A User-Side Energy-Saving Video Streaming Scheme for LTE Devices

Song Gun Lee, Jeman Park, and Hyogon Kim, *Member, IEEE*

***Abstract*—Today, most streaming services rely on HyperText Transfer Protocol (HTTP) to deliver the contents to users, and mobile users are increasingly choosing Long Term Evolution (LTE) as the wireless access mode. In this letter, we propose a user-side scheme that takes LTE modem operation into account in video streaming over HTTP, leading to as much battery saving gain as 40% without Discontinuous Reception (DRX) and potentially 70% if DRX is deployed in future.**

***Index Terms*—Long Term Evolution (LTE), video streaming, HTTP, user-side, energy saving.**

I. Introduction

NOWADAYS, live and on-demand streaming services on mobile devices are becoming increasingly popular. The streaming flow has a distinctive characteristic, that is, it is less bursty than other data traffic but lasts long. The property arises because the system does not need to transmit streaming data at a higher rate than the user consumes. One drawback of this paced transfer, however, is that the modem gets to stay awake over longer period of time and hence suffer greater battery drain. In this letter, we account for the power dynamics of LTE protocol in altering the streaming protocol behavior to be more energy-efficient. We show that this scheme can lead to as much battery saving gain on LTE user devices as 40% without Discontinuous Reception (DRX), and potentially 70% if DRX is deployed.

LTE modem has to be in Active state while a mobile device goes through four phases for video streaming (Fig. 1). First, connection setup acquires necessary radio resource through LTE signaling [1]. Second, over the obtained radio channel, the signaling for TCP and then HTTP are performed to initiate the video download. Third, video streaming commences and finishes. Finally, there is Tail period, which is used to reduce the latency of transition from Idle to Active state in the modem in case another data transfer follows soon. The default value for the Tail is 10 seconds, and the modem gets to waste energy if there is no ensuing data transfer during this period. A remedy to the energy waste problem in the Tail period is Discontinuous Reception (DRX) mode [2]. The DRX-activated mobile stays in Active for up to 2.5 seconds, after which it wakes up every DRX cycle and checks if there is data to receive. If there is none, it goes back to Idle.

In HTTP streaming, mobiles use HTTP GET for streaming service request. An interesting characteristics of the HTTP transaction is that the client can affect the server's data transmission behavior through the `Range` field in the GET request or using a parameter in the URI. If the byte range covers the entire contents body, it is called *progressive* download, otherwise, *chunk* download. In the former, a single HTTP request suffices. But in the latter, multiple HTTP requests are periodically sent. Most video streaming services use the progressive download.

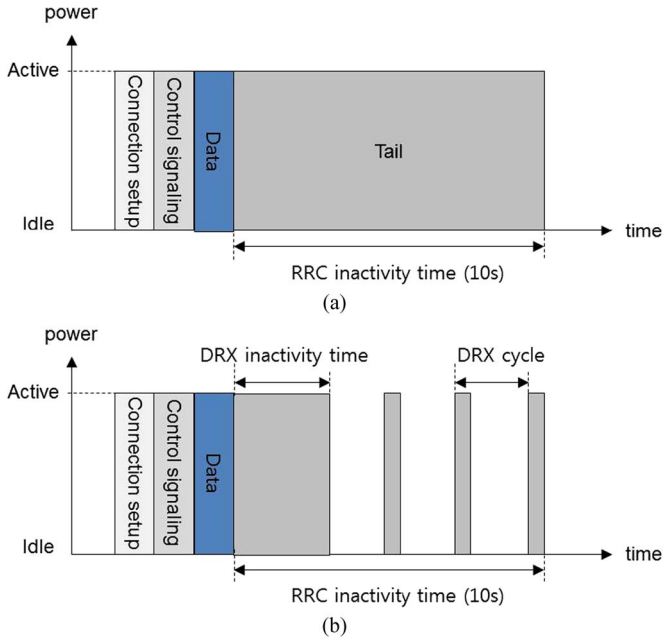

Fig. 1. Power consumption models in LTE communication.

