TCP Receive Buffer Aware Wireless Multimedia Streaming–An Energy Efficient Approach

Mohammad Ashraful Hoque, Matti Siekkinen, Jukka K. Nurminen
Aalto University School of Science, Finland
firstname.lastname@aalto.fi

ABSTRACT

Shaping constant bit rate traffic into bursts has been proposed earlier for UDP-based multimedia streaming to save Wi-Fi communication energy of mobile devices. The relationship between the burst size and energy consumption of wireless interfaces is such that the larger is the burst size, the lower is the energy consumption per bit received as long as there is no packet loss. However, the relationship between the burst size and energy in case of TCP traffic has not yet been fully uncovered. In this paper, we develop a power consumption model which describes this relationship in wireless multimedia streaming scenarios. Then, we implement a cross-layer stream delivery system, EStreamer. This system relies on a heuristic derived from the model and on client playback buffer status to determine a burst size and provides as small energy consumption as possible without jeopardizing smooth playback. The heuristic greatly simplifies the deployment of EStreamer compared to most existing solutions by ensuring energy savings regardless of the wireless interface being used. We show that in the best cases using EStreamer reduces energy consumption of a mobile device by 65%, 50-60% and 35% while streaming over Wi-Fi, LTE and 3G respectively. Compared with existing energy-aware applications energy consumption can be reduced by 10-55% further.

1. INTRODUCTION

Energy consumption is an important issue with modern smartphones. Among the most popular applications are video and audio streaming which can drain a fully charged battery in only a few hours. The fact that shaping constant bit rate UDP multimedia traffic into bursts can save energy is well known, specifically in the case of Wi-Fi [1, 7].

In this paper, we study the interplay between the burst size and power consumption for TCP-based audio and video streaming services. We model the energy consumption of bursty TCP traffic. We show that in practice the smallest power consumption can be obtained when the burst size exactly equals the sum of application playback buffer and TCP receive buffer regardless of the wireless interface used. Consequently, we propose an energy-aware multimedia stream delivery system, EStreamer. It uses a heuristic to find an energy optimal burst size and idle period length for a streaming client so that smooth playback of the mobile applications is not distorted. The heuristic is such that if a burst exceeds TCP receive buffer then power consumption increases. The heuristic relies on standard TCP feedback from the client. Specifically, EStreamer checks from the received TCP acknowledgements whether the client TCP sent zero window advertisements and tunes the burst size based on this feedback.

We evaluate the energy savings for popular streaming services, such as Internet radio, YouTube, Dailymotion, where constant bit rate content is delivered. The results demonstrate that the potential of EStreamer to save smartphone's energy is large, but also that it varies significantly between devices, wireless radio interfaces and multimedia encoding rates. The largest energy savings can be achieved with Wi-Fi access about 65%, followed by LTE of 50-60%, after which comes 3G with which a maximum of roughly 35%. The long inactivity timers currently imposed in the 3G networks are the main reasons for not achieving larger savings.

2. POWER MODELING OF BURSTY STREAMING TRAFFIC OVER TCP

The common feature of different radio interfaces, e.g. Wi-Fi, 3G and LTE, is that they all exhibit a certain amount of tail energy which is spent keeping the radio idle after a transmission (of a burst or just a single packet) ends and before it transitions into a low power state. Reducing this tail energy is the main target for energy savings when shaping traffic into bursts. We denote tail energy as E_{tail}.

The amount of tail energy is controlled through inactivity timers with all the different access technologies. These timer values are fixed for a specific TCP connection or even for a given device and operator combination. Wi-Fi PSM-A exhibits small tail energy by using typically 200 ms timer value [12]. 3G has two timers, T1 and T2, which control the transition from CELL_DCH to CELL_FACH and CELL_PCH states. These timers are usually set to few seconds. LTE comes with connected mode DRX (cDRX) which may or may not be supported in today's commercial networks. If cDRX is not supported, LTE has a large tail energy due to an inactivity timer (typically around 10s) controlling the transition from connected to idle mode. If cDRX is supported, LTE behaves much like Wi-Fi and has

