An Empirical Approach to Smartphone Energy Level Prediction

Earl Oliver, Srinivasan Keshav
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
{eaoliver, keshav}@uwaterloo.ca

ABSTRACT

We conduct a large-scale user study to measure the energy consumption characteristics of 20,100 BlackBerry smartphone users. Our dataset is several orders of magnitude larger than any previous work. We use this dataset to build the *Energy Emulation Toolkit* (EET) that allows developers to evaluate the energy consumption requirements of their applications against real users' energy traces. The EET computes the successful execution rate of energy-intensive applications across all users, specific devices, and specific smartphone user types. We also consider active adaptation to energy constraints. By classifying smartphone users based on their charging characteristics we demonstrate that energy level can be predicted within 72% accuracy a full day in advance, and through an *Energy Management Oracle* energy intensive applications can adapt their execution to achieve a near optimal successful execution rate.

ACM Classification Keywords
C.4 Performance of systems

General Terms
Experimentation, Human Factors, Measurement

Author Keywords
smartphone, energy, user study, energy emulation toolkit, energy management oracle

INTRODUCTION

The proliferation of smartphones is driving a rapid growth in mobile applications. The design of these applications is governed by several constraints that are unique to mobile computing environments. Of these constraints, energy is the one resource that when depleted will render all of the applications on the mobile device, including emergency and essential applications such as the phone, inoperable. Unfortunately, while global demand and use of mobile applications continue to expand, the energy density of smartphone batteries has grown at a comparably insignificant rate [18]. Energy is therefore a critical resource that must be carefully considered in the design of mobile applications.

We consider a class of applications that must operate with little or no user-intervention, must operate over long durations, and consume large amounts of energy. Delay tolerant applications, such as [2, 9, 12, 17], are examples of this class of application. Each application must forward data bundles autonomously and operate continuously since being offline negates the possibility of exchanging data with neighbouring devices. These applications also consume large amounts of energy due to their use of wireless network and file I/O during opportunistic connections, and can easily deplete a battery if left unchecked. We are therefore faced with the question of how to maximize benefit to the user while minimizing impact.

We approach this complex question through a large-scale user study that examines the charging characteristics of smartphone users. Using this information, we build the tools and libraries that allow mobile application developers to adapt to increasing energy constraints. The contributions of this work are threefold:

- First, we build a dataset containing the smartphone usage and energy consumption characteristics of 20,100 BlackBerry smartphone users. Our dataset contains over 1150 years of cumulative data from users spanning 23 time zones and every BlackBerry device type released since early 2006.
- Second, we exploit user energy traces in our dataset to build the *Energy Emulation Toolkit* (EET). Using this tool, developers of energy intensive applications can test their applications' energy consumption behaviour against existing energy traces. Developers may then alter their design or tune algorithm parameters to reduce energy consumption and ensure that their application does not deplete users' batteries.
- Third, by classifying users into one of three groups according to their unique energy consumption characteristics, we demonstrate that energy level can be predicted to within 7% error within an hour and within 28% error a full day in advance. Using our prediction algorithm, we build the *Energy Management Oracle* (EMO) library. By querying the EMO before executing an energy intensive operation, applications can adapt their execution to achieve a near optimal successful execution rate.

RELATED WORK

Although energy consumption has been widely studied [1, 7, 8, 10, 11, 14, 16], little is known about the charging characteristics of users. Rahmati et al. were the first to qualitatively and quantitatively assess how users consumed energy. They provide preliminary insight into charging behaviour, user interfaces for power-saving settings, user knowledge, and user reaction to battery levels [13]. Their experiment consisted of ten users using unfamiliar devices over a one-month period. Their experiment driver also reduced the life of the device to about 40% of its usual lifetime. Falaki et al. have measured the energy consumption and application usage behaviour of 33 Android and 222 Windows Mobile devices [6]. Our BlackBerry user study complements Falaki's work; however, our dataset is approximately two orders of magnitude larger. Moreover, all of our 20,100 users are running our experiment software on their *personal* devices for periods up to several months in duration. We believe that our work is a more representative study. Finally, our experiment driver is fully event driven, does not record data to flash memory, and consumes less than 1% of additional battery life. Its operation is fully transparent to the user.

Cignetti et al. present the design of an energy model that estimates energy consumption with respect to the actions performed in an application [5]. Given that today's mobile devices are highly concurrent and that energy intensive tasks may take place without user intervention, actions performed by users are insufficient indicators of future energy levels. Bellosa examines OS-directed power management in [3]. In this work, system activity is throttled to achieve a desired energy consumption goal. This work is practical for homogeneous server workloads, but would not work under the heterogeneous workloads of a personal computing environment. Ravi et al. exploit a user's location (via cell tower) to predict when the user will charge their device [14]. Based on this prediction and the current energy consumption rate, their work determines the portion of the battery that can be used by other applications and warns the user of potential battery depletion. Unfortunately, all of these systems require knowledge of the underlying hardware.

SMARTPHONE USAGE STUDY

Our measurement study is focused on understanding how smartphone users interact with their devices and how they consume and replenish energy. We begin with a brief overview of the measurement study driver and data that we collect. We then discuss the top challenges in implementing the driver and producing a large-scale dataset of this kind.

Measurement driver

To the detriment of many user-centric experiments, smartphones are often distributed to a mere 10-30 users: fellow researchers, staff members, or undergraduates. Although this has been a convention for several years of mobile computing research (primarily due to convenience), this method of experimentation has several significant flaws:

- Experiment-enabled devices are typically given to participants on a temporary basis, which can affect users' interaction with the device and degrade the accuracy of user-centric results.
- Participants are often unwilling to use an experiment-enabled device as their primary/personal device. This can further bias the accuracy of an experiment.
- Participants are often intimately aware of their role in an experiment, which may further bias the results both positively and negatively.

Our measurement study is shaped by three high-level objectives. First, to achieve a high degree of accuracy, we require significantly more participants than any previous study [6, 13]. Second, to achieve this goal our measurement driver must run on a heterogeneous set of mobile devices with a diverse set of provisioned capabilities. Finally, our driver runs autonomously on participants' personal mobile devices and must therefore be production-grade software.

