Design and Implementation of a Network-Adaptive Mechanism for HTTP Video Streaming

Yo-Han Kim, Jitae Shin, and Jiho Park

This paper proposes a network-adaptive mechanism for HTTP-based video streaming over wireless/mobile networks. To provide adaptive video streaming over wireless/mobile networks, the proposed mechanism consists of a throughput estimation scheme in the time-variant wireless network environment and a video rate selection algorithm used to increase the streaming quality. The adaptive video streaming system with proposed modules is implemented using an open source multimedia framework and is validated over emulated wireless/mobile networks. The emulator helps to model and emulate network conditions based on data collected from actual experiments. The experiment results show that the proposed mechanism provides higher video quality than the existing system provides and a rate of video streaming almost void of freezing.

Keywords: HTTP video streaming, network adaptive.

Manuscript received Dec.13, 2011; revised May 20, 2012; accepted Aug. 22, 2012.

This work was supported by the R&D program of the Ministry of Knowledge Economy (MKE), Korea entitled "Study on Ubiquitous Infra, Core Technology on Enhanced QoS/QoE."

Yo-Han Kim (phone: +82 31 290 7153, dos95@skku.edu), Jitae Shin (corresponding author, jtshin@skku.edu), and Jiho Park (jihopark@skku.edu) are with the College of Information and Communication Engineering, Sungkyunkwan University, Suwon, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.13.0111.0788

I. Introduction

User demand has increased for multimedia services within wireless networks. HTTP-based video streaming [1] has many benefits due to the use of HTTP/TCP/IP protocol, which is simple and easy to implement, has a low computation server load, and does not cause connection problems for network address translation (NAT)/firewall-applied networks. Moreover, with HTTP-based protocols, the network load can be reduced through caching or proxy technique. Research on HTTP-based streaming is actively continuing with consideration for wireless/mobile environments. Wi-Fi and 3G systems are used widely, so multimedia streaming techniques for mobile networks need to be improved. Adaptive streaming and lightweight network estimation techniques should be developed within wireless networks to address limited mobile resources and time-varying channel conditions.

This work proposes and implements a network-adaptive HTTP video streaming system over Wi-Fi and 3G mobile networks. The streaming system adopts the version of HTTP live streaming from the Internet Engineering Task Force (IETF) [2]. Additionally, the proposed network-adaptive mechanism, consisting of throughput estimation and adaptive video rate selection, is enhanced to ensure quality of service (QoS) as well as improve efficiency.

II. Existing HTTP Video Streaming System

In [2], they proposed and modified an HTTP live streaming method as an IETF Internet-Draft, in which a server segments a single contiguous stream as segmented media files according to a constant period and generates a metafile (that is, m3u8 playlist file) having the information of the media files. After a client receives the metafile and identifies the media structure, it sequentially requests the appropriate media file upon the condition of the client's buffer and channel, decodes the file, and then plays the media for the user. This protocol is compatible with that of the general HTTP web server. Therefore, only minor modification is necessary to use the metafile.

A smooth streaming [3] embodies adaptive streaming features in the web server. This method can provide sufficient video quality according to the communication channel conditions so as not to hinder continuous video playback during video streaming service. This scheme is more efficient during a server's content management because a whole stream is put into one stream container, which has more information and is more segmented than HTTP live streaming. However, the server reanalyzes the stream container and then corresponds individually to a client's request. This is required to change the structure of the server to support these actions. Additionally, a client needs the corresponding video player for the different structure, which is another weak point.

In the standard body of the 3rd Generation Partnership Project (3GPP), the related schemes are standardized and referred to as adaptive HTTP streaming (AHS) [4]. 3GPP AHS adopts similar server-driven structures, such as the HTTP live streaming that stores the metafile and media streams on the server side and sends the data through HTTP upon a client's request. However, it has features that expand the form and option of the metafile and reduce the overhead via header compression. Recently, the MPEG group standardized dynamic adaptive streaming over HTTP (DASH) [5] as a major transport format. DASH is based on 3GPP AHS but technically improved.

Compared to HTTP live streaming, DASH has detailed but complex metainformation. Even though various kinds of adaptive streaming are adoptable, the server and client must support DASH. In contrast, HTTP live streaming has a simple metafile, and the list of video levels is fixed and unchangeable once streaming has started. HTTP live streaming is available with a general web server and is easy to implement, but managing streaming data is difficult because all the segmented streams and metadata must be stored in separate files. Therefore, the server must handle large-scale file management as well.

This paper is based on the structure of HTTP live streaming to embody and validate the proposed network-adaptive mechanism because the large-scale media file management is out of this paper's scope. The proposed schemes of throughput estimation and video rate selection are applicable to any streaming system structure.

