Multimed Tools Appl (2016) 75:5397–5415
DOI 10.1007/s11042-015-2509-5

Server side, play buffer based quality control for adaptive media streaming

Keunsoo Kim · Benjamin Y. Cho · Won Woo Ro

Received: 1 August 2014 / Revised: 13 December 2014 / Accepted: 10 February 2015 /
Published online: 4 March 2015

Abstract Existing media streaming protocols provide bandwidth adaptation features in order to deliver seamless video streams in an abrupt bandwidth shortage on the networks. For instance, popular HTTP streaming protocols such as HTTP Live Streaming (HLS) and MPEG-DASH are designed to select the most appropriate streaming quality based on client side bandwidth estimation. Unfortunately, controlling the quality at the client side means the effectiveness of the adaptive streaming is not controlled by service providers, and it harms the consistency in quality-of-service. In addition, recent studies show that selecting media quality based on bandwidth estimation may exhibit unstable behavior in certain network conditions. In this paper, we demonstrate that the drawbacks of existing protocols can be overcome with a *server side, buffer based* quality control scheme. Server side quality control solves the service quality problem by eliminating client assistance. Buffer based control scheme eliminates the side effects of bandwidth based stream selection. We achieve this without client assistance by designing a play buffer estimation algorithm. We prototyped the proposed scheme in our streaming service testbed which supports pre-transcoding and live-transcoding of the source media file. Our evaluation results show that the proposed quality control performs very well both in simulated and real environments.

Keywords Adaptive streaming · Quality control · Media streaming · Server side

1 Introduction

In mobile media streaming, media contents are streamed in real time from the remote servers of service providers to the mobile devices of service users. A number of media

K. Kim · B. Y. Cho · W. W. Ro (✉)
School of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea
e-mail: wro@yonsei.ac.kr

K. Kim
e-mail: keunsoo.kim@yonsei.ac.kr

B. Y. Cho
e-mail: youngjcho@yonsei.ac.kr

streaming protocols have been developed to realize such ubiquitous access to media contents. Recently, HTTP streaming protocols such as MPEG-dynamic adaptive streaming over HTTP (MPEG-DASH) [10, 11] or HTTP Live Streaming (HLS) [8] have been recognized as a popular method for media streaming in addition to the existing protocols such as the real-time streaming protocol (RTSP). HTTP streaming implements media streaming on top of the standardized HTTP protocol, which enables effective reuse of the existing infrastructure and optimization techniques designed for the HTTP protocol.

Because mobile devices are exposed to unpredictable network conditions by nature [12], mobile media streaming protocols should support bandwidth adaptation feature that dynamically adjusts the quality of the media stream over time-varying network conditions. Many HTTP streaming protocols employ *client-side*, *bandwidth-based* schemes [8, 10, 11]. The streaming client first estimates the network bandwidth of the link to the server. Based on the estimated bandwidth information, the client selects the most appropriate stream among the multiple quality streams available at the server, which method is often called stream switching.

However, because the quality control is driven by the streaming client, the quality of service may not controllable by service providers due to the variety of client-side bandwidth adaptation policies [1, 9]. In addition, due to the difficulties in correctly estimating current network bandwidth, bandwidth-based quality selection is exposed to the possibility of ineffective and unstable quality selection. A recent study [3] has shown that the stream selection mechanism based on client side bandwidth estimation may trigger unintended quality drop due to the transport-level interference between media streams when the streams share a single network link.

We summarize the challenges addressed in this work in two perspectives. The first problem is client-side service quality dependency that comes from the client-controlled design of quality adaptation. The second problem is unstable quality selection of the existing streaming protocols, which is incurred by incorrectly and unstably estimated network bandwidth. Therefore, the aim of this work is to develop an effective adaptive streaming system that satisfies the following requirements.

- To resolve the client-side service quality dependency problem
- To provide robust quality adaptation in the presence of abrupt bandwidth fluctuation

To this end, we propose an adaptive streaming scheme that fully driven by streaming servers, and a quality adaptation algorithm based on the client-side play buffer state. Firstly, *server-side* quality control alleviate the client dependency problem on service quality by giving full quality control to the streaming server. In addition, when the streaming media is live-transcoded, the server can proactively drop media quality when the playback buffer is about to be empty soon. Secondly, *buffer-based* quality control algorithm enables stable media quality selection under time-varying network condition by considering the playback buffer state of the client directly. This is based on the recent observation that the objective of adaptive streaming is to eliminate playback stall when no media data is available in the client side playback buffer, and the side effects of bandwidth based control schemes are eliminated by considering the playback buffer state [3]. To enable buffer-based quality control at the server side, the playback buffer state of the client should be tracked by the server. To achieve this, we developed a buffer estimation technique that makes the server aware of the playback buffer state without explicit communication between the client and server, which reduces network traffics of state information exchange and imposes no client dependency.

We prototyped the proposed streaming scheme in our testbed. Our experimental results show that the proposed buffer-based quality control algorithm achieves more stable quality selection compared to the bandwidth-based quality adaptation schemes. In addition, we show that the proposed buffer-estimation scheme effectively estimates playback buffer state of the streaming client for both pre-transcoded and live-transcoded media sources. We also evaluated the proposed quality adaptation scheme in real-world 3G network and the result shows the proposed scheme is also effective in minimizing play stall in abruptly changing network environment.

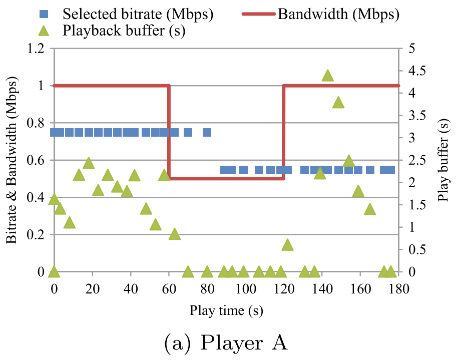
2 Drawbacks of client side, bandwidth based quality control

In this section, we present motivational results for the proposed server-side, buffer-based quality control scheme. We evaluated an adaptive HTTP streaming system in our testbed consisting of a commercial streaming server and two different mobile media players. We used three different media streams with the bitrates of 350 Kbps, 550 Kbps, and 750 Kbps.

2.1 Service quality depends on client side adaptation policy

In this section, we investigate how the existing control policy of client side media players harm the effectiveness of adaptive streaming. Figure 1 illustrates that the play stall (where the buffer level of Player A is equal to 0 seconds) has occurred when the play time is between 60 and 120 seconds. Although the server provides the 350 Kbps stream, of which the bitrate is lower than the lowest bandwidth, the stall is not prevented because the Player A decides to not to switch to the lowest bitrate stream. On the other hand, Player B properly responds to the abrupt bandwidth drop by switching to the lowest bitrate stream. Consequence, playback stall is prevented in this case. In conclusion, service quality (in terms of minimizing playback stall) depends on the policy of client devices instead of the policy of service providers.