In progressive download, most video or audio streaming servers (in particular the five most popular services, YouTube, Vimeo, Metacafe, Hulu, Veoh) initially use "fast start" stream ing [3] to quickly fill up the play-out buffer of the client. Once the client buffer is filled, most servers (e.g., Apache, MS IIS, NGINX) typically transmit at a low rate proportional to the encoded video bitrate, which is called *throttling*. The purpose of throttling is to reduce the peak traffic load at the server and minimize the chances of wasting downloaded data if the user aborts [4].

II. Dynamics of Progressive Download

We measure the energy consumption that video streaming incurs on a LTE user device that uses a commercial mobile service provider network for LTE access. For the user device, we use a LG F240 smartphone, to which we attach a Monsoon power monitor. The video clip we use in the measurements is Gangnam Style on YouTube, whose playtime is 252 seconds. The video format is MP4 480∗360 (360p), the video codec is H.264, and the total video file size is approximately 22.8 MB.

Manuscript received December 24, 2014; accepted April 2, 2015. Date of publication April 6, 2015; date of current version June 5, 2015. This work was supported by the Ministry of Education, Science and Technology of Korea through the Mid-career Researcher Program under NRF Grant 2011-0028892. The associate editor coordinating the review of this paper and approving it for publication was G. Giambene.

S. G. Lee is with SIC R&D Center, LG Electronics, Seoul 150-721, Korea.

J. Park and H. Kim are with Department of Computer Science and Engineering, Korea University, Seoul 136-701, Korea (e-mail: hyogon@korea.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2015.2420626

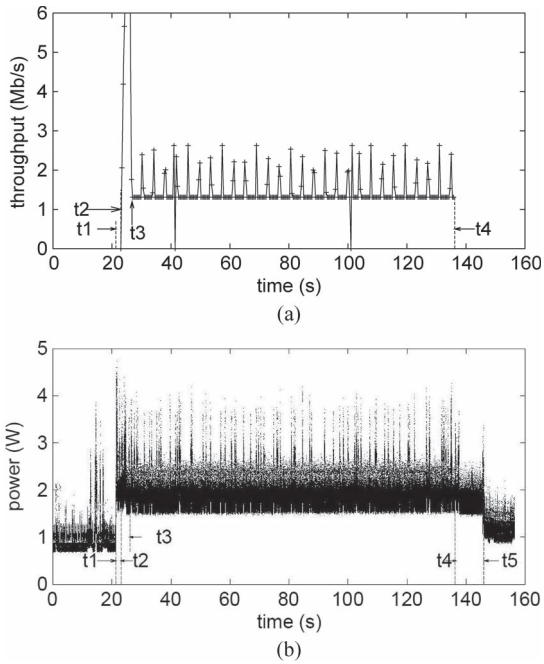

Fig. 2. Progressively downloading Gangnam Style from YouTube over LTE. (a) Video download rate. (b) Power consumption at the LTE user device.

TABLE I
AVERAGE POWER AND ENERGY CONSUMPTION

Modem active time (s)	Total		Modem		Others (Display *etc.*)	
	Avg. power (mW)	Batt. drain (uAh)	Avg. power (mW)	Batt. drain (uAh)	Avg. power (mW)	Batt. drain (uAh)
122.4	1851	17082	950	**8771**	901	8310

Fig. 2(a) shows the throughput during the progressive download. The LTE connection set up starts at $t_1 = 21.3$ s, and the video download begins at $t_2 = 23.4$ s. It ends at $t_4 = 135.8$ s, so the total download time is 112.4 seconds. As mentioned earlier, YouTube exhibits a peculiar variable bit rate (VBR) dynamics. In the first 3 seconds $(t < t_3 = 27.0)$, the server delivers the first 40 seconds worth of video, but throttles the download at 1.4 Mbps thereafter. At 360p, the encoded video bitrate is 724 Kbps, so the throttled rate is approximately twice the value. Fig. 2(b) shows the corresponding power consumption during the download. Before the modem is used (for $t < t_1$), the mobile consumes 901 mW on average, which is mostly incurred by display. Once the download begins, however, the power consumption approximately doubles. Finally, the Tail consumes slightly lower power ($t_4 < t < t_5$). After the Tail ($t > t_5$), the power consumption goes down significantly.

