Energy Consumption Measurement Frameworks for Android OS: A Systematic Literature Review

Vladislav Myasnikov*, Stanislav Sartasov†, Ilya Slesarev‡ and Pavel Gessen§
*Saint Petersburg State University
vladislav.myasnikov@bk.ru
†Saint Petersburg State University
stanislav.sartasov@spbu.ru
‡Saint Petersburg State University
slesarev.pr@gmail.com
§Saint Petersburg Polytechnical University
pashagess2@mail.ru

***Abstract*—In a modern world smartphones became a commonly used electronic devices performing numerous day-to-day tasks and much more. Quick battery discharge degrades user experience, and computationally intensive or badly written programs are responsible for it. It is not always evident which tool to use and how to set up an experiment to estimate energy consumption of a specific application. For this we conducted a Systematic Literature Review (SLR) to list existing frameworks to measure application power metering for Android OS, to classify the approaches used to create them and also to assess their accuracy and experimental methodology. Our findings indicate that although there is a considerable amount of studies in this field with various approaches, there is still a vacant place for a readily available tool, and it is difficult to compare accuracy of different frameworks. However there is a solid set of practices and techniques for experimental setup in application energy measurement.**

I. Introduction

It is impossible to imagine a modern world without smartphones and other mobile devices. Their compact size combined with significant computational capabilities grant them a firm position as a day-to-day informational and recreational tool. At the end of 2019 a number of smartphone users is expected to be 3.2 billion with Android OS as a leading mobile operating system [1].

As smartphones and tablets are mobile electronic devices, its user experience is substantially defined by battery lifetime. While hardware components are constantly improving with power-saving electronics and more capacitous batteries being available to the market, inefficiently written software causes degradation of user experience due to elevated charge drain.

Different applications and even different versions of the same application consume energy differently. A school of thought called ”green software development” advocates a need to consider energy consumption as well as performance metrics during application development [2]. A common practice is energy-efficient refactoring — such a change in software code that doesn't change its end user functionality but decreases its energy footprint.

Was a particular energy refactoring effective? How much energy did we save by applying it? To answer these questions one should be able to compare application or module energy consumption before and after refactoring. Therefore a tool for conducting such experiments is required. Such a tool may come as one-time testbed for a specific project or a more generic and reusable framework or utility. It should be noted that there's little uniformity among such tools. They differ not only in metering methodologies but in measurement results as well, i.e. some frameworks measure battery charge percentage change, other calculate consumed power in watts, while another group operates in abstract units of measure.

In order to help practitioners and engineers better understand current state-of-the-art approaches to energy consumption measurement and to select proper approach, technique or tool for a particular experiment, we conduct a systematic literature review (SLR) on the energy consumption measurement frameworks for mobile devices using Android OS. We selected this mobile platform as it is the most presented platform on the market compared to iOS and Windows Phone [3], and it is also open-sourced, meaning that some types of frameworks (for example those that modify OS kernel) might be absent on other platforms.

This paper is organized as follows. In Section 2 our methodology for SLR is described and research questions (RQs) are formulated. Section 3 contains answers for RQs. Limitations of this study are reported in Section 4. A side question of relating our proposed framework classification to a number of commercial profilers is addressed in Section 5. Conclusions are drawn and future work is outlined in Section 6.

II. Method

We followed SLR guidelines by Kitchenham and Charters [4] with a number of differences:

- Instead of manual search process we addressed online search engine. While this decision certainly affects the selection of studies compared to manual search, we used a large number of papers to make a preliminary list and introduced additional phases to study selection process, so we think the impact of this change on SLR quality is negligible.

- Quality assessment was done along with the data extraction process and in some sense — as a part of it. However Kitchenham and Charters [4] allow such methodology, noting that quality information can be used either to assist primary study selection by constructing detailed inclusion/exclusion criteria prior to the main data collection activity or to assist data analysis and synthesis, so it is collected along with the main data.

A. Research Questions

A following list of research questions (RQs) was compiled during initial literature assessment:

- RQ1: What approaches exist to estimate code fragments, methods or application energy consumption under Android OS?
- RQ2: Are there open source code repositories or other programming artifacts for corresponding frameworks?
- RQ3: What devices are used in measurement experiments?
- RQ4: How many frameworks specify or suggest experimental methodology?
- RQ5: What is the measurement precision and what is the base scenario to compare to?

However after preliminary data extraction we also added the following questions due to the freqently occuring gaps in methodology in the reviewed studies:

- RQ6: What units of measurements are used in experiments with a particular framework and what is measured?
- RQ7: How do frameworks deal with metering hardware frequency being considerably lower than CPU frequency?

RQ6 is answered to classify all the different ways report measurement results. RQ7 came into attention due to the possibility of a child thread be entirely executed between multimeter synchronization impulses therefore its contribution to the energy consumption would not be properly recorded.

B. Search Process

A preliminary list of 931 studies was formed with a Google Scholar[1] resource. The query was ”android ”energy-efficiency” framework”, therefore ”energy-efficiency” was required to appear as a phrase in a study text.

C. Study Selection

At first inclusion criteria were rather simple — include studies about energy measuring frameworks, exclude all others — but it was quickly became apparent they were not sufficient. Indeed some studies were discussing a software and hardware complex or purely software tool for measuring energy expenditure which was generic enough to be called a framework. However many articles contained a description of a testbed — a testing environment that measures specific program energy consumption in one way or another. In a considerable amount of cases the distinction between a testbed and a framework was blurry. We consider a programming artifact to be a testbed if the experiment itself and its conclusions are the centerpiece of the corresponding article. If, however, this artifact is the main discussion point for the article, we consider it to be a framework.

[1]`https://scholar.google.com`

Usually a testbed is only vaguely described, while frameworks are documented in detail, up to the point of open source repositories, so this separation worked well. There was a handful of notable exceptions however where a study focused on a specific energy-efficiency question also contained a detailed description of its testing environment along with the experimental methodology and software architecture. In such cases a question was asked if this testbed is described in such a detail that it can be a subject of a framework-centered article. In the case of positive answer a study was added to short-list.

Additionally we do not include articles regarding smart watches and similar devices as its functionality is rather limited compared to the smartphone, therefore only mainstream Android OS is considered as a focus of this SLR. We also exclude articles focused on Windows Phone and iOS.

To summarize, our final criteria to include a study into a short-list was as follows:

- Studies describing technical artifacts for other operating systems than mainstream Android OS are excluded.
- Studies focused on a framework as an end result of a research are included.
- Studies focused on other topics but containing enough information on a testbed or a method to make one for it to be considered alienable as a standalone framework are included.
- All other articles are exlcuded.

