Journal of Network and Computer Applications 58 (2015) 42–59

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

A Review on mobile application energy profiling: Taxonomy, state-of-the-art, and open research issues

Raja Wasim Ahmad [a,*], Abdullah Gani [a], Siti Hafizah Ab. Hamid [a], Feng Xia [b], Muhammad Shiraz [a]

[a] Center for Mobile Cloud Computing Research (C4MCCR), Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
[b] School of Software, Dalian University of Technology, China

ARTICLE INFO

Article history:
Received 14 April 2015
Received in revised form
11 August 2015
Accepted 11 September 2015
Available online 21 September 2015

Keywords:
Energy profiling
Rich mobile applications
Context-aware
Energy estimation

ABSTRACT

The shift of the information access paradigm to a mobile platform motivates research in mobile application energy profiling to augment device battery lifetime. Energy profiling schemes estimate mobile application power consumption when it is executed on resource-constrained mobile devices. Accurate power estimation helps identify rogue applications to optimize mobile battery power usage. The lack of a comprehensive survey on mobile application energy profiling schemes that covers various energy profiling aspects, such as profiling granularity, types, measurement resources, and model flexibility, has motivated us to review the existing literature comprehensively. Application energy profiling schemes exploit either hardware-equipment or software-based solutions to track battery-draining behavior during application execution in mobile devices. This study comprehensively reviews state-of-the-art mobile application energy profiling schemes to investigate the strengths and weaknesses of existing schemes. We propose a detailed thematic taxonomy based on the extensive literature review on mobile application energy profiling to classify the existing literature. The critical aspects and related features of existing energy profiling schemes are examined through an exhaustive qualitative analysis. The significant parameters from the reported literature are also extracted to investigate commonalities and differences among existing schemes. Finally, several research issues in mobile application energy profiling are put forward that should be addressed to increase energy profiling strength.

Contents

* Corresponding author. Tel.: +60 107891697; fax: +60 379579249.
E-mail addresses: wasimraja@siswa.um.edu.my (R.W. Ahmad), abdullah@um.edu.my (A. Gani), sitihafizah@um.edu.my (S.H.Ab. Hamid), f.xia@ieee.org (F. Xia), muh_shiraz@yahoo.com (M. Shiraz).

http://dx.doi.org/10.1016/j.jnca.2015.09.002
1084-8045/

1. Introduction

The ever-increasing energy demands of innovative mobile applications reduce battery lifetime because of inadequate advancements in contemporary mobile phone battery design (Alagöz et al., 2014). Recent mobile phone applications, such as video on demand (Abolfazli et al., 2014b), mobile-gaming (Hsieh et al., 2014), location-aware social networking (Durand et al., 2011), real-time pedestrian tracking (Yoon et al., 2012), and context-aware advertisement services (Giammarini et al., 2011), are the most energy consuming and resource-intensive services. However, while serving these applications on the resource constrained mobile phones, a noteworthy amount of processor, network bandwidth, and storage capacity, is needed that rapidly depletes the battery-charge. Nowadays, mobile application development is swiftly expanding because users prefer to continue their social, entertainment, and business activities while on the go. According to *Forrester* report, the mobile application development market will detonate exponentially to $38 billion by 2015 because of substantial growth in the popularity of the mobile phones (Donohoo et al., 2011). However, despite the tremendous growth in the application market of mobile phones, their usage remains limited by battery life-time, size, and intermittent wireless connectivity. The lifetime of a battery can be improved by optimizing the application design of mobile phones. The key enabler to mobile application energy optimization is energy profiling, which pinpoints the contribution of individual application components to the total energy consumption budget (Alagöz et al., 2014).

Application energy profiling characterizes the energy consumption behavior of a mobile application while executing it on resource constrained mobile devices. Energy profiling schemes exploit the power models of mobile components (e.g., CPU, Wi-Fi, LCD, and 3G) to estimate mobile application power consumption (Alagöz et al., 2014). The design of power models is based on either software solutions or hardware equipment (e.g., power meter) to characterize the power drawn behavior of mobile applications during their execution on mobile phone (Abolfazli et al., 2014b; Hsieh et al., 2014). Application energy profiling facilitates in (a) identifying rogue applications (Durand et al., 2011), (b) diagnosing system energy consumption (Yoon et al., 2012), (c) estimating per-application energy usage (Giammarini et al., 2011), (d) optimizing application energy usage, and (e) designing an energy-aware application scheduler (Lin et al., 2013; Navda et al., 2013; Papalkar et al., 2014; Van Beeck et al., 2011). Based on the energy requirements of applications, a CPU reconsiders an application execution schedule to augment mobile battery lifetime. Similarly, for pedestrian tracking applications, increasing delay-interval among position updates adaptively surges battery lifetime (Geronimo et al., 2010).

To best of our knowledge, there exists only one survey on mobile application energy profiling schemes. However, reported survey (Cui et al., 2013) is too generic and has only considered wireless communication power consumption methods with special emphasis on Wi-Fi power management. Moreover, authors neither analysed the existing profiling schemes nor discussed the potential future research directions. Major contribution of this study is to conduct a comprehensive literature review on state-of-the-art mobile application energy profiling schemes by considering several diversified aspects of energy profiling such as, profiling granularity, overhead, design pattern, and energy measurement methods. In this current study, state-of-the-art mobile application energy profiling schemes are critically reviewed and their strengths are identified. The weaknesses and issues that need further research in this domain of research are also identified. A novel thematic taxonomy for mobile application energy profiling schemes is proposed to classify the literature based on the common characteristics among existing schemes. The critical aspects and significant features of the mobile application energy profiling schemes are investigated through qualitative analysis. Finally, some open research issues in energy profiling are discussed in order to design an optimized energy profiler.

The organization of paper is as follows. Section 2 discusses background on mobile application energy profiling with special emphasis on recent mobile application features and energy models. Section 3 briefly discusses a thematic taxonomy on mobile application energy profiling schemes. Section 4 debates on state-of-the-art energy profiling schemes. Section 5 critically analyses the energy profiling schemes. Section 6 debates on some interesting open research issues in mobile application energy profiling domain. Section 7 concludes the whole paper and presents some future directions for further research in this domain of study.

2. Background

This section debates on mobile phone energy features and profiling models. Throughout this article, we used the keywords “mobile” and “smartphones” interchangeably to denote the resource constraint battery driven mobile phones. Furthermore, for the ease of readers, we provided a list of most frequently used acronyms in the article in Table 1.

2.1. Mobile energy features modeling

Recently, mobile phones have gained remarkable popularity in various computing rigorous domains (Dubey et al., 2014) such as, education (Haffey et al., 2014; Seop et al., 2011), management information system (Wisniewski et al., 2013; Yan et al., 2014), health monitoring (Marotz, 2014; Mosa et al., 2012), and enterprise applications (Hayes et al., 2013; Pandhare and Joglekar, 2011), due to its support for context-aware, portable, and multi-tier applications (Hsieh et al., 2014; Koo et al., 2013). However, enriching mobile application increases demand for light-sensors, GPS, compass, and accelerometers (Liu et al., 2015; Wang, 2011). As a result, power hungry mobile features such as, GPS, wireless radio, multi-core processor, and large-sized bright screens, deplete the battery capacity rapidly (Amini et al., 2013; Huang et al., 2010; Sanaei et al., 2014; Sanaei et al., 2013). Optimizing the power-hungry mobile features (components) demand accurate mobile components power estimation. However, mobile phone component's tight-integration inaccurate the individual mobile component power estimation. For instance, in the modern smart phones, the Bluetooth and Wi-Fi components are embedded on the same chip. Switching on the Wi-Fi radio turns on the entire chip which affects the single component power estimation accuracy. Hence, individual mobile component power consumption is influenced by complex engineering designs and architectures (Geronimo et al., 2010).

CPU frequency (e.g., 385 MHz and 246 MHz) and utilization level affect CPU power consumption during application execution on mobile phones. LCD power consumption is also affected by LCD screen size and brightness level (e.g., low or high). However, the total power consumed by a GPS component depends on the GPS communication mode (e.g., active, sleep, or off), the number of satellites in the GPS vicinity, and signal strength. Among all mobile components, Wi-Fi and cellular interfaces dominate energy consumption. Wi-Fi power consumption budget depends on four Wi-Fi power states, namely, low power, high power, low transmit (l transmit), and high transmit (h transmit). During low-power state, a Wi-Fi module does not send or receive data at a high rate (defined by a predefined threshold). The Wi-Fi interface triggers state transitions between low- and high-power states based on l- and h-transmit data rates. Cellular power consumption is a factor in the power states of radio, including IDLE, CELL-DCH, and CELL-FACH. During CELL-DCH and CELL-FACH states, a mobile phone employs dedicated and shared channels to communicate with the base station. During IDLE mode, however, the radio interface consumes minimum power because it only receives paging messages in this state (Kumar and Lu, 2010; Vallina-Rodriguez and Crowcroft, 2013; Wigley and Shantikumar, 2013). The power model of a mobile component analyzes the aforementioned attributes to quantify application energy consumption.

Mobile application energy profiling schemes utilize the power models of mobile components to estimate application power consumption at diversified granularities, such as process, thread, or execution path. The fundamental power models that provide a foundation for application energy estimation are presented in Eqs. (1)–(4) (Do et al., 2009). In general, energy estimation models foresee application power consumption through its resource utilization (U_{ab}) and the inter resource interaction energy model during application execution on mobile phones as illustrated in Eq. (1). Among system components, CPU and network interface (i.e., Wi-Fi and 3G) are the utmost power hungry entities within a mobile phone. Eqs. (2) and (3) highlight the fundamental elements of a CPU and network power models. For the CPU component, total power budget depends on CPU execution time for the running process (t_i), transitions among CPU states (n_j), and energy consumption during each transition (E_j), as presented in Eq. (2). The knowledge of network activity duration ($t_{sending}$ and $t_{receiving}$) and mobile battery power drawn rate ($p_{sending}$ and $p_{receiving}$) during such activity estimates network component power consumption during a specific period as defined in Eq. (3) (Alagöz et al., 2014; Do et al., 2009). In comparison with CPU and network models, Eq. (3) presents a complete system power model that comprises mobile components (e.g., CPU, Wi-Fi, GPS, and LCD) and system variables (utilization, CPU_on, Wi-Fi, and brightness level) that obviously contribute to total power consumption (Alagöz et al., 2014).

$$E_{Application} = \sum U_{ab} \times E_{resources} + E_{interaction} \tag{1}$$

$$E_{CPU-Resource} = \sum_i P_i t_i + \sum_j n_j E_j \tag{2}$$

$$E_{Net-Resourcei} = t_{sendingi} \times p_{sending} + t_{receivingi} \times P_{receiving} \tag{3}$$

$$\begin{aligned}(\beta_{\mathrm{uh}} \times \mathrm{freq}_h + \beta_{\mathrm{ul}} \times \mathrm{freq}_l) \times \mathrm{Utilization} + \beta_{\mathrm{CPU}} \times \mathrm{CPU}_{\mathrm{on}} \\ + \beta_{\mathrm{br}} \times \mathrm{brightness\_level} + \beta_{\mathrm{Gon}} \times \mathrm{GPS}_{\mathrm{on}} + \beta_{\mathrm{Gsl}} \times \mathrm{GPS}_{\mathrm{sl}} \\ + \beta_{\mathrm{Wi-}F\_l} \times \mathrm{Wi\_Fi}_l + \beta_{\mathrm{Wi-}F\_h} \times \mathrm{Wi\_Fi}_h + \beta_{\mathrm{3G\_idle}} \times 3G_{\mathrm{idle}} \\ + \beta_{\mathrm{3G\_FACH}} \times 3G_{\mathrm{FACH}} + \beta_{\mathrm{3G\_DCH}} + 3G_{\mathrm{DCH}}\end{aligned} \tag{4}$$

Table 1
List of acronyms.