Table I breaks down the power consumption numbers during the progressive download. We see that the LTE modem remained active for 122.4 seconds, where it spent 112.4 seconds for data transmission and 10 seconds for the Tail. It consumed 8771 μAh of battery during the time. Although the download time more than halves the total play time of 252 seconds, it would be best in terms of power consumption not to throttle but to transmit the video in one-shot burst. However, it could be wasteful if the user aborts viewing [4]. Another issue arises

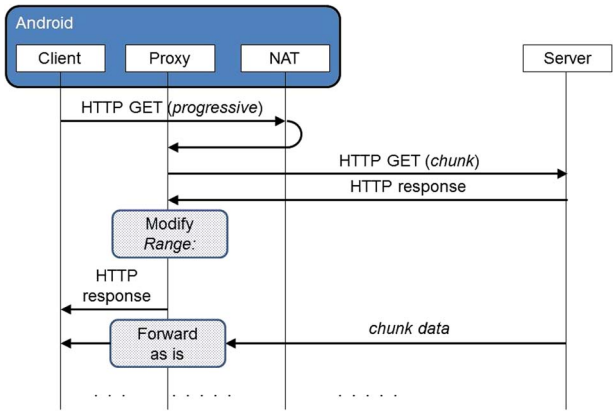

Fig. 3. Using a user-side proxy and a NAT (Android `iptables`) to implement the proxy for chunk download over LTE; alternatively, similar logic can be implemented in apps.

for the progressive download when we apply DRX. We need at least DRX inactivity time (e.g., 750 ms) between adjacent download activities so that the LTE modem can go into Idle state between them. Unfortunately, most inter-packet gaps in the throttled progressive download are smaller than the DRX inactivity time. DRX would not be effective in such situation.

There are prior works that attempt to reduce battery use by transforming low bit rate traffic into higher bit rate chunks. Siekkinen *et al*. [5] uses a network-side TCP-level proxy to accumulate stream data and form 5 to 7 seconds chunks. Besides the per-flow buffering and management burden, it can cause TCP retransmissions on the receiver side due to the large accumulation time. Li *et al*. [4] uses chunk download from mobile, where the chunk size is determined by user's video watching history. The chunk size is dynamically selected between 20 MB and 80 MB. But for some combinations of video resolution and streaming servers, these chunk sizes can be large on time axis and trigger throttling. For example, a 20 MB chunk will contain 235 seconds of Gangnam Style in 360 p and 59 seconds in 720 p. Recent dynamic video quality adaptation schemes such as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [6] and HTTP Live Streaming (HLS) [7] also use chunk download. However, the chunks in these schemes are defined to create the switch point between different quality video streams for smooth video quality adaptation, and this feature is agnostic to the last hop technology. As a matter of fact, typical chunk sizes used for them are 10 seconds (HLS) or less (DASH) [8], for which RRC inactivity timer will not expire so there is no energy saving since the LTE modem will be always on. There should be separate LTE-specific engineering for energy saving on LTE video streaming.

III. SOLUTION APPROACH

Our solution is to use a user-side proxy as in Fig. 3. It transparently transforms any progressive HTTP request into a series of HTTP chunk download requests. Note that this functionality can be similarly implemented directly inside an app in case the app needs to optimize for multiple metrics including energy saving. In this letter, we will take the proxy implementation case. To detour progressive download requests to the proxy,

