DESIGN AND IMPLEMENTATION OF AN ENERGY-AWARE MOBILE APPLICATION FOR SCIENTIFIC FIELD STUDIES

By

BO WANG

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Engineering and Computer Science

May 2014

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of BO WANG find it satisfactory and recommend that it be accepted.

David Chiu, Ph.D., Chair

Xinghui Zhao, Ph.D.

Thanh Dang, Ph.D.

ACKNOWLEDGMENTS

I would like to begin by expressing my genuine gratitude to my advisor, Professor David Chiu, who introduced me to this program and this project. His passion and insightful guidance are felt throughout this work. I have greatly benefited from working and studying with him. Secondly I would like to thank my other advisor, Professor Xinghui Zhao. Her ideas about the project and all her support are very much appreciated. I would also like to thank the other member of my committee, Professor Thanh Dang. The way he teaches and conducts research has inspired me a lot.

I would like to thank Professor Wayne Cochran for registering all iPads and his valuable help on iOS programming. I would like to thank Professor Sarah Mocas for all her valuable guidance through my CS study and useful suggestions on the algorithm part of this project.

Finally I would like to thank all the CS professors and my colleagues. You have made my two-year study here an enjoyable experience.

DESIGN AND IMPLEMENTATION OF AN ENERGY-AWARE MOBILE APPLICATION FOR SCIENTIFIC FIELD STUDIES

Abstract

by Bo Wang, M.S.
Washington State University
May 2014

Chair: David Chiu

In this thesis, we have designed a mobile application that can simplify the management, data collection, and communication among team members during field studies. The application handles both real-time and offline field study management. It allows scientists 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, (4) enable real-time communication exchange, (5) log the trip events and collect statistics for future data analysis.

However, a key challenge is to ensure that the mobile devices would have enough power to carry out the experiment. To this end, we first worked with educators to define all functionalities and fit them into different stages of a field study. We then profile the energy consumption of our application on real mobile devices and propose a power model that estimates the power usage of the application according to user actions. Based on this model, we develop a power controller module, which dynamically adjusts power usage of the application. It maximizes the user experience but saves enough battery life to meet the desired duration of the field study. We show that our model can predict the energy consumption in different scenarios with an error less than 2% compared to real power measurements.

TABLE OF CONTENTS

List of Tables

List of Figures

Dedication

This thesis is dedicated to my parents.

Chapter 1

Introduction

Field research, or a field study, refers to the collection of data outside of a laboratory, library, or workplace setting [1]. It is widely applied across many different disciplines including anthropology, archaeology, biology, earth and atmospheric sciences, economics, public health, management, and sociology. This thesis focuses on enhancing scientific field studies by leveraging the advances of mobile technology. In addition, it proposes approaches to address one of the most critical challenges in mobile applications: energy awareness.

1.1 Challenges in Conducting Field Studies

Different field studies have different objectives. For example, a natural scientist who conducts a field study may want to observe the environment which she is interested in and collect samples for further analysis, while a social scientist may interview or interact with people to study their culture and behavior. Although the methods and approaches being taken in different field studies vary across disciplines, all field studies share some similarities in terms of the procedures they follow.

A typical field study usually involves three stages: pre-trip planning, real-time man-

agement, and post-trip analysis. Let us consider a scenario where a group of high school students are organized by their teachers to conduct a field study to investigate the local water quality. In the pre-trip planning, teachers usually give students some time in class to prepare themselves for the trip and assign each individual a set of tasks. In a traditional pre-trip class, students also receive some kind of data sheets, which they will fill out during the trip and hand in after the trip. During the trip, students might form 4 or 5 groups, and each group is responsible for a specific task. In this example, the tasks that students will be performing are to measure the temperature, pH, water clarity, dissolved oxygen, nitrates, phosphates and stream width, depth, and flow. After the field trip, teachers will collect data from their students and teach them scientific methods to analyze the collected data. Finally, the students will be able to analyze the data by themselves and finish their own report.

Field studies are the most practical way to learn and interact with the environment, and it benefits students in may ways. However, managing field studies is challenging and costly for scientists and educators. In fact, most of the institutions can only afford few trips per year [2], not to mention that outdoor environments often causes unexpected difficulties like communication breakdown. Therefore, it is important for us to maximize the efficiency and user experience of the field studies to ensure an ideal experience and accurate results.

However, the data quality depends on the field researchers who conduct the trip, their professional knowledge, level of involvement, and ability to record important discovery. In a traditional field study, the main source of data is from the notes taken by the researchers. Key words or phrases are written down, while the researchers perform the study. In some disciplines where field studies involves interacting with people, the way how notes are taken may even affect the results. Specifically, some researchers prefer to take notes with the presence of those being studied, while others try to avoid taking notes in front of them do that when they are alone. In the former case, the researcher may retrieve more accurate data, but will inevitably affect the participants while in the latter case the notes are less

accurate. This problem is less pronounced in a natural science field study because in such disciplines, there rarely are human participants involved, but still such note-taking activities can distract the researcher from what is happening in the immediate scene in which he or she is investigating.

It is also tedious for a researcher to record all the timestamps and locations of all the events. In our case, to manage a group of students, especially young students, is a more challenging task. On the one hand, the students need to focus on their tasks and not influenced by others. On the other hand, they also need to be able to communicate efficiently with other group members. In addition, the teachers need to keep track of the progress and status of each team and their data without interrupting the students too often. For the post-trip analysis, the teacher must extract the collected data manually from the data sheets, which is also a tedious task.

1.2 Mobile Application for Field Studies

To better address these challenges, we propose a mobile application that is designed specifically for the purpose of field study management. With the rapid development of mobile devices in recent years, scientific instrumentation, including sensors, gauges, and measuring devices have become increasingly portable, allowing groups of scientists to conduct *ad hoc* labs and experiments. Even in a traditional classroom environment, it has become increasingly popular that students and educators share data or information through portable devices. In the case of field study, we believe the advantages in information organizing and retrieval makes mobile technology a great enhancement to the field study.

During different stages of a field study, a faculty usually spends tremendous amount of time on planning such nuances as:

- Who will be participating a field study?

- What, and how many samples do we require?
- Which students should collect these samples?
- Where and when should these samples be collected?
- Can we track each other's geographic location and communicate over a simple visual interface?
- Is the study complete? Is there any details were missed?
- Can we automatically log the provenance of the field study?
- How to store and manage the collected data?

Based on these requirements, we design a mobile application for real-time field study management. Our tool would allow scientists 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) enable live communication exchange. All user actions, such as completing a recording or sampling, are logged by time and location. In this way, scientists can gain an insight on the study and even replay the actions offline. With the help of smart mobile devices and their rich set of utilities (*e.g.*, wireless networking, GPS, compass, etc.), our application can help alleviate the process of carrying out a field study.

There are two objectives of building this application. The short term goal is to improve the way science is done by simplifying management before, during, and after field experiments. The long term goal is to advance scientific knowledge via federation of field data and enable sharing, integration, and collaboration among scientists.

1.3 Energy Awareness

Energy consumption has become a critical factor in computer systems. Electricity costs impose a substantial strain on the budget of data and computing centers. In office environments, computers and monitors accounts for the highest energy consumption. Google engineers, maintaining thousands of servers, warned that if power consumption continues to grow, power costs can easily overtake hardware costs by a large margin [3]. This challenge is even more pronounced for mobile devices, because of their limited battery life. As mobile applications become ubiquitous today with a rapid growing rate [4], energy efficiency of applications is becoming an important topic of interest. Finding an optimal management of power usage on mobile devices is critical.

Despite improvements in low-power hardware design and battery life, there is now growing awareness that a strategically viable approach to energy management must include higher levels of the system [5]. However, recent studies show that most of the third party applications in the mobile application markets are not designed or engineered with power consumption as a priority [6]. Moreover, application developers are usually unaware of cellular specific characteristic that incur potentially complex interaction with the application behavior. How do the energy consumption characteristics of network activity on mobile phones compare with each other? How can we reduce the energy consumption while maintaining good performance? How can we estimate the battery life based on current usage pattern? To investigate these questions, we need first to generate a power consumption model. This is crucial for understanding, designing, and implementing better mobile application software.

In our application, a power management module will assist the user to better utilize the energy and extend the battery life. The power management can estimate the remaining battery life by observing the current usage pattern. Based on this, it can adaptively adjust its energy saving strategy. For the purpose of this study we focused only on our application

with certain usage patterns on tablet devices but we plan to evaluate this approach using other applications on different platforms in the future.

1.4 Contributions

We make the following contributions in this work: First, we present a prototype of mobile application that can be used in a scientific field study. We define all the functionalities and fit them into different stages of a field study. Second, we develop the actual application and deploy it on mobile devices. Third, we propose a power consumption model that estimates the power usage of our application on a tablet. Fourth, we profile how the devices consume power with different usage patterns. Fifth, we verify the power consumption models by performing simulated experiments on the devices. Finally, we develop a power management module that is embedded in the application which uses the power consumption model to estimate and control the power usage of the application.

1.5 Organization

The rest of this thesis is organized as follows. Related works are reviewed in Chapter 2. Chapter 3 describes the overall architecture of the application and implementation of all the functionalities. In Chapter 4, we describes our power consumption model in detail and perform a series of characterization on the impact of different user actions on power usage. We also describe the algorithm applies in our power management module in this Chapter. The results of evaluating the power model are presented in Chapter 5. In Chapter 6 we discuss our results from the experiments and conclude the thesis.

Chapter 2

Related Works

In this chapter, we review related works covering the three major topics in the thesis, including Mobile Applications in Education, Mobile Device Power Modeling, and Mobile Device Power Management.

2.1 Mobile Applications in Education

With the aid of wireless communication technology, education can be more mobilized, portable, and individualized. As a result, the learning style has changed dramatically. To exploit the potentials of the new mobile technology, new educational models have been developed. Motiwalla proposes a mobile learning framework, which helps explore the extension of electronic learning into wireless/handheld computing devices [7]. This work investigates the effectiveness of wireless devices, such as PDAs and Smart phones for data services like Wireless Access Protocols (WAP), Short Message Service (SMS) and Wireless Markup Language (WML) in higher education.

Besides the learning framework, there are studies focusing on the improvement of mobile learning. Studies of Adult Learning Practices [8] show that learning activity is mobile

between locations, time slots, and topic areas. Moreover, the learning process can be divided into three hierarchical operation levels: discrete acts, learning episodes, and learning projects. Based on this hierarchical structure, Giasemi, *et al.* develop a personal mobile knowledge and learning organisation system [8], which seeks to support a person's everyday learning over a lifetime. In a more specific work, Chen, *et al.* propose a mobile learning system for scaffolding students to learn about bird-watching [9]. The Bird-Watching Learning system is designed using a wireless mobile *ad hoc* network. Each learner holds a handheld mobile device with a wireless network card. A purpose of their work is to explore the possible roles and scaffolding aids that the mobile learning device offers for bird-watching activities and to investigate whether students benefit from such learning experience.

In terms of supporting more generalized scientific field studies, Robert, *et al.* investigate the possibility of applying computers and mobile devices in an outdoor field study [10]. They set the background for the mobile computing project for outdoor experiment and described two prototype field applications that are developed for mobile learning environments. One of the applications they have developed is simply for data collection and information browsing. It provides users with an interface to search field information. Important concepts in plant genetics and evolution are demonstrated via access to digital libraries. During a field study, students are provided with flash cards which store the data collected in their experiment. Their flash cards are then given to the instructor who loads data from each team onto a laptop computer for more sophisticated analysis and visualization.

In a more recent study, Kennedy carries out some case studies of mobile applications in scientific field studies [11]. Their studies focuses on two aspects of mobile learning (mLearning). The first is the use of smart phones and mobile devices by allowing students to collect data during a geography field study. The second aspect is the use of an innovative *Moodle Block* [11], an open source learning management system that enables students to upload and download files to a central server using mobile devices. The study was undertaken by a group

of college students who use notebook computers and smart phones to carry out a range of activities during their field research. Once they return at the camp site, the collected data was transferred to their notebook computers. After the trip, 20 students were interviewed individually to reveal the user experience on the mobile applications designed for the field study and the use of the system for sharing and processing the data. The interview results show that most of the students and teachers agreed that access to the mobile system and the ability to share and exchange files was a very positive support for their learning and subsequent processing the data.

2.2 Mobile Device Power Modeling

Effective energy management requires a good understanding of where and how the energy is used: how much energy is consumed by various parts of the system and under what circumstances. As the first step towards effective energy management, power measurement and modeling have received increasing interest, and significant amounts of work has been done in this area.

2.2.1 Power Measurement

Rahul, *et al.* propose a general methodology for measuring power usage on smart phones [12]. The idea of their system is to account for the power usage of all of the primary hardware subsystems on the phone. In this work, they use the Android operating system, which requires that every device driver register a *WakeLock* with a power suspend driver, so that the operating system can calculate the amount of time when a particular driver is in an active state. The system is then trained to learn the contribution of power that each subsystem makes based on the power usage data collected in the past. Once the system is trained, it is able to predict the real-time power usage by analyzing the per-subsystem time sharing data

pulled from the operating system's power management module.

In another work, Aaron, *et al.* present a methodology for measuring breakdown of power consumption by the smart phone's main hardware components [13]. They run their measurements in a number of realistic usage scenarios such as audio playback, video play back, text messaging, etc. They set up their measurement system by inserting sense resistors on the power-supply rails of the relevant components and measure the power consumption directly using external hardware instruments. The limitation with direct measurement is that it usually requires the device to be taken apart, therefore the circuit schematics of the device need to be available.

2.2.2 Activity-based Approaches

Some works focus on power consumption of network applications [14] [15]. In [14], the authors characterize the power usage of three mobile networking technologies: 3G, GSM, and WiFi. They also develop a protocol that optimizes power usage of web applications using some scheduling algorithm. On the other hand, Feng Qian, *et al.* focus on the interactions between applications and the radio access network [15]. They build a tool which analyzes the power consumption of various networking layers and their cross-layer interactions.