Symbol	Description
Wi-Fi	Wireless Fidelity
NEP	Nokia energy profiler
ARO	Application resource optimizer
GPS	Global Positioning System
LCD	Liquid Crystal Display
SOC	State-of-charge
DuT	Device under test
DAQ	Data acquisition
SEMO	Smart energy monitoring system
PCA	Principal component analysis
ACPI	Advanced Configuration and Power Interface
RLS-ED	Linear regression with exponential decay
GA	Genetic algorithm
RRC	Radio resource controller
SBS	Smart battery system
BMU	Battery monitoring unit
DVFS	Dynamic voltage frequency scaling
GSM	Global System for Mobiles
DHCP	Dynamic Host Configuration Protocol
HTC	High-Tech Computer Corporation
OS	Operating system

2.2. Mobile applications energy profiling

Mobile application energy profiling schemes either employ existing power models or build their own to characterize the energy consumption behavior of mobile applications. The power model constitutes mathematical equations that quantify the effect of several factors, such as usage level, component state, usage time, and inter-component dependency on the power consumption of mobile component (Alagöz et al., 2014). Power modeling designs utilize hardware, operating system (OS), or application traces to define variables that constitute power models (Duan et al., 2013; Sasu Tarkoma et al., 2014; Zhang et al., 2010). Furthermore, the co-efficient of variables is derived based on deterministic or statistical power measurement models (Hao et al., 2013; König et al., 2013; Qian et al., 2011). During application

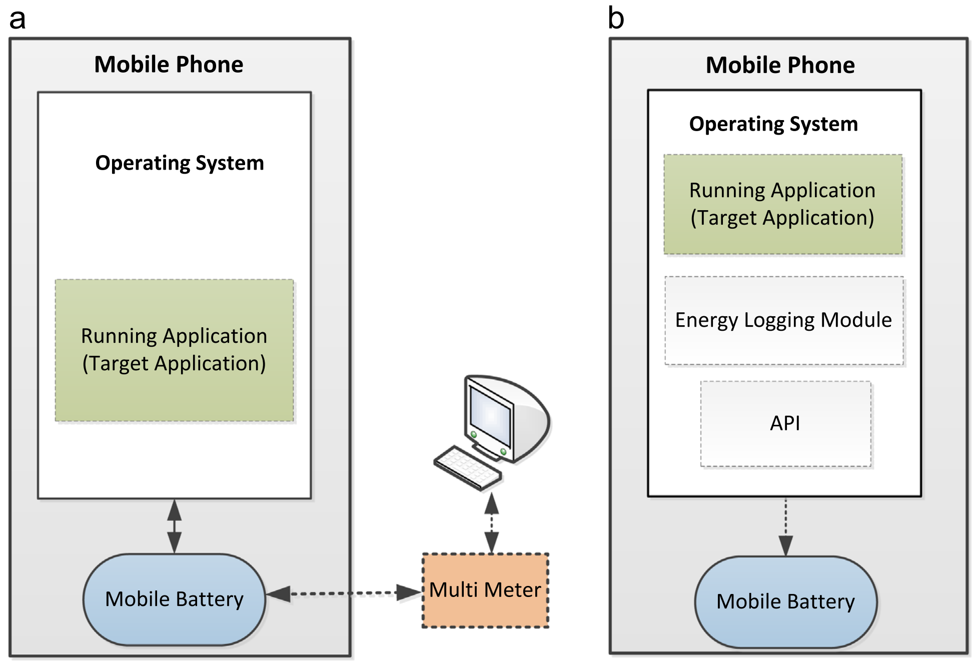

Fig. 1. Hardware based vs. software based energy profiling.

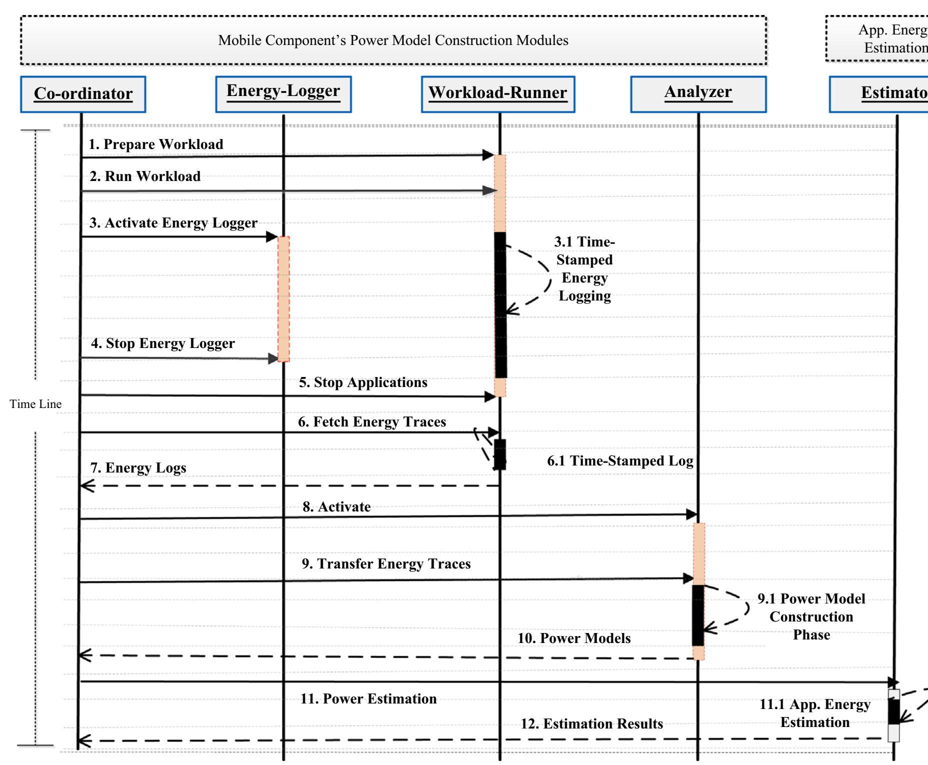

Fig. 2. Software based mobile application energy profiling sequence diagram.

execution on mobile phones, the energy profiler logs the execution traces and energy footprints in either on-line (Lee et al., 2014) or off-line (Brouwers et al., 2014; Conte et al., 1996) modes, as presented in Fig. 1. The off-line profiling mode uses a physical server to correlate energy statistics with mobile phone-derived data. By contrast, online profiling does not entail additional external hardware resource for energy profiling, as depicted in Fig. 1 (Bernstein et al., 2013; Creus and Kuulusa, 2007; Das et al., 2013; Lauridsen et al., 2014; Oliner et al., 2013; Ryan et al., 2014; Yoon et al., 2012).

Figure 1 demonstrates design of hardware and software-based energy profiling schemes. Hardware-based energy profiling paradigm, as depicted in Fig. 1a, exploits external power measurement equipment (e.g., multi-meter) to determine power drop-rate across the battery terminals. Instead, software-based energy profiling schemes use smart battery interface to capture the battery discharge-rate as illustrated in Fig. 1b. Furthermore, the post-processing on power logs assist to construct power models as discussed in Fig. 2. Figure 2 determines a conceptual overview of software-based application profiling mechanism, wherein, power models are generated based on power log analysis. In the said figure, coordinator module control and triggers the rest of the modules to perform specific tasks. Energy logger continuously records time-stamped power consumption statistics during mobile application execution. In addition, Analyzer thoroughly inspects energy-traces to build power models for application power estimation.

3. Taxonomy of mobile application energy profiling schemes

This section highlights and discusses a thematic taxonomy for the classification of mobile application energy profiling schemes. We categorized the existing energy profiling schemes based on their design pattern. Energy profiling schemes are classified into two main categories: (a) software based and (b) hardware based. The software and hardware based energy profiling schemes are further categorized based on the common characteristics among existing schemes. Common characteristics among software-based energy profiling schemes include, granularity, model flexibility, measurement source, and profiling type. Alternatively, the common characteristics among hardware-based energy profiling schemes include, granularity, power model design, measurement source, and execution environment, as illustrated in Fig. 3.

3.1. Software based profiling

Software-based energy profiling paradigm exploits a software module to collect mobile component's power usage statistics to construct power models to estimate application's energy consumption. The attribute of granularity identifies the level at which energy profiler estimates the mobile application's energy consumption. The energy profiler estimates the application energy consumption either at process, thread, path, or source-code line level, as shown in Fig. 3. The model flexibility parameter specifies whether the energy profiler requires application source-code to profile application energy consumption or not. Alternatively, the attribute of the measurement source specifies the method opted to extract power consumption statistics to estimate the application energy consumption. The profiling type metric specifies whether the mobile application energy profiler exploits a dedicated physical server to analyse the collected power consumption statistics to construct power models or uses mobile phone resources to analyse the logged energy statistics. The detailed description of aforementioned attributes is as given below.

Granularity indicates the extent to which an energy profiler estimates the power consumption of an application. The attributes of granularity include mobile application, application execution path, thread, source-code line, function, burst, application component, and process. Fine-granular (e.g., source-code line and path) estimations result in high estimation accuracy compared with coarse-granular estimations. However, fine granularity requires the following: (a) extensive resource monitoring, (b) high resource synchronization between application activities and the updating rate of smart battery sensors, and (c) elongated mobile application profiling time. Coarse-granular profiling (e.g., process, thread, and function) requires the following: (a) low resource monitoring, (b) limited profiling time, and (c) insufficient resource utilization. Estimating power consumption at application granularity supports ranking applications based on their power consumption budget. Similarly, thread/function level energy estimation provides the opportunity to optimize thread/function power consumption to increase battery lifetime. Moreover, application component level energy estimation such as GUI, logic component, and content rendering, assists in optimizing application power consumption at the sovereign component level. For example, a gaming application consumes considerable energy while rendering graphical objects. Optimizing gaming rendering feature can also significantly augment mobile battery lifetime. Among others, path attribute denotes the set of routines traversed during an application activity (e.g., playing a video track). Burst attribute identifies consumed energy while continuously transferring a bunch of network packets using smartphone radio. When a user wishes to estimate energy consumption for a specific use case of a

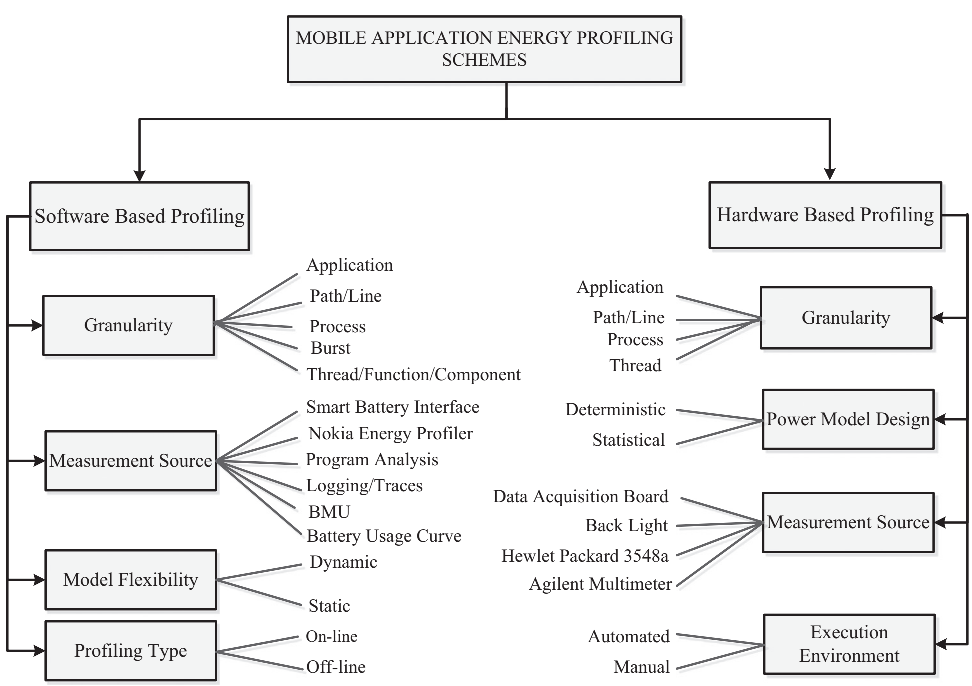

Fig. 3. Taxonomy of mobile application energy profiling schemes.

mobile application, the energy profiler exploits path attribute to identify the traversed set of routines and assigns a unique ID (to the path) to differentiate it.

The attribute of *measurement source* highlights the measurement mode selected by mobile application energy profiling schemes to access the resource utilization of the application and the power consumption statistics of the smartphone component. The attributes of resource measurement include smart battery interfaces, Nokia energy profiler (NEP), battery monitoring unit (BMU), program analysis, energy logs, and battery discharge curve. NEP (Damaševičius et al., 2013) examines the power consumption of mobile applications using the application programming interface (API) for the Symbian mobile series. It incessantly observes various mobile phone components such as, CPU usage, WLAN/cellular signal power, and uplink/downlink channel rates, using voltage and current sensors during the training phase to profile energy consumption. However, running NEP on mobile phones consumes a significant amount of power and introduces additional noise to the original measurement if quantities are improperly measured. Similarly, smart-battery measurements are imprecise because the readings of voltage and current sensors are inaccurate; moreover, such sensors demand a long profiling time that rapidly depletes battery charge. Furthermore, battery discharge curve-based power measurement is marginally inaccurate and non-scalable because of the divergence in discharge curve designs for each type of smartphone. The behavior of battery discharge curve differs with the change in the smartphone internal temperature. To solve this issue, BMU (Gurun and Krintz, 2006) devices are available within smart phones to monitor, (a) voltage, (b) temperature, and (c) current against system activities. However, the power consumption readings provided by these devices are coarse grained. Given the limited power update rate offered by BMU devices, determining the power states of mobile features is difficult. Kernel instrumentation (Do et al., 2009) and logging based profiling (Oliver and Keshav, 2011; Pathak et al., 2012) solutions exploit probes that run at kernel level and monitors system activities before transferring time-stamped traces to a remote server for analysis to construct power models. Compared with kernel instrumentation, program analysis (Hao et al., 2013) based energy profiling is time-consuming and requires longtime smartphone monitoring to calculate paths for each activity of the mobile application. However, program analysis methods are fine grained compared with the aforementioned attributes because they exploit per-instruction power profile for estimation.

