On the Energy Efficiency of Proxy-Based Traffic Shaping for Mobile Audio Streaming

Mohammad A. Hoque, Matti Siekkinen
Dept. of Computer Science and Engineering
Aalto University School of Science and Technology, Finland
mohammad.hoque@tkk.fi,matti.siekkinen@tkk.fi

Jukka K. Nurminen
Nokia Research Center, Finland
jukka.k.nurminen@nokia.com

***Abstract*—We study how much energy can be saved by reshaping audio streaming traffic before receiving at the mobile devices. The rationale is the following: Mobile network interfaces (WLAN and 3G) are in *active* mode when they transmit or receive data, otherwise they are in *idle/sleep* mode. To save energy, minimum possible time should be spent in *active* mode and maximum in *idle/sleep* mode. It is well known that by reshaping the usually constant bit rate multimedia traffic into bursts, it is possible to spend more time in *idle/sleep* mode leading to impressive energy savings. We propose a proxy-based solution that shapes an audio stream into bursts before relaying the traffic to the mobile device. The novelty of our work is an evaluation of the energy savings using such a proxy with different configurations for both WLAN access with standard 802.11 Power Saving Mode and 3G access. We conclude that for WLAN access, proxy causes power savings of 30%-65% depending on the audio stream rate, location of the proxy and amount of cross traffic. In the case of 3G, the effectiveness of our proxy seems to vary depending on the phone model and operator. In some cases, the energy savings are encouraging, while in other cases the proxy turns out to be ineffective due to abnormal delay variation and TCP flow control behavior.**

KEYWORDS: Power Consumption, Traffic Shaping, Proxy, TCP, 3G, Fast Streaming.

I. Introduction

Mobile phones are nowadays routinely capable to access the Internet through wireless network interface (WNI). We have also witnessed the coming of broadband wireless Internet access through 3G networks with flat rate pricing models in many countries. This evolution has boosted the popularity of streaming applications usage with mobile devices. However, the limited battery capacity of mobile devices presents an increasingly important challenge since data transmission consumes a lot of battery power regardless of the interface used.

In this paper we focus on the power consumption of audio streaming on a mobile phone. Multimedia streaming applications require continuous delivery of media traffic and they typically force a mobile device to keep its WNI or 3G interface most of the time in high power consuming *active* mode. According to Chandra et al. [1], 802.11 power saving mode (PSM) as such is not suitable for reducing the energy consumption with multimedia streaming applications, as most of the media traffic tends to be delivered as a constant bit rate traffic which does not allow the network interface to efficiently enter the *sleep* state in between packets. In case of 3G, there are so called inactivity timers which control the transition of the interface from high-power active states to lower-power idle states. These timers have usually rather long values which prohibit the interface from entering lower power states while receiving a multimedia stream.

We present in this paper a simple proxy-based solution for reducing the energy consumption of mobile devices while using an audio streaming application which we call Internet radio from hereon. Our solution handles audio streams transmitted over TCP, instead of RTP over UDP, for instance, since that is by far the most common case [2]. The proxy is placed between the mobile phone and the radio server and it repeatedly buffers the constant bit rate traffic received from a radio station for a fixed period of time, which we term *buffering period*, and forwards the buffered data to the mobile client in form of a burst. This makes PSM more effective by enabling the client WNI to spend more time in the *sleep* state and to be in the *active* state only for the duration required to receive the bursts from the proxy. In other words, it reduces frequent WNI transitions from one state to another and almost every transition from *sleep* to *active* state is effectively utilized to receive the burst. As for the 3G (WCDMA) access, the idea is to buffer the constant bit rate traffic such a long time that the idle time between bursts is enough to let the interface transition into lower-power states.

The idea of shaping traffic in this way in order to reduce power consumption is not novel. Most of the previous solutions are based on the following approaches: i) shaping the traffic at the server [1], [3] ii) shaping the traffic at the proxy [4], [5] iii) history based [1], [6]–[8] or proxy assisted prediction [4], [8] at the client to switch off the WNI or transits to the sleep mode during the idle periods, iv) reducing the total transfer delay by transcoding [8] v) new power saving protocol [4]. In addition, they also discard the standard PSM altogether and propose customized scheduling solutions for the MAC layer.