Description	Parameter	Description	Parameter
Stream rate	r_s	Fast Start Streaming Period	t_{fs}
TCP bulk transfer capacity to client	r_{btc}	Transmission rate during t_{fs}	r_{fs}
TCP receive buffer size at client	R_F	Data Transferred during t_{fs}	$L = r_{fs} \cdot t_{fs}$
Idle time in between two consecutive bursts	t_{idle}	burst interval	T
Power draw of application playing the stream	P_a	Optimal burst interval	T_{opt}
Power in sleeping state, i.e. baseline power	P_s	Maximum burst interval	T_{max}
Power when receiving data at rate r	$P_{rx}(r)$	Power increase when receiving data at rate r	$\Delta P_{rx}(r) = P_{rx}(r) - P_s$
Tail energy	E_{tail}	Minimum burst interval during traffic shaping	T_{min}
Inactivity timer values	T_1, T_2	Burst Size	$BS = T \cdot r_s$
Power draw in tail energy states	P_1, P_2	Application Buffer Size	A_F
Avg power while spending tail energy(radio on but no rx/tx)	P_{tail}	Total Buffer Size	$B = R_F + A_F$

Table 1: Description of the parameters used in the modeling and describing EStreamer. For Wi-Fi, P_{tail} is the power draw in an idle state, i.e. radio on but no rx/tx. For 3G it is a 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.

relatively small tail energy which is due to the DRX inactivity timer (typically a few hundred milliseconds) that triggers DRX in connected mode.

2.1 Assumptions and parameters

The parameters used in the following equations are described in Table 1. We consider the power consumption to be fixed when actively receiving data at a given rate (see e.g. [13]) and we assume that streaming rate (r_s) is always lower than TCP bulk transfer capacity (r_{btc}), i.e. there is some spare bandwidth to exploit.

We assume that multimedia players maintain a fixed size of playback buffer. The size can depend on the implementation of the player and also can be restricted by the operating system of resource constraint mobile devices. During a streaming session the player reads data from the TCP receive buffer through socket API and stores the data into the buffer. The player decodes content from the playback buffer at the encoding rate. When some buffer is freed (at least stream encoding rate amount), the player again reads from the socket. Therefore, TCP receive buffer acts as a secondary buffer at the streaming client for TCP-based multimedia streaming. Total available playback buffer can be considered as the sum of application buffer and TCP receive buffer.

Since the client has a limited buffer size, we consider two corresponding cases in the modeling: 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.

2.2 Receiver buffer is sufficiently large

First, we consider the case when the fixed size burst can be entirely absorbed by a streaming client, i.e. $r_s T \leq B$. We obtain the total power draw for a given T and its derivative with respect to T as follows:

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

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

To compute the derivative of E_{tail}, 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}) is smaller than the sum of the inactivity timers, which can happen especially with 3G. Since we have maximum of two timers (the case of 3G), we treat three different cases (we skip the straightforward manipulation steps):

$$0 < t_{idle} < T_1 : \; E_{tail} = P_1 t_{idle} = P_1 T(1 - \frac{r_s}{r_{btc}}) \quad (3)$$

$$\frac{dE_{tail}}{dT} = P_1(1 - \frac{r_s}{r_{btc}}) \quad (4)$$

$$T_1 < t_{idle} < T_2 : \; E_{tail} = P_1 T_1 - P_2 T_1 + P_2 T(1 - \frac{r_s}{r_{btc}}) \quad (5)$$

$$\frac{dE_{tail}}{dT} = P_2(1 - \frac{r_s}{r_{btc}}) \quad (6)$$

$$T_2 < t_{idle} : \; E_{tail} = P_1 T_1 + P_2 T_2 \quad (7)$$

$$\frac{dE_{tail}}{dT} = 0 \quad (8)$$

When we substitute (3)-(8) into (2), we obtain the following:

$$0 < t_{idle} < T_1 : \; \frac{dP}{dT} = 0 \quad (9)$$

$$T_1 < t_{idle} < T_2 : \; \frac{dP}{dT} = \frac{T_1}{T_2}(P_2 - P_1) < 0, \text{ since } P_2 < P_1 \quad (10)$$

$$T_2 < t_{idle} : \; \frac{dP}{dT} = -\frac{E_{tail}}{T^2} < 0 \quad (11)$$

The above result shows that the power draw either stays the same or decreases when T is increased until the TCP buffer can absorb the whole burst. Thus, the larger the value of T, the less energy is consumed by the client device.

