On Power and Energy Consumption Modeling for Smart Mobile Devices

M. Ferroni, A. Cazzola, F. Trovò, D. Sciuto, M. D. Santambrogio
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
Politecnico di Milano, Milano, Italy
{matteo.ferroni, fancesco1.trovo, donatella.sciuto, marco.santambrogio}@polimi.it
{andrea.cazzola}@mail.polimi.it

***Abstract*—In nowadays life, mobile phones are becoming a cheaper and smaller alternative to laptops for simple, everyday tasks. They experienced an astonishing growth in functionalities and, because of their constant presence in our life, mobile phones became fundamental for the interaction with information coming from the environment. Nevertheless, their resources are limited, both in terms of performance and power, and their availability can greatly vary over time. Especially when dealing with power consumption, mobile devices cannot disregard environment conditions and user habits. Both internal and external conditions are rapidly changing and may influence the response of the entire system, e.g., switching between network types may causes an unpredictable power consumption. In order to puzzle out all these issues, we regard the definition of a power/energy model for mobile devices as a first mandatory step. In literature, several attempts to do so are present, basing their approaches on techniques coming from different computer science fields. They differ in the way they consider hardware components, in the operating system they are suitable for and in the scope of their tests and experiments. Within this paper, we categorize techniques presented in the major works in the field, in order to be able to compare different methods, highlight open issues and give suggestions on future works.**

***Keywords*-Introductory; Survey; Measurement; Theory**

I. Introduction

Mobile devices are getting more and more popular in nowadays life. They are now cheaper, smaller and much portable than laptops and they are substituting them in a large variety of functionalities [1]. Due to their growth in functionalities and constant presence in our life, mobile devices are one of the most important interaction points between people and the surrounding environment. However, it has to be considered that resources management in such devices should be specifically crafted to satisfy the mobile requirements: power sources availability can greatly vary over time, computational power is limited [2], as well as the disk space. Both internal and external conditions change quickly, influencing the behavior of the entire system, e.g., not having a fixed localization causes the switch between network types, causing an unpredictable power consumption and network instability. Within this context, it is clear that mobile devices belong to a constrained and fluctuating environment, which requires resource allocation policies that differ from those which have been developed for traditional computational world [3], [4], [5]. One of the most remarkable constraint is the battery power: this is a fundamental and highly constrained resource, thus its management has both great research interest and practical applications [6], [7]. The constraint imposed by the battery capacity suggests the idea of developing accurate models for power consumption of mobile devices, to understand if an optimized usage profile may increase the device lifetime.

In literature, there exist several attempts to define power/energy models for mobile devices, whose approaches come from different fields of computer science. Carroll and Heiser presented a work about the power consumption of a Openmoko Neo Freerunner mobile phone, whose circuit schematics are publicly available [8]. The main purpose of their studies was to induce a power distribution breakdown in the main components of a smart device. Even if it provides a useful insight on power consumption of smartphone components, their technique relies in publicly available component schematics, therefore they are not portable. Another approach overcoming this issues relies on the use of sensors available on the smartphone, to build the power model. PowerBooter and PowerTutor [9] automatically build a model using built-in battery voltage sensors to monitor power consumption and control the power management of individual components. A third approach is based on a logging application released *into the wild* [10], using the data retrieved to built the power model. Thanks to this technique, the authors demonstrated that the power consumption strongly depends on the owner's behavior too, highlighting its impacts on the modeling abilities.

Given the huge number of works in the field of power modeling, we strongly believe that a categorization is necessary, to promote the comparison among these different techniques. Differently from what presented in [2], we decided to give an overview on the characteristics of the most influential research works in this field. Thus, this survey is written focusing on measurements and models, to catalog the state-of-the-art according to the level of abstraction used in estimation, i.e., system, application or user level. In addition, a set of discriminant features able to characterize mobile devices power estimation methods is presented here.

The remaining part of this paper is organized as follows: Section II presents the adopted categorization criteria; Sections III to V describe and analyze the models developed so far, with a detailed insight of the most remarkable works; Section VI highlights the open issues with currently available methods, and finally Section VIII discusses the conclusions of this work.

II. Structure

In order to compare and analyze the most influential research works in the field of power consumption modeling on mobile devices, we defined a set of **categories** to group similar methodologies and better evidence their differences. This taxonomy is based on the level of abstraction used for data gathering, which induces different modeling approaches. The chosen categories are:

- **System level**: the works in Section III have, as their primary goal, the development of a model able to compute the whole system power consumption at a given time. This category is further divided by considering the way in which power measurements are performed, i.e., using external systems, internal HW/SW components or via internal APIs of the operating system.
- **Application level**: in Section IV, the power consumption is modeled per single application or per applications that have specific workload characteristics, e.g., an intensive network usage.
- **User level**: Section V includes all the works taking into account also the user behavior and the impact of different usage patterns in overall power consumption.

