Energy Cognitive Dynamic Adaptive Streaming over HTTP

Seohyang Kim[1], Hayoung Oh[2] and Chongkwon Kim[1]
[1] Dept. Of Computer Science, Seoul National University
Seoul 151-744, Republic of Korea
[e-mail: shkim@popeye.snu.ac.kr, ckim@snu.ac.kr]
[2] School of Electronic and Engineering, Soongsil University
Seoul 156-743, Republic of Korea
[e-mail: hyoh@ssu.ac.kr]
*Corresponding author: Chongkwon Kim

Received December 18, 2014; revised March 4, 2015; revised April 24, 2015; accepted May 3, 2015; published June 30, 2015

Abstract

CISCO VNI predicted an average annual growth rate of 66% for mobile video traffic between 2014 and 2019 and accordingly much academic research related to video streaming has been initiated. In video streaming, Adaptive Bitrate (ABR) is a streaming technique in which a source video is stored on a server at variable encoding rates and each streaming user requests the most appropriate video encoding rate considering their channel capacity. However, these days, ABR related studies are only focusing on real-time rate adaptation omitting energy efficiency though it is one of the most important requirement for mobile devices, which may cause dissatisfaction for streaming users. In this paper, we propose an energy efficient prefetching based dynamic adaptive streaming technique by considering the limited characteristics of the batteries used in mobile devices, in order to reduce the energy waste and provide a similar level of service in terms of the average video rate compared to the latest ABR streaming technique which does not consider the energy consumption. The simulation results is showing that our proposed scheme saves 65~68% of energy at the average global mobile download speed compared to the latest high performance ABR algorithm while providing similar rate adaptation performance.

Keywords: Mobile Network, Video Streaming, Energy Efficient Communication, ABR, MPEG-DASH

A preliminary version of this paper appeared in IEEE BigComp 2015, Feb 9-11, Jeju Island, Republic of Korea. This version includes a concrete analysis and supporting implementation results on LTE mobile network simulation with advanced logic and algorithms. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(NRF-2014R1A1A1003562).

http://dx.doi.org/10.3837/tiis.2015.06.010 **ISSN : 1976-7277**

1. Introduction

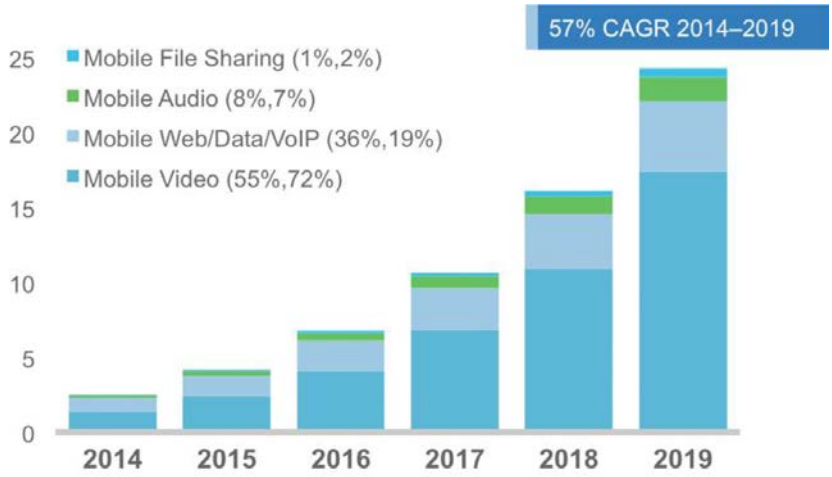

Fig. 1. Expectation for global mobile traffic growth from 2014 to 2019 from CISCO VNI [1]

Nowadays, the vigorous development of mobile devices and mobile communication techniques is allowing the multimedia content service area to grow at a rapid speed [16, 17, 18, 19, 22]. With this trend, the demand for mobile video streaming services has also caused video streaming to receive a great deal of attention from many researchers, service providers and operators[15]. **Fig. 1** shows the Visual Networking Index[1] from CISCO. It is annually updated and they are predicting that the annual growth rate of the quantity of mobile video streaming traffic will be 66% between 2014 and 2019 and the quantity of mobile video traffic will occupy 17.4 exabytes/month among that of whole mobile traffic which is 24.3 exabytes/month. Considering that the proportion of the other types of mobile traffic, such as File Sharing, Audio, and Web/data traffic, among whole mobile data traffic in 2019, it can be easily seen that the scale of the future video streaming services is very huge.

Recent video streaming services have made use of HTTP and the Adaptive Bitrate (ABR) technique. HTTP video servers store many chunks of a single video at different encoding rates[21]. These chunks are fractions of the video and each chunk consists of 4 to 10 second sequences[14]. The streaming client requests different encoding rates on the fly. In the past, there was only one encoding rate available for a given video on the server, so every streaming user had to receive the same chunk at the same encoding rate. Compared with Single Bitrate Encoded (SBE) videos, the greatest merits of using ABR streaming is that each user can watch the video at the most appropriate encoding rate. Given this trend, many papers have been published about the rate adaptation of ABR streaming and most of them have been focused on adapting the user's video rate in real time based on the user's possible receiving rate and the buffer occupancy. However, they do not take into consideration the fact that the mobile device's energy is limited by their battery. They only concentrate on real time rate adaptation to achieve a goal of higher average video rate and lower rebuffering rate.

In this paper, we suggest a novel approach to ABR streaming by considering the energy limited properties of mobile devices. ePF-DASH is an energy efficient prefetching based ABR streaming technique. We identify the source of the energy waste and propose an alternative strategy for dealing with this by prefetching the content. However, unlike SBE video, there are several difficulties involved in prefetching Multiple Bitrate Encoded (MBE) videos. In this paper, we describe answers to the following questions: which bit-rate should be selected and how can we compose a set of consecutive video chunks for one prefetching process with multiple video bitrates?