2.3 Burst Size Exceeds TCP Receive Buffer

Now, we consider the other case where the burst is larger than the TCP receive buffer, i.e. $r_s \cdot T > B$. In this case, the portion of the burst which 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. The reader can easily check this for each of the three different cases of tail energy as we did in (3)-(8). Thus, we can treat E_{tail} as constant wrt. T. We obtain the following formula for the power consumption and its derivative:

$$P(T) = P_a + P_s + \frac{B\Delta P_{rx}(r_{btc})}{Tr_{btc}} + \frac{(Tr_s - B)}{Tr_s}\Delta P_{rx}(r_s) + \frac{E_{tail}}{T} \tag{12}$$

$$\frac{dP}{dT} = \frac{B}{T^2}(\frac{\Delta P_{rx}(r_s)}{r_s} - \frac{\Delta P_{rx}(r_{btc})}{r_{btc}}) - \frac{E_{tail}}{T^2} \tag{13}$$

The tail energy in (13) is bound by the idle time left after receiving the whole burst as follows:

$$E_{tail} \leq P_{tail}(T - \frac{B}{r_{btc}} - \frac{(Tr_s - B)}{r_s}) = P_{tail}(\frac{B}{r_s} - \frac{B}{r_{btc}}) \tag{14}$$

When we substitute (14) into (13) and set $\frac{dP}{dT} \geq 0$ and $P_{rx}(r_{btc}) = (1 + x_1)P_{tail}, P_{rx}(r_s) = (1 + x_2)P_{tail}$, where P_{tail} is the average power draw while consuming tail energy, we obtain the following inequality:

$$\frac{x_1}{x_2} \leq \frac{r_{btc}}{r_s} \tag{15}$$

If the above inequality holds, power consumption either stays the same or increases when T increases. This balance is dependent on the ratios of the power consumption while receiving data at different rates and during the tail. Note that the inequality is independent of T, which means that for a given r_s and r_{btc}, the minimum power draw is obtained when $T \cdot r_s = B$ (the inequality holds) or otherwise when T is arbitrarily large (the inequality does not hold).

A typical characteristic of 3G radio is that the power draw is quite insensitive to the data rate. For this reason, we can say that $P_{rx}(r_{btc}) \approx P_{rx}(r_s)$. Consequently, (15) holds because $r_s < r_{btc}$.

Unlike with 3G, the power draw while receiving data over Wi-Fi can depend on the rate significantly if the rates are relatively low, as shown in [13]. The relationship is such that a higher transmission rate draws more power but consumes still less Joules per bit transmitted, i.e. the energy utility is better with higher rates. Thus, we cannot directly say that Eq. (15) always holds. However, the typical tail energy with Wi-Fi is much smaller than the upper bound in Eq. (14) we used to derive Eq. (15). To show this, we solved Eq. (13) for B to find the minimum TCP receiver buffer size that guarantees that the inequality holds, i.e. power consumption strictly increases when T grows beyond the receiver buffer set limit, and computed the result with parameter values measured from a real mobile device. We discovered that a few kilobytes of the receive buffer suffices for the inequality to hold. In practice, this is always the case with current mobile devices. Since the LTE's DRX inactivity timer value is in similar orders of magnitude as the Wi-Fi's timer, we consider the same reasoning to hold for that technology.

The above analysis shows that in practice the power consumption should always increase when T is increased beyond the threshold defined by the receiver buffer size. In next section we apply this heuristic in a multimedia delivery system called EStreamer.

3. EStreamer

As a whole, EStreamer is a cross layer mechanism. At the network layer, it checks the TCP acknowledgements received from the streaming clients to identify client TCP receive buffer status. Based on client receive buffer status it selects the burst size at the application layer. Hence, EStreamer consists of two components; (1) Traffic Profiler and (2) Traffic Shaper.

3.1 Traffic Profiler

Like other streaming services, EStreamer begins with Fast Start and this phase lasts for t_{fs} time. During Fast Start, traffic is sent using maximum bandwidth to the streaming client. However, the job of the Traffic Profiler is to look into ACK packets during Fast Start phase and corresponding to a sent burst. Then it reports the playback buffer status of the client player to the Traffic Profiler based on the TCP receive window size in the packet. Window size zero indicates that client buffer is full.

3.2 Traffic Shaper

The first task of the Traffic Shaper is to determine the maximum burst interval, T_{max}, that a client can wait without its buffer running dry. 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 $T_{max} = \frac{L}{r_s}$ seconds of media stream in its buffer (assuming that r_s is constant). The second task is to send burst according to the player buffer status received from the Traffic Profiler.