We develop two BlackBerry applications to collect the desired data: the Standard Logger and the Background Logger.

Standard Logger

The Standard Logger is an event-driven BlackBerry application that runs continuously in the background of a device. Upon installation, the logger records the following information:

Backlight activity: we infer user-interaction with a device by exploiting the device's LCD backlight being on is a necessary condition of user interaction. Using callbacks from the OS, the logger records the time that the backlight turns on and off.

Idle counter: the OS maintains a counter since the last user gesture. The counter is reset whenever the user presses a button, touches the screen, or moves a trackball on the device. The logger records the counter's value when the backlight turns off to determine the duration that the backlight is on but the user is not interacting with the device.

Charging activity: users' battery charging behaviour can be determined by recording when a device is plugged and unplugged from an external power source such as a USB cable or power adapter.

Battery level: the logger records the battery level every ten minutes to determine how users consume energy throughout the day.

Soft shutdown: the logger records signals that the device is powering on and off to account for inconsistencies in the data. For example, a user may power off a device when the battery is low, plug it in, unplug it at a later time, and power on at a full charge.

Device type: to differentiate BlackBerry devices we record the device type and OS version.

Background Logger

The Standard Logger alone was not capable of achieving our desired scale. We therefore partnered with a major BlackBerry software developer to augment their existing software quality logging tools with a simplified version of the Standard Logger. We refer to this augmentation as the Background Logger. The Background Logger collects the same information as the Standard Logger and uploads the data to the company's servers each week. In the remainder of this paper, we will refer to both loggers generically as the *Logger*.

Assumptions
The Logger makes one fundamental assumption: that the user does not run applications that programmatically enable and disable the device's backlight while resetting the device's idle time by simulating user input (keystroke injections). Given that application developers rarely use both operations, we consider these to be reasonable assumptions in our study; however, we do take steps to detect and discard invalid data.

Challenges
Achieving our three high-level experiment objectives was challenging. We believe that other mobile researchers can benefit from our insight on the following top four challenges.

Volatile file systems
Logging data to a file is a standard technique to ensure persistent storage in the presence of power failures or other interruptions. However, file systems can be unmounted at any time either manually by the user or automatically when connected to a PC. Adapting to this problem by frequently flushing I/O buffers to persistent storage can have a significant impact on battery life. The Logger therefore exploits two trends among BlackBerry users to provide reliable logging and save energy: users maintain a high battery level and rarely fully power off their devices. Our technique buffers log data in volatile memory until uploading it to our collection server each night.

Energy constraints
An application that has a noticeable impact on energy consumption will not be successful. In preliminary experimentation, we found that users were intimately aware of their device's normal battery life; any noticeable or perceived decrease in battery life would be attributed to the experimental application. As observed in previous studies [13], polling a device's state can have a detrimental impact on battery life. The Logger exploits callbacks from the operating system whenever possible. Unfortunately, an event driven model can produce inconsistent sequences of events. For example, when powering off a BlackBerry, we frequently did not receive an event signaling that the screen was off until after powering the device on. Similar scenarios were observed when charging. When analyzing the output of an event driven application, one must take into account possible race-conditions in the underlying OS.

Third-party application intervention
User-centric studies must take into account intervention by third party applications. For example, an application that records a user's battery consumption can be affected by a spyware application that was accidentally installed on the device [4]. Unfortunately, collecting a manifest of all applications present on the mobile device would be in violation of the user's privacy. Instead, during the analysis phase of our study, we checked for statistical abnormalities, such as increased energy consumption or excessively long activity blocks. When an abnormality was found, the data was flagged for review, manually inspected, and discarded if suspected to be fraudulent.

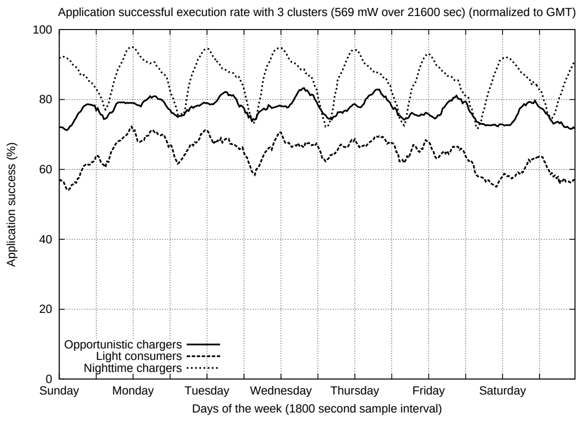
Non-linear time
Time synchronization, a task often taken for granted, is the most challenging problem to overcome in autonomous logging. A device's clock can be updated through three means. Plugging a device into a personal computer and synchronizing with a 'device manager' can change the clock. Users can manually change the time or time zone at any time. A device may also synchronize its clock with the timestamp broadcasted by the cellular network. Changes to the device clock manifest as the appearance of non-linear time. Unfortunately, there is no way to programmatically detect changes to the device clock. We mitigate errors introduced by time changes by detecting when the device is connected to a computer and analyze these events separately from non-connected events. In a large-scale study, users can also traverse many different time zones. UTC time should be used to provide the absolute time of an event. Experiments must also record the device's time zone and when a time zone change takes place.

Aggregate summary
Over six months of data collection, we constructed a dataset that consists of 1150 years of cumulative interaction and energy consumption behaviour from over 20,100 smartphone users. Approximately 15 years of suspicious data from 213 users (a significant quantity of data in its own right) was discarded due to the reasons previously discussed. Our participants span 23 time zones and all BlackBerry device types released since early 2006 [15]. These devices span a wide range of hardware characteristics and similar devices can be found from other manufactures. Our dataset is approximately two orders of magnitude larger than any previous work.