There are two main ways to attain available throughput estimation for the adaptive streaming system: active monitoring and passive monitoring. Additionally, the cross-layer approach uses information from several other layers. The probe packet method, which is an active monitoring method, uses additional packets to measure the available throughput. The self-loading periodic streams method [6] uses the increasing trend of periodic packet delay when a stream's rate is higher than the available throughput. The CapProbe in [7] included convergence tests and convergence speed-up techniques by varying probing parameters. It combined the delay as well as the dispersion measurements of packet pairs to filter out the samples distorted by cross traffic. The Wbest in [8] utilized a packet pair and a packet train technique to estimate effective capacity and achievable throughput for 802.11 wireless networks.

In the passive monitoring, an exponentially weighted moving average (EWMA) with static coefficients was used to apply it to general networks [9]. A bandwidth estimation scheme over WLAN networks [10] uses success or failure rates and average packet sizes to calculate the available bandwidth. This scheme focuses to reflect characteristics of the specific networks, so it is not suitable for a general network structure combining wired and wireless networks.

In this paper, a passive monitoring method based on EWMA is proposed for both reducing redundant probe packets and adopting a time-varying network condition.

III. Proposed System

1. Implemented Video Streaming System

The proposed system structure shown in Fig. 1 consists of a general HTTP server and a client with network-adaptive modules based on HTTP live streaming. The implementation of the proposed system uses Gstreamer [11] as an open source multimedia framework that provides the plug-in-based functions for multimedia streaming service, such as format conversion, image resizing, encoding and decoding, and other expanded plug-ins for users' own data types. The server generates several segmented streams (that is, transport streams [TSs]) and a related metafile (that is, m3u8 file) for adaptive streaming. The server sends the data via HTTP, which is requested by the client. The client requests the optimized bitrate of a proper stream to the server using its network information and the client's buffer conditions. For video streaming, the client requests the metafile, and the received metafile is analyzed by the metafile parser in the client. Extracted information from the metafile is stored in the playlist buffer, shown in Fig. 1. The client requests a proper stream of the optimized bitrate to the server using its network information

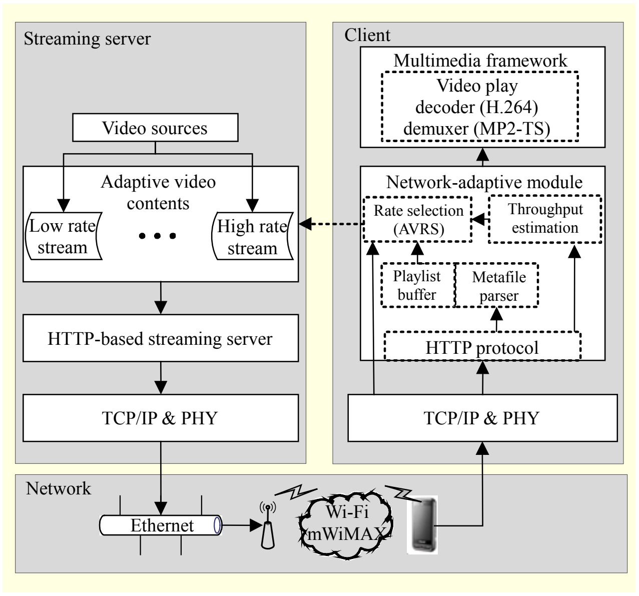


Fig. 1. Overall structure of implemented video streaming system.

and buffer conditions. The playlist buffer consists of a classified table of segmented stream URIs that is sorted by time and rate.

The newly added throughput estimation module in Fig. 1, described in subsection III.2, estimates the current network conditions with sudden change. However, the estimation error still remains big in cases of abruptly changed conditions. To cope with the estimation error, a new rate selection module, described in subsection III.3, is added to select a properly segmented stream based on the current buffer information and network conditions estimated by the throughput estimation module.

2. Advanced Throughput Estimation

Firstly, we explain a throughput estimation method [9] based on EWMA with a constant weight coefficient. It is considered for the referenced throughput estimation of a time-varying network. It is assumed that the available channel throughput changes linearly. Specifically, the current changing trend of channel throughput is similar to that of the previous channel's throughput. The available throughput of the next time slot is estimated using all the previous throughput data with the most recent proportional weights. The EWMA scheme that halves the coefficient of weights is as follows:

$$\begin{aligned} d_n &= T_n - T_{n-1}, \\ D_n &= \frac{1}{2}D_{n-1} + \frac{1}{2}d_n, \\ \hat{T}_{n+1} &= T_n + D_n, \end{aligned} \tag{1}$$

where T_n is the measured throughput at the n-th time and D_n is the estimation of throughput difference between n and (n–1) times. The estimated throughput of the (n+1) time is $\hat{T}_{n+1}$. From these steps, the moving average weights of the previous estimated throughputs are decreased on the order of 1/2, 1/4, 1/8, and so on.

The proposed throughput estimation algorithm using the dynamic fluctuation index (DFI) mechanism improves the EWMA scheme by reflecting the degrees of fluctuation of the throughput difference. The DFI concept checks the change in measured throughputs, d_n, over time and dynamically updates the weight to estimate D_n, while the EWMA scheme uses a static weighting value of 1/2. To reflect the network condition for each time, a fluctuation index (*FI*) is introduced to determine the difference between the measured and estimated throughputs and is defined as follows:

$$FI_n = \left| T_n - \hat{T}_n \right|. \tag{2}$$

The *FI* enables verification of whether the network condition has changed within a predictable range. If the absolute value of the *FI* is greater than an error bound, *e*, the network condition deviates from the estimated trend and changes rapidly.

