Energy Management Techniques in Modern Mobile Handsets

Narseo Vallina-Rodriguez and Jon Crowcroft, *Fellow, IEEE*

***Abstract*—Managing energy efficiently is paramount in modern smartphones. The diverse range of wireless interfaces and sensors, and the increasing popularity of power-hungry applications that take advantage of these resources can reduce the battery life of mobile handhelds to few hours of operation. The research community, and operating system and hardware vendors found interesting optimisations and techniques to extend the battery life of mobile phones. However, the state of the art of lithium-ion batteries clearly indicates that energy efficiency must be achieved both at the hardware and software level. In this survey, we will cover the software solutions that can be found in the research literature between 1999 and May 2011 at six different levels: energy-aware operating systems, efficient resource management, the impact of users' interaction patterns with mobile devices and applications, wireless interfaces and sensors management, and finally the benefits of integrating mobile devices with cloud computing services.**

***Index Terms*—Operating systems Energy efficiency, Energy management, Human computer interaction, Sensors, Context awareness, 3G mobile communication, IEEE 80211 Standards.**

I. Introduction

TODAY'S mobile phones are equipped with a wide range of sensing, computational, storage and communication resources that bootstrapped the birth of rich mobile applications such as location-aware services and mobile social networks. However, those applications can potentially reduce the battery life of mobile handsets to few hours of operation.

Unfortunately, battery technologies have not experienced the same evolution as the rest of hardware components in mobile handsets. Most mobile phones are powered by lithium-ion batteries that can provide many times the energy of other types of batteries in the same fraction of space. However, the state of the art in battery technology shows that the only alternative left at the moment to extend the battery life of mobile phones is reducing the power consumption at the hardware level and designing more energy efficient applications and operating systems.

The research community, hardware manufacturers and OS designers already found positive solutions to extend the battery life of mobile handsets at different levels such as hardware, operating system, wireless technologies and applications. However, these efforts are limited by the heavy layering existing on smartphone platforms that makes difficult exploiting cross-layer optimisations, which might be fairly straightforward otherwise. One of the main reasons behind this limitation is a complex business ecosystem in which multiple players (e.g. cellular network providers, content providers, cloud service providers, hardware manufacturers and OS vendors) compete to retain their share of the mobile business.

In this survey we analyse and compare general solutions for energy efficiency on mobile devices at the software level on six major axes: from operating system solutions to energy savings via process migration to the cloud and protocol optimisations. Our main contribution is to classify and provide a short summary of the multiple efforts on studying, modeling and reducing energy consumption in mobile devices. The work we consider in this study are ones published between 1999 and May 2011. Future hardware and next generation low power wireless interfaces are excluded since they could be part of an independent survey themselves. We selected the solutions that can still be realistically implemented in current and future mobile handsets at the software level. Although including a quantitative comparison of the different solutions was one of our initial goals for the survey, mobile platform fragmentation issues and the diversity of evaluation methodologies made it unviable. Nevertheless, as far as we are concerned, this is the most recent survey about energy-efficient techniques on mobile operating systems since the one published in 1995 by Welch [1].

II. Survey structure

As we have mentioned in the introduction, power-efficiency in mobile systems can be achieved at different levels. Hence, the survey is structured following a taxonomy of the papers under study based on the type of optimisation they are proposing. This classification is as follows:

- *Energy aware operating systems* (Section III, Table I). The main question about energy efficiency in mobile devices is *Who should be responsible for energy management? Applications or operating system?*. Probably the right answer is both. At the operating system level, the main idea is to reduce energy consumption by unifying resource and energy management and by leveraging collaboration between applications and operating system. In fact, a key part of energy-efficient resources and energy management is having a good understanding of how resources are demanded by users and applications in the system. This section describes some attempts towards energy aware mobile operating systems, energy-efficient resource management and resource profilers.
- *Energy measurements and power models* (Section IV, Table II). Understanding how energy is being consumed by the hardware components is essential in order to design energy-aware systems. This section describes some

Manuscript received 14 March 2011; revised 9 February 2012.

N. Vallina-Rodriguez and J. Crowcroft are with the Computer Lab, University of Cambridge, 15 JJ Thomson Avenue, Cambridge, CB3 0FD, UK (e-mail: name.surname@cl.cam.ac.uk).

Digital Object Identifier 10.1109/SURV.2012.021312.00045

papers that measured the energy consumption and defined power models for modern smartphones.

- *Users' interaction with applications and computing resources* (Section V, Table III). Battery lifetime has become one of the top usability issues of mobile systems. Hence, improving battery lifetime is highly related to a better understanding of how users interact with their battery and their resources. Any energy-aware system must be able to know when, where and how the user drains the battery and when there will be future charging opportunities. This section comprises different papers trying to understand battery charging cycles and users' resource demands.
- *Wireless interfaces and protocol optimisations* (Section VI, Table IV). Wireless interfaces are major power consumers on mobile systems. There are multiple ways of making wireless interfaces more efficient at every layer of the protocol stack (also cross-layer optimisations) by taking advantage of the different power states. However, they usually require application, operating system and network infrastructure cooperation. As we have already mentioned in the introduction, discussing new wireless interfaces and link layer optimizations are not within the scope of this survey.
- *Sensors optimisations* (Section VII, Table V). Location-aware applications became one of the most popular services in mobile systems. A mobile device has sensors such as GPS, network-based positioning systems and accelerometer for location with different resolutions and power demands. As a result, there is a trade-off between energy-consumption and accuracy. This section discusses solutions to minimise the energy consumption of continuous sensing at the software level.
- *Computation off-loading* (Section VIII, Table VI). Cloud computing is opening new possibilities to mobile systems in many ways. Computation off-loading has been shown to be effective for extending the computational power and battery life of resource-restricted devices since the late 90s. In fact, even modern mobile operating systems rely more and more on online services running in the cloud. Remote execution allows migrating computation from battery-powered mobile devices to wall-powered, higher performance machines hosted somewhere on the Internet. However, there are factors such as network state that can clearly affect its performance. This section covers the most relevant works about computation off-loading in mobile devices from an energy perspective.

Finally, Appendix A contains all the energy measurements of smartphone components and power models that can be found in some of the papers included in this survey. As we have already mentioned in the introduction, it is not possible to make a quantitative comparison of the solutions along the different areas due to the diversity of mobile platforms and the diversity of evaluation methodologies used in the papers under study.

III. Energy aware operating systems

Most of the literature about energy-awareness on mobile devices takes the perspective of managing individual components efficiently rather than from an overall operating systems viewpoint. This section describes the motivations behind considering energy as a fundamental resource in mobile operating systems as they currently do with memory, I/O and CPU. We will describe works about mobile energy-aware operating systems, energy-aware scheduling mechanisms, non-intrusive resources profilers and the benefits of leveraging the collaboration between applications and operating systems to save energy. Note that resource-specific optimisation will be described in sections VI and VII.

A. The need of energy-awareness in mobile OS

The concept of an energy-aware operating system has been proposed in the late 90s with energy-aware operating systems for laptops like Odyssey [5] and ECOSystem [17] (the later work is within the umbrella of the Milly Watt project [18]). In 2000, Ellis pointed up that energy should be considered as a first-class resource in addition to the traditional OS perspective of maximizing performance [3]. Although this topic has been almost abandoned during the mid 2000s, it has regained researchers attention recently due to the energy limitations of current smartphones in which power-hungry applications (or even malware) can reduce the battery life of the handset to few hours of operation. This was the motivation behind mobile energy-aware operating systems for mobile handsets such as Cinder [9] and ErdOS [10].

There are two opposite propositions about how and by whom energy-aware policies in mobile devices should be performed. On the one hand, some authors suggest that applications must adapt dynamically to energy limitations as in Chameleon [19] but this approach lacks of a central entity responsible for monitor all the resources consumption caused by other applications. On the other hand, other researchers suggest that resources and energy management should be entirely done at the operating system. However, this solution can present scalability problems. Both Odyssey and ECOSystem present an intermediate solution. They follow a hybrid approach in which both applications and operating system collaborate to reduce the power consumption in a mobile phone. Ideally, the operating system must know applications' resource demands and the available energy resources until the next charging opportunity to reduce the power consumption while maximising user experience. However, new programming models, schedulers, energy measurement tools, resource profilers and power-based APIs must be developed in order to support software-level energy management.

In 1995, Noble *et al.* introduced Odyssey [5]. In general terms, it can be defined as a Linux-based energy-aware operating system in which applications can adapt in runtime the quality of service delivered to the users (e.g. degrading video bit-rate or *fidelity* to reduce bandwidth and energy requirements by the applications) based on the available energy and resources. The system design can be found in different papers as can be observed in Table I. Odyssey monitors resource demands and provides an API to applications in order to enable a bi-directional communication channel between them and the operating system to notify the availability and demand of resources. *Odyssey* also addresses applications'

TABLE I
TOWARDS AN ENERGY-AWARE OPERATING SYSTEM. HYBRID SOLUTIONS ARE THOSE THAT LEVERAGE THE INTERACTION BETWEEN APPLICATIONS AND OPERATING SYSTEM TO MANAGE RESOURCES MORE EFFICIENTLY AND THEREFORE, ACHIEVE ENERGY SAVINGS.

		Energy-aware operating systems
Cite	*Name*	*Description*
[2], [3], [4]	EcoSystem	Hybrid energy-aware operating system for mobile devices (mainly laptops) that relies on an energy-aware scheduller.
[5], [6], [7]	Odyssey	Hybrid Linux-based energy-aware operating system that adapts applications' QoS (quality of service) to energy demands.
[8], [9]	Cinder	Hybrid mobile operating system built on top of HiStar OS. It allows users and applications to control and manage limited device resources in a similar fashion to ECOSystem.
[10]	ErdOS	Centralised energy and context-aware mobile OS built as an extension of Android OS. It exploits proactive resource management techniques and it enables transparent opportunistic access to remote resources in nearby devices.
[11]	CondOS	Context-aware OS architecture to manage efficiently sensing resources.

		Resource profilers and resource management techniques
Cite	*Name*	*Description*
[12]	PowerScope	Hybrid resource profiler (part of Odyssey OS). It combines off-line and online techniques and maps energy consumption to program structure (procedure level).
[13]	Joule Watcher	Fine granularity and event-driven resource profiler at the thread-level.
[14]	STPM	Hybrid middleware solution for Linux to support adaptive power management based on the observed demand patterns from applications.
[15]	CIST	Hybrid application-aware middleware framework which adapts its behaviour to application's needs. Focused on wireless interfaces.
[16]	Koala	Centralised and proactive resource manager.

diversity and concurrency by taking advantage of applications' history and feedback. In other words, energy management is performed by estimating the future resource and energy demands in order to determine what fraction of the total energy is consumed during a certain period of time by each application [7]. To do so, Odyssey takes advantage of the online profiler PowerScope [12], which will be described later in this section, to build a power model of the system.

ECOSystem is another linux-based operating system that allocates energy to competing tasks, also taking into account the fairness between applications and the trade-off between performance and energy consumption. In ECOSystem, energy is fairly allocated to multiple hardware components and applications using the *currentcy*[1] unit abstraction as it is described in [4] and [2]. With this model, ECOSystem aims to achieve a battery lifetime goal using the discharging rate as an indication of energy consumption instead of using an energy model as Odyssey does. In order to do it, ECOSystem tracks applications' resource demands using an adaptation of *Resource Containers* [20]. ECOSystem has an energy-centric scheduler that selects the next process to be executed depending on the currentcy spent by those tasks relative to its specified share. All those features allow defining policies that selectively degrade applications' service level to preserve energy capacity for more important tasks. Nevertheless, there are some critics to this approach. Flinn and Satyanarayanan claim that ECOSystem degrades performance by de-scheduling applications to leverage non ideal battery characteristics while Odyssey leverages adaptivity [7].