This dependency problem has also been investigated in several empirical studies. Akhshabi et al. [1] performed similar measurements for three commercial media players and showed that the players have different characteristics in performing adaptive streaming. Similarly, Riiser et al. [9] investigated four other media players in a 3G broadband environment and reached a similar conclusion.



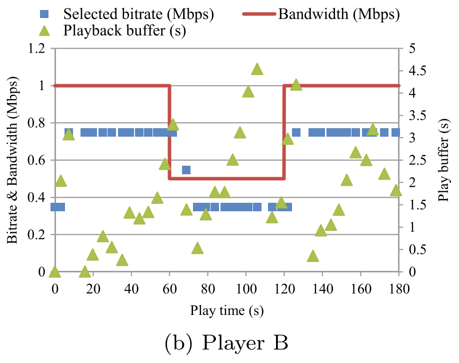


Fig. 1 Response of two different media players for a square-shaped bandwidth change scenario

2.2 Drawback of bandwidth-based quality selection

We investigate the drawbacks and limitations of bandwidth based quality selection algorithms. In general, bandwidth-based quality selection algorithms choose the stream quality based on the estimated average bandwidth during a time period which is calculated by dividing transferred data size by transmission time. However, in HTTP streaming that adopts TCP for the transmission protocol, the throughput varies over time because of the congestion control mechanism of TCP that uses time-varying transmission window (number of packets) to provide fair share of bandwidth when multiple peers share a link. Therefore, when multiple streaming clients share a link, bandwidth-based quality selection algorithm may be disrupted by the competing streams and may result in undesired behaviors. Figure 2 shows an example of such a case. In this experiment, we used a quality control algorithm that selects the streaming bitrate closest to the current estimated bandwidth with 20 % margin. For instance, the client will select 800 Kbps for 1 Mbps bandwidth estimate. Even though the two cases presented in Fig. 2 have 1 Mbps fair share bandwidth, the quality selection of the client shows an unstable behavior. This means the observed bandwidth is not stable, which implies designing a stable quality control algorithm based on the unstable bandwidth information is difficult.

Similar to this result, Huang et al. [3] reported that bandwidth-based adaptation schemes may trigger a negative feedback effect in streamed media quality. Once a client selects a lower quality stream, the transferred media segment size is decreased. The smaller media segment size incurs lower transmission throughput because the TCP transmission window size is re-adjusted for the smaller segment. Therefore, the client will pick even lower quality, which lead to an abrupt drop in service quality.

3 Comparison with related works

Tan et al. [12] showed that the analytical estimation of network bandwidth is difficult to accomplish because of the inherent unpredictability of the mobile network. Therefore, in general TCP rate-control schemes have difficulties in providing a stable behavior for real-time applications that requires transmission delay should be in a predictable range. To improve media delivery performance in that condition, Mehrotra et al. [7] proposed an improved TCP rate-control algorithm for media delivery. Their adaptive rate control

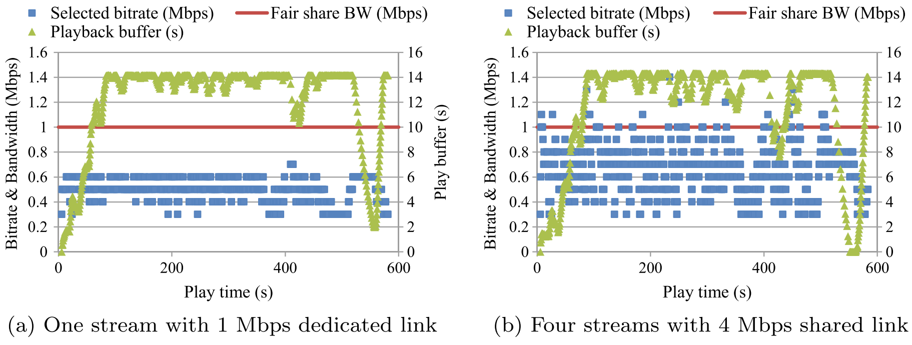


Fig. 2 Instability of bandwidth based quality selection when the shared link is congested by multiple clients

algorithm operates on the transmission layer based on the parameter observation of the channel, while our adaptive streaming mechanism operates on the application layer.

Similar to this work, Liu et al. [5] proposed an application-layer control algorithm that involves the relationship between the fetch and play time of each segment. In our work, compared to their threshold-based algorithm, we have adopted an error-proportional control algorithm that provides a tunable response. In addition, their algorithm is not free from the presented bandwidth sharing problem since the fetch time of each segment is correlated to the bandwidth. In Section 6, we present experimental results that compare their threshold-based control algorithm and the proposed algorithm.

The proposed control algorithm is inspired by control-theoretic approaches for adaptive media quality control [2, 4]. However, we take the perspective that the real system is too complicated to be accurately modeled for applying control theory. Alternatively, we provide a reactive control mechanism in terms of client side playback buffer movement. Furthermore, we focus on important factors for service designers and providers such as performance tunability, buffer estimation techniques for avoiding explicit communication between the client and server, and additionally considered response delay problem which is inherent in live-transcoded media sources.

4 Proposed adaptive streaming system

Figure 3 illustrates the proposed media streaming system. The basic design is similar to the conventional HTTP streaming system. The media content is stored and transmitted as a unit of segment. Each segment represents a piece of the media stream and has distinct properties, such as sequence number, byte size, play time, and bitrate. To provide server-side quality control, only a single media stream is exposed to the client. Therefore, the client has no choice in selecting a media stream. The bandwidth adaptation is controlled by the server when the transmission of each segment is completed. Because we adopted buffer based quality control, the server should be aware of the client buffer state. One possible choice is that the client periodically sends the buffer state to the server; however, this process increases the system complexity and network traffics. Alternatively, we designed a controller to estimate the state of the playback buffer. Based on the estimation, the controller decides the new target media bitrate.