Our approach differs from those above by relying on the standard PSM and simply placing a proxy in between the end points. In this way, neither the mobile phone's or access point's protocol suites and drivers, nor the server or application data need to be modified. We study the potential for power savings in such a straightforward scenario considering the location of the proxy, the bit rate of the audio stream, and the length of the buffering period. In addition, our work is the first to investigate

the effectiveness of such mechanisms with 3G access.

We report in this paper that our proxy mechanism can save power up to 65% depending on the stream rate. If the proxy is located far away from the mobile device, the achieved power savings vary from 55% to 60%. We also evaluate the impact of local area cross traffic in the case of WLAN access. As for the case of 3G access, we discover that the proxy mechanism works well with some phone model and 3G operator combinations, but with some combinations it fails to deliver significant power savings due to abnormal delay and peculiar flow control behavior by the TCP protocol.

II. Internet Radio and Mobile Power Consumption

A. Energy Inefficiency of 802.11 PSM and 3G

802.11 PSM behaves well if the traffic is periodic or *bursty* in nature since that allows the WNI to spend time in the *sleep* state and just to wake up periodically to receive a burst of traffic. Multimedia streaming traffic often resembles constant bit rate traffic with evenly spaced out packets. Hence, the mobile client WNI is most of the time in the *active* state while using streaming applications over WLAN. It is a well known fact that PSM works inefficiently in this kind of scenario [1]. The reason is the overhead of its underlying scheduling mechanism which requires the client to explicitly request queued data from the access point. In addition, there is a fixed cost associated with the transitioning from one state to another that depends on the hardware. Therefore, the fewer are these transitions, the better is the power efficiency.

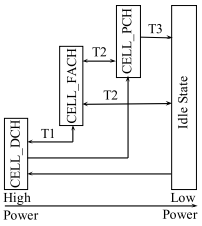

Fig. 1. WCDMA 3G States

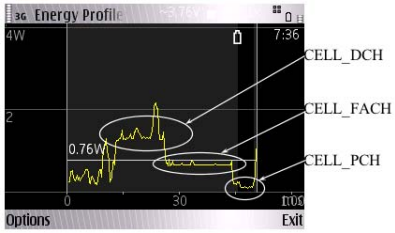

Fig. 2. 3G Power Consumption with Nokia E-71

In the case of 3G, there are three states [9], as illustrated in Figure 1. At CELL_DCH state, a dedicated data channel and timer T1 are assigned to a mobile phone. Timer T2 and a shared data channel are assigned at CELL_FACH state. A paging channel and timer T3 are allocated at CELL_PCH state. If the data channel is inactive for T1 time then mobile switches from CELL_DCH state to CELL_FACH state. Again, if the mobile is idle in CELL_FACH state for T2 time it transits to the CELL_PCH or idle state and so on. As illustrated by the Nokia Energy Profiler (NEP) [10] screenshot (Figure 2), CELL_DCH is the most power consuming state (1.4W with the tested phone), CELL_FACH is the second (0.7W), after which come the CELL_PCH and idle states. However, the values of these timers are in order of seconds and service provider specific. In the case of our target application, the continuous flow of audio stream together with these inactivity timers may prevent the mobile station from switching to a lower power consuming state. Thus, the power consumption will be constantly high.

B. Baseline Power Measurements

In all the experiments described in this paper, we used the internet radio application that comes with Nokia phones. We used mainly E-71 for the experiments but used also N95 and N900 in the 3G experiments (cf. Section IV-C). The underlying protocol is TCP and it sends a single HTTP GET request to the radio station in order to start the streaming.

In the beginning of a connection, radio stations commonly use *fast start streaming* [2] technique to fill out the client application play-out buffer very quickly. Soon after the buffer is filled, server sends traffic at the stream-encoding rate. We worked with three different Internet radio stations* with different media encoding rates (Table I). We used NEP to measure the average power consumption at the mobile phone while playing the radios.

