Power Estimation for Mobile Applications with Profile-Driven Battery Traces

Chengke Wang, Fengrun Yan, Yao Guo, Xiangqun Chen
Key Laboratory of High-Confidence Software Technologies (Ministry of Education)
School of EECS, Peking University, Beijing, China
{wangchk11, yanfr11, yaoguo, cherry}@sei.pku.edu.cn

Abstract

It becomes very important to understand power characteristics of mobile applications because more and more complex applications are running on modern smartphones. Although many techniques have been proposed to estimate the power dissipation rate for mobile applications, it typically requires hardware support (i.e., power meters) or complex power models (software profiling or hardware parameters). These techniques might work well in labs with a small set of applications. However, it becomes impractical when we try to estimate the power of mobile applications in an uncontrolled environment.

This paper proposes a novel method for estimating the power consumption of mobile applications with profile-based battery traces. Battery traces can be easily collected through a user-level application on any devices. Although it is difficult to achieve accurate results for only a few users because battery changes are coarse-grained, the method is expected to reach an accurate estimation when the number of battery traces reaches a certain scale. Our experiments based on battery traces from more than 80,000 users demonstrate that it is possible to estimate application power with only coarse-grained battery traces. The results are also validated with measured power numbers from a Monsoon power monitor.

1. Introduction

As mobile devices such as smartphoens and tablets have more powerful functions and stronger calculation capacity, they can support a lot of complex applications. These applications might reduce the battery life to as short as several hours for many users. Thus it becomes important to understand how batteries are consumed by mobile applications.

In order to understand the power consumption pattern of mobile applications, we need to either measure or estimate the power dissipation rate of different mobile applications on mobile devices. Although measuring power with meters could achieve the most accurate results, it only applies in a lab environment and cannot be widely applied on a variety of mobile devices in the wild. Another issue with measured power numbers is that they are for the whole device and cannot be easily attributed to specific applications without software support.

On the other hand, many researchers have built different power models on mobile devices. Once we setup a power model, it becomes possible to estimate application power online or with profiled statistics. Existing power models can be classified into two main categories: hardware-based model and software-based model.

Hardware-based models are typically built according to the power dissipation rate of each energy-consuming components on a mobile device [8, 9, 16, 18]. The first step is build a power model for every power consuming component on the mobile devices [1, 2, 4, 14], such as CPU, OLED or LCD screen, network, cellular and sensors. Then, based on the working state timing information collected from online monitoring or offline profiling, we could calculate the power numbers of each component and estimate the power consumption of an application by summing up the power consumption of all relevant components. [11, 12, 17] A major weakness of this method is that it must consider a lot of hardware components in order to build a more accurate (usually more complex) model. However, the power consumption of applications can be greatly different in different running environment or with different system configuration.

Another way is to build a software-based power model [10, 15], which could be based on program analysis, profiling, or other software techniques. For example, we can build a power model for each software component, which could be a subroutine or an execution path. Based on the model and software execution traces, we could calculate the total power/energy consumption for each application. It is also possible to estimate software energy at compile-time with static analysis [7], however, these work has been in the early stages and have not been applied widely.

The above techniques might work well in labs with a small set of applications. However, it becomes impractical when we try to estimate the power of mobile applications in an uncontrolled environment. For example, software-based estimation sometimes requires extensive instrumentation and profiling of every application [10], or requires modification to the device operating system [15], which cannot be applied widely in a large-scale study.

In order to alleviate these challenges, our main objective is to perform power estimation based on easy-to-collect tracing data, such that the method could be applied to most smartphones straightforwardly. One potential candidate of power data is battery traces, which can be collected with a user-level service. An application could be easily written to collect battery traces and application context (switching) information on any devices.

However, directly using battery traces to calculate application power is very difficult because the battery traces are usually coarse-grained. For example, most Android smartphones could only record their battery levels at one-percentage granularity. (Some Android phone models even uses minimum battery levels at 10 percent granularity, these phone models will be excluded from our research.) Because many application usage periods are so short that they will consume less than 1% of battery, they might not be able to record any battery changes even they actually used the battery. In this case, if we calculate average power with these coarse-grained battery traces, we might end up with skewed power numbers.