This high level classification summarizes the existing power modeling methodologies, giving just a bird's eye view of the approaches to the problem: each method should be then properly described to better compare techniques and to perform a more detailed analysis. A set of discriminant features/aspects is here reported:

(a) **Model generation**: the method used to build the model from the available data, e.g., Finite State Automata (FSA) or Linear Regression (LR), (see Table I for further details);
(b) **Model granularity**: the level of detail of the model, i.e., the entire system, a single application, or a single component;
(c) **Model adaptivity**: the ability of the model to change according to changes into the environmental conditions;
(d) **User behavior**: whether the user behavior is taken into consideration during the model construction phase or in the testing phase;
(e) **Analyzed data source**: whether the evaluation phase is performed using ad-hoc stress-applications or real applications or data (see Table I);
(f) **Target devices/OS**: the devices that the system is able to model and/or the OS they are running on (see Table II);
(g) **Measurement method**: whether the information about the battery status or the current power consumption is retrieved using an external or internal tools/instrumentation (see Table I).

III. System-level Power Modeling

A large number of works aims at creating system-wide power models of the entire mobile device. The required data can be gathered either by attaching external sensors to the device (**offline** method) or relying on the operating system's APIs (**online** method).

An **offline** approach allows an accurate inspection of the behavior of the mobile device, given predetermined and controlled external conditions. Given the great variety of mobile devices on the market and the OS release versions they may run, it is almost unfeasible to generate offline power consumption models for each combination of device and OS version. Moreover, this analysis does not evolve with the running life of the device, while an **online** methodology and a run-time generated model is able to adapt itself to new software updates or even new devices. On the other hand, an adaptive model generation cannot be as precise as the one developed using external measurement system, relying only on software APIs to gather information on the actual battery state.

Recently, some smartphones are given a wider set of hardware sensors, thus allowing an intermediate approach to be used, with internal sensors providing precise data about the device power consumption. This last approach provides intermediate precision results between the external measurements and the APIs approach, but it lacks of flexibility, since can be applied only to those devices equipped with internal sensors.

With these considerations in mind, we will now analyze how different measurement methods have been applied in some remarkable works in this field: Section III-A presents some results using external measurements, while Section III-B discusses those works that make use of system APIs; finally, approaches based on custom and internal measurements are described in Section III-C.

A. Using External Measurements

The models shown in this section rely on external tools to gather data from the device. This approach provides very precise information, but it is really invasive.

The goal of [19] is to consider the contribution of independent component on the whole energy consumption. The power consumption is measured by inserting a resistor in series between the battery and its connector on the phone and a sampling board is used to measure the battery voltage. Measurements are managed by a central *Power Server*, which is in charge of sending the test scripts on the device and retrieve the resulting traces without requiring any user interaction. Tests are specifically designed to stress each considered component, providing a model of power consumed by a specific component. The analysis performed involved data sending over a wireless network, using an Android device. Collected traces allowed the analysis of the energy consumed in different communication phases, e.g., during data sending phase.

In an influential study, Anand *et al.* [11] propose a 2-way Operating System-level power management. This system consists in a modified version of Linux, running on a handheld iPAQ. I/O peripherals are fully described using a state enumeration, called *power modes*, each one associated to a power consumption values expressed in mW. In order to measure the power states, the authors removed the battery from the iPAQ device and sampled the amount of current drawn at 50Hz through an

TABLE I
CATEGORIES AND RELEVANT FEATURES OF MAIN WORKS (MODEL GENERATION TECHNIQUES ARE FINITE STATE AUTOMATON (FSA), LINEAR REGRESSION (LR), GENETIC ALGORITHM (GA), PRINCIPAL COMPONENT ANALYSIS (PCA), FINITE STATE MACHINE (FSM), AUTO REGRESSION (AR) AND FACTOR ANALYSIS (FA).

