Power Efficient Mobile Video Streaming Using HTTP/2 Server Push

Sheng Wei Viswanathan Swaminathan Mengbai Xiao
Adobe Research
Adobe Systems Inc.
345 Park Ave, San Jose, CA 95110, USA
{swei, vishy, menxiao}@adobe.com

***Abstract*—This paper proposes a power efficient video streaming mechanism on mobile devices over cellular networks. We first develop an analytical model to identify and quantify the power inefficiency in mobile video streaming, due to the mismatch between HTTP request schedule and the radio resource control schedule. Based on the analytical model, we develop a low power video streaming mechanism by employing the server push technology available in the HTTP/2 protocol. We implemented the server push-based low power streaming mechanism in an HTTP DASH video streaming prototype involving mobile devices and the 4G/LTE cellular network. Our experiments show significant battery power savings on mobile devices using our server push strategy.**

I. Introduction

HTTP has been widely adopted as the protocol for video streaming over the Internet, because of its ease of deployment with existing web servers and scalability with the web caches [1][2]. More recently, with the rapid growth and popularity of smart phones and tablets, video streaming on mobile devices over cellular networks (e.g., 3G and 4G/LTE) have become a trend.

Unlike traditional video streaming platforms, mobile devices are greatly power constrained. For example, most of the newly released smart phones require nightly charging, making the battery power an extremely constrained resource compared to other components on the device. However, in a video streaming scenario, a large amount of data must be transmitted to the device via the wireless radio interface, which consumes a large amount of battery power.

Furthermore, the radio resource control (RRC) protocol [3], widely used in the cellular networks, specifies that the radio power would remain in the full power or half power mode after the data transmission has been completed, until the inactivity timers expire. The intent of the RRC protocol is to save the initialization delay and thus ensure premium network performance, especially when the user makes two consecutive network accesses within a short period of time. However, the "long tail" feature in the RRC protocol leads to significant power inefficiencies in the HTTP video streaming scenario [4][5]. In HTTP streaming, the video file is packaged into segments that are used as unit resources for HTTP requests and responses. During a video streaming session, the client maintains a buffer and issues a request to the next video segment whenever the buffer falls below a certain threshold. Consequently, at steady state, there is an HTTP request/response communication at an approximately fixed interval, which is typically the segment duration (e.g., a few seconds). In case the HTTP communication interval ends before the radio inactivity timer expires, the cellular radio would never go to the idle state (i.e., the state in which the radio power consumption is close to zero) during streaming, resulting in significant power inefficiency.

978-1-4673-7478-1/15/$31.00

Since the RRC protocol has already been widely deployed in today's cellular network infrastructure, and it was carefully designed and optimized for various use cases of the cellular network, it is not realistic to make changes to the RRC protocol itself just for the video streaming scenario. Instead, researchers have been focusing on adapting the HTTP streaming schedule at the application level to achieve power efficiency, such as the dynamic cache management [4] and the bundled download [5] approaches. Despite the effectiveness in lowering battery power consumption, the dynamic cache management approach [4] may require OS kernel changes, which is hard to deploy. The bundled download approach [5] may affect the client's ability to switch to a different bitrate in a timely manner. Also, the bundled multiple requests would hit the server in a short range of time causing potential scalability issues.

We develop a power efficient mobile video streaming mechanism by using the server push technology available in the HTTP/2 protocol [6]. In our prior work, we have developed a K-Push strategy, which makes the server actively push the next K segments, without requiring an individual request for each segment [9][10]. In this paper, we employ the K-Push strategy to change the HTTP request/response schedule, by manipulating the parameter K and matching it with the radio schedule for reduced power consumption. To summarize, our contributions in this paper include the following: (1) using HTTP/2 server push to achieve lower power consumption in mobile video streaming, while maintaining the scalability of HTTP streaming; (2) developing a power aware K-Push strategy and applying it to a DASH [2] streaming system; and (3) providing an analytical model to quantify the impact of parameter K in K-Push on power consumption.

