Saving Energy in Mobile Devices for On-Demand Multimedia Streaming – A Cross-Layer Approach

MOHAMMAD ASHARFUL HOQUE, MATTI SIEKKINEN, JUKKA K. NURMINEN, SASU TARKOMA, and MIKA AALTO, Aalto University

This article proposes a novel energy-efficient multimedia delivery system called EStreamer. First, we study the relationship between buffer size at the client, burst-shaped TCP-based multimedia traffic, and energy consumption of wireless network interfaces in smartphones. Based on the study, we design and implement EStreamer for constant bit rate and rate-adaptive streaming. EStreamer can improve battery lifetime by 3x, 1.5x, and 2x while streaming over Wi-Fi, 3G, and 4G, respectively.

Categories and Subject Descriptors: C.2.1 [**Network Architecture and Design**]: Wireless Communication; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Performance of Systems**]: Measurement Techniques, Design Studies, Performance Attribute

General Terms: Design, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Constant bit rate, cross-layer, DASH, energy efficiency, multimedia streaming, radio signaling, rate-adaptive streaming, video streaming, wireless network

ACM Reference Format:

Mohammad Asharful Hoque, Matti Siekkinen, Jukka K. Nurminen, Sasu Tarkoma, and Mika Aalto. 2014. Saving energy in mobile devices for on-demand multimedia streaming – A cross-layer approach. ACM Trans. Multimedia Comput. Commun. Appl. 10, 3, Article 25 (April 2014), 23 pages.
DOI: http://dx.doi.org/10.1145/2556942

1. INTRODUCTION

On-demand multimedia streaming services, such as Spotify, Netflix, Dailymotion, Vimeo, and YouTube, have gained great acceptance among smartphone users. It is reported that there are six billion hours of YouTube video views every month and 25% comes from mobile devices [YouTube 2013]. However, the battery life of smartphones becomes critical when accessing these multimedia services via wireless networks (e.g., Wi-Fi, 3G, and 4G). Most often the power consumption of mobile devices for using these wireless networks is equivalent to or higher than the playback power consumption [Hoque et al. 2013b]. This is because there is continuous flow of data packets during streaming and a smartphone must keep the Wireless Network Interface (WNI) always active for receiving those packets. In this

This work was supported by the Academy of Finland: grant no. 253860 and FIGS.

Authors' addresses: M. A. Hoque (corresponding author), M. Siekkinen, J. K. Nurminen, S. Tarkoma, and M. Aalto, Department of Computer Science and Engineering, Aalto University School of Science, Aalto University, Finland; email: mohammad.hoque@aalto.fi.

DOI: http://dx.doi.org/10.1145/2556942

work, we optimize the wireless communication energy spent by mobile devices for TCP-based multimedia streaming.

It is well known that shaping multimedia traffic into periodic bursts can save energy, specifically for UDP-based multimedia traffic shaping over Wi-Fi [Korhonen and Wang 2005]. A number of packets are collected over a period of time and then sent together as one burst to the client using all the available bandwidth. In this way, the WNI is kept active only for a short period of time to receive the burst, instead of keeping the interface always active. When the burst size exceeds the playback buffer, there is packet loss. Therefore, the maximum size of a burst, or equivalently the length of a burst interval, is tuned based on an acceptable range of packet loss [Korhonen and Wang 2005].

Today HTTP over TCP is by far the most prevalent set of protocols used for streaming [Guo et al. 2006]. Traffic shaping saves energy with TCP-based streaming as well [Hoque et al. 2011]. TCP-based streaming differs from the case of UDP-based streaming in one major way: TCP is a reliable protocol. When the player buffer becomes full, data accumulates into the TCP receive buffer. As a result, an arbitrarily large burst, which is larger than the playback buffer, can be used without any packet loss. However, the exact impact on the energy consumption of smartphones is not well understood. In this article, we first study the interplay between burst size and power consumption of smartphones. Specifically, we model the energy consumption of WNIs for bursty TCP traffic. We show that the power consumption of a smartphone decreases when the receiving burst size increases and as long as the client device can accommodate the entire burst. In contrast, the power consumption rapidly increases if the burst size is too large compared to the available buffer space at the client.

As a proof of concept, we design and implement an energy-efficient multimedia delivery system called EStreamer. It determines an energy-optimal burst size so that smooth playback of the streaming applications is not distorted. EStreamer relies on standard TCP feedback from the client. We thoroughly evaluate EStreamer through measurements with real streaming services. We focus on two aspects: (a) the potential to save energy in smartphones using Wi-Fi, 3G (HSPA), and 4G (LTE); and (b) the impact on the Radio Access Network (RAN) signaling load when using a cellular access network.

Concerning the first aspect, we measure the energy savings in four different smartphones while streaming from popular streaming services via EStreamer, such as Internet radio, YouTube, and Dailymotion. The results demonstrate that the largest energy savings can be achieved with Wi-Fi (65%), followed by 4G (50–60%), after which comes 3G (38%). It is also shown that a client can achieve a similar range of energy savings when streaming from a rate-adaptive streaming service which is powered by EStreamer.

As for the second aspect of evaluation, the energy savings with traffic shaping arise from the ability of the radio interface to transition to lower-power consuming states in between two consecutive bursts of content. The state transitions, specifically when using a cellular access network (HSPA or LTE), require signaling message exchange between smartphone and network. The amount of signaling traffic can be a severe problem for network operators. Globally many major operators have suffered service quality deterioration and even network outages because of the signaling storms created by smartphones [LLC Signals Research Group 2010]. Consequently, we measure signaling load in the network when applying traffic shaping with different parameter settings in the network. In summary, this article makes the following contributions.

(1) We uncover the relationship between available buffer space at the client, burst size, and power consumption in TCP-based multimedia traffic shaping. Specifically, we develop novel power consumption models for bursty multimedia traffic over TCP to derive a heuristic for energy-optimal burst size for a given client. We show that our models are independent of the kinds of streaming for which it is applied.

(2) We design and implement a cross-layer multimedia delivery system called EStreamer which implements the heuristic. The heuristic makes EStreamer agnostic to device variation and the wireless access (Wi-Fi, 3G, or 4G) used by a smartphone for multimedia streaming. EStreamer has built-in support also for rate-adaptive streaming in addition to constant quality streaming.

(3) We evaluate EStreamer in a number of scenarios focusing on the amount of energy saved and the impact on the RAN signaling load. Our study covers four smartphones, both audio and video streaming services, and cellular networks with different configurations. The results show that network parameter values have no impact on the energy consumption when streaming audio/video, whereas they have a major impact when EStreamer is used. We show that there are differences in terms of signaling load with optimization mechanisms that provide a similar amount of energy savings. For HSPA, the best choice is to use shorter inactivity timers with traffic shaping. However, when discontinuous reception mechanisms are available, such as DRX in LTE, those mechanisms together with long inactivity timers are the best choice.

2. ENERGY EFFICIENCY OF WIRELESS MULTIMEDIA STREAMING OVER TCP

In this work, we focus on the energy consumption of wireless network interfaces for TCP-based multimedia streaming. We first outline the power consumption characteristics of Wi-Fi and cellular network interfaces in smartphones. Then, we describe the characteristics of mobile multimedia streaming. Finally, we explain how burst-shaped multimedia delivery can save energy.

2.1 Energy Consumption of Wireless Network Interfaces

Some inactivity timers control the energy consumption of different types of wireless communication. Because of these timers, there is residual time spent by any interface in active state after each transmission or reception for the sake of user experience, which leads to some energy spent doing nothing useful. We refer to this energy as *tail energy* [Balasubramanian et al. 2009]. Shortly, we discuss the power-saving mechanisms for wireless radios in smartphones to access the three most commonly used wireless networks, namely Wi-Fi, HSPA, and LTE.

2.1.1 *Wi-Fi.* The 802.11 standard includes a Power-Saving Mechanism (PSM). Commercial mobile phones typically implement a slightly modified version of it which is sometimes referred to as PSM-Adaptive (PSM-A). This version differs from the original one proposed by the standard in that it keeps the interface in the *idle* state for a certain fixed period of time (e.g., 200 ms) instead of switching to the *sleep* state immediately after the transmission or reception of traffic [Hoque et al. 2012]. This amount of time also determines the amount of tail energy in Wi-Fi communication. Usually, the power draw in sleep state is an order of magnitude lower than in idle state. In both receive and transmit states, the Wi-Fi interface may consume up to 50% more power than in the idle state.

2.1.2 *HSPA/3G.* HSPA is the widely deployed 3G access technology. In the network, the usage of radio resources and power consumption of a mobile phone is controlled by the Radio Resource Control (RRC) protocol. This protocol has four different states. Figure 1 illustrates the RRC state machine and the inactivity timers that control the transitions among these states. These timer values are in the order of several seconds and controlled by the network operators. A mobile device terminates RRC connection after the CELL_PCH→IDLE transition which happens upon expiry of the T3 timer. Some operators may disable CELL_PCH in their network and in that case the device would terminate the RRC connection when the T2 timer expires. The figure also shows that operating in different states draws different amounts of current. Fast Dormancy (FD) tries to reduce the tail energy effect as the mobile phone can request the network to transition directly CELL_DCH→CELL_PCH (FD Rel-8) or it may terminate the RRC connection and directly end up in idle mode (typical for legacy FD).

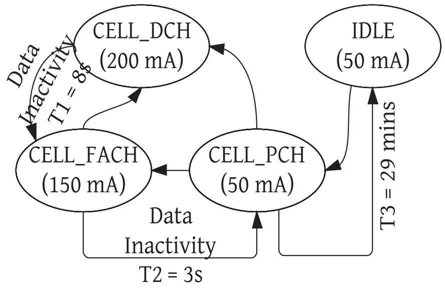


Fig. 1. 3G HSPA state transitions and power consumption.