The DFI mechanism consists of the following three main parts as shown in Fig. 2: (Step 1) data acquisition from the application layer and calculation of basic parameters; (Step 2) determination of the moving average weight from *FI*; and (Step 3) throughput estimation for the next time interval.

The role of each step is described as follows. Step 1 calculates (or measures) the current throughput from the received data packets. The packet sizes and elapsed times are used. Using this information, the change in calculated throughputs over time and the *FI* are obtained. Step 2 involves the determination of the moving average weight, α, used to estimate the next-time throughput based on the *FI* value. The absolute value of *FI* is used to check the changing status of the network condition as follows. If the absolute value of *FI* is less than error bound, *e*, then the network condition does not change as quickly and is predictable via previously calculated values and the weight, α, can be reduced. Otherwise, α is increased while providing more weight to recently calculated throughputs because the network conditions change fast. Step 3 is performed to estimate the next-time throughput ($\hat{T}_{n+1}$) via (3) and uses the items determined via Steps 1 and 2.

$$\begin{aligned} D_n &= (1-\alpha_n)D_{n-1} + \alpha_n d_n, \ 0 \le \alpha_n \le 1, \\ \alpha_n &= \begin{cases} \alpha_{n-1}(1-\varepsilon), \ FI \le e, \\ \alpha_{n-1}(1+\varepsilon), \ FI > e, \end{cases} \\ e &= cT_n, \end{aligned} \tag{3}$$

where c is a constant with $0 \le c \le 1$. This DFI mechanism updates α dynamically, according to the network conditions,

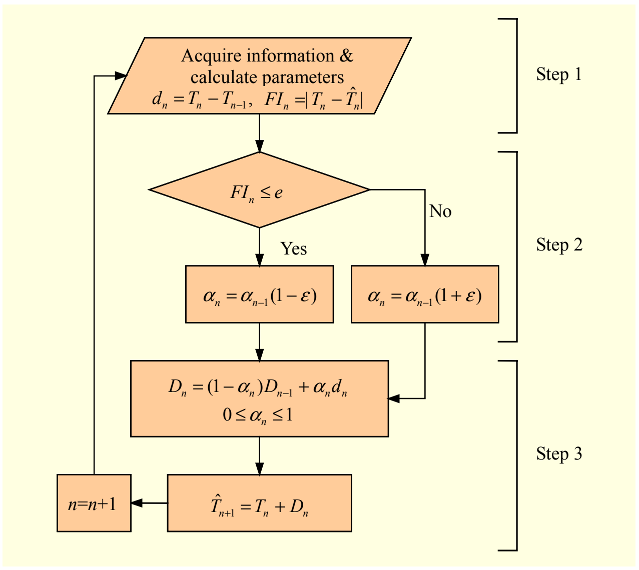

Acquire information & calculate parameters
$d_n = T_n - T_{n-1}$, $FI_n = |T_n - \hat{T}_n|$
$FI_n \le e$
No
Yes
$\alpha_n = \alpha_{n-1}(1-\varepsilon)$
$\alpha_n = \alpha_{n-1}(1+\varepsilon)$
$D_n = (1-\alpha_n)D_{n-1} + \alpha_n d_n$
$0 \le \alpha_n \le 1$
$n=n+1$
$\hat{T}_{n+1} = T_n + D_n$
Step 1
Step 2
Step 3

Fig. 2. Proposed DFI throughput estimation mechanism.

and it operates constantly whenever packets arrive. Then, it estimates the available network-adaptive throughput.

3. Adaptive Video Rate Selection

The different video qualities of a segmented sequence (or stream) on the available throughput are considered. The different qualities of the same video sequence are stored in a video sever, as shown in Fig. 1, and the corresponding bitrates are R_{Q1}, R_{Q2}, …, R_{Qm} for m different video qualities.

A video segment (or stream) among the different qualities tagged as Q_1, Q_2, ..., Q_m is selected with a rate that is high but less than the estimated throughput. This is done using (4).

$$\begin{aligned}&\arg\max_{R_{n+1}}\{R_{n+1} \le \hat{T}_{n+1}\},\\ &R_{n+1} \in \{R_{Q1}, R_{Q2}, \ldots\ldots, R_{Qm}\},\end{aligned} \tag{4}$$

where R_{n+1} is the rate of the (n+1)th segment corresponding to different video qualities and R_{Q1}, R_{Q2}, and R_{Qm} respectively represent the required throughput for stream Q_1, Q_2, and Q_m. When the (n+1)th segment is required to send, $\hat{T}_{n+1}$ is the estimated throughput.