II. Background

A. HTTP Streaming

HTTP streaming is a scalable and easy-to-deploy video streaming solution over the Internet [1][2], which leverages the existing web servers and content distribution networks (CDNs). In HTTP streaming, video content is divided into multiple small segments and deployed on the web servers for the client to request via HTTP. As shown in Figure 1(a), the client requests video segments periodically from the web server for video playback. The client maintains a video buffer and requests a segment whenever the content in the buffer falls below a certain threshold. At steady state, the client would consume one segment in the buffer within one segment duration, which requires that it issues HTTP requests at the interval of a segment duration for a smooth video playback.

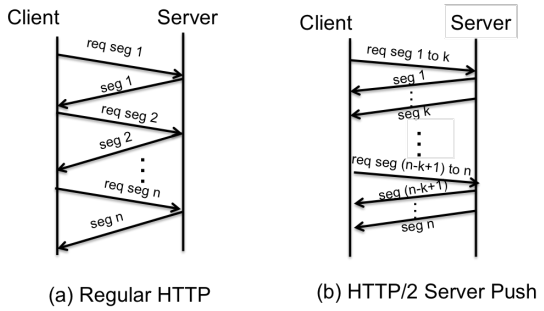

Fig. 1. Regular HTTP vs. HTTP/2 server push.

B. Radio RRC States

In cellular networks, the RRC protocol defines a state machine for the radio activity [3], as shown in Figure 2. In the RRC Idle mode, the power consumption is close to 0. Once data communication is initiated, the mobile device switches to a full power state (i.e., Dedicated Channel, or DCH) after a certain delay (i.e., delay 1 in Figure 2), namely promotion delay. Then, data can be transmitted via the radio link. Once the data transmission is completed, the radio interface would stay at the full power state until an inactivity timer (i.e., timer 1 in Figure 2) expires, during which time, the device still consumes full power. After the state transitions to the half power state (i.e., Forward Access Channel, or FACH), the power consumption reduces to approximately half of the full power state. While at the half power state, the device may either switch back to the full power state, after another promotion delay (i.e., delay 2 in Figure 2) if there is data transmission, or switch to the Idle state after another inactivity timer (i.e., timer 2 in Figure 2) expires.

In 4G/LTE, there is an improved RRC protocol with discontinuous reception (DRX) modes, which involves more sophisticated RRC state transitions and inactivity timers [7]. According to the 4G/LTE measurement results reported by Huang et al. [7], although there are three inactivity timers involved, there is only one timer (i.e., the tail timer) that is significant in terms of time and power consumption. Also, the power consumption in the tail period is not necessarily half of the full power mode. However, in our work, we consider the RRC states and power modes described in Figure 2 as a generic model for discussion. Our model can be adapted to the 4G/LTE case by adjusting the timer 1 and timer 2 values, as well as the power consumption levels in the half power mode.

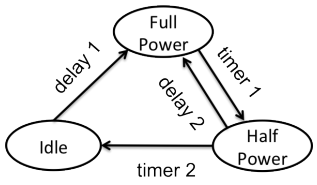

Fig. 2. Radio RRC state machine.

C. HTTP/2 Server Push Technology

HTTP/2 is a new revision to the currently used HTTP protocol, which is developed by IETF. It intends to enable a faster Internet by introducing a variety of new features, such as server push and stream multiplexing. Among all the new features, server push enables a server-to-client communication channel unlike the traditional client-pull only mechanism. As shown in Figure 1(b), the server push feature enables the web server to actively push HTTP resources to the client without requiring an explicit request for each resource. Although the server push technology was originally designed for reducing web page loading latency, it provides an elegant way of changing the HTTP request schedule in video streaming without compromising the scalability of HTTP streaming or making changes to the HTTP resources.

D. Related Work

There have been several research efforts that are focused on power efficient video streaming on mobile devices. Li et al. proposed a dynamic cache management scheme to address the power consumption issues in the scenario of HTTP progressive download [4] . Tian et al. developed rate adaptation algorithm for mobile devices considering both the TCP throughput and the battery consumption [5]. Their approach to adapting the HTTP request/response schedule to the radio activity is bundled download, where the client aggregates and requests multiple segments periodically. Nazir et al. suggest modifications to the TCP protocol in order to adapt to the HTTP video streaming schedule [8]. In our prior work [9][10], we have developed a K-push strategy for low latency live video streaming and cost effective video-on-demand streaming. This paper for the first time investigates the benefits of using HTTP/2 server push for low power video streaming over cellular networks.

