# Lightweight Online Power Monitoring and Control for Mobile Applications

Bo Wang Xinghui Zhao
School of Engineering and Computer Science
Washington State University
14204 NE Salmon Creek Ave
Vancouver, WA 98686
{bo.wang,x.zhao}@wsu.edu

David Chiu
Department of Mathematics and Computer Science
University of Puget Sound
1500 N. Warner St,
Tacoma, WA 98416
dchiu@pugetsound.edu

***Abstract*—Limited battery power has long been a challenge for mobile applications. As a result, the work in power monitoring and management has attracted a great amount of interest. In this paper, we propose a model to estimate power consumption of mobile applications at run-time, based on application-specific per-action power profiling. In addition, we have developed on-line optimization techniques which help maximize users' experience while conserving power. Our power model is lightweight and flexible, in that it can be used by any mobile applications as a plug-in, and it can support user-defined optimization mechanisms. This approach has been evaluated using a case study, a mobile application for field studies, and the experimental results show that our model accurately captures power consumption of the application, and the model can be used to optimize the power consumption based on users' needs.**

## I. Introduction

There has been rapid growth in the population of mobile device holders. As shown in a recent study [1], by the year 2013, 91% of the world population have access to smart phones. This number is expected to continue growing, and in fact, the number of mobile devices in use will soon exceed the worldwide population by 2017 [2]. As for the time mobile phone holders spent on these devices, 80% is within various applications [1]. Within the past decade, both the number of mobile applications as well as their functionalities have been dramatically increasing. However, despite the widespread functionalities that mobile applications provide, they all rely on the battery life of the hosting devices to operate, which has become a major bottleneck [3]. As a result, approaches in power modeling and management for mobile devices have been attracting a significant amount of attention. While most of the existing power modeling and optimization approaches focus on profiling and controlling power consumption at the level of system [4] or hardware [5], we believe that an application level, lightweight power modeling and management technique is more flexible and highly desirable.

Consider a motivating example: a group of scientists wish carry out a field study, for which they use a mobile app to track locations, collect samples, communicate with each other, *etc.* During the trip, this app will be the dominating application running on the mobile devices of all the participants. Here in this case, the power optimization objective is very application-specific. To guarantee the quality of the field study, if the battery power is sufficient, the application should be executed as is, and there is no need to perform power optimization. On the other hand, when the battery is not sufficient for the field trip, power optimization related actions should be taken. In that case, however, the objective of the optimization then is to conserve the power consumption so that the battery life can last until the end of the field study, instead of optimizing the power consumption as much as possible for the entire system at all time, as seen in generic power management approaches for mobile devices. Therefore, the power modeling and management module must be application-specific for multiple reasons. First, it should be aware of the power consumption of each type of user actions in the application, in order to provide fine-grained power monitoring and control. Second, it must be able to invoke the power optimization when it is necessary to do so, instead of running it all the time. Third, the power optimization should aim for conserving power to a certain extent so that the predefined tasks can be completed, while still maximizing users' experience. These objectives are difficult to achieve without application-level power modeling and management.

In this paper, we address these challenges by developing a power modeling, monitoring, and management framework. Specifically, we first propose a model which can be used to estimate the power consumption of a mobile application at run-time. This model takes into consideration both the system parameters and application-specific user actions. We show that based on a set of per-action power profiling data, the model can accurately estimate the power consumption of the application over time. We then demonstrate how the proposed model can be incorporated with the power profiling data of an application, and used as the application's power control plug-in, for monitoring and optimizing power consumption as needed at run-time. To this end, we have implemented the motivating example described above, and performed a case study in the context of an actual usage scenario in field science. Finally, we have developed and evaluated different online optimization techniques which reflect users' needs. The experimental results show that our model can accurately capture the power consumption of the application over time, and it can be used to support user-defined run-time optimizations.

The contributions of this work are multifold:

- The power management module is lightweight, and easy to implement as a plug-in instrument for mobile applications.
- Our approach monitors and dynamically regulates power consumption for mobile applications at run-time while maximizing user experience.
- To test our framework's applicability and to measure its effectiveness, we performed a case study in the context of an actual usage scenario in field science.
- We have open-sourced our power management framework for the Apple iOS [6].

The rest of the paper is organized as follows. Section II reviews related work. Section III presents our approach, including power profiling and modeling. Section IV uses a case study, a mobile application for scentific field study, to illustrate how our model can be incorporated in a mobile application. To evaluate the effectiveness of our approach, experiments have been carried out, and the exeperimental results are presented in Section V, and Section VI concludes the paper and proposes future directions of this work.

## II. Related Work

Limited battery life has long been a challenging problem for handheld devices. Recently with the growing popularity of smart phones, which are equipped with various of sensing instruments, this challenge has become even more pronounced. Therefore, power modeling and management techniques on smart phones have received increasing attention.