Since HTTP protocol uses TCP over IP as a best-effort network service, if the estimated throughput is less than the measured one, the video segment is received before the anticipated arrival time. In this case, the next video segment is received consecutively, and the receiver's buffer can overflow because the incoming rate exceeds the consumed rate of the decoder and player. Therefore, the receiver needs to be on standby to prevent overflow, which decreases efficiency.

In general, a rate-selection scheme performs buffer status monitoring to avoid buffer overflow. This paper proposes an

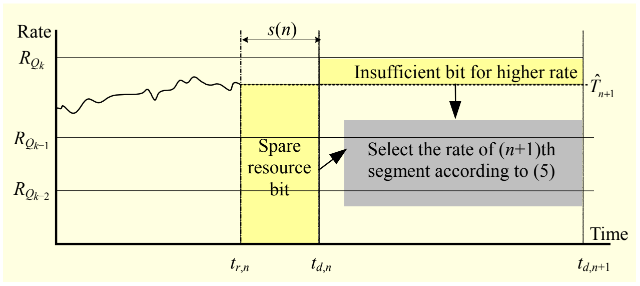


Fig. 3. Proposed AVRS algorithm.

adaptive video rate selection (AVRS) algorithm that uses the HTTP transmission information (that is, 200 OK status code), the duration, and the rate of each video segment. Figure 3 illustrates the concept of the proposed AVRS algorithm.

Let the current segment n be actually received at time $t_{r,n}$ and the given anticipated arrival time for the received n-th segment be $t_{d,n}$. Then, the spare time $s(n)$ remains as ($t_{d,n} - t_{r,n}$). Next, the additionally addable network capacity for the (n+1)th segment is $s(n) \times \hat{T}_{n+1}$. Therefore, available capacity of the next (n+1)th segment is $(\Delta t_{d,n+1} + s(n)) \times \hat{T}_{n+1}$, where $\Delta t_{d,n+1}$ is ($t_{d,n+1} - t_{d,n}$) and the acceptable rate, R_{n+1}, for the next time (n+1) is selected by

$$\begin{aligned}&\arg\max_{R_{n+1}}\{R_{n+1} \times \Delta t_{d,n+1} \le \hat{T}_{n+1} \times (\Delta t_{d,n+1} + s(n))\},\\ &R_{n+1} \in \{R_{Q1}, R_{Q2}, \ldots\ldots, R_{Qm}\}.\end{aligned} \tag{5}$$

IV. Performance Evaluation

1. Experimental Environments

Wireless networks are measured with several different conditions for the performance evaluation using the off-the-shelf software IP Traffic Test & Measure [12]. Based on these experiments, typical network conditions are profiled and then applied in the NetDisturb emulator [13], which is used to illustrate performance comparisons among the proposed mechanism and others. The networks used for the experiment are Wi-Fi (IEEE 802.11g) and mobile WiMAX (IEEE 802.16e), and a schematic diagram of the network measurement is shown in Fig. 4. A traffic measurement tool [12] is used to measure network conditions, which generates HTTP packets and forwards them to a receiver over networks to measure throughput, delay, and loss rate over time.

The network conditions of suburban and downtown areas are measured on the subway while in motion. The measured time zones are morning, noon, rush hour, and night. The locations are diverse places, such as the suburbs, the subway, a downtown mall, and a downtown movie theater. The rapidly changing network profiles are selected for the test because

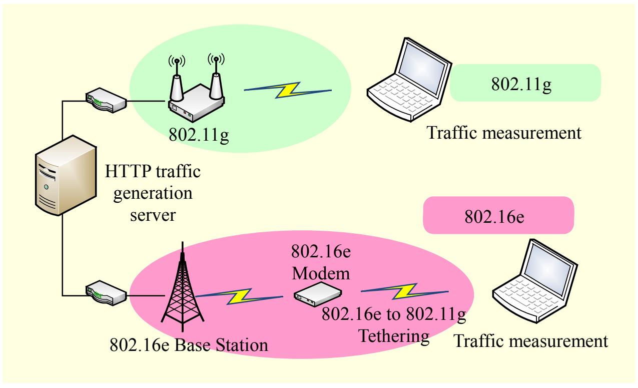


Fig. 4. Throughput measurement experiments over actual networks.

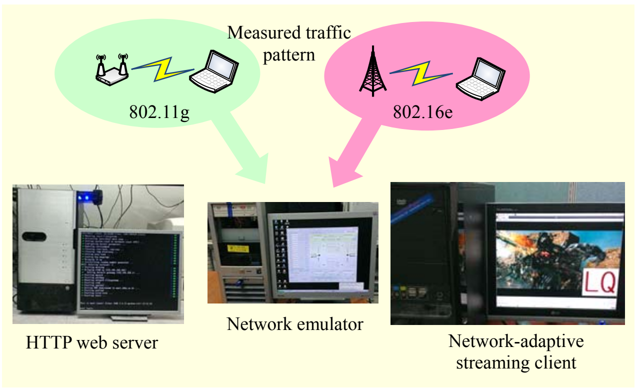


Fig. 6. Setup of video streaming experiment with network emulator.