Category	Work	Features						
		Model Generation	Model Granularity	Model Adaptivity	Measurement Method	Analyzed Data Source	User Behavior	Target OS
System Level	[10]	FSA + LR	System/Component	no	External Tools	stress/users	yes	Android
	[11]	Heuristic				stress	no	Linux
	[12]	GA			Custom Internal	stress/users		Symbian
	[9]	LR				stress		Android
	[13]	PCA + LR	System	yes				Linux/Symbian/Android
	[14]	FSA + LR			Internal			Windows Phone/Android
Application Level	[15]	Component	Application	yes	External Tools	stress	no	Android
	[16]			no	Custom Internal	stress/users		Symbian
User Level	[17]	PCA + FA	System	yes	Internal	user logs	yes	Android
	[18]	AR						

TABLE II
EMBEDDED HARDWARE COMPONENTS CONSIDERED BY SOME RELEVANT PAPERS IN THIS SURVEY

Work	HW Components Treated as Independent	Screen	CPU	GPS	SD Card	Bluetooth	Audio	Network	Wi-fi	None
[10]	√	√	√		√		√	√	√	
[11]										√
[12]	√		√	√				√	√	
[9]	√	√	√	√	√	√	√	√	√	
[13]	√	√	√		√			√	√	
[14]	√		√		√			√	√	
[15]	√	√	√	√		√		√	√	
[16]	√								√	
[17]		√	√	√		√		√	√	
[18]										√

external power supply. In this system, applications are allowed to query the power mode of I/O peripherals and can disclose *hints* about expected device discharge rate in a different power-mode. The system provides also a decision phase for power saving, relying on heuristics based on the *hints* given to the applications. Experiments are performed focusing on a Web browser and an email reader application.
Another remarkable contribution is Cinder, an OS based on the HiStar kernel [20]. Cinder provides a *low-level abstraction* for energy management and it is meant specifically for mobile, energy constrained devices. Measurements are taken through an Agilent Technologies E3644A (i.e., a DC power supply with a current sensor) on an HTC Dream mobile phone. Even though this research work does not directly focuses on power modeling, the authors clearly state that Cinder needs a solid model in order to work properly: the suggested approach was to "*build a model from offline-measurements of device power states in controlled setting*".
A very deep analysis of power consumption on smartphones was done by Carrol and Heiser [8]. They tried to understand where and how energy is drained, to provide basis for understanding and managing mobile devices power consumption. Their considered physical measurements of supply voltage and current at component level on a piece of real hardware, using a National Instruments PCI-6229 DAQ. With this experimental setup, the authors were able to directly estimate the power consumed by the main components of the mobile device (see Tab. II for details). Moreover, they measured total power consumption by inserting a sense resistor between the power supply and the device. To construct their model, they ran two different benchmarks: at first, a series of micro-benchmarks designed to independently characterize single components, then a series of macro-benchmarks based on real usage scenarios. They noticed that the majority of power consumption can be attributed to the GSM module, the display (including the LCD panel and touchscreen), the graphics accelerator/driver, and the backlight.
In almost all benchmarks, the brightness of the backlight was the most critical factor in determining power consumption. They also showed the impact of the screen image on power consumption. The GSM module consumes a significant amount of both static and dynamic power (both maintaining a connection with the network and during a phone call), while the RAM, audio and flash subsystems showed the lowest power consumption. Using the data collected, Carrol and Heiser built an energy model based on usage patterns in order to understand when and where the daily energy use and battery life is wasted. The defined usage patterns are: *suspend* (a baseline case of a device which is on standby), *casual* (user who uses the phone for a small number of voice calls and text messages each day), *regular* (a commuter with extended time of listening to music or podcasts, combined with phone calls of different length, messaging and emailing) and *business* (a user that makes talking and email use together with some web browsing). This kind of analysis clearly shows how different usage have different impact on the power consumption.
Shye et al. describe in [10] both a power-modeling approach for Android-based mobile devices and a novel energy-saving policy to manage the screen brightness. Measurements are performed with a Fluke i30 AC/DC current clamp, while the operating voltage is retrieved through Android APIs. The modeling phase consists of a 2 states FSA: *stand-by mode* and

active mode. In the former state the consumption is accounted as fixed, while in the latter one the power consumption is computed through a LR model, given by the R-Tool fed with data collected by stressing a single hardware component at a time. This model predicts the power behavior of each scenario with a median 6.6% relative error. Moreover, the authors used data coming from several users to identify the most consuming component, which turn out to be the screen. By making the system automatically adjust the screen brightness, they managed to save up to 10% total energy, with minimal impact on user satisfaction.