The main contribution of this paper is as follows. First, we introduce a prefetching technique for ABR streaming which is designed to reduce the energy waste by keeping the mobile device in the communication OFF mode for a longer period. Second, with a slow fading channel and LTE smartphone, the proposed technique reduces the energy consumption by about 56~59% and provides a similar video bitrate to recent ABR streaming techniques at the same average global mobile download speed.

This paper is organized as follows. We give background knowledge about HTTP video streaming technique and research trends with related works in Sections 2 and 3, respectively. We next describe the novel energy-efficient ABR streaming technique, ePF-DASH, in Section 4. Sections 5 and 6 describe the experimental environment and results, respectively, and the conclusions are given in Section 7.

2. HTTP Video Streaming Technique

HTTP video streaming is classified into Single Bitrate Encoding(SBE) and Multiple Bitrate Encoding(MBE) in accordance with the encoding scheme of the video stored on the server. In the case of SBE video, one video is stored on the server at one encoding rate and the most important problem for the client is to decide the size of the content to download at one time. However, in the case of MBE video, the most critical problem is the consecutive selection of the most appropriate video rate each time.

Fig. 2 describes representative client side chunk downloading strategies for SBE and MBE video. In the figure, the x axis represents the timing of the video streaming service in progress and the gray rectangles represent the periods of effective video chunk downloading and their height represents the transmission speed, indicating the mobile device's download speed of the corresponding chunk. The red rectangles represent the periods where the mobile device's communication mode is in the ON state.

2.1 Tail Energy of Mobile Device

In mobile device, there are roughly 2 states in communication mode. One is ON mode and the other is OFF mode. Mobile device can receive or transmit data packets only in the communication ON mode. In the OFF mode, it can not receive or transmit any data packets. In the **Fig. 2**, it can be seen that the mobile device's communication ON mode continues for some duration, even after it finishes downloading the corresponding chunk. For the mobile device's communication, there exists a constant time out period after its actual communication for the purpose of reducing the frequent switching of communication mode between ON and OFF, since mode switching consumes a great deal of energy. [3] and [6] provide tables detailing the

power and duration values of timeout and mode switching regarding to the mobile device's radio resource control state transition. Since such switching consumes energy while effectively doing nothing, streaming service providers or application developers should keep this in mind when they design energy efficient streaming applications for mobile devices.

2.2 Single Bitrate Encoding

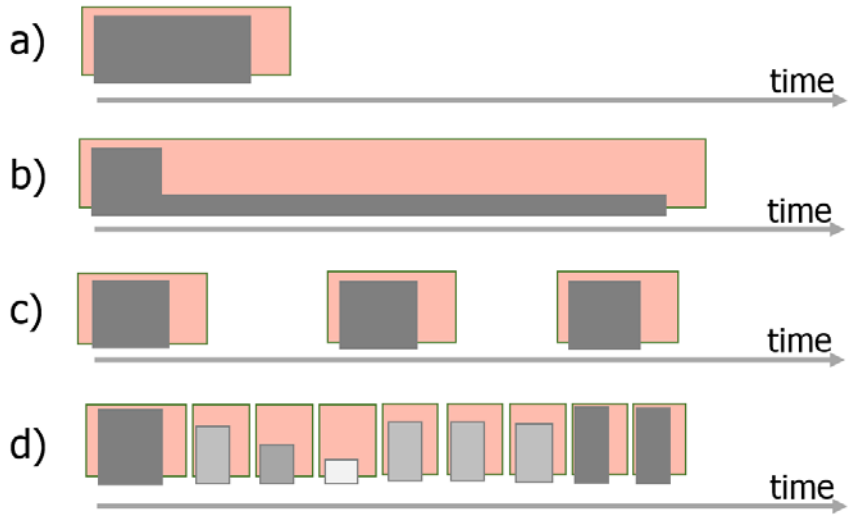

Fig. 2. Various chunk download strategy for HTTP streaming

[5] summarized the various download strategies for SBE video and we introduce 3 representative ones in this paper. **Fig. 2(a)** describes a Fast-caching strategy when the receiver downloads an entire video in one go with its maximum possible download speed. By using this strategy, the user can download the whole video in a short time and tail energy happens only one time at the point where the video download is finished. So mobile device stays in the communication ON mode in very short time and it can save energy. Therefore, this strategy is the best at the view of energy efficiency if streaming user watches whole video. However, if user stops watching the video right after whole video has been downloaded, a lot of useless data will have been downloaded and a huge amount of energy and communication resources will have been wasted. This kind of uselessly downloaded data is called data waste.

Fig. 2(b) describes the process of Throttling which downloads video chunks at the maximum speed until the buffer is filled up to some threshold as does Fast-cashing. Once the threshold has been reached, it throttles the download speed to its video playback speed and downloads the video little by little. In this strategy, device downloads video chunk while consuming the video sequences already downloaded in the buffer so it can reduce the amount of data waste. Compared to Fast caching, it can reduce the amount of data waste considerably when user abandons video watching in the middle of the video playback. However, in Throttling, mobile device should stay at communication ON mode for most of the video playback time consuming huge amount of energy. When we compare Fast-cashing with Throttling, Fast-cashing is energy efficient but not considers data waste and oppositely, Throttling is designed to reduce data waste but it is not energy efficient. In the medium of these two extreme strategy, ON-OFF strategy has been suggested and mostly used for mobile video streaming. **Fig. 2(c)** describes this strategy which downloads the video by dividing it

into several big chunks. In this scheme, it downloads some amount of the video at once with fast download speed until it reaches to specific maximum threshold as does Fast-cashing. After it reaches to this threshold, it turns off its communication mode and plays the video until the video sequence remained in the buffer is diminished until it reaches to specific minimum threshold. When video sequence in the buffer in second reaches to this minimum threshold, it turns on its communication mode again and downloads next sequence of video until the buffer filled up till the maximum threshold. It repeats this process over and over to the end. This strategy balances efficiency between data waste and energy consumption by downloading specific reasonable amount of video chunks at once in fast download speed and turning off the communication mode saving energy. Even though there exist several times of tail energy, it can reduce data waste, which occurs when user downloads long sequence of video and abandons watching, and energy consumption, which occurs from mobile device's staying at communication ON mode for a long time.