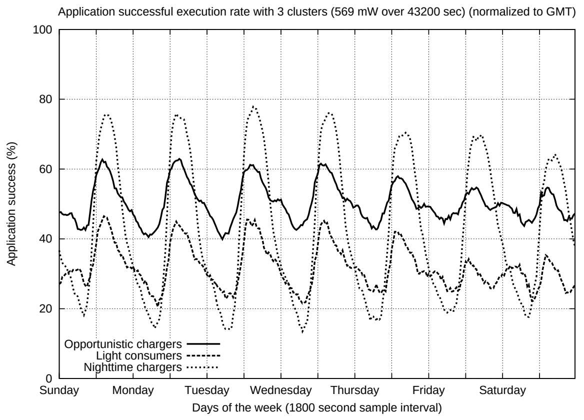
SUCCESSFUL EXECUTION PREDICTION
Using energy consumption traces stored within our dataset, this section examines how knowledge of user charging characteristics can improve the design and implementation of energy intensive applications. We begin with an overview of the Energy Emulation Toolkit (EET) followed by an evaluation using a simple sample application.

Energy Emulation Toolkit
The EET is designed to predict the *successful execution rate* of energy intensive applications given an energy consumption requirement and a specified duration of execution. An application is considered to have 'successfully executed' if it is able to run to completion without draining a device's battery.

The EET operates by iterating over all smartphone users in a target population of either: all users, users with a specific



(a) Six hour execution duration.



(b) Twelve hour execution duration.

Figure 1. Successful execution rate by user type.

device type, or users within a specific *user type*. We will examine the characteristics of each user type in the next section. For each user, the EET assembles their energy traces as a linear timeline and begins stepping through in five minute intervals. Through each step of the timeline, the EET initiates a test to determine if an application invoked at the current time, t, with the specified requirements would succeed. A value of 1 or 0 is recorded for the 30 minute bucket corresponding to t's position in the week to represent a successful or failed execution. The successful execution rate of an application can therefore be provided as a function of the starting time or as an aggregate mean rate.

Implementation
We implement the EET as a stand-alone PERL module that can be imported into any existing PERL program/script. The EET module is initialized with the root directory of the dataset and invoked using a single function call that accepts the energy consumption requirements, the duration of execution, and a value that indicates the target population. Because the EET scans energy trace files linearly it accumulates little state during execution; the EET's runtime footprint when traversing the entire dataset is approximately 46 MB.

Application	Duration	Rate
GSM phone call (900 mW)	1 hour	95.2%
Video playback (1.1 W)	2 hour	84.3%
Bluetooth Class 2 (idle) (2.5 mW)	24 hours	100%
Bluetooth scan (1 min. period) (75 mW)	24 hours	100%

Table 1. Sample successful execution rates.

Evaluation
The EET can be applied to both simple and complex applications. Table 1 provides examples of simple applications and their mean successful execution rate.

We demonstrate the use of the EET through an application that mimics the energy consumption behaviour of existing mobile delay tolerant network applications. It consists of a thread that scans for neighbouring Bluetooth devices every one minute. On every second scan, the application connects to a server on the Internet over WiFi and uploads 100 KB of memory-resident data. Once per hour the application reads a 100 MB file from flash memory and uploads the contents to the server. It then downloads 100 MB of data from the server and subsequently writes the data to persistent flash memory. We implement the simple application, run it across a testbed of BlackBerry devices, and measure the energy consumed. Assuming that the energy consumed by the hourly network and flash memory I/O is amortized over the full hour and found that the average consumption rate is 569 mW. We illustrate the successful execution rate for the sample application over a six hour and twelve hour duration for each user type in Figure 1(a) and Figure 1(b) respectively. We will not analyze these sample figures because the results for each user type will be obvious to the reader later in this paper.

ENERGY PREDICTION
The EET can provide developers useful insight into the expected behaviour of the application when deployed across real devices. Using the EET, developers can alter their software design or tune parameters to reduce energy consumption and the likelihood that the application will deplete the user's battery. Unfortunately, the EET cannot benefit applications whose energy consumption rate is not known *a priori*. We address this problem using the EMO library.

We begin our design of the EMO by first describing the following high-level energy characteristics. We then describe our user classification method, user types, and present an analysis of our user classification based energy level predictor. We conclude this section with an analysis of the EMO library.

Energy characteristics
Charge/discharge durations: The charge and discharge durations are the quantities of time that a device is plugged into or unplugged from an external power source. This property has a direct impact on a device's battery life. Devices that charge for longer durations could have a higher expected battery level than those that do not. Similarly, devices that discharge for long durations could have a battery level that is lower than expected.

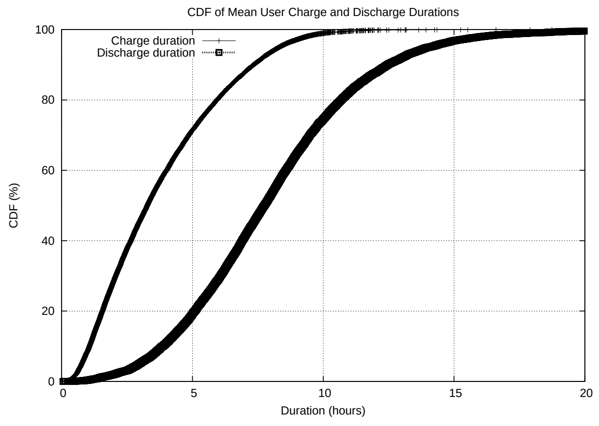


Figure 2. CDF of participants' mean charge/discharge duration.

Charge initiation time/level: The charge initiation time is the time of day when the user begins to charge their device. Regularity in charge initiation time could be used to infer when energy is likely to be replenished. A relationship between charge initiation and battery level could also be used to predict battery level.

Battery level: Patterns in battery level over the course of a day or week could be an ideal parameter for predicting future battery life.

Charge/discharge rates: The charge and discharge rates are the percentage of total battery capacity that is replenished or depleted while plugged or unplugged from a power source. We observed a significant correlation between the discharge rate, device type, and the time of day. This result is not surprising considering the energy consumption disparity between a device sitting idle at night and a device that is used frequently throughout the day.

Other characteristics: Although our dataset contains a rich body of smartphone interaction data, we found that there was little correlation between energy consumption and user interaction. We plan to explore user interaction characteristics in future work.