The streaming system can use both pre-transcoded and live-transcoded media sources. With pre-transcoding, source media files are transcoded into multiple qualities and stored in the storage server before the streaming is initiated. When the controller selects a new bitrate for the next media segment, the storage server reads the requested segment and the

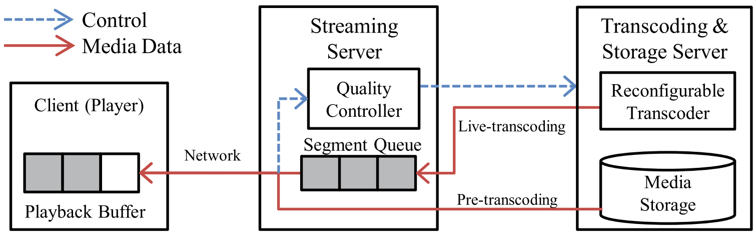


Fig. 3 Organization of the proposed media streaming system

streaming server transmits the segment immediately. Therefore, with pre-transcoding there is no server side response delay time in switching media quality.

With live-transcoding, the transcoding is performed when the streaming is requested by the client. We adopt a reconfigurable transcoder, which performs on-the-fly changing of stream quality based on the target media bitrate assigned by the controller. By re-configuring outgoing bitrate from the transcoder immediately, a responsive quality change is possible without resetting the transcoder. We implemented the live-reconfiguration mechanism based on the rate control algorithm of the H.264 [6] codec. In a group of pictures (GOP) boundary, the bitrate is reconfigured by changing the base quantization parameter (QP), which is a reference value for determining the GOP bitrate. From the next GOP, the rate control algorithm recalibrates the quality of each frame inside the GOP. Note that the transcoder sequentially emits the output media stream and the stream is segmented and queued in the segment queue. Because the segments are delivered sequentially, the client is not able to see the reconfigured bitrate segments until all the previously encoded segments are sent to the client. This server side response delay time for quality change directly affects the required client side play buffer capacity to minimize play stall, since the play buffer continuously decreases until the updated bitrate stream is started to be delivered. In Section 5.5, we further analyze the relationship between the required play buffer level and the server-side response delay.

5 Buffer based quality control

The key objective of bandwidth adaptation is to minimize client device playback stall when the network bandwidth varies or fluctuates over time. The adaptation algorithm is responsible for selecting the media quality to minimize playback stall while maximizing video quality. A popularly used bandwidth adaptation method is stream switching based on bandwidth estimation. Once a media segment is downloaded by a streaming client, the current network bandwidth is estimated by dividing the segment size by the transmission time. The client then switches to an appropriate bitrate stream by selecting a media stream which bitrate does not exceed the estimated bandwidth. Unfortunately, as discussed in Section 2, adaptive quality control based on the bandwidth measurement is quite difficult and sometimes not effective.

To avoid such difficulties, our algorithm is designed to consider the state of client side playback buffer, regarding that the primary objective of adaptive streaming is to ensure that the playback buffer does not become empty. Therefore, compared to the control algorithms based on the estimated current bandwidth, the proposed algorithm prevents potential side effects caused by incorrect bandwidth estimation.

5.1 Relationship between playback buffer and bandwidth

In the HTTP streaming systems, a media stream is separated into multiple media segments. Therefore, the unit for stream delivery is a media segment. For further discussion, we denote the time when k-th segment delivery is completed as T_k, and the play time of segment k as L_k. The playback buffer state at k-th delivery event is denoted as P_k. The amount of media data available in play buffer (*buffer level*) is determined by cumulating the play time of media segments. For instance, if a client has downloaded within seven seconds media segments comprising a total play time of ten seconds, the client buffer is increased by three seconds. To be specific, after k th segment is delivered, the playback buffer state P_k can be represented by the following recurrence relation:

$$P_k = \sum_k (L_k - (T_k - T_{k-1})) \tag{1}$$

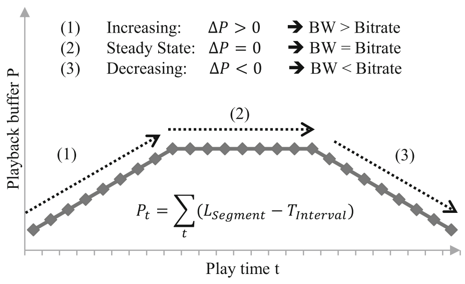
Note that (1) depends on the interval between the segment upload $\Delta_k = T_k - T_{k-1}$. This is important because the interval indirectly reflects the current network condition. Assuming the client downloads the media segment with the best effort, if the network bandwidth is sufficiently larger than the average bitrate of the media segment, then the interval will be shorter than the play time of each segment L_k. As a result, the buffer state P_k will be increased. Therefore, without observing the network bandwidth directly, we can control the media quality effectively by monitoring the variation of available media data in playback buffer.

Figure 4 illustrates how the server awares segment delivery state by observing the changing rate of the client playback buffer state. The changing of the buffer reflects the inflow and outflow of the media stream; the buffer will be in a steady state if the inflow of the media stream to the buffer (limited by network bandwidth) is equal to the outflow from the buffer (limited by media bitrate). This gives the motivation of the proposed control algorithm. When the playback buffer level increases, the current bandwidth may exceed the bitrate of the currently transmitting media segment, which means it is safe to switch to the higher bitrate stream. If the bitrate of the media stream is equal to the current bandwidth, it is expected that the buffer state will be unchanged because the current bitrate is appropriate to be delivered. Finally, when the playback buffer is decreasing due to abrupt bandwidth shortage, the streamed media bitrate should be lowered to prevent playback stall.

5.2 Estimating client playback buffer state at server-side

Note that all information to calculate the current client buffer state with (1) is also available at the server-side, which enables to estimate the client buffer information by the server. This brings some advantages in developing a server-side quality control scheme that requires client-side buffer state. By estimating the client-side play buffer state, additional communication cost to share the buffer information between client and server is eliminated. Furthermore, this enables quality control without assistance of the streaming client.

The server estimates the client-side playback buffer state after a segment is transmitted to the client. In practice, the server can recognize only the completion of copying the segment data into the server side TCP transmission buffer in the OS kernel. Therefore, in the application level, it is difficult to recognize the exact time when the segment data has completely transfered to the client. To avoid this problem, we assume that the client requests the



(1) Increasing: $\Delta P > 0$ ➔ BW > Bitrate
(2) Steady State: $\Delta P = 0$ ➔ BW = Bitrate
(3) Decreasing: $\Delta P < 0$ ➔ BW < Bitrate
Playback buffer P
(1)
(2)
(3)
$P_t = \sum_t (L_{Segment} - T_{Interval})$
Play time t

Fig. 4 Time-varying buffer state reflects bandwidth information

next segment right after it receives a media segment, and the segment request is performed sequentially in playing order. With this assumption, when k-th segment is requested, the server can aware $(k-1)$-th segment is successfully delivered. The delivery time for $(k-1)$-th segment, Δ_{k-1}, is computed by subtracting the timestamps between segment upload initiating events. In this sense, we can implement the subscript-shifted version of (1) to estimate the buffer state.

