Going Green: An Exploratory Analysis of Energy-Related Questions

Haroon Malik, Peng Zhao, Michael Godfrey
David R. Cheriton School of Computer Science
University of Waterloo, ON, Canada
{hmalik, p29zhao,migod}@uwaterloo.ca

***Abstract* — The popularity of smartphones — small computers that run on battery power — has exploded in the last decade. Unsurprisingly, power consumption is an overarching concern for mobile app developers, who are anxious to learn about power-related problems that are encountered by others. In this paper, we present an empirical study exploring the characteristics of energy-related questions posed in StackOverflow, issues faced by the developers, and the most significantly discussed APIs. We extracted a sample of 5009 StackOverflow questions, and manually analyzed 1000 posts of Android-related energy questions. Our study shows that developers are most concerned about energy-related issues that concern improper implementations, sensor, and radio utilization.**

***Index Terms*—StackOverflow, Energy, Power, API.**

I. Introduction

The last decade has witnessed an explosion in mobile smart phone and tablets — surpassing the 1 billion mark in 2013, and expected to double again by the end of 2015. A mobile phone is no longer just a phone, but a complete online client system that is capable of executing sophisticated applications, from mission-critical business apps to casual apps of everyday life.

Advancement in hardware technology, such as large screens, graphical processing units (GPUs), multi-core processors along with many sensors on mobile smart phones are becoming more energy demanding for their operations. Because users are demanding, mobile apps often offer a near-full scale range of features when compared to their desktop equivalents: If a computer can stream videos over WiFi, why can't my phone also? Of course, mobile apps that require significant processing can quickly deplete battery life, and users are unhappy when they find they can no longer make a phone call part-way through their workday. Thus, mobile app developers must always be keenly away of power consumption, and seek ways to smartly reduce the requirements of their apps.

Although many improvements have increased battery lifetime [1], it has not kept pace with the rapid advancement of smart phone hardware [2]; indeed, the complexity and integration density of digital systems has been growing exponentially. In contrast, the growth of battery technology has been very incremental [3]. For example, Nokia communicator (released in 1996) to Samsung S3 (released in 2012) processing power increased from 24MHz to 1.GHz, whereas the battery capacity has only increased from 800mAh to 2100mAH.

Going green — i.e., reducing energy consumption[1] — has been a secondary concern for much of the last decade; now, it has become one the most important. Traditionally, power consumption issues were typically considered to be in the hands of electrical and computer engineering, and were linked to the pure hardware optimization of the embedded systems.

As the demands of complex application that exploited the direct interaction with hardware — such as GPUs — many sensors locating on smartphones, (e.g., Accelerometer, proximity, magnetometer, pedometer, ambient light and barometer) continue being pushed from users, much of the responsibility to manage the power consumption is being shifted to developers, both application and operating (OS) developers.

Many unsuccessful apps are the target of complaints about battery consumption in market place reviews [4]. These complaints can have serious ramifications for developers, as they can influence users' app purchasing decisions. Energy consumption has therefore become an important quality metric for market applications. However, the challenges faced by developers in designing and developing "greener" applications are not thoroughly documented. Such information is valuable to OS designers to improve their APIs to cater to the energy related challenges faced by developers, who in turn are using their API to build more energy-efficient apps. Most of the work in literature is geared towards helping developers model the energy consumption of applications [5] and attempts to characterize energy use in mobile devices using software repositories (e.g., bugs reports, app store, forums. Among these repositories, Q&A websites, such as Stack Overflow (SO), are invaluable since they provide a comprehensive view of energy-related issues as experienced by users and other developers.

The goal of this paper is to investigate energy-related posts at SO to better understand the issues faced in energy-efficient software development. More specifically, the paper explores the following questions:

RQ 1. *What are the distinct characteristics of energy-related posts on SO?*

RQ 2. *What energy-related issues do the developers discuss?*

RQ 3. *Which APIs are significantly discussed in the energy-related posts?*

[1] Throughout the rest of the paper, we use the term energy, power, and green interchangeably.