Let's focus now on mobile energy-aware operating systems. *Cinder* [9] is a mobile OS designed on top of the HiStar exokernel that exploits device-level accounting and power modelling. It has been tested on HTC Dream devices but at the moment it requires building an offline energy model of the system as in Odyssey. Cinder tracks applications and services responsible for resource use even across interprocess communication calls serviced in other address spaces. It aims to provide effective energy allocation by providing isolation, subdivision and delegation to applications. The authors claim that this approach also allows detecting malware and buggy applications and it can easily limit their access to computing resources and energy.

Similarly to ECOSystem, Cinder allocates energy to applications using two abstractions called *reserve* and *taps* to form a graph of resource consumption. In Cinder, the system battery is represented in the resource graph as the root reserve. When an application consumes a resource, the Cinder kernel reduces the right values in the corresponding reserve and its scheduler only allows threads to run if they have enough reserves to run. The rate at which the reserves are being consumed is controlled by *taps* which are defined as special-purpose threads whose only role is to transfer energy between reserves (they support constant and proportional rates). Once an application has consumed all its reserves, the kernel prevents its threads to perform more actions. Nevertheless, Cinder allows *reserve debits* between tasks for performing additional actions.

ErdOS [10] proposes a different philosophy to save energy compared to the previous cases. ErdOS has been conceived as an extension of Android OS[2]. ErdOS work was motivated by the fact that resources' state (e.g. GPS and cellular networks) and the usage patterns and habits of mobile users are diverse and highly context-dependent. Moreover, the way users interact with their applications cause complex interdependencies between hardware and software components. As a consequence, managing computing resources to applications proactively based on predictions of the resources state and

[1]Energy currency used by EcoSystem to allocate energy to applications

[2]It can execute any existing Android application

the users' demands is more flexible and efficient than algorithmic resource management [21]. In order to support this feature, ErdOS proposes monitoring resources state, applications resources' demands and users' interaction patterns with applications and learning from users' behaviour and habits (defined as *users activities*) to predict future resource demands and availability in an event-based fashion. In addition, the authors consider that computation, sensing and networking should not be exclusively limited to local machine resources so that accessing others in neighbouring handsets could be beneficial both in terms of energy, quality of service and usability. To do so, ErdOS leverages low power wireless interfaces and existing social links between mobile user to enable opportunistic resource sharing in a secure and fair way.

CondOS is a context-aware OS that also has saving energy in mind [11]. In this positioning paper, the authors claim that instead of allowing applications to manage independently but inefficiently the available sensing resources, the operating system must be responsible for that. This consideration could provide different benefits such as better memory management, scheduling and security as well as in terms of energy savings. Finally, it is worth mentioning an interesting and totally decentralised energy allocation model that was proposed for the Nemesis OS [22], (an operating system designed for multimedia applications). In this case, the authors describe how to allocate energy using a micro-economy model. The system allocates energy to applications based on their cost and utility but it also considers how much congested the system resources are. Applications have a defined credit to use when accessing resources so they have a motivation to adapt their energy consumption.

B. Resource profilers and resource management techniques

Efficient power management in mobile platforms is a complex and challenging research problem due to the multitude of possible hardware configuration options and power states, and the interdependencies between computing, sensing and networking resources caused by applications. As Snowdon *et al.* claim in [16], power management in present mainstream operating systems tends to be simplistic and as a consequence, sub-optimal. Traditional operating systems run the workload to completion at the maximum performance setting and then they shift into a low-power mode (or to the lowest performance setting) to achieve energy savings. There are more efficient ways of managing resources from an energy-aware perspective. However, this requires an accurate knowledge of any task being performed inside the computer system (at the process or thread level) in order to be aware of the resource demands.

Hardware resource monitors have proved to offer valuable information in the field of performance analysis. This section shows how similar techniques are being applied to investigate the energy usage patterns of individual threads and processes. We will cover the different types of resource profilers (from thread-level to application one) and general resource management techniques (sections VI and VII will look at resource-specific management and optimisations). Similar systems can be found in following sections regarding remote execution in order to estimate the cost of executing a task locally or remotely.

Odyssey OS utilizes PowerScope [12] to map energy consumption to program structure in a similar fashion to CPU profilers such as *prof* command in Unix machines. In other words, it maps energy to program structure at the procedure level so it can help to identify applications behaving as energy sinks. PowerScope combines both online and off-line techniques to profile applications. It requires an external power meter and a second computer to reduce the energy overhead and any possible interference at the profile stage, using statistical sampling to collect traces. As it happens with energy models obtained using off-line methods, this approach is not scalable since it requires repeating the off-line training for every single hardware configuration and machine. Nevertheless, this kind of tool has two advantages: it allows developers to re-implement applications in order to meet the design goals and it also allows the operating system to manage resources more efficiently.

PowerScope estimates the energy use of applications by measuring the time spent in each state and by counting the number of state transitions. The authors claim that, in addition to mapping energy costs to specific processes, it is necessary to identify thread-level activities. For instance, a task which blocks frequently (e.g. sockets) may expend larger amounts of power on other resources such as the screen, disk, and network when the processor is idle. An evolution of PowerScope is described in [23]. In this case, the authors introduce a predictive system to manage resources proactively in order to support multi-fidelity computation. The resources monitoring tool takes advantage of simple machine learning techniques so the resources manager can process larger number of adaptions with much more accuracy in a single step.

Joule Watcher [13] is a fine-grained and event-driven energy profiler that also accounts the energy consumption at the thread-level. In this case, they use counters embedded in the target hardware drivers to register events that imply the consumption of energy by monitoring CPU, battery and memory. Those parameters are accessed in the thread and from the system context so it makes possible estimating the energy consumption of individual threads and the whole system. Obviously, because of the limited number of resources it monitors, its real integration on a modern energy-aware OS can be compromised.

Regarding resource management, earlier works described that energy-efficiency must be performed on the application side. Nevertheless, applications are usually responsible for performing transformations to increase the possible energy savings while the management policies are implemented in the operating system. Heath *et al.*described how application transformations can increase device idle times by informing the OS about the length of an upcoming period of idleness [24].

Self-tuning power manager (STPM) [14] is an energy-aware resource management middleware framework focused on I/O devices (i.e. wireless interfaces) which leverages collaboration between applications and system and also provides an energy-aware caching system. The idea is that applications disclose *ghost hints* about their intentions of using any of the I/O devices while the system exposes context information about the general state of the system to applications. With this knowledge, STPM adapts the power management strategy to the observed pattern from the applications. The author claims that STPM makes better power management decisions because it tries to achieve the best combination of performance and energy conservation given a specific application workload. In theory, this approach does not limit which requests are serviced by hardware devices to applications as it might happen in ECOSystem [2].

Likewise, CIST [15] shows that it is possible to achieve higher energy savings by considering the needs of the applications rather than analyzing the behaviour of resource. It does not require learning from applications as STPM does [14]. CIST exploits Dynamic Power Management (DPM) techniques [25] to reduce power consumption at runtime and it takes advantage of idle times from applications to selectively turn off hardware resources when idle. It takes into consideration the overhead caused by the implicit energy handoff described in [26]. The system evaluation was done with the wireless card in Windows Mobile devices.

Koala [16] is a recent pro-active approach that allocates resources based on the prediction of the performance and energy consumption of each piece of software by collecting fine-grained statistics about them. At each scheduling event, the application behaviour is matched against the system policy to find the most appropriate operating condition to achieve an energy goal. They used an arbitrary policy to demonstrate that it is possible to dynamically trade performance and energy demands with an *energy-delay* policy. Such a policy provides a single parameter for tuning the system to achieve energy-savings. Finally, *Llama* [27] is another adaptive resource management technique that estimates (for a particular user and device) how much battery is likely to remain unused. Based on this prediction, it adjusts the quality of service of the applications in order to achieve the forecasted energy requirements. However, the authors have not properly evaluated the potential energy savings that the system can achieve and they were more interested in how Llama modifies the charging patterns of the few participants in their experiment.

IV. Energy measurements and models

Designers of modern smartphone hardware and vendors have incorporated power-saving features to allow hardware components to dynamically adjust their power consumption based on required functionality and performance. Many of these features are available for software developers; however, making an efficient use of them requires software developers to have a good understanding of the implications of their design decisions in terms of energy. In fact, mobile phones present significant differences in terms of power consumption signatures depending on the manufacturer, operating system and other contextual factors such as network coverage. This section introduces several analysis of the power consumption in modern smartphones. Many of these models and measurements can be found in the tables available in the Appendix.

One of the most detailed and general analysis of energy consumption in mobile handsets is the study completed by Carroll and Heiser in [28]. The authors present a detailed break-down of the power consumption of the Openmoko Neo Freerunner phone (2.5G) using an external high-resolution power meter. The authors analysed this device because its circuit schematics are freely available as opposed to other closed platforms such as iPhone, Windows Mobile, Android and Symbian. As a result, this platform allows the authors to break down the overall energy consumption per hardware component accurately at different power states.

Their results show that the most energy intense component are the GSM module (700mW at full capacity and 800mW when performing a phone call approximately) and the display (400mW including LCD panel, touch-screen, backlight and graphics accelerator). They also demonstrated that the content displayed in the screen can affect the energy consumption in the LCD panel: 33.1mW with white screen and 74.2mW for a black screen. The authors validated the overall energy consumption of a HTC Dream and Google Nexus One with the ones they obtained with the Openmoko Neo Freerunner. The last part of the paper is a coarse-grained estimation of the potential energy consumption of different usage patterns. They modelled five usage profiles (the paper does not include any justification of the values used to model each profile) and they simulated the energy consumed by these usage profiles per day. They estimated that the total battery life varies by almost a factor of 2.5 between the two extreme use cases: the business user and a phone permanently at the suspend state.

Other studies looked at the energy consumption of specific hardware components. Fitzek *et al*. analysed the energy impact of 2G and 3G network usage for Nokia N95's [29]. Specifically, they analysed the energy consumption of three common services like text messaging, voice and data using an application called *Nokia Energy Profiler*[3] and an external power meter for correctness. Their experimental results report a larger energy consumption in 3G networks for text messaging (SMS) and voice services compared to 2G networks. The energy consumption of sending text messages increases linearly with the length of the message while the signal strength clearly affects the time required to transmit the message in both types of networks. Interestingly, this parameter is not considered in some of the power models that will be described later. In the case of voice services, using GSM requires around 46% less energy than UMTS networks. However, 3G+ technologies become more energy efficient to transmit large volumes of data.

The work by Balasubramanian *et al*. in [30] goes a bit deeper in the analysis of IEEE 802.11 standards and cellular networks (using exclusively Nokia Energy Profiler as measurement tool). They found that cellular networks present a high *tail energy* overhead by staying in high energy-states after completing a transfer. This effect is much lower in GSM than in 3G networks. On the other hand, IEEE 802.11 networks do not present any tail energy and they are more efficient than cellular networks. However, they have an energy overhead caused by associating to the access point procedures. The authors modelled the energy consumption required by the

[3]Nokia Energy Profiler is a proprietary application available for some Nokia devices. Its maximum resolution is 4Hz. It is a popular energy measurement tool for many energy papers in Symbian handsets. It allows measuring energy consumption on the go.