B. *Using only system APIs*

Different power modeling approaches rely only on device APIs to collect data. OS APIs have some limitations in terms of precision, but they are present on every device and are less invasive than direct measurements.
For instance, Xiao et al. [21] present a methodology for building a system-level power model, without requiring laboratory measurements. This approach relies only on data provided by the OS, thus it can be adapted to any device. They developed a LR model with non negative coefficients, describing the aggregate power consumption of processors, wireless network interface and display. In order to perform LR, they analyzed data regarding the activity levels of each hardware component, i.e., the hardware performance counters (HPCs) for processor, the downlink and uplink data rates for the wireless interfaces, and the brightness level for the display. They discovered that a LR model is sufficient to model the relation between the variables they chose and the power consumption. They used five types of different workloads to construct their model: idle with different brightness levels, audio/video playing, audio/video recording, file download/upload at different network data rates and data streaming. One of the main features of this work is the possibility to adapt the model to new hardware. In fact, when a new hardware component is installed into the mobile device, it is possible to add regression variables describing the activity levels of the new component, define new test cases to stress it and, finally, fit the new data with a regression model. If an analytical power model of the new hardware component is provided, it is also possible to merge it with the existing system-level model. They reported a median error of 2.62% in the power estimation in real mobile Internet services. Moreover, they provided a power model that is independent from usage scenarios and that can be used for runtime power estimation with reasonable accuracy.
In [22], a context-aware system able to accurately predict the battery lifetime is proposed. The energy consumed by each system component is considered dependent on its operational state and on the amount of time spent in the state. As a consequence, the system power consumption is modeled as the sum of single components. Data about the discharge rate were collected in several contexts, where a single system context is a combination of CPU utilization, LCD brightness, Wi-Fi state, IO idle rate and volume of data transferred. Finally, multiple LR technique was used to build a model able to describe the battery discharge rate, by basing on the system component states. The system was tested using a HTC G1 smart phone, running the Android OS. Data were collected using 40 different test scenarios: 16 of those were used to build the model, the remaining to evaluate the generated model. The proposed model was able to predict the remaining battery lifetime with a relative error of 10%.
Another remarkable contribution is given by Pathak et al. [14], who propose a new power modeling approach, able to capture both utilization-based and non-utilization-based power behavior. This methodology is based on a fine-grained energy estimation, obtained by tracing applications system calls. Their scheme consists of two major components. The first one is a Finite State Machine (FSM) to model the power states and state transitions. Some of the states have constant power consumption (they represent non-utilization power consumption), while others leverage on a LR model (the second major component of the model) to capture the power consumption due to system calls of a generated workload. Moreover, a testing application is used to systematically uncover the FSM transition rules. Tests were performed on a HTC Touch Pro and a HTC Tytn 2, powered by Windows Mobile 6, and a HTC Magic running Android. They claim this new modeling approach improves the accuracy of fine-grained energy estimation compared to utilization-based model. Indeed, their model have a 80th percentile error of less than 10% estimating the power consumption of 50 ms of a generic application execution, while the utilization-based model have an error that varies between 16% and 52%. Its error for the whole application, with 1 sec granularity, varies between 0.2% and 3.6%, compared to the error 3.5-20.3% given by utilization-based model.

C. *Using Custom and Internal Measurements*

Several works have been specifically designed to exploit interfaces already available on specific devices or OSes, e.g., the *Advanced Configuration and Power Interface (ACPI)* or the *Nokia energy profiler*. The use of these interfaces provides precise information about the battery status (e.g., voltage, current and temperature) or even the current power consumption of the device at a specific time instant. These choices lead to the generation of very precise power consumption models, but they are not portable on devices without specific hardware interfaces, e.g., many Android-powered device may be unable to provide the required data with the required precision.
An interesting system called *Sesame* is proposed in [13]. It is a self-modeling approach to build high-rate mobile system models without any need for external measurements. At first, Sesame collects data traces, i.e., system statistics and data provided by the ACPI, building an initial set of predictors; they may be user-defined and may depend on the platform in use. Predictors may include, among the others, CPU utilization, cache misses, Wi-Fi traffic and/or LCD backlight level. Then, the model is built using two iterative techniques, *model molding* and *predictor transformation*. The former generates a model using LR, while the latter improves the accuracy of a molded model, transforming the original predictors and finding better linear combinations of the original predictors. The

model is composed by a set of sub-models, each corresponding to a different system configuration. Sesame was implemented both for laptops and smartphones, using a Linux kernel. The system overhead, in the worst case, has been computed to be 3% and 12% on the laptop and on the smartphone, respectively, while average values are measured to be around 1% and 5%. This study highlighted that data collection frequency influences the system accuracy, drastically dropping when the data collection frequency matches the frequency of the register used to store the monitored value. Accuracy was reported as 86% and 82% at 1 Hz and 100 Hz on the smartphone and 95% and 88% at 1 Hz and 100 Hz on the laptop, respectively. In the model generation, Sesame provides energy estimations with error of at most 12% at 1 Hz and 18% at 100 Hz for laptops, and 12% at 1 Hz and 22% at 100 Hz for a Nokia N900 smartphone.