Many power modeling approaches for mobile devices measure and estimate the power consumption for specific hardware components. This can be done by inserting sense resistors on the power-supply rails of the hardware components being measured [7], or using a power suspend driver to keep track of the power consumption of each hardware component, which is required to register with the power suspend driver in advance [8]. Another class of approaches focus on a specific type of power-consuming functions, account for power consumed by that particular function, and then perform corresponding optimization algorithms. These function types being studied include mobile networking technologies: 3G, GSM, and WiFi [9], radio access network [10], and global positioning system (GPS) [11]. In these approaches, the power consumption can be optimized by designing power-aware algorithms which properly schedule the execution of those activities [9], or scheduling operations of turning on and off of those functions [11]. Similarly, a variety of system-level, energy-efficient scheduling algorithms have been proposed, which incorporate deadlines [12], CPU frequencies [13], and temperatures [14].

Besides the approaches that measure and modele the power consumption of hardware components or some specific functions, a finer-grained approach for power modeling is based on program analysis, in which energy consumption is mapped to program structure. These approaches include PowerScope [15], eLens [16], and Instruments [17]. PowerScope combines hardware instrumentation with kernel software support to measure the current magnitude and perform statistical sampling of system activities. Elens provides a mechanism to profile energy consumption at varying levels of granularity including application, method, path, and line of source code. Instruments profile processor behavior and energy consumption of user applications on iOS devices. It supports both real-time monitoring of the system, as well as offline data collection and analysis.

Our approach presented in this paper is different from the above existing work. It is an application-based approach, which can be implemented as a plug-in for any mobile applications. It does not focus on any hardware component or any specific functions of the device. Instead, it accounts for and estimates power consumption of various activities of the application. In addition, our approach enables users to define optimization algorithms based on their own needs, which is different from the existing power optimization approaches.

## III. Power Modeling

In a mobile application, usually there are a limited number of user action types. Many of these action types are generic, and can be seen in a variety of applications, such as switching views, sending messages, and pulling information from servers. Therefore, it is possible to build a generic energy consumption model based on users' actions, and then refine it by profiling energy consumption of each action. Once an accurate energy consumption model is built, energy optimization techniques can be developed to minimize the energy consumption while maintaining users' experience. In this section, we present our energy model and an example optimization technique.

### *A. Modeling Energy Consumption*

As the first step toward optimizing power consumption, we estimate the energy consumption for a mobile application in a per-time-interval basis. That is to say, in each time interval, we calculate the energy consumption in the next time interval by adding up energy that is consumed by each action. We are particularly interested in two salient characteristics that typically dominate a mobile application's energy consumption: the device's backlight level $x \in [0,1]$, and a set of energy-intensive actions $A = (a_1, ..., a_k)$ that are performed during use, such as typing, exchanging information with a server, updating GPS location, *etc.* With respect to $A$, we further define a power function vector $P = (p_1, ..., p_k)$, where $p_i : \mathbb{R} \to \mathbb{R}$ maps a given backlight level $x \in [0,1]$ to the power consumed by action type $a_i \in A$. Similarly, an action duration vector $L = (l_1, ..., l_k)$ can be defined, where $l_i$ is the time it takes to perform action $a_i \in A$. For the time interval, which is the basis of estimating energy consumption, we assume a fixed-time scale $t_1, \dots, t_n$, where $t_i$ is a time point, and $\tau = t_j - t_{j-1}$ denotes the time interval.

We define $E_\tau$ – the estimated energy consumption of a mobile application within time interval $\tau$ – as the combination of the energy required to complete all actions and the static energy leakage during the idle time within $\tau$:

$$E_\tau = E_{actions} + E_{idle} \tag{1}$$

Given a backlight level $x$, power vector $P = (p_1, ..., p_k)$ and action duration vector $L = (l_1, ..., l_k)$, we expand our model as follows,

$$E_\tau = \sum_{i=1}^{k} (p_i(x) \cdot n_i \cdot l_i) + p_{idle}(x) \left[ \tau - \sum_{i=1}^{k} (n_i \cdot l_i) \right] \tag{2}$$

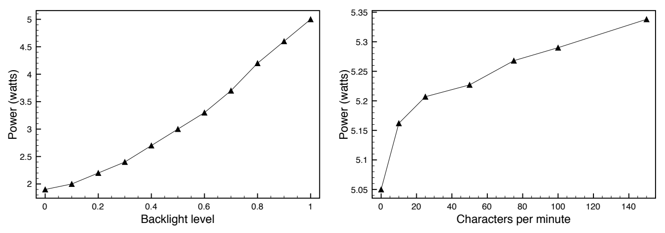
where $n_i$ denotes the number of times that action $a_i$ is performed within time interval $\tau$, and $p_{idle}(x)$ is the power consumed by the application given backlight level $x$ when no actions are being performed. This energy estimation model is generic, however, several terms in the model, namely $p_{idle}(x)$, $p_i(x)$, and $l_i$, are either device-dependent or application-dependent. They can be obtained statistically through profiling, which is described in greater detail in sections III-B and IV-B.