III. Analytical Model

A. Overview

In order to optimize the power consumption during an HTTP streaming session, we first develop an analytical model to calculate the total power consumption (P). The model of

TABLE I
NOTATIONS FOR THE ANALYTICAL MODEL.

Notation	Definition
v	Video length
d	Video segment duration (the request interval)
r	Video bitrate
b	Available bandwidth
λ	Bitrate/Bandwidth ratio, $\lambda = r/b$
p	Power consumption per unit time
P	Total power consumption
w_1	Inactivity timer of the DCH mode
w_2	Inactivity timer of the FACH mode

P depends on both the HTTP streaming and the radio activity parameters, as outlined in Table 1.

Our goal in developing an analytical model is to calculate the total power consumption P during the video streaming session, in order to evaluate and optimize the power efficiency of video streaming over cellular network. We note that the value of P can be determined by analyzing the three cases (i.e., small, medium and large segment durations) presented in the next subsections, based on the relationship among segment duration d, bitrate/bandwidth ratio λ, and inactivity timers w_1 and w_2. Our analysis follows the HTTP streaming scheme described in Section II, where the client requests each individual segment (with duration d) at a fixed interval d at steady state. For each segment, the network download time is λd. After the download, even in the case without any more network activities, the radio interface of the client device would still remain active due to the RRC protocol, with power consumption rates p for duration w_1 and $p/2$ for duration w_2.

B. Case 1: Large Segment Duration

In the case where the segment duration d is larger than the download time and inactivity timers combined, as shown in Figure 3, d satisfies the following:

$$\lambda d + w_1 + w_2 < d < v \tag{1}$$

which is,

$$\frac{w_1 + w_2}{1 - \lambda} < d < v \tag{2}$$

The total power consumption P, is given by:

$$P = \frac{v}{d}[p(\lambda d + w_1) + \frac{p}{2} w_2] \tag{3}$$

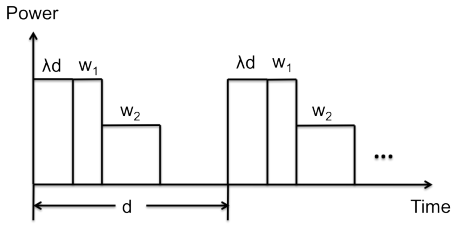

Fig. 3. Power consumption when a large segment duration (i.e., $\frac{w_1+w_2}{1-\lambda} < d < v$) is used.

C. Case 2: Medium Segment Duration

In the case where the end of segment duration d ends after w_1 and before w_2, as shown in Figure 4, d satisfies the following:

$$\lambda d + w_1 < d < \lambda d + w_1 + w_2 \tag{4}$$

which is,

$$\frac{w_1}{1 - \lambda} < d < \frac{w_1 + w_2}{1 - \lambda} \tag{5}$$

At the end of duration d, the client starts the next request cycle and thus the half power mode (i.e., w_2) is finished early and escalated to the full power mode. In this case, the total power consumption can be calculated as:

$$P = \frac{v}{d}[p(\lambda d + w_1) + \frac{p}{2}(d - \lambda d - w_1)] \tag{6}$$

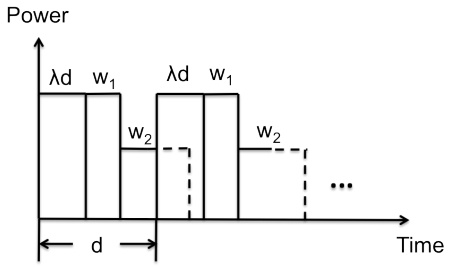

Fig. 4. Power consumption when a medium segment duration (i.e., $\frac{w_1}{1-\lambda} < d < \frac{w_1+w_2}{1-\lambda}$) is used.