Classification method

Our user classification method is an iterative process to determine the subset of high-level energy characteristics that best differentiate users. Our metric to determine the utility of a characteristic selection is the mean prediction error derived by clustering users on the selected parameters. Given that the choice of prediction algorithm is dependent on the choice of input parameters, a circular dependency exists. We therefore begin our process by defining a fixed prediction algorithm.

Prediction algorithm

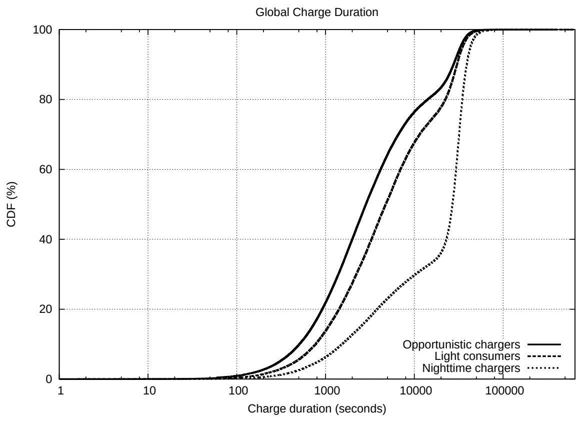
The design of our prediction algorithm was shaped by one high-level goal: to provide predictions over long durations that can easily traverse multiple charge/discharge cycles. The prediction algorithm must therefore adapt to three dominant

Algorithm 1 Predict(t_{curr}, b_{curr}, t_{last}, t_{pred}, γ)

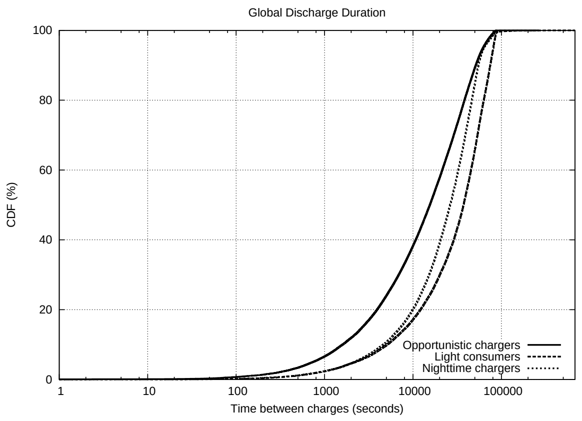
```
Require: $t_{last} \leq t_{curr}, t_{curr} < t_{predict}$
  $b_{pred} \leftarrow b_{curr}$
  while $t_{curr} < t_{predict}$ do
    # Retrieve current cycle duration
    if $\gamma$ (CHARGING) then
      $t_{duration} \leftarrow \delta_{charge}($ bucket($t_{last}$))
    else if $\not\gamma$ (DISCHARGING) then
      $t_{duration} \leftarrow \delta_{discharge}($ bucket($t_{last}$))
    end if
    if $t_{last} < t_{curr}$ then
      # Algorithm invoked mid-cycle
      $t_{duration} \leftarrow t_{duration} - (t_{curr} - t_{last})$
    end if
    for bucket $\beta \in t_{duration}$ do
      # Increment/decrement predicted battery level
      if $\gamma$ (CHARGING) then
        $b_{pred} \leftarrow min(100, b_{pred} + \rho_{charge}(\beta) * |\beta|)$
      else if $\not\gamma$ (DISCHARGING) then
        $b_{pred} \leftarrow max(0, b_{pred} - \rho_{discharge}(\beta) * |\beta|)$
      end if
    end for
    # Advance time by duration of cycle
    $t_{curr} \leftarrow t_{curr} + t_{duration}$
    # The end of a cycle is the beginning of another
    $t_{last} \leftarrow t_{curr}$
    # Alternate the charging state
    $\gamma \leftarrow \not\gamma$
  end while
  return $b_{pred}$
```

trends identified during an aggregate analysis of our dataset. First, the charge/discharge durations vary widely among users. The CDF of each mean duration is illustrated in Figure 2. These durations are also time-varying throughout the week. Second, the mean discharge rate varies significantly over the course of day: a device sitting idle at night consumes energy at a lower rate than a device that is active throughout the day. Moreover, there is a large disparity between the mean charge rate (46.8%/hour) and the mean discharge rate (4.2%/hour).

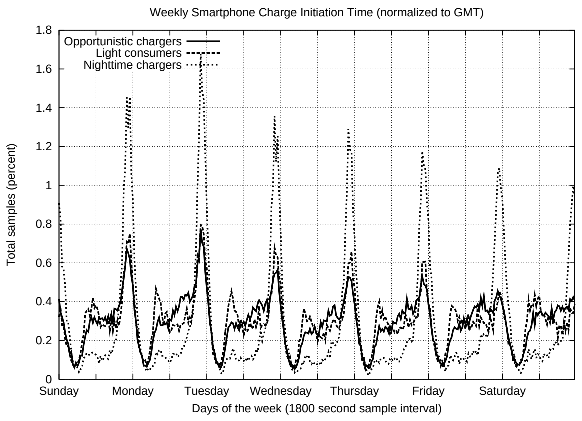
Our prediction algorithm is specified formally in Algorithm 1. We sub-divide each week into 336 discrete 30 minute buckets and initialize the algorithm with four vector parameters: δ_{charge}, $\delta_{discharge}$, ρ_{charge} and $\rho_{discharge}$. The δ vector contain the mean charge/discharge durations for cycles initiated during a specific bucket. If δ is undefined in a given bucket, then the algorithm uses the mean value of the target population. The ρ vector contains the mean charge/discharge rate for each bucket. Again, if a value within ρ is undefined then the mean charge/discharge rate over the target population is used. Each invocation of the algorithm requires the following parameters: the current charge/discharge state (γ), the time that the current charge/discharge cycle (t_{last}) began, the current time (t_{curr}), the current battery level (b_{curr}), and the desired prediction time (t_{pred}). Using the time that the current charge/discharge cycle began, the algorithm examines δ to determine the expected durations of the cycle and the time that the next cycle will begin. While stepping through each



(a) CDF of charge duration.