Another important feature of modern mobile devices is that they can locate themselves. However, global positioning system (GPS) tracking may have major impact on the battery life of the device. Kjargarrd, *et al.* propose a system that can schedule the position updates to minimize energy consumption and optimize robustness [16]. They first profile power consumption of a Nokia N95 cell phone and proposed a device model that can account for its real power consumption. For the power management module, they proposed an algorithm that minimizes power consumption while maintaining performance by generating and enforcing an optimal schedule for turning on and off features of the mobile device.

2.2.3 Program Analysis

Some efforts focus on program analysis tools and map energy consumption to program structure. This approach is similar to CPU profilers that help identify code components that waste processor cycles. *PowerScope* [17] is an early attempt in this direction. It combines hardware instrumentation with kernel software support to to measure current level and perform statistical sampling of system activity. The tool involves a system monitor which records the program instructions, and an energy monitor which records correlated current levels as input. Then an energy analyzer runs an off-line analysis to generate the energy profile. This tool is claimed to be fine-grained at the procedure level.

Shuai Hao, *et al.* propose *eLens* [18], which can profile energy consumption at varying levels of granularity including application, method, path, and line of source code. *eLens* can be integrated into development environments such as the Eclipse IDE. The functionality of *eLens* is slightly different from *PowerScope*. Two main components of *eLens* include a workload generator and analyzer. Users could input a workload, or usage pattern of a program, which is translated into sets of paths by the workload generator. The analyzer uses the paths and system profiles which are pre-characterized with the platform to estimate energy consumption of the program. The output is the source code annotated with paths and energy estimation.

Similarly, the Apple development tool *Instruments* [19] can 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. It helps developers test their applications and gain a deeper understanding on how their applications work.

2.3 Power Management

Mayo, *et al.* presented a thorough introduction on why mobile devices require power management and propose a model for energy scale-down [20]. They introduce the terms *general-purpose* device and *special-purpose* device. They argue that one approach to design scale-down is to allow a general-purpose device to use both hardware mechanisms and software policies to adapt energy use to user's requirements for the purpose of achieving low energy use of a *special-purpose* device. They also propose scale-down optimizations in the context of the display, wireless, and CPU components of the system. In practice, there are multiple approaches to reduce energy consumption of computing devices. Krisztian, *et al.* describe a software approach to automatically control dynamic voltage frequency scaling (DVFS) [21]. They embedded their system into the Linux kernel so that it requires no modification of users' programs. The operating system can scale the frequency and voltage for different types of applications. This idea will be reviewed in Chapter 2.3.1.

A similar work which focuses on when to spin down the disk of a mobile computer to save energy is introduced by Helmbold, *et al.* [22]. They use a simple algorithm based on machine learning and achieve notable results in reducing power consumption. Hai Huang, *et al.* focus their study on reducing the power consumption of memory systems [23]. Based on the fact that for a data-centric application, more power is needed to sustain a memory system, they propose and implement a power-aware virtual memory system that significantly reduces power dissipated by the memory. Vivek Tiwari, *et al.* introduce energy-saving techniques from an perspective of software optimization [24]. These techniques include reordering instructions, code generation through pattern matching, and reduction of memory operands. They also discuss the effect of traditional compilation techniques on energy reduction. Jacob, *et al.* propose a power management system *Turducken* [25]. Their idea is to combine several optimized mobile platforms to form one integrated system. The system is composed

of a set of tiers, each with a set of capabilities and a power mode. The system executes tasks by waking the tier that has the capabilities to execute the task in the most energy efficient manner.

2.3.1 Algorithms

Recently, there have been growing interests in techniques to save energy [26]. The goal is to design energy-efficient algorithms that reduce energy consumption while maintaining the quality of service (QoS). For general computing devices, there are two major approaches for power management: *Power-Down Mechanisms* and *Dynamic Voltage Frequency Scaling (DVFS)*. *Power-Down Mechanisms* assume that a processor can operate in multiple states which cost different amount of energy, for instance, standby, suspend, sleep, and full-off states. The energy cost in transition from a higher-power state to lower-power state is negligible. On the contrary, a power-up operation consumes significant amount of energy. *DVFS* assumes that processor can run at various frequencies. Running in higher speeds results in higher performance, but also higher energy consumption. Both approaches apply to the field of mobile devices. There are several algorithms, especially scheduling algorithms, can be applied in a mobile power management module.

The *YDS* algorithm is one such energy-efficient scheduling algorithm first proposed by Yao, *et al.* [27]. The goal is to finish all the jobs by a deadline while minimizing the energy consumption. The algorithm applies a greedy strategy that schedules the most urgent jobs first, and distributes less urgent jobs over those time slots when CPU runs at a lower speed. Based on *YDS*, Bansal, *et al.* present a new online algorithm, *BKP* [28]. *BKP* can be viewed as approximating the optimal speeds of *YDS* in an online manner. They also proved that *BKP* achieves a competitive ratio of $2(\frac{\alpha}{\alpha-1}^{\alpha}e^{\alpha})$, with α being the exponential relation between CPU frequency and power usage. The authors also consider the temperature changing

rate during processing [28]. They assume that cooling follows Newton's Law, which indicates that the rate of cooling of a body is proportional to the difference in temperature between the body and the environment. Irani, *et al.* analyzed the combination of speed scaling and power-down mechanisms [29]. In this configuration, the processor may be transferred to a sleep state in which its energy consumption is 0. In the standard setting, we tend to use a lowest possible speed level as long as the job can be finished by its deadline. In the setting with sleep, it can be beneficial to speed up a job so as to generate idle times in which the processor can be transitioned to the sleep mode.

Another class of algorithmic approaches focus on developing scheduling algorithms for heterogeneous mobile computing system. A heterogeneous computing system differs from a homogeneous computing system as the jobs are assigned on different devices, and each job is independent from other jobs. The *HEFT* algorithm is one of the influential works on heterogeneous systems [30]. In this algorithm, the application is viewed as a directed acyclic graph (DAG), where nodes represent computation devices and edges represent communication. By assigning a weight to each node and edge, the algorithm prioritizes the nodes to be scheduled based on a rank function, which is a function of the weights assigned to the nodes and edges of the graph. In another paper, E. Ilavarasan, *et al.* present a variation of $HEFT$ algorithm which takes energy consumption into consideration [31]. The authors propose an interpolation function which balances between finish time and energy. By selecting a proper constant, the algorithm could minimize both executing time and energy.

Chapter 3

System Design

The design of our scientific field study application 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. The system overview is shown in Figure 3.1.

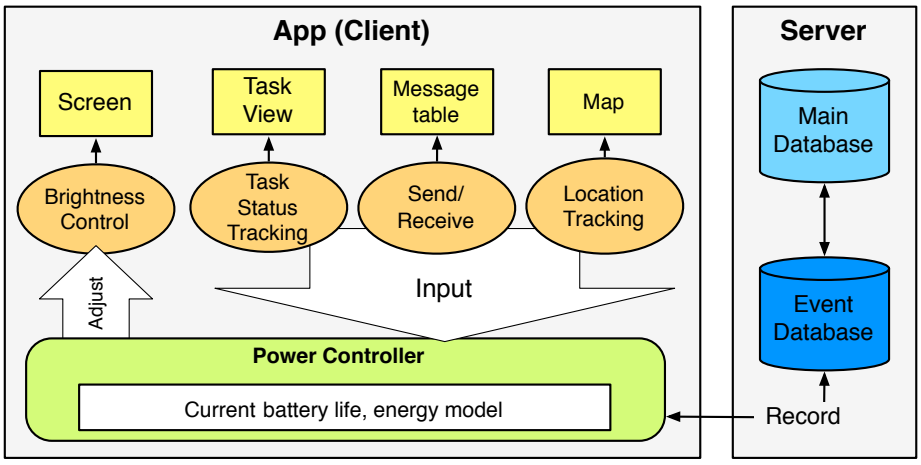


Figure 3.1: System Overview

3.1 Back-end Server

For the back-end server, a database is developed to store information about users and field events,*e.g.*, students' names and IDs, event names and locations, etc. The entity-relation diagram of the database is shown in Figure 3.2. The users interact with the database using PHP scripts. An important feature of the application is that it enforces access control based on the user's identity. For example, a student is only able to view her own field study history, analyze her own samples, and register/drop a trip she participates, while an organizer can view the information of all students and assign tasks for them in a trip.

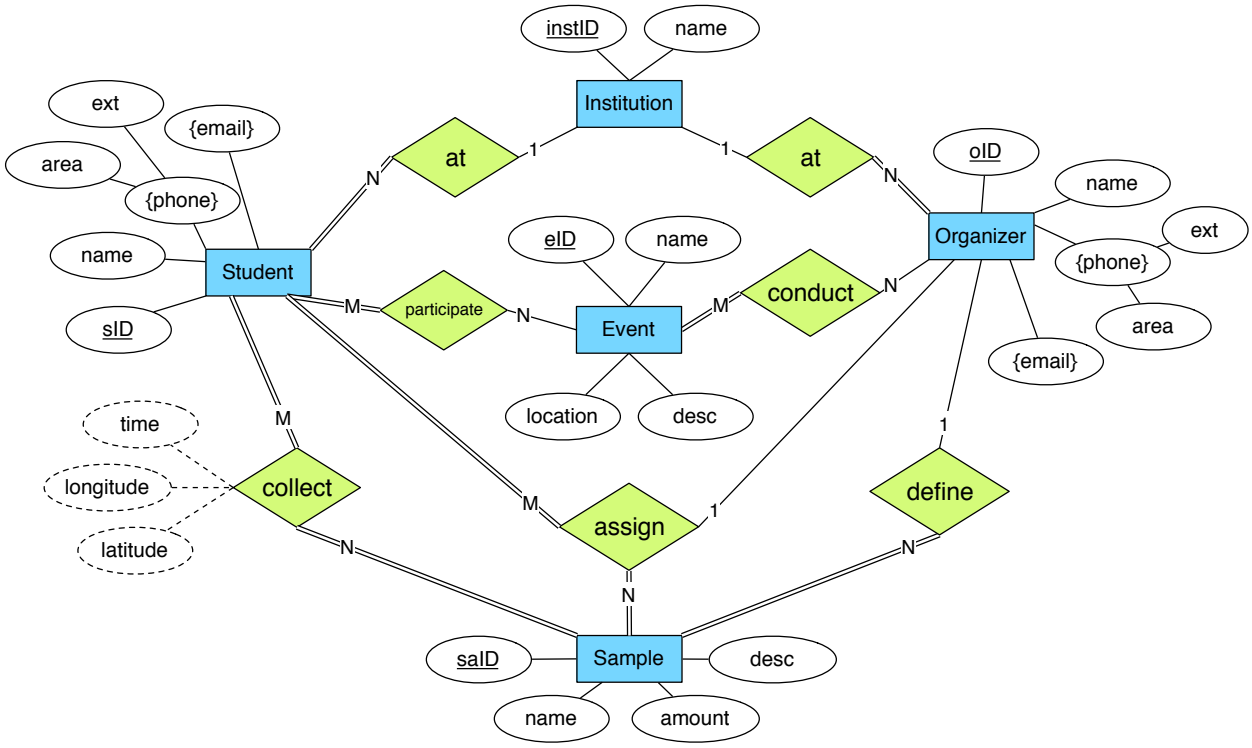


Figure 3.2: Entity-Relationship Diagram of Database

3.2 Front-end

The front-end mobile application is composed of four components, each of which is responsible for one of the four main functions:

- *Location Tracking Component*: This component is responsible for loading the map, and tracking every user's location. It frequently records the user's current location by reading and sending global positioning system (GPS) data to the server.

- *Message Handling Component*: Users can send text messages through this interface. The messages can be peer-to-peer, inside a group, or broadcast to all teammates. All messages are sent to the server. The server establishes a list for conversation within each group. The clients periodically fetch new messages from the server.

- *Task Status Tracking Component*: A user can submit a finished task or pull updated task information through this interface. The organizer version enables one to assign or modify tasks.

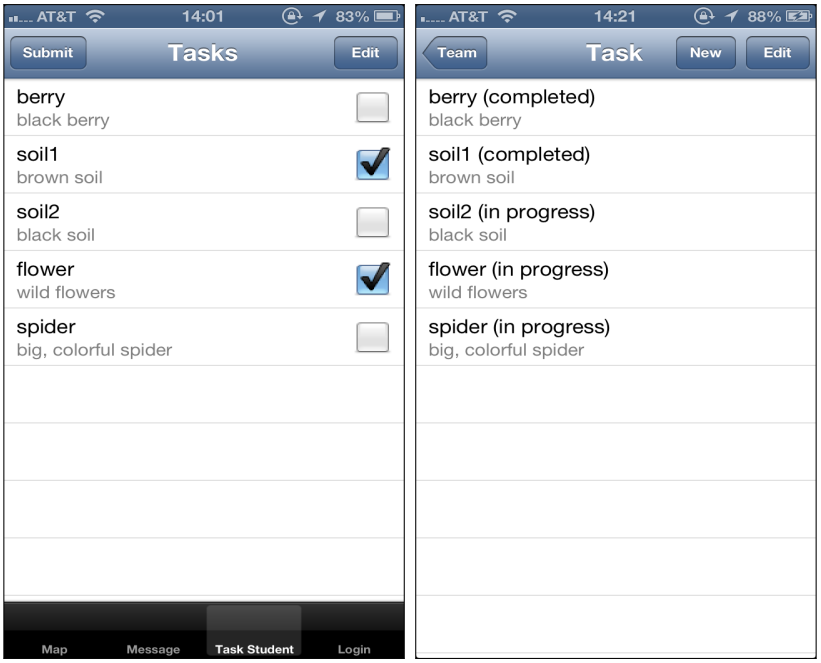
- *Record Component*: The record component fetches information from the server. When the user wants to see a playback, it receives new updates from the server. Once fetched, the data will be stored locally. [1]

Before a field study, the application loads all the information about the upcoming trip from the database. After the trip, all activities performed by each user will be locally stored in a log file.