Case Study

A. Data Corpus

Our data set is based on an official Stack Overflow data dump (September 26, 2014) [6], which is a representative dataset of developers discussing a wide range of topics in computer programming. We used the entire data corpus to answer both the qualitative and quantitative aspects of *RQ 1*. From this larger corpus, we have selected out the Android-related posts — including textual data and code snippets — due to the manual overhead involved (+11,318 energy related SO posts) in analyzing posts at a fine granularity. We chose to concentrate on the 'Android' project at SO for three main reasons: first, 57% of all smartphones/tablets are Android based; second, Android applications have received strong complaints from users because of energy consumption problems; and third, the Android platform was released in 2007 — one year before the launch of SO — so we can capture a relatively complete history of green questions and answers.

B. Post-processing

Using the SO dump, we extracted questions, answers, and other metadata and stored it in a relational database. To achieve a wide and comprehensive coverage of energy-related posts on SO, we used a threefold filtering approach:

First Fold: We first applied a simple regex on the text of the posts to match energy related keywords — such as found in the work of previous researchers [7] — resulting in 972 questions and 1700 answers.

Second Fold: We manually analyzed a random sampling of 500 questions found in the first fold, and extracted more related keywords and added these to the keywords found during the related literature review. The keyword search resulted into 11,318 green posts (5009 questions and 6309 answers).

Third Fold: We filtered the energy related posts based on the *Android* tag, resulting into 2479 green posts. Our search string is as follows*: *energy consum*,* energy efficien*, *energy sav*, *save energy*,*power consum*,*power efficien*,*power sa*,*save power*,*high CPU*, *power aware*,*drain*, *no sleep*,*battery life*,*battery consum*.*

II. Case Study Findings

RQ 1. *What are the distinct characteristics of energy-related posts at SO?*

Motivation*:* To gain an understanding of how energy-related questions compare to other questions at SO.

Approach: This research question is a partial replication of work by Pinto et al. [7]. Pinto explored only a few of the quantitative aspects of green questions at SO at a higher granularity. In contrast, a) our work is based on a larger set of green questions that we obtained by using more keywords in order to get a wider coverage of energy related posts at SO, and b) we report our findings along both the quantitative and qualitative aspect of Green posts at SO, and at a finer granularity, i.e., we grouped green posts into three categories. *i*) *Explicit questions* — energy consumption is the primary co-

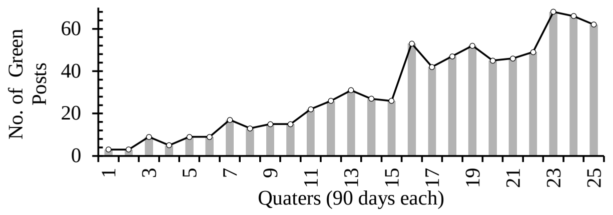

Figure 1: Evolution of Green Posts

ncern expressed in the question threads[2], *ii) Implicit questions* — No upfront ties to energy consumption are exposed and energy consumption is expressed as the secondary goal of the question thread[3], and *iii) Noise* — False positive posts that contains the search keywords, but the discussion is not related to energy[4].

Findings: We analyzed several *quantitative aspects* of the implicit and explicit green questions, and compared them with those of non-green questions**.**