At first stage Myasnikov, Sartasov and Gessen processed every article independently. An article was assigned ”+” if a reviewer considered it to be worthy of inclusion, ”-” if a review was negative and ”$\sim$” if in doubt. Three pluses or two pluses and a tilde meant inclusion to short-list, while an article with three minuses was excluded from further reading. Other review combinations signified review conflicts. About 45% of studies were marked conflicts at the end of this phase.

Secondly, studies with conflicting reviews were additionally reviewed by Slesarev, and each article was read in greater detail and discussed until a consensus was formed among the researchers. At the end of this phase 51 articles were added to short list.

Some studies in the result list deserve specific mention. Although WattsOn framework [5] targets Windows Phone, it was included because authors claim their approach is portable to Android OS. 3 additional articles were added to the short list manually as they were not a part of original study list:

- Yoon, Kim et al. [6]
- Zhang, Tiwana et al. [7]
- Li and Gallagher [8]

Those studies became known either due to past literature reviews in similar topics or during review process when they were referenced in multiple other articles. They were also reviewed independently by three researchers and got 3 pluses for review.

Third stage of study selection was done along with data extraction. 6 studies in fact were written about a different subject while looking like a framework article at a top level, so our short-list contained 48 articles.

However at this time we've found out that in some cases multiple studies were describing the same framework under different angles or at different points of development cycles. It is a common situation for an ongoing research project. As the focus of our SLR is the frameworks and not the articles per se, if multiple studies were written by the same or similar collective of authors from the same institution and described a similarly named frameworks based on a similar principles, after discussion and consensus between researchers they were considered to be written about a single framework, and data extraction results were merged for these studies. In particular we grouped the articles concerning the following frameworks:

- JouleUnit by Wilke et al. [9], [10]
- Greendroid by Couto et al. [11], [12], [13]
- PETrA by Di Nucci, Palomba et al. [14], [15]
- Aggarwal et al. [16] and Feghi [17]
- Li and Gallagher [18], [8]
- Ahmad et al. [19], [20]

In the end, out of 931 studies aggregated automatically and 3 papers added manually our short-list contained 48 articles summarized into 41 frameworks.

D. Data Extraction

To answer the stated RQs we answered the following data extraction questions for each included study or study group:

- Conceptual questions:
  1) What is the method of reading current battery charge level or energy expenditure? Examples include external multimeter, internal sensor, Android OS API etc.
  2) How large is the part of an application to be profiled? For example, one may measure energy consumption of a single line of code, a code block, a method, a unit-test, an application or all currently running applications.
  3) What is the end result of a measurement experiment?
  4) What are the units of measurement utilized in a framework (i.e. watts, joules, relative units)?
- Technical questions:
  1) Is the application code executed on a smartphone or special test device and what is its Android OS version?
  2) What kind of code instrumentation is used if any? In this context we define code instrumentation as addition of framework-specific subprograms (i.e. for method start and end logging) to the target source code.
  3) How is energy consumption measurement started from a technical standpoint?
  4) How is raw metering data downloaded or otherwise obtained for processing?
  5) What are the raw data transformations used to present experiment outcome?
  6) Is there an open source code for a framework?
- Experimental methodology questions:
  1) What are the preliminary actions to undertake on a test device before starting measurements? Examples include turning off Wi-Fi, stopping applications, charging up to 100%, controlling device temperature etc.
  2) Is measurement accuracy estimated? If yes, what is the base of comparison?
  3) Does framework experimental methodology consider frequency of metering tools? What is the frequency of a tools used by authors?

If an article didn't contain an information required to answer the question or if a question was not applicable to a specific article, it was also noted.

A pool of works was distributed along Myasnikov, Slesarev and Sartasov, and data from each study was extracted independently. Quality control was applied selectively if there was a misunderstanding between team members regarding particular data extraction result for a specific study. In this case data extraction was repeated collectively until consensus was formed.

Data extraction results were aggregated in an online document[2]. We were able to extract relevant information for each question for each framework in the list.

E. Quality assessment

Generally SLRs are accompanied with a quality assessment procedure concerning every included article. As we're more interested in frameworks than in individual articles themselves, we decided to modify this process by assessing the quality of individual studies if they were not grouped with other publications, otherewise we assessed a group of articles as a whole. Therefore if a framework is described in two articles, we assess the quality from both of them simultaneously.

To assess the quality we formulate the following questions:

1) Is there an open-source code for the framework?
2) Is framework measurement approach described?
3) Is experimental methodology described?
4) Is framework accuracy assessed?

These questions intentionally overlap with our RQs, as this information is in our opinion essential if a reader wants to understand a proposed framework. They were scored as follows:

- Question 1: yes (Y) if a repository can be found, partly (P) if a program using a framework can be found, but not a source code, no (N) otherwise. Even if an article states

[2]`https://docs.google.com/spreadsheets/d/17D1ArPavFQaFPGnU-OI3r8x1QoboWOoEa-rHR-9U98I/edit?usp=sharing`

a number of lines of code for a described framework or gives code fragments or algorithms, it is still a no.

- Question 2: yes (Y) if a framework approach is described in great detail, and entire thought process from principles to implementation can be tracked, partly (P) if only a general description of framework principles is present, no (N) otherwise.
- Question 3: yes (Y) if a methodology is described in great detail and limits and bounds are stated, partly (P) if only a broad description of measurement methodology is given, no (N) otherwise.
- Question 4: yes (Y) if an experiment and a base case are described, partly (P) if some consideration is given to accuracy, but not a thorough one, no (N) otherwise.

The scoring procedure was Y = 1, P = 0.5, N = 0. Article quality was obtained as a sum of individual question scores. Results are given in Table I.

III. Discussion of research questions

This sections contains our findings in relation to the specific research questions as stated above.

A. RQ1: What approaches exist to estimate code fragments, methods or application energy consumption under Android OS?

We can classify different approaches for software energy consumption estimation into two bins: if power measurement is direct or indirect.