Now we describe the usage and behavior of the application in detail. When a user logs into the server using a GPS-enabled mobile device, the server records the user's current coordinates and tracks the user's location in real-time. Once logged in, the Map View

[1] This feature is still under development, right now we store all the user actions locally

(a) Map View (b) Message View



(c) Student Task View (d) Organizer Task View

Figure 3.3: User Interface at Client Side

displays a customized map centered on the user's current location, shown in Figure 3.3(a). The user will also be able to view nearby team members' locations. A search bar on the top enables one to look for team members who are out of the view range. Furthermore, the user can use this view to raise questions or send messages, which are displayed on other team members Map View as a notification.

One can also send messages by switching to the Messages tab (Figure 3.3(b). The message handling component resembles the SMS function on a regular cell phone. The task status tracking component (Figure 3.3(c) and Figure 3.3(d)) lists each individual's tasks in a table. It confirms the task one has accomplished and reminds one of those that have not yet been accomplished. A detailed view of each task provides more information about the task such as location, time and description. When an action is performed, (*e.g.*, a soil sample is collected), the client application notifies the server as well as saves the action details locally. Each individual app polls for updated status of the tasks from the server.

3.3 Power Controller Module

The power controller module embedded in the application (Figure 3.1) is used for monitoring current battery life, current running components, and current power usage. It outputs the back-light level based on the algorithm that optimizes the power usage adaptively.

To inform the power controller, we first collect information about the power usage when running the app, and then analyze the power consumption patterns for different activities, develop an optimization algorithm for power conservation. We will elaborate our power management approach in Chapter 4.

Chapter 4

Energy Profiling and Modeling

In this chapter, we introduce the power model we have established for the application and the optimization algorithm used in the power management module. We first propose a power model that is based on users' actions and then characterizes the energy consumption of each action. Finally we develop an optimization algorithm based on the power model.

4.1 Modeling Energy Consumption

Our goal is to estimate the energy consumption in the next time interval by adding up energy that would be consumed by each action would take place. We are initially interested in two salient characteristics that define our 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 (*e.g.*, typing, fetching/sending a message, submitting a task, updating GPS location, etc.) during use. With respect to A, we further define a power function vector $P = (p_1, ..., p_k)$, where $p_i : \mathbb{R} \rightarrow \mathbb{R}$ maps a given backlight level $x \in [0, 1]$ to the power consumed by action type $a_i \in A$. Similarly, the action latency vector $L = (l_1, ..., l_k)$ can be defined, where l_i is the latency taken to perform action $a_i \in A$. For simplicity, we assume

a fixed-time scale $t_1, \ldots, t_n$, where t_i is a time point, and $\tau = t_j - t_{j-1}$ denotes the time interval. We can define E_τ to be the estimated energy consumed by our mobile application within time interval τ as the combination of the energy required to complete all actions and the static energy leak during the idle time within τ,

$$E_\tau = E_{actions} + E_{idle} \tag{4.1}$$

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

$$E_\tau = \sum_{i=1}^{k} (p_i(x) \cdot n_i \cdot l_i) + p_{idle}(x) \left[\tau - \sum_{i=1}^{k} (n_i \cdot l_i) \right] \tag{4.2}$$

where n_i denotes the number of times that action a_i is performed within time interval τ, and $p_{idle}(x)$ is the power consumed by the application given backlight level x when no actions are being performed.

4.2 Energy Consumption of Action Types

To inform the power model, we must generate the power vector P and the latency vector L, which are power usage of actions and execute time of actions, respectively. In total, there are six types of basic actions in our application:

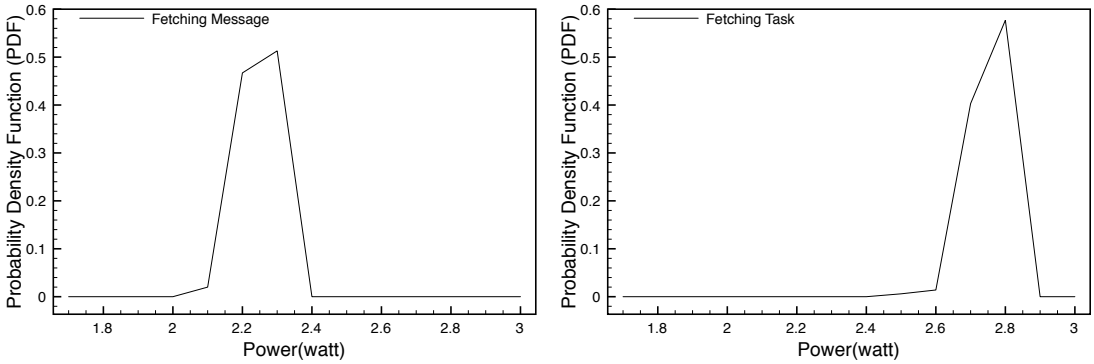
- Updating and requesting location information through GPS and network.
- Sending messages.
- Fetching messages.

- Submitting finished tasks.
- Fetching new tasks.
- Switching between different views.

At the same time, we need to know the power usage when no action is performed $p_{idle}(x)$. We consider this scenario as the baseline. For each scenario listed above, including the baseline, we ran a 500-second experiment, and record the power usage data measured by Watts-Up. Specifically, when we are measuring the power usage for updating and requesting location, we disable all other actions and perform solely the action we want to characterize. This is done by invoking that function repeatedly, *e.g.*, when we receive the response from the server, we immediately send another request to the server. We assume the power usage in this period of time reflects the average power needed to perform the action.

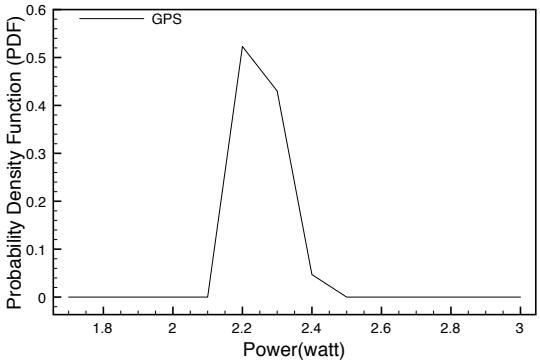
For each action, we plot its probability density function (PDF) based on the experimental data. To interpolate the influence of backlight, we carry out the characterization experiments under two different backlight levels. Finally, we choose the peak value from the probability density function graph to be the power usage of that action.

To form latency vector L, we also need to characterize the time it takes to perform each action. For all the actions that interact with the server, we define the time to perform the action to be the period from when the user starts sending a request to the time when she finishes receiving the response from the server. For switching views, we label the start time when switching view begins in origin view and the end time when the target view finishes loading. Again, we carry out the experiments repeatedly and use the average time in our model. Figure 4.1 gives the measured power usage of various actions.

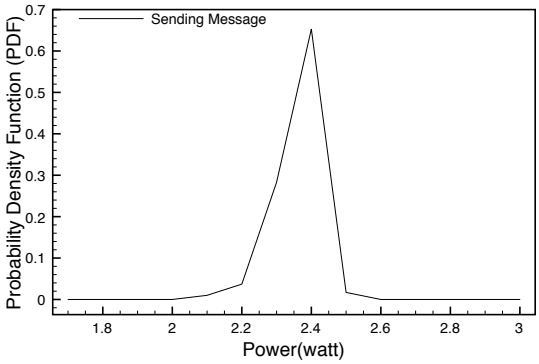


(a) Fetching Message

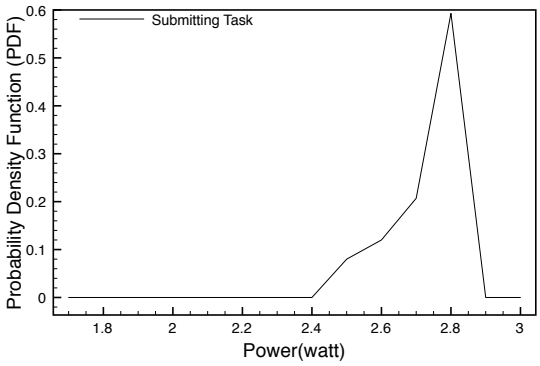
(b) Fetching Task



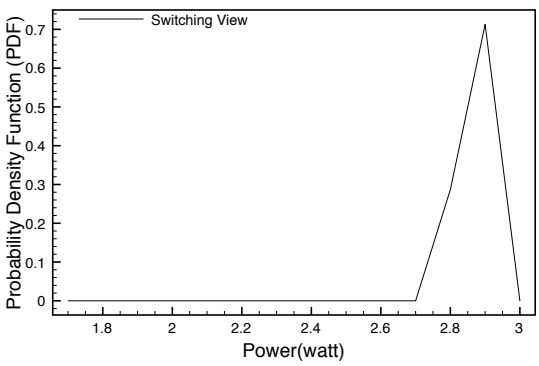
(c) GPS Request



(d) Sending Message



(e) Submitting Task



(f) Switching View

Figure 4.1: Probability Density Functions (PDF) for power consumption of App Actions (backlight=0.0)

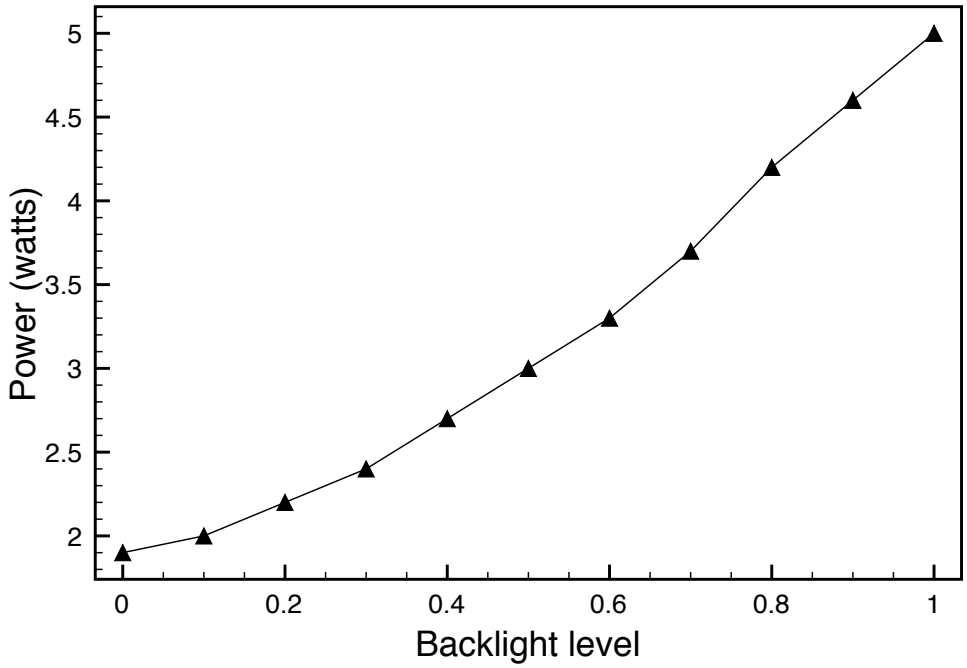


Figure 4.2: Power vs. Backlight level

4.3 Energy Consumption over Backlight Levels

Our measurements show that the screen brightness of an iPad2 is a large energy consumer. Changing the backlight level significantly affects the power of the device. Objective-C provides a programming interface that controls the backlight level from 0.0 to 1.0, with a 0.1 interval. In this section, we analyze the relationship between power and backlight level. To capture the power for all 11 levels from 0.0 (complete darkness) to 1.0 (full brightness), we test the power usage with the baseline configuration described above for each level. Then we perform a regression on the plot Figure 4.2. The result we obtained is listed in Equation 4.3, where P is power and x denotes the backlight level from 0.0 to 1.0:

$$P(x) = -0.9518x^3 + 3.3042x^2 + 0.77x + 1.9021 \tag{4.3}$$

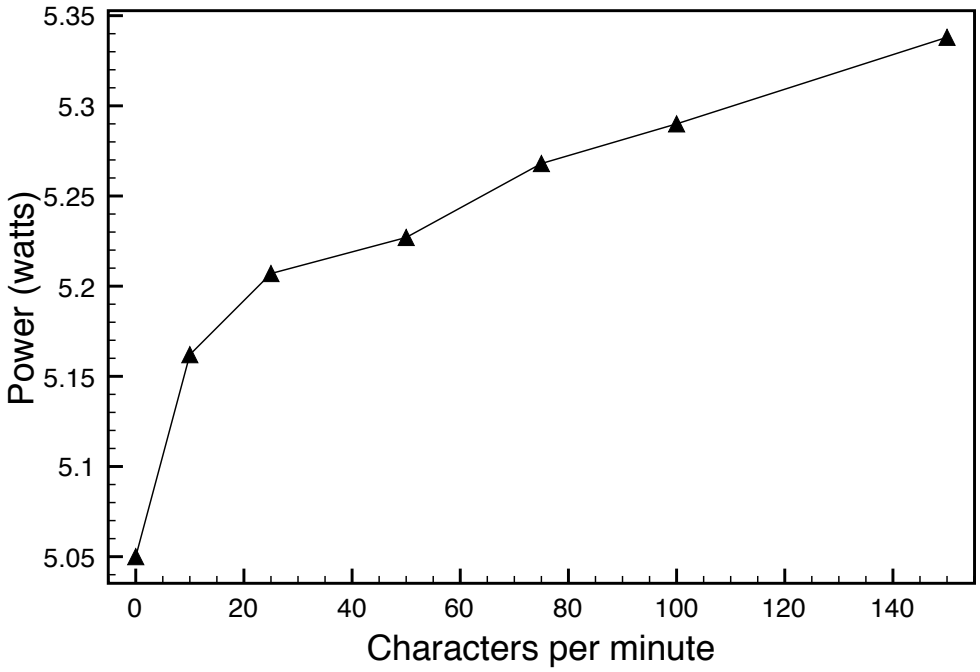


Figure 4.3: Power vs. Typing Rate

4.4 Characterizing Energy Consumption of Typing

Some actions, such as sending messages, are always accompanied with some typing. The energy consumed by typing depends on the number of characters that are typed and the time that is used to type the characters. To better understand how typing rate affects energy consumption, we characterize the typing operation. We do this by first fixing the experiment time to be 1 minute. Then, for each experiment, we type a different number of characters with the iPad keyboard. Finally, we calculate the average power usage in this $1-minute$ window as the power level for a specific typing rate. Figure 4.3 shows the relationship between power consumption and typing rate. The regression function is formulated in Equation 4.4:

$$p_{typing}(\alpha) = 0.1369log(\alpha) + 5.0536 \tag{4.4}$$

where α denotes the typing rate. With the regression function, the energy model can be customized from person-to-person. However, we use a fixed value in our model according to previous research showing that the average typing speed on iPad is approximately 45 words per minute [32]. In addition, statistics show that the average number of letters in an English word is 4.5 [33]. According to the above equation, the power usage of typing is 5.4 watts when backlight level is 1.