Another work exploiting the ACPI interface is proposed in [23]: the proposed methodology involves a mixed-approach, combining the recently observed device power consumption history with offline benchmark measurements to predict the battery lifetime. The offline measurements are used to offer a baseline for prediction, while more recent data are used to react to unpredictable events online. Offline measurements were computed running on a quiescent system with constant-power workloads, i.e., repeated executions of a single program to completely discharge a fully charged battery. Starting from the offline history curve, several LRs are used to predict the battery lifetime, taking into consideration the current system conditions too. The online data are retrieved using Smart Battery [24] and the ACPI interface. Their method achieved a 5% maximum error for predicting the power consumption of constant workloads, with the exception of media, whose fluctuations during execution cause the error rate to reach the 10%.

The work described in [12] targets Nokia devices with Symbian OS and exploits the Nokia Energy Profiler tool, which provides precise power measurements with respect to OS APIs. An unsupervised method to model a device energy consumption is proposed, based on Genetic Algorithms (GA). This solution relies on online measurements through system APIs to generate on-demand models: for instance, a new model may be needed when the system configuration changes. Data retrieved through the available APIs include the battery voltage, current and other informations. In order to build a model, their method needs a training dataset where data are collected exercising independently all the remarkable phone features; a GA is then used to choose the relevant features. The features exercised include Wi-Fi, GSM, CPU and GPS. The presented experiments considered training data mostly collected in predefined conditions. However, the system was tested using a real-location use case scenario too. The model predicts power consumption with a 95th percentile error of 0.313. This error increases when the application is released "into the wild" with more frequent and overlapping feature usage.

In [25] an energy estimation system has been presented: it is able to compute at run-time the power consumption of tasks executed in RAM and to perform data transfers over the Wi-Fi network. In this work, a battery monitor unit is assumed to be already integrated into the target devices. At first, an initial offline profiler is used to generate a first model state, using the profiled data. Then, a power estimator component produces the energy estimation on a defined interval, taking into consideration a computation model and a communication model. Those models are characterized by a set of parameters, computed at run-time using a recursive least square LR with exponential decay method. This phase uses the feedbacks from a runtime profiler, which gather information about the battery voltage, temperature and current from the battery monitor.

The last remarkable contribution in this section is by Zhang et al. [9], specifically crafted for online generation of power models. They started their study analyzing each component separately, using external tools. They realized that the power consumption of major components affects the system independently, e.g., power consumption of the entire system when 3G and Wi-Fi are active is the sum of 3G and Wi-Fi contributions. After that, they evaluated the intra and inter class variance and noticed that different devices have power models pretty far from each other: this justifies the need to develop a power model for each device. In order to do so, their method was modified to be able to create power models, starting from data coming from sensors within each device. The result indicates that the power model built with PowerBooter is accurate to within 4.1% of measured values for 10-second intervals. However, this model presents two main limitations: the need of a specific discharge curve for every specific device and the need for a mobile phone that allows superuser access in the OS.

IV. Application-level Consumption

This section discusses those works that profile applications power consumption. Compared to system-wide models, this approach allows a more fine-grained profiling. In addition, several works studied the impact of network data transmission of the single application, since the network activity consumes a significant part of the available battery power.

A tool for power-aware applications is proposed in [15]. The idea is to predict the power consumption of an application during its development phase. The goal is to identify "hot-spots" inside the code before the application is released and tested on an actual device: the tool presented, named *System Power Optimization Tool (SPOT)*, allows developers to create high-level architectures representing the program and the components it uses. SPOT generates the code and integrates the information about energy consumption, with an error of 3-4%, with respect to the actual power consumption. Alternatively, it can create the code to be run on a real device, adding some logging functionalities to save useful information to refine the power model offline. The main limitations of SPOT are the lack of integration with the Android SDK and the limited number of devices available to test an application.

In [26], the goal is to model energy consumption of those applications that require a network activity, for each available technology (i.e., 3G, GSM and Wi-Fi). This study considers two main contributions to power consumption due to network activity in cellular mode: the transmission energy and the Radio Resource Control protocol, responsible for scaling the power consumed by the radio basing on inactivity timers. They reported an initial high cost of association with an access point in Wi-Fi mode, but a low cost of maintaining the connection active. Experiments were performed using the *Nokia Energy Profiler* to gather precise power consumption information in real-time. Results have shown that GSM consumes 40% to 70% less energy compared to 3G to download data, while Wi-Fi is more energy efficient than both cellular networks once it is associated. The energy model was build taking into account both the size of transfer and the time between consecutive transfers.

In [16], the analysis focuses only on the Wi-Fi connection. Even in this case, power consumption was measured using the *Nokia Energy Profiler*. During the measurements, the basic components of the devices were in use, to minimize the dependencies with other components. Power consumption data has been collected testing the network interface in different operational modes and during TCP download at different data rates. The power consumption was modeled using a set of linear equations. With these equations, they estimated the power consumed to upload or download with a mean absolute percentage error of 6.8% and 5.8%, respectively.