5.3 Buffer based quality control algorithm

The quality control algorithm determines the amount of bitrate change according to the change of client playback buffer level. An effective bitrate control algorithm should meet two, however contradictory, objectives: responsiveness and stability. For example, if the control algorithm is designed to be highly responsive, the media bitrate can rapidly change after the network bandwidth abruptly drops; i.e., the possibility of a playback stall can be minimized by decreasing the bitrate.

Unfortunately, increasing responsiveness incurs low stability. That is, as the system becomes more responsive over the change of network condition, it becomes less tolerant of noise signals because the system is too sensitive for the input. As a result, the quality of the media stream is not stable. Low stability potentially incurs user inconvenience in a rapidly changing network environment because even an instantaneous network condition change can trigger the rapid change of media stream quality. In other words, with highly responsive systems, if network bandwidth oscillates for some reason (e.g., network congestion), the user will perceive the frequent changing between low and high quality streams. To address the tradeoff between responsiveness and stability, we designed the proposed algorithm to be tunable. Accordingly, the service provider or client can adjust the degree of responsiveness based on their own policy. We provide controlling parameters for this purpose.

Figure 5 presents a pseudo-code of the proposed bitrate control algorithm. The control algorithm adjust the media bitrate when the server initiates transmission of a media segment. Essentially, the proposed algorithm performs error-proportional quality control that follows the change of playback buffer level. To the play time of each media segment to maintain the steady buffer state, the time difference between segment delivery events should be equal. We first compute the error value for k-th segment delivery E_k as follows:

$$E_k = P_k - P_{k-1} \tag{2}$$

Then, the amount of bitrate change for each segment upload event is calculated by multiplying a constant (denoted as a gain G) with the current error value. Unlike conventional error-proportional controllers, we use two separate gain values for increasing and decreasing bitrates. This separation is based on the observation that a rapid bitrate increase causes frequent playback stall in a fluctuating network environment. For example, consider a situation in which network bandwidth is increased sharply, and then reduced to the initial bandwidth (i.e., Low $\rightarrow$ High $\rightarrow$ Low). In the low-to-high transition, the system will quickly increase the media quality if the gain value when increasing the bitrate, G_{Up}, is high. However, as the counter-effect of this increase, in the high-to-low transition, the buffer should decrease faster because the gap between the rapidly increased bitrate and the bandwidth is large. If the buffer level at the beginning of the high-to-low transition is not sufficient, the client playback will stall. This means the gain value when decreasing the bitrate, G_{Down}, should be larger than G_{Up}, to prevent a playback stall in the high-to-low transition. Decreasing the bitrate rapidly makes the system responsive to the abrupt bandwidth reduction. To meet

```
const $G_{Up}$; /* Up gain */
const $G_{Down}$; /* Down gain */

time $P_{k-2}$; /* Estimated buffer at k-2 */
time $P_{k-1}$; /* Estimated buffer at k-1 */
time $L_{k-1}$; /* Last segment length */
time $T_{k-2}$; /* Last upload start time */
bitrate $R$; /* Media quality */

/* called when k-th segment upload is initiated */
on_segment_upload(segment) {
        /* Estimate buffer level */
        $\Delta_{k-1}$ = now() - $T_{k-2}$;
        $P_{new}$ = max(0, $P_{k-2} + (L_{k-1} - \Delta_{k-1})$);
        $P_{k-2}$ = $P_{k-1}$;
        $P_{k-1}$ = $P_{new}$;

        /* Select quality */
        $E_{k-1}$ = $P_{k-2} - P_{k-1}$;

        run_quality_probe();

        if ($E_{k-1}$ > 0) {
                $R$ = $R + G_{Up} \times E_{k-1}$;
        } else if ($E_{k-1}$ == 0) {
                /* Do nothing */
        } else if ($E_{k-1}$ < 0) {
                $R$ = $R + G_{Down} \times E_{k-1}$;
        }

        $T_{k-2}$ = now();
        $L_{k-1}$ = segment.length;
}


const $\lambda$; /* Threshold */
const $\beta$; /* Coefficient */

run_quality_probe() {
        if ($P$ > $\lambda$) {
                $E_{k-1}$ = $E_{k-1} + \beta \times L_{k-1}$;
        }
}
```

Fig. 5 Buffer-following bitrate control algorithm

these two independent tuning objectives, we provide separate parameters to manage the tradeoff between responsiveness and stability.

5.4 Probing optimal quality for clients with finite play buffer capacity

In the proposed algorithm, we focus only on the error value defined as the difference between the buffer inflow and outflow. This means the algorithm focuses only on the change rate of the play buffer, not the amount of the media data left (i.e., buffer level) in the buffer. As shown in Fig. 4, the buffer change rate well reflects bandwidth information if the client always strive to fetch next data segments with the best effort. This also assumes that the client has the play buffer of infinite capacity (or as large as the media file size). However, the capacity is limited in practice. When the play buffer is full, even though with infinite network bandwidth, the segment download interval is equal to the play time of each segment, because the next segment can be transmitted only after the segment in the head is removed from the buffer. Then, assuming the play time of all segments are equal, the error value becomes always less than or equal to zero from (2). This means the server cannot distinguish whether the buffer is full or is in the steady state of Fig. 4. Therefore, without a proper compensation mechanism, switching to the higher media quality cannot happen in this case.

To mitigate this problem, we have designed a simple heuristic control algorithm that considers the amount of data available in the playback buffer. If the buffer level P_k exceeds a pre-defined threshold λ, it increases the estimated error value E_k with the amount of $\beta \times L_k$. In fact, this is equivalent to increase the measured bandwidth with the ratio of β. Let the ratio between play time of a segment and delivery interval is r, as follows:

$$r = \frac{L_k}{T_k - T_{k-1}} \tag{3}$$

Note that r contains the bandwidth information. For instance, when $T_k - T_{k-1}$ is 0.5 s when L_k is 1 s, it means the bandwidth is twice the k-th segment bitrate. After applying the heuristic, r becomes r' equal to the following.

$$r' = \frac{L_k + \beta L_k}{T_k - T_{k-1}} = (1+\beta)\frac{L_k}{T_k - T_{k-1}} = (1+\beta)r \tag{4}$$

From the definition of r, this is equivalent to the case that the estimated bandwidth is increased by the factor of β. Consequently, the control algorithm will try to increase the quality more as if they have sufficient bandwidth for further increasing media bitrate, and the amount of the additional increase can be controlled by adjusting the parameter β.