Taken together, the best energy efficienct streaming technique is in the order: Fast-caching, ON-OFF, and Throttling, while with respect to the data waste, Fast-caching is the worst one. Therefore, for mobile streaming, the ON-OFF strategy, which provides pretty great solution for balancing trade-off between the energy waste and data waste, is most commonly used for SBE video streaming.

2.3 Multiple Bitrate Encoding

General Adaptive Bitrate Streaming adopts a strategy such as the one described in **Fig. 2(d)**. DASH (Dynamic Adaptive Streaming over HTTP) [11, 12] services video streaming by using the ABR streaming technique and, because DASH server stores each source video at various encoding rates, DASH client selects the most appropriate one by calculating it in real-time, unlike in the case of SBE video streaming which downloads the video in large amounts at a time. However, the previous studies only focused on "real-time" transfer, but did not take into consideration the energy limited characteristics of mobile devices. In the case of general ABR streaming strategies such as the one shown in **Fig. 2(d)**, energy waste results not only from the tail energy, but also from the additional frequent mode switching. Because the impact of the server's encoding a video at multiple rates and the client's adaptation to the various video rates by calculating the most appropriate one in tidal flow is reduced when ABR streaming downloads large video chunks at one time, previous studies only concentrated on reducing the rebuffering rate and increasing the average video bitrate. This paper proposes an energy efficient strategy which improves on the existing works by considering the energy consumption.

3. Related Works

These days, many researchers are studying about balancing the tradeoff between energy efficiency (EE) and spectrum efficiency (SE), since EE is one of a big issue for mobile device and SE is an important topic for network communication while both one should be pursued even though these are conflicting each other. With this trend, [28] has proposed the system framework of cooperative green heterogeneous networks and [29] has addressed SBE video streaming over mobile ad hoc networks while both of them are providing solutions for balancing this tradeoff between EE and SE.

As we seen from **Fig. 1**, mobile video traffic occupies considerably large portion of whole mobile traffic. Different from wired network scenario, in the case of mobile device, its connection to the Internet is not stable and many researchers have been studied to achieve good video quality in the mobile scenario [2, 9, 10, 14, 26].

Before the advent of adaptive streaming, many researchers have been studied about energy efficient mobile video streaming. However, it was only in the case of SBE video and in this case, it is easy to devise energy efficient video chunk download strategy balancing between data waste and energy consumption because there is no necessity for client application to select video encoding rate. In these days, most of the streaming services are using adaptive streaming which has multiple version of video in the server with MBE scheme, and consequently client application has to select appropriate encoding rate every few seconds according to their varying channel state [2, 14, 20, 26, 27]. Since there are many things to considerate to achieve pretty great average video rate, researchers have their focus only on rate adaptation ignoring energy efficiency. However, mobile device depends on their battery and consuming the battery in efficient way is critical factor for user satisfaction. Streaming service providers and developers should keep it in mind when they create any services.

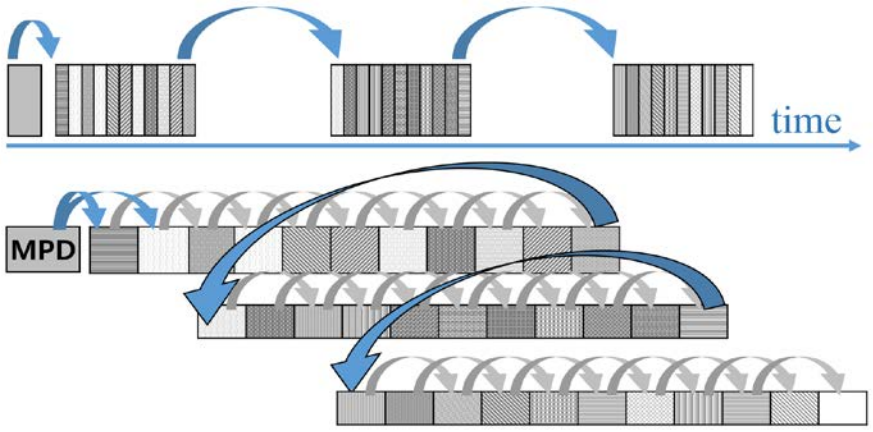

Fig. 3. Rate adaptation for ePF-DASH

[13] has addressed the energy consumption of video streaming with an Android mobile device in various environments and [3], [5] and [23] have proposed an energy efficient chunk scheduling strategy for SBE mobile video streaming. [3] dynamically adjusts mobile device's buffer size considering user's viewing statistics to save energy. [5] also reduces the energy waste and data waste by using crowd-sourced video viewing statistics, which are from YouTube[7], with consideration on tail energy. [23] used Gaussian Mixture Model (GMM) to predict the user demand in video playback time and calculated the buffer size dynamically to reduce device's energy consumption. However, since these schemes are based on SBE video, they cannot be applied to MBE video which has different chunk sizes and download times depending on the encoding rate, which exceedingly increases the complexity of the problem.

On the other hand, [2] have proposed a rate adaptation technique for MBE streaming, while only focusing on reducing the rebuffering rate and increasing the average video rate ignoring the importance of considering energy efficiency for mobile device. Consequently this technique wastes a lot of energy in the tail time by staying in the communication ON mode during most of the streaming time. Even though a lower rebuffering rate and higher average video rate were achieved by selecting a higher video rate when there is a low probability of

occurrence of rebuffering with an almost full buffer in this scheme, user satisfaction is likely to be low because the issue of energy consumption is not taken into consideration.