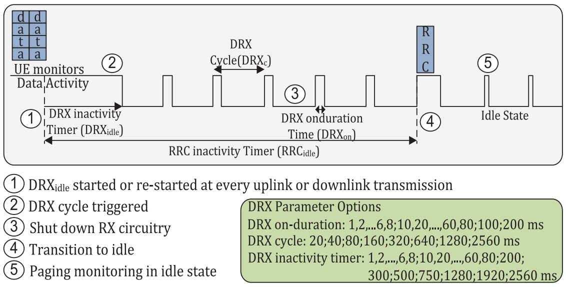

d a t a
d a t a
UE monitors
Data Activity
DRX
Cycle(DRX$_c$)
R R C
DRX inactivity
Timer (DRX$_{idle}$)
DRX onduration
Time (DRX$_{on}$)
Idle State
RRC inactivity Timer (RRC$_{idle}$)
1
2
3
4
5
① DRX$_{idle}$ started or re-started at every uplink or downlink transmission
② DRX cycle triggered
③ Shut down RX circuitry
④ Transition to idle
⑤ Paging monitoring in idle state
DRX Parameter Options
DRX on-duration: 1,2,...6,8;10,20,...,60,80;100;200 ms
DRX cycle: 20;40;80;160;320;640;1280;2560 ms
DRX inactivity timer: 1,2,...,6,8;10,20,...,60,80;200;
300;500;750;1280;1920;2560 ms

Fig. 2. LTE RRC with connected state DRX.

Different state transitions require some signaling message exchange between a mobile phone and the network. When a device is disconnected from the RRC, the reconnection requires comparatively a lot more signaling than for normal state changes [NSN 2011]. If the device supports only legacy FD or the network does not support CELL_PCH then there will be more signaling in the network as they both require RRC reconnection upon data activity.

2.1.3 *LTE/4G.* LTE (Long-Term Evolution) is the fastest growing 4G cellular network technology, with commercial networks having already been deployed in many countries. The LTE RRC protocol contains two states: RRC_IDLE and RRC_CONNECTED. Similar to the HSPA, there is an inactivity timer with a typical value of 10 s associated to the transition from the connected to the idle state which may cause a significant amount of tail energy to be spent. However, LTE includes a discontinuous reception mechanism specifically to be used in the RRC_CONNECTED state, hence called connected state DRX (cDRX) which can drastically reduce the tail energy. Figure 2 illustrates this mechanism. The idea is that after no packets have been received for a time period specified by the cDRX inactivity timer (DRX$_{idle}$), the device starts a duty cycle so that it wakes up only periodically (every DRX$_c$) for DRX on-duration specified amount of time (DRX$_{on}$) to check for new incoming packets. If no packets are received during an interval specified by the inactivity timer (RRC$_{idle}$), the device transitions from RRC_CONNECTED to RRC_IDLE state. In this way, there are only two states and thus less signaling due to state transitions in the network.

2.2 Mobile Multimedia Streaming

In this section, we describe the key properties of on-demand multimedia content and the techniques used by the streaming services to deliver content to the client. This is because some of them have direct impact on the energy consumption of smartphones and thus influence the design of our energy-efficient streaming.

2.2.1 *Playback Duration, File Size, Formats, and Quality.* The key properties of on-demand content are the content length, file size, format, and the bit rate that is an indicator for the playback quality. In the case of audio, the duration of music files are within the range of 3–5 minutes. However, ShoutCast-like services give support for Internet radio stations to provide continuous audio streaming. A number of studies have investigated these properties at the Video-on-Demand (VoD) streaming Web sites, such as YouTube. Cheng et al. [2013] described that the distribution of video length has three peaks. The first peak is within one minute of playback and consists of more than 20% of the videos. The second peak is between 3 and 4 minutes and contains about 16.7% of the videos. The third peak is much smaller and it is near the length of 10 minutes. Thus VoD content typically has a playback duration of several minutes. Recent studies also show that video viewing abandonment affects the video viewing time distribution [Hwang et al. 2012]. This means that the actual viewing time can be less than the video length. Later in Section 4.1, we will see how the length of videos influences in selecting a binary search algorithm for energy-efficient streaming.

The file sizes were also observed to follow a similar distribution to the video length and be small, typically less than 30MB. Specifically, 87.6% of the crawled videos used CBR with most videos having a bit rate around 330 kbps and two other peaks at 285 kpbs and 200 kpbs [Cheng et al. 2013]. Many different combinations of the container, encoding algorithm's bit rate, quality, and resolution are in use today. In mobile multimedia services, the most popular combination is H.264-based MP4-360p [Finamore et al. 2011], where MP4 is the container and 360p is the resolution. Other popular containers are FLV, WebM, and 3GPP. In Section 3, we make an analytical study of how different bit rates affect the energy consumption. In our experiments, we also use audio/videos of similar bit rates and show how they are related with the energy consumption (refer to Sections 5.1 and 5.2). FLV and WebM containers are exclusively used by the Flash and HTML5-based players, respectively, in Web browsers [Hoque et al. 2013b]. However, the iOS devices do not support FLV.

2.2.2 *Buffering and Adaptive Streaming.* Streaming services need to deal with bandwidth fluctuation and jitter which are prevalent on the Internet. Therefore, streaming clients do initial buffering of the content at possible maximum rate. This phase is also known as Fast Start. After this phase, different approaches are used to send content to the streaming clients over TCP. Some services send content at the encoding rate to serve more clients. Services like YouTube throttle the sending rate a little bit higher than the encoding rate to reduce the number of playback pause events due to bandwidth fluctuation. So-called Fast Caching downloads the whole content as fast as possible, which can eventually be very resource wasteful when the whole content is not consumed by the user. Guo et al. [2006] proposed to use a buffer-adaptive mechanism that downloads content to refill the playback buffer only when it touches a low-level threshold. This mechanism creates an ON-OFF traffic pattern. Hoque et al. [2013b] identified that video players in Android devices use such a method.

The methods described previously are used by video streaming services, such as YouTube and Dailymotion, which cannot switch the video stream bit rate, that is, quality, on-the-fly according to the available bandwidth. On the other hand, Vimeo, Netflix, and a number of IPTV services use rate-adaptive streaming in order to avoid playback pause events which annoy clients and reduce user engagement. These services apply different adaptive protocols, such as Apple's HTTP Live Streaming (HLS), Microsoft's Smooth Streaming (MSS), etc. In Section 4.2, we design algorithms to deal with bandwidth fluctuation and to provide energy savings for both CBR and rate-adaptive streaming.

2.3 Delivering Multimedia Content in Bursts Using TCP

In Section 2.2, we have described that a large fraction of the content has the bit rate of a few hundred kilobits per second. On the other hand, smartphone users enjoy 2Mbps mobile network speed on average [Cisco 2013]. Therefore, there is typically more bandwidth available on the path between the client and server than the encoding rate of the content. The WNI will be less utilized if the same amount of content can be received in bursts at the maximum possible rate instead of continuously receiving at the encoding rate. Figure 3 shows that if the encoding rate is r_s and BS amount of data is collected during a period of duration T and then sent to the client, the client's WNI is idle most of the time and can reside in a low-power state until the next burst is received. Thus, receiving the multimedia stream in the form of periodic bursts is attractive and causes the tail energy to be suffered by the client only once per each burst, as opposed to being present with each packet when receiving a constant bit rate streaming. Streaming clients that use a rate-adaptive protocol also retrieve content in a similar bursty manner with a periodic interval of 2–40 seconds.

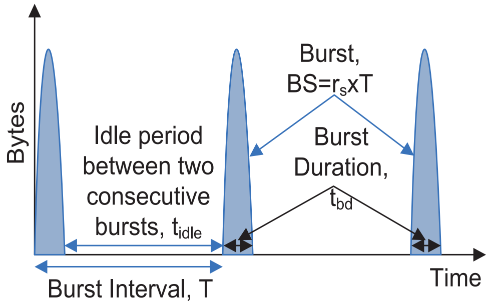


Fig. 3. Bursty traffic pattern and properties.

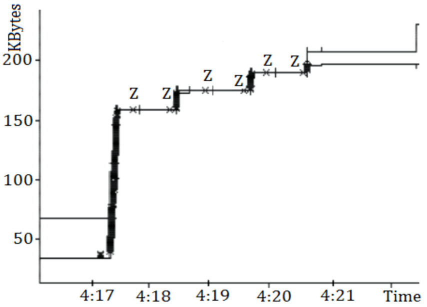


Fig. 4. Burst split in constant bit rate streaming.

There is a caveat. Multimedia players maintain a fixed-size playback buffer. The size depends on the implementation of the player and can be restricted by the operating system of the smartphone. If the size of a burst is larger than the available space in the playback and TCP receive buffer together, then TCP flow control becomes active at the client. Since the player decodes content at the encoding rate, TCP flow control and this player behavior together ensure that the excess bytes are received separately at a lower average rate. Huang et al. [2012] also observed similar behavior for rate-adaptive streaming. From their traffic traces, we have identified that switching to an upper quality (>1500 kbps) triggers TCP flow control at the client when streaming from Hulu. Figure 4 demonstrates how TCP flow control at the client controls the receiving of an oversized burst when the burst interval is of 10 s for CBR. “Z” in this figure indicates zero window advertisement from the client. The activation of TCP flow control leads to increased energy consumption, as we will show in the next section.

3. POWER MODELING AND ANALYSIS OF BURSTY STREAMING

Based on the bursty streaming behavior and energy consumption characteristics described in the previous section, we now develop power consumption models for delivering streaming content in bursts. We use the models to study the power consumption of different burst sizes in different scenarios. We identify the optimal burst size and quantify the potential energy savings and losses using the optimal and a nonoptimal size, respectively. In the previous section, we explained that the limited buffer causing activation of TCP flow control can take place during both constant-quality and adaptive streaming. Therefore, the models we develop are also applicable for both kinds of streaming.

3.1 Parameters

The parameters used in the following equations are described in Table I. We consider the power consumption to be fixed when actively receiving data at a given rate [Xiao et al. 2010], and we assume that the average stream encoding rate (r_s) is always lower than the TCP bulk transfer capacity (r_{btc}), that is, there is some spare bandwidth to exploit. Following the earlier discussions, we consider the limited buffer size of the client (B) and look at two corresponding cases: (i) the total buffer size is sufficiently large to fit an entire burst and (ii) the buffer gets full when receiving a burst and TCP flow control is activated. Power consumption for executing the application and baseline idle power of the WNIs (i.e., sleep power) are excluded from the models as they merely add a constant overhead.