TABLE II
COMPARISON OF THE DIFFERENT ENERGY MEASUREMENTS AND POWER MODELS. THE TABLE HIGHLIGHTS THE MOBILE PLATFORM, WHETHER OR NOT THE POWER MEASUREMENTS WERE DONE WITH AN EXTERNAL MULTIMETER, AND THE RESOURCES UNDER STUDY. RESOURCES SUCH AS CAMERA, AUDIO SUPPORT, SD CARD AND ACCELEROMETER ARE NOT CONSIDERED IN THIS TABLE BECAUSE OF SPACE LIMITATIONS. (SEE APPENDIX FOR DETAILS)

Energy measurements and power models

			Resources analysed							
Cite	*Platform*	*Powermeter*	*CPU*	*Display*	*GPS*	*Bluetooth*	*WiFi*	*GSM*	*3G*	*Description*
[28]	Openmoko	✓	✓	✓	✓	✓	✓	✓		Power measurements.
[29]	Symbian	✓						✓	✓	Energy impact of 2G and 3G cellular networks.
[30]	Symbian						✓	✓	✓	Energy costs of wireless interfaces. Impact of *tail energy*.
[31]	Android	✓					✓			High resolution analysis of 802.11 interfaces.
[32]	Symbian		✓				✓			Energy model for data transmissions on WiFi as a function of the traffic burstiness.
[33]	Android	✓	✓	✓	✓		✓	✓		PowerTutor: online Power Model based on the voltage curve and linear regression techniques to infer power consumption at each different power state.
[34]	Android		✓	✓			✓	✓		Power Model for Android using application benchmarks.
[35]	Symbian		✓	✓					✓	Power Model using linear regression.

wireless interfaces in the devices they studied. Those findings were used to implement a protocol called *TailEnder* which is described in section VI.

Another interesting power model for wireless interfaces in Symbian devices has been done by Xiao *et al.* in [32]. In this case, the authors aim to model the energy impact of data transmission over IEEE 802.11g as a function of the traffic burstiness and an off-line measurement of the power consumed by the devices at a specific power state. Their model, validated using both an external multimeter and Nokia Energy Profiler, can be used to estimate the energy consumption of IEEE 802.11g interfaces in runtime but it is not clear the power overhead that this technique will have in the system due to the computation requirements.

The work by Rice and Hay [31] is probably the more accurate energy measurement of WiFi interfaces in smartphones. In this paper, the authors present a platform to run automatic measurements in mobile phones using high-resolution power meters. Their platform synchronises the device and the measurement tool which is sampling at 250KHz with minimal error; using short screen pulses for synchronisation. The paper also incorporates a detailed analysis of the cost of sending messages over a IEEE 802.11 links. Their results reveal that the energy cost per KB transmitted varies with the buffer size, and interesting effects during transmissions and idle power states.

The rest of the papers described in this section attempt to create a general system-level power model. Xiao *et al.* [35] consider the processors, wireless LAN interface and display in Symbian devices. Their model uses linear regression with nonnegative coefficients and the Nokia Energy Profiler to know the total energy consumption in the handset. In the case of Android devices, PowerTutor[4] uses information about the discharging rate of the voltage curve to estimate the power consumption [33]. Despite that it is probably the most complete model, it does not consider resources like accelerometer and camera, and it does not take into account the impact of signal strength and burstiness on wireless interfaces[5]. In order to obtain the power model, PowerTutor uses linear regression to compute the coefficients about the energy consumption of each individual resource by combining all the hardware power modes. In theory, this model will not require using an external multimeter to measure the power consumption and it enables online estimation of the power consumption looking at the power state and the resources usage in the handset. However, one of its limitations is that it requires a quite expensive computational training to obtain the model and it does not present an evaluation of the overhead caused by estimating the power consumption in runtime and how frequently this action is done.

A different approach compared to PowerTutor is the one suggested by Shye *et al.* [34] (more detailed in section V-B). This solution uses a background logger that samples resources utilisation at 1Hz to estimate the power consumption of mobile devices during normal users activity. As the previous models, this model has been derived by linear regression techniques using a power meter. However, they used application benchmarks rather than power states to derive the model. As a result, its measurements can be inaccurate because of relying exclusively on applications. It has been validated for HTC G1 devices and it only considers EDGE as possible cellular interfaces.

V. USERS' INTERACTION WITH APPLICATIONS AND COMPUTING RESOURCES

An important aspect of energy management is having a good understanding of how, when and where users interact with their handsets and how they demand energy. This problem can be divided in two as it is shown in Table III. The first one is about understanding the charging-discharging cycle

[4]The application is available in the Android Market. In their website, the authors claim that it has been validated for HTC G1, HTC Dream and Nexus One devices

[5]The authors mention in the paper that is currently under investigation

of the users. Both users and the operating system must be able to detect future power limitations to prioritise amongst various tasks that the device can perform. However, in order to efficiently prioritise those tasks without affecting the user experience, we need to know how users interacts with their handsets (and particular resources), and therefore how they demand energy.

A. Battery interaction and charging cycles

Ravi *et al.* propose a system for context-aware battery management that warns the user when it detects a power limitation before the next charging opportunity [36]. They use the current set of applications running, the discharging rate (arguably an inaccurate indicator) and phone call logs as inputs of their forecasting algorithms. Their motivation is guaranteeing that crucial applications (e.g. phone, messaging) should not be compromised by non-crucial ones. The evaluation of their algorithm was simulated with the *Reality Mining Project* [46] traces. Their results indicate that their algorithm can predict battery consumption and charging opportunities for users without a high usage entropy. The authors stress the difficulty to predict phone calls because of the impact of social factors and the high variability of calling patterns between weekends and weekdays.

Similarly, Oliver [37] uses classification methods to identify three distinct types of charging patterns among 17,300 blackberry users. Those clusters are defined as opportunistic chargers, light-consumers and nigh-time chargers. The logs were created using a light and event-based background application that monitors charging activity, battery level and system shutdowns. In a following technical report [38], they used clustering and classification algorithms to predict the remaining energy level in mobile handsets. They implemented a toolkit called EET designed to predict the successful execution rate of energy intensive applications. This tool allows developers to evaluate the energy consumption requirements of their applications against real user energy traces. EET tries to determine the subset of high-level energy characteristics that best differentiate users. Their results show that it is possible to predict the energy level on a mobile handset within 7% error within an hour and within 28% error within 24 hours.

Rahmati *et al.* [39] analysed how mobile users interact with their handsets from a humancomputer interaction (HCI) perspective. Their results are obtained from direct surveys to mobile users and from a background process used to monitor ten mobile users. They found qualitative and quantitative evidences about the problems caused by energy user-interfaces which can cause under-utilised power-saving settings, under-utilised battery energy and, consequently, dissatisfied users. A similar procedure has been followed by Banerjee *et al.* in [27]. They monitored and interviewed 10 mobile users between 42 to 77 days. In this case, the authors claim that, despite the fact that there is a great variation among users, most of recharges happen when the battery has substantial energy left and a considerable portion of the recharges are driven by context (location and time). That information was used to implement *Llama*, an adaptive resource management already described in section III.

To conclude this section, it is necessary to mention some of the previous works on predicting the remaining battery capacity. In [47], the authors introduce a history-based and statistical technique for online battery lifetime prediction based on the battery voltage curve while [48], [49] and [50] describe four different stochastic battery models to estimate the battery life in embedded systems.

B. Users' interaction with mobile applications and resources

The works in this section are highly related with the previous ones covering resources allocation and power models. As we have already remarked, an important requirement for effective and efficient energy management in mobile devices is a good understanding of where and how energy is used and how much of the system's energy is consumed by each part of the system. System architects should consider users' interaction patterns with the devices to clearly understand the impact of any optimisation on user experience. However, monitoring resources usage is still a challenging task due to the high diversity of applications and resources in moderns smartphones and the need to design non-intrusive and privacy-aware monitoring tools.

Some studies are based on traces collected directly from the network [51]. The main advantages of that kind of study are that they do not have any impact on the mobile device and they can gather a larger number of traces. However, unlike background loggers running on the device, they are limited about the information they can collect. For instance, Trestian *et al.* performed a large-scale network-based study of mobile usage to characterize the relationship between users, applications (based on URL requests) and their mobility patterns [40]. Their results demonstrate that applications usage is highly correlated with users' habits and location.

The largest-scale experiment performed from the end users' perspective was done using traces collected from a cross-platform application that checks the state of cellular networks and the performance of network-based applications called *3GTest* [41]. Despite the fact that this is not a completely energy-oriented paper, the traces from 30,000 mobile users all over the world can be considered representative. They provide rich information to understand the performance of cellular networks before designing energy-aware systems. With the traces obtained, the authors identified bottlenecks in the wireless network and also performance limitations and bugs in operating system and popular network-centric applications. In their results, the authors mention that network properties can vary depending on the time-of-day and location for a specific operator. Similar results were found by Tan *et al.* [52] in a shorter and more geographically limited scenario.

In [42] the authors analysed the mobile traffic patterns of 43 mobile users across 2 different mobile platforms using packet sniffers. Their results indicate that the amount of daily traffic generated by a user can vary from 2 to 500MB (browsing contributes over half of the total amount). They also looked at the inefficiencies caused by the generally small transfer size of the packets (median size of 3KB). This fact incurs a huge overhead in the lower layer protocols that varies from 12% to 40% when using transport security. Moreover, the uplink and

TABLE III
STUDIES ABOUT USERS' INTERACTION WITH BATTERIES, APPLICATIONS AND COMPUTING RESOURCES

Battery interaction and charging cycles

Cite	*Platform*	*Description*
[36]	Linux and Symbian	Battery life and charging opportunities predictor using contextual information and machine learning algorithms.
[37] [38]	Blackberry	Charging cycle analysis. Energy level prediction using clustering and classification algorithms
[39]	Windows Mobile 5	Surveys with mobile users and battery monitoring to understand charging patterns.
[27]	Windows Mobile 5	Battery monitoring and power manager system "Llama".

Users' interaction with mobile applications and resources

Cite	*Platform*	*Participants*	*Length*	*Description*
[40]	n/a	281.394	1 week	Online-applications usage and mobility patterns from network traces.
[41]	iPhone, Android, WiMo	30.000	n/a	Cellular networks analysis from an end-user perspective.
[42]	Android and WiMo	43	26-147 d.	Analysis of smartphone traffic in 2 platforms.
[43]	iPhone	25	10 weeks	Monitoring tool for iPhone handsets. Privacy and design considerations with a brief usage analysis.
[34]	Android	20	12 days	Resources profiler and energy model for Android G1 devices. Break down of energy consumption in mobile handsets per component in real scenarios.
[44]	Android and WiMo	33/222	7-28 weeks	Resources usage diversity of users from two different platforms. Temporal impact.
[21]	Android	18	1-2 weeks	Impact of temporal and spatial context and users' habits in resource demands and energy consumption. Resources interdependencies caused by users' interaction patterns.
[45]	WiMo	4-16	1-2 weeks	Resources profiler in mobile handsets which incorporates users' feedback.

downlink retransmission rates are 3.7% and 3.3% respectively. Consequently, 25% of the TCP sessions in cellular networks can experience a retransmission. They emphasize that while packet loss is the main factor that limits the throughput of smartphone traffic, larger buffers at Internet servers can improve the throughput of a quarter of the transfers. Moreover, a good knowledge of the traffic patterns in smartphones can enable new radio management policies that can save up to 35% of energy by reducing the idle time with minimal impact on the system performance and usability as we will see in section VI.