4. Energy Efficient Dash Video Streaming

The video encoding scheme used in this paper is Multiple Bitrate Encoding and it services ABR streaming. We describe an energy efficient video chunk downloading strategy, as shown in **Fig. 3**. The main points of Section 2 can be summarized as follows:

- Downloading large amounts at one time is more energy efficient.
- Downloading small amounts many times is more data efficient.
- Downloading at the maximum download speed is more energy efficient.

The proposed scheme, ePF-DASH, adopts a prefetching based ON-OFF strategy, which is commonly used in streaming services with Single Bitrate Encoded video, for DASH, a representative ABR streaming method. ePF-DASH provides similar average video rates and 56~59% energy savings at average global mobile download speeds compared to BBS[2], which is the latest rate adaptation technique for ABR streaming.

In the case of BBS, it adapts the bitrate based on the size of the buffer in seconds. That is, because the probability of occurrence of rebuffering is high when there is a small amount of video in the buffer and the probability of occurrence of rebuffering is low in the opposite case, it downloads the video at a lower rate and fills the buffer quickly in the former case, whereas it downloads the video at a higher rate in the latter case where it is safe from rebuffering events. Therefore, it can achieve higher video rates while providing protection against rebuffering events. Because it frequently downloads very small amount of video chunks continuously during the whole video playback time, it is quite efficient in the view of data waste, however it is extremely inefficient in the view of energy consumption.

As we described in Section 2, research topics in MBE video streaming have been only concentrated on real-time rate adaptation. MBE adaptive streaming chunk downloading strategies such as BBS can be compared to SBE Throttling strategy because they are downloading small amount of chunk continuously filling up the buffer in the speed of video playback. For the MBE video streaming, Fast-cashing or ON-OFF strategies have not been used because these strategies are based on video sequence prefetching technique. The reason these prefetching techniques could be applied for SBE video streaming is that in SBE video, there are no necessity to select current best encoding rate because there are only one version of the video in the server with specific one encoding rate. However, with MBE adaptive streaming, because there are various versions of video in the server with various encoding rate, client application should select current most appropriate encoding rate as time goes on considering their varying channel state. Consequently, researchers studying about MBE adaptive streaming have had their focus on real-time rate adaptation and could not see the importance of considering mobile device's energy limited characteristics. As we seen from Section 2, ON-OFF strategy is in the middle of extreme two strategies which are Fast-cashing and Throttling, and therefore it is mostly used for mobile device.

In this paper, we are proposing ON-OFF strategy for MBE streaming. With this scheme, one whole video sequence is divided into several adequate amount of video sequences and each sequence is downloaded as does ON-OFF strategy. By using this strategy, we can balance

Algorithm 1. Rate adaptation strategy of ePF-DASH

```
Functions:
    availableRate( r )
        input: rate r in bps
        output: maximum available encoding rate below the rate r
    rateIncrease ( r )
        input: encoding rate r in bps
        output: increase r by 1 grade higher among {r[1], …, r[M]}

Parameters:
    Video rate of the n^th chunk : c_r[n] , i=1…N
    Encoding rate of the video : r[m], i=1…M
    Start time of downloading the n^th chunk : t_s[n]
    End time of downloading the n^th chunk : t_e[n]
    Size of the n^th chunk : c_s[n] , i=1…N
    Download speed of the n^th chunk : d[n], n=1…N
    Video content in the buffer in sec at time t: B(t)

Initial Settings:
    n = 1
    B(0) = 0
    c_r[1] = availableRate( download speed of MPD file )
    c_r[2] = availableRate( download speed of MPD file )

Algorithm:
    While( n<=N ){
        d[n] =  c_s[n]/(t_e[n]−t_s[n])
        c_r[n+2] = availableRate( d[n] )
        if ( B(t) >= thrsh1 )  rateIncrease( c_r[n+2] )
        if ( B(t) >= thrsh2 )  rateIncrease( c_r[n+2] )
    }
```

the tradeoff between data waste and energy efficiency. However, because MBE video streaming client application has to adapt its rate selection to its current channel capacity, we use this ON-OFF prefetching technique only when it's chunk downloading speed reaches to maximum encoding rate. Since before it's download speed reaches to maximum encoding rate, client streaming application has to strive to increase its encoding rate to achieve higher video quality, there are no composure to prefetch video sequence in this case. However, after it's download speed exceeds the maximum encoding rate downloading video chunks with the highest encoding rate, it means there are no other higher video quality to achieve and client application has enough composure to prefetching next video sequence with highest video quality. Consequently, client can adopt prefetching technique to download more video sequences in Fast-caching like manner and turn off its communication mode to save energy. With these logics, our proposed scheme adopts Throttling like strategy in slow download speed to achieve higher video quality with communication ON state during the most of the playback time. However, it adopts ON-OFF strategy in fast download speed where device's download speed exceeds the maximum encoding rate with highest video quality having enough composure to turn off its communication mode saving energy.

Algorithm 2. Chunk download strategy of ePF-DASH

```
Parameters:
    rxONmode = TRUE or FALSE

Initial Settings:
    n = 1
    B(0) = 0
    rxONmode = TRUE

Algorithm
    While( n <= N ){
        If( rxONmode = TRUE ){
            Download n^th chunk
            if ( B(t) >= maxThrsh ) rxONmode = FALSE
        }else{
            if ( B(t) <= minThrsh ) rxONmode = TRUE
        }
    }
```

As we described above, our proposed scheme ePF-DASH introduces a prefetching technique and downloads video with an ON-OFF strategy. In other words, it downloads large amounts of video content in one time when the mobile device's communication state is ON and, after it reaches to maximum buffer size, it stays in the OFF mode for a long period to save energy consuming buffered video sequence and, when the buffer size shrinks again to the minimum threshold, it changes the state to ON and downloads again until the buffer is filled up to the maximum threshold, as in the case of ON-OFF streaming in SBE video streaming.