3.2 The Receiver Buffer is Sufficiently Large

First, we consider the case when the fixed-size burst can be entirely absorbed by a streaming client, namely $r_s T \leq B$. We obtain the average power draw for a given T according to Eq. (1), which comprises

Table I. Description of the Parameters Used in the Modeling and Describing EStreamer

Description	Parameter	Description	Parameter
Avg stream encoding rate	r_s	Fast Start Streaming Period	t_{fs}
TCP bulk transfer capacity to client	r_{btc}	Data Transferred during t_{fs}	L
Data Transferred during t_{fs}	$L = r_{btc} \cdot t_{fs}$	Current burst interval	T
Power when receiving data at rate r	$P_{rx}(r)$	Burst Duration	t_{bd}
Power increase at rate r	$\Delta P_{rx}(r) = P_{rx}(r) - P_s$	Energy-optimal burst interval	T_{opt}
Inactivity timer values	T_1, T_2	Burst interval that avoids starvation	T_{max}
Power draw in tail energy states	P_1, P_2	Idle time between two consecutive bursts	t_{idle}
Avg power while spending tail energy(radio on but no rx/tx)	P_{tail}	Current lower bound while autotuning T	T_{min}
		Burst size	$BS = T \times r_s$
Tail energy	$E_{tail}(T)$	Optimal Burst Size	BS_{OPT}
		Available buffer space at the client	B

For Wi-Fi, P_{tail} is the power draw in an idle state, that is, radio on but no rx/tx. For HSPA, it is the combination of the power draw in CELL_DCH and CELL_FACH states depending on the timer values and t_{idle}. In case of LTE (with DRX), it is the power draw from the DRX inactivity timer.

the burst download power and the power drawn by the tail, and its derivative with respect to T according to Eq. (2).

$$P(T) = \frac{T r_s \Delta P_{rx}(r_{btc})}{T r_{btc}} + \frac{E_{tail}(T)}{T} \tag{1}$$

$$\frac{dP}{dT} = \frac{1}{T}\frac{dE_{tail}(T)}{dT} - \frac{E_{tail}(T)}{T^2} \tag{2}$$

To compute the derivative of $E_{tail}(T)$, we need to consider the impact of different inactivity timers on the tail energy. This energy is not always fixed but depends on T when the idle time in between receiving two consecutive bursts ($t_{idle}(T)$) is smaller than the sum of the inactivity timers, which can happen especially with 3G. In Eqs. (3)–(6), we compute $E_{tail}(T)$ and its derivate in three different cases, because we have a maximum of two timers (the case of HSPA), and apply them to Eq. (2) (we skip the straightforward manipulation steps). Wi-Fi and LTE cases only have one inactivity timer and hence the resulting models can be applied to them trivially by simplifying the equations so that T2 = 0.

$$0 < t_{idle}(T) < T_1 : \; E_{tail}(T) = P_1 t_{idle}(T) = P_1 T\left(1 - \frac{r_s}{r_{btc}}\right) \implies \frac{dE_{tail}(T)}{dT} = P_1\left(1 - \frac{r_s}{r_{btc}}\right) \implies \frac{dP}{dT} = 0 \tag{3}$$

$$T_1 < t_{idle}(T) < T_1 + T_2 : \; E_{tail}(T) = P_1 T_1 - P_2 T_1 + P_2 T\left(1 - \frac{r_s}{r_{btc}}\right) \implies \frac{dE_{tail}(T)}{dT} = P_2\left(1 - \frac{r_s}{r_{btc}}\right) \tag{4}$$

$$\implies \frac{dP}{dT} = \frac{T_1}{T_2}(P_2 - P_1) < 0, \text{ since } P_2 < P_1 \tag{5}$$

$$T_1 + T_2 < t_{idle}(T) : \; E_{tail}(T) = P_1 T_1 + P_2 T_2 \implies \frac{dE_{tail}(T)}{dT} = 0 \implies \frac{dP}{dT} = -\frac{E_{tail}(T)}{T^2} < 0 \tag{6}$$

The preceding result shows that the power draw either stays the same or decreases when T is increased as long as player buffer and TCP buffer together can hold the whole burst. Thus, the larger the value of T, the less energy is consumed by the client device.

3.3 Burst Size Exceeds TCP Receive Buffer Size

Now, we consider the other case where a burst is larger than the available buffer space at the client, namely $r_s \times T > B$. In this case, the portion of the burst that exceeds the buffer size can be transmitted to the receiver at an average rate of r_s because that is the rate at which the application consumes data from the buffer. Note that tail energy is no longer dependent on T and is consumed over a fixed-length interval since the time it takes to transmit the whole burst grows at the same rate at which T is increased (transmission rate equal to encoding rate). The reader can easily check this for each of the three different cases of tail energy as we did in Eqs. (3)–(6). Thus, we treat E_{tail} as a constant. We obtain Eqs. (7) and (8) for the power consumption and its derivative. They now comprise three parts: the power drawn during download of the portion of the whole burst that fits the receiver's buffer at TCP bulk transfer rate, the power draw during download of the leftover data at stream encoding rate, and power drawn by the tail.

$$P(T) = \frac{B\Delta P_{rx}(r_{btc})}{T r_{btc}} + \frac{(T r_s - B)}{T r_s} \Delta P_{rx}(r_s) + \frac{E_{tail}}{T} \tag{7}$$

$$\frac{dP}{dT} = \frac{B}{T^2}\left(\frac{\Delta P_{rx}(r_s)}{r_s} - \frac{\Delta P_{rx}(r_{btc})}{r_{btc}}\right) - \frac{E_{tail}}{T^2} \tag{8}$$

The amount of time that the WNI consumes tail energy cannot be longer than the leftover idle time after the reception of the whole burst and before the beginning of the next burst. We express this bound in Eq. (9). $\bar{P}_{tail}$ is the average power draw while consuming tail energy and when substituting it into Eq. (7) and setting $\frac{dP}{dT} \geq 0$ we obtain the inequality in Eq. (10).

$$E_{tail} \leq \bar{P}_{tail}\left(T - \frac{B}{r_{btc}} - \frac{(T r_s - B)}{r_s}\right) = \bar{P}_{tail}\left(\frac{B}{r_s} - \frac{B}{r_{btc}}\right) \tag{9}$$

$$\frac{r_{btc}}{r_s} \geq \frac{P_{rx}(r_{btc}) - \bar{P}_{tail}}{P_{rx}(r_s) - \bar{P}_{tail}} \tag{10}$$

We now use a model written out in Eq. (11) for the power consumption while receiving content, which linearly depends on the data rate. This is in general a safe assumption for WNIs [Xiao et al. 2010]. Because obviously the tail power never exceeds the power drawn while receiving at any rate, the constant a must be greater than or equal to one. When we substitute Eq. (11) into (10), after some straightforward manipulation, we obtain Eq. (12).

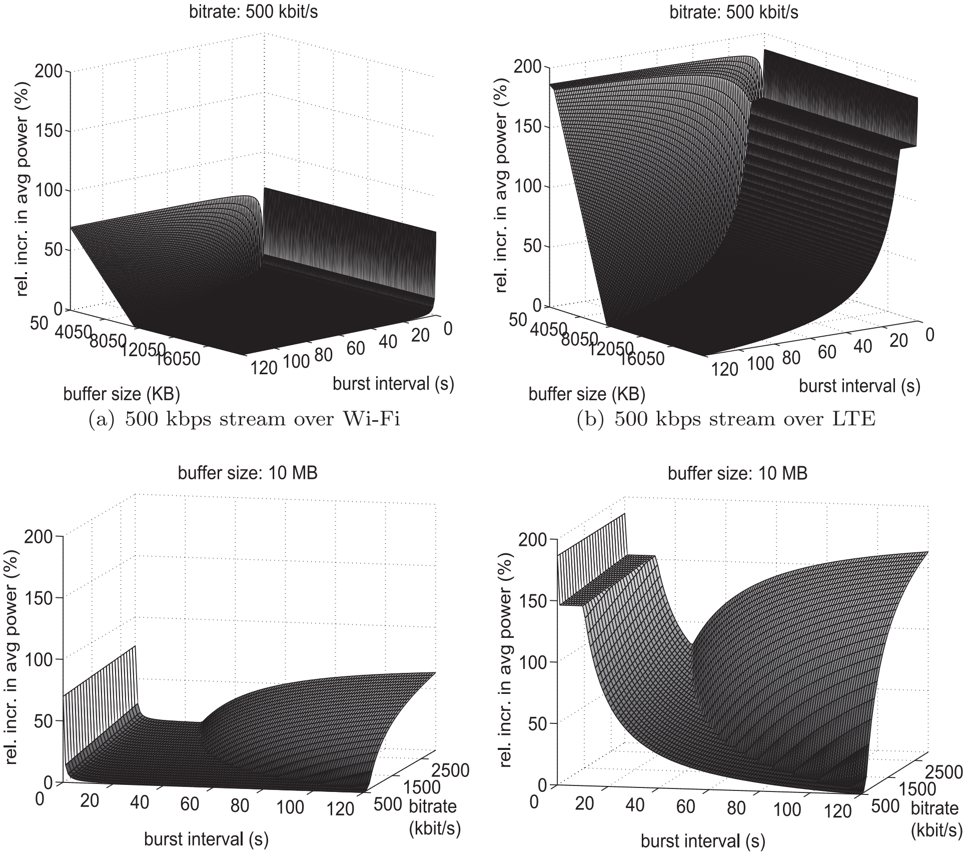
$$P_{rx}(r) = (a + kr)P_{tail}, \text{ subject to: } a \geq 1 \tag{11}$$

$$(a-1)(\frac{r_{btc}}{r_s} - 1) \geq 0 \tag{12}$$

The inequality in Eq. (12) is always true. Note that we did not even assume anything about the slope of the linear function (k). Hence, we have established that when the burst size exceeds the receiver buffer space, the power consumption is a nondecreasing function of T. Therefore, the lowest energy consumption can be achieved when the burst size exactly matches the available buffer space at the receiver.