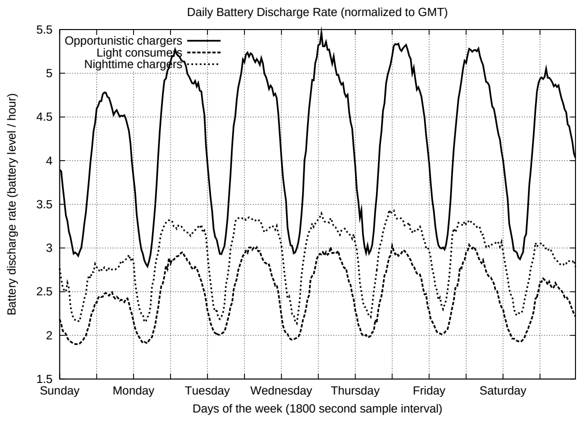


(b) CDF of discharge duration.

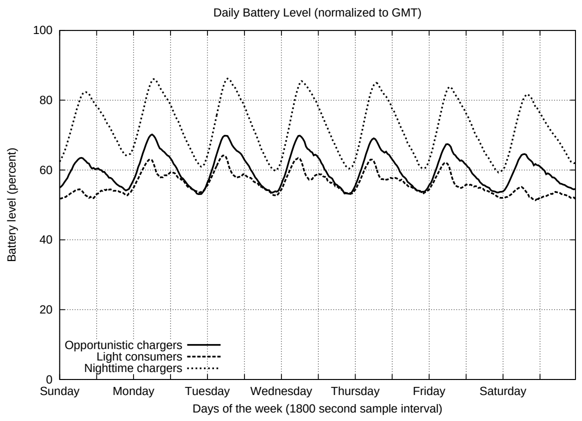


(c) PDF of charge initiation time.

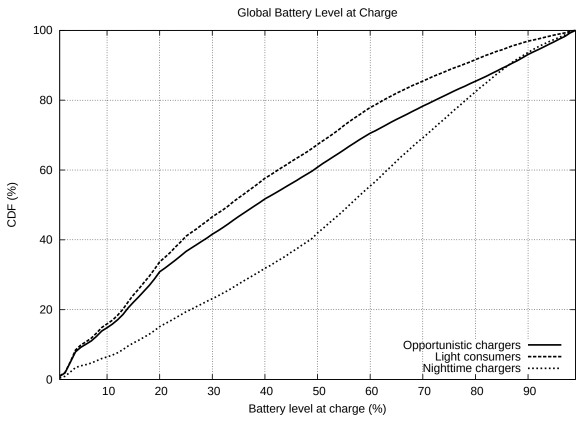
Figure 3. Statistics by user type.



(a) Discharge rate over the week.



(b) Mean battery level over the week.



(c) CDF of battery level when initiating a charge.

Figure 4. Statistics by user type.

cycle, our algorithm increments (charge) or decrements (discharge) the predicted battery level (b_{pred}) at the rate specified in ρ.

Clustering strategy

With a fixed prediction algorithm we now evaluate the prediction accuracy derived by different user classifications. This iterative process involves selecting a subset of energy characteristic, clustering users, and applying the prediction algorithm to each cluster. At the end of each iteration we evaluate the gain/loss in prediction accuracy and select a new subset of characteristics.

Clustering was performed using the k-means algorithm with the mean Euclidean distance between variables as our distance metric. In each iteration we loop through the values

$k = [2, 6]$ clusters. The prediction algorithm is applied to each user clustering for $t_{pred} = \{1, 2, 6, 12, 18, 24\}$ hours using three-way cross validation. We calculate prediction error by iterating through each user's energy trace file in five minute steps and apply the algorithm using the file's current state. The prediction error for each prediction is taken as the absolute difference between the predicted battery level and the true battery level in the underlying dataset. The mean prediction error over all trials and users is averaged over each cross validation stage to produce the mean prediction error for a given subset of characteristics and cluster count (k).

At the end of this iterative process we found that clustering users by their mean weekday and weekend charge/discharge durations yielded the highest prediction accuracy. Prediction accuracy converged at three clusters.

User classification

Before a formal evaluation of the predictor, we briefly describe each user type and their differentiating characteristics.

Opportunistic chargers: Opportunistic chargers are the most common type of smartphone users and represent approximately 63% of the population. These users are primarily characterized by frequent, short charge durations during the hours of 8am to 5pm. The CDF of charge duration for opportunistic chargers is illustrated in Figure 3(a). Of the three user types, these users are the most aggressive energy consumers, consuming nearly 4.8% of their device's energy per hour. The discharge rates for each user type throughout the week are illustrated in Figure 4(a).

Light consumers: Light consumers have the lowest energy discharge rate among all three user types. These users represent approximately 20% of the population. These users charge for longer durations than opportunistic chargers, but discharge their devices over a longer duration. The CDF of discharge duration is illustrated in Figure 3(b). Despite having the lowest discharge rate, light consumers surprisingly maintain the lowest mean battery level of 56.0% as illustrated in Figure 4(b). These users also allow their battery to drop to its lowest level before initiating a charge. On average, a light consumer initiates a charge when their battery level has dropped to 34%. The CDF of battery level when a charge is initiated is illustrated in Figure 4(c).

Nighttime chargers: Our final class of user is those that charge at night. These users represent 17% of the population. The charging behaviour of the user is best illustrated in Figure 3(c) as the PDF of the time that users initiate a charge. The daily spike between the hours of 10pm to 11pm illustrate that these users initiate a charge (probably) before going to bed. Given that these users charge predominantly during the night, their mean charge duration is significantly higher than the other two groups, as illustrated in Figure 3(a). Although nighttime chargers consume only 0.5% more of their battery per hour than light consumers, their long charging durations serve to maintain a mean battery level of 72.5%. Similarly, nighttime chargers initiate a charge at an average battery level of 56%.