Different from ordinary rate adaptation technique derived from universal ABR streaming techniques, ePF-DASH selects the similar rate to its effective chunk download speed based on a calculation of downloading speed of Media Presentation Description (MPD) file. After this, it decides appropriate rate for the after next chunk of video based on download speed of current chunk by calculating the chunk size divided by the downloading time. In other words, it adapts the rate of the $(n+2)^{th}$ chunk to the download speed of the n^{th} chunk as described in **Fig. 3**. By doing this, it can reduce the latency occurred from HTTP request messages and HTTP response messages of every video chunks. The maximum and minimum thresholds for the buffer are set to 200s and 20s, respectively, for a simple comparison with BBS. The longer the time-gap between minimum and maximum threshold is, the longer the mobile device can stay in the communication OFF mode. With this setting, when the size of the buffer is between 20s and 200s, it turns its communication state from the ON to the OFF mode after reaching its maximum threshold and turns it from OFF to ON after reaching to minimum threshold. Details on rate adaptation and chunk downloading strategy are addressed in **Algorithm 1** and **Algorithm 2** respectively.

Algorithm 1 describes the process of the rate adaptation for ePF-DASH. With this algorithm, the video rate for the first two chunks is set to similar rate of effective mobile device's download speed based on the calculated MPD download speed. After this, every chunk's after next chunk download speed is set to the rate similar to mobile device's current effective download speed based on the calculation of current chunk download speed. As described above, minimum and maximum threshold are set to 20s and 200s repectively for the

simple comparison with BBS. In the ePF-DASH, we adopt 2 more threshold to increase its video rate when the buffer is filled up with enough video sequence since there are low probability of occurrence of rebuffering events when the buffer is full enough as does BBS. Consequently we adopt threshold1 and threshold2 to enhance video quality. If the buffer size exceeds the threshold1, it increases the selected encoding rate by 1 level higher and if the buffer size exceeds the threshold2, it increases the selected encoding rate by 2 level higher. However, the buffer should keep the amount of video sequence above the minimum threshold even though the network channel state is not good even in case it requests video chunk with 1 or 2 level raised encoding rate compared to its effective download capacity. So we set the threshold1 and the threshold2 to the value which ensure the buffer size keeps above the minimum threshold for a specific duration, which is called endurance time, even though mobile device's effective download speed can not support raised encoding rate. With this logic, threshold1 and threshold2 are calculated as follows:

$$thrsh1 = \min Thrsh + \max_{m}\left(\frac{r[m+1]}{r[m]}\right) \cdot endureTime \tag{1}$$

$$thrsh2 = \min Thrsh + \max_{m}\left(\frac{r[m+2]}{r[m]}\right) \cdot endureTime \tag{2}$$

where r[m] represents m^{th} encoding rate among the avaiable M encoding rate. To talk about encoding rate r[m], the number of encoding rate is M and array r[m] is in an ascending order. For example, r[1] is the minimum encoding rate and r[M] is the maximum encoding rate. A parameter endureTime represents endurance time which was addressed above. In accordance with endureTime, if some streaming client application developers want to design application which has high rebuffering rate with high average video quality, he or she can assign small value for endureTime. Oppositely, low rebuffering rate with relatively low average video quality can be achieved with larger endureTime value. With this, ePF-DASH can enjoy higher video quality in a stable manner. Emphatically, encoding rate of ePF-DASH could be raised like this based on the fact that if the buffer is full enough, there are low probability of occurrence of rebuffering event.

In **Algorithm 2**, boolean value rxONmode represents whether mobile device can download video chunk or not. Device can download video chunk only when rxONmode is TRUE. As described above, mobile device's buffer is filled up until it reaches to the maximum threshold and after it reaches to that threshold, it changes its rxONmode from TRUE to FALSE and stop downloading video chunk. After a few seconds of tail duration, it turns off its communication mode from ON to OFF and save energy. While its staying at communication OFF mode with rxONmode is set to FALSE, video sequence in the buffer is consumed and buffer size shrinks. When the size of the buffer becomes below the minimum threshold, rxONmode is set to TRUE and mobile device's communication mode is turned on so that it can download video chunks

Table 1. Power and duration of mobile device on LTE environment

	receive power	tail power	promo-tion power	tail duration	promo-tion duration
LTE	1.58W	1.3W	1.2W	10s	2.6s
LTE+DRX				0.75s	

Table 2. Experimental Results

Download Speed [Mbps]		60 ±15	11.3 ±3	5 ±1.5	2 ±1.5
Average Video Rate [Mbps]	BBS	2.49	2.49	2.49	1.99
	ePF-DASH	2.5	2.5	2.5	1.94
Total Energy [Ws]	BBS	1814 (1530)	1898 (1809)	2033 (1987)	2440 (2428)
	ePF-DASH	247 (139)	670 (586)	1317 (1257)	2388 (2376)
Tail Energy [Ws]	BBS	1706 (339)	1341 (328)	753 (298)	13 (1.4)
	ePF-DASH	117 (9)	331 (7)	65 (5)	13 (1.4)
Promotion Energy [Ws]	BBS	3 (1085)	3 (924)	3 (412)	3 (3)
	ePF-DASH	28 (28)	21 (21)	15 (15)	3 (3)
Energy Saving Rate		86% (91%)	65% (68%)	35% (37%)	2% (2%)

again. When the device's download speed is fast enough to reach at the maximum threshold of the buffer, it can go through ON-OFF streaming mode saving energy. On the contrary, when the device's download speed is slow to such an extent as below the maximum encoding rate, it has no composure to fill the buffer to the extent of maximum threshold and it can not go into ON-OFF streaming mode since it has to strive to achieve higher video quality turning on its communication mode ON for a whole video playback time.