Some existing bitrate selection algorithms [2, 4] have adopted proportional-integral (PI) controlling schemes to address this issue. In such schemes, the integrator accumulates error values, and the accumulated value is used to compensate steady state error. However, adopting the integrator may complicate the system because careful parameter calibration is required to prevent the possibility of unstable system responses. We believe that the presented method is the better alternative which is easier to implement and tune.

5.5 Analytic model of delayed response with live-transcoded media sources

When the media stream is live-transcoded, some response delay is inevitable due to transcoding and server side segment queuing. It affects the behavior of the proposed quality control mechanism. In this subsection, we model the activity of the control algorithm in terms of the response delay. The main goal of the modeling is to estimate the value of control parameters to meet the given system condition. The estimated values are the starting point in fine-tuning the parameters to optimize the control algorithm performance. For brevity, we introduce only the modeling process for the high-to-low case because it is more important for preventing a play stall.

Figure 6 illustrates an example in which the available bandwidth decreases from $R\alpha$ to R. We denote the ratio between the bandwidths as α. In the figure, each dot indicates an event of segment delivery. We first separate the response of the system for two regions, A and B. Region A indicates the systematic delay, and region B is the result of delay caused by limiting decrease gain G_{Down}. Let the number of segment delivery events in each region be equal to N and M, respectively. First, the minimum buffer level can be calculated by

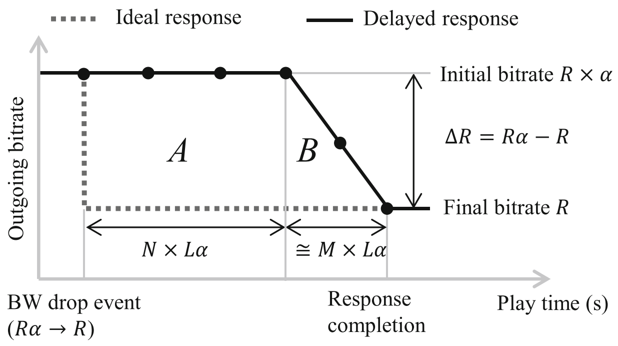


Fig. 6 Response modeling for abrupt bandwidth decrease

subtracting the play time of the delivered segment from the delivery time. Therefore, the required playback buffer level to eliminate play stall is as follows.

$$P > (N \times L\alpha + M \times L) - (N \times L + M \times L) \tag{5}$$

$$\therefore P > NL(\alpha - 1) \tag{6}$$

Secondly, we model the relationship between M and G_{Down} for region B. The decreased bitrate in a delivery event is calculated by multiplying the error value and G_{Down}. In addition, we assume that the amount of decreased bitrate in a delivery event is approximately equal for all events until the control is complete (i.e., the slope of the decreasing bitrate does not change). These two values can be independently calculated, and they must be equal; therefore, the following relationship should be satisfied.

$$(L\alpha - L) \times G_{Down} = \frac{R\alpha - R}{M} \tag{7}$$

$$\therefore G_{Down} = \frac{R}{ML} \tag{8}$$

Finally, we calculate the total response delay in terms of the down gain as follows.

$$NL\alpha + \frac{R}{G_{Down}} \tag{9}$$

By combining (6), (8), and (9), we can estimate system parameters required to meet certain performance conditions. As a concrete example, assume that the bandwidth is decreased from 2 Mbps to 500 Kbps (equal to $R = 500,\ \alpha = 4$), and each media segment is configured to have 1 second of play time ($L = 1$). The system has a response delay corresponding to $N = 3$ and $M = 2$. Then, from (6) and (8), the calculated results of the required buffer level is $P = 9$ seconds and $G_{Down} = 250$.

6 Evaluation

6.1 Methodology

We have developed the proposed adaptive streaming system testbed with a prototype streaming server and clients. In our testbed, four different clients are serviced by one streaming server. We use a source media file encoded to the H.264 Main profile and AAC audio codec with a resolution of 720p. The media stream is transcoded to H.264 Baseline profile with 360p resolution. For pre-transcoding, we prepared 13 different streams, of which the bitrates are from 300 Kbps to 1500 Kbps. For live transcoding, we use variable bitrate from 200 Kbps to 2 Mbps, which is reconfigured on-the-fly. The media segment play time is configured to be 1 second. We configured the capacity of client-side play buffer as 15 seconds.

We simulated time-varying bandwidth in our testbed by configuring the uplink bandwidth limit of the server dynamically. We configured two bandwidth changing scenarios. In the *Square* scenario, the bandwidth initially maintains 6 Mbps, abruptly drop to 2 Mbps, and quickly recover to the original value. In the *Sawtooth* scenario, the bandwidth is rapidly changed with the sawtooth pattern of 20 second period. Note that in our setting four clients are serviced at the same time, therefor the fair share bandwidth for each client is up to 1.5 Mbps and down to 500 Kbps for each pattern. To correctly capture the statistical characteristics, all experiments performed in our testbed are repeated at least five times for each scenario.

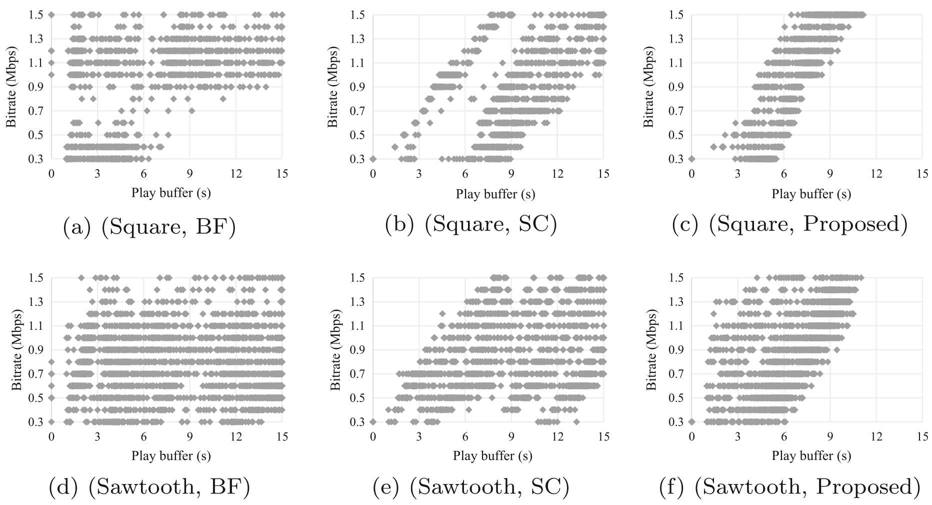
6.2 Bandwidth *vs*. Buffer based control

In this subsection, we compare the characteristics of bandwidth based and buffer based quality control schemes. In this experiment, only pre-transcoding media sources are used to exclude the effect of server side response delay. We compare the behavior of three adaptive quality control schemes.