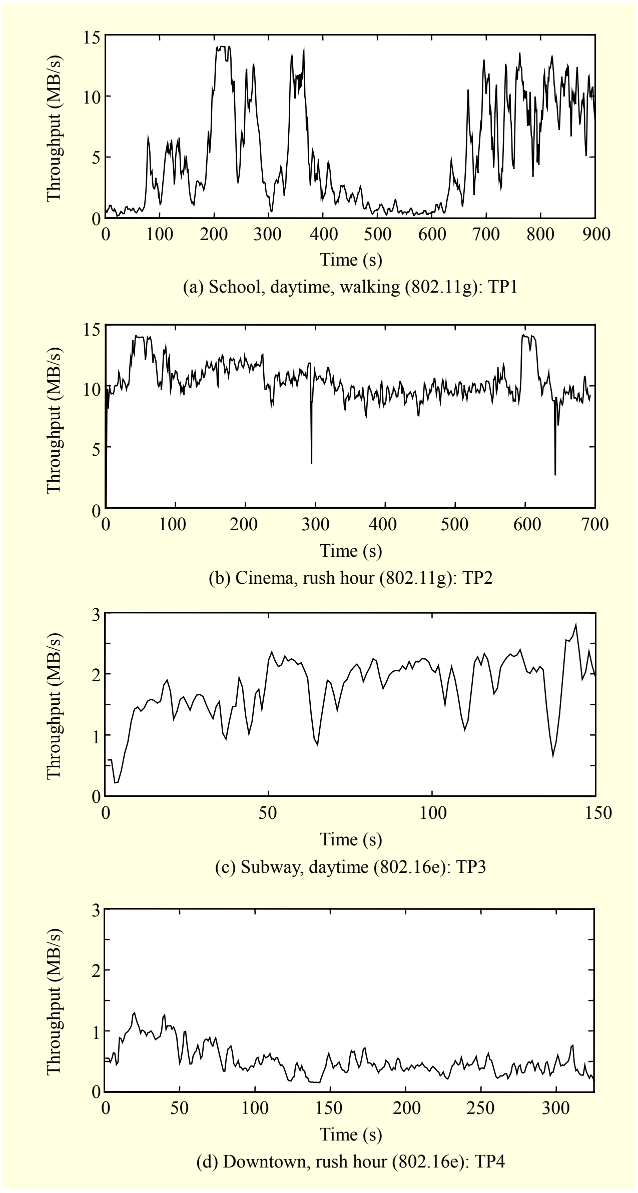


Fig. 5. Typical measured traffic patterns.

Table 1. Video stream parameters.

Parameter		Value
Video encoder		H.264 AVC (JM 13.4)
Video container		MPEG2-TS
Sequence name		Music video
Video quality	HQ	1,280×720, 5 Mbps
	MQ	800×480, 3 Mbps
	LQ	400×240, 1 Mbps

these cases pose difficult situations for video streaming and are suitable for demonstrating the effectiveness of the proposed mechanism. The selected traffic patterns (TPs) are illustrated in Fig. 5.

The measured data (that is, throughput, delay, and packet loss rate) are logged, and their TPs are stored and emulated with the NetDisturb network emulator [13], as shown in Fig. 6. Then, the network adaptive mechanisms with the HTTP server and video client are tested on the emulator with different TPs.

The general web server and Linux client are used and the proposed algorithms are implemented as a plug-in of the GStreamer. We set a video buffer with a 10-MB size to provide a smooth video play during simulation. All algorithms are adopted into the HTTP live streaming for an easy implementation. However, we expect that the proposed schemes, when ported to DASH, have similar results because DASH provides a framework without any specified adaptive mechanism.

The video stream consists of a metafile as well as the segmented streams, which are encoded in three different qualities, that is, low (LQ), medium (MQ), and high (HQ), without loss of generality, as described in Table 1.

One quality-level stream has a 60-second total length and each segmented stream among the overall stream is three

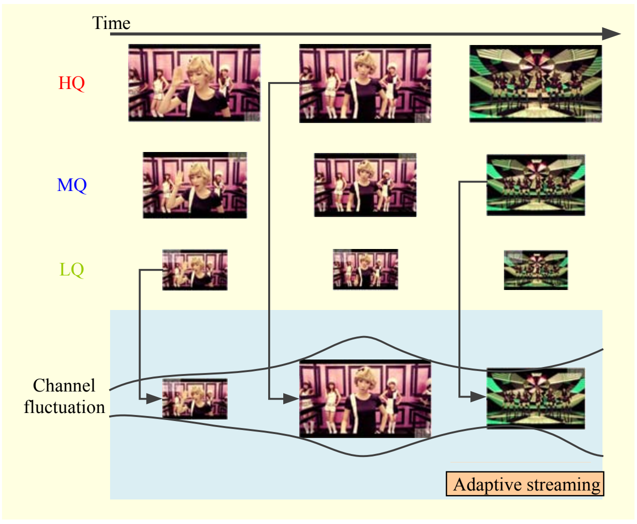


Fig. 7. Schematic diagram of network-adaptive video streaming.

seconds and can be switched to the other qualities of the segmented stream every three seconds. Therefore, there are 60 segmented streams and four metafiles. Figure 7 is a schematic diagram illustrating network-adaptive video streaming. The typical TPs are stored and emulated in the emulator, and a streaming client requests and receives video streams from an HTTP web server via the emulator.