There are other works that monitor mobile users from a more general perspective. An example is *LiveLab*, a re-programable and event-based resources logger for jail-broken iPhone devices [43]. The authors describe a non-intrusive methodology to measure real-world smartphone usage and wireless networks taking into account users' privacy. They ran surveys to understand the privacy concerns of the users which revealed that they are happy to be monitored if their identities are not associated to the data collected. This paper seems to be an extension of a previous work by Rahmati and Zhong [53] which tried to understand how users interact with non-voice applications over a 4 months period. Their results also indicate that users' location affect the applications they use. However, the results presented in this paper are not representative since experiment subjects did not have cellular data plans so they mainly interacted with their online applications in locations with open wireless access points.

In LiveLab, the authors describe how a logger must be designed to not incur additional energy and computational costs in the handset. They consider that a non-intrusive logger must be event-based to avoid periodical polling by taking into account the state of the resources, it must take advantage of information already available in the handset and it should leverage system wake-ups to reduce additional overhead for data collection. However, the paper does not mention the resolution of the data logger and the energy cost of this technique. The paper also contains a short study of 25 iPhone smartphone users. The results show that users' interaction with the device and the state of the resources depend on contextual information such as time and space. Moreover, usage patterns also evolve in time. However, the analysis of the human interaction is very simple and seems to be focused exclusively on only two users.

Shye *et al.* [34] developed a background logger that periodically monitors resource utilisation at 1Hz during normal usage. In order to estimate how much energy accounts to each specific application and resource, they used an energy model built using linear regression techniques (already described in section IV). Their results show that energy consumption depends on the way each individual user interacts with the device. The authors found a large variation in the power breakdown between users and they claim that the screen and the CPU are the two largest power consuming components while the idle state accounts for 49.3% of the total system power when averaging across all the users. However, those numbers can be biased by their own logger and its high sampling frequency[6]. This paper also suggests optimisations to increase the battery life of mobile devices. For instance, the authors describe a technique to reduce the energy consumption of the long screen usage intervals by slowly reducing the screen

[6]The overhead caused by the background logger in the system resources is not included in the paper and the CPU cost can be associated with it.

brightness and the CPU frequency over time taking advantage of a phenomena called *change blindness* (this concept refers to the inability for humans to detect large changes in their environment). This approach is similar to the one suggested by Falaki *et al.* for Symbian devices screen in [54] and the one described by Brakmo *et al.* to set the CPU in sleeping mode during small periods of time in which the CPU load is low (e.g. situations like when the user is reading a document or looking at a website) [55].

A second work by Falaki *et al.* also looked at the diversity of usage patterns among 33 Android and 222 Windows Mobile users[7] [44]. They collected two different dataset in order to see the differences between screen state, applications, traffic patterns and battery level across all those users. In the case of Windows Mobile dataset, it only has data about which application was running and the screen state. Consequently, the dataset does not contain all the possible interactions with the device and diversity of mobile users. The authors aim to characterise intentional users activities and how they can impact network and energy usage. They also highlight that the results suggest that a proactive system-centric resource management customised to each user can achieve important energy savings due to the wide spectrum of usage patterns.

Vallina-Rodriguez *et al.* performed a study using a background application to collect traces from 18 mature Android users during 2 weeks [21]. The dataset contains contextual information and more than 25 state and usage statistics from multiple resources and applications, sampled every 10 seconds[8]. This study notably differs from previous works because it tries to understand the existing relationships between resources caused by the interaction patterns of the users.

The authors take advantage of machine learning techniques to better understand these dependencies and to see how contextual information and users' habits cause resource demands. The paper demonstrates that it is possible to predict how users will interact with their mobile devices using location and temporal information. Consequently, the authors claim that based on the experimental results, an algorithmic resource management is not feasible for mobile devices and it is necessary to find more flexible ways of managing resources which can adapt to users' interaction patterns. They suggest that an efficient resource management in mobile systems must be proactive, system-centric and user-centric by leveraging contextual and fine-grained information from the system.

MyExperience [45] is a system for capturing both objective and subjective data directly from Windows Mobile handsets. In the paper, it is not clear what kind of traces they are collecting and how detailed their logs are. The work describes design considerations and the system architecture as well as a performance evaluation of its footprint in the CPU, memory and the battery life. The authors estimated that their logger can reduce the battery life of the handset around a 12% despite using context triggers to sample the information. What really differentiates this work from others is that it requests users' feedback to understand the perceptual experience of the user after accessing an specific service. The paper also presents a brief study of the battery interaction patterns, mobility and text messages from 14 users during 1-2 weeks[9]. Interestingly, the charging pattern results obtained in this paper are similar to the ones that can be found in [27].

[7]The windows mobile traces were not collected by the authors and were part of a previous experiment.

[8]The subjects noticed a slight reduction of their battery life and the highest power consumption caused by the monitoring tool is around 300mW

VI. Wireless interfaces and protocol optimisations

Battery drain is a general problem in any portable wireless device. In the case of cellular networks, new technologies such as LTE aim to reduce the energy cost per transmitted bit compared to previous standards [56]. Other studies are focused on solutions that extend the battery life of mobile devices by improving the handover mechanisms [57] and on local wireless connectivities. There have been great improvements such as Bluetooth Low Energy (BLE, previously known as Wibree) [58], low-power WiFi techniques, 6LowPAN on top of IEEE 802.15.4 standards [59] and new technologies such as Qualcomm's FlashLink [60].

Moreover, it is possible to achieve energy savings at each layer of the protocol stack. A recent survey by Tsao and Huang [61] details the efforts to implement energy-efficient MAC protocols in IEEE 802.11 standards while the survey by Jones *et al.* collects all the studies performed before 2001 [62]. In fact, it is important to highlight that optimisations for IEEE 802.11 standards are not usually applicable for cellular networks because of their different power modes and features. Nevertheless, this survey is mainly focused on solutions for energy-efficient management of IEEE 802.11 interfaces that can be implemented at the software level in current mobile phones. Techniques involving protocols below the network layer are excluded.

Generally, there is an energy-usability trade-off when managing networking resources in mobile systems. As an example, cellular interfaces present three power modes: DCH (Dedicated Channel) FACH (Forward Access Channel), an intermediate power mode, and IDLE. In the case of WCDMA technologies, a large fraction of energy is wasted in these intermediate but still high-power states after the completion of a typical transfer in case there is going to be an immediate transmission once the current one is finished in order to improve the user experience in cellular networks. Note that in GSM technologies, the inactivity timer in the FACH state is much smaller compared to 3G (6 vs. 12 secs). As a consequence, as it was already described in previous sections, the work by Balasubramanian *et al.* [30] shows how cellular networks present a high *tail energy* overhead caused by the FACH power state. *TailEnder* is a protocol designed to save energy in mobile handsets by scheduling data transfers using prefetching and caching for applications which can tolerate it so it can minimise the tail energy caused by the inactivity timer [63].

On the other hand, IEEE 802.11 networks are more efficient than cellular networks for transmitting data but they present a higher overhead when the device is associating to the access point and a higher power consumption in idle mode. Because of this reason, most of the works described in this section try

[9]The authors clearly mention that their results are not representative

to leverage low power interfaces or contextual information to smartly wake up the WiFi interface from sleep mode when there is likely to have an access point instead of being permanently in a higher power state (or even idle) as we can see in table IV.

A common solution for IEEE 802.11 networks (i.e. WiFi) is based on exploiting traffic shaping techniques. There are periods of time during transport-layer sessions in which it is possible to set the wireless interface in low power modes. For instance, Bertozzi explores transport protocol optimisations for reducing the energy consumption of IEEE 802.11 standards with minimum performance overhead [64]. This paper describes how it is possible to monitor the run time parameters in the transport protocol (such as the receive buffer utilisation) to identify idle periods with a much higher granularity than the RTT values. Similarly, Krashinsky and Balakrishnan [65] propose setting the wireless interface to sleep during TCP slow start phase by designing an alternative adaptive algorithm to the IEEE 802.11 power saving mode (PSM).

Two recent publications regarding energy efficient management of the power states of wireless networks are Catnap [66] and Micro Power Management [67]. Both papers exploit the fact that wired networks are usually the bottleneck while IEEE 802.11 standards support higher data rates. Catnap reduces the energy consumption by allowing the wireless interface to sleep during data transfers. This is achieved by shaping the traffic to combine small gaps between packets into bigger sleep intervals, even during data transfers. However, this solution relies on the collaboration of a middle-box which decouples the wired segment from the wireless one. The system evaluation was performed under realistic network conditions. The results reveal that this technique can set the wireless interface to sleep for almost 40% of the time for a 10MB transfer. Nevertheless, its performance might be compromised under common mobile traffic because of the small transfer size of mobile traffic [42].

Likewise, Micro Power Management [67] takes advantage of short time intervals during wireless transmissions (in the order of 100ms) in which it is possible to set the NIC interface into idle state. This paper differs from Catnap [66] because it does not require cooperation from a middlebox and it simply adapts frame delay to demanded network throughput with minimal cooperation from the access point. In the case of incoming data, it exploits IEEE 802.11 retransmission mechanism to wake up the device. Their results show that it is possible to achieve more than 30% power reduction for the wireless transceiver for various applications without affecting the user experience.

There are also indications about the fact that energy efficiency of wireless networks can be affected by the traffic pattern. Tan *et al.* describe a new client-centric protocol called PSM-throttling[10] in [68]. The authors mention that PSM mode usually requires infrastructure support and it can degrade the transmission throughput thus increasing transmission delay. Consequently, that kind of solution is not suitable for QoS applications. The authors describe how it is possible to reshape the TCP traffic into periodic bursts with the same average throughput as the server transmission rate so the client can accurately predict the arriving time of packets and set the wireless interface in a lower power mode accordingly. Other works such as [77], [78] and [79] also analyse the energy savings that can be achieved by traffic shaping techniques.

[10]PSM stands for Power Saving Mode

Compression can also be used for reducing energy consumption. It has been estimated that sending a bit on a wireless link requires over 1000 times more energy than a single 32-bit computation. Several research projects looked at the benefits of compressing text data (or degrading the quality in case of video and audio) on a wireless transmission despite the implicit computational and memory costs. Barr and Asanovic [69] indicate that it is possible to save up to 57% of energy by compressing text data before transmitting it on a WiFi link. However, those techniques require tools to estimate in runtime when compression will cause energy savings or not in order to decide when it must be used. Previously, both Housel and Lindquist [80], and Krashinsky [81] demonstrated the benefits of using HTTP compression in terms of energy savings and bandwidth requirements without adding any additional delay in the transmission. In fact, compression can reduce the parsing time of XML and JSON data when it is transmitted over a bandwidth-constrained link. Most of modern web browsers (including mobile ones) support GZIP because of this reason.

A third type of work takes advantage of the complementary strengths of the several wireless interfaces supported in mobile phones and tries to combine them efficiently. In [70], Agarwal *et al.* describe an energy management architecture called *Cell2Notify* which uses the cellular radio on a smartphone to wake-up the wireless LAN interface upon an incoming VoIP to redirect the call through the WiFi interface. This system leverages the better quality and energy-efficiency of WiFi for data transmissions while minimising its expensive scanning cost. Their simulations results indicate that it is possible to save around 70-80% of energy compared to the single radio case. Shih *et al.* had previously described in [71] a similar solution.

Agarwal *et al.* [72] and the authors of *Blue-Fi* [73] describe how to save energy by taking advantage of Bluetooth radios and contextual information to serve as a paging channel for IEEE 802.11b. The results (obtained with a real system deployment with iPAQ PDA's equipped with Bluetooth radios and Cisco Aironet wireless networking cards) show that it is possible to save between 23% to 48% of energy compared to the present IEEE 802.11b standard operating modes with negligible impact on performance. The system also predicts when there will be Wi-Fi connectivity by combining contextual information obtained from Bluetooth scans contact-patterns and from cell-tower information. It allows the system to switch wireless interface on and off depending on their availability. A slightly different approach is the one leveraged by *ZiFi* [74]. This system provides a similar functionality to the previous system but it takes advantage of the interference signature generated by the WiFi beacons on the ZigBee scans. The authors did not evaluate the potential energy savings that they can achieve with that system.