3.4 Burst Size vs. Buffer Space Analysis

To quantify the impact of using optimal or nonoptimal burst size on energy consumption, we study the energy consumption in a few cases using Eqs. (2) and (7). To capture the effect of different amounts of



(a) 500 kbps stream over Wi-Fi

(b) 500 kbps stream over LTE

(c) 10MB buffer space using Wi-Fi

(d) 10MB buffer space using LTE

Fig. 5. Average power consumption with different burst intervals and amount of buffer space.

tail energy, we compare LTE without DRX and Wi-Fi. HSPA results are rather similar to LTE results so we omit them for brevity. Parameters were set according to measured values from experiments with HTC Velocity LTE: $\Delta P_{rx}(r_{btc})$ was set to 760 mW and 1520 mW, and r_{btc} to 20Mbit/s and 16Mbit/s, respectively for Wi-Fi and LTE. The tail energy (E_{tail}) was computed using typical values for the inactivity timers (Wi-Fi's PSM timer 0.2 s and LTE's inactivity timer 10 s) and P_{tail} was set to 435 mW for Wi-Fi and to 1216 mW for LTE. In the case of Wi-Fi, we set $\Delta P_{rx}(r_s)$ to a value between the idle power (P_{tail}) and $\Delta P_{rx}(r_{btc})$ assuming a linear relationship with the data rate. For LTE, we set $\Delta P_{rx}(r_s)$ to the same value as $\Delta P_{rx}(r_{btc})$.

Figures 5(a) and 5(b) plot the average power consumption as a function of both burst interval (T) and available buffer space (B). We make two main observations. First, the average power consumption decreases much more sharply with Wi-Fi than with LTE when the burst interval (and size) is increased. The reason is the difference in the length of inactivity timers: it takes a much longer burst cycle with LTE to amortize the tail energy associated with each burst than with Wi-Fi. Note that the 10 s inactivity timer is visible as a plateau in the LTE surface plot when the burst cycle is shorter than that. Second, and more importantly, while setting the burst cycle to a few tens of seconds may yield good energy savings immediately, setting the burst size just a bit too large leads to a significant increase in power consumption. This penalty is more striking with LTE where the power draw does not scale with the data rate and downloading content at the encoding rate is very energy inefficient. Hence, it is not a good idea to just blindly choose a "large enough" burst size. Instead, a smarter mechanism is required. We propose such a mechanism in the next section.

In Figures 5(c) and 5(d), the client buffer size was fixed to 10MB while bit rate and burst size vary. We observe that the lower the stream encoding rate, the larger the energy-saving potential. This result stems from the fact that the lower the stream encoding rate, the more spare bandwidth is available to leverage by the bursts. In other words, while two streams with different encoding rates consume the same amount of energy when received at constant bit rate (WNI continuously powered on), when they are shaped into fixed-size bursts, the one with a lower encoding rate leads to smaller-size bursts which take less time and, consequently, energy to receive than the larger bursts of the stream with higher encoding rate. This property is also much more pronounced with LTE compared to Wi-Fi.

4. EStreamer

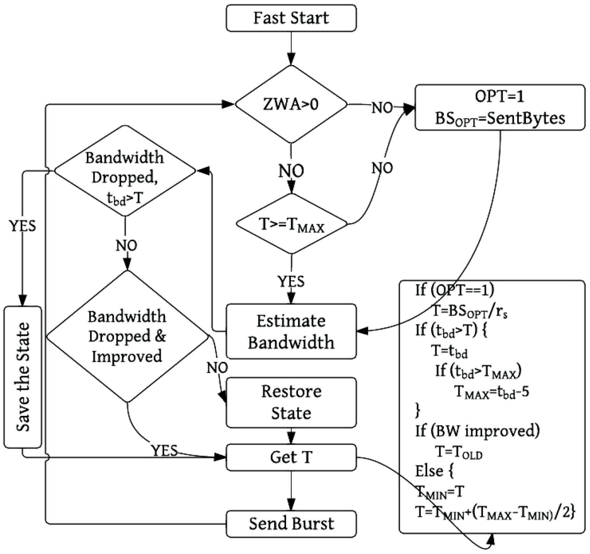
In this section, we propose a multimedia delivery system called EStreamer that can be integrated with an audio/video streaming service. It works with both constant-quality and rate-adaptive streaming. EStreamer turns an input multimedia stream into bursts and delivers these to the client. During a streaming session, it determines the optimal burst size for the client and uses that when constructing the bursts. It is a cross-layer system and consists of two components: (1) Traffic Profiler which works at the transport layer and (2) Traffic Shaper which works at the application layer.

Traffic Profiler (Profiler). The job of the Traffic Profiler is to capture TCP acknowledgement packets coming from the streaming clients and log the arrival time of the ACKs. From the ACK packets, it collects the sequence numbers of the ACKs and advertised receive window size. The log of the ACKs arrival times are used to calculate the duration of a burst (t_{bd}) and the receive window size is used to estimate the total buffer, B, status at the streaming client. Window size zero, namely ZWA, indicates that the client's total playback buffer B is full. The sequence number is used to determine whether a streaming client received all the packets of a burst.

Traffic Shaper (Shaper). The first task of the Shaper is to find the encoding rate of the content, r_s, by parsing the stream header. The next task is to keep track of the amount of data sent to the client during the Fast Start to calculate the maximum burst cycle, T_{max}. It is the maximum duration that the player can play without distorted playback unless more content is received. If the encoding rate is r_s bytes per second and L bytes are transferred to the client during t_{fs}, then the client has $T_{max} = \frac{L}{r_s}$ (assuming that r_s is constant) duration of playback content. Finally, it decides a burst cycle, T, according to the player buffer status received from the Profiler. In the meantime, the Shaper continues buffering the incoming traffic and sends a burst of size $T \times r_s$ to the client when T expires. The Shaper can also determine the end-to-end bandwidth for a burst, when it receives the corresponding burst duration from the Profiler.

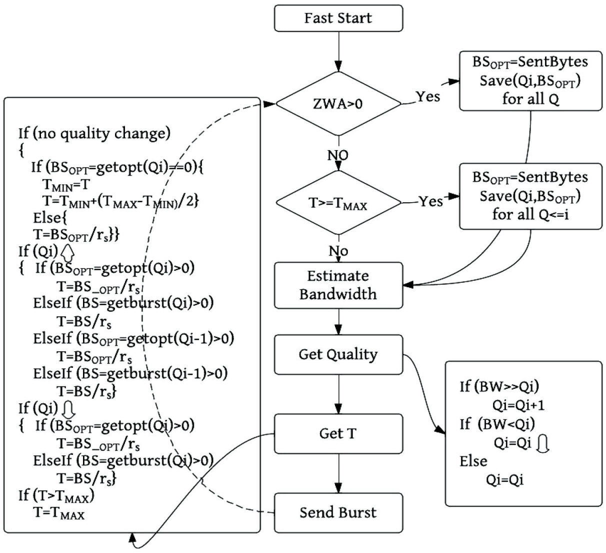
4.1 Finding the Optimal Burst Interval T_{opt}

Since encoding rate, r_s, is constant for a given stream quality, the burst size only depends on T. The Shaper seeks an optimal burst interval, (T_{opt}), with the help of the Profiler. In Section 2, we showed that the client TCP sends ZWAs to the sender to halt data transmission when the client's TCP receive buffer becomes full. The Profiler uses ZWAs to find the optimal burst size, BS_{OPT}, and reports to the Shaper. Then, the Shaper finds $Topt$, which is calculated as $T_{opt} = \frac{BS_{OPT}}{r_s}$. In this section, we discuss the techniques that EStreamer uses to find the optimal burst size at different phases of a streaming session.



```
If (OPT==1)
    T=$BS_{OPT}/r_s$
If ($t_{bd}$>T) {
    T=$t_{bd}$
    If ($t_{bd}>T_{MAX}$)
        $T_{MAX}=t_{bd}$-5
}
If (BW improved)
    T=$T_{OLD}$
Else {
$T_{MIN}$=T
T=$T_{MIN}+(T_{MAX}-T_{MIN})/2$}
```

Fig. 6. EStreamer's operation to find the optimal burst size for CBR streaming.



```
If (no quality change)
{
  If ($BS_{OPT}$=getopt(Qi)==0){
    $T_{MIN}$=T
    T=$T_{MIN}+(T_{MAX}-T_{MIN})/2$}
  Else{
  T=$BS_{OPT}/r_s$}}
If (Qi) ⇧
{  If ($BS_{OPT}$=getopt(Qi)>0)
        T=BS_$_{OPT}/r_s$
   ElseIf (BS=getburst(Qi)>0)
        T=BS/$r_s$
   ElseIf ($BS_{OPT}$=getopt(Qi-1)>0)
        T=$BS_{OPT}/r_s$
   ElseIf (BS=getburst(Qi-1)>0)
        T=BS/$r_s$}
If (Qi) ⇩
{  If ($BS_{OPT}$=getopt(Qi)>0)
        T=BS_$_{OPT}/r_s$
   ElseIf (BS=getburst(Qi)>0)
        T=BS/$r_s$}
If (T>$T_{MAX}$)
  T=$T_{MAX}$
```

```
If (BW>>Qi)
    Qi=Qi+1
If  (BW<Qi)
    Qi=Qi ⇩
Else
    Qi=Qi
```

Fig. 7. EStreamer's operation to find the burst interval for optimal power consumption for rate-adaptive streaming.

Finding the optimal burst size during Fast Start. EStreamer begins a streaming session with Fast Start and it is possible that the Profiler would find that the client's total buffer is filled even before the Fast Start is over. This event is more likely with very high bit rate streams. In this case, $BS_{OPT} = SentBytes$. Here, *SentBytes* is the amount of data delivered to the client untill the first zero window advertisement received by the Profiler.

Finding the optimal burst size after Fast Start. A straightforward strategy for finding such a burst size after Fast Start would be to start traffic shaping with a small value of T and then to increase it gradually. The shortcoming of this approach is that it may take a very long time to find the optimal one. For example, if $T_{opt} = 15$ s then it would take 120 s to reach, whereas the on-demand videos are of a few minutes in length (see Section 2.2). Therefore, EStreamer uses binary search to speed up. The mechanism is presented in Figures 6 and 7 as flow charts for two different streaming systems.

The basic mechanism works as follows. The Shaper calculates T_{max} during t_{fs} as described earlier. Then, it begins traffic shaping. Initially, it selects $T = T_{max}/2$ seconds and starts buffering incoming traffic from the streaming server over this period. The Profiler checks for the ZWAs in the traffic profile and reports to the Shaper. If there are no ZWAs, then T is increased. Finally, there will be one of the following two outcomes; (I) EStreamer reaches T_{max} without experiencing any ZWA and finds BS_{OPT}, (II) EStreamer observes ZWAs and $BS_{OPT} = SentBytes$ for a T which is less than T_{max}.