This paper proposes a new method to estimate power consumption of mobile applications by profile-based battery traces. Our method use a statistical method to estimate power consumption with large-scale battery traces to reduce the error of direct calculation. With the application switching information and battery level drops in all battery traces, we are able to calculate a combined average power consumption rate for each mobile application based on traces collected from thousands of users. Although it is difficult to achieve accurate results for only a few users because battery changes are coarse-grained, the method is expected to reach an accurate estimation when the number of battery traces reaches a certain scale.

We developed an Android application to perform data collection. Battery traces have been collected from over 80,000 valid users. We evaluated our estimation method to calculate power consumption rate for several popular applications with different user quantities. The results show that when the scale of users is small, the variance of our method is significant. But once the number of users reaches beyond 10,000, the estimated power numbers start to converge to a relatively stable number. The results are also validated with measured power numbers using a state-of-the-art power monitor on smartphones devices.

The main contribution of this paper is that we introduce a method to calculate accurate power dissipation rate for mobile applications based on only coarse-grained battery traces. Our experiments show that it is feasible to calculate mobile application power accurately once the collected battery traces reach a certain large scale. The method is very easy to implement compared to previous work and could be applied to millions of mobile devices easily.

The rest of this paper is organized as follows. Section 2 described the background of our study. Section 3 presents the energy estimation method based on battery traces. Section 4 presents and analyzes the evaluation results. We describe the related work in Section 5 and conclude with Section 6.

2. Background

In this section, we introduce the background of our work, including how to collect battery traces and what is included in the traces.

2.1 Data Collection

We have developed a simple application for the Android platform to collect battery and application related user data. The application consists of the following components:

- A data collection component listens to the events about batteries, applications and other components of mobile devices broadcast from the android system.
- A data storage component writes the events information to the SD-cards and compresses the data.
- A data transmission component sends the compressed data to server via Internet when the network of mobile devices is available.

The data collection tool starts a background service on the mobile devices when the devices are turned on. With this tool, we have collected data from over 120,000 users for about 4 weeks. After filtering out invalid users and unusable traces, we use more than 80,000 valid users in our study.

2.2 Battery Traces

The collected data includes a lot of information of events and mobile devices, such as network events, screen events, device type, etc. In this paper, we are concerned with only the battery traces for all applications.

A battery trace consists of two parts of information, *the battery level change events and the application switching events*. Each of these events is attached with a timestamp. All the battery level change events within a continuous running period of a foreground application is recorded as a battery trace of that application. Because a user might start and close an application many times, there exist many battery traces corresponding to one specific application.

2.2.1 Battery Level Change Events

On mobile operating systems such as Android, battery level is normally used to represent the remaining percentage of battery capacity accurate to 1%. When the battery level changes(goes up or goes down), the operating system will broadcast an event indicating the current battery level. We collect these events and add a timestamp to each of them to record the battery level history of the mobile device.

In the same device, every 1% battery level represents the same battery power. But when it comes to devices who have different battery capacities, 1% battery levels of different mobile devices always represent different battery power values. Thus, during calculation with different mobile devices, we must consider their battery capacities. We collected the battery capacity information based on the standard configuration of each mobile devices and use this information together with the battery levels.

2.2.2 Application Switching Events

Besides battery level changes, we also need to know which application is running at a given moment. When an application starts to run or an application is switched to the foreground, the operating system will broadcast an event including the application name. We collect these events and add a timestamp to each of them to record the foreground application history on the mobile device.

We are concerned with the following three types of applications:

- Normal applications: Most of applications in our data belongs to this type. We denote the running time of these applications with the application start events and application switch events. If there is a relative long period of screen off in the running period of an application, we considered the screen off period as in standby mode instead of the running period of the application.
- Standby: We denote the standby mode as the time periods while the screen is off. However, the screen may be turned off automatic when users do not operate the devices for a while. If users turned on the screen immediately, we do not regard this screen off time period as standby.
- Phone calls: In our study, phone calls are regarded as a special kind of application. We denote the time periods of phone calls by the phone call events. Since smartphones will typically turn off the screen during calling to save battery, we also do not regard this screen off time period as standby.

3. Application Power Estimation

In this section, we present our method to calculate the power consumption rate for each mobile application, with the collected battery traces described above.