2. Experiment Results

First, the performance of the DFI scheme is evaluated based on how accurately it tracks the measured throughput data. The difference between measured and estimated data is illustrated in Fig. 8, which shows that the DFI scheme catches up with the measured throughput better than the EWMA does.

Figure 9 illustrates a sensitivity analysis of ε by showing the mean absolute percentage error (MAPE). Changing ε makes a difference in the estimation error. Larger ε makes higher oscillation, and the lowest MAPE is achieved when ε is 0.05, but there is no remarkable performance change over the different values of ε.

Second, the streaming tests are varied on the experiment bed, as shown in Fig. 6. The compared mechanisms are "Nonadaptive," "EWMA," "DFI," and "DFI + AVRS." "Nonadaptive" means that only MQ is serviced. The EWMA mechanism is adaptive video streaming via the EWMA scheme, the DFI mechanism is the applied DFI scheme, and the DFI + AVRS mechanism uses both the DFI and AVRS schemes simultaneously. The traffic patterns in the network emulator adopt the experiment traffic patterns shown in Fig. 5. Figures 5(a), 5(b), and 5(c) represent TP1, TP2, and TP3, respectively, while Fig. 5(d) is not adopted because the measured throughput is too low for use in the high-volume

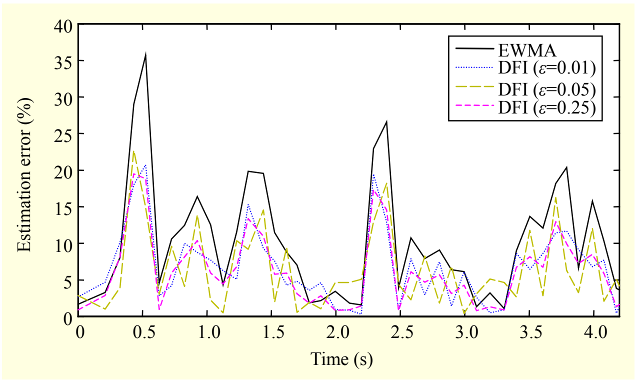


Fig. 8. Estimation error comparison between DFI and EWMA using TP1.

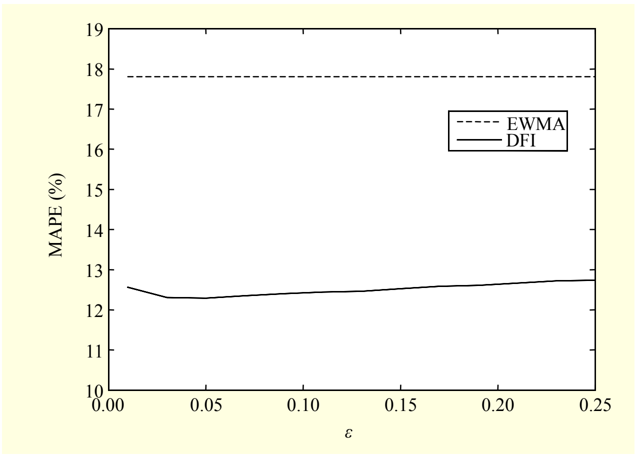


Fig. 9. MAPE of DFI throughput estimation scheme with variable ε over TP1.

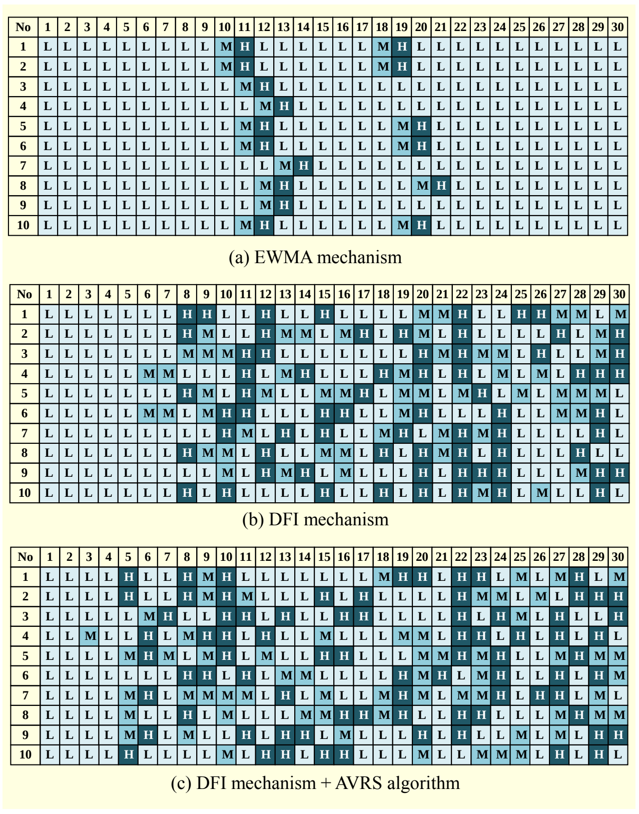
video streaming test.