4.5 Optimization

We are now ready to formulate the optimization problem. In general, we wish to maximize the *user experience*, which may increase energy costs, but save enough battery life to meet the desired duration of the field study. We assume that the user experience can be improved by having a high backlight level x, GPS update rate, and message fetch rate. However, since it is a non-linear problem (the relation between energy and backlight level x is cubic), it could be time-consuming to solve for all the variables. Therefore we decide to only solve for backlight level x. We describe the optimization problem as follows:

$$\textbf{maximize} \quad x \tag{4.5}$$

subject to

$$\sum_{\tau=t_{now}}^{t_{next}} \sum_{i=1}^{k} (p_i(x) \cdot n_i \cdot l_i) + p_{idle}(x) \left[\tau - \sum_{i=1}^{k} (n_i \cdot l_i) \right] \leq R_{t_{now}} \tag{4.6}$$

$$t_{now} \leq t_{next} \leq t_{finish} \tag{4.7}$$

$$0 \leq x \leq 1 \tag{4.8}$$

Equation 4.6 describes the constraint that the estimated energy cost for the next time step must not exceed remaining energy, $R_{t_{now}}$. The estimated energy consumption is calculated using Equation 4.2. Our model is flexible, in that one can choose any desired duration length t_{finish} of the field study that is suitable for her case. However, it is constrained by Equation 4.7, which means the estimation stops at the end of the experiment. The algorithm for solving this problem is straightforward. We iterate through all the possible values of x, which is $O(1)$ for the iPad, and select the largest x value that meets all the constraints. The power controller module will run this algorithm regularly after the app has launched, and adjust the backlight level based on the output value of x.

Chapter 5

Evaluation

We have carried out real field experiments to evaluate our power model. In this chapter, we first describe our experimental setup and the tools involved, and then we present the evaluation results.

5.1 Experimental Setup

Our application is developed using Objective-C in *Xcode 4.6*, the database is developed using *MySQL*, and the back-end server is developed using PHP. All the tests and experiments are run on iPad2s, which have a 1.0 GHz Apple A5 CPU, a 512MB RAM, a total battery capacity of 6583 mAh, and runs iOS 6.1.2.

To effectively evaluate the model proposed in Chapter 4, we design several experiments that can represent some typical usage patterns of the app. The first experiment represents a regular field study. In our results, we label this set of experiments *Regular-user experiment*. It involves all the operations that a student would perform in a field trip. Those operations include sending messages, request GPS, and submitting a finished task. The second experiment represents the "message-heavy" mode, where user sends messages more

frequently while performing less other operations. In our results, we label this set of experiments *Message-heavy experiment*. This pattern is more likely to happen on the organizer side, who often gives instructions to students but does not collect samples by herself. The third experiment represents the "note-taking" mode, where a user focuses on taking notes of the samples while performing less interactions with the server. In our results, we label this set of experiments *Note-taking experiment*. All three experiments are designed to be 15 minutes long. We believe these modes are representative of realistic scenarios. All experiments are conducted in the ENCS building in Washington State University Vancouver campus with a WiFi connection.

5.1.1 Energy Measurement

We use the *Watts-Up* meter for power measurement [34]. The meter measures voltage and current a thousand of times per second so it has a fast response time which enables user 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 read 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 read from Watts-Up reflects the real-time power usage of the application.

The data is recorded once per second.

5.1.2 Field Study Setup

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

- Thread 1: User reports her location to the server and pulls other team member's location every 5 seconds.
- Thread 2: User pulls new messages from server every 5 seconds.
- Threads 3: 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. The polling frequency can be adjusted to balance between performance and energy consumption, but we leave that for future work.

5.2 Model Evaluation

With all the characterization and data preparation, we can evaluate our model proposed in Chapter 4. As we described in Chapter 5.1, we evaluate our model under three different scenarios. For each scenario, we repeat the experiment with three backlight levels: 0, 0.5, and 1. With each experiment, we log the actions performed by the user and generate the estimated power consumption by our model in onesecond granularity. We are interested in two types of results.

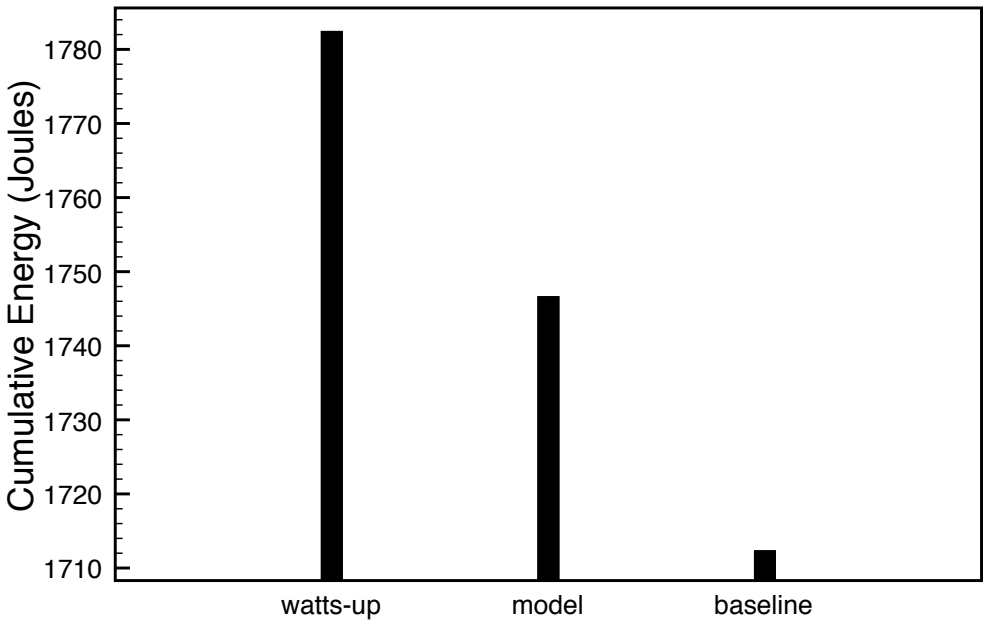
- Power *versus* Time. This plot shows the variation of power level through the whole experiment. We compare the results got from our model and *Watts-Up* meter.

- Cumulative energy consumption. This plot is the integral of the previous power-time function, summing up the total energy consumed in an experiment. We compare the results estimated by our model, the baseline, and the actual measurements from the *Watts-Up* meter.

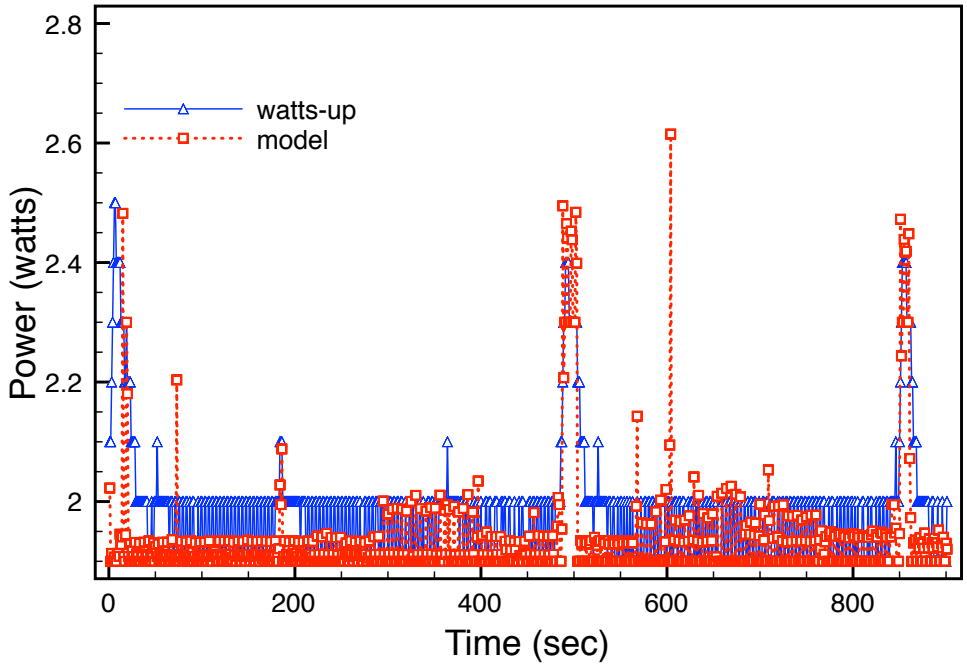
Figure 5.1 to Figure 5.3 show the real-time power usage and energy consumption in the regular-user experiment. From the real-time power usage graph, we can see that in most cases, the power bounces between 1.9 watts and 2.0 watts when backlight is at 0.0. It is very close to the idle power, since the majority of energy is spent on screen display.

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, like 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 capturing the energy spent by these actions. Sometimes, the model gives a much higher prediction than the actual measurement. We believe this is due to the inaccuracy in predicting action execution time. 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, if the transmission time is very long, the model could give a very high energy consumption prediction by using that value. We will improve the time predicting mechanism in the future.

Figure 5.1(a), 5.2(a), and 5.3(a) show that the difference among actual measurements, our model, and baseline model are relatively small compared to the total value, again this is because screen brightness is the major energy consumer, which is also included in the baseline model. However, our model out-performs the baseline model by including all other types of actions. Moreover, the error of our model does not grow with backlight level: 2% error when backlight level is 0.0 and 0.7% error when backlight level is 1.0. The errors

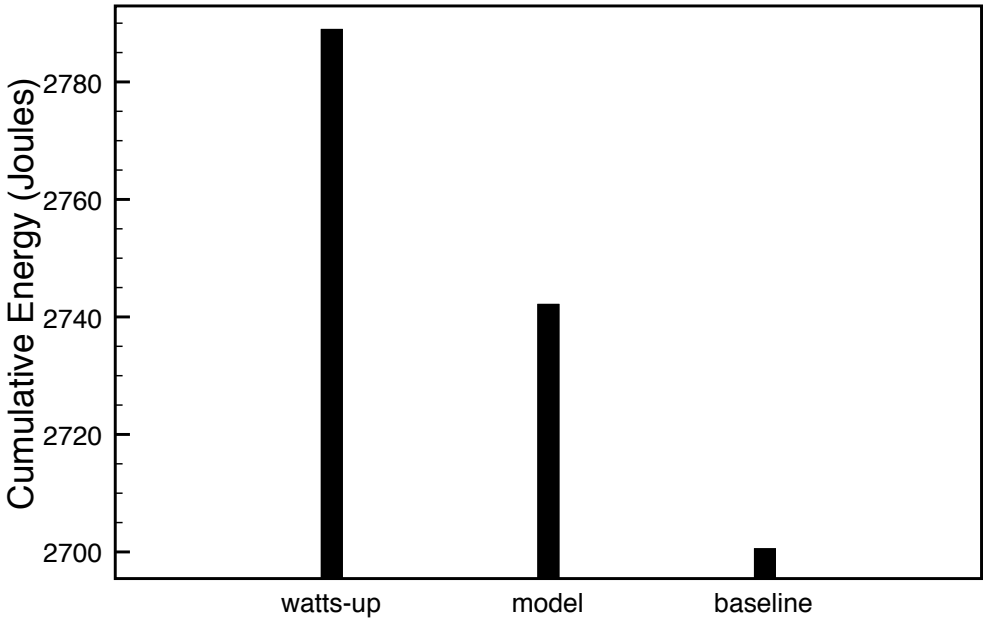


(a) Total energy consumption

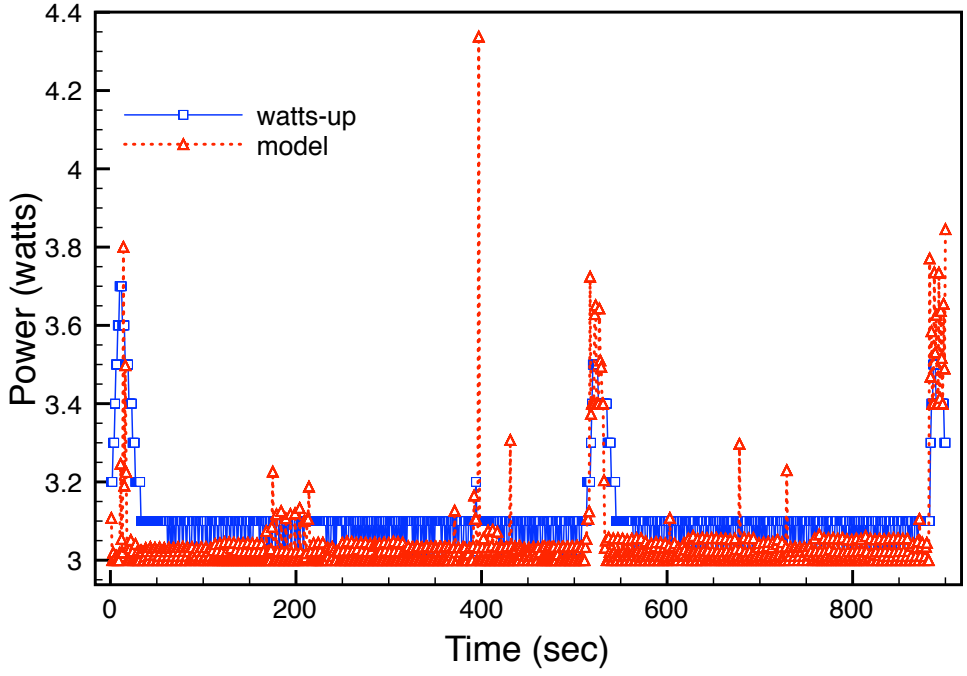


(b) Power vs. Time

Figure 5.1: Model Accuracy (Regular-user experiment with backlight = 0.0)