3.1 Challenges

Our basic method is trying to calculate the average power consumption rate for each application based on the collected battery traces. However, we face many challenges:

- The accuracy of battery level is typically 1% of the battery capacity. Thus we could not detect the battery level changes that are less than 1%. For some shorter battery traces, we usually get zero battery level drop. This indicates that if we calculate the power for this specific trace, the power is always 0.
- Battery charging events might occur during the application running period. We must eliminate the impact of charging when calculating the power consumption.
- Background applications and operating systems also consume battery. However, we could not separate these battery consumption from the foreground applications. For simplicity, we assume that at each moment only the foreground application is consuming the battery.
- The timestamps of battery level change events and application events are often different of each other. Thus, we could not get the exact battery level when an application event occurs.

3.2 Battery Trace Distribution

Figure 1 shows the distribution of battery traces based on the energy level drops (in percentages of battery capacity). We could see that about 90% of the battery traces have consumed no energy because their usage period is too short. If we

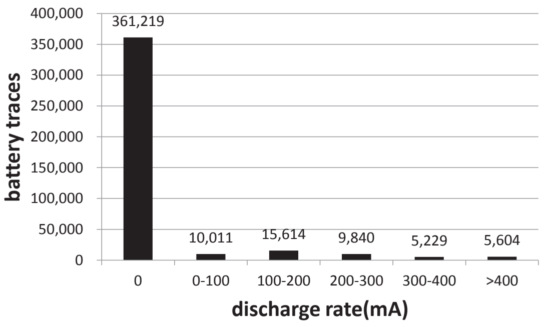


Figure 1: The distribution of battery traces.

calculate average power with these battery traces, the result would be 0. This shows that it is impossible to achieve an accurate power number with only a few battery traces. However, if we calculate the average combined power for a large scale of energy traces, we expect that we might get an accurate number if the scale of the traces is large enough.

3.3 Power Estimation Method

Our estimation method is to calculate the power consumption of applications based on statistics with large-scale battery traces to reduce the error during calculation. Firstly, we calculate the running time and battery level drops of every battery trace, then simply combine them together to calculate the *TotalUsingTime* and *TotalBatteryLevelDrops*.

We denote the power consumption by the discharge rate. The average discharge rate of each application can be calculated as follows:

$$AvgDischargeRate = \frac{TotalBatteryLevelDrops}{TotalUsingTime}$$

Besides the coarse-grained battery level issue, we deal with the other challenges described above in Section 3.1 as follows:

- To deal with the charging influence, we detect the charging events by the battery level change events. We consider that the mobile device is charging if the battery level goes up. When calculating the power consumption, we discard the battery traces with charging events, consider the running periods without charging.
- To deal with the background battery consumption, we regard all the battery level drops as consumed by the foreground application. It is acceptable because foreground applications usually use the CPU, screen and other components much more than the background applications and system services, therefore consuming more energy. We consider the background battery consumption as the general running environment of the foreground application in our large-scale study.
- Because of the timestamp problem, when we calculate the battery level drops, if we use the battery level of the nearest battery level change events to represent the battery level of the application switch events, there may be some errors for each battery trace. However, since the battery levels always change by 1%, the error of one battery trace is less than 1%. We suppose that the error could be

reduced to a tolerable level by the statistic method with large-scale analysis.

4. Evaluation

In this section, we present our evaluation method and the results.

Although directly using each of the battery traces to calculate the power consumption would be inaccurate, we expect that the large-scale analysis could reduce the error into a tolerable level. Thus we perform a series of evaluation of different scale to evaluate our estimation method.

4.1 Experimental Setups

Since the duration of battery traces vary from several seconds to several hours, it is inappropriate to represent the scale of battery traces by the number of battery traces. In our evaluation work, we denote the scale of battery traces by the number of users.

In our evaluation, we select several different scales of users to calculate the power consumption, from 10 users to all the 80,000 users. We randomly select 5 groups of users at every scales, from 10 users to about 10,000 users. Then we calculate the average and standard deviation of the results of every scales.

4.2 Estimation with Different User Scales

Figure 2 shows the results of calculated power consumption of phonecall with different scales of battery traces. It is obvious that the variance becomes smaller when the scale of battery traces increases. Then we analyzed the average, variance and the standard deviation (shown in Figure 3(a) and Figure 3(d)). At the scale of 10 users, the calculated power consumption vary from 0 mA to 474.55 mA, while converge to a variance of 278.89 mA to 309.13 mA at the scale of 10,000 users. At the scale of 10,000 users, the average power consumption is 296.95 mA while the standard deviation is 12.90 mA (about 4.3% of the average power consumption).