Mobile application developers integrate desired application features into the energy profiler to estimate application power consumption at diversified granularities, including process, function, block, thread, and individual application components. The *model flexibility* parameter stipulates whether the energy profiler necessitates the source code of the mobile application or it works with an executable file to estimate the energy consumption of an application. A static profiling attribute specifies that the energy profiler requires the application source code to estimate energy consumption. By contrast, a dynamic attribute indicates that the energy profiler utilizes the symbol table to generate processes and functions and does not require the application source code. Static attribute requires the support of the application developer while profiling the energy consumption of mobile applications. Static energy profiling is also not scalable because it requires abundant system resources to compile applications. Dynamic application energy profiling is fast and scalable; moreover, it eliminates the necessity for programmer support during application energy profiling.

Software-based energy profiling continuously examines the power drain rate (in terms of mobile component usage) of an application to construct smartphone power models. The power model construction life cycle consists of two stages, namely, power measurement collection and analysis. Both stages are highly time-consuming and computationally expensive. Mobile application energy profiling schemes are categorized into off-line and online classes from the perspective of the *profiling type* parameter. An off-line attribute specifies that a dedicated server is used to control and analyze the collected power consumption measurements (against system activities) to generate power models. An off-line attribute augments smartphone battery lifetime because analysis requires adequate system resources. However, employing off-line profiling is expensive because it requires an additional dedicated physical hardware resource. Being non-scalable, off-line profiling constructs power models in non-real-time execution mode. Meanwhile, an online attribute states that the resources of mobile phones can be utilized to analyze (preprocessing) power statistics to construct power models. However, employing the online profiling mode rapidly depletes battery capacity. Online analysis also takes more time than off-line analysis because servers are computationally fast compared with smartphones. Online energy profiling methods are scalable because of zero dependency on the external server during the energy profiling process.

3.2. Hardware based profiling

Hardware-based mobile application energy profiling utilizes external hardware equipment (e.g., power meter, multi-meter) to obtain voltage and current readings to estimate the power consumed by a mobile phone against system activities. The *fine-granular* power estimation methodology augments application power estimation accuracy. Figure 3 illustrates that hardware-based energy profiling estimates application power consumption at diverse granularities, including (a) system call (Cignetti et al., 2000), (b) process (Flinn and Satyanarayanan, 1999), (c) function, and (d) web components (Thiagarajan et al., 2012). The attributes of the *power model design* specify the method selected to investigate the energy consumption of a mobile hardware. The deterministic power model design approximates mobile power consumption based on the power state machine (PSM). The PSM modeling method also estimates mobile power consumption by employing the per-state energy model and hardware component transition states. The deterministic power model design maximizes the (a) OS device drivers, (b) system call traces, and (c) measured workload logs to study mobile power consumption behavior. Statistical power modeling uses pre-built statistical models, such as linear regression, to estimate software/mobile power consumption.

M*easurement source* identifies the hardware-equipment taken by the mobile application energy profilers to estimate the application power consumption. Power models quantify the impact of diversified factors on the power consumption behavior of smart phone using mathematical models. The attributes of *execution environment* states whether the application-testing framework design is automated or manual. The manual execution environment attribute (Carroll and Heiser, 2010) requires high user interaction as a user has to intermingle with the mobile phone in order to adjust different mobile settings such as, (a) switching on the components, (b) switching off the Wi-Fi connection, (c) pausing executing applications, and (d) adjusting LCD brightness levels. Alternatively, the automated execution environment is ideal as it program the hardware/software entities to automatically adjust the required settings without requiring human intervention. Furthermore, the readings from automated execution environment are more reliable as a user can run same test-case several times. Moreover, automated execution environment attribute quickly logs the application energy consumption as delays during physical adjusting the system setting are exterminated.

4. State-of-the-art mobile application energy profiling schemes

This section briefly discusses state-of-the-art mobile application energy profiling schemes.

4.1. Software based application energy profiling schemes

4.1.1. Power-prof

Power-prof (Kjærgaard and Blunck, 2012) is an unsupervised API-level energy profiler that employs a genetic algorithm (GA) to profile the dynamic aspects of mobile components during application execution on mobile phones. When operating on the power statistics gathered during the power profiling phase, the GA estimates the power models for mobile components, such as Wi-Fi, 3G, CPU, and LCD. The power model, which constitutes a set of conditional functions, predicts mobile battery power consumption behavior under certain context-aware conditions. During the power profiling process, Power-prof employs a smart battery interface to acquire time-stamped power measurements for each mobile component. Subsequently, to construct power models, Power-prof considers four distinct power states for each mobile feature (to reduce search space) to analyze battery power draining rate. During the subsequent steps, the GA iteratively processes the chromosome that constitutes the parameters, including the time and energy required during transitions between different states of mobile components to find the optimal parameters for the subject component power model. The design of Power-prof is efficient because it uses a dedicated server to execute computationally expensive (power model generation) tasks. However, this study does not reflect the interdependent power behavior of mobile components during the profiling phase. Incorporating the interdependent characteristics of the mobile feature into the model can significantly improve estimation accuracy. In terms of time and required system resources, GA-based solutions are expensive. The accuracy of this scheme can be improved further by increasing the power states for each mobile component in the search space.

4.1.2. Power Booter

Power Booter (Zhang et al., 2010) is an automated power model construction method that builds power models for mobile components by employing built-in battery voltage sensors and knowledge of battery state discharge rate. During the training phase, Power Booter keeps individual mobile components in a particular state for a fixed period to determine the state of discharge of the mobile battery. Consequently, during the power model construction phase, Power Booter applies a multi-variable regression analysis on the collected battery discharge curve and system utilization traces to construct power models for mobile components, including CPU, LCD, Wi-Fi, cellular, and GPS. However, the training phase is a resource rigorous and time consuming step as it iteratively varies state of individual mobile features while keeping the state of the rest of the mobile components constant to eliminate power estimation noise. The advantage of this scheme is its scalability because it rapidly generates power models for smartphones and does not require external hardware equipment for power profiling. Furthermore, Power Tutor highlights the power consumption during a process, and mobile component granularity is based on Power Booter power modeling methodology. However, the variance in voltage discharge curves for different types of mobile phones causes inaccuracies in power consumption readings captured during the training phase. However, the validity of power models is affected by changes in the voltage discharge curve because of changes in temperature. Power Booter also rapidly discharges the mobile battery (caused by the profiling process) because it does not use an external physical server (e.g., case of simulation-based power modeling) to perform computationally expensive regression analysis on power statistics. Similar to Power-prof, this scheme also does not consider inter-mobile component effects during power profiling.

4.1.3. Se-same

Se-same (Dong and Zhong, 2010) is a self-modeling paradigm that employs linear regression statistical analysis of power measurements to construct accurate and high-rate energy models compared with smart battery interface-based power models. High-rate power models estimate energy consumption for a time interval of less than a second (e.g., 10 ms or 100 Hz) and are less accurate than state-of-the-art low-rate energy profilers. The design of Se-same is based on three key modules, namely, collector, model constructor, and model modeling. The collector module, which is found within the kernel, employs ACPI battery interface and OS native services to attain mobile power consumption and system utilization statistics, respectively. The model constructor component uses model-modeling and prediction transformation component modules to generate high-rate power models using principal component analysis (PCA) heuristics. The strength of Se-same is its competence in terms of efficient resource utilization because it schedules resource-intensive operations that are executed during the idle hours of a mobile. Moreover, Se-same-based power modeling induces limited profiling overhead because it implements the collector module within the OS kernel to lessen the total number of system calls. Se-same has also achieved a higher rate than that of a simple smart battery interface-based scheme with similar accuracy. However, for the power modeling of mobile components, Se-same only deliberates the components (e.g., memory and CPU) that are visible to the OS. It ignores the power modeling of various important mobile components, such as radio and Wi-Fi, which are the foremost energy-consuming components in mobile phones. The data collection module (for a high-rate case) of Se-same can be optimized to minimize profiling overhead further.

4.1.4. Hybrid-feedback

Hybrid feedback (Gurun and Krintz, 2006) is an adaptive profiling scheme that employs BMU to monitor voltage and current drops during the execution of an application in mobile phones to construct power models to estimate mobile battery power consumption. Hybrid feedback utilizes first-order regression analysis to map software counters to the energy consumption of an application. This power estimation framework consists of three modules, namely, runtime profiler, off-line profiler, and power estimator. The runtime profiler polls the BMU to determine application execution and communication profiles before applying the recursive least square linear regression with exponential decay (RLS–ED) to update model parameters. The estimator module uses computation and communication models, as defined in Eqs. (5) and (6), to predict application energy consumption during a time interval t. In the presented models, x_i, B_{tx}, B_{rx}, and α denote the count of CPU cycles, transfer bytes, receive bytes, and weight to processor and Wi-Fi interface, respectively. The off-line profiler module provides initial profiling statistics to complete the recursive RLS–ED algorithm.

$$E_{CPU} = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 \tag{5}$$

$$E_{Net} = \alpha_1 x_1 + \beta_1 B_{tx} + \beta_2 B_{rx} + K \tag{6}$$

The critical aspect of the proposed method is this that it has overlooked the fine-granular details like amount of energy required during switching off/on radios while constructing power models. Moreover, as BMU offers very low update rate; therefore, power consumption pattern of a network interface (Wi-Fi link) is hard to measure using BMU.

4.1.5. SEMO

Smart energy monitoring system (SEMO) (Ding et al., 2011) continuously monitors and analyzes the power consumption behavior of a mobile device to estimate the power consumption behavior of its applications. The design of the SEMO system consists of three key modules: inspector, recorder, and analyzer. The inspector module monitors the battery health status, voltage level, total battery charge remaining, and temperature of mobile device during application execution. Consequently, this system notifies the application user when the device is about to reach its critical condition (e.g., low power, low temperature, overutilization, and voltage level). The recorder is responsible for collecting various battery and program-related parameters, such as execution time t, power consumption during time t, and number of applications running at time t. The analyzer module utilizes the profile of the recorder and the power-remaining historic curve of the mobile battery to estimate the energy consumption rate of an application. SEMO assists application developers in ranking mobile applications based on energy consumption behavior to optimize power-hungry applications. The critical aspect of the SEMO framework is that it is a low-rate scheme because it collects power statistics once per minute. Thus, it has a high probability of missing many high-power consumption statistics. This framework also does not consider the effect of the components that are invisible to the running OS.

4.1.6. Elens

Elens (Hao et al., 2013) is a fine-granular energy profiling tool that combines program analysis and per-instruction energy profiling paradigms to estimate application energy consumption accurately. During the training phase, the program analysis module records the paths traversed during application execution and uses this information to estimate energy consumption at different levels, such as path, function, thread, program, and instruction, without requiring the support of the developer. The system design of Elens consists of a workload generator, an analyzer, and source code annotator modules. The workload generator module accepts the workload description document and software artifacts to translate workloads proactively into a set of paths. The analyzer module exploits path information and software energy profile to estimate the energy consumption of an application. The source code annotator visualizes energy usage to the application source code to assist application developers in optimizing the power consumption of an application. The advantage of this scheme is its simplicity because it does not require lower-level hardware details (e.g., system states) and OS kernel instrumentation to profile application energy consumption. However, Elens assumes that per-instruction power profile is always available, which is false because an instruction set varies from architecture to architecture. Furthermore, Elens requires the application source code to implement the application for path finding, which makes it impractical. Elens can be optimized further by applying weighted probability and branch prediction knowledge to predict the code that will be executed during a specific use case to eliminate the overhead of application execution for path finding.

4.1.7. ARO

The application resource optimizer (ARO) (Qian et al., 2011) tool exposes a cross-layer interaction model among the radio resource channel, transport layer, and application layer to identify inefficient resource usage of mobile applications. Network activity during application execution rapidly depletes mobile battery charge if radio (e.g., 3G network) states are utilized inefficiently. ARO determines radio states by using a radio resource controller to optimize radio and eliminate short traffic bursts that occupy the channel for a long period while performing no activity. The architectural design of ARO requires collector and analyzer modules. The collector module collects network packet traces, as well as user and system inputs from a mobile phone; whereas the analyzer determines radio state transitions using the collected traces. Based on the ARO model, mobile applications access the state of the mobile radio channel prior to data transfer to utilize mobile battery charge optimally. The advantage of ARO is that it helps application developers minimize trade-off among efficiency, performance, energy, and functionality given the availability of device- and network-specific information at the application layer. However, this tool only considers mobile radio power consumption and disregards system-/component-level power consumption, such as CPU, application API, GPU, LCD, and GPS. The ARO tool can be optimized further to diagnose power consumption at a fine granular level by incorporating other cross-layer events, such as cellular hand-off and OS events.

4.1.8. Wattson

Wattson (Mittal et al., 2012) empowers application developers to identify energy-critical sections within mobile applications to highlight utmost energy-consuming entities. This tool maps application behavior on a mobile platform by emulating processing speed and network characteristics without requiring laboratory equipment. The design of Wattson is based on power modeling and resource scaling tools. The former measures the energy consumed by an application under different environmental conditions and settings. The power modeling entity applies the utilization-counter method to compute application energy consumption (using power models), and the scaling module emulates the mobile environment within a workstation to resolve platform heterogeneity issues. Wattson emulates the power consumption of the display, CPU, and network components. It is flexible because it does not require external equipment and mobile phones to estimate power consumption by a mobile application. However, this tool does not consider power consumption by various smartphone components, such as GPU and GPS. Furthermore, the effect of background noise on the accuracy of application energy estimation is overlooked during the estimation process. Profiling time is also high because the emulation process is slow.