## B. Device Power Profiling

As shown in our energy consumption model (equation 2), we model the idle power consumption $p_{idle}(x)$ as a function of the backlight level $x$. This function is device-dependant, and we use an iPad2 as an example to illustrate how to derive this function. We set the backlight level to be different values ranging from 0.0 (complete darkness) to 1.0 (full brightness), let the device idle for 500 seconds, and measured the power consumption using a *wattsup* meter [18]. The results are shown in Figure 1(a). Then we performed a regression on the plot, and derived Equation 3, where $P$ is the power consumption and $x$ denotes the backlight level from 0.0 to 1.0:

$$p_{idle}(x) = -0.9518x^3 + 3.3042x^2 + 0.77x + 1.9021 \tag{3}$$

Equation 3 can then be used in the model to represent idle power for the specific device iPad2.



(a) Power vs. Backlight Level (b) Power vs. Typing Rate

**Fig. 1: Power Profiling for Idle and Typing Status**

A generic type of action on mobile devices is *typing*. Many other actions are always accompanied with typing. The energy consumed by typing depends on the number of characters that are typed and the time interval during which the typing is performed. To better understand how typing rate affects energy consumption, we characterize the typing operation. To this end, we first fix the experiment time to be 1 minute, and then for each experiment, we type a different number of characters with the keyboard. Finally, we calculate the average power usage in this 1-minute window as the power level for a specific typing rate. Figure 1(b) shows the relationship between power consumption and typing rate. The regression function is formulated in Equation 4:

$$p_{typing}(\alpha) = 0.1369log(\alpha) + 5.0536 \tag{4}$$

where $\alpha$ denotes the typing rate. With the regression function, the energy model can be customized from person-to-person.

For other application-specific user actions, application developers can perform power consumption profiling statistically, and derive corresponding $p_i(x)$ for their applications. This is described in greater detail in our case study, in Section IV-B.

## C. Optimization

Since the energy consumption model has been built, we can then use it to implement optimization techniques for different purposes. For instance, suppose we want to maximize the *user experience*, which may increase energy costs, but meanwhile save enough battery power to meet the desired duration of the application. For illustration purpose, we assume the user experiment is proportional to the backlight level, i.e., the higher the backlight level, the better the user experience is. For this simple scenario, we can formulate the optimization problem as follows:

$$\textbf{maximize} \quad x \tag{5}$$

$$\textbf{subject to}$$

$$\sum_{\tau=(t_{now}, t_{now}+w)}^{(t_{last}, t_{finish})} E_\tau \leq R_{t_{now}} \tag{6}$$

$$0 \leq x \leq 1 \tag{7}$$

Specifically, the optimization goal is to maximizing the backlight level – so that the user's experience is maximized – under the constraint that the device's battery can last until a predefined time point ($t_{finish}$). We divide the time period from the current time ($t_{now}$) to $t_{finish}$ into a number of time intervals with size $w$, [1] and use the energy consumption model (Equation 2) to estimate the required energy for each of the time interval. Equation 6 describes the constraint that the estimated energy cost for the next time step must not exceed remaining energy in the battery, $R_{t_{now}}$. At the end of each time interval, the backlight level can be adjusted on the fly, in order to meet the optimization goal.

Note that the above optimization problem is merely an example for illustration purposes. More complicated optimization techniques can be developed given the proposed energy consumption estimation model.

[1]The size of the last time interval may be less than $w$.

## IV. Case Study: A Mobile App for Field Scientists

To illustrate how our energy estimation model can be incorporated in a specific application, we developed a mobile app for coordinating field study trips. This application allows users to (1) design a field outing, (2) share logistical plan with all participants in real-time, (3) share the geographic location and actions performed by all users, and (4) communicate in real-time via instant messages. All user actions, such as completing a recording or sampling, are logged by time and location. We chose this application to illustrate our energy model because it is a representative mobile application which involves several common user actions, such as sending/receiving messages, updating GPS locations, and switching views.

### A. System Design

The design of our scientific field study application is shown in Figure 2. The system is composed of three parts: a back-end server, a front-end user interface, and a power management module. The design adopts a client-server architecture pattern, which is widely used in many mobile applications.