The last remarkable application of power profiling is described in [27]. In this study, application power profiles were used for malware detection. The authors estimated the power consumption of various activities done by the user, such as *calling* or *surfing the Web*. Using these measures as a ground truth, they monitored power consumption during device usage, in order to spot abnormal consumptions. Then, when the phone was recharging, they used a more fine grained analysis on each application, in order to spot the ones that are contributing the most to the power consumption variations. Using this kind of analysis, they managed to achieve a 89% detection rate, in case of a message forwarding malware.

V. User-level Consumption

Mobile devices are able to run several different applications and the amount of time spent by a user on each one of those contributes to determine the device battery lifetime. Since each user has different needs, it would be reductive to build efficient power consumption models without taking into consideration the impact of the user behavior. This section discusses those works which took into account the user experience, to properly build personalized power consumption models.

Vallina-Rodriguez *et al.* clearly explains why those models that ignore the users usage patterns are limited [17]. They developed an application to collect information about the users, which were analyzed using the Principal Component Analysis (PCA), a statistical approach able to discover the component with the biggest impact on power consumption. The results showed that the components consuming the most were strictly dependent on the user. As a consequence, power optimizations may be performed on unused components, if user's behavior would not have been taken into consideration. However, this work showed that PCA can be used to highlight each device major consuming components and to build personalized power models.

In [28], Kang et al. created a power model considering the usage pattern, thus overcoming the limitation of the models considering only the device. They consider a mobile device having a constant number of states, each of which is a tuple containing the condition of the phone components, e.g., a state is determined by (LCD;VOICE;DATA) = (ON;ON;ON). The authors noticed that users spend different time into each state. In [18] they create a daily and weekly user profile, reflecting life patterns, i.e., working, resting, and sleeping for both weekdays and weekends. Then, they measured the battery consumption and time spent in each operational state, using a data collecting application running on Android-based platforms. The application periodically recorded information about usage patterns and power consumption to a log file, later sent to and processed by a server, to build the model. This was the same concept of PowerDoctor [29]: it makes use of real world data to build user-specific power consumption models. As far as we know, it has been tested considering only the CPU and the display as modeled components, but the authors claim that an extension of the work to all the device components is possible. PowerDoctor groups data into chunks, one for every 1% drop of the battery level. These chunks are then organized according to the active hardware components, considering only those with little standard deviation; a LR is finally used to build the model. Experimental results showed that this application can predict power consumption with a percentage error between 5.7% and 7.2%.

A detailed and wide study on how intentional user activities impact on the device power consumption is performed in [30]. One aspect of this data analysis involves the energy consumption: it is claimed that the energy consumption is influenced by the platform itself (hardware/software components) and by the applications and the user interaction. Their results showed that user activities contribute heavily towards energy drain. The same authors presented a study about network connectivity on smartphones [31], implementing a tool that provides an application-level view of mobile devices traffic. Analyzing data gathered with this tool, they noticed that Web browsing contributes over 50% of the traffic, while email, media and maps contribute to roughly 10% each. They also discovered that most data transfers are small, making the radio controller waste power in its sleep-active-idle cycle. Also, this fact makes tools like Catnap [32] almost useless, since they aim to reduce power consumption during long transfers. In their study, Falaki et al. showed that reducing the *tail* time from 12 seconds to 4.5 allows to save 35% power in radio communications. Thus, having perfect knowledge of the incoming traffic permits to save up to 60% power, meaning that this methodology can be improved by a model considering usage patterns.

Finally, a Battery Lifetime Predictor is presented in [33]. The approach compared the device actual discharge with a measured base curve, obtained when the device was in idle, i.e. no applications were running. To compute the base curve, it is necessary a one-time offline measurement, that can be repeated periodically, to consider also battery aging phenomena. This method works only with a predefined set of measured applications and the prediction is not valid anymore, if a new application is added. CABMAN was prototyped for both Linux and Symbian OS, using as test devices an HP laptop with a new battery, a Dell laptop with a very old battery and a regular HP iPAQ PDA. The battery lifetime predictor has a percentage error rate varying between 1.2%, having a new battery, and 6.1%, when the battery is very old.