4.1.9. Eprof

Application energy profiling experiences the unique challenge of asynchronous power consumption behavior, wherein the state of the mobile component remains unchanged even beyond the lifetime of the entity that triggers it. Eprof (Pathak et al., 2012), a fine-granular energy profiling tool, has resolved the asynchronous power consumption behavior issue of mobile components by employing the last-trigger accounting policy. Based on power state analysis, Eprof identifies the key reasons for asynchronous power consumption behavior, which include component/application tail-power state, persistent power state wake-locks, and exotic components within smartphones. The tail energy (e.g., disk, Wi-Fi, 3G, and GPS) of mobile components represents the state in which the activities in one entity (e.g., function, thread) push the mobile component to high power states and the component remains in that state even after that entity is terminated. Meanwhile, persistent state wake-lock occurs because of aggressive CPU/screen sleeping policies as legacy OS exports wake-lock APIs to ensure that the components of the smartphone are awake during application execution. Exotic components, such as GPS, camera, and accelerometer, consume a significant amount of battery power because they are activated and deactivated by distinct entities. The advantage of Eprof is its capability to map the energy consumption of a mobile component at diversified levels, such as process, subroutine, thread, and system call. However, Eprof overlooks energy consumption by OS policies and poor software design.

4.1.10. P-top

P-top (Do et al., 2009) estimates the energy consumption of an application at process granularity. The design of P-top consists of four vital components: energy profiler daemon, in-memory data, display utility, and API-kit. The energy profiler daemon runs within the OS kernel and records the resource utilization degree using a process for several smartphone components, including CPU, hard disk, and network connectivity. The energy consumed by each mobile component during a specified time interval is calculated through power models, as presented in Eqs. (7) and (8), prior to transferring it to the in-memory module for temporary storage. In the presented models, the parameters t_j, n_k, E_k, and P_{end} represent process execution time, count of transition between CPU states, energy consumption at a particular state, and power consumed during data sending via Wi-Fi link, respectively. The display utility exploits the collected execution log (from the in-memory module) to generate a detailed report that highlights the power consumption of several mobile components, such as CPU, network, and disk components. P-top offers the API interface to enable application developers to access the power log of their desired processes. Consequently, application developers can optimize application scheduling and lightweight application design with the help of this API interface. This tool is helpful because it is embedded as an OS service. Furthermore, the profiling overheads of P-top in terms of CPU and memory resource are only 3% and 0.15%, respectively. However, numerous factors, such as GPS and LCD brightness, are disregarded during energy profiling. The asynchronous power features of mobile components are also overlooked.

$$E_{CPU} = \sum_j Pj * tj + \sum_k nk * Ek \tag{7}$$

$$E_{Net} = t_{send} * P_{send} + t_{receiving} * P_{receiving} \tag{8}$$

4.2. Hardware based application energy profiling schemes

4.2.1. DuT

In Carroll and Heiser (2010), the design of a hardware-based power modeling scheme is proposed to analyze mobile battery power consumption behavior when concurrently executing a bunch of mobile applications. The proposed scheme employs the device under test (DuT) and hardware data acquisition (DAQ) components to investigate application power consumption behavior. This scheme also uses a sense resistor placed at the power supply rail of each component to collect the current drained during activities in the mobile. In the DuT framework, the power for the DuT component is provided through a power supplier unit connected to the battery interface of the mobile device to avoid OS intervention. The software module (which runs on a PC) employs the DAQ library to extract power and execution statistics from DAQ to generate the power model and estimate the energy for numerous applications, such as audio and video, as presented in Eq. (9), where t denotes the total execution time and P_{BL} represents backlight power. The advantage of this scheme is that it uses a free-runner smartphone whose design files, particularly the circuit schematic, are freely and easily available. However, the Openmoko Neo free-runner smartphone is old and does not support 3G network connectivity. The scheme also does not consider power loss while converting supply voltage to the level required by the component.

$$E_{audio} = 0.32\ \mathrm{W} * t, \quad E_{video} = (0.45\ \mathrm{W} + P_{BL}) * t \tag{9}$$

4.2.2. Netw-trace

The framework proposed in Rice and Hay (2010a) uses fine granular annotated traces to investigate mobile energy draining behavior. The key features of this framework include batch processing, automated test execution, and support for untethered operations. This framework consists of power server and measurement hardware components. The measurement hardware tracks power consumption by inserting a resistor in the series between the battery terminal and its connector. The software module (test client) that runs within the handset connects to the power server to download the test script. After running the test script, the test client prepares the synchronization pulse to correlate timing logs once power consumption in the mobile phone is stabilized. The power server collects and aligns the received traces from the mobile phone. The power server module analyzes the traces (network traces) to investigate major entities that affect the mobile battery. The strength of this scheme is its automation capability, which enables it to run a series of tests without user intervention. However, this scheme requires a stable network to acquire a test script and upload trace files, which are is not always the case.

4.2.3. Power Memo

The asynchronous nature of I/O operations during application execution complicates the process of identifying what is responsible for an I/O activity because of rapid context switching. Power Memo (Tsao et al., 2012) is a measurement-based energy profiler that overcomes the aforementioned issue by adding the process identifier (PID) to the socket data structure to identify precisely the process that has generated or received the packet. Power Memo estimates application power consumption at process and function granularities. The Power Memo architecture deploys the required system modules at host and target sides. The host side implements the GUI module, which (a) acts as a control center and access the data-acquisition card (DAQ), (b) emulates mobility using a single attenuator module, and (c) maps power measurements to calculate the total energy for each system activity. On the target side, the kernel module uses kernel probes (K-probes) and user space probes (U-probes) to support static and dynamic profiling. Compared with static profiling, dynamic profiling does not require the source code of the application to generate a report on profiling. The kernel module logs system activities and other necessary parameters in a file before it transfers it to the user space for energy consumption estimation using National Instruments PCI-6115 data-acquisition board. However, the accuracy of I/O energy estimation is affected by noise pattern on channels, type of radio, distance to the access point (in case of Wi-Fi), and the available bit rate that has not been focused by the Power Memo.

4.2.4. power scope

Power scope (Flinn and Satyanarayanan, 1999) is a hybrid solution that combines hardware instrumentation and kernel-based software module to profile the energy consumption of mobile applications at the procedure level. The design of power scope consists of data collection and analysis modules. To collect voltage and current readings, power scope employs a digital multi-meter to sample the current drawn from the profiling host through its external power supply. The data collection module utilizes energy and activity monitors to collect required statistics. The system monitor module uses the program counter and PID of recently running processes to sample system activities. The energy monitoring module collects and stores current samples periodically. Similarly, the energy analyzer component generates the energy profiles for system activities while using instantaneous current and voltage over time. The analyzer module, which performs on an external server, optimizes the resource overhead of the profiling device. However, the profiling results can be imprecise when the power scope is used because it measures total energy and depends on an external triggering mechanism to synchronize the host and the target. Moreover, the power behavior of this asynchronous I/O event is not thoroughly explored by the power scope. The profiling overhead of the existing scheme should also be identified because it affects the performance of the co-hosted application in terms of execution time and throughput. The advantage of this scheme is that it minimizes the energy budget by 46% for the tested application (video compression).

4.2.5. Network

The profiling scheme, as discussed in (Rice and Hay, 2010b), generates fine-grained annotated power traces to analyze the behavior of mobile power consumption using hardware instrumentation. This scheme consists of hardware equipment and software module. The former measures voltage drop across a sense resistor to estimate power consumption, whereas the latter acquires test scripts from a remote server to generate the execution log during application execution in mobile phones. The profiling scheme observes that Wi-Fi network power consumption is a function of network traffic, connection establishment method, data transmission rate/size, network protocol type, and sender buffer size. The power consumption of an application during network activity is costly when considering the DHCP protocol (dynamic addressing) compared with when static addressing is considered. Similarly, the idle power consumption of a Wi-Fi network connection is lower than 2G and 3G because Wi-Fi requires a lower base power while maintaining cellular network connection. However, access point position, attenuation pattern, and types of radio significantly affect battery power consumption, which is not considered in this scheme. The profiling scheme also does not consider the asynchronous behavior of network interfaces, such as Wi-Fi. Furthermore, it requires a stable network connection to download the test script to analyze power consumption behavior. The effects of the distance between the mobile node and the Wi-Fi access point on the total power consumption of the Wi-Fi component should be modeled in detail to improve estimation accuracy.

4.2.6. Web-browser

In Thiagarajan et al. (2012), authors have investigated the power consumption behavior of smart-phone web browser components. Design of the proposed energy measurement scheme consists of server, mobile phone, and multi meter modules. The server entity control and manages the mobile phone and multi meter equipment during the power profiling process. In the initial step, the server module communicates to web browser profiler on the mobile phone to repeatedly load a specific URL. During the subsequent stages, the profiler measures energy consumption of web browser using a multi-meter hooked with mobile battery while the browser renders the web pages. The software part of the proposed scheme includes web browser profiler and android-browser. During browsing activity on mobile phone, the profiler caches the web page elements and monitors 3G/Wi-Fi signal strength, data transfer rate, and page loading time to estimate the power consumption. The web browser loads web pages either in "with cache" or "no cache" mode. The authors concluded that the cascade style sheet and java scripts are expensive elements while accounting browser energy due to their larger energy demands. The critical aspect of the proposed scheme is that the sampling rate of the chosen multi meter is very high that leads to estimation inaccuracy. In the current study, the authors have considered energy estimation of only a few web pages; however, the proposed model can be extended by considering complete web session energy estimation. The simple solution to sum the individual web pages for web session energy calculation does not work due to the cache element involved during cascading pages loading.

4.2.7. Multi-core CPU

Performance monitor counter (PMC) based power estimation is undesirable for resource-constrained mobile devices because they offer limited support to such devices. In Walker et al. (2015), the authors identified the issues in implementing PMC on mobile phones (e.g., works only for a set of embedded systems) and proposed a utilization-based power estimation method for mobile applications. To design the power model, the authors utilized the ODROID-XU+E board that employed current and voltage sensors to estimate mobile power consumption. However, the sample rate offered by the acquisition board was considerably low. In Lin et al. (2014a) the authors estimated application energy consumption by employing a two-phase calibration approach that considered a system-on-chip processor with two cores. The proposed scheme constructed the power tables during the first phase. During the second phase, the faulty energy formulas were replaced with the correct ones that assumed a linear regression analysis. However, employing a two-phase calibration process resulted in a high overhead (i.e., in terms of required computational resources), which led to rapid battery charge depletion.

5. Comparison of mobile application energy profiling schemes

Table 2 highlights the difference between hardware based and software based mobile application energy profiling paradigms.

The following section critically analyses mobile application energy profiling schemes based on the parameters selected form the existing literature as reported in Section 3.1.

5.1. Software based energy profiling

This section compares software-based energy profiling schemes based on granularity, model flexibility, measurement source, profiling type, resource overhead, and platform parameters as presented in Table 3.

Granularity indicates the extent to which an energy profiler estimates power consumption during mobile application execution on resource-constrained smartphones. The granularity parameter divides software-based energy profiling schemes into several categories, as illustrated in Table 3. For example, energy profiling schemes, such as Se-same (Dong and Zhong, 2010) and SEMO (Ding et al., 2011) estimate power consumption at the application granularity level. However, Se-same is more detailed than SEMO because it considers profiling overhead during energy profiling. The wattson (Mittal et al., 2012) energy profiling scheme estimates power consumption at the application component level (e.g., application GUI component). Similarly, profiling schemes, including Elens (Hao et al., 2013), ARO (Navda et al., 2013), and Eprof (Pathak et al., 2012), estimate power consumption at method/path/line, burst, and system call level, respectively. Among Elens, ARO, and Eprof, the accuracy of Elens is the highest because it estimates mobile power consumption at the fine granularity level (instruction level). However, in terms of mobile battery resource utilization efficiency, ARO is the best because it employs external server resources to execute complex and resource-intensive operations during the profiling process. Numerous mobile application energy profiling models, including P-top and Eprof (Do et al., 2009; Pathak et al., 2012), estimate the power consumption at the process granularity. However, compared with Eprof, P-top profiling scheme is energy efficient

Table 2
Hardware based vs. Software based profiling.