5. Experimental Environment

We simulated ePF-DASH in a slow fading channel in a Microsoft Visual Studio 2010 development environment. [5] and [8] investigated the power and duration values of different communication states using an LTE smartphone. With this information, we assumed the values of the power and duration of communication for the LTE smartphone listed in **Table 1** [5][8]. **Table 2** shows the experimental results with LTE simulation based on the value from **Table 1**.

We compared the experimental results with those of BBS, which greedily selects a higher video rate when there is a low probability of rebuffering. BBS[2] is the latest rate adaptation technique for ABR streaming, which has high performance in terms of the video rate with a low rebuffering rate. We conducted experiments for 2 tail timeout values: LTE and LTE+DRX (Discontinuous Reception) which is a technique employed for reducing the tail energy by

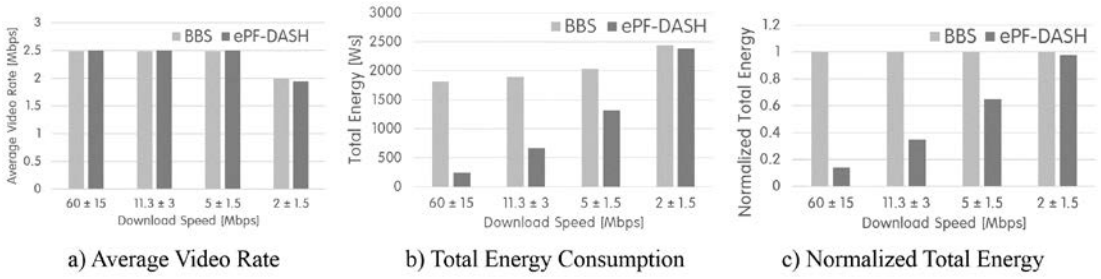

Fig. 4. Experimental comparison between BBS and ePF-DASH

cutting down the tail timeout from 10s to 0.75s. We assumed a slow fading channel with various download speeds of the mobile device, viz. 2Mbps, 5Mbps, 11.3Mbps, and 60Mbps. The value of 60Mbps comes from measurements of the download speed of an LTE smartphone in Seoul, Republic of Korea, over a one week period using the Android application[24] for network speed measurement, where the average speed was 65Mbps. 11.3Mbps is the average global mobile download speed from the Netflix Net Index[27]. We used 2Mbps as the low download speed, which is approximately equal to the encoding rate. We set the mobile device's download speed to fluctuate with a sine wave around the average bitrate and assumed videos with a length of 25min and encoding rates from 0.5Mbps to 2.5Mbps with 0.5Mbps interval. We also assumed every video chunk stored on the DASH server is in the size of 4 seconds as Netflix[2].

As described above, we set the minimum and maximum threshold as 20s and 200s and set the endurance time as 25s. Consequently, threshold1 and threshold2 are set as 70s and 95s respectively based on Equation (1) and Equation (2). With this, if the size of the buffer is larger than 70s, it increases the encoding rate by 1 grade and if the size of the buffer is larger than 95s, it increases it by 2 grades as described in **Algorithm 1**. This is based on the fact that if the amount of content in the buffer is large enough, there is a low probability of rebuffering as does BBS.

6. Experimental Results

In **Table 2**, the values in parentheses refer to the results from LTE+DRX. Even though LTE+DRX reduces the tail time from 10s to 0.75s, the energy consumption of streaming is not reduced much because of the mode switching time and corresponding energy consumption. The higher the download speed of the mobile device, the higher the energy efficiency of ePF-DASH. Additionally, even if the tail duration is reduced by using DRX mode, additional energy waste still occurs due to the frequent communication mode switching.

We have visualized **Table 2** as **Fig. 4** for perspicuous comparison. **Fig. 4(a)** is representing the average video rate of BBS and ePF-DASH and **Fig. 4(b)** is representing total energy consumption which is the sum of packet download energy, tail energy and promotion energy. **Fig. 4(c)** is normalized energy consumption of ePF-DASH compared to BBS. As **Fig. 4** describes, ePF-DASH achieves almost same performance as BBS does while saving significant amount of energy, especially when a mobile download speed is very fast.

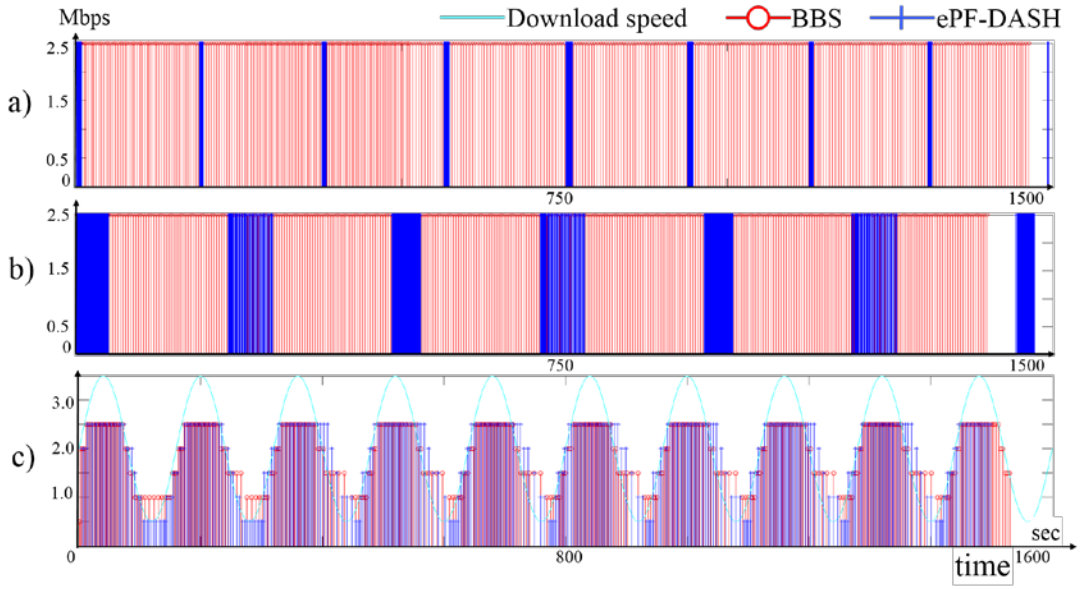

Fig. 5. Comparison of rate adaptation at mobile device download speeds of 60Mbps(a), 11.3Mbps(b) and 2Mbps(c)