D. Case 3: Small Segment Duration

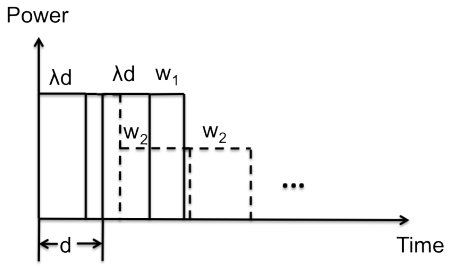

Fig. 5. Power consumption when a small segment duration (i.e., $\lambda d < d < \frac{w_1}{1-\lambda}$) is used.

In the case where duration d ends before the end of w_1, as shown in Figure 5, d satisfies the following:

$$\lambda d < d < \frac{w_1}{1 - \lambda} \tag{7}$$

which is,

$$0 < d < \frac{w_1}{1 - \lambda} \tag{8}$$

In this case, the radio remains in the full power state throughout the entire video streaming session and, therefore,

$$P = pv \tag{9}$$

E. Discussions of the Analytical Model

Putting together Cases 1, 2, and 3, we note that λ and d play important roles in power consumption. In particular, in Figure 6, we plot the total power consumption with varying d according to the analytical models described in Equations (1) to (9), where

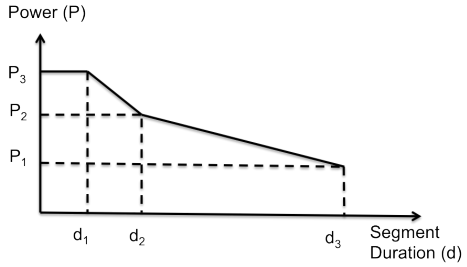

Fig. 6. Power consumption vs. segment duration.

$$P_1 = p(\lambda v + w_1 + \frac{w_2}{2}) \tag{10}$$

$$P_2 = \frac{pv}{\lambda d + w_1 + w_2}[\lambda^2 d + (1+\lambda)w_1 + (\frac{1}{2} + w_2)] \tag{11}$$

$$P_3 = pv \tag{12}$$

and

$$d_1 = \frac{w_1}{1-\lambda}; d_2 = \frac{w_1 + w_2}{1-\lambda}; d_3 = v \tag{13}$$

We observe from Figure 6 that we can achieve the lowest possible power consumption by assigning the largest possible segment duration, which equals to the entire video length (v). To be precise, the maximum power consumption savings that can be achieved by varying the segment duration d is the following:

$$\Delta P_{max} = P_3 - P_1 = p[(1-\lambda)v - w_1 - \frac{w_2}{2}] \tag{14}$$

In practice, ΔP_{max} can be a large positive number, with a large enough video duration v and the high bandwidth provided by the 4G/LTE network (5 to 12Mbps downlink speed, with peak speed approaching 50Mbps) [11]. For example, even for a short 5-minute 1080p movie trailer streamed at 3Mbps, with 5Mbps 4G/LTE speed and AT&T's typical radio modes (w_1=5 sec and w_2 = 12 sec) [12], the power saving can be up to $109p$, which is approximately the power consumption of streaming 1/3 of the original video content.

Despite the promising potential power savings, the use of large segments in HTTP streaming is ineffective in practice. For example, during bitrate switching, the client may have to delay the request of the next segment with new bitrate until the current large segment is played, or it may request the current segment with new bitrate immediately causing additional bandwidth consumption.

IV. Low Power mobile video streaming over HTTP/2

A. Server Push and K-Push Strategy

In order to reduce the power consumption in mobile video streaming, while still maintaining the benefits of HTTP streaming, we develop a new power efficient video streaming scheme employing the server push feature in HTTP/2. Our approach is based upon the K-Push strategy that we developed for low latency and cost effective video streaming [9][10], which enables the server to actively push multiple (i.e., K) segments to the client without requiring an individual request for each segment. In this work, we further develop a power-aware K-Push strategy for low power mobile video streaming. Our key observation is that by employing K-Push we can obtain an equivalent HTTP download schedule as compared to using large segment durations, without changing the segment duration. More importantly, the K-Push strategy enables us to maintain the HTTP streaming benefits, such as bitrate switching and scalability. For example, we can maintain the flexibility of bitrate switching by adapting the parameter K. Also, since it requires only one request for every K segments, the number of requests reaching the server is significantly reduced, which eliminates the scalability issues.

