  .61  .54  .46 .40”/>
          <qi:RepsQuality
              repIDs=”a0  a1   a2    a3”
              repQs=”1.0 0.95 0.86 0.64”/>
          <qi:AVQualityModel vi=”0.58” au=”0.35” av=”0.07”/>
        </SubSet>
    </Period>
</MPD>
```

Fig. 7. An example MPD with extended features

Fig. 7 shows an example MPD of an audiovisual content item. This MPD has one Subset element informing that adaptation sets #1 and #2 could be combined to make a presentation. In this example, adaptation set #1 (#2) contains all representations of video (audio) component. Inside the Subset element, the part of extension is marked by prefix "*qi:*" which is the namespace for our extended elements. Audio component of this content has four representations, with bitrates of 128, 96, 64, 32 (kbps). Video component has 16 representations, with bitrate values ranging from 1024 to 64 (kbps). This content item contains music and descriptive scenery, whose quality model's basic parameters are 0.58, 0.35, and 0.07 (adopted from [11]).

So, using the above extended metadata, the optimal solution of the above problem can be found. Because the solution space of this problem in practice is usually small, the optimal solution could be easily obtained by a full search. In case the problem has many components and alternatives, fast algorithms based on dynamic programming could be used to find the optimal solution [24][25].

VI. Experiments

The organization of our test-bed is shown in Fig. 8. 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 [26], which is installed at the client to simulate network fluctuations, has RTT of 40ms and packet loss of 1%. Available bandwidth for the client is controlled by DummyNet as shown in the following.

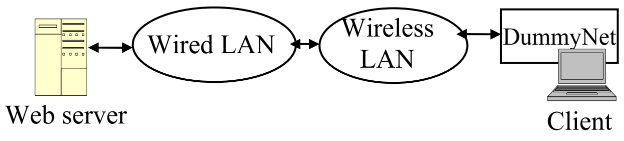
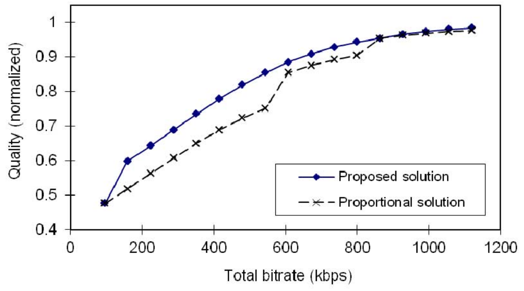


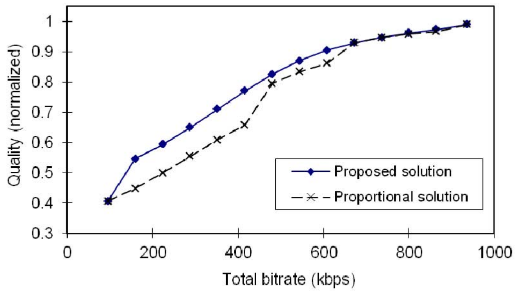
Fig. 8. Test-bed organization for the experiments

In our test-bed, the client (including HTTP client and media player) is implemented in Java. The client runs on a Windows 7 Professional notebook with 2.0GHz Core2Duo CPU and 2G RAM. Video and audio components are encoded by the main profile of AVC (Advanced Video Coding) and AAC-LC (Advanced Audio Coding - Low complexity profile) formats respectively. Video component has frame rate of 30fps, resolution of 320x240. All media segments have the same duration of 2s and the length of media presentation is 30s. Media segments of each component are stored in separate small files. The segments are then accessed by HTTP requests, each containing the URL of a segment. The initial buffering level is set to 4s (i.e. two segments). The rule for request scheduling is the same as in Section 4. Audiovisual content in our experiment is described by an MPD with only one period. Note that small initial buffering is used here to see if the system can support live streaming. For on-demand streaming, it is easy to speed up the requests to maintain a larger buffer level [14].