This is because about 90% of the battery traces have consumed no energy. If we calculated the power consumption with 10 random users, the probability is very high that all battery traces of these users are zero-energy-consuming. Thus, the calculated power consumption could very likely be zero. When the user scale reaches a certain level, the battery traces are more likely to cover all the possible energy consumption level. Thus, we could calculate a relative stable power consumption value on average.

We also do similar experiments for other applications, Figure 3(b), Figure 3(e), Figure 3(c) and Figure 3(f) show the results of AngryBirds and SinaWeibo. The convergence of power consumption is also obvious.

The results show that the power dissipation rate converges after the number of users reaches 10,000, thus we are able to achieve an accurate estimation with the combined average.

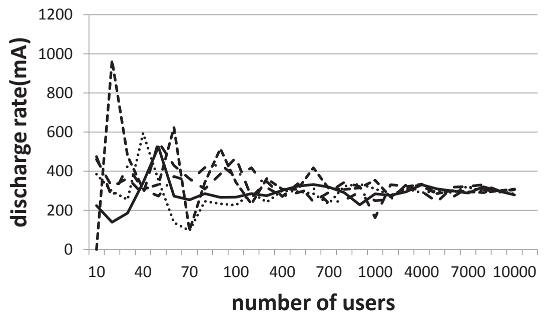


Figure 2: Calculated power of phone call at different user scales.

4.3 Comparison with Measured Power Consumption

We also measured the power consumption of applications with the Monsoon PowerMonitor, a hardware instrument to measure the current and voltage of mobile devices. The smartphone we measured is a Tianyu W806 device with dual-core Nvidia Tegra2 processor at 1GHz and 4.3 inch screen. The phone model is representative of the mainstream smartphones in the market during our data collection period.

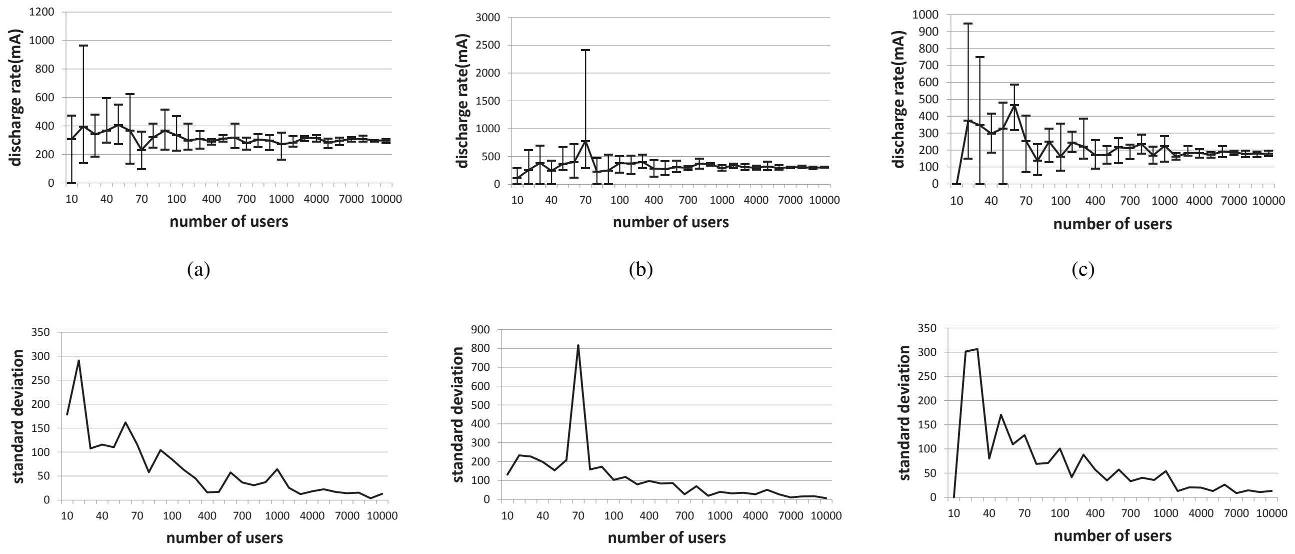
Figure 4 shows the comparison of the calculated power consumption and measured power consumption. The calculated average discharge rates are 178.48 mA, 296.95 mA and 307.03 mA (as shown in Figure 3), while the measured current are 218.30 mA 262.46 mA and 322.79 mA (shown in Figure 4) for SinaWeibo, phone call and Angrybirds.