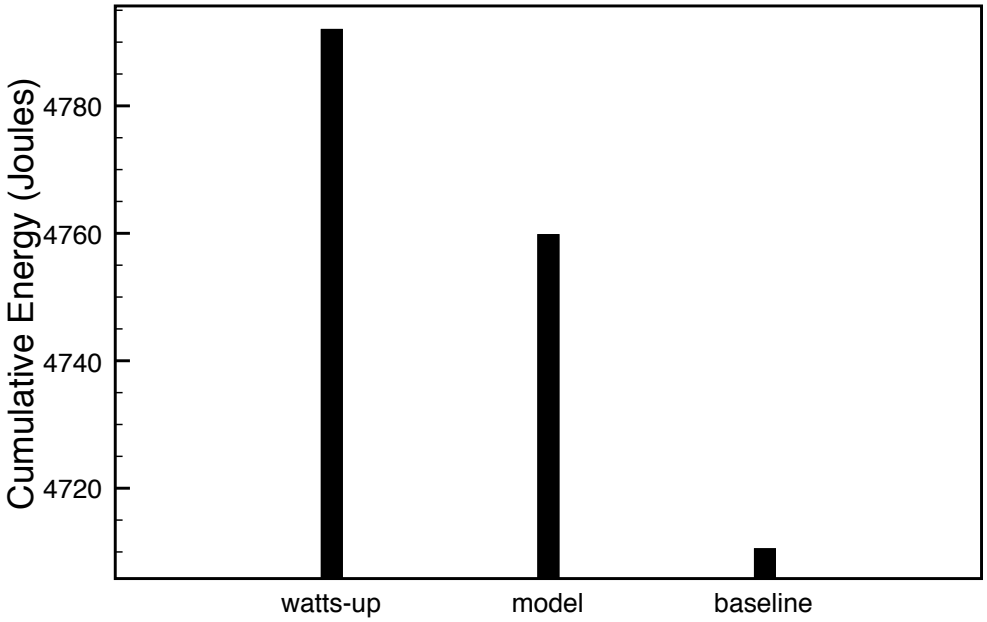


(a) Total energy consumption

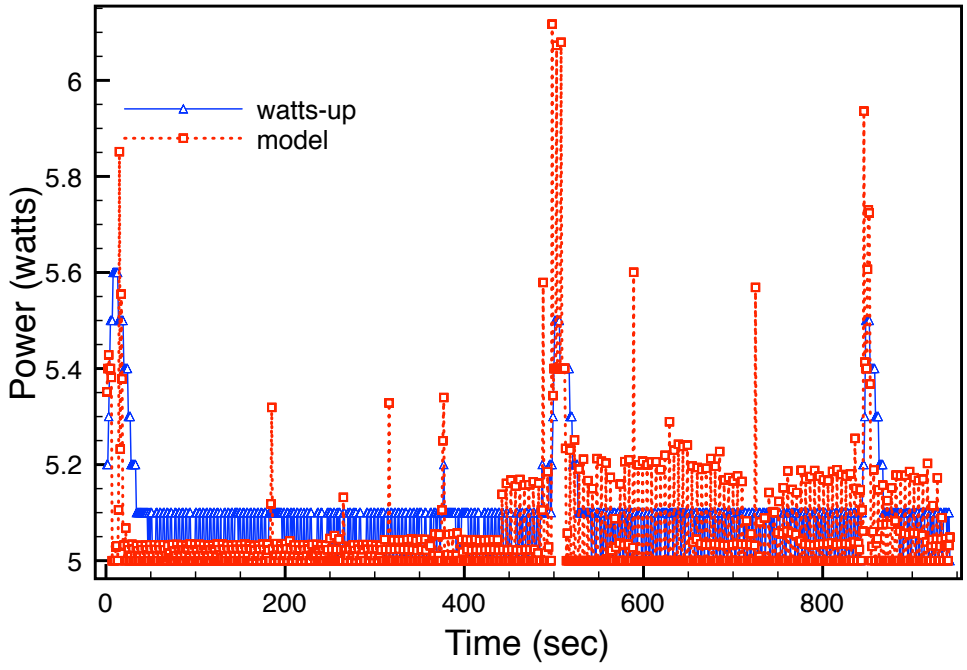


(b) Power vs. Time

Figure 5.2: Model Accuracy (Regular-user experiment with backlight = 0.5)



(a) Total energy consumption



(b) Power vs. Time

Figure 5.3: Model Accuracy (Regular-user experiment with backlight = 1.0)

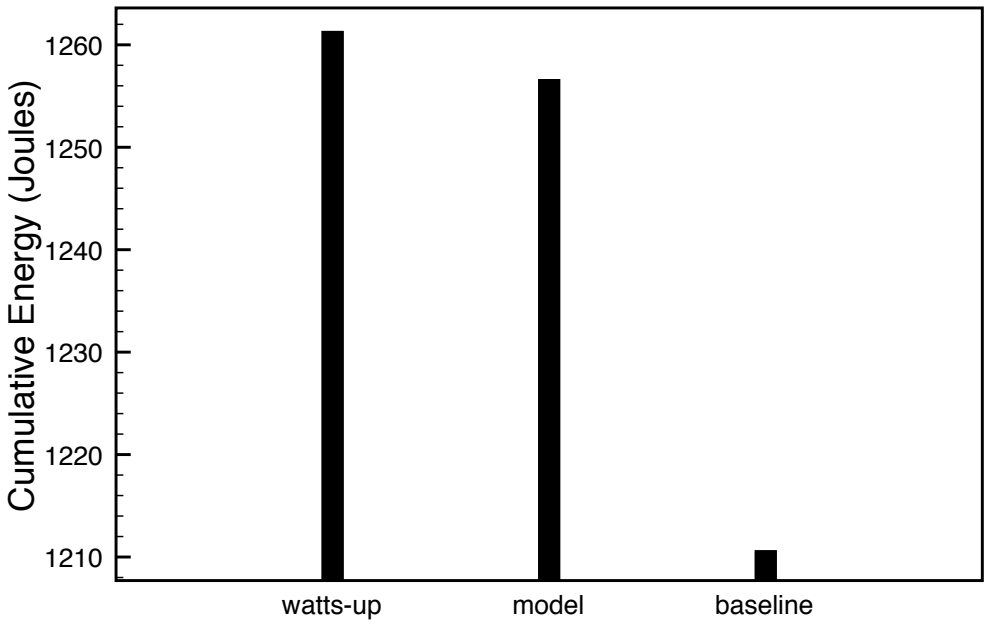
between our model and the measurement data may come from the low-level system calls that we are not able to monitor, such as all the initialization works being done when we launch the application.

In the message-heavy experiment, we perform the "sending message" action every 60 seconds without performing other types of actions. Figure 5.4 to 5.6 show 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. The average error for total energy consumption between the model and measurements is less than 20 joules, showing that our model has well characterized the message sending action.

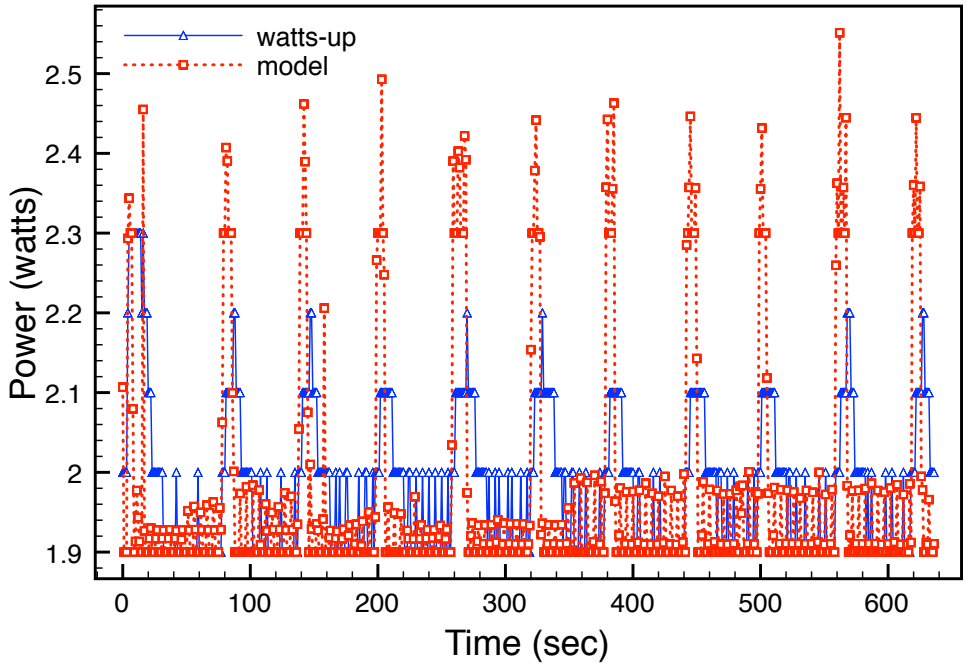
In the note-taking experiment, we perform the notes-taking action every 180 seconds. This experiment involves many typing actions, but less interaction with the server. Figure 5.7 to Figure 5.9 show the results. As can be seen in the figures, although we use a static typing rate assumption, the model still predicts the real-time power usage very well. This is partly because we adapt a moderate typing rate during the experiment. For more accuracy, we will monitor the user's type rate and adapt a dynamic typing rate mechanism for power estimation in the future work. The total energy consumption figures show that the difference between the true measurements and baseline is around 100 joules, which is larger than both previous experiments. This implies that typing costs more energy than other actions.

5.3 Energy Optimization Evaluation

With the optimization algorithm and power management system installed, we are interested in how the application will behave 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 the experiment, we define

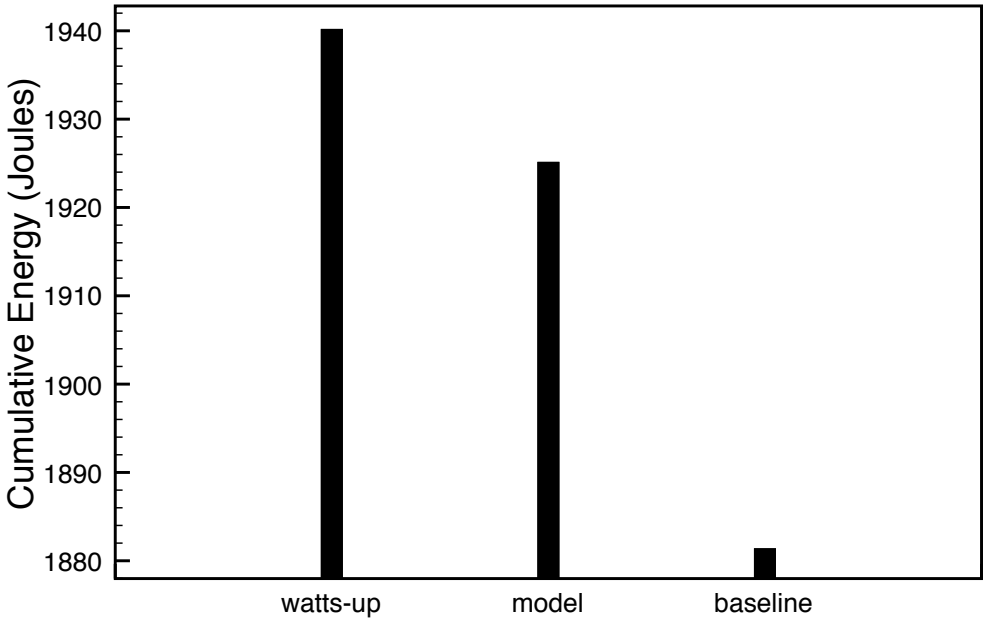


(a) Total energy consumption

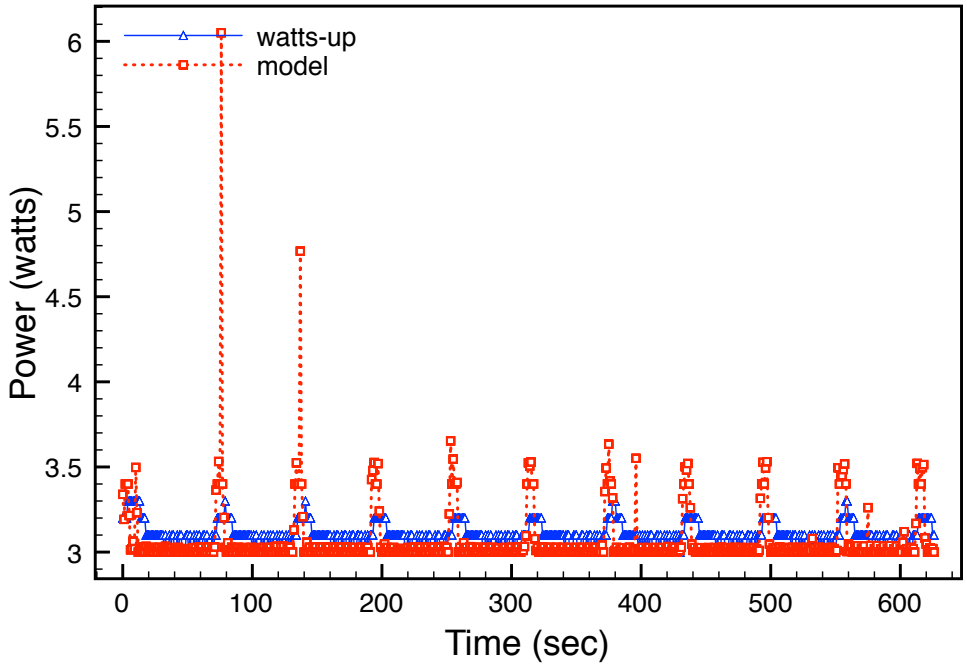


(b) Power vs. Time

Figure 5.4: Model Accuracy (Message-heavy experiment with backlight = 0.0)