Figure 7 shows the end-to-end work flow of our K-Push strategy specifically designed for mobile video streaming. The mobile client determines the parameter K, i.e., the number of segments it expects the server to push, for the current push cycle. Then, the client signals the K value to the server, either via a light weight push trigger GET message, or piggybacking it as an HTTP header extension in the existing regular HTTP request for the first segment. Upon receiving the push trigger message, together with the value K, the server starts to push the indicated K segments without expecting an explicit request for each of them. The K pushed segments, while reaching the client, will be stored in the web cache. Thereafter, the client's request to the set of K segments will be served by the local web cache without hitting the network or server.

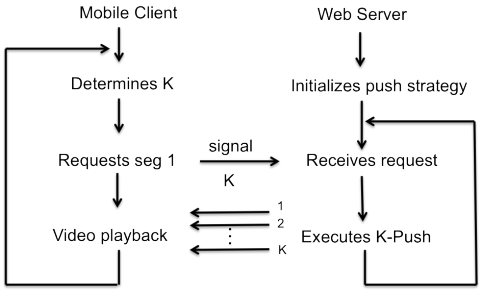

Fig. 7. K-Push strategy for mobile video streaming over HTTP/2.

B. Selection of K

From the work flow presented in Figure 7 we can observe that the K pushed segments, with duration d each, are equivalent to the regular HTTP request/response of a large segment, with duration Kd. Therefore, we can apply the same analytical model presented in Section III for determining the impact of

parameter K to power consumption. In particular, considering a commonly deployed fixed segment duration d_p, e.g., d_p = 2 sec, we have

$$K = \frac{d}{d_p} \tag{15}$$

Therefore, the power consumption model with varying K can be presented in Figure 8, where the power values P_1, P_2, and P_3 are the same as those in Equations (10) to (12), and the three boundary values k_1, k_2, and k_3 are the following:

$$k_1 = \frac{w1}{(1-\lambda)d_p}; k_2 = \frac{w_1 + w_2}{(1-\lambda)d_p}; k_3 = \frac{v}{d_p} \tag{16}$$

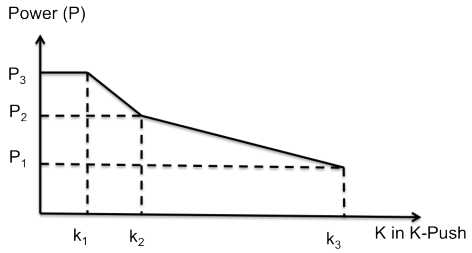

Fig. 8. Power consumption vs. parameter K.

It is worth noting that K must be larger than k_1 in order to obtain power savings in the mobile video streaming scenario. Therefore, $k_1 = \frac{w_1}{(1-\lambda)d_p}$ is the lower bound of determining a K for low power streaming. The power consumption decreases linearly as K increases from k_1 to k_2. The upper bound for K is k_3, which is essentially the total number of segments for the video. However, in practice, there is often a tighter upper bound of K determined by the constraints of bitrate switching delay and bandwidth overhead, as we have reported in our prior work [10]. For example, the bitrate switching delay would increase linearly with K, resulting in a tradeoff between power efficiency and the ability of HTTP adaptive streaming.

Furthermore, in the scenario of a sudden bandwidth drop, which is common in the cellular radio case, a large K would result in more higher bitrate segments being pushed under lower available bandwidth. It may cause additional congestion in the network, impacting the video streaming experience. In that case, we can design the K-Push strategy in an interruptive manner, where the currently pushed segments can be cancelled and a new (smaller) K can be applied immediately after the bandwidth drop is detected. Since the implementation of the "cancel" feature is still evolving and not yet supported by the existing browsers, we leave this as future work.

In summary, the K-Push strategy provides us with a flexible and non-intrusive method of adjusting the tradeoff between power efficiency (e.g., battery consumption) and video streaming experience (e.g., bitrate switching delay). Comparing to the traditional progressive download (i.e., equivalent to pushing all segments) or regular HTTP streaming (i.e., equivalent to no push), the K-Push strategy enables the content/service providers and/or the end users to adjust the key video streaming parameters at runtime for a premium and power-efficient video streaming experience.