Station Id	Data Rate (kBps)	Start-up Time (Sec)	WLAN		3G	
			PSM (W)	CAM (W)	48kBps (W)	2Mbps (W)
1	8	18	0.53	1.06	1.30	1.30
2	16	10	0.99	1.07	1.30	1.30
3	24	10	1.04	1.07	1.27	1.35

TABLE I
Power Consumption of Internet Radio Stations

The results shown in Table I confirm the anticipated power consumption behavior of mobile device with streaming. We can see that if the wireless interface is in continuous active mode (CAM), the power consumption is high and almost same for all radio stations, as expected. Indeed, since the interface is always in active mode, the encoding rate of the stream does not make any difference. On the contrary, while using PSM, the mobile phone uses significantly less power with the lowest encoding rate, but fails to deliver similar power savings with higher data rate streams. In case of 3G, we have used two different data rate subscription with 48kBps and 2.0Mbps downlink rate. The table reveals that the power consumption is overall clearly higher than with WLAN access and does not seem to be dependent on the streaming rate of the radio stations or the downlink rate of the subscription.

III. Traffic Shaping With a Proxy

Streaming traffic rarely utilizes all the available bandwidth of the underlying TCP/IP path. Therefore, it is possible to reshape the traffic by "squeezing" the constant low bit rate traffic into higher bit rate bursts, which is indeed the purpose of our proxy. It repeatedly buffers the radio stream and forwards the buffered stream in a single burst to the mobile client. Since the proxy does not send any data to the mobile while buffering and PSM is enabled at the mobile station, the interface will be

*1) stream.rawfm.com.au:8004 2) radio.internode.on.net:8100 3) 83.145.201.209:8000

in the active state only to receive the periodic bursts. Therefore, there are fewer transitions and power consumption is reduced. In case of 3G, the goal is that the proxy allows the mobile to be idle during long enough buffering periods so that the inactivity timers get to expire and the phone switches to the lower power consuming states.

According to our proposal, client sends the play request to the proxy server using HTTP protocol over TCP. Proxy establishes another TCP connection with the radio station and forwards the client request. Then, it relays the server response to the client which contains stream meta data followed by a continuous flow of MP3 or AAC media stream. Proxy accepts client request on a TCP stream server socket and connects to the radio server using the client socket. Afterwards, the media data is simply forwarded from one socket to another.

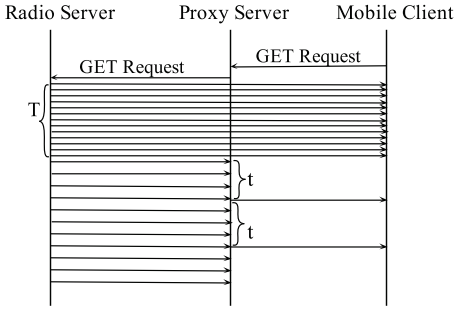

Fig. 3. Proxy Mechanism

We introduce two timers: *start-up* timer T and *buffering period* timer t. The first timer starts when the proxy forwards the stream request to the server. Consequently, the streaming server sends traffic to the client using fast start streaming technique to fill the play-out buffer quickly, as illustrated in Figure 3. During this period, proxy simply forwards traffic to the client, measures the amount of traffic transmitted to the client, i.e the size of the play-out buffer and calculates the maximum amount of time that the client player can play until the play-out buffer is empty.

When the *start-up* timer is expired, the second timer t is activated. During this period, proxy buffers media traffic and sends the buffered traffic in a single burst to the mobile client as shown in Figure 3. The second timer is repeatedly initialized and this process continues until the connection is closed. The proxy parses the media encoding rate of the radio stream from the server HTTP response. However, in absence of fast start streaming, buffering at the proxy may cause additional *start-up* delay.

If the encoding rate is *E* bytes per second and *K* bytes are transferred to the client during the initial *start-up* time *T*, then the maximum amount of time the proxy can buffer without affecting the stream quality and smooth playback is $t_{max} = \frac{K}{E}$. Thus, the amount of data to be sent in a single burst is limited by the t_{max}: $BurstSize = E \times t; t \in [0, t_{max}]$. Note that the start-up time at the client can be shorter than the fast start streaming phase, i.e. the client can start playing already before the fast start phase is over. The client will still accumulate data in the play-out buffer during the fast streaming phase since the transmission rate is by definition clearly higher than the encoding rate. The problem is that the proxy does not know when the client starts to play the stream and, therefore, is unaware of the exact amount of data buffered during the fast start phase, which is required for computing t_{max}. In our experiments, we manually measured the value for client start-up delay T (typically 10-20s) beforehand and configured it to the proxy. However, one solution to the problem is to assume $T = 0$, i.e. the client starts to play the stream immediately, and to compute a lower bound for t_{max} at the proxy as follows: $\lfloor t^{lb}_{max} \rfloor = \frac{T_{fs} \times (r_{fs} - E)}{E}$, where T_{fs} and r_{fs} are the duration of and transmission rate during the fast streaming phase, respectively. This lower bound can be automatically computed by the proxy and it turns out to be more than sufficiently long in many cases. For instance, we obtain $t^{lb}_{max} = 15$s for one of our experiments where the optimal value for t is 5-7s, as we show in Section IV-A.