Fig. 5 shows the rate selection of ePF-DASH and BBS in tidal flow with different download speeds of the mobile device. **Fig. 5(a)**, **Fig. 5(b)** and **Fig. 5(c)** refer to the cases where the download speeds are 60Mbps with a slow fading sine wave from 45Mbps to 75Mbps, 11.3Mbps which is an average global mobile download speed with a slow fading sine wave from 8.3Mbps to 14.3Mbps and 2Mbps with a sine wave from 0.5Mbps to 3.5Mbps respectively, with period of 160s. When the download speed of the mobile device is high enough, ePF-DASH downloads about 180 sec of content - which is the gap between minimum and maximum threshold - as quickly as possible and then turns off the device's communication mode and saves energy by staying in communication OFF mode for a long time. However, in contrast with ePF-DASH, BBS downloads chunks periodically regardless of whether the download speed is high or not: it just selects a higher bitrate greedily. Though **Fig. 5(a)** shows the case of a high download speed which is much higher than the video rate, it is a good example to understand how much energy ePF-DASH can save when the download speed is higher than the encoding rate. In the case of average global mobile download speed, as shown in **Fig. 5(b)**, video sequences are downloaded in energy efficient way too. We assumed a lower download speed of around 0.5~3.5Mbps, as shown in **Fig. 5(c)**. In case the download speed is lower than encoding rate, the download speed is almost equal to the playback speed and there is no time for mobile devices to stay in the communication OFF mode. Therefore, in this case, we should attempt to answer the question as to how ePF-DASH manages to adopt video rate rather than how long it stays in the communication OFF mode saving energy. Though ePF-DASH selects the next video rate based on the current download speed, it greedily selects a higher video rate when the amount of content in the buffer is large enough, so there is a low probability of rebuffering, as in the case of BBS. As shown in **Fig. 5(a)**, in the case of a very high download speed, both BBS and ePF-DASH selects the highest video rate except for the first rate selection. The difference is that BBS downloads whole video over entire streaming time and ePF-DASH downloads it with 9 times of separate prefetching. Since ePF-DASH downloads quickly and stays in communication OFF mode for most of the

streaming time, it provides an energy saving of 86~91%. During the simulation, ePF-DASH downloaded whole video with 9 big chunks at 60Mbps, 7 big chunks at 11.3Mbps and 5 big chunks at 5 Mbps.

Different from BBS, ePF-DASH's rate selection for the first 2 chunks is set to the highest encoding rate below the download speed of MPD file as described in **Algorithm 1**. However, BBS always sets the first chunk's video rate as minimum encoding rate. It explains why the average video rate of ePF-DASH is higher than BBS when device's chunk downloading speed is fast enough.

Fig. 5(c) shows that in the case of a very low download speed, the average video rates for BBS and ePF-DASH are 1.99 and 1.94Mbps, respectively, and the energy saving of ePF-DASH is 2%. When device's downloading speed is low below the maximum encoding rate, there is no composure for streaming application to save energy since it is busy to increase average video quality downloading video chunks with the highest encoding rate within its channel capacity. Consequently, it can not reach to the maximum threshold and can not go into ON-OFF energy saving mode. It explains why energy saving rate of ePF-DASH is not that high when device's download speed is below the highest encoding rate.

We also conducted experiments with the average global mobile download speeds, as shown in **Fig. 5(b)**, which is 11.3Mbps, and the results show that the video rates for BBS and ePF-DASH are 2.49 and 2.5Mbps, respectively, and the energy saving rate of ePF-DASH is 65~68%. Considering that ePF-DASH saves much energy while providing almost similar average video rate performance, it is quite competitive streaming technique compared with the latest ABR streaming technique. Now, ABR streaming users can watch videos while consuming much less energy with ePF-DASH.

7. Conclusion

Recent ABR streaming related research papers have proposed sophisticated rate adaptation methods which have good performance in reducing the rate of rebuffering and increasing the average video rate. However, they only focus on real-time rate adaptation and do not consider the energy limited characteristics of mobile devices. In this paper, we propose ePF-DASH, whose performance in rebuffering and rate adaptation is similar to that of the strategies proposed in recent studies, while also considering the issue of energy efficiency as an additional property. ePF-DASH selects the video rate by considering the download speed and refers to the quantity of video content in the streaming buffer when the amount of content in the buffer is big enough. By selecting a higher video bitrate than the download speed when the buffer is full enough, ePF-DASH achieves a higher average video bitrate in a stable manner. Moreover, by using a prefetching technique and staying in communication OFF mode for a longer period, it provides an energy saving of 65~68% at the average global mobile download speed compared to the existing ABR algorithm with similar performance in terms of the rate adaptation. We will develop an Android implementation of ePF-DASH with more refined performance improvement in a future work.