3.3 Finding Optimal Burst Interval

Since r_s is constant, burst size only depends on T. The Shaper seeks an optimal burst interval, (T_{opt}), for each client with the help of the Profiler reported information concerning full client buffer and maximum burst interval T_{max}.

The Traffic Shaper uses a binary search approach in resizing the burst interval in order to find the optimal value. It selects an initial burst interval of $T_{max}/2$. Then, it gets the client buffer status report from the Profiler and knows whether the previous burst size was too large or not. If not, the burst interval is increased and if yes, it is decreased, both in binary search manner. The end result will be one of the two: i) the Shaper finds the limit set by client's buffer size, i.e. $T_{opt} < T_{max}$, ii) the Shaper reaches T_{max} without filling client's buffer, i.e. $T_{opt} = T_{max}$.

4. EVALUATION

Since we wanted to experiment with popular streaming services and had no possibility of deploying EStreamer on the corresponding servers, we chose to implement and deploy it in an HTTP proxy server. Mobile devices were configured to use that HTTP proxy.

We measured the energy consumption of four mobile devices with three different wireless access networks; Wi-Fi (802.11b/g), 3G (WCDMA) and LTE. For Wi-Fi network we used a DLink DIR-300 wireless AP supporting 802.11 b/g. In 3G network, the inactivity timer values are 8 s and 3.5 s respectively for $T1$ and $T2$. The subscription rate was 2.0 Mbps. In the case of LTE, we used Nokia Siemens LTE Test Network. cDRX was enabled in the network. The RRC inactivity timer and DRX inactivity timer was configured as 10 s and 680 ms respectively. The downlink capacity of the Test network was 16 Mbps. We used Monsoon power monitor and Nokia Energy Profiler to measure the energy. We show our measurement results as combinations of the power consumption and the corresponding burst interval whenever

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/3G/LTE	38%–128–14 s	24%–128–14 s	2%–128–4 s	60%–128–18 s
YouTube Bro/Wi-Fi	14%–912–36 s	20%–328–39 s	18%–280–4 s	
YouTube Bro/3G/LTE	16%–328–38 s	14%–328–39 s	4%–280–3 s	50%-2000-31 s
YouTube App/ Wi-Fi	13%–458–38 s	–	–	–
YouTube App/3G/LTE	27%–458–38 s	–	–	54%–2000–39 s
Dailymotion/Wi-Fi	15%–452–33 s	–	–	–
Dailymotion/3G/LTE	30%–452–33 s	–	–	55%–452–33 s

Table 2: Maximum power savings at T_{opt} for different mobile devices using EStreamer, while playing different audio and video streaming applications over Wi-Fi, 3G and LTE. The LTE measurement results are only with HTC Velocity phone.

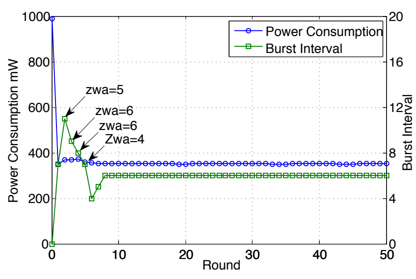

Figure 1: Streaming 128 kbps Internet Radio to Nokia E-71 over Wi-Fi.

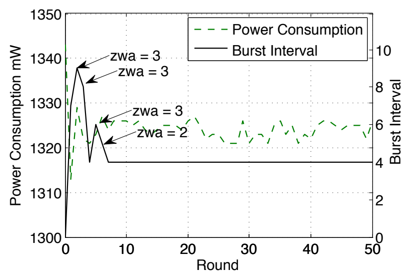

Figure 2: Streaming 128 kbps Internet Radio to Nokia E-71 over 3G.

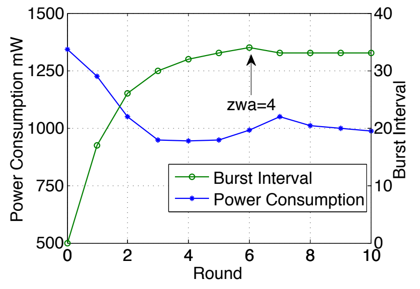

Figure 3: Streaming 452 kbps Dailymotion video to Nexus One over 3G.

EStreamer sends a burst. We name this burst sending event as `round`.

We used three popular audio/video services with constant bit rate streaming in the experiments: Internet Radio, YouTube, and Dailymotion. The encoding rates of the streams were between 128 kbps and 2 Mbps.