In the first part of experiments, we will investigate the optimal selection of audio and video alternatives. Fig. 7 presents the MPD of a content item used in this experiment. The proposed adaptation solution is compared with a simple solution, where the video and audio bitrates are proportionally reduced so that their portions in the total audiovisual bitrate are roughly unchanged. Fig. 9(a) shows the normalized quality of the two solutions w.r.t. total bitrate (R_1+R_2). We can see that our proposed solution can improve the overall quality up to 0.1, which corresponds to 0.5 on the Mean Opinion Score (MOS) 0~5 scale. A similar result is shown in Fig. 9(b) for an education content item, where the video component has 13 bitrates from 832 to 64(kbps) and audio component has the same four bitrates as above. For this second content item, the quality model's values are 0.44, 0.38, 0.18, which are also obtained from [11]. Note that, to obtain audiovisual quality, the quality values of components should be on the same quality scale. In practice, audio and video media are often associated with PSNR quality values (e.g. provided by video and audio encoders). These quality values could be used in our model by first converting into MOS values and then normalized into [0, 1] range.



(a)



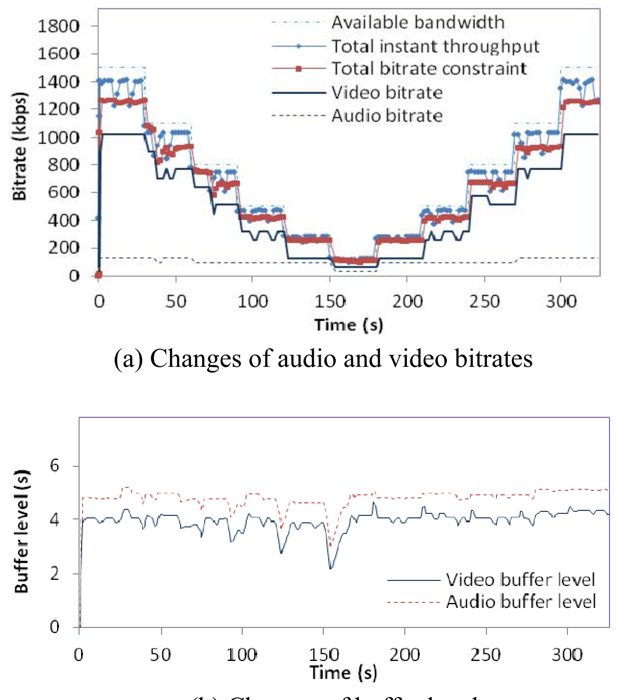
(b)

Fig. 9. Comparison of the proposed solution and proportional solution

The second part of experiments is to investigate the adaptivity of the client to available bandwidth (adjusted by DummyNet). Two sub-experiments are carried out: 1) using one TCP connection (for both video and audio) and 2) using two TCP connections (one for video and one for audio). For both cases, the client repeatedly downloads and plays the first content item to make a long session of 330s.

In the first case, video and audio data are delivered on the same connection. To improve throughput usage, we allow pipelining within a segment interval. Specifically, the request for video is sent immediately after the request for audio. However, the data for the next segment interval is requested only after all responses for the previous interval are fully received. The total segment/instant throughput is computed using the total data size of audio and video in one segment interval. The estimated throughput is computed using the total instant throughput as presented in Section 4. Finally, total bitrate constraint R^c is obtained by Eq. (6) with $\mu = 0.1$.

Fig. 10(a) shows the change of video and audio bitrates according to the available bandwidth which is drastically changed by DummyNet. In this figure, we can see that the bitrates of audio and video change well in accordance with the available bandwidth. This means that the throughput estimation method works well in a wide range of bandwidth. Corresponding audio and video buffer levels provided in Fig. 10(b) show that both audio and video buffers usually are maintained at 4s. Actually, the buffer level of audio is higher than that of video because, for each segment interval, we always receive audio data before video data while the playouts of audio and video are at the same time. These small but stable buffer levels imply that the client can play the content in live streaming mode.



(a) Changes of audio and video bitrates

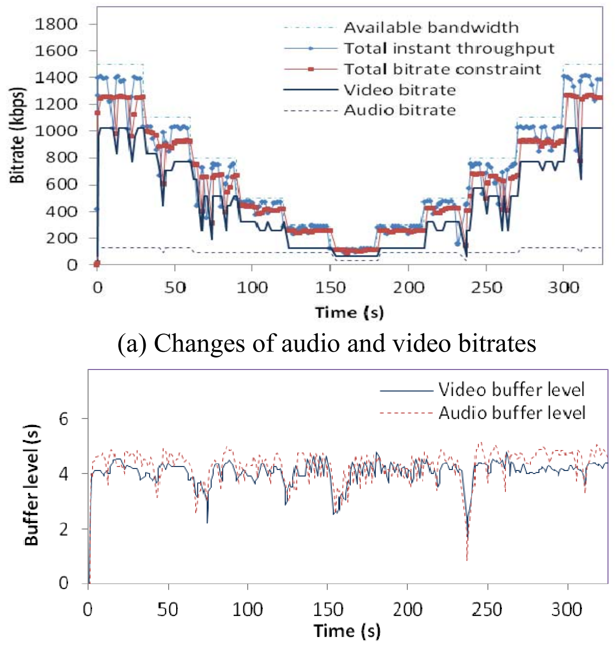
(b) Changes of buffer levels

Fig. 10. Adaptive behavior of the client to changes of available bandwidth in one connection case.