IV. Evaluation

A. Local Proxy Over WLAN

In the beginning of our experiment we hosted the proxy in the same local network with the mobile client.

We conducted experiments on three Internet radio stations (Table I). Each of the radio stations was played through the proxy server several times, each time with a different length of the buffering period. During each experiment, we measured the average power consumption using the NEP. This average includes the initial high power due to the *start-up* buffering at the application, display and back light. However, since every time the duration of playing has been roughly 20 minutes, the initial high power consumption is negligible in the results.

The start-up time for *radio.internode.on.net* is about 10 seconds which we calculated by playing the radio without the proxy. Thus, we set $T = 10s$. The amount of data transferred during this period to the client using fast streaming is 410kB (*K*). The client starts playing when f_s phase is over. With a streaming rate of 16kBps (*E*), we obtain t_{max} = *K/E* = *25*s.

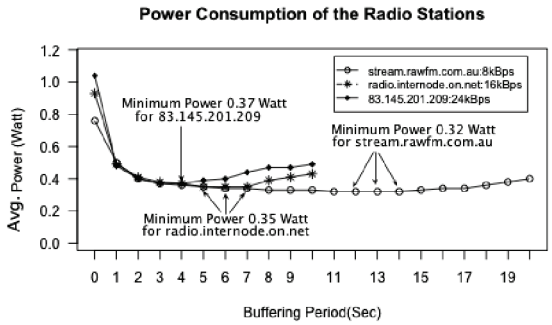

Fig. 4. Power Consumption of the Radio Stations

Figure 4 shows the average power consumption at the mobile device with different buffering periods t at the proxy

server. We notice that even short buffering at the proxy delivers significant power savings. The power savings become larger when t is increased. The quality of the audio stream at the client remains the same throughout the experiments. We observe that for 16kBps radio there are values for t ($5-7$s) that lead to minimal power consumption of 0.35W. Compared to the PSM without the proxy, the power consumption is reduced from 0.99W by 65%. For the two other radio stations the power consumption is also reduced in similar fashion as shown in Figure 4.

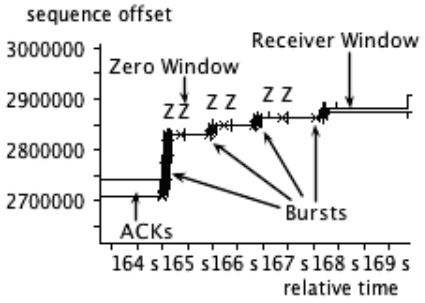

Fig. 5. Time Sequence Graph: Burst Split at $t = 10$

Contrary to what we expected, Figure 4 shows that if t is increased enough, the power consumption starts slowly increasing again. The reason is the following: We observed that when t is long enough, the proxy generated bursts are split into several smaller bursts due to zero window advertisements (ZWA) from the client. In this case, TCP sender overflows the receiver's buffer and flow control kicks in, i.e. the client TCP sends ZWA to the sender in order to pause the transmission. Client sends the requests with proper receiving window size only when enough buffer is freed. As a result, the mobile client switches its interface between active and sleep states several times corresponding to the smaller bursts instead of just once, hence, leading to increase the power consumption. The time sequence graph (Figure 5) shows an example of a burst is getting split into multiple bursts by ZWAs from the client for the 16kBps radio.

Besides serving a single mobile client we also checked mobile power consumption locally in cross traffic scenario with another client sharing the same AP. We observed that potential mobile power consumption increases with the download rate at the other client. When the download rate were 1.2 and 2.4MBps minimum power consumption for the second radio station increased from 0.35W to 0.38W and 0.41W respectively. However, power consumption increased aggressively, when the download rate was split to multiple downloads on the same client. For example, if the 2.4MBps download was split into two 1.2MBps download connections, the power consumption increased to 0.55W.