Surprisingly, the difference between the measured data and the calculated power numbers are very small, with all differences less than 18%. Although our data collection and energy calculation method might not be accurate enough to calculate energy numbers for individual users with limited data, this results shows that this method is good enough to estimate the energy consumption rate of applications over a large-scale data.

5. Related Work

Many research work have been focused on estimating power consumption rate of mobile devices and mobile applications, including hardware-based models for mobile device components, and software-based models based on program analysis.

In recent years, many hardware-based approaches have been proposed to estimated the power consumption rate of mobile devices and applications. Several research work have focused on developing power models for a certain component on mobile devices. For example, Dong *et. al* [4] modeled the power consumption of OLED screen at color-level, image-level, code-level, etc. Panigrahi *et. al* [14] proposed a stochastic model for mobile batteries to estimate the battery life. Balasubramanian *et. al* [2] developed a model for the energy consumed by network activity for each network technology and presented a method to reduce the tail energy



(a) (b) (c)

(d) (e) (f)

Figure 3: Estimation results at different user scales for phone call(a)(d), Angrybirds(b)(e) and Sinaweibo(c)(f). (a)(b)(c) is the average power consumption and the variance at different user scales. (d)(e)(f) is the standard deviation at different user scales.

according to the RRC protocol.

Other researchers have tried to combine the power models of these components to estimate the power of mobile devices [9, 16, 18]. For example, DevScope [8] developed an autonomous power modeling tool for smartphones. Application-level power models are usually based on the hardware component power models by detecting the working state of hardware components when applications are running and using a lot techniques to improve the accuracy of the models [11, 12, 17, 19]. These hardware-based approaches typically requires the modeling of hardware components based on either measuring or profiling.

Besides hardware-based approaches, some software-based methods have also been proposed to analyze mobile application power consumption. For example, Li *et. al* [10] proposed a system routine-based system power model with the estimation error less than 6%. Eprof *et. al* [15] could be used to perform detailed energy profiling using a system-call-based power model. These software based techniques normally requires instrumentation or other modifications on the mobile applications or mobile operating systems.

Related to our research, many large-scale user studies for battery usage and energy consumption of mobile devices have been proposed. For example, Falaki *et. al* [5] studied 255 smartphone users, characterized user activities and applications, and the impact of those activities on network and energy usage. Banerjee *et. al* [3] designed a user- and statistics-driven energy management system, and conducted a user study, which shows that their system could harvest excess battery energy for a better user experience without a noticeable change in battery lifetime. Ferreira *et. al* [6] presented a 4-week study of more than 4000 people to assess their smartphone charging habits to identify power intensive operations and to provide interventions to support better charging behavior. Oliver *et. al* [13] conducted one of the largest-scale study to measure the energy consumption of 20,100 BlackBerry smartphone users, and predicted energy level within 72% accuracy in advance. However, most of these studies have focused on the study of either battery usage patterns or user charging habits, instead of analyzing energy consumption patterns for specific mobile applications as in this paper.

6. Concluding Remarks

Estimating mobile application power is important to understand the energy consumption pattern for mobile applications. This paper proposes a simplistic power estimation method based on battery traces that are easy to acquire from any smartphones. Although the battery level changing information in these traces is coarse-grained, we are able to perform accurate power estimation while calculating a combined average power when a large scale of battery traces are available.

The proposed estimation method is practical and accurate based on our evaluation for several popular applications on battery traces collected from more than 80,000 users. The evaluation also demonstrate that this method is simpler than existing approaches and could be easily applied to millions of devices.

Our future work includes performing more detailed analysis on the traces and developing energy optimization techniques based on the calculated power numbers.

7. Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61103026, the National

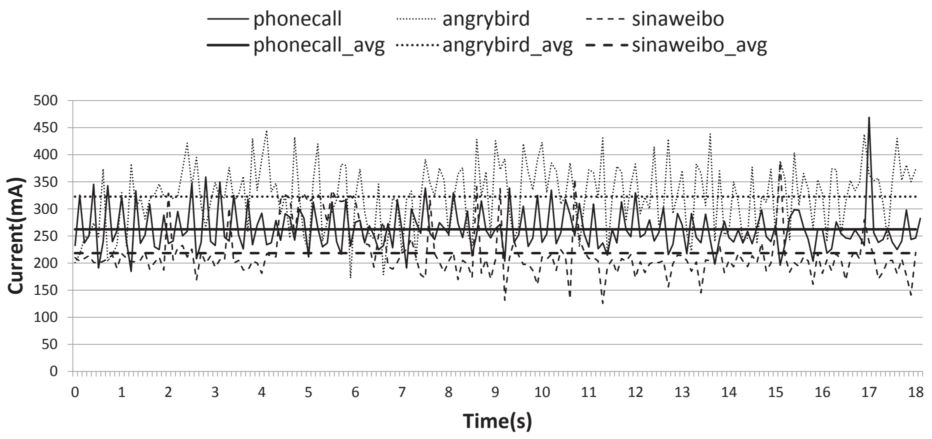


Figure 4: Measured current of the applications.

High Technology Research and Development (863) Program of China under Grant No. No. 2011AA01A202, and the National Basic Research Program of China (973) under Grant No. 2011CB302604.

References

[1] B. Balaji, J. McCullough, R. K. Gupta, and Y. Agarwal. Accurate characterization of the variability in power consumption in modern mobile processors. In *Proceedings of HotPower'12*, 2012.

[2] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of IMC'09*, 2009.

[3] N. Banerjee, A. Rahmati, M. D. Corner, S. Rollins, and L. Zhong. Users and batteries: interactions and adaptive energy management in mobile systems. In *Proceedings of UbiComp'07*, 2007.

[4] M. Dong and L. Zhong. Power modeling and optimization for oled displays. *IEEE Transactions on Mobile Computing*, 11(9), Sept. 2012.

[5] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of MobiSys'10*, 2010.

[6] D. Ferreira, A. K. Dey, and V. Kostakos. Understanding human-smartphone concerns: a study of battery life. In *Proceedings of Pervasive'11*, 2011.

[7] S. Hao, D. Li, W. G. Halfond, and R. Govindan. Estimating mobile application energy consumption using program analysis. In *Proceedings of ICSE'13*, 2013.

[8] W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha. Devscope: a nonintrusive and online power analysis tool for smartphone hardware components. In *Proceedings of CODES+ISSS'12*, 2012.

[9] M. Kim, J. Kong, and S. W. Chung. Enhancing online power estimation accuracy for smartphones. *Consumer Electronics, IEEE Transactions on*, 58(2):333–339, 2012.

[10] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. In *Proceedings of SIGMETRICS'03*, 2003.

[11] R. Mittal, A. Kansal, and R. Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the Mobicom'12*, 2012.

[12] R. Murmuria, J. Medsger, A. Stavrou, and J. M. Voas. Mobile application and device power usage measurements. In *Proceedings of SERE'12*, 2012.

[13] E. A. Oliver and S. Keshav. An empirical approach to smartphone energy level prediction. In *Proceedings of UbiComp'11*, 2011.

[14] D. Panigrahi, S. Dey, R. Rao, K. Lahiri, C. Chiasserini, and A. Raghunathan. Battery life estimation of mobile embedded systems. In *Proceedings of VLSID'01*, 2001.

[15] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of EuroSys'11*, 2011.

[16] Y. Xiao, R. Bhaumik, Z. Yang, M. Siekkinen, P. Savolainen, and A. Yla-Jaaski. A system-level model for runtime power estimation on mobile devices. In *Proceedings of GREENCOM-CPSCOM'10*, 2010.

[17] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha. Appscope: application energy metering framework for android smartphones using kernel activity monitoring. In *Proceedings of USENIX ATC'12*, 2012.

[18] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the CODES/ISSS'10*, 2010.

[19] X. Zhao, Y. Guo, Q. Feng, and X. Chen. A system context-aware approach for battery lifetime prediction in smart phones. In *Proceedings of SAC'11*, 2011.