Energy Aware Self-Adaptation in Mobile Systems

Luca Ardito
Department of Control and Computer Engineering
Politecnico di Torino, Italy
luca.ardito@polito.it

***Abstract*—The increasing proliferation of mobile handsets, and the migration of the information access paradigm to mobile platforms, leads researchers to study the energy consumption of this class of devices. The literature still lacks metrics and tools that allow software developers to easily measure and optimize the energy efficiency of their code. Energy efficiency can definitely improve user experience increasing battery life. This paper aims to describe a technique to adapt the execution of a mobile application, based on the actual energy consumption of the device, without using external equipment.**

***Index Terms*—Self-Adaptation, Energy Awareness, Energy Aware Software, Android, Context-Awareness.**

I. Research Problem and Motivation

Worldwide mobile device sales will reach 821 Million units in 2012 and will rise to 1.2 Billion in 2013 [1]. Inevitably the paradigm for accessing information and Internet services is increasingly migrating to mobiles. Despite the improvements in terms of hardware, the main issue that affects these handsets is their limited battery life. The computational capabilities of these devices are growing rapidly, while their size is decreasing and they are increasingly becoming part of our daily life by providing us a wide range of functions. However, the use of these tools significantly affects the energy consumption of the portable device and the advances in the development of battery technologies cannot keep pace with this rapid growth of energy demand. The applications, or *apps*, typically available on various marketplaces, focus on content, usability, and originality, to gain the maximum number of users. On the other hand, developers typically do not consider how much energy their app will consume because hardware is responsible for energy consumption. When developers include ads in their application to obtain revenues, they do not take into account the negative implications of this choice in terms of energy efficiency [2]. Even if software does not consume energy directly, it has a direct influence on the energy consumption of the underlying hardware. As a matter of fact, applications and operating systems indicate how the information is processed and consequently drive the hardware behaviour. In recent works, we set up and published some lab experiments [3][4][5] to understand the impact of software in energy consumption. Results showed software applications could actually increment the energy consumption of a device from 10% up to 85% but these values depend on the specific device and context in which the experiment is conducted. The main issue is the lack of metrics and tools that allow software developers to easily measure and optimize [6] the energy efficiency of their code.

II. Background and Related Work

Energy profiling of mobile devices is an active research field, especially as regards mobile phones and embedded devices. The concept of energy-awareness is based on a complete knowledge on how and where energy is consumed by a device. From a software engineering point of view, most contributions are devoted to developing frameworks and tools for energy metering and profiling to understand, design, and implement better mobile applications. Balasubramanian et al. in [7] described a power consumption model based on regression. This solution needs a training stage and it logs different device subsystems' power consumption (such as CPU, Wi-Fi, etc.) using Android Wakelock drivers. This kind of techniques does not require external equipment to build the energy consumption model but it is crucial to reduce the overhead needed to build the model [8]. Murmuria et al. in [9] dealt with the issue of "tail energy overhead": this phenomenon leaves the hardware in a state of high-energy consumption for a period of time after the end of the transmission. Pathak et al. in [10] and [11] analyzed the issue of energy bugs defined as "An error in the system that causes an unexpected amount of high-energy consumption by the system as a whole". These authors also proposed taxonomy of energy bugs in the context of smartphones and proposed a systematic framework called "EDB" to diagnose "*ebugs*" on smartphones. Qian et al in [12] studied resource usage with a cross-layer interaction starting from user input and application behaviour to HTTP transport and radio usage. This methodology is composed of two phases: the online data collection and the offline analysis of previously gathered data. The outputs are the total energy used, the event that used it, and the percentage of the total used. Other methods, such as the one proposed by Abogharaf et al. in [13], use an external measurement unit to collect energy usage data. This kind of approach obtains more precise data than the other methods described before but, on the other hand, can only be used for lab purposes. As regards software self-adaptation driven by energy information in a mobile environment, there is not much literature on the topic. Peddersen and Parameswaran in [14] dealt with the problem but in the field of embedded systems. The combination of power consumption data and self-adaptation in mobile systems is missing in all these approaches.