Device independence

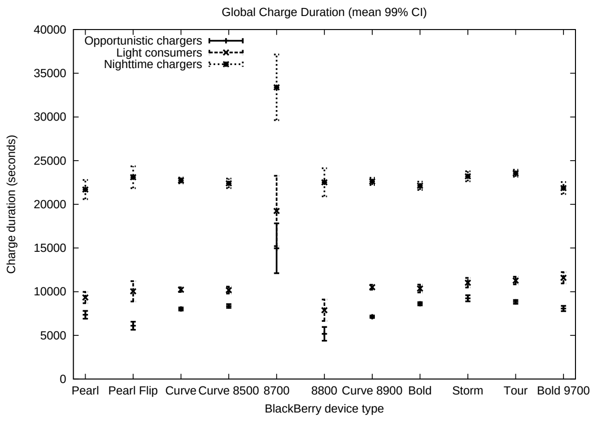
We briefly address the impact of device hardware on our two clustering parameters: charge and discharge duration. Using a t-test, we found that there was no significant different between device types at a 99% CI. These results are illustrated in Figure 5(a). With the exception of the 8700, nearly every 99% CI contains the each cluster's population mean. We observe similarly results in the discharge case, as illustrated in Figure 5(b). Although there is more variation among the CIs for each device, we observe that nearly ever CI contains the cluster's population mean. The user's charge/discharge characteristics are clearly independent of the device type used.

Predictor analysis

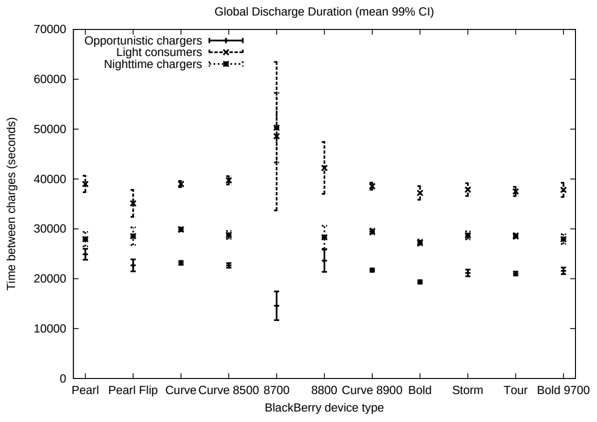
We evaluate the accuracy of user classification-based prediction (*Clustered Predictor*) by comparing it to predictions based on the entire participant population (*Simple Predictor*), device type (*Device Predictor*), and predictions made on an individual basis (*Individual Predictor*). With a fixed prediction algorithm, alternating between predictors is performed by initializing the algorithm with the δ and ρ statistics of the training population. Although there are an innumerable number of possible predictors, we believe that this set successfully demonstrates the benefit of a user classification-based prediction scheme.

We evaluate each prediction algorithm through three-way cross validation of our dataset. For the Simple Predictor, we divide the dataset into three equal sized random subsets. We train each predictor on two thirds of the population, the *training set*, and apply the predictor to the remaining third of the dataset, the *target set*. Applying the predictor on a participant's traces requires that we iterate through all energy trace files in five minute steps. At each step through the trace log, we apply each predictor, store the future predicted battery level, and compare the current true value with a previous prediction. We store the absolute error for each prediction for the current bucket. This process is repeated for two remaining portions of the dataset. After applying the predictor to each third of the dataset, we compute the mean absolute error for each bucket in the week.

We evaluate the Clustered Predictor by dividing the training set and target set into two sets of three clusters. Given that clusters are unordered sets, we order the clusters to ensure a maximal overlap with the user types previously described. For each of the three user types, we train the predictor on the cluster in the training set and apply it to the corresponding cluster in the target set. As before, the absolute error for each battery level prediction is stored in a bucket as previously discussed, and the mean absolute error can be computed over the course of the week. Figures 6(a) and 6(b) illustrate the mean absolute prediction error for one and 24 hour predictions for each of the four prediction algorithms. These figures illustrate several important findings. Predictions based on global statistics perform the worst. This is not surprising given that we have shown that users differ. As previously illustrated, there is more variability in the cluster parameters between user types than between users of the same device type. It is therefore not surprising that the Device Predictor

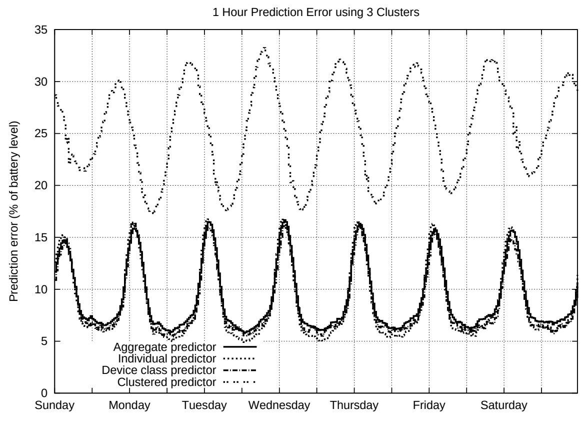


(a) Charge duration for each user type and device type.

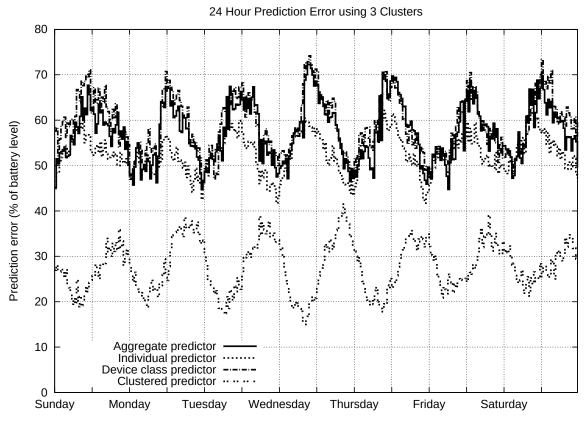


(b) Discharge duration for each user type and device type.

Figure 5. Durations by user and device type.



(a) One hour absolute prediction error.



(b) Mean absolute one day prediction error.

Figure 6. Mean absolute prediction error.

also performs poorly. The relationship between the Individual and Clustered Predictors was not expected. Over short durations, predictions made on an individual basis are significantly more accurate than user type predictions. Conversely, over long durations the Clustered Predictor significantly outperforms individual predictions. At predictions of approximately 12 hours, the predictors yield equivalent results.