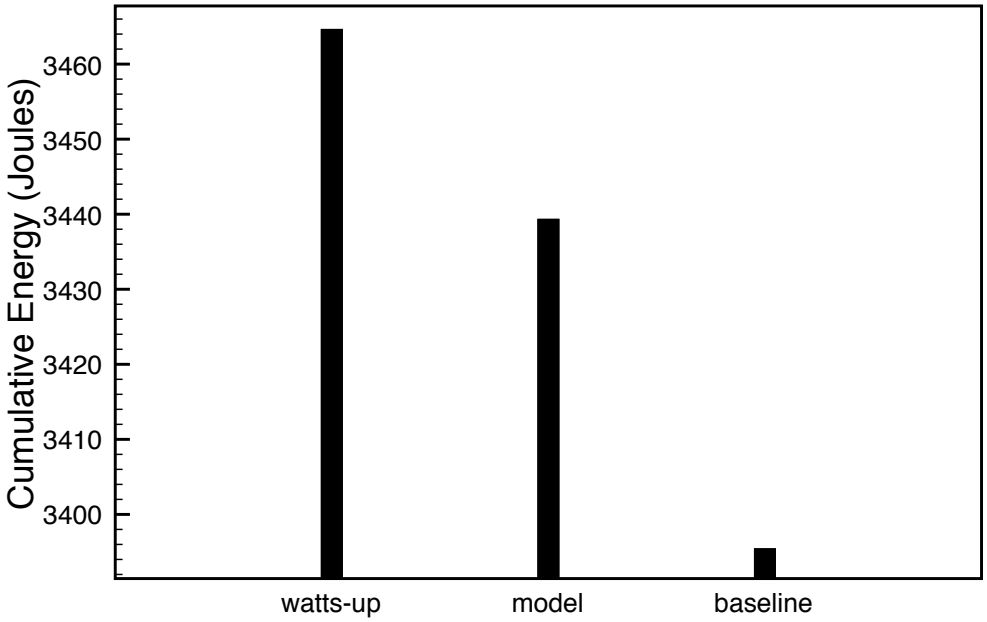


(a) Total energy consumption

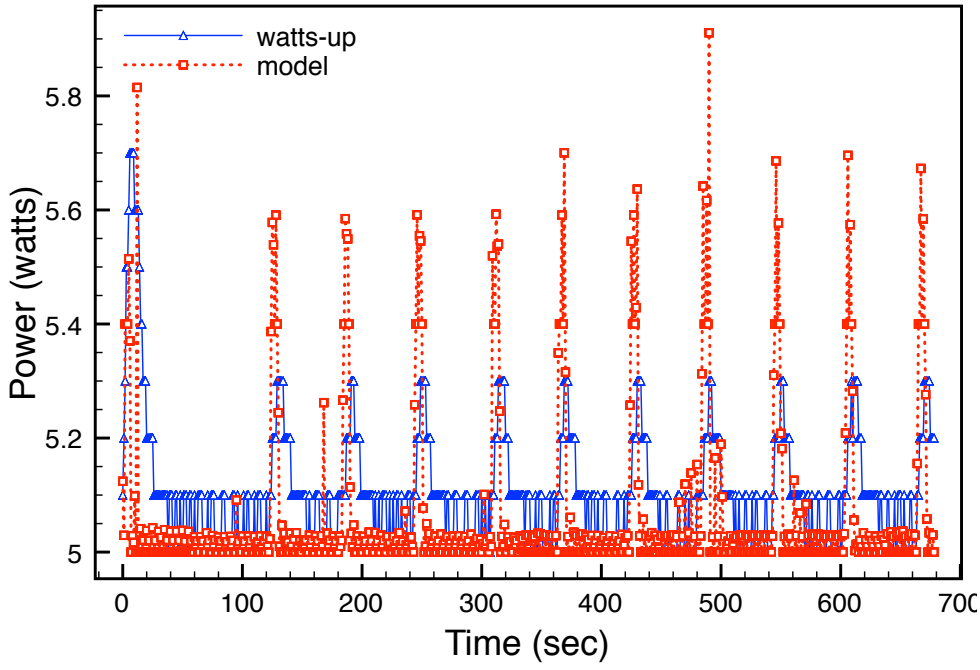


(b) Power vs. Time

Figure 5.5: Model Accuracy (Message-heavy experiment with backlight = 0.5)

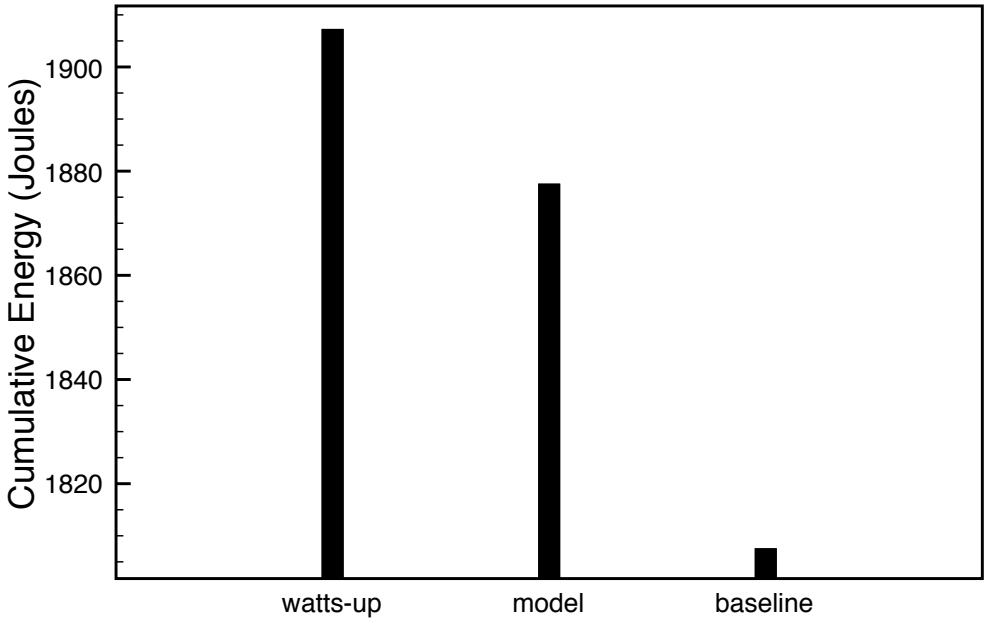


(a) Total energy consumption

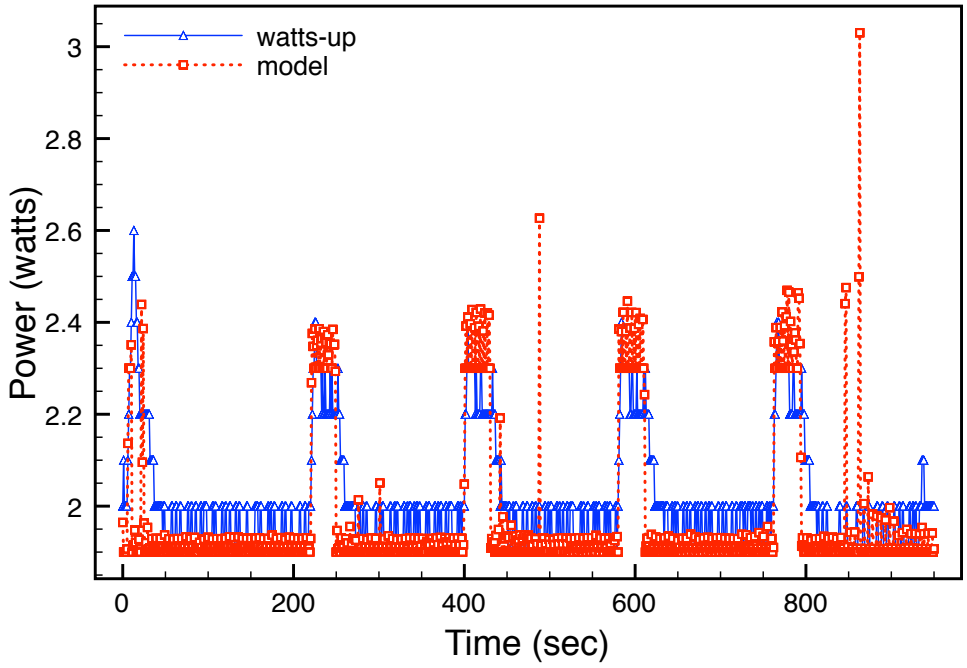


(b) Power vs. Time

Figure 5.6: Model Accuracy (Message-heavy experiment with backlight = 1.0)

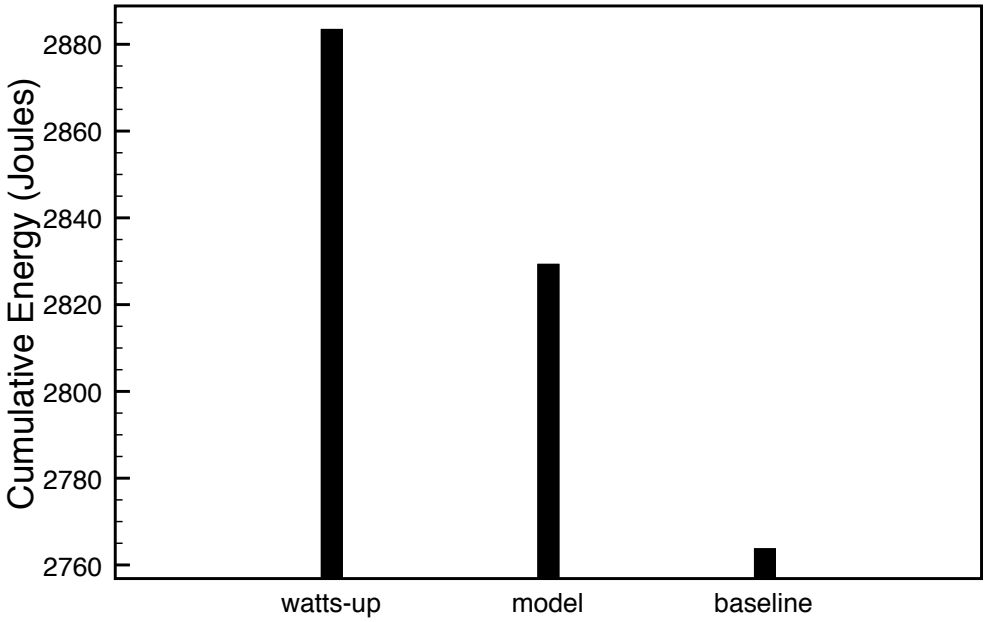


(a) Total energy consumption

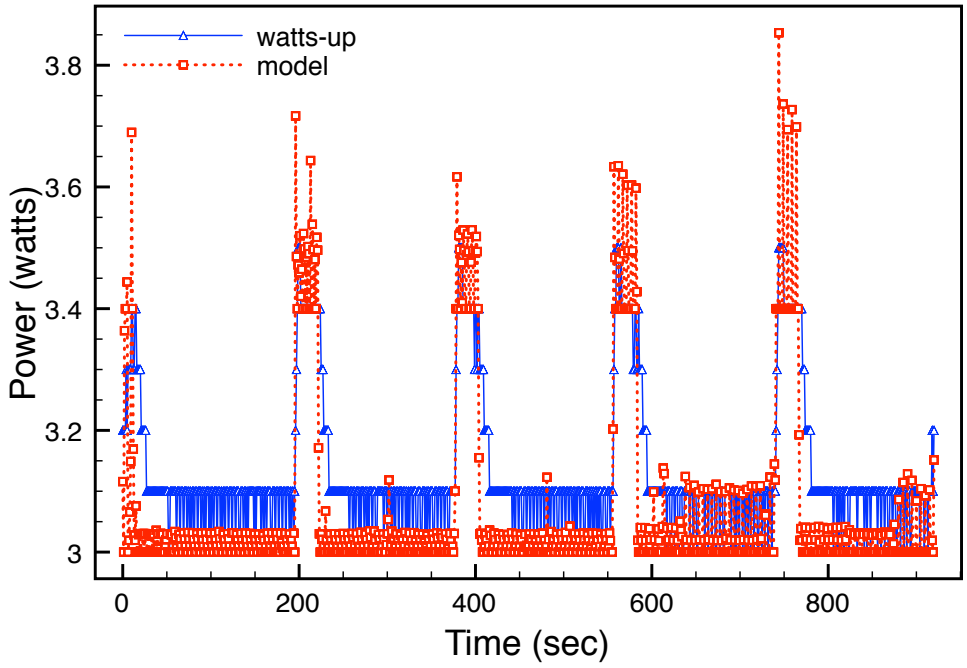


(b) Power vs. Time

Figure 5.7: Model Accuracy (Note-taking experiment with backlight = 0.0)