4.2 Dealing with Bandwidth Fluctuation

The playback at the streaming clients can be interrupted when using CBR streaming applications even without EStreamer if the bandwidth drops below the encoding rate. This situation can happen when traffic shaping is in one of the following three states: (i) BS_{OPT} has not been found yet, (ii) BS_{OPT} has been found and limited by the client's buffer space, or (iii) BS_{OPT} has been found and limited by T_{max} (see Figure 6). EStreamer preserves one of these occurred states by setting $T_{old} = T$. After that, it continuously sends content to the client while measuring the bandwidth in order to detect when the situation improves. It also continuously updates T_{max} based on the sent content and the encoding rate as long as the low-bandwidth situation continues. It restores the old state by setting $T = T_{old}$, when the bandwidth situation improves. After that it begins traffic shaping again to find the optimal burst size.

Rate-adaptive streaming is the appropriate solution to deal with bandwidth fluctuations. Such mechanisms are DASH [Stockhammer 2011], HLS, and MSS. Although these mechanisms employ their own rate-adaptation algorithms at the clients, we implement the algorithm at EStreamer. One of the main reasons is that the earlier version of EStreamer already had bandwidth detection and traffic shaping mechanisms [Hoque et al. 2013a]. We only added the quality switching functionality based on the detected bandwidth.

Figure 7 shows how EStreamer applies traffic shaping and rate adaptation together. The stream has been encoded into n different qualities which each have a different encoding rate $r_s^i, i \in 1, 2, 3 \ldots n$. At the beginning, EStreamer serves a streaming client with the maximum quality, the encoding rate of which is less than or equal to the half of 2Mbps (i.e., $r_s^i \leq (1/2 \times 2\text{Mbps})$). This initial choice is based on the finding by Cisco [2013] that smartphones enjoy on average 2Mbps mobile network speed. In addition, TCP provides satisfactory performance when the end-to-end bandwidth is twice the encoding rate, considering the effect of packet loss, congestion, round-trip time, and playback rate at the client [Wang et al. 2008]. If the client does not send any ZWA during the Fast Start phase, the content sent in that phase marks the largest possible burst size that can be used. Hence, EStreamer sets this burst size as the BS_{OPT} for the current quality and computes $T_{max} = T_{opt} = \frac{BS_{OPT}}{r_s}$. It also sets this burst interval (T_{opt}) to be the optimal for any other lower-quality stream with inferior encoding rate. The reason is that when switching to a lower-quality stream, the receive buffer cannot become a bottleneck because the size of the content for the same length (in seconds) of burst would be smaller. However, the same burst interval may not be optimal for a higher-quality stream because it is possible that the buffer space at the client becomes a limiting factor when the encoding rate increases since the amount of bytes also increases. Therefore, when switching to a higher-quality stream, EStreamer uses the previously found BS_{OPT} to determine an equivalent T and continues traffic shaping using binary search again. If ZWAs are observed, then EStreamer sets the optimal burst size for all qualities. The simple quality-switching algorithm is shown in Figure 7. A switch happens only if the determined average bandwidth permits. EStreamer allows upgrading the quality only when the current bandwidth is at least twice the encoding rate of the higher quality. The reason for this restriction is discussed earlier.

Different policies for quality switching can be plugged into the adaptive part of EStreamer. We want to emphasize that our simple quality-switching policy may be far from optimal in some cases and that our goal is to demonstrate how adaptive streaming can be supported in EStreamer.

4.3 Implementation

For using with popular constant bit rate streaming services, such as YouTube, we chose to deploy EStreamer in an HTTP proxy server and placed the proxy in the cloud. The reason is that we had no possibility of deploying EStreamer on the corresponding servers. The smartphones' proxy settings were configured to use EStreamer. EStreamer does not shape traffic during the Fast Start phase and forwards 20–45 s playback data depending on the service.

In order to experiment with rate-adaptive streaming, we integrated EStreamer with a streaming server and developed a separate client. The communication between the client and EStreamer works based on HTTP over TCP. We describe the HTTP request/response headers in Appendix A. At the beginning, the client makes the initial GET request. EStreamer selects the initial quality and responds by sending the stream initialization header of the corresponding quality. It also mentions the duration of the video, bit rate, time range, and other parameters in the response header (X-Stream-Info: duration =597;bitrate=700000;seconds=0-59;height=480; width=853). Next, the client makes the request for actual content in time range (Range: seconds=0-) to which EStreamer responds by sending the next

Table II. The Streaming Services, the Encoding Rate of the Streams, Frame Rates, Container, and their Total Playback Durations

Streaming Service	Bit rate (kbps)	Frame Rate	Container	Duration
ShoutCast(audio) (CBR)	128	–	MP3	4–10
YouTube Browser (CBR)	280,328,2000	25	FLV,MP4	4:47,9:57
YouTube App (CBR)	458,2000	25,24	MP4	4:47,9:57
Dailymotion App (CBR)	452	25	MP4	5:58
Service X	700,1200,1500,2000,2500,3000	96	MP4	9:57

chunk. Note that the client specifies only the start of the range. It is the job of EStreamer to determine an appropriate burst size and whether the stream quality should be switched or not. After that, whenever the client's playback buffer goes down to only 5 seconds, it sends a new request specifying the next missing content as the beginning of the range (e.g., Range: seconds=60-). If EStreamer decides to switch stream quality, it specifies the new bit rate in the response header (X-Stream-Info: duration=597;bitrate=2000000; height=720; width=1280; seconds=100-139;). However, response mismatch can occur when there are ZWAs from the client, as EStreamer might send less content than the time range specified in the header. This is because EStreamer aborts sending the remaining content of the determined burst size. In this case, the client again requests the remaining content and EStreamer responds with the response code 204 and correction in the response header as shown in Appendix A.4.

4.4 Discussion

We briefly discuss here the impact of different characteristics of various multimedia streams on the behavior of EStreamer. It can support different encoding rates, codecs, and containers as long as the streaming content is carried by HTTP over TCP. In Hulu traces, we have seen that TCP flow control can also take place if the client cannot accommodate a chunk when using HTTP rate-adaptive streaming. Therefore, EStreamer logic can be integrated with DASH-like clients as well. However, such a solution would have information about the playback buffer and TCP receive buffer. Consequently, the heuristic can be bypassed.

EStreamer counts in bytes when searching for the optimal burst size and then computes the burst interval. Therefore, the mechanism is robust also when Variable Bit Rate (VBR) streaming is used. Since the traffic shaper uses the average bit rate to compute the corresponding burst interval during which the target-sized burst is generated, there may be cases where the burst size is not exactly of the right size if the encoding rate deviates significantly from the average during that interval. Hence, the resulting burst may be slightly bigger or smaller than the optimal size in some cases. In some streaming systems, the server can send audio and video as two separate streams. In such a case, EStreamer must synchronize the shaping of the two streams so that the burst transmissions coincide in order to maximize the idle time in between the burst transmissions. The mechanism to detect the optimal burst size remains the same and will be done separately for the two streams.

5. PERFORMANCE EVALUATION

We used three popular constant bit rate multimedia streaming services, Internet Radio, YouTube, and Dailymotion, in four different smartphones via EStreamer. Table II shows the streaming services and the properties of the multimedia contents used in the smartphones in our experiments. A 128 kbps Internet radio channel was played in all the smartphones and two different YouTube videos were viewed. One video was encoded at 280 and 320 kbps rates with FLV container. We also streamed a 458 kbps version of the same video of the MP4 container. The rate of the other video was 2000 kbps

Table III. Test Cases with Different Network Parameter Configurations

RAN	Test cases	Network configuration
HSPA	default(def)	T1 = 8s, T2 = 3s, T3 = 29mins, CELL_PCH on
	aggressive(aggr)	T1 = 6s, T2 = 2s, T3 = 29mins, CELL_PCH on
	no PCH(noPCH)	T1 = 8s, T2 = 10s, CELL_PCH off
LTE	default without DRX	RRC_{idle} = 10s, DRX off
	default with DRX	RRC_{idle} = 10s, DRX_{idle} = 750ms, DRX_{c} = 640ms, DRX_{on} = 20 ms
	default with DRX, long idle	RRC_{idle} = 20s, DRX_{idle} = 750ms, DRX_{c} = 640ms, DRX_{on} = 20ms

and the duration was ten minutes. A 452 kbps video was streamed from Dailymotion using the native application. For rate-adaptive streaming (Service -X in Table II), we used the DASH dataset of "big buck bunny" from Lederer et al. [2012] .

5.1 Methodology

We did two sets of measurement. In the first set, we used Wi-Fi, a commercial HSPA 3G network and the LTE 4G test network of Nokia Solutions and Networks. The smartphones were connected to the Internet via a DLink DIR-300 wireless AP supporting 802.11 b/g. The bandwidth of the Wi-Fi network was 54Mbps. In the case of HSPA and LTE, the configurations used were the "def" configuration and default configuration with DRX, respectively, in Table III. The results obtained from this measurement setup are presented in Sections 5.2 and 5.3. We also describe the results obtained during bandwidth fluctuation scenarios via HSPA and LTE in Section 5.4.

The second set of measurements was done in the HSPA and LTE test networks of NSN with different timer settings. Table III summarizes the network configurations for the different test cases. For HSPA measurements, we used three different network configurations. Default configuration refers to the configuration according to the vendor-recommended parameters. Aggressive configuration refers to shorter values of T1 and T2. In the third configuration, the T2 value was set to a longer value when the CELL_PCH state was disabled. LTE experiments were done with cDRX disabled or enabled and with short or long inactivity timer, RRC_{idle}, values. The purpose of these tests is twofold: (i) study the effect of timer settings on the energy consumption of smartphones (Section 5.5) and (ii) study the effect of our energy-aware traffic shaping on radio network signaling (Section 5.6).

We used Monsoon power monitor and Nokia Energy Profiler to measure the power consumption of the smartphones. During the audio streaming sessions the displays were turned off. For video streaming sessions, the displays were on. Therefore, the video measurement results include power consumption for decoding, display, and wireless networking. We present the energy savings result in the presence of EStreamer by comparing with the power consumption of mobile devices when EStreamer was absent.