Context-for-wireless is a context-aware intelligent switching algorithm between WiFi and cellular networks to reduce the energy consumption substantially [75]. The authors propose

TABLE IV
WIRELESS INTERFACES AND PROTOCOL OPTIMISATIONS. MOST OF THE STUDIES FOR WIRELESS INTERFACE RESOURCE MANAGEMENT ARE TARGETING IEEE 802.11 STANDARDS. THEY EXPLOIT OTHER WIRELESS INTERFACES AND CONTEXTUAL INFORMATION TO IDENTIFY WHEN A WIRELESS ACCESS POINT WILL BE AVAILABLE. THIS ALLOWS TURNING ON OR OFF THE WIRELESS INTERFACE WHEN NECESSARY AND, CONSEQUENTLY, REDUCING THE ENERGY SPENT WHILE IN HIGH POWER MODES SUCH AS IDLE MODE.

Wireless interfaces and protocol optimisations

		Interfaces Used			
Cite	*Name*	*WiFi*	*Cellular*	*Others*	*Description*
[30]	TailEnder	✓	✓		Effect of Tail energy on 3G networks. Prefetching and traffic shapping techniques.
[64]	n/a	✓			Identifies idle periods in a TCP session to set the wireless interface in a lower power mode.
[65]	n/a	✓			Sets the wireless interface to sleep during TCP slow start phase.
[66]	Catnap	✓			Allows the device to sleep during data transfers with the collaboration of a middle-box.
[67]	MPM	✓			Traffic shapping technique to exploit short idle times on wireless transmissions. (Simulation).
[68]	PSM-Throttling	✓			Traffic shape techniques to generate periodic bursts to easily predict the arriving time of packets with QoS support.
[69]	n/a	✓	✓		Data compression on wireless transmissions.
[70]	Cell2Notify	✓	✓		Exploits Cellular Networks to wake up WiFi interfaces for VoIP applications.
[71]	WakeOnWireless	✓	✓		Exploits Cellular Networks to wake up WiFi interfaces in VoIP applications.
[72]	n/a	✓		✓	Exploits Bluetooth to wake up WiFi interfaces in order to reduce power consumption of WiFi in idle mode. (Windows Mobile Prototype)
[73]	Blue-Fi	✓	✓	✓	WiFi coverage prediction using cellular networks and Bluetooth scans to reduce the power consumption of WiFi in idle mode.
[74]	ZiFi	✓		✓	Exploits ZigBee to activate WiFi interfaces depending on the interference caused by WiFi beacons on the ZigBee discoveries.
[75]	Context-for-wireless	✓	✓	✓	Leverages contextual information, history, network conditions and users' mobility to enable an energy-efficient management of WiFi and cellular networks.
[29]	n/a		✓		Smart energy-efficient usage of cellular network standards (2G and 3G) depending on the requirements of the application.
[76]	SALSA	✓	✓		Decides which interface to use based on the energy-delay trade-offs of the available interfaces.

leveraging contextual information such as time, history, cellular network conditions and mobility to formulate the selection of wireless interfaces as a statistical decision problem and to predict future network conditions. As a result, they do not need to power up the WiFi interface so often. Their results over simulations show that *Context-for-wireless* can double the battery life of mobile handsets. Nevertheless, the authors also mention that both the analysis and the field validation they have performed can be compromised by the small number of participants in their experiment.

In the case of cellular networks management, the energy results from [29] described in section IV lead the authors to suggest that the operating system must be able to switch the type of network depending on which service is being requested by the user and the applications. Their simulation results show that it is possible to obtain important energy savings despite the energy and time cost of making handoffs from GSM to UMTS and vice versa. However, this study does not analyse how the transitions between networks can affect the user experience due to the implicit time required to complete the change.

Finally, Ra *et al.* use the Lyapunov optimisation framework to select the optimal link for a wireless communication taking into account the energy-delay trade-off [76]. Their algorithm known as SALSA, adapts to channel conditions and requires only local information to decide whether and when it must defer a transmission in order to save energy.

VII. SENSORS OPTIMISATIONS

Sensing and context-awareness played a fundamental role in the explosion of rich mobile applications in the last years. Mobile applications often need location data to update locally relevant information, to provide a service requested by the user, to find nearby friends or when to adapt the system to manage resources efficiently. However, accessing sensing resources can be expensive in terms of energy. Consequently, continuous energy-efficient location sensing has become a hot topic both in pervasive computing and sensor networks.

In this section we concentrate on solutions at the software level, although there are also some researchers trying to demonstrate the need to re-design the system architecture at the hardware level for energy-efficient sensing [89]. Priyantha *et al.* propose adding a low power micro-controller or an additional low power core in the multi-core processor responsible for managing sensors. Such a small modification, enables most parts of the phone to enter a low power sleep state while the low power sensor processor is continuously sampling and processing sensor data.

Modern smartphones include different types of location sensors with different resolution and energy demands: GSM, WiFi-based and GPS location sensors have an average error in the order of 400m, 40m and 10m[11] respectively. In these applications, GPS is often preferred over its alternatives such

[11]Even in clear sky, GPS can present measurement errors in the order of hundreds of meters

TABLE V
SENSOR OPTIMISATIONS. MOST OF THE WORKS AIMT TO TACKLE THE CONTINUOUS LOCATION SENSING CHALLENGE. THE TABLE HIGHLIGHTS THE TECHNIQUES USED BY EACH ONE OF THE SOLUTIONS.

Sensor-based optimisations

			Sensors Used					
Cite	*Name*	*Platform*	*GPS*	*Accel.*	*GSM*	*Probab. Models*	*Adaptation*	*Description*
[82]	SenseLess	Symbian	✓	✓				Energy-efficient location-sensing combining accelerometer and GPS. It exploits the accuracy-energy trade-off. Evaluation with a real deployment.
[83]	EnLoc	Symbian	✓	✓	✓	✓		Energy-efficient location-sensing combining accelerometer and GPS. It exploits the accuracy-energy trade-off and probabilistic models of human mobility. Evaluation via simulation.
[84]	A-Loc	Android	✓	✓	✓	✓		Energy-efficient location sensing taking advantage of probabilistic models and a combination of all the location sensing resources.
[85]	EnTracked	Symbian				✓	✓	Estimates system conditions and mobility to schedule position updates to online location services. Results obtained with simulation but validated with a real deployment.
[86]	RAPS	Symbian Android	✓	✓	✓	✓	✓	Rate-adaptive positioning system that uses a probabilistic mobility model of the user, all the location-sensing resources, opportunistic synchronisation between users and resource state monitoring.
[87]	n/a	Android	✓	✓	✓		✓	Location sensing application that takes advantage of accelerometer-based suppression, location-sensing piggybacking and energy-aware adaptation of the sensing parameters.
[88]	JigSaw	Symbian iPhone					✓	General framework for continuous sensing of any sensing resource. Supports accelerometer, microphone and GPS.

as GSM/WiFi based positioning systems because it is known to be more accurate despite its higher energy demands.

Mobile applications are becoming more context-aware, specially location-aware. Consequently, support for energy-efficient continuous sensing became a demand of mobile developers to provide richer applications. In this space, it is possible to differentiate three types of solutions which can be complementary:

- Systems that combine multiple sensors to reduce the energy consumption while minimising the error.
- Methodologies that rely exclusively on probabilistic models of users' location to infer future locations to reduce the number of sensing reads.
- Solutions that try to find heuristics to adapt the sampling rate.

Ben Abdesslem *et al.* [82] combine accelerometer and GPS reads. The main idea is using less expensive sensors more often instead of more energy expensive sensors. By choosing when to use more energy-efficient sensors, it is possible to decrease the energy consumption of mobile sensing applications. However, the evaluation of the system does not provide accurate values since the authors based their results on the battery life. They gave two devices to the same user: one running SenseLess (implemented using the Python PyS60 API) and the other one sampling GPS periodically every 10s during 30 min approximately. The results indicate that the device running SenseLess takes 58.8% less energy than the one sampling periodically its GPS. However, its measurement error is increased because the system considers actions like sitting down or standing up as movements.

EnLoc [83] provides a location sensing adaptive framework that exploits mobility patterns of the user and decides which sensor to use taking into account the accuracy-energy trade-off of the different location sensors available in mobile phones. The authors take advantage of users'*Logical Mobility Tree* (LMT). This model allows the system to sample at a few uncertainty points which may be sufficient for predicting future locations. EnLoc utilises dynamic programming to find the optimal localization accuracy for a given energy budget: it decides which localisation sensor will be the best one for a given scenario and energy budget. The system was evaluated based on the measurement error using simulations from real mobility traces and energy measurements obtained from a Nokia N95 using Nokia's Energy Profiler.

A-Loc [84] incorporates probabilistic models of user location and sensor errors. It was implemented as a middleware solution for Android devices which requires applications' collaboration. A-Loc selects the most energy-efficient sensor to meet applications accuracy requirements which are either specified explicitly by applications or automatically by the system. The system uses the probabilistic models to choose among different localisation methods and tunes the energy expenditure to dynamically meet the error requirements.

EnTracked [85] looks at the problem of uploading location information to an online location server under application-specific positioning error limits. EnTracked estimates and predicts the system state and mobility of the user[12] to schedule position updates in order to minimise the power consumption while optimising robustness. Their evaluation has been done with simulations but the results have been validated in real scenarios with Symbian devices. Their results reveal potential energy savings around 40% to 50% compared to periodic sampling with a maximum (and also unlikely) error of 200m. EnTracked uses the GPS-estimated uncertainty to quickly schedule a new measurement if a potential bad measurement is performed. A similar work has been done in [90], using what they call *uncertainty-aware tolerances* which are user-defined error bounds that provide accuracy guarantees with consideration of different sources of data uncertainty: sensing uncertainty, sampling uncertainty and communication delay. They exploit hidden Markov models to predict the mobility of the users and they also take advantage of Bluetooth scans to identify static scenarios based on devices in the same location.Their simulation results point out that it is possible to achieve at least 16% of energy savings for 17.5m of tolerance and up to 80% for 250m. The system was also evaluated with emulation and a real world deployment.

GPS accuracy in urban areas can be poor. RAPS [86] takes inspiration from this observation and uses location-time history of the user to estimate user velocity and adaptively turn on GPS in case the estimated uncertainty in the prediction exceeds the accuracy threshold. RAPS integrates a bunch of solutions to improve accuracy and to reduce the energy consumption. It allows synchronising GPS readings between neighbouring mobile devices to reduce power consumption, it blocks GPS reads when the user is subscribed to cellular base stations where it is unlikely to get a GPS read (e.g. an area where the user is usually indoors) and it exploits accelerometer data to estimate user velocity. In fact, RAPS can detect human activity which has a timescale larger than multiples of 16 seconds by simply turning on the accelerometer for a short period of time. Combining those features together, allows reducing the energy consumption around 40% compared to periodic GPS sampling ($T = 180s$). In this case, the results were validated with a real deployments using Android and Symbian devices.