In the case of two connections, each connection will deliver one content component (video or audio). Media segments in each connection are requested sequentially without any pipelining. In [7][19], it is shown that the use of multiple paths to multiple servers improves the overall throughput. However, from our observation, it seems that the two connections affect each other (Fig. 11(a)). This phenomenon is possibly due to the fact that the two connections in our experiment have the same end-points and share the same bottlenecks. Meanwhile, the multiple paths in [7][19] usually do not have to compete for the low bandwidth in the last mile. In fact, our observation is consistent with the findings in [27] where the use of parallel connections at a last-mile low-speed link is shown to be not useful. The reason could be that, in such bottlenecks, those (TCP) connections are highly bursty and have to compete with each other. Of course, when there is no bottleneck, using multiple connections could improve the total throughput [7][27]. However, this is unfair to other applications that use only one connection.

The corresponding buffer levels for the case of two connections (Fig. 11(b)) are obviously less stable than in the case of one connection. In Fig. 11(b), the audio buffer level is still a little higher than video buffer level though audio and video data are requested concurrently for each segment interval. This is because the amount of audio data usually is smaller than video and thus the reception of audio data is mostly completed before video data.

So, it is suggested that, under situations of bottlenecks, a single TCP connection should be used instead of multiple TCP connections to stream audiovisual content. A disadvantage of using a single TCP connection is that there could be some delay for a content component. For example, in our experiment above, the client starts to receive a video segment only after the audio segment in the same interval is fully received. That is, it is impossible to play an audiovisual content item immediately after receiving some small amount of data. This problem could be solved using interleaving technique we presented in [28], where the interleaving depth will determine the minimum delay (waiting time) before the content can be played.



(a) Changes of audio and video bitrates

(b) Changes of buffer levels

Fig. 11. Adaptive behavior of the client to changes of available bandwidth in two connection case.

VII. Conclusion

In this paper, we have studied the use of a new standard called Dynamic Adaptive Streaming over HTTP (DASH) for audiovisual content streaming. Using a novel throughput estimation method and additional quality-related metadata, our solution can provide the best possible quality to the users while maintaining a stable buffer under drastic changes of available bandwidth. Two different transport options were also investigated for audiovisual content. It is found that, under situation of bottlenecks, a single connection is better than two connections. In the future, we will try to automate the throughput estimation for different types of networks using machine learning approach. Also, extensive subjective experiments will be carried out with audiovisual content to find some common adaptation strategies for different content genres.

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, "Applications, Standardization, and Open Issues," *IEEE Internet Computing*, vol. 15, no. 3, pp. 59–63, Apr. 2011.

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

[4] T. Stockhammer, “Dynamic adaptive streaming over HTTP - standards and design principles,” in Proc. ACM MMSys2011, California, Feb. 2011.
[5] J. W. Kang, S. Jung, J.-G. Kim, J. Hong, "Development of QoS-aware ubiquitous content access testbed,” *IEEE Trans. Consumer Electron.*, vol. 53, no. 1, pp. 197-203, Feb. 2007.
[6] H. Lee, Y. Lee, J. Lee, D. Lee, H. Shin, “Design of a mobile video streaming system using adaptive spatial resolution control,” *IEEE Trans. Consumer Electron.*, vol. 55, no. 3, pp. 1682-1689, Aug. 2009.
[7] S. Tullimas, T. Nguyen, R. Edgecomb, S.-C. Cheung, “Multimedia streaming using multiple TCP connections,” *ACM TOMCCAP*, vol. 4, no. 2, pp. 1-20, Feb. 2008.
[8] T. C. Thang et al., “Signaling metadata for adaptive HTTP streaming”, ISO/IEC JTC1/SC29/WG11 m17771, Geneva, Jul. 2010.
