STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

DEVELOPMENT GUIDELINES FOR OPTIMIZING THE ENERGY CONSUMPTION OF MOBILE APPLICATIONS

DIANA C. ZOICAŞ

Abstract. The market of mobile devices and the power of mobile computation has increased significantly over the last years. Although the technology has evolved a lot the main issue of mobile devices is that they are and will remain severely limited by their battery life. The need to preserve this critical resource has driven mobile devices operating systems to take into consideration the power management and has driven the developers of mobile applications to optimize the energy consumption of the applications.The two main fields of research in this area are finding solutions to estimate the energy consumption of an application and finding ways to determine applications and bugs that lead to energy consumption and unexpected battery drain.

In this paper we show how we use development guidelines for mobile applications in order to determine the pieces of code that could generate a bug and could lead to an abnormal battery drain. We analyze the impact generated by the wrong usage or the lack of usage of certain development guidelines on the energy consumption. We show how the development guidelines and the best practices can be used to ensure that a mobile application is more efficient, has a better performance and consumes less energy.

1. Introduction

The market of mobile devices has increased significantly over the last years. More and more people buy mobile devices due to their usefulness and their portability. The growth of the market has also lead to an explosion of the power of mobile computation. Despite of the increased power of mobile computation the main issue of mobile devices is that they are and will remain limited by their battery life. The increased battery drain which is also called

Received by the editors: May 1, 2014.

2010 *Mathematics Subject Classification.* 68N30, 68N19.

1998 *CR Categories and Descriptors.* D.2.4 [**Software**]: Software Engineering – *Software/Program Verification*; D.2.10 [**Software**]: Software Engineering - *Design*.

Key words and phrases. mobile application, energy, energy optimization.

energy anomaly frustrates users, creates poor press for vendors and can make mobile devices unusable [8].

The construction of the batteries of the mobile devices was done mainly taking into consideration the physical size of the battery in order for the devices to be as small and light as possible and the battery capacity was not that important. The construction of the batteries has evolved a lot from the NiCD(Nickel Cadmium) bateries that were used in the 80's and 90's and were very heavy and big to the NiMH(Nickel Metal Hybride) batteries that were used in the late 90's, the Lithium Ion batteries that are still used today and to the new Li-Poly(Lithium Poly Ion) batteries which are not yet widely used but are lighter and more energy efficient than the other ones used before. The evolution of the techonlogies used for constructing the batteries is not enough for the battery life to be better so the improvement of the battery life is intended to be done in two different ways: software and processor to be less power-hungry and the devices to be constructed in order to provide the essentials and no more than that. Intel is trying to come back in the mobile world with a power-saving technology that will be constructed on a platform that is already available for the light laptops.

The need to preserve energy has driven mobile devices operating systems concentrate a lot at the power management. The two main fields of research in this area are the implementation of tools and techniques to estimate the energy consumption of different applications and the implementation of tools and techniques to discover applications and bugs that lead to energy consumption and unexpected battery drain. A research field that is still under development is the implementation of tools that would identify the code that could be improved regarding energy consumption and provide solutions for optimizing the code.

As a first step in the process of developing such a tool in this paper we are identifying the development guidelines that need to be followed in order for the mobile applications to do not generate energy bugs. The energy bugs are defined as being an error in the system, either application, operating system, hardware, firmware or external that causes an unexpected amount of high energy consumption by the system as a whole [1]. We are analyzing some of the development guidelines that can optimize the energy consumption of a mobile application. We are focusing on four types of guides: general development guidelines, data manipulation guidelines, performance guidelines and background jobs guidelines. We are identifying the reason for which these guidelines can improve the energy consumption.

2. Background

It happens sometimes that the battery of a mobile device drains very fast. This drain is caused by energy leaks which can have two causes: hardware bugs or software bugs.

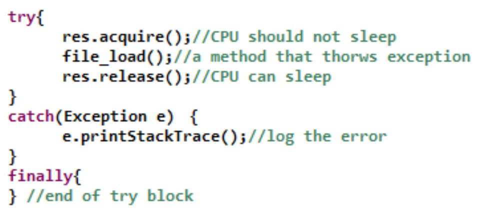
The hardware bugs can be caused by any defect of a hardware part of the mobile device. Hardware bugs are bugs that are not related to the implementation and are caused by some hardware components. The first hardware bug is related to a faulty battery. This bug is mainly solved by replacing the battery. Another hardware energy bug is related to the exterior hardware damage. The SIM card can also cause battery drain in multiple ways. An external SDCard can also trigger severe battery drain. Another generator of energy bugs is also an external hardware (phone chargers, external docks used for recharging or for audio capabilities)