Zhuang *et al.* [87] implemented a location sensing middleware system that combines four techniques to reduce energy consumption in location sensing smartly: substitution, suppression, piggybacking and adaptation. Some of those techniques can be found in some of the previous works. As the authors describe in the paper, substitution makes use of alternative location-sensing mechanisms (e.g. network-based location sensing) that consumes lower power than GPS. Suppression utilises less power-intensive sensors such as accelerometers to suppress unnecessary GPS sensing if the user is static. Piggybacking synchronizes the location sensing requests from multiple running location-based applications[13]. Finally, adaptation aggressively adjusts sensing parameters such as time and distance depending on the remaining battery capacity. The system is designed as a middleware layer on the Android OS and their evaluation results show that this approach can save up to 75% of energy by reducing the access to GPS reads on 98% using real-life measurements with an error lower than 100m.

The last paper in this section is *Jigsaw* [88], a middleware solution that improves the resolution of several sensors (such as accelerometer, microphone and GPS) while also reducing the energy consumption required to sense the environment. Although Jigsaw is application-centric, it provides specific APIs to recognize users' activities using contextual information with machine learning techniques. The authors claim that one of the most-energy consuming tasks when sensing the environment is processing raw sensor data from sensors. For example, the microphone generates data at a much higher sampling rate than other sensors and requires computational-intense algorithms to be processed. The authors claim that the system can perform energy-efficient long term GPS tracking; however, the paper lacks a detailed energy evaluation.

VIII. Computation off-loading

Cloud computing is a promising technology which has the potential of providing many benefits to mobile computing such as computational power and energy efficiency. Even in the late 90s, Vahdat *et al.* [3] and Rudenko [94] considered that energy can be considered as another internet resource similar to computation. In fact, more and more applications and even OS services rely on cloud computing every year. Leveraging cloud computing services for mobile devices has received large interest recently and will definitely gain more interest in the future with services such as Apple's iCloud or Microsoft Live Mesh[14].

Those earlier works already envisioned the possibility of integrating cloud support at different levels, from programmer-decisions [95] to compiler-based [96] and automatic solutions [97]. However, factors such as network conditions can affect the efficiency of process migration. This fact opens vast research opportunities on analysing the energy impact and benefits of process migration and virtualisation in mobile handsets.

Miettinen and Nurnimen analysed the critical factors that must be taken into account for energy-efficiency in process migration from a network perspective [98]. They claim that the benefits and costs of migrating an application to the cloud can be divided into temporal (e.g. time required to transfer the data from the device to the cloud, to compute the task in the cloud and the time to transfer result information), computational and in terms of energy too. However, those benefits are dependant on the highly-variable performance of the underlying resources (e.g. CPU, bandwidth and network latency) which must be fine-grained monitored (consequently, some of the works described in section III can be used for this scenario). In fact, some authors assert that the key to the successful implementation of remote execution is prediction. The client-side must be able to predict when the cost of performing remote execution will not outweigh its benefits based on the state and the availability of resources.

[12]The system only supports pedestrians as possible movement model

[13]The energy consumption becomes even more significant if multiple applications are requesting location reads. This paper is the only one that takes this solution into account

[14]Nevertheless, those services are more focused on storage support rather than computation offload

TABLE VI
COMPUTATION OFF-LOADING

Off-loading Computation			
Cite	*Name*	*Platform*	*Description*
[91]	Spectra	Odyssey	Self-adaptive remote execution system for pervasive computing based on Odyssey OS. Programmers must decide which methods can be offloaded.
[92]	NWSLite	n/a	Computationally-efficient and highly accurate proactive utility for mobile computing off-loading. Decides when to perform offloading based on available resources state and application needs.
[93]	Maui	WiMo	Fine-grained monitoring of applications to efficiently off-load computation to the cloud in Microsoft .Net platform. Requires developers to decide which methods can be offloaded.

Spectra [91] (built on top of Odyssey OS) is a remote execution system designed for pervasive computing which monitors environmental conditions and both local and remote resources state (mainly energy but also CPU, network and cache). With this knowledge, Spectra adjusts the relative importance of each task and decide where to locate functionality dynamically. However, Spectra requires programmers to decide how to partition an application manually for different fidelities. Spectra chooses the execution plan that maximizes a user-provided utility function at runtime. The evaluation results show that Spectra can select the best execution plan for the applications they've tested.

NWSLite [92] is a computationally efficient and highly accurate prediction utility for mobile computing off-loading. NWSLite was implemented as an extension of Network Weather Service (NWS), a forecasting toolkit for adaptive scheduling of grid computing applications. The authors draw attention to the fact that network bandwidth and latency dictate the time required for communication while CPU availability on both the device and the target impacts local and remote execution time. NWSLite leverages statistical techniques[15] which use past behavior to predict the potential costs of offloading the application to the server. The goal of the system is to enable high accuracy to reduce prediction error because a large deviation can cause incorrect decisions about the best execution choice for the device as it is also described in Maui [93]. Compared to Spectra, NWSLite has a much more fine-grained tool for monitoring applications and resources. It applies statistical forecasting techniques to individual performance histories and generates forecast reports for the resources being monitored. In fact, it selects the most appropriate statistical technique at a given time for a specific resource (e.g. from the simple *Last Value* to *Sliding Window* techniques).

Maui is a robust architecture for process migration which decides at runtime the methods that could be executed remotely with different techniques [93]. The decisions are driven by an optimisation engine that aims to achieve the best energy savings under the current connectivity constrains and historical data about the process execution. Maui supports two mechanisms for remote execution on mobile devices. As the authors say, the first one relies on programmers to specify how to partition a program by allowing them to define which states could be executed remotely and how to adapt the program partitioning scheme to the changing network conditions. Secondly, it supports full process and full VM migration to allow individual applications to migrate to the cloud. As we have previously remarked, Maui monitors resources with a fine-grained profiler to know their current state but it also uses historical program data to predict how the process will behave. In fact, Maui supports mechanisms to identify the current state of the running methods to extract only the program state needed for the migration. The evaluation of a real system deployment shows that it is possible to achieve important energy savings in resource-intensive applications.

There are also recent works looking at the integration of cloud computing services with mobile devices without an energy-efficiency perspective. *CloneCloud* [99] describes a flexible architecture for Android devices that enables accessing cloud computation to augment mobile device computational power without requiring the programer to think about application partitioning. CloneCloud enables applications to offload part of their execution from the mobile device onto *device clones* in the cloud at application-layer virtual machines such as Dalvik VM, Java VM and .NET. To conclude, *Cloudlets* [100] discusses the technical obstacles and the performance benefits behind exploiting virtual machine technology on mobile handsets to rapidly migrate the whole OS to instantiate customized service software in the cloud.

IX. CONCLUSION

Mobile handsets are still power-hungry devices despite the tremendous efforts done by hardware manufacturers and operating system vendors in the last years. Modern mobile platforms such as Android and iPhone are built as modifications of general-purpose operating systems which do not consider energy-efficiency as a key performance goal. In fact, modern handsets incorporate power-hungry hardware resources such as touchscreen displays and location sensors, and they support Internet data services so they are always connected to the network. All these resources bootstrapped a rich ecosystem of mobile applications but their design is clearly driven by usability factors rather than energy efficiency.

Since the mid-90s, researchers have been emphasizing the need of considering energy as a fundamental system resource in mobile devices. In this survey, we covered the most relevant articles about energy-efficient resource management in mobile systems that can be implemented in current mobile handsets. We classified the papers in six categories based on the type of optimisation they propose: operating system and efficient resource management, energy measurements and power models, users' interaction with mobile resources, wireless interfaces and sensors management, and finally, we talked about the new

[15]The authors mention that performing the prediction can be expensive in terms of power consumption since the statistical techniques can use floating-point operations

opportunities that process and system migration to the cloud can offer. As far as we know, this is the first survey about mobile green computing in the last decade and we strongly believe that some of the improvements highlighted in this survey will be part of future mobile OS design.

Managing mobile resources from an energy-efficient perspective without diminishing the user experience is clearly one of the most challenging problems in mobile computing nowadays. Power management considerations often require certain actions to be deferred, avoided or slowed down to prolong battery life. It can even require changing dynamically the power states of the hardware components and applications behaviour depending on the available resources. However, these techniques can impact the user experience with the handsets.

Moreover, limitations such as the lack of energy-aware support from hardware components make this problem even harder to solve. Hardware manufacturers do not offer enough information about the energy consumption in runtime to the operating system and applications. Many power-hungry resources are embedded in the same chipset as in modern ARM-based chips and the system does not have enough visibility about the power consumption and the power modes of the different resources available in the device. Consequently, most of the works rely on energy measurements obtained with external multimeters or with inaccurate power models[16] obtained from linear regression techniques.

We hope that this survey motivates the need for energy-aware support in mobile operating systems. Efficient resource management requires new application models, schedulers and also non-intrusive tools for monitoring the different resources and energy demands caused by applications and users. Understanding how users interact with their applications and resources can be directly translated into important energy savings, as demonstrated by some of the papers included in the survey. In fact, latest publications demonstrate that users interact with their devices and applications at specific locations and times but, because of the difficulty of accessing this data from the handsets, these studies generally lack statistical representativeness. Similarly, the deployment of new wireless standards such as LTE will clearly open new resource management research challenges in addition to the ones highlighted in this survey. Consequently, we believe there are enough indications to claim that an energy-efficient operating system must be context-aware and user-centric.

On the other hand, mobile operating systems must take advantage of all the possibilities they have to save energy. As we can currently see in modern platforms and applications, the dependency on cloud services is becoming more necessary for different purposes such as storage and computation offloading. However, we strongly believe that collaborative mechanisms for sharing resources opportunistically with co-located devices using low-power local wireless connectivities can have two immediate benefits. Firstly, devices can save important amounts of energy and secondly, they can improve the user experience and quality of service by enabling access to remote resources that might not be available locally. Nevertheless, the characteristics of current local wireless interfaces such as Bluetooth make supporting this feature difficult. Mobile computation should not be limited exclusively to the local device and, as a result, resources management should be distributed and collaborative within groups of collocated devices. This approach will need to face new trust schemes, access control policies, security mechanisms, privacy and possibly incentive schemes while trying to minimise the negative impact of users' mobility.

[16]Researchers are using power models validated for an specific device to measure the power consumption in phones with different hardware settings to the one the model was originally built.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers, Dr. Pan Hui (Deutsche Telekom Laboratories), Dr. Yu Xiao (Aalto University), Dr. Andrew Rice (Cambridge University), and IEEE Communications Surveys & Tutorials Editor-In-Chief for their insightful comments and suggestions.

APPENDIX A
ENERGY MEASUREMENTS AND POWER MODELS

See tables on following pages.

REFERENCES

[1] G. F. Welch, "A survey of power management techniques in mobile computing operating systems," *SIGOPS Operating Systems Review*, vol. 29, pp. 47–56, October 1995.

[2] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, "ECOSystem: managing energy as a first class operating system resource," in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS-X. New York, NY, USA: ACM, 2002, pp. 123–132.

[3] A. Vahdat, A. Lebeck, and C. S. Ellis, "Every joule is precious: the case for revisiting operating system design for energy efficiency," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, ser. EW 9. New York, NY, USA: ACM, 2000, pp. 31–36.

[4] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat, "Currentcy: a unifying abstraction for expressing energy management policies," in *Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2003, pp. 4–4.

[5] Noble, Brian and Satyanarayanan, M. and Price, Morgan, "A Programming Interface for Application-Aware Adaptation in Mobile Computing," in *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing*. Berkeley, CA, USA: USENIX Association, 1995, pp. 57–66.

[6] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in *Proceedings of the seventeenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 1999, pp. 48–63.

[7] Flinn, Jason and Satyanarayanan, M., "Managing battery lifetime with energy-aware adaptation," *ACM Trans. Comput. Syst.*, vol. 22, pp. 137–179, May 2004.

[8] S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich, "Apprehending joule thieves with cinder," *SIGCOMM Computer Communication Review*, vol. 40, pp. 106–111, January 2010.