Direct measurement approach means that some metering agent is directly measuring voltage, current or even power. Subclasses of this approach are as follows:

- **External meter**: External digital multimeter is connected to battery contacts of smart device. Sometimes to compensate voltage drop in Li-ion batteries during discharge and therefore normalize power readings an external power generator working as a constant voltage source is connected to testing device instead of default battery. Framework launches an application in question or a unit test alongside power measurement. In the end total power consumption is estimated by linear interpolation as

$$E = \sum_{i=1}^{N_{read}-1} U_i \times \frac{I_{i+1} + I_i}{2} \times (t_{i+1} - t_i) \qquad (1)$$

  where E is total energy, N_{read} is number of power readings, U_i is ith voltage read, I_i is ith current read, t_i is time of ith read. Some multimeters write the time of a read directly, while for other $t_{i+1} - t_i$ is an inverse of a multimeter frequency. Step interpolation is also used in some works.

  If a more detailed report is needed, for example at a level of methods or code blocks, then the source code is instrumented with additional logging instructions for method or code beginning and end, and two data traces are generated — power readings and application execution trace as in Couto et al. [11]. In this case system clocks on smartphone and multimeter or controlling PC should be synchronized before the start of experiments. Power readings can then be aligned with execution trace and provide an insight which method or piece of code is responsible for high energy usage. As instrumentation introduces additional code to be executed, its energy overhead should also be estimated and subtracted from final readings.
- **Internal meter**: This approach utilizes internal power meters installed on the smartphone by manufacturer and Android OS API to access them. While generally such a tool can get a good power consumption estimate for a smartphone as a whole, under specific experimental conditions it can be trimmed down to a level of a single application in question. Because smartphone and power sensor are using the same system clock, there are no issues with clock synchronization. Power readings requests may be integrated into instrumentation code. Energy consumption is estimated in the same way as before.

Indirect measurement approach (or model-based approach) means that profiling software is aggregating some information regarding code execution and relates it to energy consumption using some mathematical model. Frameworks utilizing this approach operate in two phases: model calibration and energy estimation. At first stage model coefficients are determined or tuned with preliminary experiments or reference data. It is an important step not only for different smartphone models, but for different devices of a same model as well [21]. After a model is tailored for a device, actual energy metering experiments may be commenced.

Subclasses are formed based on the type of aggregated information used in the model:

- **Working time model**: For this group information regarding working time of various smartphone systems is aggregated. Various devices may consume energy differently under different modes of operation. For example, Wi-Fi module consumes different amounts of energy when idle, when looking for a network and when transferring data over a network. Therefore energy consumption value is calculated as

$$E = \sum_{i=1}^{N_{dev}} \sum_{j=1}^{NP_i} P_{ij} \times t_{ij} \qquad (2)$$

  where E is total energy, N_{dev} is a number of tracked devices on a smartphone, NP_i is a number of different power characteristics of ith device, P_{ij} is a jth power characteristic of ith device, t_{ij} is active time for ith device operating under P_{ij} power characteristic.

  Calibration phase consists of experiments for determining power profiles for individual components using one of the direct measurement approaches: both external and inter-

TABLE I
STUDIES QUALITY

Studies	Q1 Repository	Q2 Approach	Q3 Methodology	Q4 Experiment	Quality score
Zhang et al. [7]	Y	Y	Y	Y	4
Hindle et al. [21]	Y	Y	Y	P	3,5
Di Nucci, Palomba et al. [14], [15]	P	Y	Y	Y	3,5
Tuysuz, Uçan and Trestian [22]	P	Y	Y	Y	3,5
Wilke et al. [9], [10]	Y	Y	P	P	3
Hao et al. [23]	N	Y	Y	Y	3
Couto et al. [11], [12], [13]	Y	Y	Y	N	3
Bareth [24]	N	Y	Y	Y	3
Kamiyama, Inamura and Ohta [25]	N	Y	Y	Y	3
Saksonov [26]	N	Y	Y	Y	3
Sahar, Bangash and Beg[27]	Y	Y	P	P	3
Ahmad et al. [19], [20]	N	Y	Y	Y	3
Pandiyan and Wu [28]	N	Y	Y	Y	3
Huang et al.[29]	N	Y	Y	Y	3
Oliveira et al.[30]	N	Y	Y	P	2,5
Aggarwal et al.[16], Feghi [17]	N	Y	P	Y	2,5
Westfield and Gopalan [31]	P	Y	N	Y	2,5
Dolezal and Becvar [32]	N	Y	Y	P	2,5
Hu et al. [33]	N	Y	P	Y	2,5
Yoon, Kim et al. [6]	N	Y	P	Y	2,5
Larsson and Stigelid [34]	N	Y	P	Y	2,5
Fischer, Brisolara and Mattos [35]	N	Y	P	Y	2,5
Lee, Yoon and Cha [36]	N	Y	P	P	2
Mittal, Kansal and Chandra [5]	N	Y	N	Y	2
Chen and Zong [37]	N	Y	P	P	2
Hung et al. [38]	N	Y	N	Y	2
Carette et al. [39]	N	Y	Y	N	2
Kapetanakis and Panagiotakis [40]	N	Y	P	P	2
Dong, Lan and Zhong [41]	N	Y	P	P	2
Lee et al. [42]	N	P	P	Y	2
Chung, Lin and King [43]	N	Y	P	N	1,5
Shin et al. [44]	N	P	N	Y	1,5
Jung, Kim and Cha [45]	N	Y	P	N	1,5
Tsao et al. [46]	N	Y	N	P	1,5
Metri [47]	N	Y	P	N	1,5
Gao et al.[48]	N	P	P	P	1,5
Banerjee et al. [49]	N	Y	N	P	1,5
Li and Gallagher [18], [8]	N	Y	P	N	1,5
Li et al. [50]	N	Y	P	N	1,5
Walcott-Justice [51]	N	Y	N	N	1
Kim, Kyong and Lim[52]	N	Y	N	N	1

nal meters are conceptually suitable. Linear regression is used to extract power coefficients from experimental data. As an alternative some frameworks utilize Android power profile data [53].

Energy estimation is assessed by measuring active device time during experimental code execution. Different components have different ways for measuring their active time, i.e. CPU stores information about its time in different power states in `proc` folder, while Wi-Fi generates system events when it transitions from one power state to another.

Additional estimations may also be incorporated into such model, for example, corrections for battery discharge rate [44].

- **Instruction energy model**: This group of frameworks considers energy consumptions of various code instruction types — conditional statements, loop controls, method calls, floating-point operations etc. Energy consumption is calculated as

$$E = \sum_{i=1}^{N_{instr}} P_i \times n_i \tag{3}$$

where E is total energy, N_{instr} is number of different instruction types in a model, P_i is power consumption of a single instruction of ith type, n_i is number of ith type instructions in the code.

Model calibration is done by measuring power consumption of each instruction type in a synthetic tests using direct approach.