We present results for the two different cases: *(i) Burst size exceeds TCP receive buffer size* and *(ii) Burst size never exceeds total buffer*. Besides these, we also compare the performance of EStreamer with the existing energy aware streaming applications. A summary of the achieved energy savings is shown in Table 2.

4.1 Burst Size Exceeds TCP Receive Buffer

In order to find the optimal burst interval, EStreamer begins traffic shaping with a value of $T = T_{max}/2$ seconds and makes further changes until it finds a maximum burst interval for which there would be no ZWAs.

Figure 1 and 2 show the power consumption of Nokia E-71 and also the tuning of burst interval while streaming a 128kbps Radio over Wi-Fi and 3G respectively. In the former case EStreamer starts traffic shaping with $T = 7$ s. Then it finds ZWAs during the next burst with $T = 11$ s. In order to reduce the energy consumption because of the extra data, EStreamer reduces the next burst interval and then at the 7th round it finds $T_{opt} = 6$ s. In the case of 3G, EStreamer also begins with the same value and finds $T_{opt} = 4$ s. Here two noticeable things are: i) The optimal burst interval is almost equal to the value of the 3G inactivity timer $T1$ and therefore power consumption does not reduce significantly. ii) E-71 uses a smaller TCP receive buffer for 3G as compared with the Wi-Fi scenario and this clarifies the rate control behavior that we observed in our previous work [5]. Power savings for these two experiments are presented in Table 2.

EStreamer also shows similar traffic shaping pattern for streaming 452 kbps Dailymotion video to the Nexus One phone via 3G (see Figure 3). For the T of 34 seconds, EStreamer finds ZWAs from the smartphone. Subsequently burst interval is reduced for the immediate next round and then increased again. One scenario for a very high bit rate YouTube video streaming with the Nexus One is illustrated in Figure 4. We notice that power consumption increases at the 5th round as the EStreamer sends burst higher than the client TCP receive buffer and then decreases again as the EStreamer moves toward the optimal buffering period of 36 s. In this case, there are only few rounds as the streaming lasts for around 300 seconds. However, power consumption is reduced by 14%.

4.2 Total Playback Buffer is Sufficiently Large

When T_{max} is smaller than the client's buffer size, EStreamer will reach that burst interval. The reason is that the client can accommodate the whole burst into application buffer and TCP receive buffer completely. Therefore, EStreamer will never detect ZWAs for any T and will find the optimal burst interval at T_{max}, as a result, achieve more energy savings.

For some YouTube video streaming sessions, EStreamer finds T_{opt} also at T_{max}, regardless of the wireless interface used. An example of such a case is shown in Figure 5. While streaming a 458 kbps video to the Nexus One, EStreamer begins traffic shaping with $T = 19$ s and ends up with $T_{opt} = T_{max} = 38$ s. In Figure 6 we illustrate another scenarios for streaming high definition video (720p) to HTC Velocity via

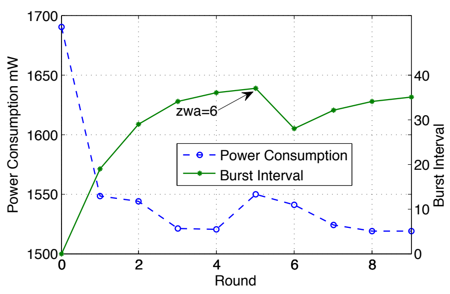

Figure 4: 912 kbps YouTube video to Nexus One over Wi-Fi using browser.

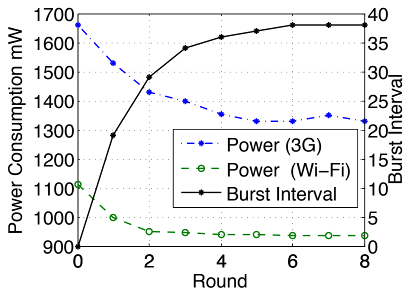

Figure 5: Streaming 458 kbps YouTube video to Nexus One over Wi-Fi and 3G.

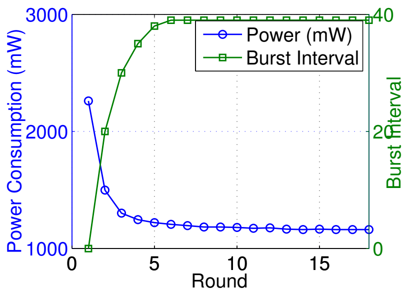

Figure 6: Streaming 2Mbps video using YouTube app via LTE in HTC Velocity.