[9] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich, "Energy management in mobile devices with the cinder operating system," in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 139–152.

[10] N. Vallina-Rodriguez and J. Crowcroft, "ErdOS: achieving energy savings in mobile OS," in *Proceedings of the sixth international workshop on MobiArch*, ser. MobiArch '11. New York, NY, USA: ACM, 2011, pp. 37–42.

[11] D. Chu, A. Kansal, J. Liu, and F. Zhao, "Mobile apps: it's time to move up to CondOS," in *Proceedings of the 13th USENIX conference on Hot topics in operating systems*, ser. HotOS'13. Berkeley, CA, USA: USENIX Association, 2011, pp. 16–16.

Device: Openmoko Neo Freenunner (rev 6). Paper: [28]
Measurement Tool: National Instruments PCI-6229DAQ @ f= n/a

Overall energy consumption per Power State. See paper for details (break-down).

State	*Power (mW)*
Suspended	68.6
Idle, (backlight off)	268.8
Audio Playback	320.0
Video Playback (excluding backlight)	453.5
Sending SMS (excluding backlight)	302.2
GSM phonecall (excluding backlight)	1054.3
Email (excluding backlight) over WiFi	432.4
Email (excluding backlight) over GPRS	610.0
WebBrowsing (excluding backlight) over WiFi	352.8
WebBrowsing (excluding backlight) over GPRS	429.0
Bluetooth (near)	36.0
Bluetooth (far)	44.9

Flash Storage

Metric	*NAND*	*SD*
Idle (mW)	0.4	1.4
Read efficiency (MiB/J)	65.0	31.0
Write efficiency (MiB/J)	10.0	5.2

GPS

State	*Power (mW)*
Enabled (internal antenna)	143.1 +/- 0.05%
Enabled (external antenna)	166.1 +/- 0.04%
Disabled	0.0

Device: Nokia N95. Paper: [29]
Measurement Tool: AGILENT 66319D @ f= n/a

Power consumption of cellular services

Scenario	*GSM (mW)*	*UMTS (mW)*
Receiving a phone call	612.7	1224.3
Making a phone call	683.6	1265.7
Idle	15.1	25.3

Duration and energy consumption for making handoffs between GSM and UMTS

Handoff	*Power (mW)*	*Time(s)*
GSM to UMTS	1389.5	1.7
UMTS to GSM	591.9	4.2

Power Model for Nokia N95. Paper: [30]
Measurement Tool: Nokia Energy Profiler @ 4Hz

	3G	**GSM**	**WiFi**
Transfer Energy (J) (x= number bytes)	0.025(x)+3.5	0.036(x)+1.7	0.007(x) + 5.9
Tail energy (J/sec)	0.62	0.25	NA
Maintenance (J/sec)	0.02	0.03	0.05
tail-time (seconds)	12.5	6	NA
Energy per 50 KB transfer with a 20s interval	12.5	5.0	7.6

[12] J. Flinn and M. Satyanarayanan, "PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, ser. WMCSA '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 2–.

[13] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, ser. EW 9. New York, NY, USA: ACM, 2000, pp. 37–42.

[14] M. Anand, E. B. Nightingale, and J. Flinn, "Ghosts in the machine: interfaces for better power management," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services* , ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 23–35.

[15] A. B. Lago and I. Larizgoitia, "An application-aware approach to efficient power management in mobile devices," in *Proceedings of the Fourth International ICST Conference on Communication System software and middleware (COMSWARE)*, ser. COMSWARE '09. New York, NY, USA: ACM, 2009, pp. 11:1–11:10.

[16] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala: a platform for os-level power management," in *Proceedings of the 4th ACM European conference on Computer systems*, ser. EuroSys '09. New York, NY, USA: ACM, 2009, pp. 289–302.

[17] C. S. Ellis, "The Case for Higher-Level Power Management," in *Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems*, ser. HOTOS '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 162–.

[18] "Millywatt proj." http://www.cs.duke.edu/ari/millywatt.

[19] X. Liu, P. Shenoy, and M. Corner, "Chameleon: application level power management with performance isolation," in *Proceedings of the 13th annual ACM international conference on Multimedia*, ser. MULTIMEDIA '05. New York, NY, USA: ACM, 2005, pp. 839–848.

[20] G. Banga, P. Druschel, and J. C. Mogul, "Resource containers: a new

Energy cost of Wi-Fi at different power modes
Device: Nokia N95, HTC G1, Nokia N810. Paper: [32]
Measurement Tool: Fluke 189 @ f= n/a

WiFi Power Mode	**Average Power (mW)**		
	N810	*G1*	*N95*
IDLE	884	650	1038
SLEEP	42	68	88
TRANSMIT	1258	1097	1687
RECEIVE	1181	900	1585

Energy Cost per Application
Device: Symbian (no specific handsedt detailed). Paper: [38]
Measurement Tool: n/a

Application	**Average Power (mW)**
GSM Phone call	900
Video Playback	1100
Bluetooth Class 2 (IDLE)	2.5
Bluetooth Scan	75

Power Model for Android (G1). Paper: [33]
Measurement Tool: Battery Voltage Curve validated with Monsoon meter at 5KHz

HW unit	**Parameter**	**Range**	**Coefficient**
CPU	Avg CPU utilisation operating @ 384 MHz	0-100	4.34 mW/%
	Avg CPU utilization operating @ 246 MHz	0 -100	3.42 mW/%
	CPU On	0-1	121.46 mW
WiFi	Channel Bitrate	0-54	$\beta_{cr} = 48 - 0.768 * ChannelBitrate$ mW
	Low-power mode	0-1	20 mW
	High-power mode	0-1	$710 + \beta_{cr} * UplinkBitratemW$
Audio	Audio On	0-1	384.62 mW
LCD	Brightness	0-255	2.40 mW/step
GPS	GPS ON	0-1	429.55 mW
	GPS Sleep	0-1	173.55 mW
Cellular	Data rate	0-inf	NA
	Downlink queue	0-inf	NA
	Uplink queue	0-inf	NA
	3G Idle	0-1	10 mW
	3G FACH (Shared Channel)	0-1	401 mW
	3G DCH (Dedicated Channel)	0-1	570 mW

facility for resource management in server systems," in *Proceedings of the third symposium on Operating systems design and implementation*, ser. OSDI '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 45–58.

[21] N. Vallina-Rodriguez, P. Hui, J. Crowcroft, and A. 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*, ser. MobiHeld '10. New York, NY, USA: ACM, 2010, pp. 9–14.

[22] R. Neugebauer and D. McAuley, "Energy Is Just Another Resource: Energy Accounting and Energy Pricing in the Nemesis OS," in *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 67–.

[23] D. Narayanan and M. Satyanarayanan, "Predictive Resource Management for Wearable Computing," in *Proceedings of the 1st international conference on Mobile systems, applications and services*, ser. MobiSys '03. New York, NY, USA: ACM, 2003, pp. 113–128.

[24] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini, "Code transformations for energy-efficient device management," *Computers, IEEE Transactions on*, vol. 53, no. 8, pp. 974 – 987, aug. 2004.

[25] N. Pettis, L. Cai, and Y.-H. Lu, "Statistically Optimal Dynamic Power Management for Streaming Data," *IEEE Trans. Comput.*, vol. 55, pp. 800–814, July 2006.

[26] F. Fitzek, S. Rein, M. Perucci, T. Schneider, and C. Guhmann, "Low complex and power efficient text compressor for cellular and sensor networks," in *15th IST Mobile and Wireless Communication Summit*, 2006.

[27] N. Banerjee, A. Rahmati, M. D. Corner, S. Rollins, and L. Zhong, "Users and batteries: interactions and adaptive energy management in mobile systems," in *Proceedings of the 9th international conference on Ubiquitous computing*, ser. UbiComp '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 217–234.

[28] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, ser. USENIXATC'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21.

[29] G. P. Perrucci, F. H. P. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the impact of 2G and 3G network usage for mobile phones' battery life," in *Wireless Conference, 2009. EW 2009. European*, 2009, pp. 255–259.

[30] N. Balasubramanian, A. Balasubramanian, and A. 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*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 280–293.

[31] A. Rice and S. Hay, "Decomposing power measurements for mobile devices," in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, 29 2010-april 2 2010, pp. 70 –78.

[32] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. 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*, ser. e-Energy '10. New York, NY, USA: ACM, 2010, pp. 75–84.

[33] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on*

Power Model for Android (G1). Paper: [34]
Measurement Tool: Fluke i30 @ 1Hz

HW unit	Parameter	Range	Coefficient
CPU	Avg CPU utilisation while operating at 384 MHz	0-100	3.97 mW/%
	Avg CPU utilization while operating at 246 MHz	0 -100	2.79 mW/%
Screen	Fraction of the time interval with the screen on	0-1	150.31 mW
	Screen Brightness	0-255	2.07 mW/step
Call	Fraction of the time interval where the phone call is ringing	0-1	761.70 mW
	Fraction of time interval during a phone call	0-1	389.97 mW
EDGE	Fraction of time interval where there is traffic	0-1	1.77 mW
	Number of bytes transferred during time interval	0-inf	3.47 mW/byte
WiFi	Fraction of time interval WiFi connection is on	0-1	1.77 mW
	Fraction of time where there is WiFi traffic	0-1	658.93 mW
	Count of bytes transferred during time interval	0-inf	0.518 mW/byte
SD Card	Number of sectors transferred to/from SD Card	0-inf	0.0324
DSP	Fraction of time interval music is on	0-1	275.65 mW
System	Fraction of time interval phone is not idle	0-1	169.08 mW

Paper: [73]
Measurement Tool: Fluke i30 @ 1Hz

HW unit	Parameter	Range	Coefficient
CPU	Avg CPU utilisation while operating at 384 MHz	0-100	3.97 mW/%
	Avg CPU utilization while operating at 246 MHz	0 -100	2.79 mW/%
Screen	Fraction of the time interval with the screen on	0-1	150.31 mW
	Screen Brightness	0-255	2.07 mW/step
Call	Fraction of the time interval where the phone call is ringing	0-1	761.70 mW
	Fraction of time interval during a phone call	0-1	389.97 mW
EDGE	Fraction of time interval where there is traffic	0-1	1.77 mW
	Number of bytes transferred during time interval	0-inf	3.47 mW/byte
WiFi	Fraction of time interval WiFi connection is on	0-1	1.77 mW
	Fraction of time where there is WiFi traffic	0-1	658.93 mW
	Count of bytes transferred during time interval	0-inf	0.518 mW/byte
SD Card	Number of sectors transferred to/from SD Card	0-inf	0.0324
DSP	Fraction of time interval music is on	0-1	275.65 mW
System	Fraction of time interval phone is not idle	0-1	169.08 mW

Energy cost of various cellular and wi-fi activities
Paper: [75]

Device	*Cellular (EDGE)*			*WiFi*				
	E_e	E_{tx} (J/min)		E_e	E_m(J/min)		E_{tx} (J/MB	
	(J/min)	Download	Upload	(J)	PSM	No PSM	Download	Upload
HTC Wizard	1.2-6	40-50	95-125	5	19	61	5-7	7-11
HTC Tornado	1.2-2	100-150	170-300	10	6	53	4-6	5-7
HP iPAQ hw6925	1-2	130-160	220-330	13	4	46	5-14	6-15

E_e = Energy required for establishing connection
E_{tx} = Energy required for transmitting 1MB of data
E_m = Energy required for maintaining the connection alive
PSM = WiFi Power Save Mode

Hardware/software codesign and system synthesis, ser. CODES/ISSS '10. New York, NY, USA: ACM, 2010, pp. 105–114.