C. Inactivity Timer Measurements

From Equation (16), we note that the boundaries of K values depend on the two radio inactivity timers w_1 and w_2, as well as the bitrate/bandwidth ratio λ. While λ can be determined and adjusted (via bitrate switching) by the client, w_1 and w_2 are carrier dependent and are unknown to the client. In order to provide a complete set of reference values for K, we must measure the inactivity timers.

Probably, the most accurate way to measure w_1 and w_2 is to use external power supplies and measuring tools [13], which can directly and precisely measure the power consumption over time during the video streaming session to determine the timer values. However, this is not practical in user devices used for video streaming. Instead, we employ the side-channel based approach as described in the literature [14]. The idea is to download the same file multiple times with predetermined intervals and measure the average load time of the file. By varying the interval incrementally and repeating the experiments, one can capture the promotion delays, included in the load time, when the radio state switches from FACH to DCH, and from Idle to DCH, respectively. The interval when the former switch occurs is approximately w_1, while that when the latter switch occurs is approximately $w_1 + w_2$.

V. Experimental Results

A. Experimental Setup

We implemented the proposed K-Push strategy in a DASH streaming system, as shown in Figure 9. The client is a Google Nexus 4 Android phone equipped with AT&T 4G/LTE cellular data service. We use the HTTP/2 enabled Google Chrome browser (Dev 45.0.2427.6) on Android to run the DASH IF dash.js player [15]. For the web server, we use the Jetty server (version 9.3.0), which supports HTTP/2 as well, with the addition of our K-Push strategy.

The web server is deployed on the public Internet, which is accessible through the 4G/LTE cellular network. The video we use for evaluation is the DASH test sequence made available by Telecom ParisTech [16]. We package the source video into main profile DASH streams, with various resolutions and segment durations.

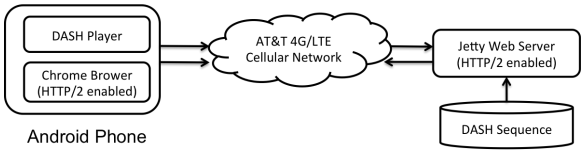

Fig. 9. Experimental Setup.

We evaluate the power savings obtained from our K-Push approach by streaming and playing back the DASH sequence on the phone and observing the power consumption. We measure the battery power by recording the battery information, including temporal current and temporal voltage, during video playback. In particular, we obtain the current (in uA) and voltage (in uV) values from certain device driver files. While video is playing, we run a bash script in the background and periodically sample the current and voltage data to a log file

on the device. Then, we calculate the temporal power as the product of voltage and current. To ensure the accuracy of the power measurements, we use a high sampling rate (i.e., approximately 1000 samples/minute) and take the average over all samples.

B. Inactivity timer measurements

We measure the inactivity timers using the side-channel based method described in the literature [14], to be able to determine a range of K values for lower power streaming. Figure 10 shows our measurement results[1]. We can observe that there are two significant increases of the load time, occurring at around the 5th second and the 21st second of intervals. According to our discussion in Section IV, this indicates that the approximate value of w_1 is around 5 seconds, and that of w_2 is around 16 seconds.

The measured w_1 and w_2 values in this step are important input parameters to the analytical model of K selection, as presented in Equation (16). Although the accuracy of these measurements is not guaranteed, and since it is dependent upon various factors, we use the measured values only as an approximate reference to guide us in selecting K.

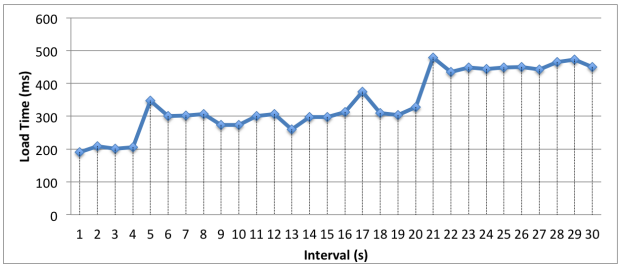

Fig. 10. Measurements for the inactivity timers.