III. Approach and Uniqueness

The objective and uniqueness of my research is to define a technique for mobile applications, which exploits self-adaptation driven by real-time energy consumption data, to increment battery life. The requirement is that energy

978-1-4673-3076-3/13 © 2013 IEEE

ICSE 2013, San Francisco, CA, USA
ACM Student Research Competition

consumption data should be collected from the Hardware Layer up to the Application Layer in real time (e.g. by built-in power meters) as shown in FIGURE I.

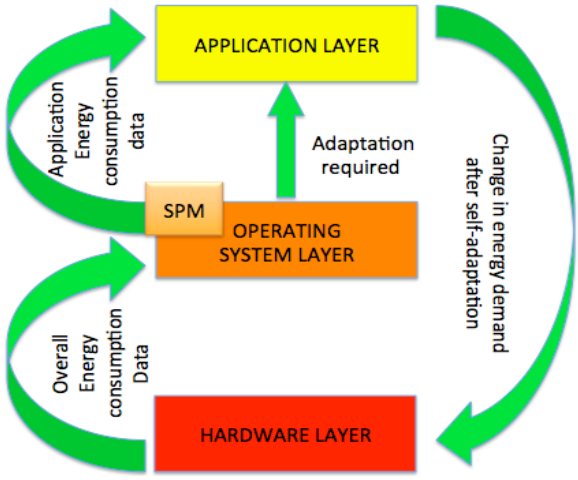

FIGURE I ENERGY CONSUMPTION DATA FLOW

Once energy consumption data is available at runtime, it is possible to apply different self-adaptation techniques such as: Conditional Operation, Function Selection, Control Flow Adaptation, etc. as described by Peddersen and Parameswaran in [14]. The technique I am proposing in this paper works as follows: the Operating System Layer manages the overall energy consumption data coming from the Hardware Layer by means of Smart Power Management module called SPM. SPM divides the overall energy consumption data based on current usage of running applications, and sends this data to each application. SPM also imposes constraints and target thresholds based on different user profiles. When necessary, SPM notifies the apps exceeding the threshold. The notifications are intended as asynchronous messages sent by the Operating System (such as Android Intents) so that every application can receive them. If an application can manage the notifications sent by SPM, it will modify its behaviour exploiting self-adaptation techniques. The notified applications will finally find the best trade-off between energy consumption and features provided to the user. The data, on which the self-adaptation is performed, is the instant energy consumption of the peripherals that compose the device. This mechanism can be better explained with an example: when a self-adapting application APP1 starts, it sends an asynchronous message to SPM to subscribe to future notifications from SPM. Based on the user profile defined by the user, SPM will send the application the energy constraints in a string format through a notification, e.g., "Resource: GPS, Max Power: 380mW, Max Time over power limit: 60sec". If APP1 crosses both the thresholds, SPM will send a warning message: "Warning, Resource: GPS, Limit: 380mW, Measured: 440mW". This warning triggers APP1 self-adaptation. If the desired result is not reached after a predefined number of iterations, or if an application has not subscribed to receive notifications, SPM will notify the user about the energy issue, and he/she can choose whether to ignore the message or kill one of the proposed applications. After the implementation phase, further studies are needed to verify the energy overhead required by this technique. The methodological approach I adopt for the validation of this technique is that of empirical software engineering [15][16], performing experiments and case studies.

IV. RESULTS

In preliminary work [17] we designed a first prototype of energy-aware application called *gLCB*. In detail *gLCB* is a Context-Aware application, which retrieves context information (such as geographical location, Wi-Fi hotspots, Bluetooth devices, etc.) from an Android OS based smartphone and sends the collected data to a server for further processing. *gLCB* has four main user profiles called VERY LOW, LOW, NORMAL, and HIGH. Each user profile differs from the others in their different configurations, which cause a variation of data retrieving process and upload ratio. Selecting different user profiles affects the energy behaviour of the application. *gLCB* also has another user profile called AUTO. The AUTO profile selects the best profile from VERY LOW, LOW, NORMAL, HIGH, on the basis of the current battery level and fixed thresholds. Experimental results show that *gLCB* obtained the best trade-off between numbers of server uploads and battery lifetime with the policies computed automatically by the device as shown in TABLE I. Although the information about the remaining battery is poor, the empirical results show good self-adaptation behaviour driven by energy information applied to our prototype.