Parameters	Hardware based profiling	Software based profiling
Granularity	Suitable for fine granular application monitoring	Suitable for coarse granular monitoring as smartphones battery APIs are best suitable for them.
Accuracy	High	Low
Power Source	Uses an external power source for hardware equipment during energy profiling process	Uses mobile's battery power for application energy profiler
Setup	It interfaces mobile battery to the power meter	APIs to access the battery draining rate, temperature, and total remaining battery power
Scalability	Not scalable	Scalable
Support	System level power consumption monitoring	Individual hardware component level monitoring
Dependency	Accuracy depends on hardware sampling rate	Accuracy depends on sense resistor's accuracy
Sampling Rate	Sampling rate as offered by hardware equipment	Power sampling rate is as offered by existing OS
Structure	It uses physical tools like power meters	It uses software stubs to capture power draining behavior of mobile battery/ components
Overhead	Do not suffer from feedback loop	Suffers from feedback loop as profiling application has to run on the same mobile phone

because it reduces a significant number of context-switching operations by instrumenting the power-collector module within the OS kernel. Fine granular schemes are more accurate than those with coarse level granularity. However, fine granularity requires the following: (a) widespread resource monitoring, (b) high resource synchronization between application activities and the updating rate of smart battery sensors, and (c) extended mobile application profiling time. Coarse-granular profiling (e.g., process, thread, and function) entails (a) low resource monitoring, (b) limited profiling time, and (c) inadequate resource utilization. The *model flexibility* parameter classifies software-based energy profiling schemes into dynamic and static profiling groups. Mobile application energy profiling schemes, such as Elens and Eprof (Hao et al., 2013; Pathak et al., 2012) apply the static profiling technique to estimate application power consumption, whereas others, such as Entracked (Kjærgaard et al., 2009), Power Booter (Qian et al., 2011), EET (Oliver and Keshav, 2011), and P-top (Do et al., 2009), employ the dynamic profiling paradigm. Compared with static energy profilers, such as Power Booter and EET, dynamic profilers (e.g., P-top and ARO) are scalable because limited programmer support is required after application development. Furthermore, the designs of P-top and ARO are ideal for application developers to work with when considering the privacy issues of an application. Static profiling is required when the profiler has to annotate the source code for programmer support. The annotated source code can be accessed and used by other programmers to optimize application source code.

The *measurement source* attribute classifies mobile application energy profiling schemes into numerous categories based on the method selected to access mobile power consumption statistics, such as NEP, smart battery interface, and program analysis, as presented in Table 3. The smart battery-empowered smartphone (Kim et al., 2014) uses the smart battery system to access the power drawn rate during application execution on mobile phones. Among these schemes, Power-prof, Power Booter, Se-same, and SEMO employ the smart battery interface to estimate application energy consumption using built-in voltage and current sensors. However, among others (e.g., Power-prof, Power Booter, Se-same, and SEMO) (Ding et al., 2011; Dong and Zhong, 2010; Gurun and Krintz, 2006; Kjærgaard and Blunck, 2012; Zhang et al., 2010), Power-prof and Se-same consume less system resources during application profiling because they schedule the execution of computationally expensive tasks on the pre-allocated resource-rich external server. Alternatively, Watts-on (Mittal et al., 2012) has considered NEP to estimate the application energy consumption within a mobile phone. Kernel instrumentation (Do et al., 2009) and logging-based schemes (Oliver and Keshav, 2011; Pathak et al., 2012) employ kernel-level probes to monitor system activities before transferring time-stamped traces to remote server for the analysis to construct power models. However, Eprof and P-top require root access to OS and mobile resources, and this condition is not true for all OS/mobile types. Furthermore, P-top is efficient compared with Eprof because it considers eliminating profiling overhead during application energy profiling. Hybrid feedback uses the BMU (Gurun and Krintz, 2006) module, which affects accuracy because of the limited power sample rate. Smart battery interface-based profiling tools are more efficient than other categories (NEP, BMU, and OS instrumentation) if voltage and current sensors provide accurate results. Compared with the remaining schemes, Elens is fine grained because it estimates energy consumption at the instruction level. However, the profiling overhead of Elens is considerably higher than those of the others because it runs mobile applications to record execution paths during application execution. The *profiling type* parameter classifies energy profiling schemes into online and off-line classes. Various mobile application energy profiling schemes, including ARO, Wattson, and Power-prof (Kjærgaard and Blunck, 2012; Kjærgaard et al., 2009; Mittal et al., 2012; Qian et al., 2011), employ the off-line energy profiling mode to analyze collected power traces. The remaining energy profiling schemes, such as P-top, Power Booter, Elens, and Se-same (Ding et al., 2011; Do et al., 2009; Dong and Zhong, 2010; Hao et al., 2013; Zhang et al., 2010), adopt the online profiling mode to construct the power models of smartphone components. In terms of profiling time, off-line profiling is superior to online profiling; whereas in terms of resource cost, online profiling is better. The majority of the profiling schemes follow the online profiling paradigm except for the hybrid-feedback solution, which partially follows the off-line mechanism.

Mobile application energy profiling is a resource-intensive process because it requires significant amounts of mobile resources during the energy logging and analysis processes. The *resource overhead* parameter specifies whether energy profiling schemes consider profiling overhead during energy logging and analysis. The attribute of the resource overhead parameter is directly affected by the online profiling mode. High profiling overhead (i.e., the “NO” attribute within a resource overhead parameter) results in (a) rapid mobile battery charge depletion, (b) high resource utilization, and (c) prolonged profiling time. Various energy profiling schemes (Dong and Zhong, 2010; Hao et al., 2013; Kjærgaard and Blunck, 2012; Kjærgaard et al., 2009; Mittal et al., 2012; Oliver and Keshav, 2011) have reduced profiling overhead by (a) eliminating the number of system calls through the implementation of the energy profiler within the OS kernel (Zhang et al., 2010), (b) employing the

Table 3
Software based energy profiling schemes comparison.

Scheme (Ref.)	Granularity	Model flexibility	Measurement source	Profiling type	Resource overhead	Platform
Power-prof (Kjærgaard and Blunck, 2012)	NA	NA	Smart battery interface	Off-line	YES	Symbian
Power Booter (Zhang et al., 2010)	NA	NA	Built in battery voltage sensors, battery discharge curve	On-line	NO	Android
Se-same (Dong and Zhong, 2010)	Application	NA	Smart battery interface	On-line	YES	(N85,N900) Symbian
Hybrid-feedback (Gurun and Krintz, 2006)	NA	NA	BMU	On-line/off-line	NO	NA
SEMO (Ding et al., 2011)	Application	NA	Battery usage curve	On-line	NO	Android (HTC desire)
ELens (Hao et al., 2013)	Path, line, method	Static	Program analysis (path level)	On-line	YES	NA
ARO (Qian et al., 2011)	Burst level	Dynamic	Traces driven	Off-line	NO	Android (Google Nexus)
Watts-on (Mittal et al., 2012)	Application components	Static	NEP	Off-line	YES	Android
Eprof (Pathak et al., 2012)	Process, thread, system call	Static	System calls log traces	NA	NO	Android, Windows
P-toP (Do et al., 2009)	Process	Dynamic	Kernel instrumentation	On-line	YES	Windows (ThinkPad T42)

off-line profiling mode, (c) opting for a lightweight profiling design (Do et al., 2009), and (d) ensuring unmodified OS services (Hao et al., 2013). Embedding the energy profiler into the OS kernel minimizes system calls to reduce context switching overhead because the profiler can directly access OS services. However, embedding the energy profiler into the OS kernel requires root access to the mobile phone. The lightweight energy profiler design employs intelligent/smart classification methods (e.g., PCA) to optimize application power consumption. The "No" attribute under the resource overhead parameter (Table 3) specifies that the proposed energy profiling scheme ignores profiling overhead while estimating application power consumption. Off-line-based schemes schedule energy-consuming activities on a nearby server platform to save mobile battery at the cost of privacy. The attributes of a *platform* parameter describes the smartphone model and OS type selected by the application energy profiler during energy profiling. The majority of the energy profiling schemes have considered the N95 (Balasubramanian et al., 2009), N8 (Kjærgaard and Blunck, 2012), ADP1 (Zhang et al., 2010), ADP2 (Zhang et al., 2010), N85 (Dong and Zhong, 2010), N900, T61 (Dong and Zhong, 2010), Android (HTC desire) (Ding et al., 2011), Google Nexus (Qian et al., 2011), Blackberry (Oliver and Keshav, 2011), and iPhone (Oliver and Keshav, 2011), platforms models during application energy profiling. However, Table 3 shows that the power measurement sources offered by different platforms vary. For example, N97 (Kjærgaard and Blunck, 2012) offers an internal smart battery interface to exploit voltage and current sensors to estimate power consumption. The power consumption of different mobile phone types also varies. For example, multi-core processors (ARM-15) are high-performance processors compared with ARM-7, which is an energy-efficient processor. Therefore, per-operation power consumption in ARM-15 is costly compared with that in ARM-7.

Table 4 labels mobile features that energy profiling schemes have considered during power modeling in order to estimate application energy consumption. Majority of energy profiling schemes (see Table 4) have reflected a limited set of mobile features during energy profiling. The description parameter defines the main aim of the reported energy profiling schemes.

Table 5 has categorized the software-based energy profiling schemes in numerous categories based on: (a) method the power models are built from, (b) information-type the model presents, (c) rate of power estimation, (d) accuracy of the desired models, and (e) power model construction environment. Software-based power models are designed by exploiting measurements of system utilization statistics (Ding et al., 2011; Do et al., 2009; Dong and Zhong, 2010; Mittal et al., 2012; Zhang et al., 2010), OS system calls (Hao et al., 2013; Pathak et al., 2012), and API calls (Hao et al., 2013; Kjærgaard and Blunck, 2012) in a programing platform. The system-utilization based power model design exploits system states such as, system-disk and CPU-cycles, to generate power models. Whereas, API and system-calls based profiling schemes exploit API/System-call interfaces to capture power consumption during system activity. The power models either estimate (Ding et al., 2011; Do et al., 2009; Dong and Zhong, 2010; Mittal et al., 2012; Zhang et al., 2010) the mobile application energy consumption during the recent-time or predicts (Kjærgaard and Blunck, 2012) the power consumption for a longer period of time (e.g., how long application can run). The energy profiling schemes construct power models either in a supervised or unsupervised execution mode. The supervised profiling (Gurun and Krintz, 2006; Pathak et al., 2012; Qian et al., 2011, Zhang et al., 2010) necessitates human intervention during the total profiling period whereas, unsupervised (Ding et al., 2011; Dong and Zhong, 2010; Kjærgaard and Blunck, 2012) profiling is a completely software driven process. High-rate energy profiler (Dong and Zhong, 2010; Kjærgaard and Blunck, 2012; Pathak et al., 2012; Zhang et al., 2010) offers high sample-rate as

Table 4
Comparison of mobile application energy profiling schemes based on mobile features.

Schemes (Ref.)	Resources analysed								Description
	CPU	LCD	GPS	Wi-Fi	UMTS	GSM	Accelerometer	Compass	
Power Proff (Kjærgaard and Blunck, 2012)	✓		✓	✓		✓	✓	✓	Unsupervised GA based energy profiling
Power Booter (Zhang et al., 2010)	✓	✓	✓	✓	✓				Automated smart battery interface based power model construction
Se-same (Dong and Zhong, 2010)	✓	✓		✓					High rate and accurate self-power modeling
Hybrid-feedback (Gurun and Krintz, 2006)	✓					✓			Hybrid (on-line/off-line) feedback based energy profiling to reduce power model construction time
SEMO (Ding et al., 2011)	✓								Low-rate smart energy monitoring system based profiling to improve accuracy
ELens (Hao et al., 2013)	✓		✓	✓					Program analysis based fine-grained energy profiling
ARO (Qian et al., 2011)	✓				✓				Cross-layer interaction based power optimization
Wattson (Mittal et al., 2012)	✓	✓		✓	✓				Emulation based profiling
Eprof (Pathak et al., 2012)	✓		✓	✓					Fine-grained energy profiling for smart phone applications
P-top (Do et al., 2009)	✓	✓				✓			Process level software power profiling

Table 5
Comparisons of energy profiler based on the diverse power modeling dimensions.

Power modelling dimensions		Scheme (Ref.)									
Parameter	Attribute	Power proff (Kjærgaard and Blunck, 2012)	Power Booter (Zhang et al., 2010)	Se-same (Dong and Zhong, 2010)	Hybrid-feed-back (Gurun and Krintz, 2006)	SEMO (Ding et al., 2011)	ELens (Hao et al., 2013)	ARO (Qian et al., 2011)	Wattson (Mittal et al., 2012)	Eprof (Pathak et al., 2012)	P-toP (Do et al., 2009)
Method	Utilization		✓	✓	✓	✓		NA	✓		✓
	System call						✓	NA		✓	
	API	✓					✓	NA			
Information type	Estimation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Prediction	✓									
Environment	Supervised		✓		✓		✓	✓	✓	✓	
	Unsupervised	✓		✓		✓					✓
Profiler rate	Rate	4 Hz	5 KHz	1 Hz	10 KHz	NA	10 KHz	NA	NA	NA	NA
Accuracy	Error rate	0.145 W	4.1%	5%	6%	NA	10%	NA	4–9%	6%	<2 W

compared to low-rate profiling (Ding et al., 2011; Gurun and Krintz, 2006; Hao et al., 2013) which estimates energy consumption for the timing of 1 s or higher. An accurate mobile component power model (Dong and Zhong, 2010; Kjærgaard and Blunck, 2012; Zhang et al., 2010) precisely estimates mobile application power consumption for effecient power optimization. Table 5 indicates that the power profiler rate affects the accuracy of a power model. For example, at 4 Hz, Power-prof predicted application energy consumption with a mere 0.145 (95th quartile) marginal error for pedestrian tracking applications. Similarly, hybrid feedback and Wattson reported 6% and 4–9% error, respectively, when estimating energy consumption for both network-bound and image-processing modules at varying rates. By contrast, Elens applied the learning, evaluation, and planning framework, and estimated energy consumption with a 10% error rate for video-based modules. Compared with state-of-the-art profiling schemes, Se-same, Power Booster, and Eprof reported 5%, 4.1%, and 6% energy estimation error, respectively, for different mobile applications. Ptop also estimated process energy consumption with <2 W marginal error for an image viewer application. Table 6 shows the comparison of energy profiling schemes in terms of their availability in the research community. This table highlights open source and free profiling solutions. A web link to access the reported open access energy profiling solutions is also provided.