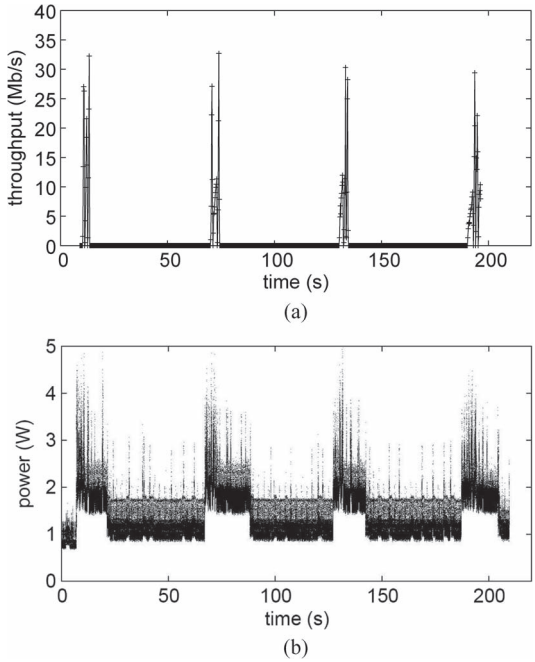

Fig. 4. Downloading Gangnam Style using the controlled chunk download over a commercial LTE network (power spikes between bursts are from background Android tasks). (a) Download rate. (b) Power consumption.

we activate the Android `iptables` command and configure it to forward the packet having popular streaming server IP addresses as the destination. When the proxy finds a HTTP GET request in the packet, it changes only the Range field into the chunk download format with desirable range values. As to responses, the first chunk data is given to the streaming client by the proxy as is. But from the second chunk response and on, the header of the corresponding HTTP chunk response is not forwarded but only the contents body is forwarded to the client as if it were the continuation of the progressive response. Therefore, the whole translation process is visible to neither the client application nor the streaming server nor the mobile service provider in between.

Since video streaming servers tend to automatically turn on throttling over a predefined continued download time (e.g., 40 seconds for YouTube), we limit the chunk size to a smaller value, e.g., 20 seconds. Optimal download size determination based on usage pattern is a difficult issue in itself [4]. For Gangnam Style, we divide its 252 seconds play time into thirteen 20-second chunks of 1.71 MB each that a separate HTTP GET request will fetch. Finally, if we requested an individual chunk at a time, every chunk download could entail a 10-second Tail. To reduce the number of Tails, therefore, we let the smartphone request 3 chunks back-to-back, minimizing the Tail periods in between. Sixty seconds later, we issue another three back-to-back GETs. This way, we neither trigger throttling nor incur excessive number of Tails.

IV. Performance Evaluation

We first evaluate the performance of the proposed scheme over a commercial mobile network without DRX. Fig. 4(a) shows the throughput in each burst jumps to a peak and then it quickly drops to zero, shortening the modem active time. Fig. 4(b) shows the corresponding power surge events are widely spaced according to the chunk download schedule. Note that each surge contains a 10-second Tail period. Table II summarizes the power and energy consumption for the four bursts. Notice that the average modem power consumption during bursts is 968.5 mW, similar to progressive download. But the total battery drain is 5014 μAh, 43% less than progressive download (see Table I). For another well-known video "Big Buck Bunny" from YouTube that runs longer (597 seconds) and uses a chunking technique similar to MPEG-DASH, we obtain 31% saving gain through our scheme.

To generalize the result in Table II, we need to consider the modem active time $T^{(ch)}$ as a function of the total play time T_{play} of the given video. The energy saving ratio is simply given as $S_E = 1 - E^{(ch)}/E^{(pr)}$ where $E^{(ch)}$ and $E^{(pr)}$ are the energy consumed by the controlled chunk download and the progressive download schemes, respectively. The energy used by the proposed scheme is

$$\begin{aligned} E^{(ch)} &= T^{(ch)} \times 968.5 \text{ mW} \\ &\approx \frac{T_{play}}{T_{burst}} \times \left(T_{tail} + \frac{L_{burst}}{B} \right) \times 968.5 \text{ mW} \end{aligned} \quad (1)$$

where B is the LTE bitrate available to the mobile device at the time of download, T_{burst} is the media time covered by each burst, and L_{burst} is the burst data size. For example, $T_{burst} = 3 \cdot 20\ s = 60\ s$ and $L_{burst} = 3 \cdot 1.71 \text{ MB} = 5.13 \text{ MB}$ for the Gangnam Style video. The approximation in (1) is because T_{play} is not necessarily an integral of T_{burst}. From the total modem active time of 71.9 s in Table II, we see that each burst length is $T_{tail} + L_{burst}/B \approx 18$ s. With $T_{tail} = 10$ s, the pure data transmission time in each burst is $L_{burst}/B = 8$ s in our measurement environment. As the LTE bitrate is not under our control, this value can vary. On the other hand, in the progressive download,