Table 2 shows the experiment results. We extract such performance data as the percentage of different quality video and freezing time caused from insufficient video data to play at the receiver's buffer. For TP1, the DFI+AVRS mechanism has an HQ:MQ:LQ ratio of 26.3:20.3:53.4, while the EWMA mechanism has a ratio of 5.3:5.3:89.4 and the DFI has a ratio of 23.7:18.7:57.6. Therefore, the proposed DFI+AVRS mechanism provides the highest percentage of HQ video among all of the compared mechanisms under the same traffic patterns and also has the best overall video quality. These results are similar to those for the other traffic patterns. The total freeze time is measured during 600 seconds of streaming.

Figure 10 shows detailed sequences for selecting different video qualities at each time period under the TP1 network condition. The horizontal index corresponds to the temporal order for selecting the subsequent video quality and the vertical index is the number of experiment trials. The "H," "M," and "L" in Fig. 10 mean that, at each time period, the adaptive

Table 2. Experiment results.

Parameters	Applied scheme	Test pattern		
		TP1	TP2	TP3
HQ video percentage (%)	Nonadaptive	0	0	0
	EWMA	5.3	35.3	0
	DFI	23.7	37.3	7.3
	DFI + AVRS	26.3	42.7	8.7
MQ video percentage (%)	Nonadaptive	100	100	100
	EWMA	5.3	12.3	20
	DFI	18.7	33.3	44.0
	DFI + AVRS	20.3	33.3	46.3
Freeze time (s)	Nonadaptive	15.6	6.9	23.2
	EWMA	0	0.3	3.3
	DFI	0	0.6	2.7
	DFI + AVRS	0	0	2.4



No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L	L
2	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L	L
3	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
4	L	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
5	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L
6	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L
7	L	L	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
8	L	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L
9	L	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
10	L	L	L	L	L	L	L	L	L	L	M	H	L	L	L	L	L	L	M	H	L	L	L	L	L	L	L	L	L	L

(a) EWMA mechanism

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	L	L	L	L	L	L	L	H	H	L	L	H	L	L	H	L	L	L	L	M	M	H	L	L	H	H	M	M	L	M
2	L	L	L	L	L	L	L	H	M	L	L	H	M	M	L	M	H	L	H	M	L	H	L	L	L	L	H	L	M	H
3	L	L	L	L	L	L	L	M	M	M	H	H	L	L	L	L	L	L	L	H	M	H	M	M	L	H	L	L	M	H
4	L	L	L	L	L	M	M	L	L	L	H	L	M	H	L	L	L	H	M	H	L	H	L	M	L	M	L	H	H	H
5	L	L	L	L	L	L	L	H	M	L	H	M	L	L	M	M	H	L	M	M	L	M	H	L	M	L	M	M	M	L
6	L	L	L	L	L	M	M	L	M	H	H	L	L	L	H	H	L	L	M	H	L	L	L	H	L	L	M	M	H	L
7	L	L	L	L	L	L	L	L	L	H	M	L	H	L	H	L	L	M	H	L	M	H	M	H	L	L	L	L	H	L
8	L	L	L	L	L	L	L	H	M	M	L	H	L	L	M	M	L	H	L	H	M	H	L	H	L	L	L	H	L	L
9	L	L	L	L	L	L	L	L	L	M	L	H	M	H	L	M	L	L	L	H	L	H	H	H	L	L	L	M	H	H
10	L	L	L	L	L	L	L	H	L	H	L	L	L	L	H	L	L	H	L	H	L	H	M	H	L	M	L	L	H	L

(b) DFI mechanism

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	L	L	L	L	H	L	L	H	M	H	L	L	L	L	L	L	L	M	H	H	L	H	H	L	M	L	M	H	L	M
2	L	L	L	L	H	L	L	H	M	H	M	L	L	L	H	L	H	L	L	L	L	H	M	M	L	M	L	H	H	H
3	L	L	L	L	L	M	H	L	L	H	H	L	H	L	L	H	H	L	L	L	L	H	L	H	M	L	H	L	L	H
4	L	L	M	L	L	H	L	M	H	H	L	H	L	L	M	L	L	L	M	M	L	H	H	L	H	L	H	L	H	L
5	L	L	L	L	M	H	M	L	M	H	L	M	L	L	H	H	L	L	L	M	M	H	M	H	L	L	M	H	M	M
6	L	L	L	L	L	L	L	H	H	L	H	L	M	M	L	L	L	L	H	M	H	L	M	H	L	L	H	L	H	M
7	L	L	L	L	M	H	L	M	M	M	M	L	H	L	M	L	L	M	H	M	L	M	M	H	L	H	H	L	M	L
8	L	L	L	L	M	L	L	H	L	M	L	L	L	M	M	H	H	M	H	L	L	H	H	L	L	L	M	H	M	M
9	L	L	L	L	M	H	L	M	L	L	H	L	H	H	L	M	L	L	L	H	L	H	L	L	M	L	M	L	H	H
10	L	L	L	L	H	L	L	L	L	M	L	H	H	L	H	H	L	L	L	M	L	L	M	M	M	L	H	L	H	L