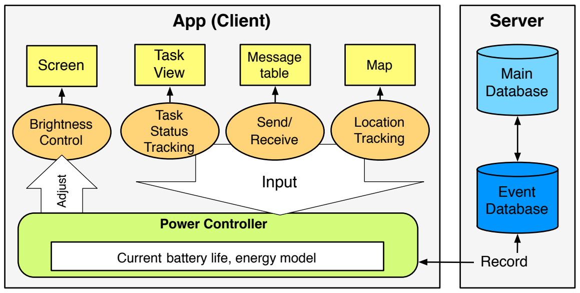


**Fig. 2: System Overview**

As shown in Figure 2, the back-end server maintains a database, which keeps track of users' personal information, locations, events, *etc.* The front-end mobile application is composed of four components: a *location tracking* component which is responsible for loading the map, and tracking every user's location; a *message handling component* which supports user communication through instant messages; a *task status tracking* component which is responsible for communicating with the back-end server to update task status; and a *record* component which enables users to playback the historical events.

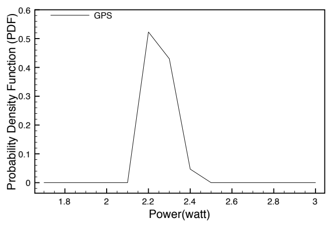
### B. Application Power Profiling

Using the field study mobile application, we show how the power consumption of application-specific user actions can be derived, i.e., $p_i(x)$ in the energy model (Equation 2).

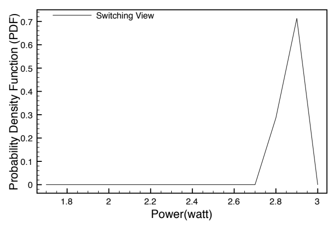
In our application, there are six types of basic actions:

- Updating and requesting location information through GPS and network
- Switching between different views
- Sending messages
- Fetching messages
- Fetching new tasks
- Submitting finished tasks

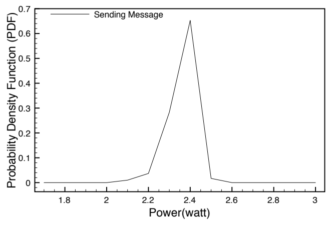
For each action, we start the application and continuously run the single action for a 500-second period, and record the power usage. We then plot the probability density function (PDF) based on the experimental data for each action, as shown in Figure 3.



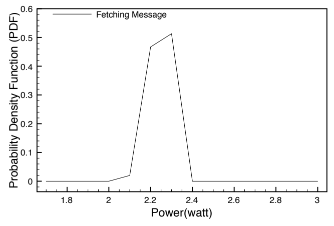
(a) GPS Request



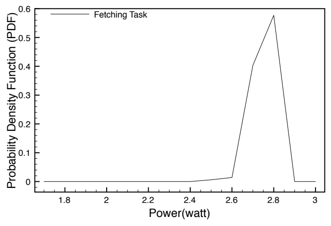
(b) Switching View



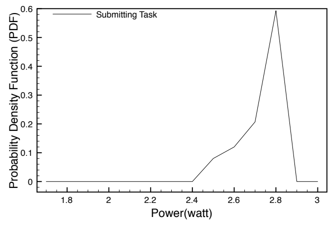
(c) Sending Message



(d) Fetching Message



(e) Fetching Task



(f) Submitting Task

**Fig. 3: Probability Density Functions (PDF) for Power Consumption of User Actions (Backlight=0.0)**

We choose the peak value from the probability density function graph, and use it in the model as the power consumption of the corresponding action. Note that the experiments are carried out under the backlight level of 0.0, therefore, these values can be used in our energy consumption model (equation 2) as $p_i(0)$. Assuming at a certain backlight level, the extra power consumed by an action on top of the idle power at the same backlight level is a constant, we can derive $p_i(x)$:

$$p_i(x) = p_{idle}(x) + (p_i(0) - p_{idle}(0)) \tag{8}$$

Equation 8 can be used in our energy consumption model, representing power consumption of a specific action.

Note that the energy model also requires an action duration vector $L$, for which we simply calculate the average duration for each action from our experiments. The two vectors, $p_i(x)$ and $L$, can be plugged in our energy model for estimating the power consumption of application-specific user actions.

## V. Evaluation

A number of experiments have been carried out to evaluate the accuracy of our energy model, as well as the optimization tehcniques.

### A. *Experimental Setup*

All the experiments were carried out on iPad2s, with 1.0 GHz Apple A5 CPU, 512MB RAM, a total battery capacity of 6583 mAh, and operating system version iOS 6.1.2.

To effectively evaluate the energy model proposed, we have designed several scenarios that represent typical usage patterns of the application. The first scenario, *Regular-user experiment*, represents a regular field study. It involves all the operations that a user would perform in a field trip, including sending messages, request GPS, and submitting a finished task. The second scenario, *Message-heavy experiment*, represents the "message-heavy" mode, where user sends messages more frequently while performing less other operations. The third scenario, *Note-taking experiment*, represents the case where a user focuses on taking notes of the samples while performing less interactions with the server. For each scenario, we design experiments with activities which take 15 minutes. All experiments are conducted in the School of Engineering and Computer Science building at Washington State University Vancouver campus. The communications between iPads and the server are through a WiFi network.