The software bugs can be generated by the operating system that is installed on the mobile device, by a certain application or by a programming mistake.The first category of software bugs are the operating system (Android, iOS, Windows Mobile) bugs that are generated by an update that was done by the user or by an automatically done update. In most of the cases the solution for this type of bugs is to do a downgrade of the operating system. The second category of software bugs are the application and framework energy bugs and the most known bugs of this type are the No-Sleep Bugs [6]. The application and framework bugs are the bugs that are generated by the implementation of an application or are generated by the framework that is used when developing a mobile application. The root cause of these bugs can be anything from a simple implementation error to complicated reasons like race condition that prevent the lock release. Another known energy bug is the loop bug. In this case a part of an application enters a looping state and performs periodically unnecessary tasks. The third sub-category is the immortality bug. The behavior of the application is the following: it is killed and it restarts. The third category of energy bug are the energy bugs triggered by External Conditions [1]. One of the external conditions that influence the battery drain is the Wireless Signal Strength [1], Wireless Handovers [1]. Besides the above energy bugs there are also unknown bugs that were reported but for which the root cause is unknown. There are a few tools that help the user to narrow down to suspicious application that generates the energy bugs.

There are some software bugs that are widely known for different operating system of the mobile devices. For Android there are two main types of energy bugs: no-sleep bugs and loop bugs. A no sleep bug occurs when the CPU is waken up by an application but it is never put back to sleep therefore excesively consuming the energy without providing any functionality [5]. The loop bugs are ocurring when a thread is waiting for a certain event in order to continue and a variable is used when testing the condition. The thread will continuously poll the variable until it is changed and therefore it consumes CPU without doing any work for the user [5]. The power models of Android and Windows are similar so also the energy bugs from Windows are similar to the ones from Android. In the iOS system an application can be only in one of the four states that are definded and the handling of the states is fully done by the developers [7] so the energy bugs can be caused by the prolonging of the transition between two states and can be caused only by the developer.

There are more types of *No-Sleep Bug* and three of these types are *No-Sleep Code Paths*, *No-Sleep Race Condition* and *No-sleep dilation* [2].

No-Sleep Code Path (Figure 1): The root cause of most of the bugs is the existence of a code path in the application that wakes up a component but does not put the component back to sleep. The first cause is that the programmer forgot to do the release through the code or he has put it on a conditional path but not on all the paths. Another cause is that the programmer did put the release code but the code took an unanticipated code path during execution and the release was not executed.

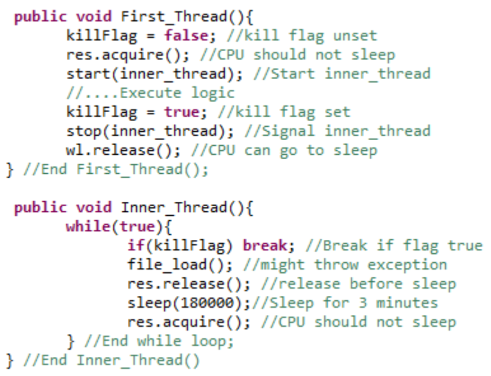


```
try{
        res.acquire();//CPU should not sleep
        file_load();//a method that thorws exception
        res.release();//CPU can sleep
}
catch(Exception e)  {
        e.printStackTrace();//log the error
}
finally{
} //end of try block
```

FIGURE 1. No-Sleep bugs: code paths

No-Sleep Race Condition: These bugs were caused by race conditions in multi-threaded applications. The power management of a particular component was carried out by different threads in the application (one thread switches the component on and later another thread should switch it off) and the sequence of execution was a different one that expected.

No-Sleep Dilation: the component woken up by the application is ultimately put to sleep by the application but only after a substantially longer period of time than expected or necessary.



```
public void First_Thread(){
        killFlag = false; //kill flag unset
        res.acquire(); //CPU should not sleep
        start(inner_thread); //Start inner_thread
        //....Execute logic
        killFlag = true; //kill flag set
        stop(inner_thread); //Signal inner_thread
        wl.release(); //CPU can go to sleep
} //End First_Thread();

public void Inner_Thread(){
        while(true){
                if(killFlag) break; //Break if flag true
                file_load(); //might throw exception
                res.release(); //release before sleep
                sleep(180000);//Sleep for 3 minutes
                res.acquire(); //CPU should not sleep
        } //End while loop;
} //End Inner_Thread()
```

FIGURE 2. No-Sleep bugs: race condition

In order to develop an application that is energy efficient it is not enough to have an application that does not generate energy bugs. We have to pay attention to the implementation that could be enhanced from the energy point of view. From the previous papers we have seen that most of the research was concentrated on implementing tools that analyze the code for energy leaks and that find energy bugs but none of these tools offer also guidelines for the developer on how to fix the issues that were discovered.

3. Main contribution

Most of the papers related to optimization of the mobile application in order to reduce the energy consumptions are describing techniques and tools that help the developers to determine the source code that generates energy leaks. These tools (ADEL, Carat, eDoctor) help developers to discover the root cause of the error but they do not offer guidelines on how the developers should improve the code. Nowadays the developers can find solutions for optimizing the energy consumption by searching on the web for an optimal solution, by consulting the specialized forums or by using the previous experience of the developer. From our point of view it is more important to implement the mobile application in such a manner that the energy leaks are reduced to minimum. It is cheaper to write correct and efficient code from the implementation phase than to re-write the code for optimizing it to reduce the energy consumption. This is the reason for which we consider that it is important to determine development guidelines that can be used by the developers in order to write correct and efficient mobile applications.Taking this in consideration, a research field that is still under development is the implementation of tools that would identify the code that could be improved regarding energy consumption and provide solutions for optimizing the code.