– *BF*: A bandwidth-following control algorithm. When a media segment is delivered, the current bandwidth is estimated by dividing the segment size by the transmission time. New stream quality is selected as the maximum available bitrate that does not exceed 20 % lower than the estimated bandwidth.
– *SC*: A step-wise control algorithm presented in [5]. When a media segment is delivered, the fetch throughput is calculated by dividing the play time of the segment by the transmission time. The quality is stepped-up when the currently measured throughput exceeds the predefined threshold ϵ, and stepped-down when the throughput is less than γ. Following the presented methodology, we used $\epsilon = 1.3$ and $\gamma = 0.8$ for our media streams.
– *Proposed*: The proposed control algorithm. We used the parameter of $G_{Up} = 150$, $G_{Down} = 150$, $\lambda = 5$, and $\beta = 0.5$.

Figure 7 shows the segment delivery pattern for each scheme under the two bandwidth variation scenarios. Each dot in Fig. 7 represents a segment delivery event. The X-axis of the figure is the play buffer level of the client when a segment is delivered, and the Y-axis is the quality of the segment. With the *BF* algorithm, the play buffer varies significantly between 0 and 15 seconds. In the *Square* scenario, even though the bandwidth abruptly changes twice (6 Mbps $\rightarrow$ 2 Mbps $\rightarrow$ 6 Mbps), the bandwidth in each section is configured not to change. However, the play buffer trend shows a random-like pattern. This suggests that the streaming quality is unstable because of the incorrectly estimated bandwidth, as discussed in Section 2. Consequently, the media bitrate fluctuates abruptly. Meanwhile, because the *SC* algorithm switches the quality in a stepped manner, the bitrate is lowered gradually after a network bandwidth drop has occurred, along with the decrease of play buffer level. Therefore, the *SC* algorithm performs the quality adaptation much smoother than the *BF* algorithm.

However, compared to the proposed algorithm, *SC* shows a larger deviation in play buffer level. For quantitative comparison, Table 1 summarizes the average bitrate, average (μ) and



(a) (Square, BF) (b) (Square, SC) (c) (Square, Proposed)

(d) (Sawtooth, BF) (e) (Sawtooth, SC) (f) (Sawtooth, Proposed)

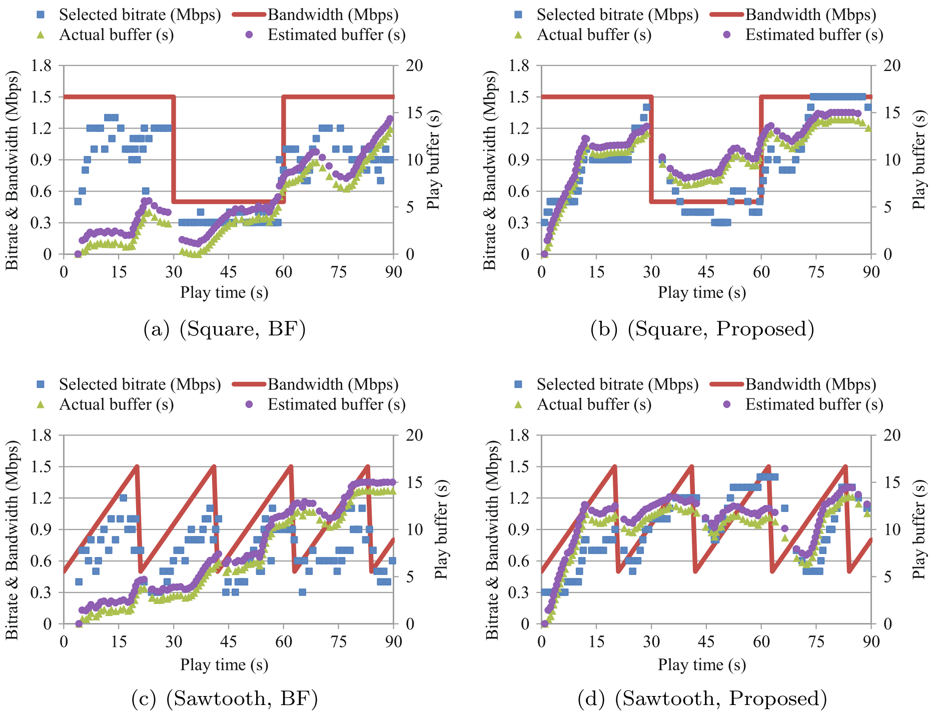
Fig. 7 Segment delivery pattern of various quality control algorithms (*Scenario*, *Algorithm*)

standard deviation (σ) of play buffer level, and the maximum play stall time of the evaluated algorithms. In terms of the quality of delivered media stream, in both scenarios, the proposed algorithm shows the highest media bitrate amongst all algorithms. In terms of minimizing play stall, we observed that all algorithms are quite effective. However, in particular, the proposed algorithm achieves similar maximum play stall time compared to *SC* with the less amount of media segments in the play buffer. In the *Square* scenario, the average play buffer is 9.4 s with *SC* and 6.8 s with the proposed algorithm. If we can provide a similar play stall prevention ability, it is better to keep the play buffer level as low as possible, because it saves client-side memory space and potentially reduces content delivery cost for adaptive streaming. For example, the client needs not to maintain the play buffer level high by pre-loading lots of media segments. In this sense, the proposed algorithm outperforms *BF* and *SC*, since it shows the lower play buffer average. In terms of the deviation of play buffer level, the proposed algorithm shows the lowest deviation; which means the proposed algorithm is able to maintain the play buffer more stable than the two other algorithms.

Finally, we compare *BF* and the proposed algorithm to analyze why the buffer-based quality control is more effective. We selected *BF* since its quality changing trend directly

Table 1 Summary of performance for compared algorithms

Scenario	Algorithm	Bitrate (μ)	Max. Play stall	Play buffer (μ, σ)
Square	BF	905 kbps	1.5 s	(5.9 s, 3.9)
	SC	932 kbps	0 s	(9.4 s, 3.6)
	Proposed	990 kbps	0 s	(6.8 s, 2.1)
Sawtooth	BF	808 kbps	0.17 s	(8.4 s, 4.7)
	SC	825 kbps	0.67 s	(8.8 s, 4.0)
	Proposed	878 kbps	0.62 s	(6.3 s, 2.4)



(a) (Square, BF)

(b) (Square, Proposed)

(c) (Sawtooth, BF)

(d) (Sawtooth, Proposed)