[34] A. Shye, B. Scholbrock, and G. 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*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 168–178.

[35] Y. Xiao, R. Bhaumik, Z. Yang, M. Siekkinen, P. Savolainen, and A. 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*, ser. GREENCOM-CPSCOM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 27–34.

[36] N. Ravi, J. Scott, L. Han, and L. Iftode, "Context-aware Battery Management for Mobile Phones," in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, march 2008, pp. 224 –233.

[37] E. Oliver, "Diversity in smartphone energy consumption," in *Proceedings of the 2010 ACM workshop on Wireless of the students, by the students, for the students*, ser. S3 '10. New York, NY, USA: ACM, 2010, pp. 25–28.

[38] E. Oliver and P. S. Keshav, "Data Driven Smartphone Energy Level Prediction," *University of Waterloo, Technical Report CS-2010-06*, 2010.

[39] A. Rahmati, A. Qian, and L. Zhong, "Understanding human-battery interaction on mobile phones," in *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, ser. MobileHCI '07. New York, NY, USA: ACM, 2007, pp. 265–272.

[40] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Measuring serendipity: connecting people, locations and interests in a mobile 3G network," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 267–279.

[41] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services* , ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 165–178.

[42] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 281–287.

[43] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum, "Live-Lab: measuring wireless networks and smartphone users in the field," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, pp. 15–20, January 2011.

[44] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proceedings of the 8th international conference on Mobile systems, applications, and services* , ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 179–194.

[45] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, "MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, ser. MobiSys '07. New York, NY, USA: ACM, 2007, pp. 57–70.

[46] Reality Mining Project, http://reality.media.mit.edu/.

[47] Y. Wen, R. Wolski, and C. Krintz, "Online Prediction of Battery Lifetime for Embedded and Mobile Devices," 2004.

[48] D. Panigrahi, S. Dey, R. Rao, K. Lahiri, C. Chiasserini, and A. Raghunathan, "Battery Life Estimation of Mobile Embedded Systems," in *Proceedings of the The 14th International Conference on VLSI Design*, ser. VLSID '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 57–.

[49] D. Rakhmatov, S. Vrudhula, and D. A. Wallach, "Battery lifetime prediction for energy-aware computing," in *Proceedings of the 2002 international symposium on Low power electronics and design*, ser. ISLPED '02. New York, NY, USA: ACM, 2002, pp. 154–159.

[50] D. Rakhamtov and S. Vrudhula, "Time-to-failure estimation for batteries in portable electronic systems," in *Proceedings of the 2001 international symposium on Low power electronics and design*, ser. ISLPED '01. New York, NY, USA: ACM, 2001, pp. 88–91.

[51] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz, "Primary user behavior in cellular networks and implications for dynamic spectrum access," *Comm. Mag.*, vol. 47, pp. 88–95, March 2009.

[52] W. L. Tan, F. Lam, and W. C. Lau, "An Empirical Study on the Capacity and Performance of 3G Networks," *IEEE Transactions on Mobile Computing*, vol. 7, pp. 737–750, June 2008.

[53] A. Rahmati and L. Zhong, "A longitudinal study of non-voice mobile phone usage by teens from an underserved urban community," *IEEE Computing and Control Engineering Journal*, 2010.

[54] H. Falaki, R. Govindan, and D. Estrin, "Smart Screen Management on Mobile Phones," *Center for Embedded Network Sensing Technical Reports. Num. 74*, 2009.

[55] L. S. Brakmo, D. A. Wallach, and M. A. Viredaz, "u-Sleep: a technique for reducing energy consumption in handheld devices," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services* , ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 12–22.

[56] S. Frattasi, H. Fathi, F. Fitzek, R. Prasad, and M. Katz, "Defining 4G technology from the users perspective," *Network, IEEE*, vol. 20, no. 1, pp. 35 –41, jan.-feb. 2006.

[57] S.-R. Yang, "Dynamic power saving mechanism for 3G UMTS system," *Mob. Netw. Appl.*, vol. 12, pp. 5–14, January 2007.

[58] Wibree, http://www.bluetooth.com/Pages/Low-Energy.aspx.

[59] Z. Shelby and C. Bormann, "6lowpan: The wireless embedded internet, wiley," 2009.

[60] M. Corson, R. Laroia, J. Li, V. Park, T. Richardson, and G. Tsirtsis, "Toward Proximity-Aware Internetworking," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 26–33, december 2010.

[61] S.-L. Tsao and C.-H. Huang, "Review: A survey of energy efficient MAC protocols for IEEE 802.11 WLAN," *Comput. Commun.*, vol. 34, pp. 54–67, January 2011.

[62] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks," *Wireless Networks*, vol. 7, pp. 343–358, September 2001.

[63] C.-C. Lee, J.-H. Yeh, and J.-C. Chen, "Impact of inactivity timer on energy consumption in WCDMA and CDMA2000," in *Wireless Telecommunications Symposium, 2004*, may 2004, pp. 15 – 24.

[64] D. Bertozzi, A. Raghunathan, L. Benini, and S. Ravi, "Transport Protocol Optimization for Energy Efficient Wireless Embedded Systems," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, ser. DATE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 10 706–.

[65] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access with bounded slowdown," *Wireless Networking*, vol. 11, pp. 135–148, January 2005.

[66] F. R. Dogar, P. Steenkiste, and K. 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* , ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 107–122.

[67] J. Liu and L. Zhong, "Micro power management of active 802.11 interfaces," in *Proceeding of the 6th international conference on Mobile systems, applications, and services*, ser. MobiSys '08. New York, NY, USA: ACM, 2008, pp. 146–159.

[68] E. Tan, L. Guo, S. Chen, and X. Zhang, "PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs," in *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, oct. 2007, pp. 123 –132.

[69] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems*, vol. 24, no. 3, pp. 250–291, Aug. 2003.

[70] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta, "Wireless wakeups revisited: energy management for voip over wi-fi smartphones," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, ser. MobiSys '07. New York, NY, USA: ACM, 2007, pp. 179–191.

[71] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: an event driven energy saving strategy for battery operated devices," in *Proceedings of the 8th annual international conference on Mobile computing and networking*, ser. MobiCom '02. New York, NY, USA: ACM, 2002, pp. 160–171.

[72] Y. Agarwal, C. Schurgers, and R. Gupta, "Dynamic power management using on demand paging for networked embedded systems," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '05. New York, NY, USA: ACM, 2005, pp. 755–759.

[73] G. Ananthanarayanan and I. Stoica, "Blue-Fi: enhancing Wi-Fi performance using bluetooth signals," in *Proceedings of the 7th international conference on Mobile systems, applications, and services* , ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 249–262.

[74] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: wireless LAN discovery via ZigBee interference signatures," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, ser. MobiCom '10. New York, NY, USA: ACM, 2010, pp. 49–60.

[75] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *Proceedings of the 5th international conference on Mobile systems, applications and services*, ser. MobiSys '07. New York, NY, USA: ACM, 2007, pp. 165–178.

[76] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 255–270.

[77] C. Poellabauer and K. Schwan, "Energy-Aware Traffic Shaping for Wireless Real-Time Applications," in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 48–.

[78] C.-F. Chiasserini and R. Rao, "Improving battery performance by using traffic shaping techniques," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 7, pp. 1385 –1394, Jul. 2001.

[79] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 323–336.

[80] B. C. Housel and D. B. Lindquist, "WebExpress: a system for optimizing Web browsing in a wireless environment," in *Proceedings of the 2nd annual international conference on Mobile computing and networking*, ser. MobiCom '96. New York, NY, USA: ACM, 1996, pp. 108–116.

[81] R. Krashinsky, "Efficient web browsing for mobile clients using HTTP compression," *Distributed Operating Systems term project, Massachusetts Institute of Technology*, 2000.

[82] F. Ben Abdesslem, A. Phillips, and T. Henderson, "Less is more: energy-efficient mobile sensing with senseless," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, ser. MobiHeld '09. New York, NY, USA: ACM, 2009, pp. 61–62.

[83] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox, "EnLoc: Energy-Efficient Localization for Mobile Phones," in *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, no. 4. IEEE, Apr. 2009, pp. 2716–2720.

[84] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, "Energy-accuracy trade-off for continuous mobile device location," in *Proceedings of the 8th international conference on Mobile systems, applications, and services* , ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 285–298.

[85] M. B. Kjaergaard, J. Langdal, T. Godsk, and T. Toftkjaer, "EnTracked: energy-efficient robust position tracking for mobile devices," in *Proceedings of the 7th international conference on Mobile systems, applications, and services* , ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 221–234.

[86] J. Paek, J. Kim, and R. Govindan, "Energy-efficient rate-adaptive GPS-based positioning for smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services* , ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 299–314.

[87] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services* , ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 315–330.

[88] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The Jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 71–84.

[89] N. Priyantha, D. Lymberopoulos, and J. Liu, "EERS: Energy Efficient Responsive Sleeping on Mobile Phones," ser. ACM PhoneSense'10, 2010.

[90] T. Farrell, R. Cheng, and K. Rothermel, "Energy-Efficient Monitoring of Mobile Objects with Uncertainty-Aware Tolerances," in *Proceedings of the 11th International Database Engineering and Applications Symposium*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 129–140.

[91] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing Performance, Energy, and Quality in Pervasive Computing," in *Proceedings of the 22 nd International Conference on Distributed Computing Systems*, ser. ICDCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 217–.

[92] S. Gurun, C. Krintz, and R. Wolski, "NWSLite: a light-weight prediction utility for mobile devices," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 2–11.

[93] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[94] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *SIGMOBILE Mobile Computing Communications Rev.*, vol. 2, pp. 19–26, January 1998.

[95] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, ser. CASES '01. New York, NY, USA: ACM, 2001, pp. 238–246.

[96] U. Kremer, J. Hicks, and J. M. Rehg, "Compiler-directed remote task execution for power management," in *Workshop on Compilers and Operating Systems for Low Power*, 2000.

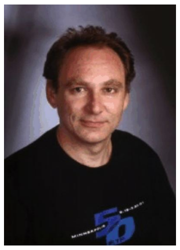
[97] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *Proceedings of the 1st international conference on Mobile systems, applications and services*, ser. MobiSys '03. New York, NY, USA: ACM, 2003, pp. 273–286.

[98] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 4–4.

[99] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314.

[100] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, October 2009.

Narseo Vallina-Rodriguez He received a B.S. and a M.S. degree in Telecommunications Engineering in 2007 from the University of Oviedo, Asturias, Spain. He is currently a PhD candidate in Computer Science in the Computer Lab at the University of Cambridge under the supervision of Prof. Crowcroft. His research interests include energy-efficiency on mobile systems, distributed systems, social networks and ubiquitous computing.



Jon Crowcroft Jon Crowcroft has been the Marconi Professor of Communications Systems in the Computer Laboratory since October 2001. He has worked in the area of Internet support for multimedia communications for over 30 years. Three main topics of interest have been scalable multicast routing, practical approaches to traffic management, and the design of deployable end-to-end protocols. Current active research areas are Opportunistic Communications, Social Networks, and techniques and algorithms to scale infrastructure-free mobile systems. He leans towards a "build and learn" paradigm for research. He graduated in Physics from Trinity College, University of Cambridge in 1979, gained an MSc in Computing in 1981 and PhD in 1993, both from UCL. He is a Fellow of the ACM, a Fellow of the British Computer Society, a Fellow of the IET and the Royal Academy of Engineering and a Fellow of the IEEE. He likes teaching, and has published a few books based on learning materials.