We use the *Watts-Up* meter for power measurement [18]. The meter measures voltage and current a thousand of times per second, so it has a quick response time which enables users to "see the surge" of power when appliances are first turned on. The peak value display captures this surge so it is displayed if it happens too fast to see live. This feature meets our requirements very well, as we want to characterize the user actions in our app, which always happen quickly and randomly. Also the accuracy of Watts-Up is within 1.5%, which outperforms many tools we have tried. The meter can record the data as fast as once per second so we can see the load profile as it changes over the course of the entire experiment. In addition, there is a USB connector on the side of the meter. With a USB cable and attached software we can download the power usage data directly to a PC.

To measure the power usage of an iPad, we first charge the iPad battery to full. Then connect the iPad to watts-up and make sure the power reading is stable at 0.5 watts when the iPad is in sleep mode. When we turn the iPad on and start our application, we can assume that the power reading from Watts-Up reflects the real-time power usage of the application. The data is recorded once per second.

Once the application is launched and the user has logged in, several threads will begin running in the background:

- Thread 1: The user reports current location to the server and pulls other team members' locations every 5 seconds.
- Thread 2: The user pulls new messages from server every 5 seconds.
- Thread 3: The user pulls new tasks from server every 5 seconds.

These threads ensure the users always obtain the up-to-date information in a field study. Note that the polling frequency can be adjusted to balance between performance and energy consumption, which is described in the following sections.

### B. *Evaluating Energy Model*

Experiments have been carried out to evaluate the accuracy of our energy model under the three different scenarios. For each scenario, we executed the same experiment with three backlight levels: 0, 0.5, and 1. For each experiment, we logged the actions performed by the user and generated the estimated power consumption by our model at a one-second granularity. Due to the limited space, we only present the experimental results from the cases with a backlight level of 0.0, as shown in Figures 4 to 6.

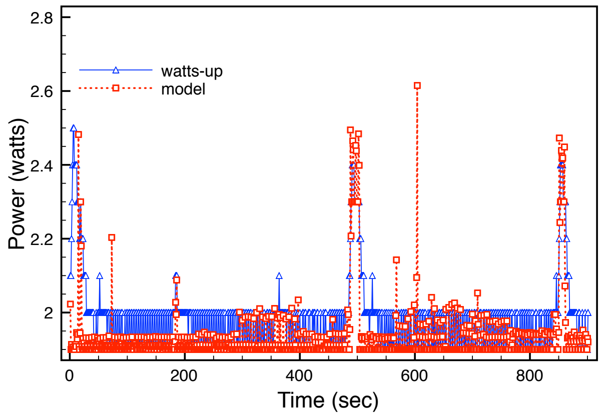


**Fig. 4: Model Accuracy Evaluation (Regular User Scenario, Backlight = 0.0)**

Figure 4 shows the real-time power usage in the regular-user experiment. In this scenario, the power bounces between 1.9 watts and 2.0 watts when backlight level is at 0.0. There are power peaks taking place at some time points during the experiment, especially at the beginning and the end of the experiments. It is because these are the action-heavy time intervals. Some actions, such as logging-in, sending instant messages, or submitting finished tasks usually involve a combination of several basic actions. It can be seen that our model is accurately capturing the energy consumption of these actions. However, the model sometimes gives a relatively higher prediction than the actual measurement. We believe this is due to over-estimation in deriving action duration $l_i$. As we clock the networking actions' time by measuring the time from sending the request to receiving the response from the server, we include transmission time into the action execution time. In a real scenario, unexpected network delay may affect this measurement, therefore, the model gives a higher energy consumption prediction.

We also calculated the cumulative energy consumption, which is the integral of the previous power-time function, *i.e.,* summing up the total energy consumed in an experiment. We compare the actual measurements from the *Watts-Up* meters and the energy consumption estimated by our model. The error of our model is approximately 2%.

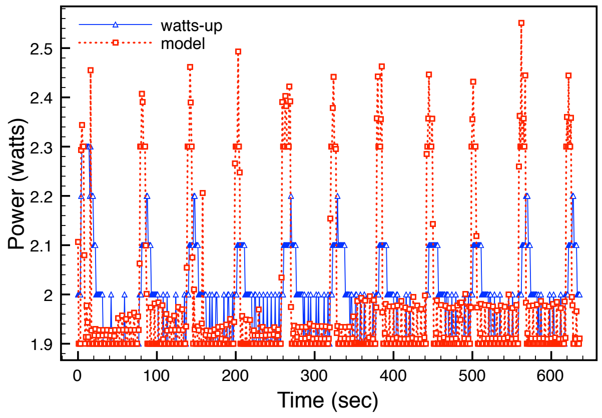


**Fig. 5: Model Accuracy Evaluation (Message-Heavy Scenario, Backlight = 0.0)**

In the message-heavy experiment, we perform a "sending message" action every 60 seconds without performing other types of actions. Figure 5 shows the real-time power usage and overall energy consumption results. The real-time power figure well captures the login action along with all the sending message actions. It over-estimates the power usage for some sending actions due to the same reasons mentioned above. However, experimental results on cumulative power show that the error of our model is less than 10 joules, *i.e.,* 0.79% of the total power consumption, indicating that it has well characterized the message sending action.