LTE. In this case, EStreamer also finds T_{opt} at $T_{max} = 39$ s and power savings can be improved by 54%.

4.3 Energy Efficient Mobile Applications

Access Network	App	Rate (kbps)	Pow(mW) (case1)	Pow(mW) (case2)
Wi-Fi	Dailymotion	452	720	609
3G	Dailymotion	452	1353	947
LTE	Dailymotion	452	1891	859
Wi-Fi	YouTube App	458	720	630
3G	YouTube App	458	1365	991
LTE	YouTube App	2000	1520	1110

Table 3: Power consumption of Nexus One for playing Dailymotion and YouTube videos without (case 1) and with (case 2) EStreamer.The presented LTE measurements are only with HTC Velocity.

In Table 3, we compare the power consumption of native video applications without and with EStreamer. The native Dailymotion application in Velocity consumes 1891 mW while streaming a video via LTE. Nexus One consumes 1353 mW for streaming the same video via 3G, but consumes only 720 mW via Wi-Fi even without EStreamer. YouTube application in Nexus One behaves in the same way. Hoque et al. identified that these applications exploit TCP flow control to generate bursty traffic [6]. However, they cannot escape the TCP flow control packets in between two bursts and hence power consumption is high with cellular networks. To mitigate this problem, Li et al. [9] proposed GreenTube which uses multiple TCP connections to receive traffic. The YouTube player in HTC Velocity also uses multiple connections as shown in Figure 7. From Table 3, we can see that YouTube App consumes 20% less power than the Dailymotion while streaming a higher rate video via LTE.

On the other hand, in presence of EStreamer 3G and Wi-Fi power consumption can be reduced by 25% and 10% respectively over the energy efficient applications. LTE power consumption can be reduced by 55% for the Dailymotion video (see Table 2). In case of YouTube HD video, power consumption can be reduced further by 26% over LTE. In other words, EStreamer outperforms existing energy saving approaches significantly. The reason is that server sends content at a maximum twice of the encoding rate. Hence, downloading a 600 s video would require 300 s. From figure 7 we can see that total downloading time is also around 300 s using multiple TCP connections. On the contrary, players receive content from EStreamer at maximum bandwidth and took total 140 s to download the whole content.

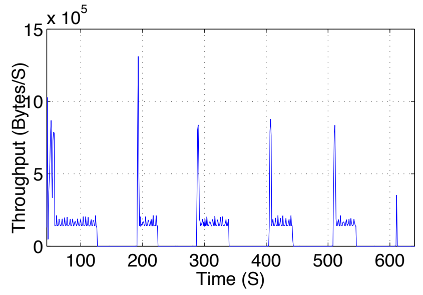

Figure 7: Throughput while downloading of a 720*p* video by the native YouTube application using multiple TCP connections in HTC Velocity.

5. RELATED WORK

Traffic shaping is a popular technique to reduce Wi-Fi communication energy for multimedia streaming to mobile devices. This had been done by Gundlach et al. [4] as well as by Chandra and Vahdat [3] for the UDP based video streaming. Shenoy and Radkov [11] applied two kinds of traffic shaping. Anastasi et al. [1] introduced a proxy-based power saving streaming protocol for UDP based streaming, named the Real Time Power Saving Protocol (RT_PS). Perhaps the most relevant work for our study was done by Korhonen and Wang [7]. They proposed an adaptive streaming technique for the UDP based multimedia streaming. The system works at the server or proxy and manipulates burst interval depending on the packet loss and network situation experienced by the streaming client.

Other approaches identify idle periods at different phases of TCP-based applications, such as during the user think time [2], TCP slow start [8] or in the middle of the data transmission [10, 12]. They discard standard PSM and drive down the Wi-Fi interface to a *sleep* state during these idle periods. Another example is choking/unchoking TCP receive window size to make the TCP traffic bursty [14]. In this case the burst interval is the duration between a choking and unchoking period. Authors in [12], applied this trick for multimedia streaming services such as RealNetwork, Windows Media and YouTube with a burst interval of 200 ms. Nevertheless, these mechanisms cannot be applied for 3G and LTE, because these solutions are wireless access technology dependent. Recently, Li et al. proposed GreenTube to save communication energy for YouTube using multiple TCP connections [9]. However, such approaches might not be successful when the application receives content at the server controlled rate (see Section 4.3).