Total energy is calculated from instruction statistics of a specific code execution trace. It should be noted that it is not required to launch test code under Android OS if no specific Android API is invoked. Instruction statistics may be aggregated in any suitable environment.

- **Method/API call energy model**: This approach is similar to the previous one, but instead of a power profile for a single instruction energy consumption of a system or API call, or framework method is calculated. Models under this approach are created under assumption that most of the time and energy is spent outside of application code, and therefore good estimation of application energy consumption can be obtained by analyzing its API usage.

We assign each of the listed frameworks to its approach in the Table II.

TABLE II
APPROACHES FOR ENERGY CONSUMPTION ESTIMATION

Direct measurement	
External meter	[43], [9], [21], [46], [39], [50], [52], [34], [41]
Internal meter	[9], [45], [51], [37], [40], [34], [35]
Indirect measurement	
Working time model	[44], [11], [14], [5], [47], [36], [32], [48], [25], [24], [49], [38], [26], [6], [7], [27], [52], [22], [29], [42]
Instruction energy model	[23], [18], [19]
Method/API call energy model	[30], [16], [31], [33]

B. RQ2: Are there open source code repositories or other programming artifacts for corresponding frameworks?

This research question intentionally overlaps with the Question 1 in Quality Assessment section. Energy consumption measurement frameworks are practical and generic tools by definition, therefore it is reasonable to expect its source code available for reuse. As an alternative a metering application may also be sufficient.

Among the 41 included frameworks the results are as follows:

- 5 frameworks have their source code available in repository: JouleUnit [9], [10], GreenMiner [21], GreenDroid [11], [12], [13], EnSights [27], PowerTutor [7]
- 3 frameworks are openly presented as a ready application: PowerTutor [7], PETrA [14], [15], "PowerProfiler & Energy-aware Network Selection" application in Google Play by Uçan, Tuysuz and Trestian [22]. A web-site for another framework, Orka [31], is available in Imperial College of London intranet.
- 3 studies include code samples or statistics in the text: Sema [35], Huang et al. [29], Kapetanakis and Panagiotakis [40].

Note that PowerTutor [7] is marked as having both a source code and an application available. Non-mentioned studies don't refer to the source code.

It should be noted separately that all of the frameworks with their code available are not kept up to date. PowerTutor and JouleUnit were abandoned over 5 years ago. At the time of writing (January 2020) most recent commits are found in GreenMiner (May 2018), EnSights (August 2018) and GreenDroid (January 2019), so we may call them semi-abandoned. In our experience lack of recent commits usually indicates either a research project being finished or it was abandoned for some reason. Additionally, none of the projects can be built out of the box, and build instructions are not provided in details. As stated before, PowerTutor was used in a number of other studies when it was supported, but it is not surprising that a significant amount of more recent testbeds we've seen in studies during study selection phase are based on industrial grade Monsoon power monitor [54][3].

We conclude that studied frameworks rarely provide their code in open source repositories. Those who do are not easy to launch. What's worse, not all of the frameworks are available to the practinioners even in the form of proprietary software.

C. RQ3: What devices are used in measurement experiments?

The number of devices running Android OS is hard to reliably count: from the very first HTC Dream model line to modern foldable smartphones as well as tablets. Android OS is also continues to being developed with a major release by Google every year. For example Android Q version released in 2019 supports foldable smartphones and 5G-devices [55]. Additionally, Android API is not static, although generally an application is forward compatible with a newer Android OS version [56]. Regardless, application well-behaving on a specific version of Android platform might be glitching on another version due to the API change. With such variety of devices and versions it is important to understand what range of devices and Android OS versions are covered by existing energy metering frameworks.

We obtained the following distribution of target devices from the listed frameworks:

- A special testbed is used in ∼17% studies. In this context a testbed is a special platform running Android OS which is not a smartphone or a tablet, although it might be functionally similar. For example, Odroid-A platform has a set of functions similar to Samsung Galaxy S2 [44].
- An emulator is used in ∼12% studies. Emulator allows to run applications without necessity to procure a real device. However emulator is not a complete substitute for a smartphone or a tablet, in particular application performance might be worse in emulated environments thus negatively affecting measurement accuracy. Another limiting factor is a number of devices available for emulation, for example, only one model of the Nexus 7 tablet is emulated in Westfield and Gopalan [31]. Hence a range of available emulated devices is significantly limited compared to commercially available smartphone and tablet ranges, albeit this approach is significantly cheaper. WattsOn framework [5] is also included into this

[3]Monsoon power monitor is a high frequency multimeter to be connected between smart device battery and electronics. It is capable to upload measurement results to a PC. While it was designed with smart device energy measurements in mind, it is just that - a high-quality multimeter, which can be used in a variety of scenarios not limited to Android power profiling. Therefore we do not consider Monsoon to be a metering framework by itself, but it can lie in a foundation of one.

category as its authors claim it to be portable to Android emulators.
- Real devices such as smartphones and tablets are used in the other studies. Smartphones are much more prevalent than tablets — the latter are experimented upon in a single study [51]. A number of devices used for experiments varies from 1 to 3.

To evaluate a potential range of devices for a particular framework one can try different methods. Firstly, this range may be estimated using target Android OS version used in the experiments described in a selected study. Then the article is considered applicable for devices with the specified version and (with a caution) higher due to forward compatibility [56]. Such an assessment, however, is rather imprecise as there are no guarantee that experiments were conducted on a minimally available version. Secondly, an estimate of supported Android versions can be based on measurement tools being used in experiments if they are specified in a study. In general this assessment is more precise than the first. Thirdly, it's worth to take into account technical restrictions imposed by the framework itself if they are mentioned.

With the listed frameworks we obtained the following results:

- In the articles where it was possible to draw conclusions on Android version experiments are conducted on Android OS versions 2 to 5, so approximately 90% of all devices are supported [57].
- Some restrictions are stated explicitly, for example, device power profile must be uploaded to the framework server prior to experiments [19]. However, in some studies its restrictions are implied, for example, Android Power Profiler tool used for energy metering is available only for Android 5.0+ devices [30], while Trepn Profiler is available for a limited range of devices for Android 4.0+ [51], [47], [33].
- At least 12 frameworks require root access for the device, mainly to use additional API. It might be unacceptable for some users as rooted devices void the manufacturer's warranty. Additionally there is a risk of disrupting proper OS functioning under root privileges compared to a non-root user.
- In 7 of those 12 frameworks it is not enough to just have a root access, as one must also integrate a kernel module into the OS [45], [46], [36], [6], [42] or otherwise modify an existing Android framework [48], [39]. Such preparatory actions impose a considerably higher entry barrier than simply obtaining root access rights for the device.