Over long durations, the Clustered Predictor outperforms the Individual Predictor due to a reduced error accumulation rate. In the individual case, the δ vector is calculated over few samples and has a correspondingly high standard deviation. In the clustered case, δ is calculated as the mean of the δ of each constituent user (a mean of a mean), which is a closer approximation of the population mean, with a reduced standard deviation window and lower propensity to accumulate error. We illustrate this explanation in Figure 7. The peaks in mean error in the evenings of each graph are attributed to the order of magnitude in disparity between the charge and discharge rates. Recall from Figure 3(c) that all three user types tend to charge their device at night. Our algorithm operates by estimating the duration of a charge/discharge cycle. An incorrect estimation of either duration results in a rapid gap between the predicted and true battery level.

Drawing from these observations, we introduce a fifth predictor, the *Hybrid Predictor*, that combines the short-term accuracy of the Individual Predictor with the long-term accuracy of the Clustered Predictor. The Hybrid Predictor is initialized with both an individual user's energy characteristics, the characteristics of all three user types, and the centroid coordinate of each user type's cluster. For predictions of less than 6 hours, the Hybrid Predictor yields the same results as the Individual Predictor. For predictions over six hours, the Hybrid Predictor examines the individual user's δ values and selects the user's type based on the closest cluster centroid. The predictor then exploits the statistics underlying the user type to provide predictions as if they were from the Clustered Predictor. In practice, both the energy characteristics of the three user types and their centroid coordinates would be hard coded into the predictor; allowing the code to be embedded into an energy-intensive application to actively predict the device's future energy level. We illustrate the mean prediction error up to 24 hours in advance from all five predictors in Figure 8.

Energy Management Oracle

Using the Hybrid Predictor, and the ability to predict the future battery level, we now present the design and analysis of

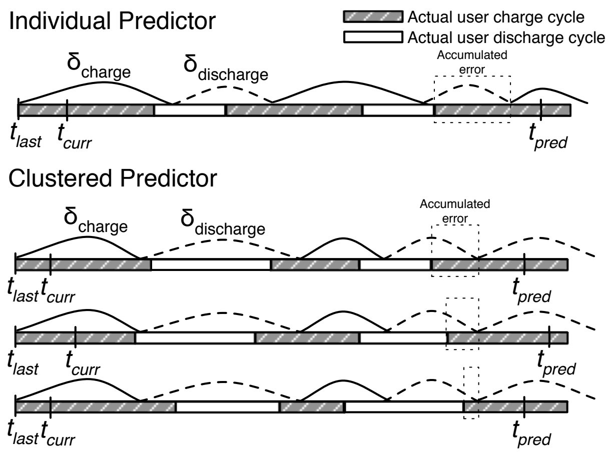


Figure 7. Accumulation of error.

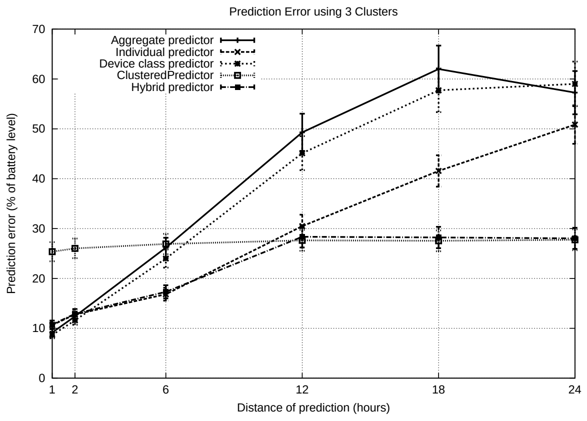
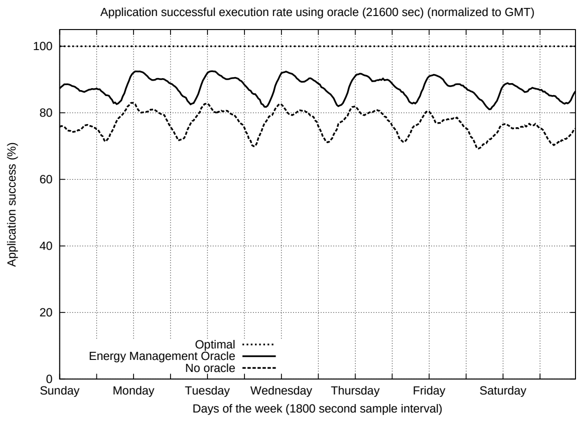
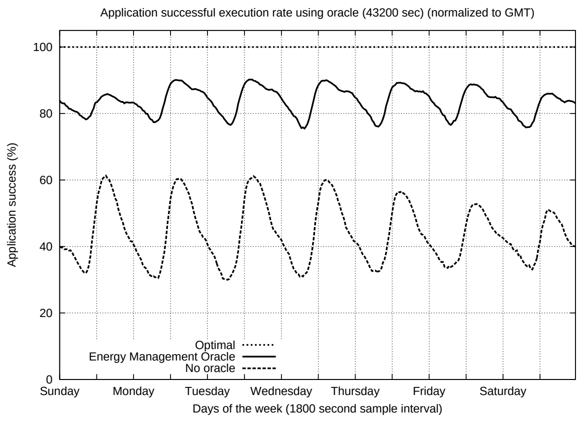


Figure 8. Mean absolute prediction error.



(a) Six hour execution duration using EMO.



(b) Twelve hour execution duration using EMO.

Figure 9. Evaluation of EMO.

the EMO library. The EMO is designed to provide energy-aware hints to applications at runtime. Using the library, applications query the EMO prior to initiating an energy intensive operation. Upon each query, the EMO decrements the current battery level by the energy consumed by the operation and initiates a 24 hour battery level prediction. The EMO query returns ‘true’ if the application can safely execute the operation, or ‘false’ if the operation will result in the depletion of the battery.

Implementation

Our implementation of the EMO modifies the Hybrid Predictor slightly by halting the prediction if the predicted battery level drops to zero. Like the EET, the EMO is implemented as a stand-alone PERL module that can be imported into any PERL program/script. PERL is supported on a wide range of mobile devices. For devices such as the BlackBerry and iPhone that do not support PERL, the algorithms underlying the EMO could be implemented in Java or Objective C in two or three days.