B. Remote Proxy Over WLAN

The deployment is challenging in the previously described local proxy scenario because a proxy should be installed in each WLAN access point. On the contrary, it is simpler to set up some number of proxies at remote locations in the Internet each of which would serve clients from several local networks. However, if the traffic shaper is not close to the client, there is a good probability that the proxy-generated bursts are reshaped on the way due to the cross traffic, variation in link capacities, increased RTT, etc. In order to understand the impact of those effects on the power consumption, we set up our proxy at different remote locations, namely in the UK and the USA using the Amazon EC2 services [11]. The RTTs of the mobile client from the UK and USA proxies were 56ms and 108ms (approx.), respectively.

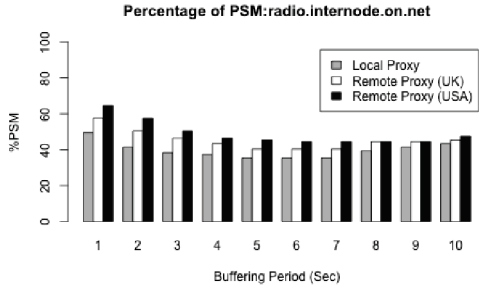

Fig. 6. Power Consumption: Percentage of PSM

From these experiments, we noticed that the power consumption behavior is similar to that of the local proxy scenario: power consumption decreases when t is increased until a minimum is reached after which the power consumption again increases with larger values of t. However, as we can see in Figure 6, the further away is the proxy, the higher is the power consumption.

The main reason for this phenomenon turned out to be the increased RTT. Incrementing the size of the TCP congestion window (CWND) is a function of the RTT (each correctly received ACK increases the window). As a consequence, the RTT directly impacts the time it takes to transmit a specific amount data: the longer the RTT, the longer the transmission.

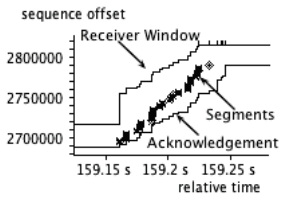

Fig. 7. Single Burst from Local Proxy

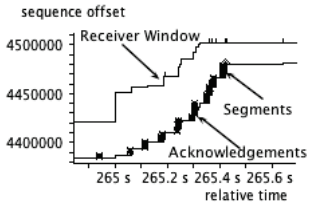

Fig. 8. Single Burst from Remote Proxy (UK)

The time sequence graphs in Figures 7 and 8 show the transmission of the segments belonging to a single 96KB burst generated from the local and remote UK proxy, respectively. The TCP sender on local proxy having a very short RTT increases the congestion window size very quickly: it takes

about 100ms to complete the transfer of the entire burst. In the case of the remote proxy, it takes about 500ms to transfer the burst. This behavior persists throughout the connection for every burst, which means that the sending TCP resets the CWND to a small value during the idle periods in between the bursts.

C. Proxy over 3G

In the case of 3G, our intention for using the proxy is to enable the inactivity timer $T1$ or both $T1$ and $T2$ to expire during the buffering period t. So, the mobile station will be in the CELL_FACH state at least for $t - T1$ seconds or in CELL_PCH state for $t - (T1 + T2)$ seconds.

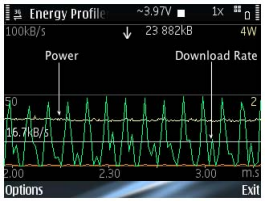

Fig. 9. Power and Data View at $t = 6$

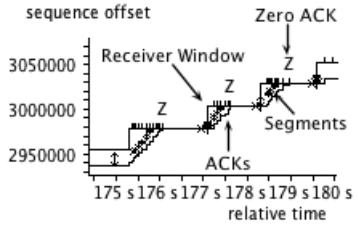

Fig. 10. Time Sequence Graph: ZWA at $t = 6$

The effectiveness of the proxy turned out to vary quite a lot in our experiments depending on the 3G operator and phone model. The NEP screenshot in Figure 9 is viewing the power consumption and instantaneous download rate on E-71 phone using a particular 3G operator (OP1) during an example streaming session from the 16kBps rate station with $t = 6s$ and 2Mbps subscription. We observe that it takes for each burst almost the entire six seconds to receive it completely. For this reason, the $T1$ timer never gets to expire and power consumption is constantly high.