- *Green questions are implicit*. We find 50% of the threads to be implicit, 35% to be explicit and 15% to be noise.
- *Green posts makeup less than 1% for the SO posts.* There are 11,318 green posts (5009 questions and 6309 answers), accounting for lest then *1%* of the total SO posts. Android has 2479 green posts (1071 questions and 1462 answers), while iOS has 1725 (732 questions and 993 answers), which makes up 21.9% and 15.2% of the total posts for the respective projects.
- *Green questions at StackOverflow have a linear growth* — Figure 1 shows the number of questions at SO, since the first one appeared in 2008. The chart shows a vacillating trajectory. By simply observing the figure, it cannot be inferred whether the vacillation is an act of pure randomness, or if there exists an underlying trend driving the vacillation. To study the trending, we used the *MannKendall* test, a non-parametric test, for trend analysis [8] since the underlying data is not normally distributed. At a significance level of 1%, i.e., 0.01, ρ-value of 0.00170 indicates that the null hypothesis (H_0) (i.e., *there is no trend in the data distribution of the number of energy related posts.*) is rejected. The positive value of MannKendall's test statistic τ in our case is 0.648, which indicates an upward (i.e., increasing) trend of Green posts at SO.
- *Low answer rate* — 85% of green questions are answered and the rate of accepted answer is 52%. Whereas, 88% of questions at SO are answered, i.e., received at least one answer, and 58% of the questions have an accepted answer.
- *Answers are slow* — Median time to get the first answer at SO is 11 min. In that respect, green questions receive their first answer late. Median time to get first answer for green Android questions is 13 days. For Android non-green questions, the median time to first answer is 13 days. For iOS, the median is 25 days. Whereas, the median time to

[2] http://stackoverflow.com/questions/6019996/
[3] http://stackoverflow.com/questions/7807610/
[4] http://stackoverflow.com/questions/13984263/

first answers for non-green iOS questions is less than 10 days. Green questions, for which the first answer arrive extremely late are either ill-posted or are based on a cutting edge idea, to which an explicit solution or workaround do not exist, e.g. an answer received after 304 days *(*post id: 13403877— *"A brilliant question, sadly though I don't think that would be possible….."*).

We manually analyzed Android green posts (300 Questions, 700+ answers and respective comments) and found that *qualitatively*, green questions fall into three major categories:

1. ***Newbie Questions:*** 19% of the green posts are newbie Questions, i.e., How-to questions that are mostly from self-confessed novice users[5] with low reputation at SO. Many of the green newbie questions do not have an accepted answer. Declaring an answer as 'accepted' is a voluntary act. A number of these novice members do not formally mark as 'accepted' one of the answers to their questions since these are information seekers who seek a one-time answer to a specific programming question, instead of community building for SO.
2. ***Implementation Specific:*** 52% of questions are implementation specific. The developer posting the question has knowledge about an energy-related problem, and has a potential solution ready. However, the solution does not work as expected, and the developer did trouble-shooting on his part, but is not sure of the root cause of the problem. Most of the answers point to developers not understanding the API appropriately, and many questions are the result of the Heisenberg effect e.g., *"No, it was gone in one of the next builds, but I have no idea why"*. 80% of such threads have code snippets.
3. ***Discussion triggers:*** 29% of the questions are discussion triggers. For these questions, there do not exist a specific solution; indeed, SO members are often providing a workaround to tackle the issue. Members often challenge each other's solutions, e.g., *"@Ebomike: "you need to make it clear to the user that your app will drink the phone's milkshake while it's off." "I do understand your concern. Please see my Edit to the question above"*.

RQ 2. *What energy-related issues do developers discuss?*

Motivation: The simplest way to develop an energy-efficient app is to understand the common issues faced by other developers, and avoid falling into known traps. Developers may lack guidance on how energy is typically consumed by mobile applications. Currently, there are no authoritative criteria for designing energy efficient apps. Identifying the most commonly discussed issues will act as anti-patterns to recurring energy related-problem discussed among mobile developers, and can help them avoid introducing potential energy bugs into their apps.

Approach: We randomly sampled 300 Android green question threads, and performed a manual analysis.

Findings: We summarized energy-related issues into four main categories, comprising 12 sub-categories.

[5] http://stackoverflow.com/questions/5816938/ & http://stackoverflow.com/questions/19712667/

- ***Inappropriate Implementations: Excessive Update, Service Trigger, Messages Transfer, Keep Screen On***

Inappropriate implementations are those cases where a developer does not understand the fundamentals of energy efficient application development. We found that implementations such as: '*updating data every second', 'send a web request every 5 mins through service', 'have activity doing some work when the user is not using their application, i.e., not releasing wakelock'*, and *'keeping screen on with a background process'* are discussed the most. The proposed solutions include efficient implementations such as '*Update data occasionally*', and using '*AlarmManager*'.

