EMaaS: Energy Measurements as a Service for Mobile Applications

Luis Cruz
University of Porto
INESC-ID
Porto, Portugal
luiscruz@fe.up.pt

Rui Abreu
Instituto Superior Técnico, University of Lisbon
INESC-ID
Lisbon, Portugal
rui@computer.org

***Abstract*—Measuring energy consumption is a challenging task faced by developers when building mobile apps. This paper presents EMaaS: a system that provides reliable energy measurements for mobile applications, without requiring a complex setup. It combines estimations from an energy model with — typically more reliable, but also expensive — hardware-based measurements. On a per scenario basis, it decides whether the energy model is able to provide a reliable estimation of energy consumption. Otherwise, hardware-based measurements are provided. In addition, the system is accessible to the community of mobile software practitioners/researchers in the form of a Software as a Service. With this service, we aim at solving current problems in the field of energy efficiency in mobile software engineering: the complexity of hardware-based power monitor tools, the reliability of energy models, and the continuous need of data to build energy models.**

***Index Terms*—Mobile Applications; Mobile Testing; Energy Consumption.**

I. Introduction

One of the main problems mobile developers face when building apps is the need to run tests under a number of different settings (e.g., mobile device models, operative system version) [1, 2]. For instance, the appearance of the user interface may change in devices with different screen sizes and resolutions. This problem is also evident when testing energy consumption.

Moreover, measuring energy consumption with power tools is a cumbersome task. Developers need to create a setup that often requires disassembling the device and acquire specialized power tools, as shown in Fig. 1. Such setup is expensive, time-consuming and requires skills out of the domain of most practitioners. Nevertheless, this setup ends up being unused most of the time, since developers do not need to run energy measurements continuously during development.

To overcome these issues, tools have been proposed to measure energy consumption using software-based estimators [3–6]. These tools are able to provide accurate estimations within the context in which they were trained [7]. However, when they are used against new settings — for instance, different devices, operative system (OS) versions, or API methods — one cannot be sure about the reliability of those measurements. Thus, there is a need to continuously collect energy measurement data in order to have accurate energy models.

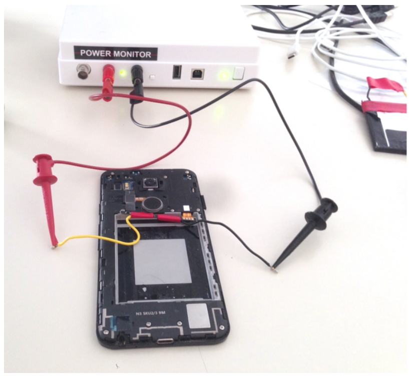


Fig. 1. *Monsoon* power monitor connected to the Nexus 5X smartphone.

Although energy models are very handy, practitioners cannot be sure of whether a model is trained for a given use case. A developer should not rely on a model that was trained for different scenarios. However, this information is not always accessible to developers. This is critical when developers need to make design decisions based on the energy efficiency of their code.

There is a tradeoff between using hardware and software-based energy estimators. None of these approaches is ready to be adopted by the community of mobile developers on a global scale. In this paper, we address the aforementioned issues by proposing a hybrid system that provides the best of these two approaches. The system provides energy measurements as a service and is able to switch between software and hardware-based measurements, depending on the app under test and its running environment.

In this paper, we propose EMaaS, a peer-to-peer cloud-based system that delivers energy measurements as a service for mobile applications. In particular, the system addresses the following issues:

1) Power monitor tools are complex and impractical in a real mobile development scenario.
2) Reliable energy models need to be continuously updated with new data, collected using hardware-based power monitors.
3) The context in which a given energy model can provide reliable measurements is not always clear.

II. The Vision

We envisage a crowd-sourced system to deliver lightweight energy tests as a service. It combines energy models with power monitors to provide the most accurate energy measurements without requiring a cumbersome setup of power tools. Moreover, the system allows developers to measure their apps in mobile devices from different manufacturers.

Developers only need to provide an executable package with (1) the application build (e.g. an *APK*[1] in the case of the Android OS) and (2) the instrumentation build with the test cases to be measured (e.g., the instrumentation *APK* for *Espresso* test cases on Android).

Fig. 2 outlines the high-level architecture of the system. It features three types of users:

- **Developers** use the system to collect energy consumption measurements for their apps.
- **Providers** allow other users to use their mobile devices to run energy measurements. In this case, energy consumption is estimated using an energy model.
- **Super-Providers** are special **providers** that provide energy measurements using a hardware-based power monitor.