In the end the range of supported devices is wide despite some of the frameworks targeting testbeds or emulators. While individual restrictions make some frameworks harder to set up than others, we conclude that development tools versioning and framework accessibility in general (see RQ2) is more limiting for their usage than device range.

D. RQ4: How many frameworks specify or suggest experimental methodology?

It is not enough to write code, launch a framework to estimate its energy consumption and get results. Experimental methodology is as important as framework itself. By methodology we mean a set of rules, procedures and techniques aimed to decrease, eliminate or otherwise take control of external and internal influences, which are not independent variables of the experiment, on experimental outcome. In a way, methodology defines a context of framework usage as authors see it, and non-stringent approach to experimental setup and outcome interpretation leads to innacurate or outright wrong results. Ill-thought procedures or rules or lack of thereof in a study is an alarming signal.

In this regard studies in our list generally take experimental methodology into consideration with only 6 articles not mentioning it. We extracted this information from the others and aggregated them into following groups:

- Testing environment setup.
- Repeated launch of tests, benchmarks etc.
- Overhead estimation.

Table III contains study distribution into each group.

A smartphone or a tablet is a complex device with numerous settings. Using it as a testing device requires to pay attention to its configuration. To a lesser extent this is also true for emulator. Such parameters include the following:

1) Reducing activity of background processes (both system and non-system) [23], [51], [39], [27], [40], [28].
2) Increasing scheduling priority of the subject application [23].
3) Decresing screen brightness or switching it off completely [21], [25], [39], [27], [28], [22], [29].
4) Switching off all unnecessary modules (Wi-Fi, 3G, GPS, accelerometer, proximity etc.) [39], [26], [20], [40], [28].
5) Charging battery for the same level [24], [39], [22].
6) Shutting down for cooling down the battery heat [39].
7) Warmup execution [30].
8) Time waiting before beginning a new test in order to let background processes calm down [32], [27].
9) Application reinstalling [21].
10) Application cache erasing [14], [27].
11) Rebooting the device [26].
12) Factory data reset [24], [26].

If left unchecked these factors introduce additional energy drain and thus skew measurement results and decrease repeatability, therefore a proper setup is required. About 40% of the listed frameworks take it into consideration one way or another.

A framework itself might introduce additional energy consumption. For example, code instrumentation is required to obtain execution trace, but as it is implemented in the form of additional code, it inevitably introduces energy overhead. Aggregating system data while executing a test suite is another reason behind elevated energy consumption during measurement. To alleviate this problem one should estimate this overhead and subtract it from measurement results. In particular code instrumentation is estimated by running all the tracking code from a selected trace without the original test code and measuring its energy footprint. Around 50% of all studies address this issue, with some studies only stating measures to overcome it [37], [35], while other actually estimate its effect [25], [6].

A notable number of studies — about 37% — repeat their measurements multiple times to get an average or mean value of consumed energy. In this way authors reduce random metering equipment jittering and OS scheduling impact on measurement results [9], [30], [17], [29].

Chung, Lin and King [43] and Jung, Kim and Cha [45] should be mentioned separately. Those studies specify amount of time for the experiment to last to produce adequate results. However they do not specify reasoning behind selecting particular values, so we do make another group for them.

Overall, we conclude that collectively studies in our list contain enough guidelines for a practitioner to properly conduct energy consumption estimation process.

TABLE III
EXPERIMENTAL METHODOLOGY GROUPS

Group	Number of papers
Overhead estimation	20
Environment setup	16
Relaunching	15
None	6

E. RQ5: What is the measurement precision and what is the base scenario to compare to?

Measurement accuracy is one of the key characteristics of energy profiling framework. Thus a question of framework accuracy might be expected to be answered in a corresponding study. Our review shows that accuracy estimation of some sort was carried out approximately in 61% works, which can be divided by a type of estimation into the following groups:

- Comparison with direct measurement.
- Comparison with another tool (frameworks, power models).

Study distribution into each group is shown in Table IV. Some studies fell into several groups at the same time.

If a study compares framework accuracy with a baseline in the form of external power or current metering device data, we correspond it to comparison with direct measurement group. Such a device can be an ordinary multimeter [20] or a special equipment like, for example, Monsoon [14], [5], [6]. Accuracy is estimated as a relative difference between framework and equipment data. Note that baseline scenario organization for this group is similar to the direct measurement approach in framework building (see RQ1). A significant limitation of this technique is also the similar — usage of an external meter comes at a price of estimating only the total power consumption of a test device without breaking it down by hardware components, and therefore intercomponent comparative analysis may be limited and require specific experimantal setup or even impossible. This accuracy estimation technique is used both for direct measurement and model-based.

Another way to assess accuracy of a newer framework is to compare it with an already existing tool which is considered to be a baseline. Once again accuracy is a relative difference between frameworks. Several studies match themselves against a PowerTutor framework [31], [22], while one of the studies compares the proposed framework with Appscope [42]. While such comparisons allow researchers to evaluate measurement accuracy with respect to the few existing solutions, lack of homogenity in baseline framework selection indicates that there is no universally accepted accuracy standard.

As stated above, several studies fell into several groups at the same time. For example, in some studies a comparison is made both with real measurements and with other frameworks [19], [35], [42]. One interesting take on this approach is described in Saksonov [26] where a derivable energy profile is compared to the default Android Power Profile[4] and direct measurements.

Based on this we conclude that such two-factor estimation process provides better context of actual framework accuracy. Complete lack of accuracy assessment strongly indicates a study of a subpar quality.

TABLE IV
MEASUREMENT ACCURACY EVALUATION GROUPS

Comparison group	Number of papers
Real measurements	18
Another tool (framework, power model)	13
None	17

F. RQ6: What units of measurements are used in experiments with a particular framework and what is measured?

Code energy consumption measure using a framework can be expressed in different terms. For example, recall equation 2. With the direct approach total energy is the most obvious way to estimate consumed energy. However when voltage is constant amount of consumed Ampere-hours is as informative as total energy. Likewise, if power reads are regular their amount and distribution contain all necessary information. Similar idea is also frequently used in model-based approach for estimating CPU power usage [53]. Moreover, the same value can be presented with varying degree of accuracy. For example, under some experimental setups energy may be better measured in μJ than in mJ or J for a more conclusive results. Therefore lack of homogenity in result presentation among the listed frameworks can be explained.

