Energy Consumption on Android

TIMO TIIRATS, Institute of computer science, University of Tartu, Estonia

Energy-efficiency in mobile apps plays an important part in today’s life as all mobile devices are battery powered. This increases the need to develop more sustainable applications to fulfill the customer needs, while increasing battery life. As smartphone processing power is continuously growing, it creates a dangerous moment, where developers could write more energy-inefficiency code, which increases the energy consumption in the background.

This paper gives an overview of different energy consumption measurement approaches used in Android OS. They can be divided to direct and indirect measurements with each having its own subcategories. The paper includes examples of frameworks using different approaches including description between fine- and coarse-grained models. Based on empirical study comparison between direct and indirect measuring approaches is conducted. It is found that indirect measuring approaches still largely depend on direct measurements for calibration, which decreases their usability and advantages.

1 INTRODUCTION

Nowadays there are a lot of battery powered smart devices used on daily basis. Starting from activity trackers and smartwatches until most advanced smartphones and tablets. As these all are working by using limited battery power, then there is a need to evaluate and decrease the energy consumption on each device, in order to extend the battery life. One possibility to decrease the energy consumption is to use more power-saving electronics, but using better electronics does not matter if the software using it causes unnecessary energy drain. In example, when application is moved to background task, but it does not release any computing power to the OS and will run indefinitely. Therefore, it is important to have energy consumption measuring tools or frameworks to evaluate the energy usage of certain applications, which can be done already in the application development stage.

This report is based on Android operating system devices, because Android OS is the most popular open-source mobile platform operating system [StatCounter 2022]. This means that the methods described in this paper can be used to test applications on a platform that has a lot of users and can benefit most of them. The overview is focusing on comparing two different energy consumption measuring approaches. The first one being direct measurements, that measures the energy consumption using built-in or external power meters. It means that the direct measurement approach outputs the exact energy consumption the phone is using at the moment. The second one is indirect measurements, that estimates the energy consumption with analyzing the CPU tasks it’s performing or making code analysis. For making the estimations, it needs calibration or reference data beforehand as it needs something as a base to predict the energy consumption. The report answers the following research questions:

- RQ1: Which challenges limit indirect measurement methods compared to direct measurement methods?
- RQ2: Do the best indirect methods rely on large historic datasets?

To answer these research questions I will study papers describing previously made frameworks. Some of them describing one concrete direct or indirect measurement approach, while others make overview of multiple frameworks. In total 61 different frameworks are described in the papers and are used to write this report.

The rest of the paper is organized as follows. Section 2 will give a brief overview about other researches conducted in the same area. It is followed by measuring approaches in section 3, which will describe different methods, how it is possible to measure Android OS applications energy consumption. Section 4 contains discussion related to raised research questions, followed by conclusion in section 5.

2 RELATED WORK

One of the main articles used in this paper contains a Systematic Literature Review (SLR), which was published in 2020 by [Myasnikov et al. 2020]. Their main focus was to find, categorize and compare existing frameworks or solutions to measure energy consumption on Android OS. To find articles they conducted a search in Google Scholar with the following keywords: ”android”, ”energy-efficiency” and ”framework”. From a total of 931 studies, they finally used 45 articles plus 3 manually searched papers about 41 different frameworks.

Regarding this paper a search in Scopus was conducted with the same keywords that were used in the SLR article. Based on that, 7 articles were chosen, which described some new framework or tool that can be used to measure energy consumption in Android OS and were not concluded in the SLR article. Articles only newer than 2016 were considered. Most of them use code or Android OS Application Programming Interface call analyzing techniques to predict the energy consumption, for example [Liu et al. 2018] and [Banerjee et al. 2016]. In addition to code analyzing and problem reporting, two of them [Banerjee et al. 2018] and [Banerjee and Roychoudhury 2016] also support automated code repair functionality. The Navitas framework proposed in [Myasnikov et al. 2021] is from the same authors as the SLR, but they decided to measure the energy consumption based on CPU usage with additionally considering screen on-time. Framework described in [Li et al. 2020] only was
able to use an external power meter to measure the power trace of the application to determine the energy consumption. Lastly, Android Runner proposed in [Malavolta et al. 2020] can be used with external or internal power meters and is highly customizable for each user and test case.