The same user can act as a developer, provider, or super-provider, simultaneously. When super-providers are not available to run a given measurement, providers take their place using energy models.

As an example, Fig. 2 highlights three connected peers, marked with green thick lines: one developer, one provider, and one super-provider. The same developer asks the system to run energy tests in two different device models, X and Y. The system assigns a super-provider for device model X but no super-provider was available for device Y. Although a super-provider for Y exists in the system, it is busy dealing with another measurement task. Thus, since the system had a reliable energy estimator for device model Y, it assigned the task to a provider, as depicted in the illustration. If the system did not have a suitable energy model for the given scenario, the task would wait until a super-provider for model Y would become available.

In this setting, the developer does not own any mobile device or power monitor tool. They are being provided by other users in the system. With this approach, it delivers 1) a better use of resources, 2) access to a bigger set of devices and tools, and 3) a simple/affordable setup to measure energy consumption.

Under the hood, the system integrates two modules to bring the best of both hardware and software-based power monitors: the **Energy Model** and the **Reliability Consultant**.

The **Energy Model** is based on approaches from previous work, which uses data collected from hardware-based power monitors to train models of power consumption. The main limitation of current state-of-the-art solutions is the lack of data to train energy models [5]. We mitigate this limitation by continuously updating models using data collected from hardware-based power monitors (i.e., super-providers). Thus, in this solution, providers always estimate energy consumption using up-to-date energy models.

[1]APK is the package file format used by the Android OS for distribution and installation of mobile apps and middleware.

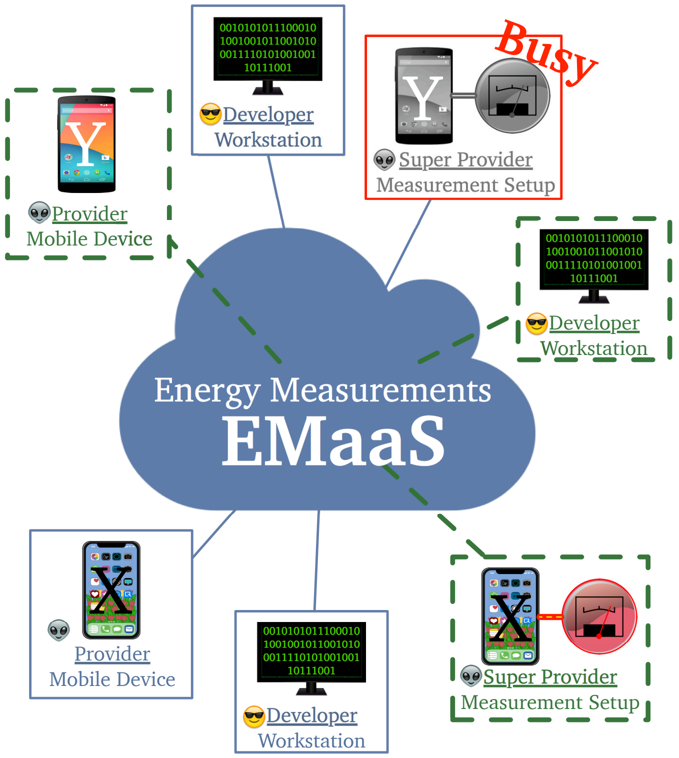


Fig. 2. High-level vision of the system for energy measurements as a service.

Another limitation is the fact that these solutions rely on in-vitro data to train their models. Extrapolating these estimators to different devices can compromise measurements. On contrary, this solution can virtually scope any execution environment, given that it is available from a **super-provider**. In addition, since the power model is continuously improving every time developers run their energy tests, it is able to adapt to new paradigms, devices, or OS versions when they come to market. Going back to the example given by Fig. 2, if a reliable energy model for the given test cases was not available for device Y, the system would collect data from super-providers over time. Eventually, the system would have enough data to learn a new energy model that could provide reliable estimations for the new scenario.

Although energy models can provide very accurate estimations, one needs to assess whether they are ready to be used in a given context. I.e., a given energy model might not be ready yet to measure the energy consumption of apps that use specific libraries. The system needs to assess whether, for a given developer request, it is acceptable to return an energy estimation. If not, the system has to alert the developer or wait for a super-provider to be available. We propose the module **Reliability Consultant** to solve this problem.

In parallel with the **Energy Model**, the **Reliability Consultant** module will inspect the reliability of the estimator. Super-providers will run simultaneously hardware and software-based measurements. The reliability of energy models is assessed using results from hardware as ground-truth. We then construct reliability as a metric that is negatively correlated to the power error (ϵ), given by the following equation:

$$\epsilon = \frac{E_{measured} - E_{estimated}}{\Delta t} \quad (1)$$

where $E_{measured}$ and $E_{estimated}$ are the energy consumption measured by the power monitor and the estimator, respectively, and Δt is the duration of execution of the energy test. For high reliability, ϵ should be close to zero.