VI. Existing Techniques Open Issues

So far we have seen how the problem of building power consumption models for mobile devices has been tackled using a variety of different approaches. Here we present the issues arising when comparing the discussed methodologies.
In general, **offline** data gathering mechanisms give more precise results, but it is more static and consequently less prone to adaptation. As a consequence, it is rarely used when the goal is to give the users an estimation of their devices battery lifetime. Moreover, it lacks of generalization abilities, since the gathered data are specific for a given device. Conversely, it is possible to use an **online** data gathering approach to build power models *at runtime*. These models, despite being less precise, have better adaptation capabilities and are the most suited to cope with the fast growing market of smart devices. Among these, we can distinguish models that take into account the user interaction [17], [18] from those which do not [22], [21]. User-based models provide a better estimation of battery lifetimes, since they model the user behavior on the same device used.
However, the approach which models the user along with the device requires strong assumptions, first of all that the device is used by only one user. This fact is commonly true for smartphones, but it may not be the case for tablets (e.g., they can be used by a whole family or by a working team). Moreover, the idea of modeling the user implicitly makes the assumption that he/she will always use his/her device in the same way: this may not hold, for instance, if he/she goes into vacation or he/she changes his job or habits. Some of the methodologies analyzed try to cope with this making the continuously adapt the model over time, even if the user behavior is not changing. On the other hand, state-of-the-art component-based models are built using benchmarks applications: since these applications are built to stress one component at a time, the models built on top of them can rarely predict if any interaction arises among different components from everyday usage of the phone.

VII. Direction of Future Works

In order to overcome the limitations highlighted in Section VI, we believe that a new approach would be useful. Our opinion is that models relying on data gathered by external tools are unfeasible: given the growing number of smart devices currently available on the market, it is impossible to test and profile the discharge behavior of all of them. A solution to this problem is to use data coming from the device itself, even if this approach is slightly less precise. In order to give a precise estimation of mobile battery lifetime, we cannot disregard the user behavior while building models. However, user-centric models available nowadays are built considering only high-level actions, such as *Calling* or *Surfing the Web*, whose power consumption can vary depending on the state of the phone components (e.g., the signal strength has a huge impact on the power consumption while calling over the GSM network). We believe that a solid model has to be built for every phone component before taking into account the user impact on power consumption. This separation between user and device would allow to make comparisons among different devices and could lead to power benchmarking of devices still not available on the market. Then, a strong power profiling of phones components can be integrated into a fine user-profiling model, to give even better estimation capabilities.

VIII. Conclusions

In this survey, we reviewed the state-of-the-art of power and energy consumption modeling for mobile devices, highlighting the remarkable open issues. This topic is very crucial nowadays, since mobile devices derive the energy required for their operations from batteries with limited capacity. As a consequence, energy efficiency and optimal power management are crucial on those devices. Hence, the need to create models to describe their power consumption behavior.
As shown in previous sections, many techniques are available in literature. It is possible to group them into three big sets by basing on the approach they adopted: system level, application level and user level. In this survey, differently from previous approach [2], we also defined a set of feature able to characterize those techniques.
By basing on the previous analysis, it emerged that a run-time system level generated model is able to adapt itself to new software updates or even new devices, but it is less accurate because is does not have access to data coming from device real usage scenario. Moreover, it is useful to model the power consumption of a single application in execution on the system to perform fine-grained power consumption techniques and to profile the application consumption over time. Finally, it has been demonstrated that it is not possible to build efficient power consumption models without taking into consideration the device usage patterns: a knowledge on user's behaviors allows to build more precise battery lifetime estimators, since each device is tuned on specific habits. However, those techniques modeling usage patterns along with power consumption often make strong assumptions and have some limitations. In our opinion, a future power model should be portable, thus not relying in data coming from external measurements, but should also have a strong component profiling technique integrated with a fine grained one on the user profile.

References

[1] Cecilia Mascolo. The power of mobile computing in a social era. *IEEE Internet Computing*, 14(6):76–79, 2010.

[2] N. Vallina-Rodriguez and J. Crowcroft. Energy management techniques in modern mobile handsets. *Communications Surveys Tutorials, IEEE*, PP(99):1 –20, 2012.

[3] M. Maggio, H. Hoffmann, M.D. Santambrogio, A. Agarwal, and A. Leva. Power optimization in embedded systems via feedback control of resource allocation. *Control Systems Technology, IEEE Transactions on*, 21(1):239–246, Jan.

[4] Martina Maggio, Henry Hoffmann, Anant Agarwal, and Alberto Leva. Control-theoretical cpu allocation: Design and implementation with feedback control.

[5] Davide B Bartolini, Filippo Sironi, Martina Maggio, Riccardo Cattaneo, Donatella Sciuto, and Marco D Santambrogio. A framework for thermal and performance management.

[6] Jason Flinn and M. Satyanarayanan. Managing battery lifetime with energy-aware adaptation. *ACM Trans. Comput. Syst.*, 22(2):137–179, May 2004.