In summary, EStreamer differs from previously published work in the following ways. i) We uncover the relationship between burst size and total buffer available at the client in TCP-based multimedia traffic shaping. ii) We developed a power consumption model for bursty TCP multimedia traffic, and derive a heuristic from the model. iii) Based on the heuristic we implement EStreamer which adaptively finds an energy optimal burst interval regardless of the wireless interface.

6. CONCLUSION

In this work, we designed an energy efficient streaming multimedia delivery system, EStreamer. We evaluated it's performance and energy savings of smartphones by setting it in a proxy. We showed that EStreamer strives for as large energy savings as possible for each client without compromising the quality of the streaming service. It is more energy efficient than the existing mechanisms. This energy efficiency is irrespective of the wireless interface being used for streaming. To carry out these tasks, it uses a heuristic derived from a model and it does so by shaping traffic for each client in an energy optimal way. At present, we are studying the effect on energy consumption when EStreamer handles few hundred clients. We are also investigating the impact on cellular network signaling messages while using EStreamer.

7. REFERENCES

[1] Anastasi, G., Passarella, A., Conti, M., Gregori, E., and Pelusi, L. A power-aware multimedia streaming protocol for mobile users. In *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on* (july 2005), pp. 371 – 380.

[2] Brakmo, L. S., Wallach, D. A., and Viredaz, M. A. Îijsleep: a technique for reducing energy consumption in handheld devices. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services* (New York, NY, USA, 2004), MobiSys '04, ACM, pp. 12–22.

[3] Chandra, S., and Vahdat, A. Application-specific network management for energy-aware streaming of popular multimedia formats. In *Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2002), USENIX Association, pp. 329–342.

[4] Gundlach, M., Doster, S., Yan, H., Lowenthal, D. K., Watterson, S. A., and Chandra, S. Dynamic, power-aware scheduling for mobile clients using a transparent proxy. In *Proceedings of the 2004 International Conference on Parallel Processing* (Washington, DC, USA, 2004), ICPP '04, IEEE Computer Society, pp. 557–565.

[5] Hoque, M., Siekkinen, M., and Nurminen, J. On the energy efficiency of proxy-based traffic shaping for mobile audio streaming. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE* (jan. 2011), pp. 891–895.

[6] Hoque, M., Siekkinen, M., and Nurminen, J. Energy efficient multimedia streaming to mobile devices – a survey. *Communications Surveys Tutorials, IEEE PP*, 99 (2012), 1 –19.

[7] Korhonen, J., and Wang, Y. Power-efficient streaming for mobile terminals. In *Proceedings of the international workshop on Network and operating systems support for digital audio and video* (New York, NY, USA, 2005), NOSSDAV '05, ACM, pp. 39–44.

[8] Krashinsky, R., and Balakrishnan, H. Minimizing energy for wireless web access with bounded slowdown. In *Proceedings of the 8th annual international conference on Mobile computing and networking* (New York, NY, USA, 2002), MobiCom '02, ACM, pp. 119–130.

[9] Li, X., Dong, M., Ma, Z., and Fernandes, F. GreenTube: Power optimization for mobile video streaming via dynamic cache management. In *Proceedings of the ACM Multimedia* (New York, NY, USA, 2012), acmmm'12, ACM.

[10] Liu, J., and Zhong, L. Micro power management of active 802.11 interfaces. In *Proceedings of the 6th international conference on Mobile systems, applications, and services* (New York, NY, USA, 2008), MobiSys '08, ACM, pp. 146–159.

[11] Shenoy, P., and Radkov, P. Proxy-assisted power-friendly streaming to mobile devices. In *In MMCN* (2003), pp. 177–191.

[12] Tan, E., Guo, L., Chen, S., and Zhang, X. Psm-throttling: Minimizing energy consumption for bulk data communications in wlans. In *Proceedings of the IEEE International Conference on Network Protocols, ICNP 2007, October 16-19, 2007, Beijing, China* (2007), IEEE, pp. 123–132.

[13] Xiao, Y., Savolainen, P., Karppanen, A., Siekkinen, M., and Ylä-Jääski, A. 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* (New York, NY, USA, 2010), e-Energy '10, ACM, pp. 75–84.

[14] Yan, H., Watterson, S. A., Lowenthal, D. K., Li, K., Krishnan, R., and Peterson, L. L. Client-centered, energy-efficient wireless communication on ieee 802.11b networks. *IEEE Transactions on Mobile Computing 5* (November 2006), 1575–1590.