Table V aggregates the characteristics measured in the listed studies and distributions of the corresponding measurement units. Among absolute characteristics power and energy were most often measured with their values presented in mW and J, respectively, current, voltage and electric charge are estimated less often. In specific cases higher accuracy measurement was

[4]An XML file detailing power characteristics of smartphone components provided by its manufacturer.

used with μW for power [47], and nJ for energy [28]. However, some studies are not precise in terminology, in particular difference in battery charge is also labeled as consumed energy [44].

It is not uncommon to estimate specific energy calculated per unit of device operation. Examples include instructions [43], MHz of processor, screen inch or dB of speakers [32], bits of transmitted traffic [29].

Relative quantities are also encountered in the listed studies, although rarely. Banerjee et al. [49] introduce a measure reflecting the energy efficiency of the application for a certain time period. Dong, Lan and Zhong [41] measure energy consumption in percentages relative to the total energy consumption of the system.

Jung, Kim and Cha [45] organize their measuring process using both absolute and relative quantities. The percentage of screen brightness, the percentage of processor utilization and its frequency, the strength of Wi-Fi and 3G signals in dBm were measured along with many other characteristics of a device energy behavior.

Overall, about 65% of the listed studies report a single quantity, in other studies two or more are reported.

TABLE V
MEASURED CHARACTERISTICS

Quantity	Number of papers	Accuracy distribution
Power	21	mW — 14, W — 6, μW — 1
Energy	22	J — 12, mJ — 7, kJ — 1, μJ — 1, nJ — 1
Current	3	A — 2, mA — 1
Voltage	2	V — 1, mV — 1
Electric charge	3	mAh — 2, mAms — 1

G. RQ7: How do frameworks deal with metering hardware frequency being considerably lower than CPU frequency?

This RQ was included at later stages of data extraction process as a reaction to the following phenomena.

Modern CPUs including systems on a chip like smartphones or tablets operate at frequencies of several GHz. While the operating voltage for a CPU is more or less constant[5], the current it draws can vary tremendously. When multiple cores are working, it is higher than in the case of a single core working. Entire CPU or even each of its cores may potentially work at different frequencies with different corresponding current, changing dynamically — this process is called Dynamic Voltage Frequency Scaling (DVFS). Such changes in drawn current may appear at frequencies of at least MHz range.

On the contrary metering hardware in the listed works operates at considerably lower frequencies. Maximum frequency of 100 KHz is found in Wilke et al. [9], [10], but the bulk of studies uses multimeters operating way below 10 KHz. As amperemeters average variable current between synchronization impulses, power spikes of higher frequencies may be smoothed out to levels of measurement errors and pass undetected. Therefore it is important to understand if studies dealing with direct measurement approach frameworks or direct measurement baseline comparison alter their methodology to address this issue.

We've identified 10 frameworks that in one way or another acknowledge it:

- Mittal et al. [5], Dolezal and Becvar [32], Saksonov [26], Pandiyan and Wu [28] turn off DVFS completely or set it to a more controlled mode of operation, so CPU current draw is at least more predictable and consistent.
- Hung et al. [38], Yoon et al. [6], Zhang et al. [7] include CPU frequency data as an additional input for total energy consumption estimation.
- Couto et al. [11], Li and Gallagher [18] adjust test running time to be comparable to metering hardware frequency.
- Larsson and Stigelid [34] adjust metering frequency to test running time. We include this study with caution though, as authors access Android API and not an internal meter itself, therefore frequency of API calls might be unrelated to frequency of internal meter readings.

We conclude that the issue is not widely acknowledged, but it can be alleviated at least to an extent both by configuring the test device and by properly setting up experiment. We suggest that testing those measures for adequacy in a real-life multicore scenario with threads created and destroyed regularly and at high frequency is an open challenge.

IV. STUDY LIMITATIONS

One of the main issues when conducting SLR is to find all relevant studies to include. We used automatic search system (Google Scholar) and therefore our search is as efficient and thorough as this service is. While the query term "energy efficiency" allowed us to find a good amount of relevant studies, we admit that in some of the additionally included articles it was absent. There is a probability that we missed some other relevant studies not using this term in the text.

Additionally several months passed since our original search and article publication, so there is also a possibility of some newer articles published after summer 2019 being missed in our study.

It is also possible that we missed to group some articles, although we don't think there's a high probability to it. We assume our merge process was reliable enough as an idea to merge [16] and [17] into a single group came initially from their common institution of origin despite completely different sets of authors.

Another limitation was a selective quality control of a data extraction process. In some cases particular studies were handled by a single researcher with extracted data being well-written and not raising questions, so other researchers relied on this data instead of original study. Such process could introduce errors in extracted data due to misunderstanding of a study.

[5]For Li-ion and Li-pol batteries commonly used in modern smartphones voltage drops slightly between 90% and 20% of charge, but it is largely insignificant drop. Electronic devices can in fact safely operate in a range of voltages, so only a significant drop in voltage after 20% of charge results in functional degrade.

V. Classification of some commercial power profilers

As we prepared this article, a frequently asked question was how to relate results obtained from SLR to a number of commercially available power profilers used in practical Android development. While comparative accuracy overview is out of the scope of this article, we think it is useful to apply framework classification obtained in RQ1 to those profilers. Our goal here is to help practitioners better understand their intended use cases. The list is definitely not exhaustive, but those were the most frequently mentioned profilers.

Trepn Power Profiler [58] is a tool developed by Qualcomm which is no longer supported. It is a direct measurement framework using internal meter, and therefore its energy consumption data is as accurate as the meter itself is. This issue was acknowledged by developers and a list of accurate devices was compiled [59].

BatteryHistorian [60] is a tool for background system information aggregation in Android OS. Not only it provides power drain information but also statistics of runtime system events like Wi-Fi scans or wakelocks. Being classified as a direct measurement framework with internal meter, it displays energy consumption as percentage of battery charge which is a victim to battery degradation. A noticable drop in battery charge usually requires a significant amount of computations and/or peripherals work, therefore this value is not suitable for short time experiments, for example, those that run only for several seconds.

Android Studio Energy Profiler [61] shows energy consumption as a real-time graph in relative units. Regretfully, there is virtually no documentation on the model itself, but after analyzing the information obtained from various sources (such as StackOverflow [62]) and experimenting with tool itself we came to the conclusion it is a working time model-based solution. It aggregates various system events, assigns them weights and gives a resulting relative power drain metric. While it is granular enough to be used for energy refactorings efficiency estimation, we have concerns if its "one-size-fits-all" model is representative of real smart devices hardware energy consumption.