5.2. Hardware based energy profiling

This section relates the hardware-based energy profiling schemes on the basis of several parameters selected from the existing literature. The parameters include granularity, platform, measurement source, power model design, processor core, and execution environment, as illustrated in Table 7.

Granularity provides the extent to which energy profiling schemes estimate mobile power consumption. The DuT (Carroll and Heiser, 2010) energy profiling scheme considers application energy consumption at software component granularity. Power Memo (Tsao et al., 2012), Power Scope (Flinn and Satyanarayanan, 1999), and Web-browser (Thiagarajan et al., 2012), employ hardware instrumentation to estimate power consumption at process/function, procedure, and web component granularities, respectively. The platform parameter attributes represent both the mobile phone model and the OS type selected by the mobile application energy profiling schemes during mobile energy profiling. In the literature, the majority of the hardware-based energy profiling schemes have selected the Neo Free-number (Carroll and Heiser, 2010), G1 (Android) (Rice and Hay, 2010a), Magic (Android) (Rice and Hay, 2010b), Magic (Nexus) (Rice and Hay, 2010b), and ADP2 (Android) (Thiagarajan et al., 2012) mobile models for the profiling process. During the profiling process, numerous mobile platforms have also supported original battery replacement with artificial battery (reusable) to obtain power readings (e.g., Google

Table 6
Energy profiler availability to the research community.

Scheme (Ref.)	Open Source		Freely Available		Web Link
	YES	NO	YES	NO	
Power proff (Kjærgaard and Blunck, 2012)		•	•		http://originaldll.com/file/powrprof.dll/4541.html
Power Booter (Zhang et al., 2010)	•		•		https://github.com/msg555/PowerTutor
Se-same (Dong and Zhong, 2010)		•	•		http://sourceforge.net/projects/sesame/files/Sesame%202
Hybrid-feedback (Gurun and Krintz, 2006)		•			NA
SEMO (Ding et al., 2011)		•		•	NA
ELens (Hao et al., 2013)		•		•	NA
ARO (Qian et al., 2011)	•			•	https://github.com/attdevsupport/ARO
Wattson (Mittal et al., 2012)		•	•		https://play.google.com/store/apps
Eprof (Pathak et al., 2012)		•	•		http://mobileenerlytics.com/eprof/demo/gui/pages/
Ptop (Do et al., 2009)		•		•	NA

Note: “Web Link” indicates location where source code or executable can be downloaded.

Nexus “S”) during application execution. Support for power estimation by different mobile platforms varies. For example, the API kits of Android, iOS, and Blackberry support state-of-charge (SOC) to estimate battery power consumption. The API kits of Android and Blackberry also support voltage sensors in estimating battery power consumption.

Measurement source parameter categorizes the hardware-based energy profiling schemes based on the hardware equipment selected by profiling schemes to measure the mobile power consumption during application execution. Energy profiling schemes have considered several multi-meter models such as, Agilent 34110A (Jiemsakul et al., 2010; Thiagarajan et al., 2012) and Hewlett Packard 3458a (Avellaneda and Lopez, 2014; Flinn and Satyanarayanan, 1999), to estimate the mobile power consumption. Furthermore, plentiful energy profiling schemes deliberated a number of data acquisition (DAQ) boards such as, PCI-6229 DAQ (Carroll and Heiser, 2010), PCI-MIO-16E-4 (Rice and Hay, 2010a), PCI-6115 DAQ (Tsao et al., 2012), and PCI-MIO-16E-4 (Rice and Hay, 2010b), to estimate the mobile/application power consumption. DAQ equipment measures the current, temperature, and voltage of the desired mobile platform if its support is available (Keleshis et al., 2014). To estimate the network application power consumption, Tsao et al. (2012) has exploited DWL-G122 USB wireless adaptor. Energy profiling schemes such as Netw-Trace (Rice and Hay, 2010a) have exploited oscilloscope (Rice and Hay, 2010b) to closely observe the voltage drop during mobile application execution. Power model design parameter defines the method opted to generate the power models. Various energy profiling schemes (Carroll and Heiser, 2010; Cignetti et al., 2000; Tsao et al., 2012) followed the deterministic power modeling approach to estimate power consumption at the application component level. However, Power-scope (Tsao et al., 2012) and Network (Rice and Hay, 2010b) has considered statistical modeling paradigm to construct power models. Execution environment parameter classifies the energy profiling schemes into manual and automated classes based on the execution-setup. Several hardware based energy profilers (Rice and Hay, 2010a, 2010b; Thiagarajan et al., 2012; Tsao et al., 2012) exploited automated execution environment attribute to control software and hardware entities. However, DuT followed manual attribute during application energy profiling. Automated execution settings are preferred as the same test can be run several times to verify the battery discharge behavior. Processor core attribute specifies the computational capacities of a mobile phone. Many of hardware-based energy profiling schemes, including DuT (Carroll and Heiser, 2010), Netw-trace (Rice and Hay, 2010a), Power Memo (Tsao et al., 2012), power scope (Flinn and Satyanarayanan, 1999), and Network (Rice and Hay, 2010b), considered single core processor based mobile platforms. Likewise, Multi-core (Walker et al., 2015) and Multi-core- CPU (Lin et al., 2014a) profiling schemes considered multicore processor during energy profiling. However, energy profiling for multi-core processor is a complex process due to high context switching involved during profiling process.

6. Discussion on research issues in mobile application energy profiling

This section thoroughly discusses recent issues in mobile application energy profiling research domain. Mobile application energy profiling recent issues include, limited estimation accuracy, high profiling overhead, low profiling rate, high energy-performance trade-off, high-granularity, multi-core based profiling, limited OS' API support, and emerging trends in context aware energy profiling paradigms as highlighted in Fig. 4.

The restricted battery capacity of smartphones prohibits resource-rich mobile applications from fully utilizing their peak competencies to enrich user experience (Abolfazli et al., 2014b; Rahimi et al., 2014). Therefore, the design of a resource-constraint smartphone has motivated application developers to optimize application architecture efficiently to augment mobile battery lifetime (Abolfazli et al., 2014a; Alam et al., 2014). A precise and accurate power consumption estimation during application execution also assists application developers in designing efficient adaptation schemes to optimize mobile battery power consumption (Gurun and Krintz, 2006). Furthermore, accurately estimating the power consumption of mobile components assists OS designers to manage mobile resources efficiently. Application energy profiling schemes employ mobile power models to estimate application energy consumption at various granularities, such as application, path, process, or thread. However, existing mobile power models are negatively affected because of limited precision and coarse-grained energy estimation issues (Liu et al., 2012; Sasu Tarkoma et al., 2014; Tsao et al., 2012). Consequently, these issues affect the power estimation accuracy of an application. In the literature, the proposed power models efficiently estimate fine-grained usage statistics; however, these models lack the capacity to estimate power consumption at fine granularity (Flinn and Satyanarayanan, 1999; Kjærgaard et al., 2009; Lin et al., 2014b). The proposed power models are also considerably generic, inflexible, and platform dependent (Abolfazli et al., 2014b; Liu et al., 2012; Rahimi et al., 2014; Sasu Tarkoma et al., 2014; Tsao et al., 2012). Consequently, the tools based on these models are also platform dependent and inflexible. A fine-grained power model should be designed with minimum complexity to enhance power estimation accuracy. However, the design of fine-granular power modeling is affected by the accuracy of voltage and current sensors (smart battery interface) within smartphones. Recent smartphone battery interface supports voltage and current sensors with

Table 7
Hardware based energy profiling schemes comparison.

Scheme	Granularity	Platform	Measurement source	Power model design	Execution environment	Processor core
DuT (Carroll and Heiser, 2010)	Software components	Linux based (Neo Free-runner) Android (Nexus one)	National Instruments PCI-6229 DAQ	NA	Manual	Single
Netw-trace (Rice and Hay, 2010a)	NA	Android (G1) Android (Magic)	National Instruments PCI-MIO-16E-4, Oscilloscope	Deterministic	Automated	Single
Power Memo (Tsao et al., 2012)	Process and function level	Android and Linux systems	PCI-6115 data acquisition board, DWL-G122 USB Wireless NIC	Deterministic	Automated	Single
Power scope (Flinn and Satyanarayanan, 1999)	Procedures and process	NA	Digital multi meter Hewlett Packard 3458a	Statistical	NA	Single
Network (Rice and Hay, 2010b)	NA	Android (G1) Android (Magic) Android (Nexus)	National Instruments PCI-MIO-16E-4, Oscilloscope	Statistical	Automated	Single
Web-browser (Thiagarajan et al., 2012)	Web component	Android (ADP2)	Multi meter Agilent 34110A	NA	Automated	Single
Multi-core (Walker et al., 2015)	NA	Android (Cortex A7, Cortex A-15)	Agilent N6705	Statistical	Automated	Multi
Multi-core-CPU (Lin et al., 2014a)	Process	Android (Dev)	Power meter	Statistical	Automated	Multi

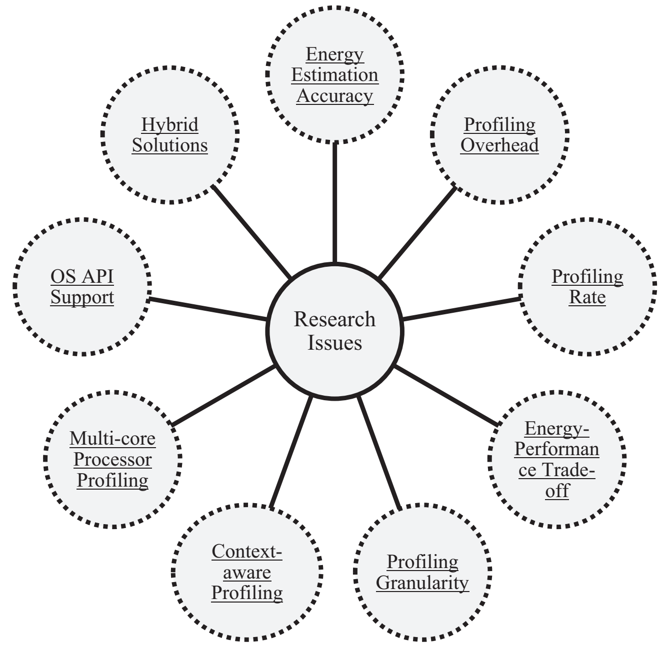

Fig. 4. Mobile application energy profiling issues.

limited accuracy. Nevertheless, the smart battery interface profiling rate is significantly low, which affects power estimation accuracy. Therefore, the power model design should be flexible and accurate while considering a dynamic environment (energy profiling during mobility). Extending application energy profiling time can improve application profiling energy estimation accuracy at the cost of additional mobile battery usage. Research on determining optimal balance between profiling time and energy consumption remains lacking when a certain accuracy level has to be assured.

Mobile application energy profiling schemes adopts hardware-based (Carroll and Heiser, 2010; Rice and Hay, 2010a, 2010b) or software-based (Kjærgaard and Blunck, 2012; Kjærgaard et al., 2009; Zhang et al., 2010) methodologies to estimate application energy consumption at diverse granularities. Energy profiling methods use either statistical sampling or source code instrumentation to estimate application power consumption. Statistical sampling-based profiling (Dong and Zhong, 2010; Flinn and Satyanarayanan, 1999) induces low profiling overhead (on mobile phones) and does not require the application source code. However, source instrumentation-based energy profiling (Do et al., 2009; Lauridsen et al., 2014; Oliver and Keshav, 2011; Pathak et al., 2012) is comparatively more accurate and requires the original mobile application source code. Hardware-based schemes are inflexible because they cannot estimate power consumption at the mobile component level (Carroll and Heiser, 2010; Tsao et al., 2012). By contrast, hardware-based energy profiling methods employ DuT, physical server, and power measurement equipment to estimate the power consumption of a mobile device. To measure the battery power consumption rate, hardware-based profiling includes a resistor in series between battery terminals to quantify voltage drop across the resistor (Carroll and Heiser, 2010; Sasu Tarkoma et al., 2014; Tsao et al., 2012). However, including a resistor increases the estimation error caused by the increase in total resistance of the circuit. Therefore, considering the induced errors is critical because of the additional increase in resistance (caused by the inclusion of a resistor in series) to improve power estimation accuracy. Hardware-based energy profiling is also costly because it (a) increases noise, (b) only works for certain types of batteries (e.g., does not work for nickel batteries), and

(c) requires additional hardware equipment for power estimation (Linares-Vásquez et al., 2014; Thiagarajan et al., 2012). However, hardware-based solutions are more accurate and do not need to interact with mobile phones because the test server can directly control mobile phones via different commands. Moreover, a software-based solution increases profiling overhead because it consumes additional power during the power logging phase (Kjærgaard and Blunck, 2012; Kjærgaard et al., 2009; Zhang et al., 2010). For a software-based energy profiling design, power estimation is also restricted by available APIs because of the existing OS limited support. Therefore, designing a multifunctional energy profiling scheme, which should impose low overhead on the mobile platform and should not require the application source code, is necessary. Energy profiling schemes must also consider the profiling overhead while estimating application power consumption to improve application profiling accuracy. To resolve the asynchronous power behavior of mobile components, several OS-assisted strategies should be designed to eliminate the effect of asynchronous power behavior on power estimation accuracy. Opting for high sample rate- and high power update rate-based power meters is necessary to improve power estimation accuracy using hardware-based schemes.