Finally, [Khan et al. 2021] is also a paper that includes multiple different frameworks and measuring methods, published in 2021. Compared to SLR, which includes empirical study with seven research questions (e.g. which units are used in different frameworks etc), it focuses more on comparing the actual possibilities or limitations and accuracy of different frameworks. In total, they tested 21 different profiling tools available on the market. Some of the frameworks are also included in SLR articles.

3 MEASURING APPROACHES

This section will give an overview of different energy consumption measuring options used in Android OS. They can be divided into two categories based on the logic they use to find the energy consumption of an application. The first type can be named as a direct measurement method. It can use external or internal power meters and based on that it is categorized as a hardware- or software-based measuring tool. Hardware-based measuring tools use external power meters and software-based tools use integrated sensors and mathematical models to estimate the power consumption. External power meters can be very precise and therefore are used as ground truth for other software-based tools, but may require external hardware components and complex setups that can be costly to the user. Software-based measuring tools are easier to use and provide more details on the application power consumption. The second way to measure energy consumption of Android applications is to use indirect measurement methods, where the energy consumption is estimated based on a mathematical model. They also classify under software-based measuring tools. [Khan et al. 2021; Myasnikov et al. 2020]

The measured energy consumption can be represented in two granularity levels. If the tool outputs the total energy consumption of an application while running some specific task or test, then it is specified as a coarse-grained measurement tool. If the tool outputs the energy consumption for each function or instruction, then it is named a fine-grained measurement tool. [Farooq et al. 2019] Using fine-grained measurement tools can be useful to app developers for troubleshooting and testing the application power consumption in the development phase. The fine-grained methods give the possibility to develop energy-sustainable applications. Coarse-grained measurement tool gives the overall energy consumption of the application so it is less informative as which is the most power consuming part. Mostly used to help users or application testers to measure the total energy consumption of the application. [Khan et al. 2021]

Energy consumption is represented in different units. The most common units to measure the energy consumption in Android OS when evaluating the application are joules (J) and wattage (W). Few frameworks use voltage (V), amperes (A) and ampere-hours (Ah). With direct measurements the most informative is to show the energy consumption in joules as consumed energy. To show the results in some other units there is a need to make some assumptions or fix other variables. For example, to show energy consumption in ampere-hours the voltage should be fixed to get correct measurements to overcome battery voltage drop while discharging. To get the right measurement using wattage the period of the measurements needs to be periodic and distributed equally. [Myasnikov et al. 2020, 2021]

3.1 Direct measurements

**External** power meter can be multimeter or power supply with energy consumption measuring option [Myasnikov et al. 2020]. The most commonly used power meter is Monsoon Power Meter [Khan et al. 2021]. When using external power meter there is an option to use power supply instead of battery to have a constant voltage source. Frameworks using external power meters collect all the power consumption data during the test and later linearly interpolate over the measurements to get coarse-grained results. It is possible to get fine-grained results as well, but this assumes additional logging instructions in the application code while saving power trace and application execution trace at the same time. In addition, a synchronization between phone and measuring tool is needed to align power trace with application trace. This gives the opportunity to evaluate the energy usage more precisely, but at the same times add some additional instructions to the code, which have to be accounted for. [Myasnikov et al. 2020] For example, framework described in [Li et al. 2020] uses external power meter to find the energy issues in applications. The framework inspect different parts of the power trace to discover energy issues. It does not exactly pinpoint the line of code where the energy issue is rather gives the developer the user interaction sequence (e.g. button presses or swipes done in application) and environment variables (e.g. weak Wi-Fi signal), to help the developer to reproduce and study the reasons for the discovered energy issue. [Li et al. 2020]

**Internal** power meters are built-in to the phone and can be accessed through Android OS API calls1. With internal power meters there is not a need to synchronize the power meter and application code execution as they are using the same system clock. Using an internal power meter gives the possibility to evaluate single app energy consumption and exclude everything else. [Myasnikov et al. 2020]
Android Runner (AR) framework supports using both external or internal power meters. For external power meters it uses Monsoon Power Meter and for internal Trepn Power Profiler [Qualcomm Technologies 2022] framework, which is limited to devices that use Qualcomm processors. It is a framework to automatically carry out multiple measurement based experiments on Android devices with support for native and web based applications. The framework is highly customizable with different user writable scripts with open-source code repository in GitHub. Android Runner framework can be run on users computer or other device that run python code connecting to phone via USB or Wi-Fi. [Malavolta et al. 2020; Myasnikov et al. 2020]

3.2 Indirect measurements

Indirect measurement methods are often named as model-based methods, as they use previously trained mathematical models to estimate the code energy consumption based on the information gathered from code execution [Khan et al. 2021; Myasnikov et al. 2021]. Their usage is divided into two steps. Firstly, model calibration or training is done and in the second phase the energy consumption can be estimated. [Myasnikov et al. 2020] Model calibration is usually done using external direct measuring methods [Khan et al. 2021]. The calibration phase must be done when using different phone models for testing, but also when changing devices in the same model range as well [Hindle et al. 2014].