- ***Sensors Utilization*: *Location, Motion***

One of the root causes of energy waste is associated with the ineffective use of sensors (e.g., gyroscope, accelerometers). We have found that sensor listener misuse is common, and often discussed among developers. To use a sensor, an application needs to register a sensor listener with the Android OS, and specify its sensing rate. Defining the sensing dynamics is not simple and depends on many factors, such as the purpose of the app and the location it is used. For example, we found some developers who suggest tracking location with *Location Manager* at predefined distance intervals in order to reduce energy consumption. Some developers have suggested that *Location Client* (recently released) would be more energy efficient. We also noticed complaints from developers that the accelerometer and gravity sensors consume a lot of energy without good accuracy in comparison to other sensors. We found that motion sensors (accelerometers and gyroscopes) are discussed the most in green posts, i.e., 12 occurrence, followed by position sensors (orientation and magnetometer), 19 occurrence.

- ***Radio Utilization: Wi-Fi, Bluetooth, GPRS/3G***

Ineffective utilization of radio is a big cause of battery drain. Among all the discussion related to radio, Bluetooth is discussed most among developer. '*Reducing discovery time'*, *'managing concurrent connections' and 'low energy pairing'* are most common issues faced by developers. Methods to calculate Wi-Fi power consumption were posted numerous times, with no accepted answer. Wi-Fi power management remains the second most discussed issues by developer in this category, with the emphasis being on the related '*PowerManager*' API.

- ***Data Manipulation: Big Data Transfer, SD card***

Storing a large amount of data into SQLite and Files consumes a lot of energy, since limited data can be transferred each time. Therefore, developers discuss solution of storage with **fewer loops**. Reading and writing into SD cards have more energy issues than internal storage.

RQ 3. *Which APIs are significantly discussed in the energy related posts?*

Motivation: To enable the development of rich applications with fine grained hardware and power management control, platform provide developers with a set of APIs (e.g., an API for controlling screen brightness, sleep, and a power manager API and to provide improved visibility into system power use).

TABLE I: SIGNIFICANTLY DISCUSSED APIS IN THE GREEN POSTS

	Green		Random	
APIs	Freq	View	Freq	View
int	29.59	368581	24.47	333063
java.lang.String	26.62	331803	25.59	189695
android.content.Intent	16.33	281203	7.66	221445
android.content.Contex	12.51	147480	3.71	82644
android.app.Activity	10.50	345152	7.79	67382
boolean	9.54	78147	3.71	144577
long	8.80	62409	3.46	92657
android.location.LocationManager	8.70	117077	0.87	70830
java.lang.Thread	8.17	128385	2.47	25824
java.lang.Exception	7.95	74645	4.57	18423
android.app.PendingIntent	7.74	74582	1.36	56583
android.os.Bundle	7.74	73328	8.03	29285
android.content.BroadcastReceiver	7.00	35254	0.37	11568
android.location.Location	6.47	83832	0.25	122959
java.lang.System	6.15	55385	1.98	63925
android.app.AlarmManager	6.04	51601	0.49	20591
android.os.PowerManager	6.04	80859	0.12	21337
android.view.View	5.94	64466	10.38	65128
java.lang.Runnable	5.73	33029	2.72	5084
android.widget.Toast	5.30	112517	2.72	41975
android.location.LocationListener	4.88	70440	0.12	17417
android.os.PowerManager$WakeLock	4.03	48371	0.12	13749

These APIs provide methods that are designed in a one-size-fits-all fashion — if not used carefully; they can introduce energy bugs that may drain a phone's battery in as little as five hours. Furthermore, framework designers may fail to predict how their APIs will be used in scenarios that lie outside of the most common task. Identifying the most significantly discussed APIs in energy related posts can help API designers identify problematic APIs and uncover and rectify the underlying usability issues associated with them.