In the note-taking experiment, we perform the note-taking action every 180 seconds. This experiment involves many typing actions, but less interaction with the server. Because it is difficult to accurately capture a user's typing rate, we use a fixed value in our model. Based on previous studies [19], the average typing speed on iPad is approximately 45 words per minute. In addition, statistics show that the average number of letters in an English word is 4.5. Figure 6 shows the results in both real-time and cumulative energy consumption. Although we use a static typing rate assumption, the model still predicts the real-time power usage very well. The total energy consumption figure shows that the error of our model is approximately 1.5%.

The experimental results from all three scenarios illustrate that our energy model can accrately estimate energy consumption of a mobile application in real-time, despite of different usage patterns.

## C. Evaluating Optimization Techniques

For optimization, we have implemented the simple optimization algorithm described in Section III-C, and investigated how the application behaves when the battery is running low. Specifically, we want to know whether the battery could last long enough to meet the desired duration of a field trip and how the backlight level will be adjusted. Before describing

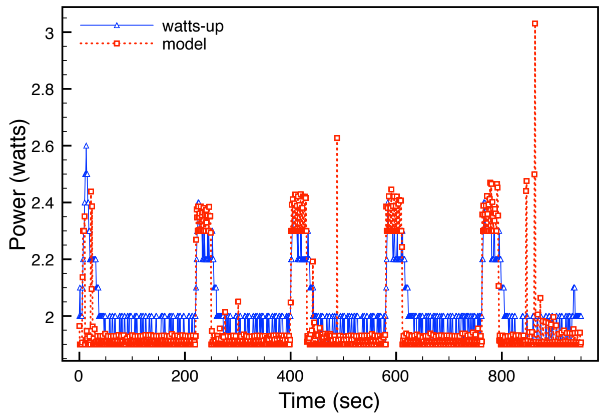


**Fig. 6: Model Accuracy Evaluation (Note Taking Scenario, Backlight = 0.0)**

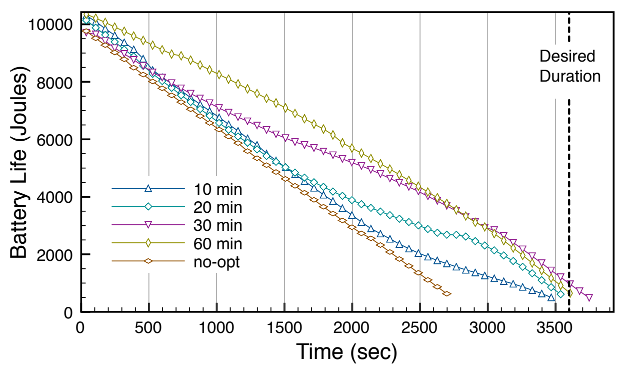
the experiments, we define the parameters we used in our optimization, as shown in Table I.

| Parameters | Definition |
|---|---|
| $t_{finish}$ | Desired experiment duration, or the finish time of the experiment. This value is pre-defined before an experiment. |
| $w$ | The sliding window size, or the estimation length. It defines how far into the future the model will be predicting. |
| $\tau$ | Optimization interval. This value defines how frequently the optimization algorithm is invoked. |
| Action rate | For the time being, all the fetching actions are performed at a frequency of every 5 seconds. |
| $x$ | Backlight level. This value will be changed from time to time based on the output of the algorithm. |

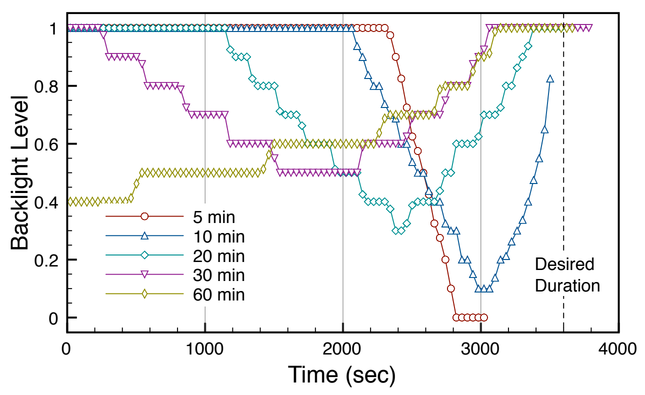
**TABLE I: Parameters Used in Optimization**

In our experiments, we used five different sliding window sizes $w$, which are 5 minutes, 10 minutes, 20 minutes, 30 minutes and 60 minutes. We set the desired experiment duration $t_{finish}$ to be 1 hour. In each experiment, we drain the battery of the iPad to 12%, which can be approximately converted to 10000 joules remaining. In all the experiments, the optimization interval $\tau$ is set to be 5 seconds. We in total conducted four sets of experiments, including three scenarios described earlier, and an idle mode, where the user does not perform any active actions.