$$E^{(pr)} = T^{(pr)} \times 950 \text{ mW} = \left(\frac{T_{play}}{F_{th}} + T_{tail} \right) \times 950 \text{ mW}$$

where F_{th} is the download-to-play ratio on throttling, which is 2 in most streaming servers, including YouTube's. Then,

$$\begin{aligned} \frac{E^{(ch)}}{E^{(pr)}} &= \frac{\frac{T_{play}}{T_{burst}} \cdot \left(T_{tail} + \frac{L_{burst}}{B} \right) \times 968.5 \text{ mW}}{\left(\frac{T_{play}}{F_{th}} + T_{tail} \right) \times 950 \text{ mW}} \\ &\approx \frac{F_{th}}{T_{burst}} \cdot \left(T_{tail} + \frac{L_{burst}}{B} \right) \quad (2) \\ &= F_{th} \cdot \left(\frac{T_{tail}}{T_{burst}} + \frac{\Gamma}{B} \right) \quad (3) \end{aligned}$$

where we assume $T_{play} \to \infty$, and $\Gamma = L_{burst}/T_{burst}$ is the encoded video bitrate. We notice in Eq. (2) that the energy saving ratio in our measurement experiment is predicted here as $S_E = 1 - 2 \times 18/60 = 40\%$. We observe indeed that the energy saving gain is close to the measured value at 43% for $T_{play} = 252$ seconds. Note that the term $F_{th}/T_{burst} \cdot T_{tail}$ in

TABLE II:
AVERAGE POWER AND ENERGY CONSUMPTION WITH CHUNK BURSTS

	Modem active time (s)	Total		Modem		Others (Display)	
		Avg. power (mW)	Batt. drain (μAh)	Avg. power (mW)	Batt. drain (μAh)	Avg. power (mW)	Batt. drain μAh
Burst 1	15.1	1897	2058	997	1082	901	976
Burst 2	22.4	1838	2975	938	1518	902	1457
Burst 3	16.0	1868	2159	968	1119	899	1040
Burst 4	18.5	1871	2495	971	1295	901	1200
	71.9	1869.5	9687	968.5	**5014**	900.8	4673

Eq. (2) determines the upper bound of S_E, where L_{burst}/B further decreases it. The larger the LTE bitrate is, the smaller the decrease will be. Moreover, Eq. (3) implies that the only way to maximize the energy saving is to increase T_{burst} as other parameters are given by the video itself (Γ), by the streaming server (F_{th}), or the LTE network (T_{tail}, B). But again, increasing T_{burst} may burden the streaming server and increase the possibility of wasted download. However, we *can* obtain the lower bound of the T_{burst} as a function of T_{tail} and F_{th}. For any energy saving gain (*i.e.*, $S_E = 1 - \frac{E^{(ch)}}{E^{(pr)}} \geq 0$) from the proposed chunk download, we should have

$$\frac{T_{burst}}{T_{tail}} \geq F_{th} \cdot \frac{B}{B - F_{th} \cdot \Gamma}$$

from Eq. (3), where $F_{th} \cdot \Gamma$ turns out to be the throttling rate in the progressive download. Note that if the LTE bitrate far exceeds the throttling rate (*i.e.*, $B \gg F_{th} \cdot \Gamma$), the righthand side approaches F_{th}. Therefore, to make the chunk download profitable, T_{burst} should necessarily be at least $F_{th} \cdot T_{tail}$.