Approach: We used a static analysis tool called *Baker* [9] to find the fully qualified names of Java classes and types (hereafter referred to as APIs) used in code snippets. Only results containing a single API (*Baker* returns several suggestions if it cannot be certain) were considered as this allows for a high precision of 0.98 while still having a recall of 0.83. We parsed all code snippets in the 2479 green questions threads, that is, threads that contain a keyword in either the question or at least one of the answers. A random set of 1692 Android question threads was also parsed, to serve as a baseline for comparison. This comparison is required to filter out APIs that are significantly discussed in the green-thread set because they are also the most commonly discussed overall, rather than being related to energy consumption problems. Next, we ranked the APIs (TABLE 1.) based on the percentage of perfectly parse-able green posts they occur in. A perfectly parse-able post is one that *Baker* could find at least one exact fully qualified name in. There were 943 such posts in the green set and 809 in the random set. We included the sum of the views of API-containing threads in the table because the frequency of occurrence of an API may not yield a fair estimate of how popular it is. For example, an answer to a single question may solve a problem with a given API for many users, who will then not post a question about the API. Their interest in the API, however, is recorded in the thread's view count.

Findings: TABLE I lists the 23 Android APIs that are discussed most in green threads. The top two APIs are commonly used throughout Android applications as can be seen from the large percentage of green and random threads that contain them with high view count. Lower down in the list, in bold, are APIs such as *BroadcastReceiver* and *AlarmManager* that occur much more frequently in the green set with more view counts than the random set, an indication that they are likely related to energy consumption issues. The APIs whose prevalence differs most between the two sets are *PowerManager* and *LocationListener*, respectively occurring 97% and 98% less frequently in the random set.

III. THREATS TO VALIDITY AND CONCLUSIONS

External Validity: Our study focused on a single website, Stack Overflow. Also, our findings concern the Android project and cannot necessarily be generalized to other projects, e.g., iOS.

Construct Validity: We tried to carefully construct the keywords used in our search so as to maximize the recall and precision of our query. There are, however, a number of false positives, i.e., posts that are no truly green, and an unknown number of false negatives, i.e., posts that are green but were not found by our query. The manual inspection of posts by the first three authors is also subject to error and bias.

In this paper we studied the characteristics of the energy related discussions at StackOveflow. We observed and reported energy related issues that developer discusses the most. We also investigated the most-used android APIs that occur in discussion posts. In future, we want to extend our findings to iOS and compare with that of Android.

IV. REFERENCES

[1] M. Armand and J. Tarascon, "Building better batteries," Nature, vol. 451, pp. 652-657, 2008.

[2] J. Frenkil, "Tools and methodologies for low power design," in Proceedings of the 34th annual Design Automation Conference, pp. 76-81, 1997.

[3] J. Eager, "Advances in rechargeable batteries spark product innovation," in ELECTRO-IEEE CONFERENCE-, pp. 686-686, 1992.

[4] 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, pp. 29-42, 2012.

[5] S. Hao, D. Li, W.G. Halfond and R. Govindan, "Estimating Android applications' CPU energy usage via bytecode profiling," in Proceedings of the First International Workshop on Green and Sustainable Software, pp. 1-7, 2012.

[6] Annie Ying, "Mining Challenge 2015: Comparing and combining different information sources on the Stack Overflow data set," in The 12th Working Conference on Mining Software Repositories, 2015.

[7] G. Pinto, F. Castor and Y.D. Liu, "Mining questions about software energy consumption," in Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 22-31, 2014.

[8] C. Wilke, S. Richly, S. Gotz, C. Piechnick and U. Aßmann, "Energy consumption and efficiency in mobile applications: A user feedback study," in Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp. 134-141, 2013.

[9] S. Subramanian, L. Inozemtseva and R. Holmes, "Live API documentation," in Proceedings of the 36th International Conference on Software Engineering, pp. 643-652, 2014.