Evaluation

We evaluate the EMO through simulation of the sample application previously described. Recall that the application has two energy intensive operations: uploading 100 MB and downloading 100 MB to file. We will refer to these jointly as the *optional operation*, which consumes approximately 1727 Joules of energy when executed. If the optional operation is performed, the application will consume energy at a *standard rate* of approximately 322 J/hour by scanning neighbouring devices and uploading memory resident data.

Our simulation engine consists of a modified version of the EET that decrements the battery level at the standard rate over a specified execution duration. Like the EET, the simulator steps through each energy trace file in five minute steps. In each step, it initiates a simulated execution of the application. During each simulated hour, the application queries the EMO to decide if it should perform the optional operation. The simulator will decrement the battery level by 1727 Joules if and only if the EMO returns ‘true’. Like the EET, the simulator records a success or failure for each bucket, which allows us to compute the successful execution rate as a function of the starting time of the application.

We compare the results of the simulation with two additional scenarios that demonstrate the upper and lower bounds on the execution success rate. The first, optimal scenario utilizes an oracle with perfect future knowledge; this oracle scans ahead in the trace file and returns 'true' if and only if the application will succeed. The lower bound is determined by running the EET as normal, where the simulated application executes without the benefit of battery prediction and the EMO. Figure 9(a) and Figure 9(b) illustrates the aggregate successful execution rate of the sample application over a six and twelve hour period respectively. Over six hours, the EMO is able to increase the successful execution rate from 76.7% to 87.6%; a 46.5% reduction in failures. Over long durations, the gain is more substantial. Without the EMO, the application exhibits a successful execution rate of only 43.8%. Applying the EMO increases this rate to 84.6%, which amounts to a 93.1% improvement. We therefore believe that the EMO can add significant value to energy intensive applications.

CONCLUSION

This paper has detailed the design and deployment of a large-scale smartphone user study that examines how users interact with and consume energy on their personal mobile devices. Our dataset contains over 1150 years of cumulative data for 20,100 users from around the globe. Although this work focuses exclusively on energy consumption, we believe that it contains a wealth of knowledge outside this context; spanning areas such as interaction design, battery provisioning, and non-technical domains such as addictology, polysomnography, and psychology.

As the scale and complexity of mobile applications continue to increase, we believe that energy consumption will become a new dimension in mobile software evaluation. Applications that consume large amounts of energy over long durations will be practical for only a subset of the user population with compatible energy characteristics. We predict that in the future, software will be distributed with a specified energy demand in the same way that resources such as CPU speed, memory/disk capacity, and network bandwidth requirements are specified in today's software. The Energy Emulation Toolkit is perfectly positioned to support this trend by providing developers with tools to evaluate their applications across all users, device hardware, and user types.

Smartphone users, like all complex objects, can be classified by an innumerable number of variables. Our work differentiates users on the basis of their daily battery charge and discharge characteristics, and has identified three user-types: opportunistic chargers, light consumers, and night-time chargers. This classification scheme provides 72% accuracy on predictions between 12 to 24 hours in advance. Using our predictor we built the Energy Management Oracle library, which can be queried by applications prior to the execution of an energy intensive operation. Using the oracle, we demonstrate that applications can achieve a near optimal successful execution rate. Using our dataset, we expect that other researchers will discover more efficient energy prediction techniques and that user differentiation will continue to yield similar improvements in prediction accuracy.

REFERENCES

1. Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *MobiSys '07*, pages 179–191, New York, NY, USA, 2007. ACM.
2. G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath. Combine: leveraging the power of wireless peers through collaborative downloading. In *MobiSys '07*, pages 286–298, New York, NY, USA, 2007. ACM.
3. F. Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *SIGOPS European Workshop '2000*, pages 37–42, New York, NY, USA, 2000. ACM.
4. BlackBerry Spyware Wasn't Ready for Prime Time. Last visited: 29/09/2009. *http://www.wired.com/threatlevel/2009/07/blackberry-spyware/*.
5. T. L. Cignetti, K. Komarov, and C. S. Ellis. Energy estimation tools for the palm. In *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 96–103, New York, NY, USA, 2000. ACM.
6. H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.
7. D. Kim, J. J. Garcia-Luna-Aceves, K. Obraczka, J.-C. Cano, and P. Manzoni. Routing mechanisms for mobile ad hoc networks based on the energy drain rate. *IEEE Transactions on Mobile Computing*, 2(2):161–173, 2003.
8. K. Lahiri, S. Dey, D. Panigrahi, and A. Raghunathan. Battery-driven system design: A new frontier in low power design. In *ASP-DAC '02*, page 261, Washington, DC, USA, 2002. IEEE Computer Society.
9. L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *MobiCom '08*, pages 58–69, New York, NY, USA, 2008. ACM.
10. R. Neugebauer and D. McAuley. Energy is just another resource: Energy accounting and energy pricing in the nemesis os. In *HOTOS '01*, page 67, Washington, DC, USA, 2001. IEEE Computer Society.
11. T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys '06*, pages 220–232, New York, NY, USA, 2006. ACM.
12. A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: middleware for mobile social networking. In *WOSN '09*, pages 49–54, New York, NY, USA, 2009. ACM.
13. A. Rahmati, A. Qian, and L. Zhong. Understanding human-battery interaction on mobile phones. In *MobileHCI '07*, pages 265–272, New York, NY, USA, 2007. ACM.
14. N. Ravi, J. Scott, L. Han, and L. Iftode. Context-aware battery management for mobile phones. In *PerCom '08*. Citeseer, 2008.
15. Research in Motion. Last visited: 29/09/2009. *http://www.rim.com*.
16. E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *MobiCom '02*, pages 160–171, New York, NY, USA, 2002. ACM.
17. J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton. Haggle: Seamless Networking for Mobile Applications.
18. J. Tarascon and M. Armand. Issues and challenges facing rechargeable lithium batteries. *Nature*, 414(6861):359–367, 2001.