[9] K. Evensen et al., "Mobile Video streaming using location-based network prediction and transparent handover," in Proc. NOSSDAV2011, pp. 21-26, Jun. 2011.
[10] D. S. Hands, “A basic multimedia quality model”, *IEEE Trans. Multimedia*, vol. 6, no. 6, pp. 806–816, Dec. 2004.
[11] T. C. Thang, J. W. Kang, Y. M. Ro, "Graph-based perceptual quality model for audiovisual contents", in Proc. IEEE ICME2007, pp.312-315, Beijing, Jul. 2007.
[12] S. Winkler, C. Faller, “Perceived audiovisual quality of low-bitrate multimedia content,” *IEEE Trans. Multimedia*, vol. 8, no. 5, pp. 973-980, Oct. 2006.
[13] 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
[14] 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.
[15] 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.
[16] 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.
[17] C. Liu, I. Bouazizi, M. Gabbouj, “Rate adaptation for adaptive HTTP streaming,” in Proc. of ACM MMSys2011, California, Feb. 2011.
[18] K. Evensen, et al. “Improving the performance of quality-adaptive video streaming over multiple heterogeneous access networks,” in Proc. ACM MMSys2011, California, Feb. 2011.
[19] S. Gouache, G. Bichot, A. Bsila, C. Howson, “Distributed & adaptive HTTP streaming,” in Proc. IEEE International Conference on Multimedia and Expo (ICME), Barcelona, Spain, July 2011.
[20] L. R. Romero, “A dynamic adaptive HTTP streaming video service for Google Android,” M.S. Thesis, Royal Institute of Technology (KTH), Stockholm, Oct. 2011.
[21] T. Kupka, C. Griwodz, P. Halvorsen, "An Evaluation of Live Adaptive HTTP Segment Streaming Request Strategies," in Proc. IEEE Conference on Local Computer Networks (LCN), Bonn, Germany, October 2011.
[22] R. Kuschnig, I. Kofler, H. Hellwagner, “Evaluation of HTTP-based request-response streams for internet video streaming,” in Proc. ACM MMSys2011, California, Feb. 2011.
[23] Y. Sanchez de la Fuente, et al. “iDASH: improved dynamic adaptive streaming over HTTP using Scalable Video Coding,” in Proc. ACM MMSys2011, California, Feb. 2011.
[24] T. C. Thang, Y. J. Jung, and Y. M. Ro, "Dynamic Programming Based Adaptation of Multimedia Contents in UMA," in Proc. PCM2004, pp. 347-355, Tokyo, 2004.
[25] T. C. Thang, J. W. Kang, J.-J. Yoo, Y. M. Ro, "Optimal multi-layer adaptation of SVC video over heterogeneous environments", *Journal of Advances in Multimedia*, vol. 2008, Article ID 739192, 8 pages, 2008.
[26] 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.
[27] P. Natarajan, F. Baker, P. D. Amer, “Multiple TCP Connections Improve HTTP Throughput Myth or Fact?” in Proc. IEEE 28th Int’l Performance Computing and Communications Conference, Dec. 2009.
[28] K. Seo, S. Shim, J. Y. Lee, T. C. Thang, J. W. Kang, S.-J. Bae, S. Jung, S.T. Parck, “File Format Adaptation Based on HTTP Hint Track for HTTP Streaming,” ISO/IEC JTC1/SC29/WG11 m17772, Geneva, July 2010.

BIOGRAPHIES

Truong Cong Thang received the B.E. degree from Hanoi University of Technology, Vietnam, in 1997 and Ph.D degree from KAIST IT Convergence Campus (KAIST-ICC), Korea, in 2006. From 1997 to 2000, he worked as an engineer in Vietnam Post & Telecom (VNPT). From 2007 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 Assistant Professor of University of Aizu, Japan. His research interests include multimedia networking, image/video processing, content adaptation, IPTV, and MPEG/ITU standards.

Quang-Dung Ho received the B.E. degree from Hochiminh City University of Technology, Vietnam, in 2000, M.S. and Ph.D degrees from KAIST, Korea, in 2003 and 2007. Since 2007, he has been a post-doctoral fellow in the department of Electrical & Computer Engineering, McGill University, Canada. His research interests include new protocols in sensor networks, wireless network optimization, and optical networks.

Jung Won Kang received the BS and MS degree 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.

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. Dr. Pham received Japanese government scholarship (MonbuKagaku-sho) 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.