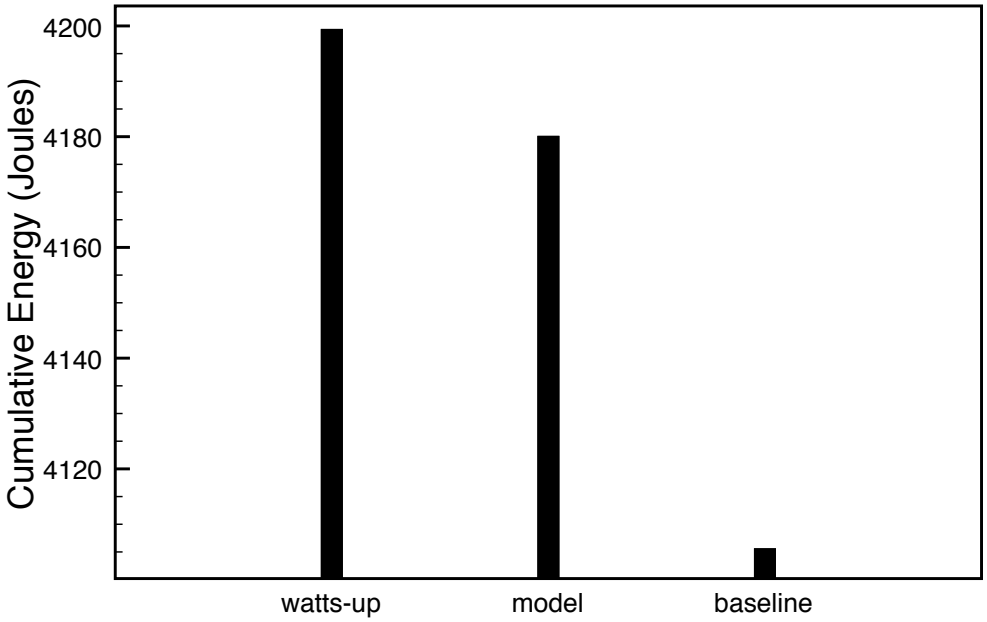


(a) Total energy consumption

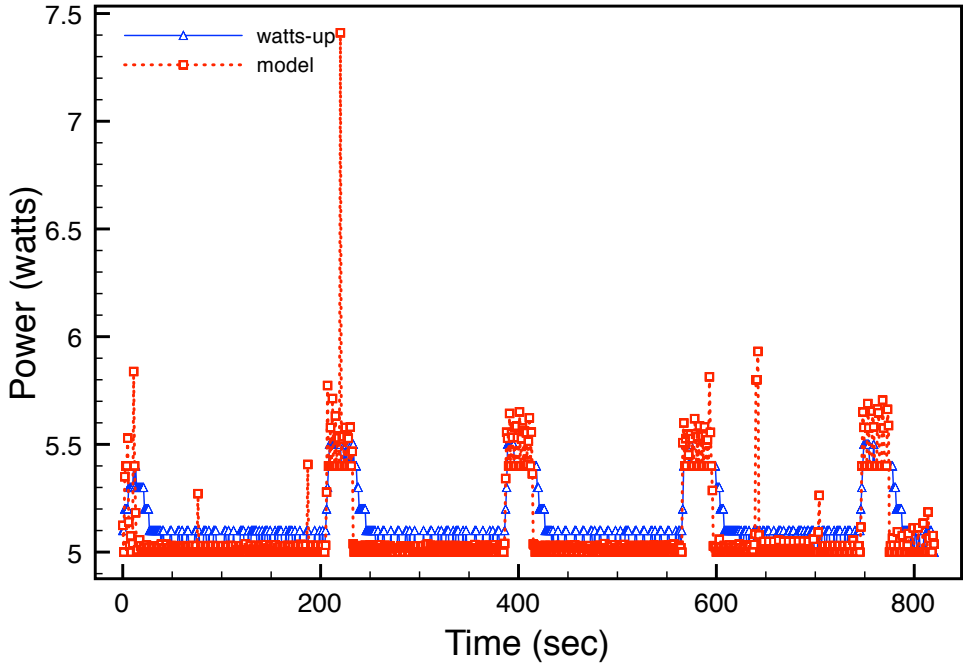


(b) Power vs. Time

Figure 5.8: Model Accuracy (Note-taking experiment with backlight = 0.5)



(a) Total energy consumption



(b) Power vs. Time

Figure 5.9: Model Accuracy (Note-taking experiment with backlight = 1.0)

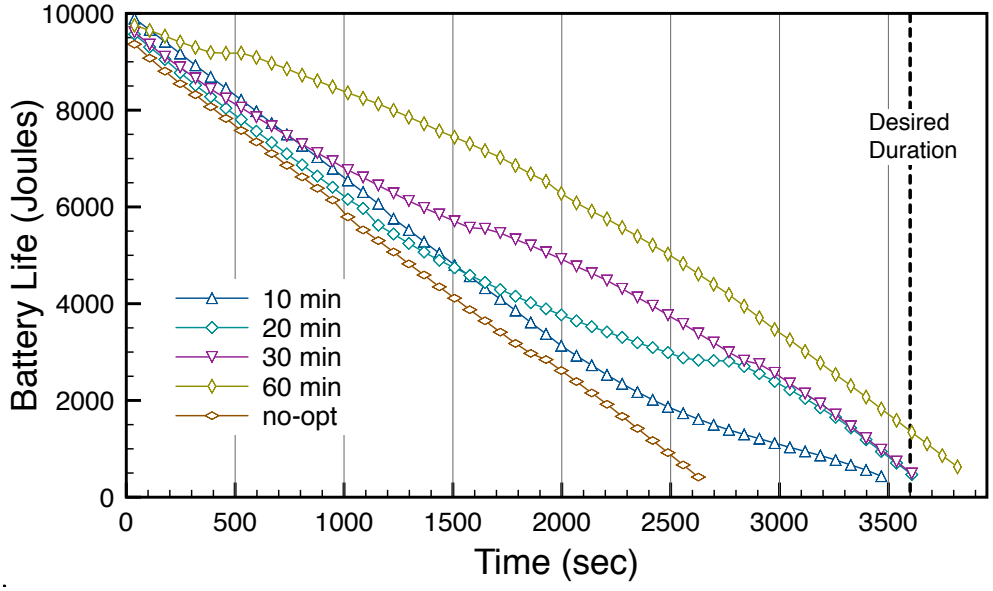
the following parameters:

Parameters	Definition
t_{finish}	Desired experiment duration, or the finish time of the experiment. This value is predefined before an experiment.
w	The sliding window size, or the estimation length. It defines how far into the future the model will be predicting.
τ	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.

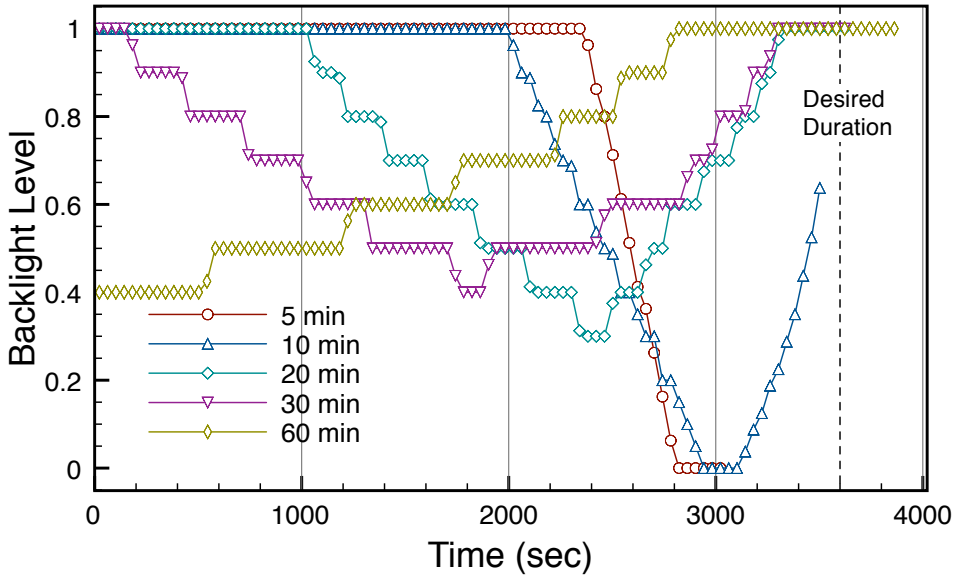
Table 5.1: Parameters used in Optimization

Now we introduce the experiment setup: First we set five different sliding window sizes w, which are 5 minutes, 10 minutes, 20 minutes, 30 minutes and 60 minutes. Second, 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 τ will be 5 seconds. We in total conduct four sets of experiments, which include three user modes we introduce in 5.1 and one idle mode, where the user does not perform any active actions.

As shown in Figure 5.10(a), optimizations with a window size $w \geq 20$ minutes meet the desired experiment duration. In general, the battery life increases with the window size. This makes sense, 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 5.10(b) plots the backlight trend as the experiment

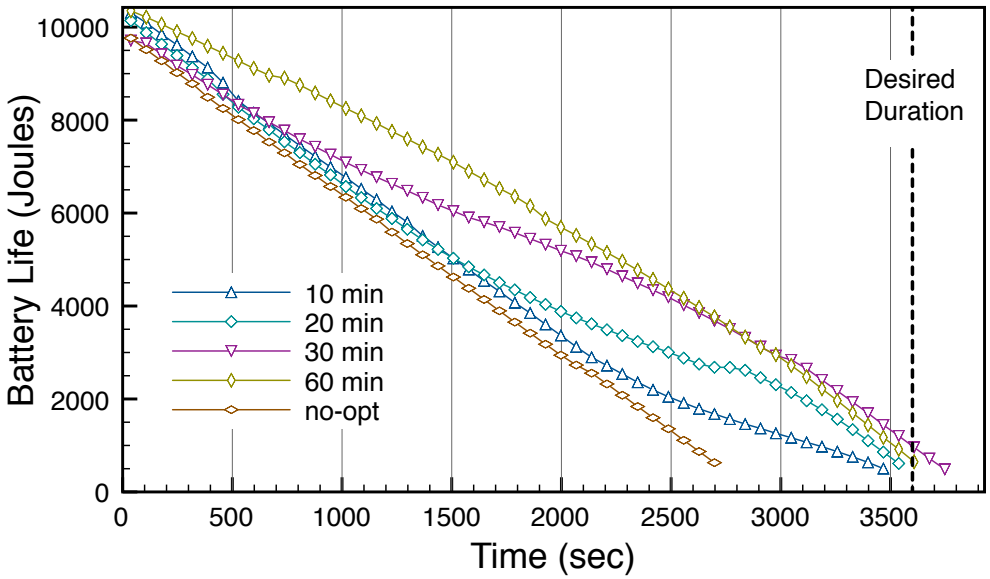


(a) Energy left vs. Time

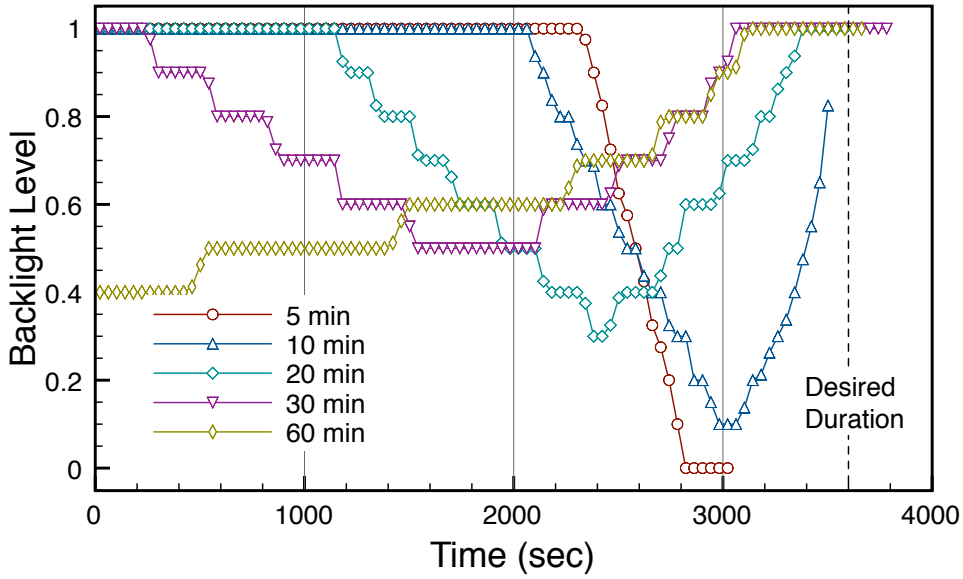


(b) Backlight level vs. Time

Figure 5.10: Optimization algorithm performance (Idle mode)