We present our measurement results and analysis in three steps. First, we evaluate the performance of EStreamer in determining the optimal burst cycle and providing energy savings. We also discuss how EStreamer deals with bandwidth variations in the network. Second, we illustrate the effect of various cellular network configurations on energy savings. Finally, we consider the impact of traffic shaping on radio network signaling.

5.2 Burst Size Tuning and Energy Consumption for Constant-Quality Streaming

In this section, we demonstrate how EStreamer finds the optimal burst size when streaming constant-quality content. We observed three different cases: In the first one, EStreamer detects ZWAs during the

Table IV. Maximum Power Savings at T_{opt} for Different Mobile Phones Using EStreamer, while Playing CBR Audio and Video Streaming Applications over Wi-Fi, HSPA and LTE

Applications Access Network	HTC Nexus One (Android-2.3.6) Sav%–kbps–T_{opt}	Nokia N900 (Maemo) Sav%–kbps–T_{opt}	Nokia E-71 (Symbian V 3.0) Sav%–kbps–T_{opt}	HTC Velocity (Android 2.3.7 Sav%–kbps–T_{opt}
Internet Radio/Wi-Fi	23%–128–14 s(⋆)	62%–128–14 s(⋆)	65%–128–6 s(⋄)	–
Internet Radio/HSPA/LTE	38%–128–14 s(⋆)	24%–128–14 s(⋆)	2%–128–4 s(⋄)	60%–128–18 s(⋆)
YouTube Bro/Wi-Fi	14%–328–39 s (⋆)	20%–328–39 s (⋆)	18%–280–4 s(⋄)	
YouTube Bro/HSPA/LTE	14%–328–39 s (⋆)	14%–328–39 s (⋆)	4%–280–3 s(⋄)	50%–2000–31 s(○)
YouTube App/ Wi-Fi	13%–458–39 s(⋆)	–	–	–
YouTube App/HSPA/LTE	27%–458–39 s(⋆)	–	–	54%–2000–39 s(⋆)
Dailymotion/Wi-Fi	15%–452–33 s(⋄)	–	–	–
Dailymotion/HSPA/LTE	30%–452–33 s(⋄)	–	–	55%–452–33 s(⋄)

YouTube videos were played using both browser ("Bro") and the native application. The LTE measurement results are only from the HTC Velocity phone.

Table V. Improvement in Battery Lifetime in Hours Using EStreamer

					Battery life (h)		
Device	Savings%	Battery(mAh)	(mA) no EStr.	(mA) EStr.	no EStr.	EStr.	bg traffic
E-71 (Wi-Fi)	65%–128–6 s(⋄)	1500	270	85	5.1	14.7	-
Velocity (LTE)	50%–2000–31 s(○)	1620	622	311	2.6	5.2	-
Nexus (HSPA)	38%–128–14 s(⋆)	1400	405	251	3.5	5.5	-
Nexus (HSPA)	14%–328–39 s(⋆)	1400	480	407	2.92	3.42	3.27
N900 (Wi-Fi)	62%–128–14 s(⋆)	1320	270	103	4.9	12.8	-
N900 (HSPA)	20%–328–39 s(⋆)	1320	540	432	2.4	3.0	2.4

Fast Start phase of streaming a 2Mbps video to the browser-based player in the HTC Velocity phone. As a result, the traffic shaper determines $T_{opt} = 31$ s and always sends a burst after every 31 s until the end of the video (see Table IV). In the second case, EStreamer determines T_{opt} using the binary search ending up with a value that is less than T_{max}, which happens when streaming the Dailymotion video. These scenarios are indicated with a diamond symbol in the table. Third, EStreamer finds T_{opt} at T_{max} when streaming the 2Mbps video to the native YouTube player in HTC Velocity and these results are indicated with a star symbol.

One observation from the results is that streaming the same bit rate audio via HSPA to N900 saves 24% energy, whereas the Nexus One saves 38% energy. The reason is that N900 can spend only (14-t_{bd}-T1-T2)≈2 seconds in CELL_PCH state, whereas the Nexus One uses legacy Fast Dormancy with an inactivity timer of 6.5 s and spends (14-t_{bd}-6.5)≈7 seconds in the idle state. We discuss the effect of FD and different timer settings in cellular networks in Section 5.5. It is also evident that the potential energy savings are larger with audio than with video streaming, which stems from the fact that the data transfer energy consumption makes a larger component in the total consumption in audio streaming compared to video streaming where display and decoding take a bigger share. Finally, the results show that the largest energy savings can be achieved when streaming via Wi-Fi (65%), which is followed by LTE (55%) and then HSPA (38%). These savings improve the battery lifetime of Nokia E-71, HTC Velocity, and Nexus One by 3x, 2x, and 1.5x, respectively (see Table V).

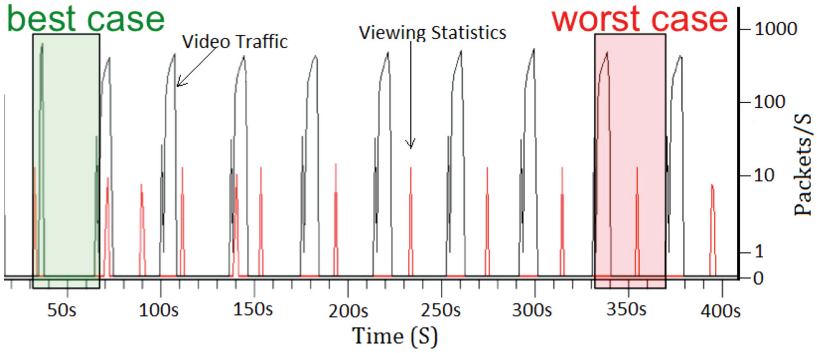


Fig. 8. Isolating best- and worst-case bursts for background traffic impact estimation. The y-axis is in logarithm scale.

5.3 Impact of Background Traffic

Surprisingly, Table IV indicates that streaming a low-quality video (328 kbps) from YouTube to browser ("Bro") in Nexus One consumes more energy than streaming a higher-quality video (458 kbps) to the native application via HSPA in the same device. Figure 8 illustrates what happens during YouTube streaming to the browser in Nexus One via EStreamer as an example. The browser-based player establishes several TCP connections to YouTube servers. EStreamer shapes only the traffic of the connection that carries the video content (black spikes). The other connections are used to report the user's viewing progress (red spikes).

Other applications installed in a smartphone can also generate such periodic background traffic. The prime examples are the email clients and free applications with embedded advertisement code of some advertisement platform, such as AdMob. We observed that the email client in an Android phone fetches email every five minutes. The interval for retrieving an advertisement is within the range of 12–120 s [Prochkova et al. 2012]. The packets of these applications can interleave with the video content bursts in an unfortunate manner and the impact is an increase in overall energy consumption. We measured an increase of 27% and 6% power consumption in N900 and Nexus One, respectively, in the worst case (in presence of a background connection which reports the user's viewing history). In Nexus One, we observed similar increase when the email client retrieves email from the server. Table V shows that background traffic can nullify the achievement from traffic shaping if the worst-case scenario appears during each burst interval.

5.4 Bandwidth Fluctuation: Constant-Quality and Rate-Adaptive Streaming

In this section, we describe how EStreamer deals with bandwidth variability. For streaming audio of constant quality, we used WonderShaper [2013] at the EStreamer hosting machine to limit the downloading bandwidth of the streaming client. Figure 9 shows a streaming session in N900 via HSPA. It shows that power consumption increases as the bandwidth decreases after the 12th round. It also illustrates that when bandwidth falls below the encoding rate, there is playback interruption. In this case, EStreamer saves this state by setting $T_{old} = 14$ s. After a few rounds, T_{max} increases to 43 s. From the 15th round bandwidth improves and EStreamer reintroduces the old state, $T = T_{old} = 14$ s, and continues to shape traffic again.

We also experimented with rate-adaptive streaming when streaming via HSPA by moving around the campus during the streaming sessions. We found that EStreamer fluctuated among the three lowest qualities. This is because of the limited downlink capacity of the subscription. Since it was not possible to measure power consumption while roaming, we present a similar scenario in Figure 10.

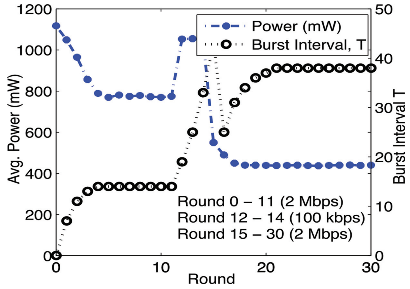


Fig. 9. Power consumption of N900 during bandwidth variation and streaming a 128 kbps audio stream via HSPA.

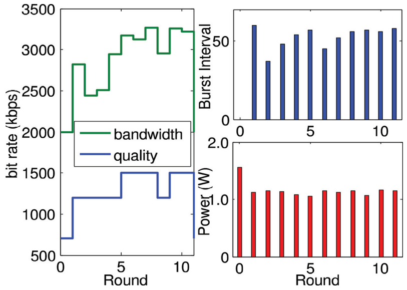


Fig. 10. Impact of bandwidth variation on quality, burst interval, and power of the HTC Velocity when streaming via HSPA.

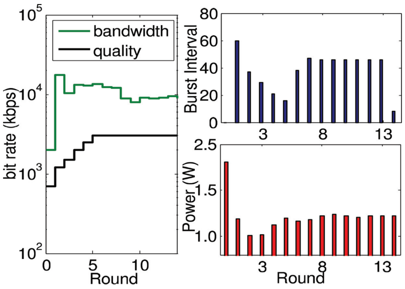


Fig. 11. Impact of bandwidth variation on quality, burst interval, and power of the HTC Velocity when streaming via LTE.