C. Power Savings

We evaluated 7 cases of K selection (i.e., K=1, 6, 10, 15, 20, 25, and 30), in order to cover the different ranges of linear segments shown in Figure 8. Table 2 shows the absolute battery power consumption values, as well as the relative power savings comparing K-Push to the No-Push case. We observe that there is up to 17% savings in the battery power by using our K-Push (e.g., K=30) approach. Also, the power consumption follows a trend of decreasing with the increase of parameter K, which approximately matches with our analytical model shown in Figure 8.

VI. Conclusion

We developed an HTTP/2 server push-based approach to reduce the power consumptions in video streaming onto mobile devices. Our approach achieves low power consumption during mobile video streaming by having the server actively push K segments to the client, which changes the HTTP request schedule to make power efficient use of the wireless radio links. We presented both our analytical model and empirical results of power consumption with regard to the selection of parameter K in the push strategy. Our results indicate that we can achieve an up to 17% of battery power savings by using the proposed approach.

[1]We conducted this measurement using the Verizon 4G/LTE network on a Samsung Galaxy S5 handset earlier. The inactivity timer values of this network could be different from the video streaming experiments described in Section V.A. due to the network (i.e., AT&T) and the handset (i.e., Nexus 4) used. We assume that this is not significant enough to impact the results.

TABLE II
Power savings using the K-Push strategy. The test sequence is a 5-minute long DASH stream packaged with 2-second segments.

Push Strategy	Average Power (mW)	Power Savings
K = 1 (No Push)	2777.73	(Baseline)
K = 6	2572.47	7.4%
K = 10	2504.30	9.8%
K = 15	2351.73	15.3%
K = 20	2294.98	17.4%
K = 25	2280.52	17.9%
K = 30	2280.65	17.9%

References

[1] V. Swaminathan, Are we in the middle of a video streaming revolution? ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 9, No. 1s, Article 40, 2013.

[2] Dynamic adaptive streaming over HTTP, International Standard, ISO/IEC 23009-1, April 2012.

[3] 3rd Generation Partnership Project (3GPP), Radio Resource Control (RRC) Protocol Specification, Tech. Spec. 25.331, 2006.

[4] X. Li et al. GreenTube: Power Optimization for Mobile Video Streaming via Dynamic Cache Management, ACM Multimedia 2012, pp. 279-288.

[5] G. Tian et al., On Adaptive HTTP Streaming to Mobile Devices, Packet Video Workshop 2013, pp. 1-8.

[6] M. Belshe et al., Hypertext Transfer Protocol Version 2 (HTTP/2), RFC7540, available online: https://tools.ietf.org/html/rfc7540

[7] J. Huang et. al., A Close Examination of Performance and Power Characteristics of 4G LTE Networks, MobiSys 2012, pp. 225-238.

[8] S. Nazir, et al. Performance Evaluation of Congestion Window Validation for DASH Transport, NOSSDAV 2014, pp. 67.

[9] S. Wei, V. Swaminathan, Low Latency Live Video Streaming over HTTP 2.0, NOSSDAV 2014, pp. 37.

[10] S. Wei, V. Swaminathan, Cost Effective Video Streaming Using Server Push over HTTP 2.0, IEEE MMSP 2014.

[11] 4G LTE Speeds vs. Your Home Network, available online: http://www.verizonwireless.com/insiders-guide/network-and-plans/4g-lte-speeds-compared-to-home-network/

[12] A. Gerber et al., A Call for More Energy-Efficient Apps, available online: http://www.research.att.com

[13] Monsoon power monitor, available online: http://www.msoon.com/LabEquipment/PowerMonitor/

[14] Making a mobile connection, available online: http://www.stevesouders.com/blog/2011/09/21/making-a-mobile-connection/

[15] dash.js, DASH Industry Forum, available online: https://github.com/Dash-Industry-Forum/dash.js

[16] DASH Sequences, available online: http://gpac.wp.mines-telecom.fr/2012/02/23/dash-sequences/