The hybrid of software and hardware-based energy profiling solutions remarkably augments battery lifetime by reducing software-based profiling overhead. During the mobile power profiling phase, a significant amount of energy is consumed while logging the battery power drawn rate of mobile components at varying inputs and rates. To reduce profiling overhead, fine-granular instruction-level power profiling can effectively assist application developers in estimating application energy consumption with high accuracy (Tiwari et al., 1994). The high-rate hardware module (i.e., power meter) can be employed to obtain the voltage and current drawn rates during instruction benchmark execution on the mobile platform for each instruction of the instruction set architecture (ISA) of an ARM-based processor. Consequently, application energy consumption can be estimated based on the per-instruction power models. However, usually the instruction power profile is unavailable for ARM-based processors. Therefore, the power profile for the ARM instruction set can be generated in off-line mode to offer it as a service that can assist mobile application users/developers in estimating the energy consumption of their applications using web services (Hao et al., 2012; Liew et al., 2015; Shuja et al., 2014). However, the accuracy of per-instruction power profiling is affected by (a) the rate offered by selected hardware modules, (b) workload on the mobile phone where the instruction benchmark is running, (c) accurate noise isolation during profiling, and (d) instruction benchmark loop-size to disregard the effect of system cache. Consequently, instruction-wise power profile can facilitate the estimation of application power consumption based on application code analysis. However, estimating mobile application energy faces several challenges, including (a) effect of multilevel cache, (b) non-deterministic application execution behavior, (c) heterogeneous memory models, and (d) inter-instruction power-varying behavior. All these issues should be addressed to improve per-instruction-based application energy estimation. Fine-granular instruction power profiling can also aid application developers in optimizing their source code to design green mobile applications.

Resource hungry high-performance computing applications (Hernández et al.) consume a significant percentage of battery capacity during application execution in mobile phones. Therefore, application developers must consider power–performance trade-off while optimizing mobile application power consumption budget because software architecture design significantly affects mobile battery discharge rate (Amini et al., 2013; Hernández et al.). For instance, while executing real-time communication applications, a quick application response time is highly crucial and consumes a significant amount of battery resource. Similarly, for security-based applications, quantifying power–performance trade-off during the encryption/decryption process is difficult because this process is highly complex (Akhunzada et al., 2014; Duan et al., 2013; Thiagarajan et al., 2012). In particular, each line of application source code consumes a dissimilar amount of energy because of the variance in the number of operations being performed (Thompson et al., 2011). Subsequently, the application segment that heavily uses mobile hardware components drains more battery power because of the dynamic behavior of the application/mobile components. For example, pedestrian tracking applications extensively use mobile GPS and radio by calculating and transferring position updates to the monitoring server for continuous observation (Kjærgaard et al., 2009). In this scenario, increasing the time interval between position updates compromises the accuracy of application performance (Kjærgaard et al., 2009; Pathak et al., 2012). Therefore, energy profiling designs should consider energy–performance trade-off while estimating and optimizing application energy consumption. For example, applying dynamic voltage frequency scaling (DVFS) (Ahmad et al., 2015a, 2015b) increases battery lifetime at the cost of application throughput. To date, limited attention has been provided to highlight the effects of applying DVFS optimization during power profiling on estimation accuracy. Energy profiling issues must also be simultaneously considered with multi-core processors.

Recent mobile execution trends have shifted the application execution platform to resource-rich cloud to satisfy the continuously increasing demand for innovative mobile applications. Rich mobile applications (Abolfazli et al., 2014b; Shiraz et al., 2014) considerably exploit system resources (computational, communication, and storage) because of the integration of emerging features, such as context awareness, augmented reality, and wearable computing (Abolfazli et al., 2014b; Shiraz et al., 2013). To conserve battery power, smartphones identify and migrate computationally expensive tasks to the nearby cloud. Mobile application models, such as MAUI (Cuervo et al., 2010; Tu, 2000) seamlessly migrate workload to the cloud to augment smart phone battery capacity (Khan et al., 2013) . However, deciding which part of the application to migrate is affected by both hardware and software constraints. Hardware constraints refer to smartphone features, such as GPS, accelerometer, and compass, which are accessed by the application during execution. Software constraints refer to the application segment that is mobile architecture-dependent (Ahmed et al., 2015a; Khan et al., 2013; Madani, 2014; Rahimi et al., 2014; Sasu Tarkoma et al., 2014). Context-aware mobile application profiling assists in enhancing the accuracy of mobile application profiling because the former considers the dynamic behavior of applications. An energy profiler assists in reducing the trade-off between energy consumption while executing an application locally, as well as the amount of battery capacity required while transferring and receiving data to and from the cloud (Khan et al., 2013; Shuja et al., 2012). An energy profiler monitors system and environmental variables; based on the profiled data, it also estimates application energy consumption on local mobile and remote servers at user-defined granularity levels (i.e., process, thread, and path). Transmission cost is estimated based on data size to be transmitted, channel bit rate, round trip time, signal-to-noise ratio, and available interfaces, such as Wi-Fi and 3G (Ahmed et al., 2015b; Khan et al., 2014; Rahimi et al., 2014; Sasu Tarkoma et al., 2014). Therefore, the designed application energy profiler should be flexible, lightweight, scalable, adaptive, and platform-independent.

7. Conclusions and future works

In this study, existing state-of-the-art mobile application energy profiling schemes are discussed and analyzed based on a thematic taxonomy to highlight the commonalties and variances among them. Mobile application energy profiling schemes follow either hardware or software-based profiling designs to profile the energy consumption of mobile applications. Hardware-based energy profiling schemes are expensive, labor-intensive, and non-scalable compared with software-based solutions. In terms of accuracy, hardware-based energy profiling schemes are highly accurate for the specific mobile device for which they are developed, but worst accurate for other device models. Accuracy is also affected by the hardware equipment sampling rate and sense resistor impedance. Software-based energy profiling estimates battery consumption at diverse granularities, such as process, thread, function, line, or path, by maximizing numerous power tracking resources, such as smart battery interface, BMU, Nokia energy profiler, and ARO profiler. The correctness of software-based energy profiling designs is affected by the accuracy level offered by the voltage and current sensors of the smart battery.

Issues in mobile application energy profiling are highlighted to design optimal energy profiling schemes. Research issues, including limited OS support for smartphone power estimation, trade-off between profiling rate and accuracy, application energy performance trade-off, and high profiling overhead, necessitate the development of a lightweight, platform-independent, highly accurate, and context-aware energy profiling model. Lightweight energy profiling design assists in reducing total development cost, extending application lifetime, and accelerating execution within a mobile device. Incorporating fine-granular power estimation into mobile energy profiling design also appears to be an optimum solution for the issues of limited accuracy in power consumption estimation. Including multifunctional objectives, such as low profiling overhead, source code independent profiling design, and dynamic environment adaptability, can also assist in designing a flexible, robust, stable, accurate, and low-cost energy profiling model. Low granularity profiling, such as source code line level energy estimation, can provide an opportunity to estimate mobile application energy consumption by predicting the application behavior for a set of use cases.

Acknowledgments

This work is fully funded by Bright Spark Unit, University of Malaya, Malaysia and partially funded by Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant UM.C/625/1/HIR/MOE/FCSIT/03.

References

Abolfazli S, Sanaei Z, Ahmed E, Gani A, Buyya R. Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. IEEE Commun Surv Tutor 2014a;16:337–68.

Abolfazli S, Sanaei Z, Gani A, Xia F, Yang LT. Rich mobile applications: genesis, taxonomy, and open issues. J Netw Comput Appl 2014b;40:345–62.

Ahmad RW, Gani A, Hamid SHA, Shiraz M, Xia F, Madani SA. Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues. J Supercomput 2015a;71:1–43.

Ahmad RW, Gani A, Hamid SHA, Shiraz M, Yousafzai A, Xia F. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. J Netw Comput Appl 2015b;52:11–25.

Ahmed E, Gani A, Khan MK, Buyya R, Khan SU. Seamless application execution in mobile cloud computing: motivation, taxonomy, and open challenges. J Netw Comput Appl 2015a;52:154–72.

Ahmed E, Gani A, Sookhak M, Ab Hamid SH, Xia F. Application optimization in mobile cloud computing: motivation, taxonomies, and open challenges. J Netw Comput Appl 2015b;52:52–68.

Akhunzada A, Sookhak M, Anuar NB, Gani A, Ahmed E, Shiraz M, et al. Man-at-the-end attacks: analysis, taxonomy, human aspects, motivation and future directions. J Netw Comput Appl 2014;24.

Alagöz I, Löffler C, Schneider V, German R. Simulating the energy management on smartphones using hybrid modeling techniques. Measurement, modelling, and evaluation of computing systems and dependability and fault tolerance. Switzerland: Springer International Publishing; 207–24.

Alam F, Panda PR, Tripathi N, Sharma N, Narayan S. Energy optimization in Android applications through wakelock placement. Design, automation and test in europe conference and exhibition (DATE): IEEE; 2014. p. 1–4.

Amini S, Lin J, Hong JI, Lindqvist J, Zhang J. Mobile application evaluation using automation and crowdsourcing. Proceedings of the PETools; 2013.

Avellaneda MS, Lopez AM. Fluke 8508A Multimeter introduction in capacitance measurement by indirect method. In: Proceedings of the IEEE conference on precision electromagnetic measurements (CPEM); 2014. p. 98–9.

Balasubramanian N, Balasubramanian A, Venkataramani A. Energy consumption in mobile phones: a measurement study and implications for network applications. In: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference: ACM; 2009. p. 280–93.

Bernstein WZ, Ramanujan D, Zhao F, Ramani K. Profiling energy consumption of smartphone users for environmentally efficient business decisions. In: Proceedings of the ASME 2013 international design engineering technical conferences and computers and information in engineering conference: American Society of Mechanical Engineers; 2013. p. V004T05A42–VT05A42.

N. Brouwers M. Zuniga K. Langendoen NEAT: a novel energy analysis toolkit for free-roaming smartphones; 2014.

Carroll A, Heiser G. An analysis of power consumption in a smartphone. USENIX annual technical conference; 2010. p. 271–85.

Thompson Chris, Douglas Schmidt, Hamilton Turner, Jules White. Analyzing mobile application software power consumption via model-driven engineering. 2011, 101–13.

Cignetti TL, Komarov K, Ellis CS. Energy estimation tools for the Palm. In: Proceedings of the 3rd ACM international workshop on modeling, analysis and simulation of wireless and mobile systems: ACM; 2000. p. 96–103.

Conte TM, Patel BA, Menezes KN, Cox JS. Hardware-based profiling: an effective technique for profile-driven optimization. Int J Parallel Program: Citeseer 1996.

Creus GB, Kuulusa M. Optimizing mobile software with built-in power profiling. Mobile phone programming. Netherlands: Springer; 449–62.

Cuervo E, Balasubramanian A, Cho D-k, Wolman A, Saroiu S, Chandra R, et al. MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th international conference on mobile systems, applications, and services: ACM; 2010. p. 49–62.

Cui Y, Ma X, Wang H, Stojmenovic I, Liu J. A survey of energy efficient wireless transmission and modeling in mobile cloud computing. Mob Netw Appl 2013;18:148–55.

Damaševičius R, Štuikys V, Toldinas J. Methods for measurement of energy consumption in mobile devices. Metrol Meas Syst 2013;20:419–30.

Das PK, Ghosh D, Joshi A, Finin T. ACM HotMobile 2013 Poster: an energy efficient semantic context model for managing privacy on smartphones. ACM SIGMOBILE Mob Comput Commun Rev 2013;17:34–5.

Ding F, Xia F, Zhang W, Zhao X, Ma C. Monitoring energy consumption of smartphones. In: Proceedings of the IEEE 4th international conference on cyber, physical and social computing, and internet of things (iThings/CPSCom); 2011. p. 610–3.

Do T, Rawshdeh S, Shi W. ptop: A process-level power profiling tool. In: Proceedings of the 2nd workshop on power aware computing and systems (HotPower'09); 2009.

Dong M., Zhong L. Sesame: Self-Constructive System Energy Modeling for Battery-Powered Mobile Systems. arXiv preprint arXiv:10122831. 2010.

Donohoo BK, Ohlsen C, Pasricha S. AURA: an application and user interaction aware middleware framework for energy optimization in mobile devices. In: Proceedings of the IEEE 29th international conference on computer design (ICCD); 2011. p. 168–74.