As shown in Figure 7, optimizations with a window size $w \geq 20$ minutes meet the desired experiment duration. In general, the battery life increases with the window size, because with a larger window size, the power module need to ensure the power supply in a longer time interval, which leads to a more conservative output of backlight level. Even with a window size of 5 minutes, the battery life is extended by 15% compared to the one without optimization. Figure 7 plots the backlight trend as the experiment goes on. In general, the backlight starts at a high level (1.0), and drops at some point and then rises in the end. With a larger window size, the backlight level tends to drop at an earlier time. This again can be explained by the fact that a larger window size leads to a



(a) Remaining Energy vs. Time



(b) Backlight Level vs. Time

**Fig. 7: Optimization Algorithm Performance (Regular-User Mode)**

more conservative power supply. In an extreme case where we set the window size to be 60 minutes, which is the same as the desired experiment duration, the backlight level starts at 0.4 and keeps adjusting to a higher level as experiment continues.

In order to learn the extra energy consumption when running our algorithm, we conduct a series of experiments with different value of $\tau$. That is, we invoke the optimization algorithm at different rate. All the experiments are conducted under idle mode, with back light set to be level 1.0. Each experiment lasts 15 minutes.

Figure 8 shows that when the optimization is run every second, the energy consumption will increase by a large amount (5858 joules in total). However, if we run the optimization with an interval more than 5 seconds, the total energy consumption remains unchanged (around 4000 joules). So we can make the assumption that with a time interval of 5 seconds or more, there are no extra energy consumption brought by running the optimization itself.

### D. Customized Optimization

The optimization results presented herein is based on the simple algorithm, described in Section III-C, which adjusts backlight level at run-time to maximize user's experience while attempting to conserve energy until a predefined time point. It is worth noting that this optimization is merely for

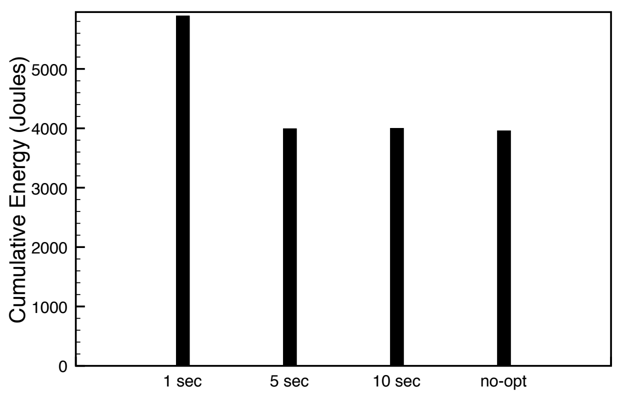


**Fig. 8: Total Energy Consumption When Executing Optimization at Different Rate**

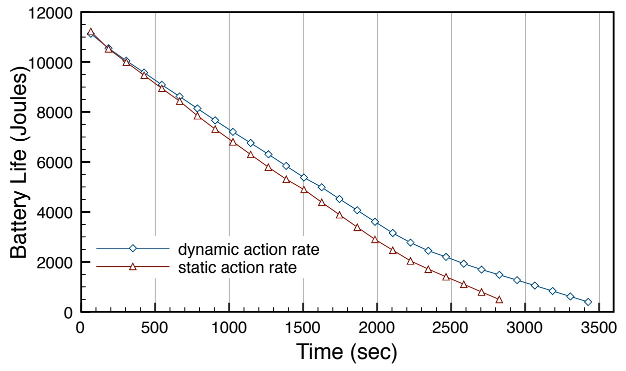
illustration purposes, and the users can easily replace this optimization algorithm with their own customized version. This enables users to optimize energy consumption based on their own needs, harnessing the accurate estimation on energy consumption provided by our model.

To illustrate this flexibility, we have implemented a different optimization algorithm which adjusts two parameters, backlight level, and the action rate. Recall that the application periodically pulls information from the server and GPS, and the frequency of the update actions can be used in the optimization algorithm as a tunable parameter, similar to the backlight level. Specifically, at the end of each time interval, the power management module estimates the energy consumption and checks if the remaining energy is enough to carry out the whole experiment. If not, it decreases both the backlight level and the action frequency, one at a time, in order to optimize the energy consumption so that the battery can last till the end of the experiment.