TABLE I. EXPERIMENTAL RESULTS

PROFILE	UPDATE PER HOUR	MEAN BATTERY DURATION	DIFF FROM VERY LOW MEAN BATTERY LEVEL [%]
VERY LOW	1,25	13h 8min 8 sec	-
LOW	1,32	12h 7min 35 sec	7,67%
NORMAL	6,35	9h 13min 6 sec	28,44%
HIGH	8,48	7h 55min 55 sec	30,05%
AUTO	**7,54**	**11h 41min 33 sec**	**4,55%**

V. CONTRIBUTIONS

Managing energy consumption data in real-time by and exploiting it to drive self-adaptation in mobile applications is the main contribution of this research. This technique will enable mobile developers to be more aware of the energy efficiency of their code. Moreover it will improve the battery life of mobile phones providing a better user experience.

REFERENCES

[1] R. Van der Meulen, Gartner says 821 million smart devices will be purchased worldwide, in 2012, Sales to Rise to 1.2 Billion in 2013, url: http://www.gartner.com/it/page.jsp?id=2227215, Last Visited 15th December 2012.

[2] R.J.G. Simons, A. Pras, “The Hidden Energy Cost of Web Advertising,” In: Proceedings of the 12th Twente Student Conference on Information Technology, pp.1-8, 2010

[3] Mejia Bernal J.F., Ardito L. , Morisio M. , Falcarin P., "Towards an Efficient Context-Aware System: Problems and Suggestions to Reduce Energy Consumption in Mobile Devices," In: 9th International Conference on Mobile Business and the 9th Global Mobility Roundtable (ICMB/GMR 2010), pp. 510-514, 2010

[4] Procaccianti G., Vetro' A., Ardito L., Morisio M., "Profiling Power Consumption on Desktop Computer Systems," In: LECTURE NOTES IN COMPUTER SCIENCE, vol. 6868, pp. 110-123, ISSN 0302-9743, 2011

[5] Procaccianti G., Ardito L., Vetro' A., Morisio M., "Energy Efficiency in the ICT," In: Energy Efficiency - The Innovative Ways for Smart Energy, the Future Towards Modern Utilities, pp. 353-372, ISBN 9789535108009, 2012

[6] A.Shye, B. Scholbrock, 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* (MICRO 42), pp. 1-11, 2009

[7] N. Balasubramanian, A. Balasubramanian, 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* (IMC '09), pp. 1-14, 2009

[8] C. Yoon, D. Kim, W. Jung, C. Kang, H. Cha, “AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring,” In USENIX Annual Technical Conference ‘12, pp.1-14, 2012

[9] R. Murmuria, J. Medsger, A. Stavrou, J.M. Voas, “Mobile Application and Device Power Usage Measurements,” *Software Security and Reliability (SERE), IEEE Sixth International Conference on* , pp.147-156, 2012

[10] A. Pathak, Y.C. Hu, M. Zhang, P. Bahl, Y. Wang, “Fine-grained power modeling for smartphones using system call tracing,” In *Proceedings of the sixth conference on Computer systems* (EuroSys '11), pp. 1-15, 2011

[11] A. Pathak, Y. C. Hu, M. Zhang, “Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices,” In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks* (HotNets-X), pp. 1-6, 2011

[12] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, “Profiling resource usage for mobile applications: a cross-layer approach,” In *Proceedings of the 9th international conference on Mobile systems, applications, and services* (MobiSys '11), pp.321-334, 2011

[13] A. Abogharaf, R. Palit, K. Naik, A. Singh, “A methodology for energy performance testing of smartphone applications,” *Automation of Software Test (AST), 2012 7th International Workshop on* , pp.110-116, 2012

[14] J. Peddersen, S. Parameswaran, "Energy Driven Application SelfAdaptation," VLSI Design, International Conference on, 20th International Conference on VLSI Design (VLSID'07), pp. 385-390, 2007

[15] C. Wohlin, “Experimentation in software engineering: an introduction,” ser. Kluwer international series in software engineering. Kluwer Academic, 2000.

[16] Forrest Shull, Janice Singer, Dag I. K. Sjøberg, “Guide to advanced empirical software engineering,” Springer, 2007

[17] L.Ardito, M.Torchiano, M. Marengo, P. Falcarin, gLCB: An Energy Aware Context Broker, Sustainable Computing: Informatics and Systems, vol. 3, ISSN 2210-5379, pp 18-26, 2013