In addition, we use static analysis to collect data regarding the app and its execution environment — for instance, frequency of library/API calls, complexity metrics, framework, API level, OS version, etc. This data will be used to train a regression model that estimates the power error of the software-based estimator.

As a side contribution we propose to answer the following research questions:

RQ1: Can static analysis be used to assess the reliability of energy models for different running environments?

RQ2: What is the improvement of using hybrid energy measurements over software-based measurements?

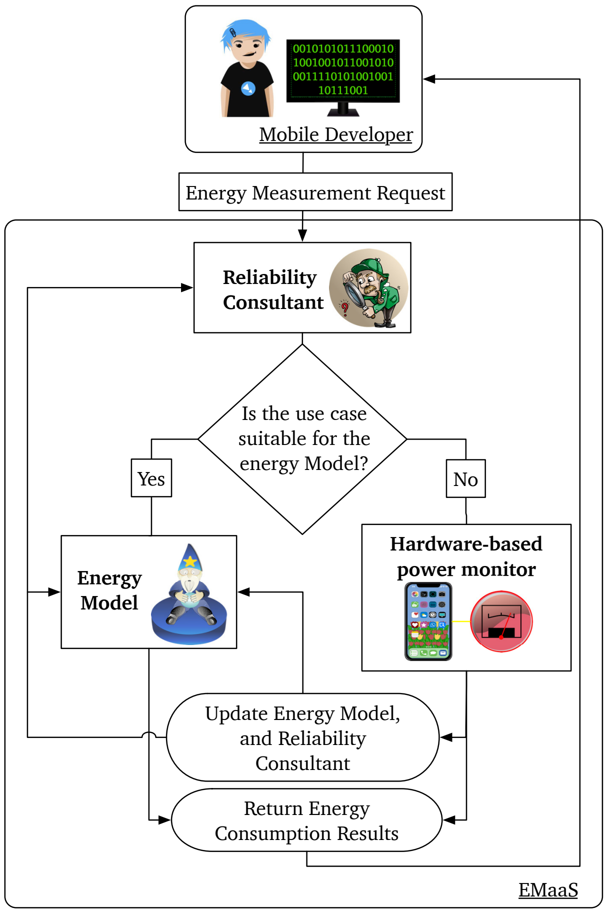
RQ3: What is the proportion of software vs. hardware measurements?

The combination of the modules **Reliability Consultant**, **Energy Model**, and **Hardware-based Power Monitor** is depicted in Fig. 3. Upon a measurement request from the developer, the system asks the **Reliability Consultant** whether the measurement needs to use a **Hardware-based Power Monitor**. If not, the estimation provided by the **Energy Model** is returned to the developer. On contrary, if the **Reliability Consultant** does not consider the **Energy Model** reliable for the given mobile app, an hardware-based measurement is provided, and the **Energy Model** and **Reliability Consultant** are updated with new data.

III. Why is it New?

Very interesting tools have been proposed by researchers to measure energy consumption. However, given the ever-changing nature of the mobile application world, keeping them up-to-date and ready to use for developers can be challenging. This system is able to be continuously updated and to adapt its energy model to new settings.

Moreover, this will help researchers having more valuable contributions in the field of energy efficiency of mobile applications. Recent contributions are using energy models to estimate energy consumption [8–11]. However, the validity of these measurements is not easily assessed. In addition, most contributions have their experiments limited to a small set of environments, providing a relevant threat to the validity.



Mobile Developer
0010101011100010
1001001011001010
0011110101001001
10111001
Energy Measurement Request
Reliability Consultant
Is the use case suitable for the energy Model?
Yes
No
Energy Model
Hardware-based power monitor
Update Energy Model, and Reliability Consultant
Return Energy Consumption Results
EMaaS

Fig. 3. How a normal energy measurement request is processed, starting from the perspective of a developer.

This work bridges the gap between industry and academia in terms of energy measurements for mobile applications. To the best of our knowledge, this is the first time reliable energy measurements can be deployed to the industry without requiring an expensive setup. Furthermore, no prior work has leveraged a tool to assess the compatibility between an energy model and its running environment.