Duan L-T, Guo B, Shen Y, Wang Y, Zhang W-L. Energy analysis and prediction for applications on smartphones. J Syst Archit 2013;59:1375–82.

Dubey D, Amritphale A, Sawhney A, Amritphale N, Dubey P, Pandey A. Smart phone applications as a source of information on stroke. J Stroke 2014;16:86–90.

Durand J, Flores J, Atkison T, Kraft N, Smith R. Using executable slicing to improve rogue software detection algorithms. Int J Secur Softw Eng 2011;2:53–64.

Flinn J, Satyanarayanan M. Powerscope: a tool for profiling the energy usage of mobile applications. In: Proceedings of the second IEEE workshop on mobile computing systems and applications (WMCSA'99); 1999. p. 2–10.

Geronimo D, Lopez AM, Sappa AD, Graf T. Survey of pedestrian detection for advanced driver assistance systems. IEEE Trans Pattern Anal Mach Intell 2010;32:1239–58.

Giammarini M, Conti M, Orcioni S. System-level energy estimation with Powersim. In: Proceedings of the 18th IEEE international conference on electronics circuits and systems (ICECS); 2011. p. 723–6.

Gurun S, Krintz C. A run-time, feedback-based energy estimation model for embedded devices. In: Proceedings of the 4th international conference on hardware/software codesign and system synthesis: ACM; 2006. p. 28–33.

Haffey F, Brady RR, Maxwell S. Smartphone apps to support hospital prescribing and pharmacology education: a review of current provision. Br J Clin Pharmacol 2014;77:31–8.
Hao S, Li D, Halfond WG, Govindan R. Estimating Android applications' CPU energy usage via bytecode profiling. In: Proceedings of the first international workshop on green and sustainable software: IEEE Press; 2012. p. 1–7.
Hao S, Li D, Halfond WG, Govindan R. Estimating mobile application energy consumption using program analysis. In: Proceedings of the IEEE 35th international conference on software engineering (ICSE); 2013. p. 92–101.
Hayes W, Naziri Q, De Tolla JE, Akamnonu CP, Merola AA, Paulino C. A systematic review of all smart phone applications specifically aimed for use as a scoliosis screening tool. Spine J 2013;13:S38.
Hernández G, Hernández CJB, Garino GDCG, Pérez-Acle SNT, Vázquez MSM. High Performance Computing, 2014.
Hsieh M-Y, Yeh C-H, Tsai Y-T, Li K-C. Toward a mobile application for social sharing contextz. Mobile, ubiquitous, and intelligent computing. Berlin, Heidelberg: Springer; 93–8.
Huang J, Xu Q, Tiwana B, Mao ZM, Zhang M, Bahl P. Anatomizing application performance differences on smartphones. In: Proceedings of the 8th international conference on mobile systems, applications, and services: ACM; 2010. p. 165–78.
Jiemsakul T, Trithaveesak O, Bunjongpru W, Hruanun C, Poyai A, Nukeaw J. Application of double gate ion sensitive field effect transistor for detection of fluid flow rate in micro-channel. Adv Mater Res 2010;93:109–12.
Keleshis C, Ioannou S, Vrekoussis M, Levin Z, Lange M. Data Acquisition (DAQ) system dedicated for remote sensing applications on Unmanned Aerial Vehicles (UAV). In: Proceedings of the second international conference on remote sensing and geoinformation of the environment (RSCy): International Society for Optics and Photonics; 2014. p. 92290H-H-8.
Khan A, Othman M, Madani S, Khann S. A survey of mobile cloud computing application models; 2013.
Khan AR, Othman M, Madani SA, Khan SU. A survey of mobile cloud computing application models. Commun Surv Tutor IEEE 2014;16:393–413.
Kim K, Shin D, Xie Q, Wang Y, Pedram M, Chang N. FEPMA: fine-grained event-driven power meter for android smartphones based on device driver layer event monitoring. In: Proceedings of the conference on design, automation & test in Europe: European Design and Automation Association; 2014. p. 367.
Kjærgaard MB, Blunck H. Unsupervised power profiling for mobile devices. Mobile and ubiquitous systems: computing, networking, and services. Berlin, Heidelberg: Springer; 138–49.
Kjærgaard MB, Langdal J, Godsk T, Toftkjær T. Entracked: energy-efficient robust position tracking for mobile devices. In: Proceedings of the 7th international conference on Mobile systems, applications, and services: ACM; 2009. p. 221–34.
König I, Memon AQ, David K. Energy consumption of the sensors of Smartphones. Wireless Communication Systems (ISWCS 2013). In: Proceedings of the tenth international symposium on: VDE; 2013. p. 1–5.
Koo C, Shin S, Kim K, Kim C, Chung N. Smart tourism of the Korea: a case study. PACIS; 2013. p. 138.
Kumar K, Lu Y-H. Cloud computing for mobile users: can offloading computation save energy? Computer 2010;43:51–6.
Lauridsen M, Noël L, Sørensen TB, Mogensen P. An empirical LTE smartphone power model with a view to energy efficiency evolution. Intel Technol J 2014;18:172–93.
Lee J, Joe H, Kim H. Automated power model generation method for smartphones. IEEE Trans Consum Electron 2014;60:190–7.
Liew CS, Wah TY, Shuja J, Daghighi B. Mining personal data using smartphones and wearable devices: a survey. Sensors 2015;15:4430–69.
Lin P-J, Chen S-C, Yeh C-H, Chang W-C. Implementation of a smartphone sensing system with social networks: a location-aware mobile application. Multimed Tools Appl 2013;74:1–12.
Lin Y-D, Rattagan E, Lai Y-C, Chang L-P, Yo Y-C, Ho C-Y, et al. Calibrating parameters and formulas for process-level energy consumption profiling in smartphones. J Netw Comput Appl 2014a;44:106–19.
Lin Y-D, Rattagan E, Lai Y-C, Chang L-P, Yo Y-C, Ho C-Y, et al. Calibrating parameters and formulas for process-level energy consumption profiling in smartphones. J Netw Comput Appl 2014b;44:106–19.
Linares-Vásquez M, Bavota G, Bernal-Cárdenas C, Oliveto R, Di Penta M, Poshyvanyk D. Mining energy-greedy API usage patterns in Android apps: an empirical study. In: Proceedings of the 11th working conference on mining software repositories: ACM; 2014. p. 2–11.
Liu J, Ahmed E, Shiraz M, Gani A, Buyya R, Qureshi A. Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. J Netw Comput Appl 2015;48:99–117.
Liu J, Priyantha B, Hart T, Ramos HS, Loureiro AA, Wang Q. Energy efficient GPS sensing with cloud offloading. In: Proceedings of the 10th ACM conference on embedded network sensor systems: ACM; 2012. p. 85–98.
Madani SA. Pirax: framework for application piracy control in mobile cloud environment. J Supercomput 2014;68:753–76.
Marotz L. Health, safety, and nutrition for the young child. Cengage Learning; 2014.
Mittal R, Kansal A, Chandra R. Empowering developers to estimate app energy consumption. In: Proceedings of the 18th annual international conference on mobile computing and networking: ACM; 2012. p. 317–28.
Mosa ASM, Yoo I, Sheets L. A systematic review of healthcare applications for smartphones. BMC Med Inform Decis Mak 2012;12:67.
Navda V, Ramjee R, Schulman A, Padmanabhan VN, Jain K. Scheduling communications in a mobile device. Google Patents; 2013.
Oliner AJ, Iyer AP, Stoica I, Lagerspetz E, Tarkoma S. Carat: collaborative energy diagnosis for mobile devices. In: Proceedings of the 11th ACM conference on embedded networked sensor systems: ACM; 2013. p. 10.
Oliver EA, Keshav S. An empirical approach to smartphone energy level prediction. In: Proceedings of the 13th international conference on ubiquitous computing; 2011. p. 345–54.
Pandhare R, Joglekar S. Smartphone enterprise applications. ACEEE Int J Netw Secur 2011;2 http://dx.doi.org/01.IJNS.02.02.68.
Papalkar RR, Ade SA, Abhale A, Nagaraj U, Yadav RK. Energy saving on Wi-Fi survey. Energy. 2014:1.
Pathak A, Hu YC, Zhang M. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In: Proceedings of the 7th ACM European conference on computer systems; 2012. p. 29–42.
Qian F, Wang Z, Gerber A, Mao Z, Sen S, Spatscheck O. Profiling resource usage for mobile applications: a cross-layer approach. In: Proceedings of the 9th international conference on mobile systems, applications, and services: ACM; 2011. p. 321–34.
Rahimi MR, Ren J, Liu CH, Vasilakos AV, Venkatasubramanian N. Mobile cloud computing: a survey, state of art and future directions. Mob Netw Appl 2014;19:133–43.
Rice A, Hay S. Decomposing power measurements for mobile devices. In: Proceedings of the IEEE international conference on pervasive computing and communications (PerCom); 2010a. p. 70–8.
Rice A, Hay S. Measuring mobile phone energy consumption for 802.11 wireless networking. Pervasive Mob Comput 2010b;6:593–606.
Ryan AG, Montgomery DC, Peck EA, Vining GG. Solutions manual to accompany introduction to linear regression analysis. John Wiley & Sons; 2014. ISBN: 978-1-118-47146-3.
Sanaei Z, Abolfazli S, Gani A, Buyya R. Heterogeneity in mobile cloud computing: taxonomy and open challenges. IEEE Commun Surv Tutor 2014;16:369–92.
Sanaei Z, Abolfazli S, Khodadadi T, Xia F. Hybrid pervasive mobile cloud computing: toward enhancing invisibility Information; October 2013.
Sasu Tarkoma MS, Lagerspetz Eemil, Xiao Yu. Smartphone energy consumption modeling and optimization. United Kingdom: University of Helsinki: University Printing House, Cambridge CB2 8BS; 2014.
Seop SH, Young MG, Hoon JD. A study on the development of disaster information reporting and status transmission system based on smart phone. In: Proceedings of the EEE international conference on ICT convergence (ICTC); 2011. p. 722–6.
Shiraz M, Gani A, Ahmad RW, Shah SAA, Karim A, Rahman ZA. A Lightweight distributed framework for computational offloading in mobile cloud computing. PloS One 2014;9:e102270.
Shiraz M, Gani A, Khokhar RH, Buyya R. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. Commun Surv Tutor IEEE 2013;15:1294–313.
Shuja J, Bilal K, Madani SA, Othman M, Ranjan R, Balaji P, et al. Survey of techniques and architectures for designing energy-efficient data centers; 2014.
Shuja J, Madani SA, Bilal K, Hayat K, Khan SU, Sarwar S. Energy-efficient data centers. Computing 2012;94:973–94.
Thiagarajan N, Aggarwal G, Nicoara A, Boneh D, Singh JP. Who killed my battery?: analyzing mobile browser energy consumption. In: Proceedings of the 21st international conference on World Wide Web: ACM; 2012. p. 41–50.
Tiwari V, Malik S, Wolfe A. Power analysis of embedded software: a first step towards software power minimization. IEEE Trans Very Large Scale Integr Syst 1994;2:437–45.
Tsao S-L, Kao C-C, Suat I, Kuo Y, Chang Y-H, Yu C-K. PowerMemo: a power profiling tool for mobile devices in an emulated wireless environment. In: Proceedings of the IEEE international symposium on System on Chip (SoC); 2012. p. 1–5.
Tu C-H. On-line learning migration: from social learning theory to social presence theory in a CMC environment. J Netw Comput Appl 2000;23:27–37.
Vallina-Rodriguez N, Crowcroft J. Energy management techniques in modern mobile handsets. IEEE Commun Surv Tutor 2013;15:179–98.
Van Beeck K, Goedemé T, Tuytelaars T. Towards an automatic blind spot camera: robust real-time pedestrian tracking from a moving camera. In: Proceedings of the twelfth IAPR conference on machine vision applications; 2011. p. 528–31.
Walker MJ, Das AK, Merrett GV, Hashimi B. Run-time power estimation for mobile ad embedded asymmetric multi-core CPUs; 2015.
Wang L. Understanding and using contextual information in recommender systems. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval; 2011. p. 1329–30.
Wigley J, Shantikumar S. Traumatutor: perceptions of a smartphone application as a learning resource for trauma management. J Biomed Educ 2013 ⟨http://dx.doi.org/10.1155/2013/149162⟩, (Article ID 149162), 3 pp.
Wisniewski M, Demartini G, Malatras A, Cudré-Mauroux P. NoizCrowd: a crowd-based data gathering and management system for noise level data. Mobile web and information systems. Berlin, Heidelberg: Springer; 172–86.
Yan Y, Xiong S, Lou X, Xiong H, Miao Q. Design and implementation of educational administration information access system based on android platform. Frontier and future development of information technology in medicine and education. Netherlands: Springer; 2525–33.
Yoon C, Kim D, Jung W, Kang C, Cha H. AppScope: application energy metering framework for android smartphone using kernel activity monitoring. USENIX annual technical conference; 2012. p. 387–400.
Zhang L, Tiwana B, Qian Z, Wang Z, Dick RP, Mao ZM, et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In: Proceedings of the eighth IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis; 2010. p. 105–14.