Fig. 8 Comparison of bandwidth based and buffer based control scheme with two different bandwidth changing scenarios

reflects the current bandwidth observed by the client. Figure 8 illustrates the play buffer and quality selection trends of the *BF* and the proposed algorithm. The figure presents the fair share bandwidth for a streaming client, the selected bitrate, and the estimated and actual playback buffer level. If the playback buffer level reaches zero, a playback stall occurs and it must be avoided. First, the results show that the proposed buffer estimation scheme is effective for the both scenarios. This suggest that the buffer based control scheme can effectively control the media quality at the server side without client assistance with the estimated play buffer information. We also observe that both control schemes react to the abrupt bandwidth change by selecting the lower bitrate stream. However, with the proposed buffer based scheme the movement of play buffer and stream selection are correlated while the bandwidth based scheme is not. This is easily understandable since the proposed control scheme is designed as a play buffer follower. In bandwidth based schemes the estimated bandwidth values, which are inputs of quality control, are discontinuous and it makes quality streams unstable. To alleviate this problem, we can adopt filtering algorithms to suppress the impact of noise signals in bandwidth measurements [3]. However, this is an ad-hoc solution; the observed bandwidth is inherently unstable since it is tightly coupled to the transport (TCP) level congestion control mechanism, which distributively allocates the bandwidth of the shared link by managing TCP transmission window size dynamically. Meanwhile, the proposed buffer based scheme enables smoother control and delivers better service quality, because the change of play buffer is inherently continuous since the level is changed gradually as the streamed media is played.

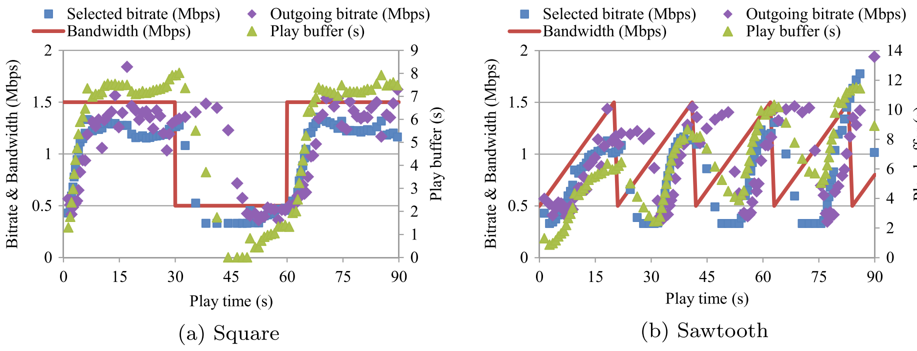


Fig. 9 Server side response delay with on-the-fly quality reconfiguration

6.3 Evaluation with live-transcoded media sources

In this subsection, we evaluate the effectiveness of the proposed algorithm in live-transcoded environment, where the transcoded media quality is reconfigured on-the-fly. Figure 9 shows the response of the proposed control scheme after an abrupt bandwidth drop scenario. The figure shows that the algorithm can properly detect and select the lower quality stream immediately. However, because of the system response delay, we can observe that the bitrates of the outgoing segments are lagged behind the selected bitrates. The delay is approximately 10 seconds. In the square-shaped bandwidth changing scenario, until the quality changed segments are delivered to the client, the buffer level is decreased. The client experiences an instant play stall around 45 seconds, and it begins to recover the play buffer.

We investigate the relationship between response delay and system parameters. Specifically, we focus on two parameters that dominate the delay. The first parameter is the length of the segment queue. As presented in Fig. 3, the segment queue is used to store the media segments after transcoding and the client can see the list of media segments in the queue. The queue hides the latency of transcoding server and helps smooth segment delivery. However, increasing queue length also increases the response delay because the user will not perceive the bitrate change until all other segments already in the queue are delivered to client. Figure 10a shows the response delay change while varying the segment queue size. In this experiment, we use a very large value for G_{Down} to exclude the effect of the parameter. To calculate the modeled response time from (9), it is required to know the value of N. We approximate the value of N from the result given in Fig. 10c; we set N as 2 + Segment queue length, since Fig. 10c shows the outgoing bitrate starts to fall after 5 segments have

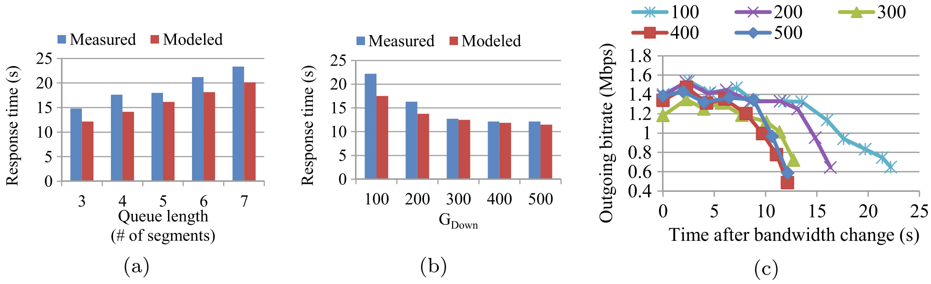


Fig. 10 Sensitivity of system parameters and comparison with modeled values

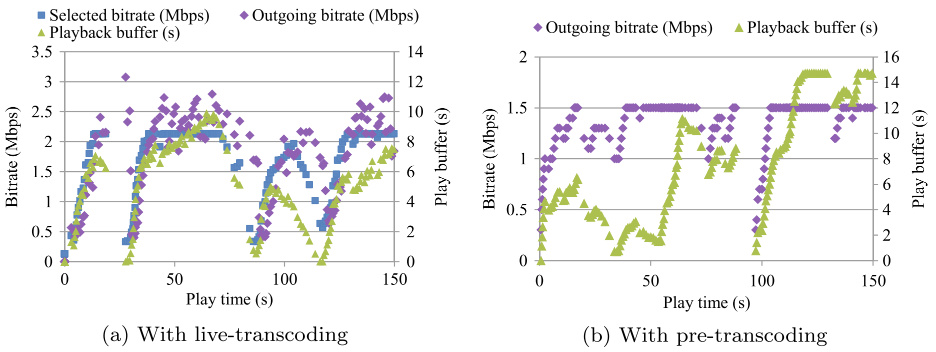


Fig. 11 Response in real 3G network environment

delivered when the queue length is set to 3. The response delay shown in Fig. 10a linearly increases as the segment queue length increased. This is because one additional queue entry indicates one additional segment should be delivered before the client perceives the updated bitrate.