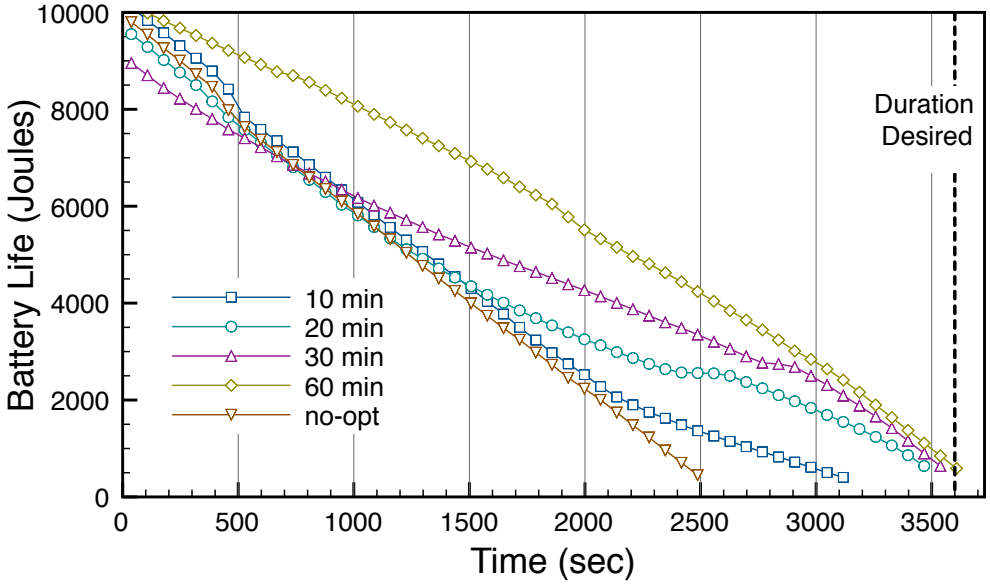


(a) Energy left vs. Time

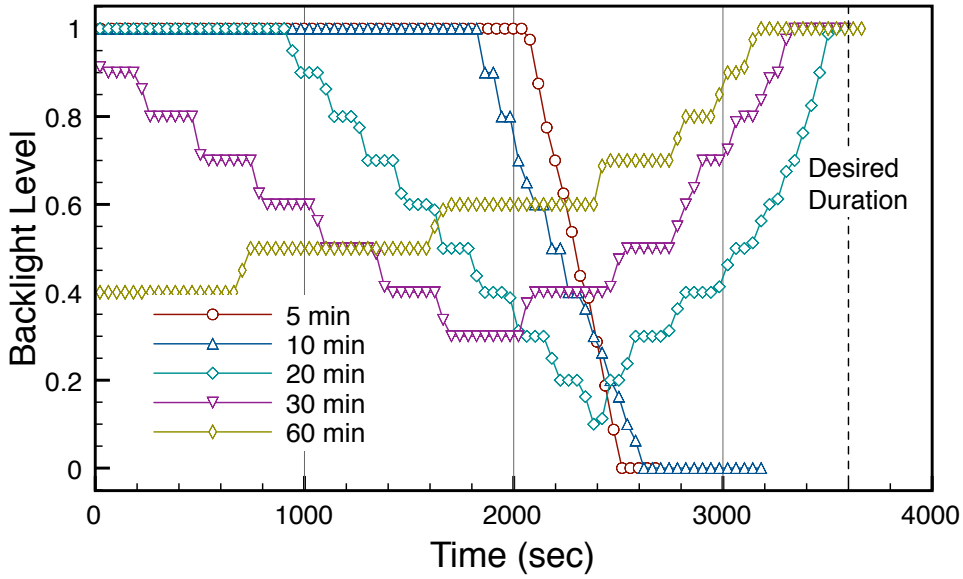


(b) Backlight level vs. Time

Figure 5.11: Optimization algorithm performance (Regular-user mode)

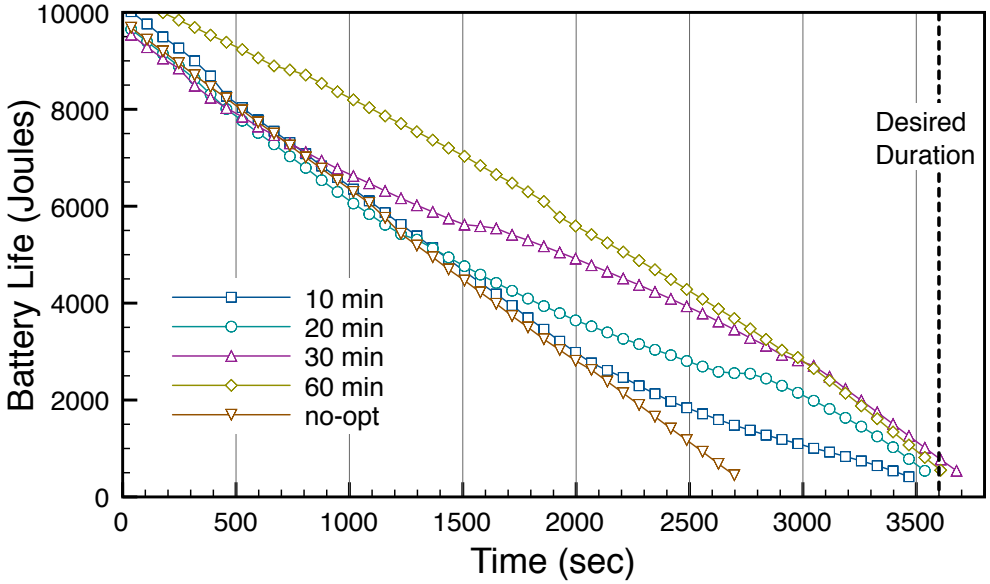


(a) Energy left vs. Time

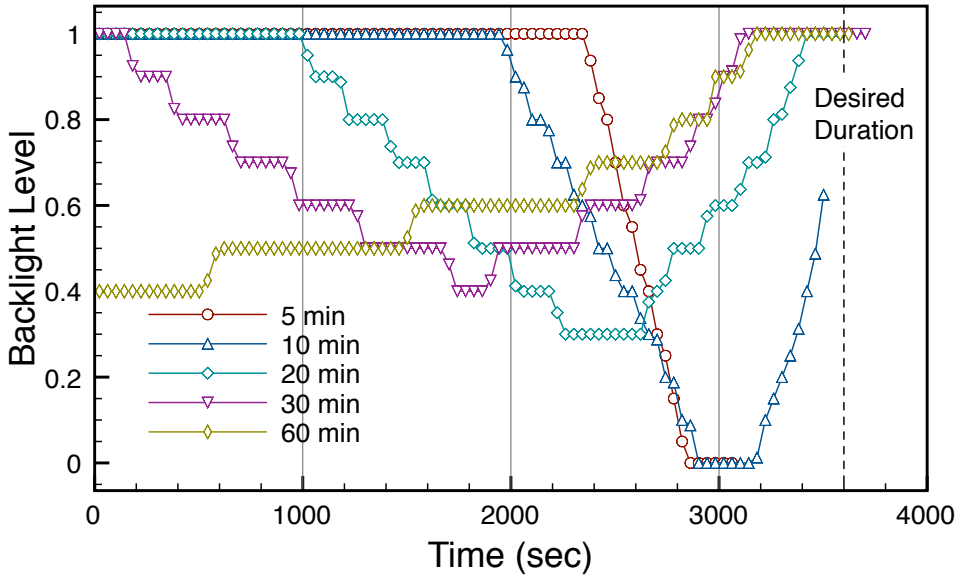


(b) Backlight level vs. Time

Figure 5.12: Optimization algorithm performance (Message-heavy mode)



(a) Energy left vs. Time



(b) Backlight level vs. Time

Figure 5.13: Optimization algorithm performance (Note-taking mode)

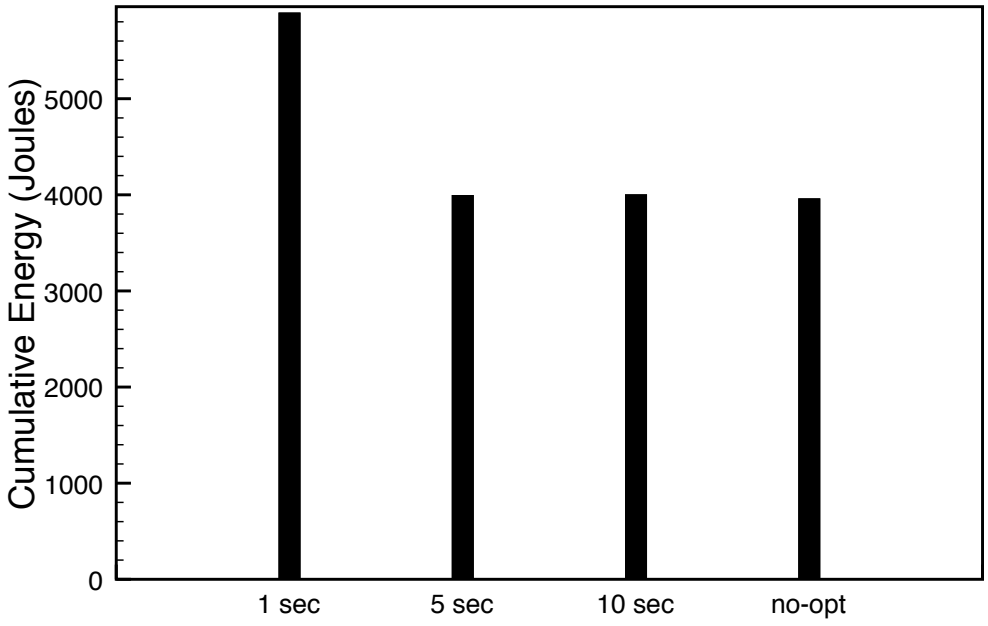


Figure 5.14: Total energy consumption when executing optimization at different rate

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 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 τ. 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 5.14 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.

Chapter 6

Conclusions and Future Work

We design and implement an energy-aware mobile application for scientific field studies. We show how the various components of the application work together to fulfill this goal. An accurate power model that predicts energy consumption based on user actions is proposed, and we evaluat the model by running an extensive set of experiments. We also propose an energy-saving strategy by optimizing the backlight level, and showed the effectiveness of the optimization by running the application under low battery situations.

For future work, we could try to find the optimal sliding window size w for our optimization algorithm. We can also try to find the best optimization intervals τ. Both of these parameters may be dynamic over time: we assumed fixed rates for all app actions – eventually they should also change. We need to find the actual time for executing network communications, which will give a more accurate energy prediction. We could also go deeper into the method/code level to find energy consumption at different level of granularities.

To support the scientists, we could also implement more user-friendly features (like the a record and playback of the experiment, and data collection, management, and integration) that were in the initial proposal. The long-term goal of this project is to advance scientific knowledge via federation of field data and enable sharing, integration, and collaboration

among scientists.

Bibliography

[1] Field research. [Online]. Available: http://en.wikipedia.org/wiki/Field_research

[2] (2009, Oct) Bad trip: School outings get downgraded. [Online]. Available: http://online.wsj.com/news/articles/SB10001424052748703574604574499283752291324

[3] L. A. Barroso, "The price of performance," *Queue*, vol. 3, no. 7, pp. 48–53, Sep. 2005. [Online]. Available: http://doi.acm.org/10.1145/1095408.1095420

[4] (2013, Oct) Apple announces 1 million apps in the app store. [Online]. Available: http://www.theverge.com/2013/10/22/4866302/apple-announces-1-million-apps-in-the-app-store

[5] N. R. C. Committee on Electric Power for the Dismounted Soldier, *Energy-Efficient Technologies for the Dismounted Soldier*. The National Academies Press, 1997. [Online]. Available: http://www.nap.edu/openbook.php?record_id=5905

[6] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 29–42. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168841

[7] L. F. Motiwalla, "Mobile learning: A framework and evaluation," *Comput. Educ.*, vol. 49, no. 3, pp. 581–596, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.compedu.2005.10.011

[8] G. N. Vavoula and M. Sharples, "Kleos: A personal, mobile, knowledge and learning organisation system," in *Proceedings IEEE International Workshop on Wireless and Mobile Technologies in Education*, ser. WMTE '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 152–. [Online]. Available: http://dl.acm.org/citation.cfm?id=645964.674386

[9] Y. Chen, T. Kao, and J. Sheu, "A mobile learning system for scaffolding bird watching learning," *Journal of Computer Assisted Learning*, vol. 19, no. 3, pp. 347–359, 2003. [Online]. Available: http://dx.doi.org/10.1046/j.0266-4909.2003.00036.x

[10] R. Rieger and G. Gay, "Using mobile computing to enhance field study," in *Proceedings of the 2Nd International Conference on Computer Support for Collaborative Learning*, ser. CSCL '97. International Society of the Learning Sciences, 1997, pp. 218–226. [Online]. Available: http://dl.acm.org/citation.cfm?id=1599773.1599800

[11] D. M. Kennedy, "Case studies of mobile applications in scientific field studies," in *eLearn*, 2009.

[12] 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. [Online]. Available: http://dx.doi.org/10.1109/SERE.2012.19

[13] 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. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855840.1855861

[14] 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. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644927

[15] 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. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000026

[16] 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. [Online]. Available: http://doi.acm.org/10.1145/1555816.1555839

[17] 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–. [Online]. Available: http://dl.acm.org/citation.cfm?id=520551.837522

[18] 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. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486801

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

[20] R. N. Mayo and P. Ranganathan, "Energy consumption in mobile devices: Why future systems need requirements: Aware energy scale-down," in *Proceedings of the Third International Conference on Power - Aware Computer Systems*, ser. PACS'03. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 26–40. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28641-7_3

[21] 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. [Online]. Available: http://dx.doi.org/10.1023/A:1016546330128

[22] D. Helmbold, D. Long, T. Sconyers, and B. Sherrod, "Adaptive disk spindown for mobile computers," *Mobile Networks and Applications*, vol. 5, no. 4, pp. 285–297, 2000. [Online]. Available: http://dx.doi.org/10.1023/A%3A1019129116852

[23] H. Huang, P. Pillai, and K. G. Shin, "Design and implementation of power-aware virtual memory," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '03. Berkeley, CA, USA: USENIX Association, 2003, pp. 5–5. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247340.1247345

[24] V. Tiwari, S. Malik, and A. Wolfe, "Compilation techniques for low energy: An overview," 1994, pp. 38–39.

[25] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: Hierarchical power management for mobile devices," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. New York, NY, USA: ACM, 2005, pp. 261–274. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067198

[26] S. Albers, "Energy-efficient algorithms," *Commun. ACM*, vol. 53, no. 5, pp. 86–96, May 2010. [Online]. Available: http://doi.acm.org/10.1145/1735223.1735245

[27] 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–. [Online]. Available: http://dl.acm.org/citation.cfm?id=795662.796264

[28] 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. [Online]. Available: http://doi.acm.org/10.1145/1206035.1206038

[29] 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. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2005.862725

[30] H. Topcuouglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/71.993206

[31] E. Ilavarasan and R. Manoharan, "High performance and energy efficient task scheduling algorithm for heterogenous mobile computing system."

[32] (2010, Nov) Keyboard performance: ipad versus netbook. [Online]. Available: http://usabilitynews.org/keyboard-performance-ipad-versus-netbook/

[33] (2010, Feb) The average number of letters per word in the english language is 4.5. [Online]. Available: http://yehweh.org/profiles/blogs/the-average-number-of-letters

[34] Watts up power meter. [Online]. Available: https://www.wattsupmeters.com/secure/products.php?pn=0#