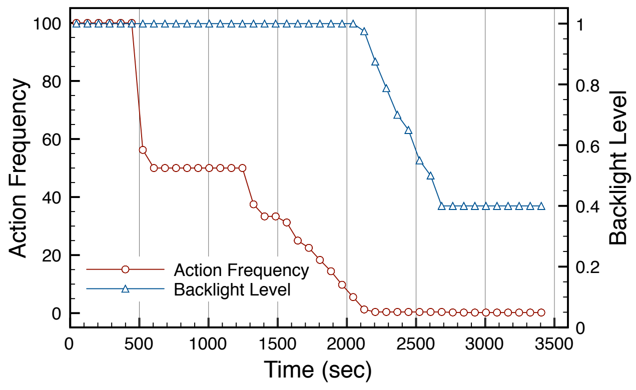
The results of this customized optimization are shown in Figure 9. Figure 9(a) compares the original optimization algorithm, which uses a static action rate, and the customized algorithm which adjusts both the backlight level and the action frequency. The experimental results show that with dynamic action rate, the customized optimization algorithm conserves more energy than the original one. Figure 9(b) shows the changes of the two parameters at run-time.

## VI. Conclusion

In this paper, we proposed a light-weight, on-line power monitoring and control mechanism for mobile applications. Specifically, we developed a power consumption model based on profiling of various types of user actions in the application. The model enables users to predict power consumption on the fly at run-time, and develop effective power optimization techniques for various purposes. We then used a case study, a mobile application for coordinating field study trips, to illustrate the effectiveness and flexibility of our approach. Experimental results show that our energy model can accu-



(a) Remaining Energy vs. Time



(b) Backlight Level & Action Frequency vs. Time

**Fig. 9: Customized Optimization Algorithm Performance**

rately predict the power consumption of the application under different usage patterns. In addition, we also developed two simple optimization mechanisms which adjust the backlight level and action frequency during the course of execution, so that the battery can last until the end of the field study. This approach opens up opportunities for accommodating various user-defined optimizations.

Work is ongoing in a number of directions. We will explore the possibilities of improving our energy consumption model by using machine learning techniques in profiling the power consumption of different user actions, as well as finding the durations for these actions. We will also investigate the tradeoff between the performance of our power optimization and the parameters we use in the optimization, including the sliding window ($w$) and the optimization interval ($\tau$). In addition, we will use different types of applications to evaluate this approach.

## Acknowledgments

We would like to thank *Google Summer of Code 2013* for supporting the open-source development of this project.

## References

[1] "Mobile Growth Statistics," 2013, http://www.digitalbuzzblog.com/infographic-2013-mobile-growth-statistics/.

[2] "Mobile Phone Sales Will Hit 1.86 Billion in 2013 as Strong Smartphone Growth Continues," 2013, http://www.ccsinsight.com/press/company-news/1655.

[3] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, 2011, pp. 1–6.

[4] K. Flautner, S. Reinhardt, and T. Mudge, "Automatic Performance Setting for Dynamic Voltage Scaling," *Wirel. Netw.*, vol. 8, no. 5, pp. 507–520, Sep. 2002.

[5] R. N. Mayo and P. Ranganathan, "Energy consumption in mobile devices: Why future systems need requirements: Aware energy scale-down," in *Proceedings of the 3rd International Conference on Power - Aware Computer Systems*, ser. PACS'03. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 26–40.

[6] B. Wang, "An Energy Efficient Mobile Application for Field Scientists," 2013, https://github.com/north212wangbo/field-study.

[7] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21.

[8] R. Murmuria, J. Medsger, A. Stavrou, and J. M. Voas, "Mobile Application and Device Power Usage Measurements," in *Proceedings of the 2012 IEEE Sixth International Conference on Software Security and Reliability*, ser. SERE '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 147–156.

[9] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "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*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 280–293.

[10] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling Resource Usage for Mobile Applications: A Cross-layer Approach," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 321–334.

[11] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, "EnTracked: Energy-efficient Robust Position Tracking for Mobile Devices," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 221–234.

[12] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, ser. FOCS '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 374–382.

[13] S. Irani, G. Singh, S. K. Shukla, and R. K. Gupta, "An Overview of the Competitive and Adversarial Approaches to Designing Dynamic Power Management Strategies," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 12, pp. 1349–1361, Dec. 2005.

[14] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed Scaling to Manage Energy and Temperature," *J. ACM*, vol. 54, no. 1, pp. 3:1–3:39, Mar. 2007.

[15] J. Flinn and M. Satyanarayanan, "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, ser. WMCSA '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 2–10.

[16] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating Mobile Application Energy Consumption Using Program Analysis," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 92–101.

[17] Apple instruments. [Online]. Available: https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40004652-CH1-SW1

[18] "Watts up power meter," https://www.wattsupmeters.com/secure/products.php?pn=0#.

[19] "Keyboard performance: ipad versus netbook," Nov 2010, http://usabilitynews.org/keyboard-performance-ipad-versus-netbook/.