References

[1] http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnel, M. Watson, "A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," *ACM SIGCOMM*, 2014. Article (CrossRef Link)

[3] X. Li, M. Dong, Z. Ma, F. Fernandes, "GreenTube: Power Optimization for Mobile Video Streaming via Dynamic Cache Management," *ACMMM*, 2012. Article (CrossRef Link)

[4] F. Qian, Z. W, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, "TOP: Tail Optimization Protocol for Cellular Radio Resource Allocation," *ICNP*, 2010. Article (CrossRef Link)

[5] M. A. Hoque, M. Siekkinen, J. K. Nurminen, "Using Crowd-Sourced Viewing Statistics to Save Energy in Wireless Video Streaming," *ACM MobiCom*, 2013. Article (CrossRef Link)

[6] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, "Characterizing Radio Resource Allocation for 3G Networks," *IMC*, 2010. Article (CrossRef Link)

[7] https://www.youtube.com/yt/playbook/yt-analytics.htm

[8] J. Huang, F. Qian, Al. Gerber, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," *MobiSys*, 2012. Article (CrossRef Link)

[9] H. Oh, "A Robust Mobile Video Streaming in Heterogeneous Emerging Wireless Systems," *KSII Transactions on Internet and Information Systems*, vol. 6, no. 9, 2012. Article (CrossRef Link)

[10] H. Cui, D. Qian, X. Zhang, I. You and X. Dong, "Optimizing the Joint Source/Network Coding for Video Streaming over Multi-hop Wireless Networks," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 4, 2013. Article (CrossRef Link)

[11] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP - Standards and Design Principles," *ACM MMSys*, 2011. Article (CrossRef Link)

[12] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE Multimedia*, 2011. Article (CrossRef Link)

[13] R. Trestian, A.-N. Moldovan, O. Ormond, G.-M. Muntean, "Energy Consumption Analysis of Video Streaming to Android Mobile Devices," *NOMS*, 2012. Article (CrossRef Link)

[14] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, M. Chiang, "A Scheduling Framework for Adaptive Video Delivery over Cellular Networks," *ACM MobiCom*, 2013. Article (CrossRef Link)

[15] A. Chan, A. Pande, E. Baik, P. Mohapatra, "Temporal Quality Assessment for Mobile Videos," *ACM MobiCom*, 2012. Article (CrossRef Link)

[16] J. Yoon, S. Banerjee, H. Zhang, S. Rangarajan, "MuVi: A Multicast Video Delivery Scheme for 4G Cellular Networks," *ACM MobiCom*, 2012. Article (CrossRef Link)

[17] H. Cui, C. Luo, Chang W. C., F. Wu, "Robust Uncoded Video Transmission over Wireless Fast Fading Channel," *IEEE INFOCOM*, 2014. Article (CrossRef Link)

[18] L. Golubchik, S. Khuller, K. Mukherjee, Y. Yao, "To send or not to send: Reducing the cost of data transmission," *IEEE INFOCOM*, 2013. Article (CrossRef Link)

[19] R. Li, B. Li, A. Eryilmaz, "Throughput-Optimal Wireless Scheduling with Regulated InterService Times," *IEEE INFOCOM*, 2013. Article (CrossRef Link)

[20] J. Lee, J. Hwang, N. Choi, and C. Yoo, "SVC-based Adaptive Video Streaming over Content-Centric Networking," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 10, pp. 2430-2447, 2013. Article (CrossRef Link)

[21] Y. Sanchez, T. Schierl, D. Hong, D. D. Vleeschauwer, Y. L. Louedec, "iDASH: Improved Dynaamic Adaptive Streaming over HTTP using Scalable Video Coding," *ACM MMSys*, 2011. Article (CrossRef Link)

[22] R. Bhatia, T. V.Lakshman, A. Netravali, K. Sabnani, "Improving mobile video streaming with link aware scheduling and client caches," *IEEE INFOCOM*, 2014. Article (CrossRef Link)
[23] H. Shen and Q. Qiu, "User-Aware Energy Efficient Streaming Strategy for Smartphone Based Video Playback Applications," *IEEE DATE*, 2013. Article (CrossRef Link)
[24] http://www.benchbee.co.kr/
[25] L. Toni, R. Aparicio-Pardo, G. Simon, A. Blanc, and P. Frossard, "Optimal Set of Video Representations in Adaptive Streaming," *ACM MMSys*, 2014. Article (CrossRef Link)
[26] G. Wang, K. Wu, Q. Zhang and L. M. Ni, "SimCast: Efficient Video Delivery in MU-MIMO WLANs," *IEEE INFOCOM*, 2014. Article (CrossRef Link)
[27] http://www.netindex.com/
[28] R. Q. Hu and Y. Qian, "An Energy Efficient and Spectrum Efficient Wireless Heterogeneous Network Framework for 5G Systems," vol. 52, no. 5, pp. 94-101, *IEEE Communications Magazine*, 2014. Article (CrossRef Link)
[29] L. Zhou, R. Q. Hu, Y. Qian, and H.-H. Chen, "Energy-Spsectrum Efficiency Tradeoff for Video Streaming over Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 5, pp. 981-991, 2013. Article (CrossRef Link)

Seohyang Kim received the B.S. degree in Computer Science from Sejong University, Seoul, Korea in 2013. She is currently a Ph.D. student in the School of Computer Science and Engineering at Seoul National University, Seoul, Korea. Her current research interests include wireless video streaming, dynamic adaptive streaming and wireless networks.

Hayoung Oh received the B.S. degree in Computer Science from Duksung Womans University and the M.S. degree in the School of Computer Science and Engineering from Ewha Womans University in 2002 and 2006 respectively. And she received the Ph.D. degree in Computer Science from Seoul National University in 2013. From 2002 to 2004, she joined Shinhan Financial Group as a developer in applied research. In 2010, she was with U.C. Berkeley as a researcher. Since 2013, she has been with Soongsil University as a professor in the School of Electronic Engineering. Her research interests include social and computer networks, and security.

Chongkwon Kim received the B.S. degree in industrial engineering from Seoul National University, the M.S. degree in operations research from Georgia Institute of Technology, and the Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign in 1981, 1982, and 1987, respectively. In 1987, he joined Bellcore as a Member of Technical Staff and worked on Broadband ISDN and ATM. Since 1991, he has been with Seoul National University as a Professor in the School of Computer Science and Engineering. His research interests include wireless and mobile networking, high speed network control, distributed processing, and performance evaluation.