Now let us consider the DRX case. From Fig. 4, we notice that the proposed chunk download widens the gap between adjacent bursts so that DRX may take effect. Unfortunately, mobile service providers have not deployed DRX, so we cannot directly measure the DRX performance. So instead, here we compute S_E using the model in Fig. 1. We can approximate the T_{tail} under DRX by

$$T_{tail|_{DRX}} \approx \left(\frac{T_{tail}}{T_{cycle}}\right) \cdot T_{on} + T_{inact} \tag{4}$$

where T_{inact} is the DRX inactivity timer, T_{cycle} is the Long DRX cycle, and T_{on} is the DRX "on" duration. By substituting $T_{tail|_{DRX}}$ for T_{tail} in Eq. (3), we get

$$\frac{E^{(ch)}}{E^{(pr)}} = F_{th} \cdot \left(\frac{T_{tail}}{T_{burst}} \cdot \frac{T_{on}}{T_{cycle}} + \frac{T_{inact}}{T_{burst}} + \frac{\Gamma}{B}\right) \tag{5}$$

Comparing Eq. (5) with Eq. (3), we notice that the first fraction is scaled down by a factor of T_{on}/T_{cycle} whereas there is a new term T_{inact}/T_{burst} added. The former is typically a small value, e.g. $T_{on} = 20$ ms and $T_{cycle} = 640$ ms. The energy saving gain reduction by this term is just over 1% hence negligible. Also, the newly added fraction T_{inact}/T_{burst} is relatively small. For example, a typical value of $T_{inact} = 750$ ms, which makes the fraction just over 1%. Therefore, the energy saving gain under DRX comes to depend largely on the last fraction Γ/B. Eq. (4) gives us $T_{tail|_{DRX}}$ of 1063 ms, which is approximately one tenth of the T_{tail} without DRX. We easily notice that it significantly reduces the first fraction in Eq. (2), which leads to the increase of the upper bound of S_E. For instance, it would make $T^{(ch)} \approx 9$ s, half the active time without DRX. Consequently, $E^{(ch)}$ is halved compared to the non-DRX case in Eq. (1). Based on Eq. (5), we can obtain $S_E \approx 1 - 18/60 = 70\%$, a much larger energy saving gain than without DRX.

V. CONCLUSION

A simple modification of the HTTP GET requests on the LTE user device can achieve a significantly large energy saving gain. By understanding the power consumption dynamics of the LTE modem and exploiting the Range header field in the HTTP request, we can roughly halve the energy consumption to stream a typical video file. It does not require any special hardware on the network or the user device, but a simple HTTP translation software on the mobile. Through a prototype implementation on an Android smartphone and measurement over a commercial LTE service, we measure that 30–40% of battery saving can be achieved for representative video clips such as Gangnam Style and Big Buck Bunny, where it could be amplified up to 70% if DRX becomes available. By incorporating the proposed scheme as a proxy on the LTE user device or as part of the streaming apps on future smartphones, we will be able to significantly save battery on this increasingly popular application category of mobile streaming. Finally, the app-based implementation may be required in case the chunk size that applications (e.g. HLS or DASH) generate can be smaller than the burst size.

REFERENCES

[1] "Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification," 3rd Generation Partnership Project, Sophia Antipolis Cedex, France, 3GPP TS 36.331, Sep. 2012.
[2] "Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification," 3rd Generation Partnership Project, Sophia Antipolis Cedex, France, 3GPP TS 36.321, Sep. 2012.
[3] M. Hoque, M. Siekkinen, and J. Nurminen, "On the energy efficiency of proxy-based traffic shaping for mobile audio streaming," in *Proc. IEEE CCNC*, 2011, pp. 891–895.
[4] X. Li, M. Dong, Z. Ma, and F. Fernandes, "GreenTube: Power optimization for mobile video streaming via dynamic cache managment," in *Proc. ACM Multimedia*, 2012, pp. 279–288.
[5] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto. "Streaming over 3G and LTE: How to save smartphone energy in radio access network-friendly way," in *Proc. 5th Workshop MoVid*, 2013, pp. 13–18.
[6] MPEG, *Dynamic Adaptive Streaming over HTTP*, ISO/IEC 23009, Jul. 2013.
[7] R. Pantos, "HTTP Live Streaming," Work in Progress (Internet-Draft), Oct. 2014.
[8] V. K. Adhikari *et al.*, "Unreeling Netflix: Understanding and improving Multi-CDN movie delivery" in *Proc. IEEE Infocom*, 2012, pp. 1620–1628.