Indirect measurement methods could be divided to multiple sub-models based on how they estimate the energy consumption:

Working time model method sums all the working time each hardware component, e.g. GPS-module and CPU, in a smartphone works during the execution of a testable application. During the calibration phase information about how much energy each individual component is using in time is gathered. In addition, energy consumption for each different operation (e.g. Wi-Fi module searching or uploading) the component is saved separately, both in calibration and testing phase, as the energy consumption is not constant between those operations. [Myasnikov et al. 2020] Total energy consumed is calculated with the following Equation 1:

\[ E = \sum_{i=1}^{N_de} \sum_{j=1}^{NP_i} P_{ij} \cdot t_{ij} \]

where \( E \) is total energy consumed, \( N_{de} \) shows the number of different components used in the experiment, \( NP_i \) is number of different operations that the \( i \)th component does, \( P_{ij} \) is the \( j \)th power trace of the \( i \)th component and \( t_{ij} \) is the working time for \( i \)th component with \( P_{ij} \) power trace [Myasnikov et al. 2020]. This model gives the developer coarse-grained results as linear regression is used to get energy consumption estimates. There is a possibility to get energy consumption per smartphone component, but not the exact line of code or function that uses a certain amount of energy.

The Navitas framework published in [Myasnikov et al. 2021] uses a working time based model to estimate the energy consumption, which is also available as an open-source plugin to Android Studio. They are more focusing on estimating the energy consumed by the CPU while working on multi-threaded tasks. They claim that direct measurement based frameworks can be easily extended to multi-threaded use cases, as they already give out the total power usage of a phone or application. Before them there were not any frameworks that estimate the current consumption on multi-threaded applications. [Myasnikov et al. 2021]

**Instruction energy model** evaluates the energy consumption based on the count of various code instruction types using weighted sum. They can be loops, floating-point operations, conditional statements etc. This means that the code analysis can be done in any platform and it does not require the Android OS device as long as no specific Android API is used. In the model calibration phase each instruction energy consumption will be measured using internal or external power meters. [Hao et al. 2012; Myasnikov et al. 2020] The total energy consumption is calculated with the following Equation 2:

\[ E = \sum_{i=1}^{N_{Instr}} P_i \cdot n_i \]

where \( E \) is the total energy, \( N_{Instr} \) is the number of different instructions used, \( P_i \) is the energy consumption of a single instruction and \( n_i \) is the number of times the instruction is used in the code [Myasnikov et al. 2020].

Instructional energy model based framework can be used with both coarse- and fine-grained level of granularity. They can output the energy consumption for the whole application, per method or per line in a source code. [Khan et al. 2021]

**Method/API call models** use a similar concept that instruction based energy models use, but these mathematical models assume that most of the energy is consumed outside of users’ written code. Instead, most of the energy consumption is happening using Android OS API. As with all mentioned indirect measurement based models it needs calibration, which finds the energy consumption each API call and uses this information, when assessing the energy consumption of application. [Myasnikov et al. 2020] In addition, the energy consumption could be measured by analyzing the source code and gathering all the power states in which different smartphone components (e.g. GPS, Wi-Fi etc) go through when using the application. Based on that necessary modifications could be done to eliminate unnecessary workloads and states. [Le et al. 2018]

**History based models** use previously found combinations or flaws to measure the energy consumption or energy issues in the applications. Systematic Literature Review conducted by [Myasnikov...
et al. 2020] analyzed 41 different frameworks and none of them used historic datasets. Study made by Khan et al. [Khan et al. 2021] analyzed a total of 21 energy consumption measuring frameworks from which 13 were not included in the SLR article. None of the 13 used historic data to predict energy consumption. Both article authors clearly state that the first stage of using mathematical models includes new experiments or comparing and training the model to reference data. [Khan et al. 2021; Myasnikov et al. 2020] It can be concluded that using historical data to predict energy consumption on Android OS is quite rare.