VI. Conclusions and Future Work

The results of our study indicate that there are various approaches to measure application energy consumption in Android OS with some frameworks utilizing external equipment to gather necessary data and others using previously calibrated models. While there are merits to all of those approaches it is difficult to compare framework accuracy with one another based on the studies themself as there is no uniform accuracy estimation methodology presented in the listed publications.

Even if one is going to compare frameworks experimentally, only a handful of studies have their source code or sample application openly available which considerably limits framework representation for such analysis. What's worse, code repositories look mostly abandoned, and framework start-up documentation is scarce. We conclude that under these circumstances accuracy comparison between approaches is currently extremely hard to undertake.

On a brighter side, these frameworks are reported to be used. To our knowledge only a PowerTutor framework [7] was able to evolve further and became a broadly used application, while some of the frameworks were documented to be used internally as a tools for higher level research projects [21], [11].

A number of devices supported by existing frameworks is large, so even if a practitioner would write a framework itself, test device itself is not a limiting factor. A set of methodological practices and techniques was established to improve measurement accuracy, and even relatively low frequency of hardware equipment compared to measured device can be offset by smart experimental design, although we think that there are still open questions in this area.

We've also shown that our proposed classification is useful to analyze possible use cases for commercially available power profilers targeting Android OS.

Results of our study affected design decisions and helped to continue developent of our own tool for energy consumption metering for Android OS - a working time model-based Navitas framework[6]. From the beginning it was conceived as an open-sourced framework which could be used in different scenarios, and one of possible applications we're developing is a plugin for Android Studio IDE. Its development, evaluation and application for energy refactorings is a focus for future work.

Acknowledgment

Authors would like to thank Lanit-Tercom company for hosting our research project as part of its Summer School'19.

References

[1] Marko Milijic, "29+ Smartphone Usage Statistics: Around the World in 2020," https://leftronic.com/smartphone-usage-statistics/, 2019, [Online; accessed 14-January-2020].

[2] C. Sahin, F. Cayci, I. Manotas, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh, "Initial explorations on design pattern energy usage," *2012 1st International Workshop on Green and Sustainable Software, GREENS 2012 - Proceedings*, 06 2012.

[3] S. O'Dea, "Share of global smartphone shipments by operating system from 2014 to 2023," https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/, 2019, [Online; accessed 14-January-2020].

[4] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," vol. 2, 01 2007.

[5] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 317–328. [Online]. Available: https://doi.org/10.1145/2348543.2348583

[6] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for android smartphones using kernel activity monitoring," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12. USA: USENIX Association, 2012, p. 36.

[6] `https://github.com/Stanislav-Sartasov/Navitas-Framework/`

[7] L. Zhang, B. Tiwana, R. P. Dick, Z. Qian, Z. M. Mao, Z. Wang, and L. Yang, “Accurate online power estimation and automatic battery behavior based power model generation for smartphones,” in *2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2010, pp. 105–114.

[8] X. Li and J. P. Gallagher, “Fine-grained energy modeling for the source code of a mobile application,” in *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MOBIQUITOUS 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 180–189. [Online]. Available: https://doi.org/10.1145/2994374.2994394

[9] C. Wilke, S. Götz, and S. Richly, “Jouleunit: A generic framework for software energy profiling and testing,” in *Proceedings of the 2013 Workshop on Green in/by Software Engineering*, ser. GIBSE ’13. New York, NY, USA: Association for Computing Machinery, 2013, p. 9–14. [Online]. Available: https://doi.org/10.1145/2451605.2451610

[10] C. Wilke, “Energy-aware development and labeling for mobile applications,” Ph.D. dissertation, 03 2014.

[11] M. Couto, J. Cunha, J. P. Fernandes, R. Pereira, and J. Saraiva, “Greendroid: A tool for analysing power consumption in the android ecosystem,” in *2015 IEEE 13th International Scientific Conference on Informatics*, Nov 2015, pp. 73–78.

[12] M. Couto, “Monitoring energy consumption in android applications,” 2014.

[13] M. Couto, T. Carção, J. Cunha, J. Fernandes, and J. Saraiva, “Detecting anomalous energy consumption in android applications,” 10 2014, pp. 77–91.

[14] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, “Petra: A software-based tool for estimating the energy profile of android applications,” in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 3–6.

[15] ——, “Software-based energy profiling of android apps: Simple, efficient and reliable?” in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Feb 2017, pp. 103–114.

[16] M. Feghhi, “Multi-layer tracing of android applications for energy-consumption analysis,” 2017.

[17] K. Aggarwal, C. Zhang, J. C. Campbell, A. Hindle, and E. Stroulia, “The power of system call traces: Predicting the software energy consumption impact of changes,” in *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON ’14. USA: IBM Corp., 2014, p. 219–233.

[18] X. Li and J. Gallagher, “An energy-aware programming approach for mobile application development guided by a fine-grained energy model,” 05 2016.

[19] R. W. Ahmad, A. Naveed, J. J. P. C. Rodrigues, A. Gani, S. A. Madani, J. Shuja, T. Maqsood, and S. Saeed, “Enhancement and assessment of a code-analysis-based energy estimation framework,” *IEEE Systems Journal*, vol. 13, no. 1, pp. 1052–1059, March 2019.

[20] R. Ahmad, A. Gani, S. h. Ab hamid, A. Naveed, K. Ko, and J. Rodrigues, “A case and framework for code analysis-based smartphone application energy estimation,” *International Journal of Communication Systems*, vol. 30, 11 2016.

[21] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, “Greenminer: A hardware based mining software repositories software energy consumption framework,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 12–21. [Online]. Available: https://doi.org/10.1145/2597073.2597097

[22] M. Tuysuz, M. Uçan, and R. Trestian, “A real-time power monitoring and energy-efficient network/interface selection tool for android smartphones,” *Journal of Network and Computer Applications*, vol. 127, 11 2018.

[23] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, “Estimating android applications’ cpu energy usage via bytecode profiling,” in *2012 First International Workshop on Green and Sustainable Software (GREENS)*, June 2012, pp. 1–7.

[24] U. Bareth, “Simulating power consumption of location tracking algorithms to improve energy-efficiency of smartphones,” in *2012 IEEE 36th Annual Computer Software and Applications Conference*, July 2012, pp. 613–622.

[25] T. Kamiyama, H. Inamura, and K. Ohta, “A model-based energy profiler using online logging for android applications,” in *2014 Seventh International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, Jan 2014, pp. 7–13.