The size of the burst in the example of Figure 9 is 96KB which should not take nowhere as long as six seconds to transfer. The reason turned out to be unexpected flow control behavior. The time vs. sequence diagram in Figure 10 visualizes a piece of this transfer containing roughly two proxy generated bursts. We notice that the bursts are split into several again due to ZWAs causing the long transfer times.

In addition to the previous example, we observed similar ZWAs and continuous oscillation in the receiving window size advertised by the client also with smaller values of t regardless of the subscription and stream rate, even with $t = 1$. In addition, we also witnessed sudden increases of RTT from a few hundred milliseconds to several seconds which caused again unusually long burst transfer times.

We also experimented with two other mobile devices (N95 and N900) with OP1. We noticed that while using OP1 with N95, TCP behavior was similar to the above described. However, with N900, this active flow control behavior with small (zero) window advertisement was almost completely absent, and the bursts generated by the proxy were not significantly affected. As a result, by using the proxy with buffering period of 10s and 20s, the power consumption was reduced to 94% and 77%, respectively, for a 16kBps radio stream over 0.5Mb/s subscription †. In addition, we experimented with another operator (OP2). E-71 with OP2 did not show signs of improvement in TCP behavior, whereas with N95 and N900 we noticed large window size advertisements allowing almost unaffected bursts from the proxy.

V. Conclusion

We have done extensive study and analysis of the effects of a traffic shaping proxy on power consumption of audio streaming for mobile devices. Using WLAN, we have discovered that the achievable power savings are from 30% to 65% depending on the buffering period, stream rate, location of the proxy, and cross traffic situation in the local WLAN network. In case of 3G, the power savings vary between phone models and operators. Certain combinations deliver encouraging energy savings while other combinations exhibit abnormal delay variation and TCP flow control behavior.

We plan to investigate further the origins of the unexpected flow control behavior of TCP in the case of 3G. Moreover, in the case of WLAN, we intend to find how reliably we can automatically identify the optimal buffering period at the proxy by studying the received ZWAs. Some servers may already transmit streaming traffic in form of bursts in which case there is no need for the proxy to intervene. Therefore, we also want to develop a solution to automatically detect when the proxy should actively shape the stream and when not. Finally, we plan to extend our proxy server for video streaming applications.

References

[1] Chandra, Surendar, Vahdat, and Amin, "Application-specific network management for energy-aware streaming of popular multimedia formats," in *ATEC '02*, 2002, pp. 329–342.

[2] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, "Delving into internet streaming media delivery: a quality and resource utilization perspective," in *IMC '06*. ACM, pp. 217–230.

[3] E. Tan, L. Guo, S. Chen, and X. Zhang, "Psm-throttling: Minimizing energy consumption for bulk data communications in wlans," in *ICNP '07*, pp. 123–132.

[4] G. Anastasi, M. Conti, E. Gregori, A. Passarella, and L. Pelusi, "A power-aware multimedia streaming protocol for mobile users," in *ICPS '05*. IEEE, pp. 371–80.

[5] M. Gundlach, S. Doster, H. Yan, D. K. Lowenthal, S. A. Watterson, and S. Chandra, "Dynamic, power-aware scheduling for mobile clients using a transparent proxy," in *ICPP '04*. IEEE, pp. 557–565.

[6] Chandra and Surendar, "Wireless network interface energy consumption implications for popular streaming formats," in *MMCN '02*, pp. 85–99.

[7] Y. Wei, Chandra, Surendar, Bhandarkar, and Suchendra, "A statistical prediction-based scheme for energy-aware multimedia data streaming," in *WCNC '04, IEEE*, vol. 4, pp. 2053–2057.

[8] Shenoy, P. J., Radkov, and Peter, "Proxy-assisted power-friendly streaming to mobile devices," in *MMCN '03*, pp. 177–191.

[9] S.-R. Yang, S.-Y. Yan, and H.-N. Hung, "Modeling umts power saving with bursty packet data traffic," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 1398–1409, 2007.

[10] G. Bosch and M. Kuulusa, "Optimizing mobile software with built-in power profiling," in *Mobile Phone Programming and Its Application to Wireless Networking*. Springer, 2007, pp. 449–462.

[11] "Aec2. http://aws.amazon.com/ec2."

†We had access to new, yet unreleased NEP software for Maemo which enabled these measurements.