IV. Risks

The main risks that can affect this system are related to *Security* and *Privacy*.

a) Security: The system has to account for malware mobile software that can affect the devices of providers.

b) Privacy: Developers do not want to disclosure their apps before deployment. Mechanisms need to be deployed to prevent mobile applications from being collected by malicious providers or super-providers. Furthermore, the mobile devices available in the system need to be protected from mal-intended developers. The data stored in the devices should not be accessed by the system or the mobile application under test.

c) Measurement best practices: Typically, energy measurements require rigorous approaches to mitigate potential bias on measurements. For instance, the app under analysis should be running in isolation on the phone while no other apps are running in parallel [12].

To address these issues, energy tests have to be executed in a closed environment, instantiated in the mobile device. In some cases, the device may have to be restored. Thus, providers may not be able to use their own personal devices. For safety reasons, the system may be limited to devices exclusively acquired for app development.

V. Next Steps

In this paper, we present the idea of using peer-to-peer cloud computing to deliver reliable energy measurements as a service. We are interested in improving the way researchers and developers are profiling the energy consumption of their mobile software. By making it available as service, we aim to help developers make informed design decisions regarding the energy efficiency of their code.

In the early stages, EMaaS will require a considerable software development effort. Thus, we divide the implementation into three steps. In the first step, we will be interested in delivering a system that yields hardware-based measurements. This will serve as a base system to add more sophisticated energy measurement techniques in the following steps.

The second step will consist of including the **Energy Model**. In particular, we will be looking at how to improve existing models for energy estimation. One big advantage of our approach is the access to real measurement data collected from a wide variety of devices and OSs.

Finally, in the third step, we will be adding the **Reliability Consultant**. We will use measurements collected from both energy models and power monitors to train a regression model of the reliability of energy estimators in a particular execution scenario.

For an initial proof-of-concept, the system will support Android apps instrumented with *Espresso* test cases, since it has been the most suitable UI test framework for energy measurements [13, 14]. In order to initially avoid privacy and security risks, the system will be exclusively available to invited users.

VI. Acknowledgments

This work is financed by the ERDF — European Regional Development Fund through the Operational Program for Competitiveness and Internationalization - COMPETE 2020 Program and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia with reference UID/CEC/50021/2019, and within projects GreenLab (POCI-01-0145-FEDER-016718) and FaultLocker-Ref (PTDC/CCI-COM/29300/2017). Luis Cruz is sponsored by an FCT scholarship grant number PD/BD/52237/2013.

References

[1] K. Moran, M. L. Vásquez, and D. Poshyvanyk, "Automated GUI testing of android apps: from research to practice," in *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017 - Companion Volume*, 2017, pp. 505–506.

[2] H. Muccini, A. D. Francesco, and P. Esposito, "Software testing of mobile applications: Challenges and future research directions," in *7th International Workshop on Automation of Software Test, AST 2012*, 2012, pp. 29–35.

[3] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *35th International Conference on Software Engineering, ICSE '13*, 2013, pp. 92–101.

[4] D. D. Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. D. Lucia, "Petra: a software-based tool for estimating the energy profile of android applications," in *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017 - Companion Volume*, 2017, pp. 3–6.

[5] S. Chowdhury, S. Borle, S. Romansky, and A. Hindle, "Greenscaler: training software energy models with automatic test generation," *Empirical Software Engineering*, Jul 2018.

[6] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis*, 2010, pp. 105–114.

[7] D. D. Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. D. Lucia, "Software-based energy profiling of android apps: Simple, efficient and reliable?" in *IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*, 2017, pp. 103–114.

[8] D. Li, S. Hao, J. Gui, and W. G. J. Halfond, "An empirical study of the energy consumption of android applications," in *30th IEEE International Conference on Software Maintenance and Evolution*, 2014, pp. 121–130.

[9] D. Li and W. G. J. Halfond, "Optimizing energy of HTTP requests in android applications," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, DeMobile 2015*, 2015, pp. 25–28.

[10] C. Sahin, L. L. Pollock, and J. Clause, "How do code refactorings affect energy usage?" in *2014 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14*, 2014, pp. 36:1–36:10.

[11] S. A. Chowdhury, S. D. Nardo, A. Hindle, and Z. M. J. Jiang, "An exploratory study on assessing the energy impact of logging on android applications," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1422–1456, 2018.

[12] L. Cruz and R. Abreu, "Performance-based guidelines for energy efficient mobile applications," in *IEEE/ACM International Conference on Mobile Software Engineering and Systems, MobileSoft 2017*, 2017, pp. 46–57.

[13] ——, "Measuring the energy footprint of mobile testing frameworks," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 400–401.

[14] ——, "To the attention of mobile software developers: Guess what, test your app!" *Empirical Software Engineering*, Feb 2019.