The second parameter is G_{Down}, which is the sensitivity parameter of the control algorithm. As G_{Down} increases, the response time is reduced because the system reacts more sensitively to the bandwidth change. Figure 10b and c present the effect of G_{Down}. As Fig. 10b shows, the delay is decreased as G_{Down} increased. However, the response time is saturated around the G_{Down} value of 300, since other factors such as queue length dominates the delay from this point. Figure 10c further illustrates the response characteristics of the proposed system with different G_{Down} values. The result shows that it is difficult to reduce the response time even if the larger value of G_{Down} is used. We can observe that the response time is saturated around 10 seconds, a time that is approximately equal to the delay corresponding to the segment queue of length 3. Therefore, to set the delay time below this value, reducing segment queue length is important; however, it is currently not easy to reduce the delay beyond this point because some streaming protocols require the server to provide at least 3 segments [8].

Overall, the modeling results preserve the characteristics of the measurement results. However, the modeled values underestimate the controlling delay because of the approximations used in the modeling process. The errors are −15.3 % and −9.3 % in Fig. 10a and b, respectively. We believe that the error values are in tolerable range for the purpose of deciding approximate values of the control parameters that meet desired performance conditions.

6.4 Evaluation in real-world mobile environment

To verify the operation of the proposed control scheme in a real-world environment, we collected the video streaming behaviors on a smartphone connected to a 3G broadband network. Figure 11a shows quality selection behavior with the live-transcoding configuration. The media bitrate is changed along with the time-varying movement of the play buffer level. Even though there is a delay time because of the systematic response delay, the streaming client experiences only instantaneous play stall. We can observe a similar trend in Fig. 11b, where the outgoing bitrate of the next media segment is switched instantaneously. In conclusion, in such a volatile network environment, the proposed scheme controls media quality effectively and minimizes playback stall under abrupt bandwidth shortage.

7 Conclusion

In this paper, we have focused on the drawbacks of the existing adaptive streaming systems. First, we observe that the effectiveness of adaptive streaming depends on the client-side quality adaptation policy, which potentially deliver low service quality. We have adopted a server-side quality control scheme to resolve the client-side quality dependency problem.

Second, recent studies have shown that network bandwidth is difficult to be estimated correctly at client-side due to transport-level interferences among streams. We observed this incorrect bandwidth estimation makes bandwidth-based control algorithms ineffective. Alternatively, we propose a buffer based control algorithm which adjusts media quality based on the estimation on the client-side playback buffer state at the server side.

The evaluation results in simulated and real environments show that the proposed quality control mechanism is effective. As future works, we will focus on further reducing response delay of the adaptive control scheme to improve effectiveness even in severely fluctuating network environment.

Acknowledgments This research was supported in part by the MKE (The Ministry of Knowledge Economy) and NHN Corp., under IT/SW Creative research program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2012-H0505-12-1011), and in part by the ICT R&D program of MSIP/IITP [2014(10041971), Development of Power-efficient High-performance Multimedia Contents Service Technology using Context-adapting Distributed Transcoding]. Won Woo Ro is the corresponding author.

References

1. Akhshabi S, Begen AC, Dovrolis C (2011) An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In: Proceedings of the 2nd annual ACM conference on multimedia systems, pp 157–168. ACM
2. De Cicco L, Mascolo S, Palmisano V (2011) Feedback control for adaptive live video streaming. In: Proceedings of the 2nd annual ACM conference on Multimedia systems, pp 145–156. ACM
3. Huang TY, Handigol N, Heller B, McKeown N, Johari R (2012) Confused, timid, and unstable: picking a video streaming rate is hard. In: Proceedings of the 2012 ACM conference on Internet measurement conference, pp 225–238. ACM
4. Huang Y, Mao S, Midkiff SF (2009) A control-theoretic approach to rate control for streaming videos. IEEE Trans Multimed 11(6):1072–1081
5. Liu C, Bouazizi I, Gabbouj M (2011) Rate adaptation for adaptive http streaming. In: Proceedings of the 2nd annual ACM conference on multimedia systems, MMSys '11. ACM, New York, pp 169–174. doi:10.1145/1943552.1943575
6. Ma S, Gao W, Lu Y (2005) Rate-distortion analysis for h. 264/avc video coding and its application to rate control. IEEE Trans Circ Syst Video Technol 15(12):1533–1544
7. Mehrotra S, Chen H, Jain S, Li J, Li B, Chen M (2012) Bandwidth management for mobile media delivery. In: Global communications conference (GLOBECOM) 2012 IEEE, pp 1901–1907. IEEE
8. Pantos R, May W (2013) Http live streaming. IETF Draft
9. Riiser H, Bergsaker HS, Vigmostad P, Halvorsen P, Griwodz C (2012) A comparison of quality scheduling in commercial adaptive http streaming solutions on a 3g network. In: Proceedings of the 4th workshop on mobile video, pp 25–30. ACM
10. Sodagar I (2011) The mpeg-dash standard for multimedia streaming over the internet. IEEE Multimedia 18(4):62–67. doi:10.1109/MMUL.2011.71
11. Stockhammer T (2011) Dynamic adaptive streaming over http–: standards and design principles. In: Proceedings of the 2nd annual ACM conference on multimedia systems, pp 133–144. ACM
12. Tan WL, Lam F, Lau WC (2008) An empirical study on the capacity and performance of 3g networks. IEEE Trans Mob Comput 7(6):737–750

Keunsoo Kim received his B.S. degree in Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2012.

He is currently a Ph.D. student in Embedded Systems and Computer Architecture Laboratory, School of Electrical and Electronic Engineering, Yonsei University. His industry experience includes a college internship at NAVER Corp. His research interests include computer architectures and network applications.

Benjamin Youngjae Cho received his B.S. and M.S. degrees in Electrical & Electronic Engineering department at Yonsei University, Seoul, Korea, in 2012 and 2014, respectively.

He is currently pursuing Ph.D. degree in the Electrical and Computer Engineering department at University of Texas at Austin. His industry experience includes a college internship at NAVER Corp. His current research interests lie in high performance computing and multiple aspects of memory system designs in large-scale computing including heterogeneous memory systems, 3D-stacked DRAM, and DRAM cache.

Won Woo Ro received the B.S. degree in electrical engineering from Yonsei University, Seoul, Korea, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California in 1999 and 2004, respectively.

He worked as a research scientist in the Electrical Engineering and Computer Science Department, University of California, Irvine. He currently works as an Associate Professor in the School of Electrical and Electronic Engineering at Yonsei University. Prior to joining Yonsei University, he has worked as an Assistant Professor in the Department of Electrical and Computer Engineering at California State University, Northridge. His industry experience also includes a college internship at Apple Computer, Inc., and a contract software engineer in ARM, Inc. His current research interests are high-performance microprocessor design, compiler optimization, and embedded system designs. He is a member of the IEEE.