[7] Narseo Vallina-Rodriguez and Jon Crowcroft. Erdos: achieving energy savings in mobile os. In *Proceedings of the sixth international workshop on MobiArch*, MobiArch '11, pages 37–42, New York, NY, USA, 2011. ACM.

[8] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[9] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. 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*, CODES/ISSS '10, pages 105–114, New York, NY, USA, 2010. ACM.

[10] Alex Shye, Benjamin Scholbrock, and Gokhan Memik. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 168–178, New York, NY, USA, 2009. ACM.

[11] Manish Anand, Edmund B. Nightingale, and Jason Flinn. Ghosts in the machine: interfaces for better power management. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 23–35, New York, NY, USA, 2004. ACM.

[12] Mikkel Baun Kjærgaard and Henrik Blunck. Unsupervised Power Profiling for Mobile Devices. In *Proceedings of the 8th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous 2011)*. Springer, 2011.

[13] Mian Dong and Lin Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 335–348, New York, NY, USA, 2011. ACM.

[14] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 153–168, New York, NY, USA, 2011. ACM.

[15] Chris Thompson, Douglas C. Schmidt, Hamilton A. Turner, and Jules White. Analyzing mobile application software power consumption via model-driven engineering. In Csar Benavente-Peces and Joaquim Filipe, editors, *PECCS*, pages 101–113. SciTePress, 2011.

[16] Yu Xiao, Petri Savolainen, Arto Karppanen, Matti Siekkinen, and Antti Ylä-Jääski. Practical power modeling of data transmission over 802.11g for wireless applications. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 75–84, New York, NY, USA, 2010. ACM.

[17] Narseo Vallina-Rodriguez, Pan Hui, Jon Crowcroft, and Andrew Rice. Exhausting battery statistics: understanding the energy demands on mobile handsets. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, MobiHeld '10, pages 9–14, New York, NY, USA, 2010. ACM.

[18] Joon-Myung Kang, Sin seok Seo, and James Won-Ki Hong. Personalized battery lifetime prediction for mobile devices based on usage patterns. *Journal of Computing Science and Engineering*, 5(4):338–345, 2011.

[19] Andrew Colin Rice and Simon Hay. Decomposing power measurements for mobile devices. In *PerCom*, pages 70–78. IEEE Computer Society, 2010.

[20] Arjun Roy, Stephen M. Rumble, Ryan Stutsman, Philip Levis, David Mazières, and Nickolai Zeldovich. Energy management in mobile devices with the cinder operating system. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 139–152, New York, NY, USA, 2011. ACM.

[21] Yu Xiao, Rijubrata Bhaumik, Zhirong Yang, Matti Siekkinen, Petri Savolainen, and Antti Yla-Jaaski. A system-level model for runtime power estimation on mobile devices. In *Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 27–34, Washington, DC, USA, 2010. IEEE Computer Society.

[22] Xia Zhao, Yao Guo, Qing Feng, and Xiangqun Chen. A system context-aware approach for battery lifetime prediction in smart phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 641–646, New York, NY, USA, 2011. ACM.

[23] Ye Wen, Rich Wolski, and Chandra Krintz. Online prediction of battery lifetime for embedded and mobile devices. In *Proceedings of the Third international conference on Power - Aware Computer Systems*, PACS'03, pages 57–72, Berlin, Heidelberg, 2004. Springer-Verlag.

[24] Smart Battery System Implementers Forum. Smart battery data specification(v1.1). 1998.

[25] Selim Gurun and Chandra Krintz. 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*, CODES+ISSS '06, pages 28–33, New York, NY, USA, 2006. ACM.

[26] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 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*, IMC '09, pages 280–293, New York, NY, USA, 2009. ACM.

[27] Lei Liu, Guanhua Yan, Xinwen Zhang, and Songqing Chen. Virusmeter: Preventing your cellphone from spies. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 244–264, Berlin, Heidelberg, 2009. Springer-Verlag.

[28] Joon-Myung Kang, Sin seok Seo, and J.W.-K. Hong. Usage pattern analysis of smartphones. In *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*, pages 1 –8, sept. 2011.

[29] J. Lee, Hyunwoo Joe, and Hyungshin Kim. Smart phone power model generation using use pattern analysis. In *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, pages 412 –413, jan. 2012.

[30] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.

[31] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 281–287, New York, NY, USA, 2010. ACM.

[32] Fahad R. Dogar, Peter Steenkiste, and Konstantina Papagiannaki. Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys, pages 107–122, New York, NY, USA, 2010. ACM.

[33] Nishkam Ravi, James Scott, Lu Han, and Liviu Iftode. Context-aware battery management for mobile phones. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, PERCOM '08, pages 224–233, Washington, DC, USA, 2008. IEEE Computer Society.