Table VI. Energy Savings of Smartphones when Streaming Video and Audio with Different HSPA Network Configurations

	YouTube		Audio	
Network Configuration	Nokia N900	Nexus One	Nokia N900	Nexus One
default	14%	16%	24%	38%
no PCH	11%	20%	18%	38%
aggressive	19%	12%	46%	43%

The stair graph in the figure shows that EStreamer switches to the maximum 1500 kbps stream at the 5th round, and transitions to the 1200 kbps at the 7th round. At the 9th round, EStreamer switches back to 1500 kbps. The stair graph in Figure 11 shows how EStreamer changes to a higher quality when there is enough bandwidth using LTE in log scale. At the 5th round it switches to the 3000 kbps stream. Therefore, from the upper bar chart we can see that initially the burst interval decreases as the quality of video increases. EStreamer increases the burst interval by applying binary search and finds the optimal burst interval for 3000 kbps video at $T = 46$ s, as the TCP flow control reacts. It sets the corresponding burst size as BS_{OPT} for all other qualities. Then, it continues traffic shaping with the same burst interval and quality until the end of the video. The lower bar chart shows that power consumption is less when streaming lower-quality videos. The key messages from these measurement results are the following. Power consumption increases when bandwidth decreases. In an extreme situation, the bandwidth may decline below the lowest available encoding rate and a user may experience distorted playback. However, EStreamer can react accordingly for both constant bit rate and rate-adaptive streaming for improving the battery life of a smartphone.

5.5 Impact of Cellular Network Configuration

Table VI presents results obtained with different network configurations. We first note that shorter timers do not generally help to save energy in the absence of EStreamer. If EStreamer is used, slightly more energy can be saved with more aggressive timer values than the default configuration. Energy savings for audio streaming sessions are higher than the video streaming scenarios. The reason is the quicker transition to lower-power state after receiving a burst. However, this observation does not hold for video streaming to Nexus One as the device uses legacy Fast Dormancy. From traffic and power traces we identified that it uses an inactivity timer value of approximately 6.5 s. This timeout is shorter than T1 values in default and no-CELL_PCH scenarios but it is just a bit longer than the T1 value in the aggressive setup. As a consequence, Nexus One activates the legacy FD in most of

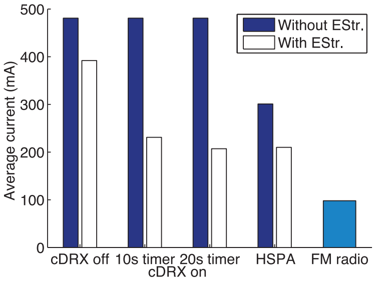


Fig. 12. Energy consumption while streaming audio via HSPA and LTE.

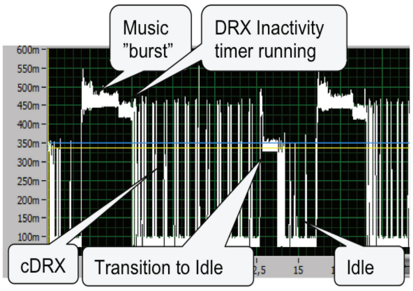


Fig. 13. DRX w/10 s inactivity timer. The x-axis is time (s) and y-axis current (mA).

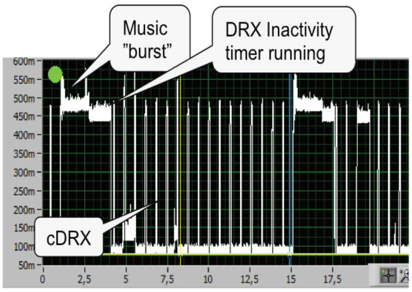


Fig. 14. DRX w/20s inactivity timer. x-axis is time (s) and y-axis current (mA).

the cases. Background traffic also contributes to the energy consumption increase, which we already explained in Section 5.3.

Energy savings with different LTE parameter configurations are displayed in Figure 12. The figure shows that larger savings can be obtained by increasing the inactivity timer value from 10 s to 20 s, which seems counterintuitive at first. When $\text{RRC}_{idle} = 10$ s, the LTE protocol transitions from RRC_CONNECTED to RRC_IDLE in between receiving the bursts because the timer is shorter than the burst interval. We can also see that without EStreamer, streaming over LTE requires more energy than over HSPA and streaming FM radio. In the case of HSPA, the energy savings is relatively lower than the LTE scenarios with EStreamer. The reason is the long inactivity timers. Figure 13 shows that this transition causes a non-negligible amount of energy to be spent due to the signaling. If RRC_{idle} is increased to 20 s, this transition no longer occurs and thus the average current is decreased (see Figure 14). Even further savings should be possible when increasing the burst interval and optimizing the cDRX profile.

5.6 Impact of Traffic Shaping on Radio Network Signaling

The energy savings achieved by reshaping traffic into bursts come with a price when using cellular network access. The savings are achieved by increasing the number of RRC state changes during the streaming session. Each state change generates some signaling traffic. We extracted the number of state changes for each test case from the signaling logs. Since we know the number of signaling messages required for a specific state change, we were able to compute the total number of signaling messages.

The signaling measurement results in the HSPA network for YouTube traffic shaping are shown on the left-hand side of Table VII. Although in each case EStreamer causes more signaling load compared to the case without EStreamer, there is a clear difference between Nexus One and Nokia N900. The reason lies in the legacy FD that Nexus One uses. Since that mechanism terminates the RRC connection, the IDLE→CELL_DCH transition in between each video content burst requires a lot more signaling than the CELL_PCH→CELL_DCH transition because of the RRC reconnection. For the same reason, the signaling load is higher with Nexus One when CELL_PCH is not available in the network. YouTube background traffic plays a negative role in signaling, as well. In the case of Nexus One, those periodic packets emerge right in the middle of the video bursts that cause extra IDLE→CELL_DCH→IDLE transitions each time. However, N900 shows a surprisingly higher increase in signaling for no-CELL_PCH and the aggressive timers cases than the default settings. The reason turned out to be that N900 applied legacy FD after some bursts. The logic explaining how N900 decides when to apply that mechanism remains unclear to us.

Table VII. Increase in Signaling in the HSPA Network due to YouTube Video and ShoutCast Audio Traffic Shaping by EStreamer

	YouTube		Audio	
Network Configuration	Nokia N900	Nexus One	Nokia N900	Nexus One
default	7 → 26(↑3.7x)	11 → 100(↑9.1x)	3 → 49(↑16x)	3 → 114(↑38x)
no PCH	6 → 47(↑7.8x)	11 → 96(↑8.7x)	4 → 49(↑11x)	3 → 102(↑34x)
aggressive	6 → 37(↑6.2x)	14 → 109(↑7.8x)	–	6 → 74(↑12x)

The results are expressed as "messages without EStreamer→ messages with EStreamer ↑ factor of increase in number of messages" per minute.

We looked at the increase in signaling traffic for the audio streaming case as well (right-hand side of Table VII). The main insight is that, overall, the signaling traffic grew more compared to the YouTube experiments as EStreamer selected shorter burst intervals in the audio streaming case because of the shorter T_{max}. Another notable difference to the video streaming results is that disabling the CELL_PCH state did not increase the signaling load as much as in the YouTube experiments. The reason is that T_{opt} is shorter than the inactivity timer values T1 + T2 (8 + 10) s, which means that no transitions to the idle state happened, except with Nexus One. In the case of LTE, there are only two states and there can be two kinds of state transitions. When $\text{RRC}_{idle} = 10$ s and cDRX is disabled, the average number of state transitions is 4.3/min. Signaling does not increase when cDRX is enabled and the inactivity timer is $\text{RRC}_{idle} = 10$ s or 20 s. Thus the cDRX mechanism does not impose extra signaling on the network.

The previous observations are important for network operators while configuring network parameters and for researchers or mobile vendors while designing an energy-aware mobile network access system or policy in general. The key implications are the following: (i) The signaling load in the network increases if a network does not support the CELL_PCH state. The reason is frequent RRC reconnection. (ii) The usage of legacy Fast Dormancy by the smartphones reduces energy consumption but increases the signaling load very rapidly as the mobile device frequently closes and re-establishes the RRC connection. (iii) The usage of very short timers reduces energy consumption but can increase the signaling load in the network because of the frequent state transitions but the effect is less pronounced compared to the two previous cases. In summary, if the network supports CELL_PCH, then the short timers or FD Rel-8.0 can be a win-win situation for both parties as the number of signaling messages can be tolerable for the network operators [NSN 2011] and energy savings are significant for the smartphones. In case of LTE, enabling cDRX does not increase in network signaling. We note that similar discontinuous reception/transfer mechanisms exist also for HSPA where they are applied in the CELL_DCH state. The concept is called Continuous Packet Connectivity (CPC) and it was introduced already in the 3GPP Release 7. However, the support in phones and networks does not seem widespread.

6. RELATED WORK

Traffic shaping is widely used to save communication energy for wireless multimedia streaming [Hoque et al. 2012]. Korhonen and Wang [2005] proposed an adaptive streaming technique for UDP-based multimedia streaming. The system works at the server or proxy and manipulates the burst interval depending on the packet loss and network situation experienced by the streaming client. On the other hand, our focus is HTTP over TCP-based streaming. We modeled the energy consumption of bursty TCP traffic. Then we developed EStreamer based on the models. EStreamer depends on standard TCP properties to apply traffic shaping, which greatly simplifies the implementation of energy-aware streaming. It does not require either the support of any secondary protocol such as RTCP or the modification of TCP or any other protocols. Therefore, EStreamer can be easily integrated with modern TCP-based streaming services.

Other approaches identify idle periods at different phases of TCP-based applications, such as in the middle of the data transmission [Liu and Zhong 2008; Tan et al. 2007]. Another example is choking/unchoking the TCP receive window size to make the TCP traffic bursty [Yan et al. 2006]. In this case, the burst interval is the duration between a choking and unchoking period. Authors in Tan et al. [2007] applied this trick for multimedia streaming services such as RealNetwork, Windows Media, and YouTube with a burst interval of 200 ms. However, these mechanisms cannot be applied for 3G and 4G, because these solutions are wireless access technology dependent as they force the Wi-Fi interface into sleep state and such an operation on cellular network interfaces would bar the smartphone from basic phone functioning. In contrast, EStreamer is independent of the WNI being used for streaming. Recently, Li et al. [2012] proposed GreenTube to save communication energy for YouTube using multiple TCP connections. However, such approaches cannot be successful in reducing energy consumption when the application receives content at the server-controlled lower rate [Hoque et al. 2013b].