Old experiment data is used in [Banerjee et al. 2018, 2016; Banerjee and Roychoudhury 2016] frameworks, where authors state that they are using Android API call patterns found in their previous work [Banerjee et al. 2014] to identify potential energy-inefficient parts of code. They are not focusing on evaluating energy consumption directly, only finding the potential places that may cause energy inefficiency [Banerjee et al. 2016]. In addition to static analyzing, dynamic code analyzing (running generated test cases) could be used to find and validate the inefficient parts as proposed in [Banerjee et al. 2018]. As they can use predetermined patterns it makes it easier to use historic datasets as they only search for certain parts in the code not evaluating the energy usage that can significantly vary between devices.

4 DISCUSSION

This section includes the discussion about raised research questions and is based on the empirical study done in the measuring approaches section.

4.1 RQ1: Which challenges limit indirect measurement methods compared to direct measurement methods?

- **Calibration phase.** To use any of the proposed indirect energy consumption measuring methods, there is a need for a separate stage to calibrate or train the model. It is suggested to do the model calibration even when switching between phones with same models to get the most accurate results. The calibration could be done with experiments or with reference data to determine and tune the model parameters. Using historic datasets for calibration is possible, but it cannot give as accurate results as with experiments for each individual device. If experiments are used, then there still is a need for external hardware that could be complicated and expensive to use or buy.

- **Less Accurate.** Indirect methods are less accurate compared to direct methods as they use some mathematical model to estimate the energy consumption based on previous data while direct methods directly measure the energy used currently. Application total energy consumption may vary compared to measuring each API or instruction call energy consumption separately, but then combining them to single application together.

- **Individual for each device.** The mathematical model used on each test case is different, when using different smartphones, even when they are the same smartphone model. This creates additional workload as multiple calibrations are needed to do, before the model is able to predict the energy usage. If using the same mathematical model for different phone models, then the developer risks getting inaccurate results.

- **Different models.** As seen from the measuring approaches section, then there are a lot of different options to evaluate the energy usage with indirect methods. They all have done their assumptions on the code, for example API call method assumes that API calls take much more energy than user code or instructional model assumes that most energy can be calculated using user source code instructional statistics. That also shows there is not a very good solution that takes all the necessary components into account, but as direct methods measure the whole consumption of the device it is able to cope with all situations.

4.2 RQ2: Do the best indirect methods rely on large historic datasets?

Best indirect methods do not use historic datasets to estimate the application energy consumption. Using historic datasets to evaluate the energy consumption is quite rare among the indirect methods, most of them use calibration instead. It may come from the fact that each device and application has its own unique properties and is harder or inaccurate to use historic datasets to predict the energy consumption. There are frameworks that search for energy inefficient parts of the application source code using code patterns. Patterns are also gathered from evaluating applications with calibrated models or with direct measurement methods. Discovering only energy inefficient parts of the source code may give less information to developers, for example when evaluating the total energy usage between two app releases. In addition, there are frameworks, in example [Li et al. 2020], that use machine learning for finding energy inefficient parts of application code using smartphone power traces. They also have to use different machine learning models for each individual application, which means that they have to train the model again for each application. [Li et al. 2020] This again confirms that it is hard to use different mathematical models or historic data to evaluate the energy consumption of a new app.
5 CONCLUSION

In this paper, we had an overview of different energy consumption measuring approaches. They were divided into two large categories - direct and indirect measurements, each with their own subcategories. In addition, a description about framework energy consumption estimation granularity was given. For empirical study, multiple articles containing comparison between different frameworks was used, which helps to increase the scope of this work as it has information about a large quantity of frameworks. The articles were described in the related work section. Finally, there was discussion about research questions raised in the introduction section. I pointed out some limitations when using indirect methods compared to direct methods and best indirect methods still need calibration or reference data to work accurately. In addition, there is not a good indirect method that takes all the necessary components into account when evaluating the application power consumption, but as direct methods measure the whole power consumption of the device they are able cope with all situations.

ACKNOWLEDGMENTS

Paper author would like to thank his supervisor Jakob Mass, University of Tartu, for valuable information and guidelines throughout the course.

REFERENCES