(c) DFI mechanism + AVRS algorithm

Fig. 10. Video quality selection of different mechanisms on TP1.

mechanisms select either the HQ, MQ, or LQ video segment, respectively. Figure 10 shows that the proposed DFI+AVRS mechanism selects many more HQ video segments than do the other mechanisms.

V. Conclusion

An adaptive video streaming system was presented and implemented in this paper, and the related network-adaptive mechanism for the implemented system was proposed. The proposed network-adaptive mechanism consists of two parts. One is a DFI scheme for advanced throughput estimation to adopt rapid network changing conditions, and the other is an AVRS scheme to efficiently select the next video segment.

Real network experiment measurement were performed to obtain some typical profiles to be used as TPs embodied within the network emulator. The proposed mechanism was evaluated in the implemented video streaming system with different TPs. The experiment results show that the proposed DFI+AVRS adaptive mechanism provides many more HQ video segments and less freezing time compared to other mechanisms.

References

[1] S. Akhshabi, A.C. Begen, and C. Dovrolis. "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP," *ACM MMSys.*, Feb. 2011, pp. 157-168.

[2] R. Pantos, Ed., "HTTP Live Streaming," IETF Internet-Draft, work in progress, Sept. 2011.

[3] A. Zambelli, "IIS Smooth Streaming Technical Overview," Mar. 2009. Available: http://www.microsoft.com/en-us/download/details.aspx?id=17678

[4] 3GPP TS 26.234, "Transparent End-to-End Packet-Switched Streaming Service (PSS): Protocols and Codecs," Dec. 2010.

[5] T. Stockhammer et al., "Text of ISO/IEC 23001-6: Dynamic Adaptive Streaming over HTTP (DASH)," ISO/IEC JTC1/SC29/WG11, N11578, Oct. 2010.

[6] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE Trans. Netw.*, vol. 11, no. 4, Aug. 2003, pp. 537-549.

[7] R. Kapoor et al., "CapProbe: A Simple and Accurate Capacity Estimation Technique," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, Oct. 2004, pp. 67-78.

[8] M. Li, M. Claypool, and R. Kinicki, "WBest: A Bandwidth Estimation Tool for IEEE 802.11 Wireless Networks," *IEEE Conf. Local Computer Netw.*, Oct. 2008, pp. 374-381.

[9] M. Gerla et al., "TCP Westwood with Adaptive Bandwidth Estimation to Improve Efficiency/Friendliness Tradeoffs," *Computer Commun.*, vol. 27, no. 1, Jan. 2004, pp. 41-58.

[10] H. Yoon and J. Kim, "Measurement-Based Achievable Throughput Estimation in IEEE 802.11a WLANs," *IEEE Commun. Lett.*, vol. 11, no. 9, Sept. 2007, pp. 714-716.

[11] GStreamer Team, "GStreamer: Open Source Multimedia Framework." Available: http://www.gstreamer.net

[12] ZTI Telecom, "IP Traffic – Test & Measure." Available: http://www.zti-telecom.com

[13] ZTI Telecom, "NetDisturb, Impairment Tool for IP Network." Available: http://www.zti-telecom.com

Yo-Han Kim received his BS and MS from Ajou University, Suwon, Rep. of Korea, in 2001 and 2003, respectively, and his PhD from Sungkyunkwan University, Suwon, Rep. of Korea, in 2012. From 2005 to 2006, he was a researcher at Pantech Inc. His research interests include reliable and efficient multimedia transmission over wireless networks.

Jitae Shin received his BS from Seoul National University, Seoul, Rep. of Korea, in 1986, his MS from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea, in 1988, and his MS and PhD in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1998 and 2001, respectively. He is an associate professor in the College of Information and Communication Engineering of Sungkyunkwan University, Suwon, Rep. of Korea. His research interests include video signal processing and transmission over next-generation Internet and wireless/mobile networks, focusing on QoS/QoE, 4G communication systems, and multimedia network control/protocol issues. He is a member of IEEE and IEICE.

Jiho Park received his PhD in electrical engineering from the University of Washington, Seattle, WA, USA, in 2002. His PhD research involved error resilient video communications and error concealment. He joined the Communication Research Laboratories of Samsung Electronics in 2002. Since 2011, he has been a faculty member at the College of Information & Communication Engineering, Sungkyunkwan University, Suwon, Rep. of Korea. His current interests include signal and video processing, pattern recognition, embedded systems, and mobile and TV platform.