Many papers have also studied the energy efficiency of 3G communication. Xiao et al. [2008] were among the first to study the energy consumption of YouTube streaming over both Wi-Fi and 3G. We go much beyond the scope of that work by considering the impact of traffic shaping and different network configurations, including LTE. Afterwards, Balasubramanian et al. [2009] performed a measurement study on the energy consumption of 3G communication but did not consider streaming applications in their study. Qian et al. [2011] characterized the energy efficiency of several different applications. They observed that some music streaming applications behave in an energy-inefficient manner due to the CBR traffic. Earlier in Qian et al. [2010a], the same authors proposed a traffic shaping scheme for YouTube and computed estimates on potential energy savings with that scheme. However, they did not consider the consequence of TCP flow control on the energy consumption of WNIs in their study. A more recent and thorough measurement study on different mobile video streaming services and the resulting energy consumption on different mobile OSs and devices is presented in Hoque et al. [2013b].

Concerning the RRC parameter configuration in cellular networks, Lee et al. [2004] first proposed to tune the inactivity timers dynamically. Qian et al. [2010a] suggested traffic-aware inactivity timer configuration to reduce energy consumption. They also proposed to trigger Fast Dormancy based on the information provided by different applications [Qian et al. 2010b]. Falaki et al. [2010] proposed the total of $T1 + T2 = 4.5$ s based on their observation on packet inter-arrival time in traffic traces. Ukhanova et al. [2012] also suggested aggressive timer configuration for CBR video transmission from mobile devices. Deng and Balakrishnan [2012] proposed to initiate FD dynamically instead of with a fixed timeout value. In fact, with these proposed timer settings a mobile device would be able to save more energy in the presence of EStreamer. At the same time, these studies do not consider the consequence of the proposed configurations on the network and these are the cases where our study also can have significant implications. We considered the benefits and disadvantages of different network configurations for bursty streaming traffic, and thus the importance of our study also lies in careful configuration of the RRC parameters in the network and designing energy-aware network access.

Compared to our earlier work [Hoque et al. 2013a], our notable new contributions are the following. First we demonstrate the relationship between available buffer space at the client, the received burst size, and the energy consumption. After that, we illustrate the implementation of EStreamer for both CBR and rate-adaptive streaming, and then demonstrate how EStreamer quickly fine-tunes the energy-optimal configuration for a given client using a binary search approach. In Siekkinen et al. [2013], we analyzed only video streaming results, whereas in this work we include power and signaling measurement results both for audio and video streaming scenarios.

7. CONCLUSIONS

In this article, we first modeled the energy consumption of TCP-based bursty streaming traffic. Then, based on the insights gathered through the modeling, we designed and implemented an energy-efficient multimedia delivery system called EStreamer. We evaluated its performance and energy savings of smartphones. We showed that EStreamer strives for as large energy savings as possible for each client without compromising the quality of the streaming service. This energy efficiency is irrespective of the WNI being used for streaming. To carry out these tasks, it uses a heuristic derived from the models and it does so by shaping traffic for each client in an energy-optimal way. Finally, the energy consumption overall differs quite a lot between the devices, which is mostly explained by the different hardware components used in the phones. An important lesson from this study is that while a similar magnitude of savings can be achieved in different ways, from the network's perspective there is a big difference because of varying amounts of signaling load.

In Section 5.3, we have seen that the share of background traffic in the total energy consumption of smartphones is significant. Those applications retrieve information after some periodic intervals. Different applications maintain different frequency of periodicity. As a result, the network access becomes frequent and sporadic. We have witnessed how such background traffic can diminish the effect of multimedia traffic shaping. Deng and Balakrishnan [2012] proposed to accumulate background traffic from different applications and then send together. Their solution did not consider any networking application, specifically VoIP or multimedia, running in the foreground. In addition, modern mobile platforms, such as Android, do not provide API to hint to any application that some other application is using networking resources. However, such hints could be useful in scheduling background traffic with respect to the traffic of VoIP or other streaming applications in order to minimize the negative effect that we experienced.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

We thank Te-Yuan Huang from Stanford University for sharing Hulu traces with us.

REFERENCES

Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference (IMC'09).* ACM Press, New York, 280–293.

Xu Cheng, Jiangchuan Liu, and Cameron Dale. 2013. Understanding the characteristics of internet short video sharing: A youtube-based measurement study. *IEEE Trans. Multimedia* 15, 5, 1184–1194.

CISCO. 2013. Cisco visual networking index: Global mobile data traffic forecast update, 2012–2017. (Feb. 2013). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

Shuo Deng and Hari Balakrishnan. 2012. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT'12).* ACM Press, New York, 181–192.

Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. 2010. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC'10).* ACM Press, New York, 281–287.

Alessandro Finamore, Marco Mellia, Maurizio M. Munafo, Ruben Torres, and Sanjay G. Rao. 2011. YouTube everywhere: Impact of device and infrastructure synergies on user experience. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC'11).* ACM Press, New York, 345–360.

Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, Oliver Spatscheck, and Xiaodong Zhang. 2006. Delving into internet streaming media delivery: A quality and resource utilization perspective. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC'06).* ACM Press, New York, 217–230.

Mohammad A. Hoque, Matti Siekkinen, and Jukka K. Nurminen. 2011. On the energy efficiency of proxy-based traffic shaping for mobile audio streaming. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC'11).* 891–895.

Mohammad A. Hoque, Matti Siekkinen, and Jukka K. Nurminen. 2012. Energy efficient multimedia streaming to mobile devices – A survey. *IEEE Comm. Surv. Tutorials PP,* 99, 1–19.

Mohammad A. Hoque, Matti Siekkinen, Jukka K. Nurminen, and Mika Aalto. 2013a. Dissecting mobile video services : An energy consumption perspective. In *Proceedings of the 14th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'13).*

Mohammad A. Hoque, Matti Siekkinen, and Jukka K. Nurminen. 2013b. TCP receive buffer aware wireless multimedia streaming – An energy efficient approach. In *Proceedings of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'13).* ACM Press, New York, 13–18.

Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the ACM Conference on Internet Measurement Conference (IMC'12).* ACM Press, New York, 225–238.

K. W. Hwang, David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, V. Misra, Kadangode K. Ramakrishnan, and Deborah F. Swayne. 2012. Leveraging video viewing patterns for optimal content placement. In *Proceedings of the 11th International IFIP TC 6 Conference on Networking (IFIP'12),* vol. part 2. Springer, 44–58.

Jari Korhonen and Ye Wang. 2005. Power-efficient streaming for mobile terminals. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'05).* ACM Press, New York, 39–44.

Stefan Lederer, Christopher Muller, and Christian Timmerer. 2012. Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys'12).* ACM Press, New York, 89–94.

Chi-Chen Lee, Jui-Hung Yeh, and Jyh-Cheng Chen. 2004. Impact of inactivity timer on energy consumption in wcdma and cdma2000. In *Proceedings of the IEEE Wireless Telecommunications Symposium.* 15–24.

Xin Li, Mian Dong, Zhan Ma, and Felix Fernandes. 2012. GreenTube: Power optimization for mobile video streaming via dynamic cache management. In *Proceedings of the 20th ACM International Conference on Multimedia (MM'12).* ACM Press, New York.

Jiayang Liu and Lin Zhong. 2008. Micro power management of active 802.11 interfaces. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys'08).* ACM Press, New York, 146–159.

LLC Signals Research Group. 2010. Smartphones and a 3g network: Reducing the impact of smartphone-generated signaling traffic while increasing the battery life of the phone through the use of network optimization techniques. Tech. rep. http://www.nokiasiemensnetworks.com/system/files/document/Signals_Ahead_Executive_Summary_100614.pdf.

NSN. 2011. Understanding smartphone behavior in the network. White paper, Nokia Siemens Networks.

Irena Prochkova, Varun Singh, and Jukka K. Nurminen. 2012. Energy cost of advertisements in mobile games on the android platform. In *Proceedings of the 6th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'12).* IEEE Computer Society, 147–152.

Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. 2011. Profiling resource usage for mobile applications: A cross-layer approach. In *Proceedings of the 9th International Conference on Mobilesystems, Applications, and Services (MobiSys'11).* ACM Press, New York, 321–334.

Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2010a. Characterizing radio resource allocation for 3g networks. In *Proceedings of the 10th Annual Conference on Internet Measurement (IMC'10).* ACM Press, New York, 137–150.

Feng Qian, Zhaoguang Wang, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2010b. TOP: Tail optimization protocol for cellular radio resource allocation. In *Proceedings of the 18th IEEE International Conference on Network Protocols (ICNP'10).* IEEE Computer Society, 285–294.

Matti Siekkinen, Mohammad Hoque, Jukka K. Nurminen, and Mika Aalto. 2013. Streaming over 3g and lte: How to save smartphone energy in radio access network-friendly way. In *Proceedings of the 5th ACM Workshop on Mobile Video (MoVid'13).* 13–18.

Thomas Stockhammer. 2011. Dynamic adaptive streaming over http – Standards and design principles. In *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems (MMSys'11).* ACM Press, New York, 133–144.

Enhua Tan, Lei Guo, Songqing Chen, and Xiaodong Zhang. 2007. PSM-throttling: Minimizing energy consumption for bulkdata communications in wlans. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP'07).* 123–132.

Anna Ukhanova, Evgeny Belyaev, Le Wang, and Søren Forchhammer. 2012. Power consumption analysis of constant bit rate video transmission over 3g networks. *Comput. Comm.* 35, 14, 1695–1706.

Bing Wang, Jim Kurose, Prashant Shenoy, and Don Towsley. 2008. Multimedia streaming via tcp: An analytic performance study. *ACM Trans. Multimedia Comput. Comm. Appl.* 4, 2.

WonderShaper. 2013. WonderShaper. http://lartc.org/wondershaper.

Yu Xiao, Ramya S. Kalyanaraman, and Antti Yla-Jaaski. 2008. Energy consumption of mobile youtube: Quantitative measurement and analysis. In *Proceedings of the 2nd International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST'08).* 61–69.

Yu Xiao, Petri Savolainen, Arto Karppanen, Matti Siekkinen, and Antti Yla-Jaaski. 2010. Practical power modeling of data transmission over 802.11g for wireless applications. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (e-Energy'10).* ACM Press, New York, 75–84.

Haijin Yan, Scott A. Watterson, David K. Lowenthal, Kang Li, Rupa Krishnan, and Larry L. Peterson. 2006. Client-centered, energy-efficient wireless communication on ieee 802.11b networks. *IEEE Trans. Mobile Comput.* 5, 11, 1575–1590.

YouTube. 2013. YouTube statistics. http://www.youtube.com/yt/press/statistics.html/.

Received May 2013; revised October 2013; accepted November 2013