[26] A. Saksonov, “Method to derive energy profiles for android platform,” 2014.

[27] H. Sahar, A. Bangash, and M. Beg, “Towards energy aware object-oriented development of android applications,” *Sustainable Computing: Informatics and Systems*, vol. 21, 11 2018.

[28] D. Pandiyan and C. Wu, “Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms,” in *2014 IEEE International Symposium on Workload Characterization (IISWC)*, Oct 2014, pp. 171–180.

[29] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, “A close examination of performance and power characteristics of 4g lte networks,” in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 225–238. [Online]. Available: https://doi.org/10.1145/2307636.2307658

[30] W. Oliveira, R. Oliveira, F. Castor, B. Fernandes, and G. Pinto, “Recommending energy-efficient java collections,” in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, May 2019, pp. 160–170.

[31] B. Westfield and A. Gopalan, “Orka: A new technique to profile the energy usage of android applications,” in *2016 5th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, April 2016, pp. 1–12.

[32] J. Dolezal and Z. Becvar, “Methodology and tool for energy consumption modeling of mobile devices,” in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2014, pp. 34–39.

[33] Y. Hu, J. Yan, D. Yan, Q. Lu, and J. Yan, “Lightweight energy consumption analysis and prediction for android applications,” *Science of Computer Programming*, vol. 162, 05 2017.

[34] M. Larsson and M. Stigelid, “Energy efficient data synchronization in mobile applications : A comparison between different data synchronization techniques,” Ph.D. dissertation, 08 2015.

[35] L. M. Fischer, L. B. d. Brisolara, and J. C. B. d. Mattos, “Sema: An approach based on internal measurement to evaluate energy efficiency of android applications,” in *2015 Brazilian Symposium on Computing Systems Engineering (SBESC)*, Nov 2015, pp. 48–53.

[36] S. Lee, C. Yoon, and H. Cha, “User interaction-based profiling system for android application tuning,” in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 289–299. [Online]. Available: https://doi.org/10.1145/2632048.2636091

[37] X. Chen and Z. Zong, “Android app energy efficiency: The impact of language, runtime, compiler, and implementation,” in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, Oct 2016, pp. 485–492.

[38] S. Hung, F. Liang, C. Tu, and N. Chang, “Performance and power estimation for mobile-cloud applications on virtualized platforms,” in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July 2013, pp. 260–267.

[39] A. Carette, M. A. A. Younes, G. Hecht, N. Moha, and R. Rouvoy, “Investigating the energy impact of android smells,” in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Feb 2017, pp. 115–126.

[40] K. Kapetanakis and S. Panagiotakis, “Efficient energy consumption’s measurement on android devices,” in *2012 16th Panhellenic Conference on Informatics*, Oct 2012, pp. 351–356.

[41] M. Dong, T. Lan, and L. Zhong, “Rethink energy accounting with cooperative game theory,” in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 531–542. [Online]. Available: https://doi.org/10.1145/2639108.2639128

[42] S. Lee, W. Jung, Y. Chon, and H. Cha, “Entrack: a system facility for analyzing energy consumption of android system services,” 09 2015, pp. 191–202.

[43] Y. Chung, C. Lin, and C. King, “Aneprof: Energy profiling for android java virtual machine and applications,” in *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, Dec 2011, pp. 372–379.

[44] Donghwa Shin, Kitae Kim, Naehyuck Chang, Woojoo Lee, Yanzhi Wang, Qing Xie, and M. Pedram, “Online estimation of the remaining energy capacity in mobile systems considering system-wide power consumption and battery characteristics,” in *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2013, pp. 59–64.

[45] W. Jung, K. Kim, and H. Cha, “Userscope: A fine-grained framework for collecting energy-related smartphone user contexts,” in *2013 International Conference on Parallel and Distributed Systems*, Dec 2013, pp. 158–165.

[46] S. Tsao, C. Kao, I. Suat, Y. Kuo, Y. Chang, and C. Yu, “Powermemo: A power profiling tool for mobile devices in an emulated wireless environment,” in *2012 International Symposium on System on Chip (SoC)*, Oct 2012, pp. 1–5.

[47] G. Metri, “Energy efficiency analysis and optimization for mobile platforms,” 2014.

[48] X. Gao, D. Liu, D. Liu, H. Wang, and A. Stavrou, “E-android: A new energy profiling tool for smartphones,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 492–502.

[49] A. Banerjee, L. K. Chong, S. Chattopadhyay, and A. Roychoudhury, “Detecting energy bugs and hotspots in mobile apps,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 588–598. [Online]. Available: https://doi.org/10.1145/2635868.2635871

[50] D. Li, C. Sahin, J. Clause, and W. G. J. Halfond, “Energy-directed test suite optimization,” in *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, May 2013, pp. 62–69.

[51] K. Walcott-Justice, “Maue: A framework for detecting energy bugs from user interactions on mobile applications,” 2016.

[52] H.-J. Kim, J. Kyong, and S.-S. Lim, “A systematic power and performance analysis framework for heterogeneous multiprocessor system,” *Journal of IEMEK*, vol. 9, pp. 315–321, 12 2014.

[53] “Power Profiles for Android,” https://source.android.com/devices/tech/power, 2019, [Online; accessed 14-January-2020].

[54] Monsoon Solutions, Inc., “High voltage power monitor,” https://www.msoon.com/online-store, 2019, [Online; accessed 18-August-2019].

[55] “Android 10 for Developers,” https://developer.android.com/about/versions/10/highlights, 2019, [Online; accessed 14-January-2020].

[56] “Application forward compatibility.” [Online]. Available: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#fc

[57] “Distribution dashboard.” [Online]. Available: https://developer.android.com/about/dashboards

[58] “Trepn Power Profiler,” https://developer.qualcomm.com/forums/software/trepn-power-profiler, 2019, [Online; accessed 14-January-2020].

[59] “Which mobile devices report accurate system power consumption?” https://developer.qualcomm.com/forum/qdn-forums/software/trepn-power-profiler/28349, 2013, [Online; accessed 14-January-2020].

[60] “Analyze power use with Battery Historian,” https://developer.android.com/topic/performance/power/battery-historian, 2020, [Online; accessed 14-January-2020].

[61] “Inspect energy use with Energy Profiler,” https://developer.android.com/studio/profile/energy-profiler, 2019, [Online; accessed 14-January-2020].

[62] “Energy consumption on Android Studio Profiler,” https://stackoverflow.com/questions/52647045/energy-consumption-on-android-studio-profiler, 2018, [Online; accessed 14-January-2020].