As a first step in the process of developing such a tool in this paper we identify the development guidelines that need to be followed in order for the mobile applications to do not generate energy bugs. We are analyzing some of the development guidelines that can optimize the energy consumption of a mobile application. We first try to identify from development guidelines provided by the different mobile devices operating systems vendors the fields that could be improved in order for the energy consumption to be reduced to a minimum. Due to the fact that the Android operating system is the most used one and the API system is presented more clear, we have tried to focus our research in this area but keeping also in mind the other operating systems for mobile devices. The identifcation of the development guidelines to be used for optimizing the energy consuption we have only taken into consideration the API that is exposed by the operating systems.We have identified different types of development guidelines that can influence the energy consumption of mobile applications: general development guidelines, data manipulation guidelines, performance guidelines and background jobs guidelines. In the future papers we wil start the implementation of the tool with the automatic identification of the development guideline that we discovered in this paper. This identification will be done for each of the discovered development guidelines.

General development guidelines. The first guidelines that should be taken into consideration when developing a mobile application are the ones for avoiding the coding errors. First of all it is important that the code of the application does not produce energy bugs. For each of the No-Sleep bugs there is a guideline to be used in order to avoid the respective bug. As mentioned in the above sections, the most important energy bugs are the No-Sleep bugs.

The root cause of the No-Sleep Code Path energy bug, as it can be seen in Figure 1, is that the resources that are being used are not released on all the paths so the CPU cannot go back to sleep and it continues to consume energy. This bug can be avoided by verifying that the code for releasing the resources that are being used is present in all of the paths of the method that acquires the respective resource. In the case of Figure 1 the solution would be to add the code for releasing the resources in the *finally* block.

The No-Sleep Race Condition energy bug can be present in all the multi-thread mobile applications. As it can be seen in Figure 2 , the problem occurs when the handling of the power management for a certain component is done in different threads. In a normal execution path of the threads the component would be acquired and released as expected but in some exceptional cases the path could be different and the resource will not be released. In the case of No- Sleep Race Condition the developer should pay special attention to the execution path for the threads [6] and make sure that the components that were acquired are released no mather the order in which the threads get executed.

Data manipulation guidelines. One of the important aspects regarding the functionality and energy consumption is related to data manipulation. The energy consumed by operations for data manipulation depends on the number of calls that are done for sending/saving data and on the size of the data that is being manipulated. In paper [3] it was determined that it is more efficient sending larger files than sending smaller files. Due to these findings, a guideline that would optimize the energy consumption would be to gather smaller data files into one bigger file and send once as much data as possible. This guideline could also be applied to the HTTP requests. It is recommended to bundle multiple small requests into one bigger request in order for the request to be more energy efficient. For verifying that more data is sent at once and not in smaller chuncks we could check that all the calls for saving the data are gathered in only one call.

Performance guidelines. The performance of an application is directly proportional to the energy consumed by the respective application. If the application runs faster it means that the resources are released earlier and the CPU can enter the sleep state earlier. The code that has a better performance is consuming in some of the cases more resources than a code which has not such a good performance. Despite of this fact it was measured that the energy consumption of a code with a better performance consumes less energy [3]. There are a lot of development guidelines for improving the performance of an application. The first and most important guideline is to do not create unnecessary objects. Each of the instantiated objects needs a garbage collection that consumes energy when it runs so we should check in our tool that if an object is declared, it is also used in the methods in which it was declared. We should also check the global variables to be used in the code of the application. Another important development guideline, at least on the Android OS, is to rather use the fields of a class directly than by calling the getters and the setters [4]. In Android it is much more expensive to call the getters and setters than the fields directly. When testing both the setters and getters and the direct access methods it results that the energy consumption decreases by 30% when accessing directly the fields and not via the getters and setters. We will check in our tool that, when using a certain variable we are accessing it directly and not via the getters and setters.

Background jobs guidelines. One of the energy consuming tasks within an application are the background jobs and the optimization for these jobs is really important. For optimizing the energy consumption of the background jobs the most important development guideline that should be followed is to group more background jobs in order to run at the same time. In this way the resources are acquired at the same time and are released at the same time and allow the CPU to be in a sleep state for a longer period. Another development guideline that should be taken in consideration when implementing background jobs is that data for the background jobs should be acquired though an asynchronous call. In this manner the application will not wait for a response and it will let at least the screen in a sleep mode.

4. Conclusions and Further Work

The users of the mobile devices tend to reject the applications that are generating energy leaks in order to benefit as much as possible of the usability of their mobile devices. Due to this new requirement the investment in finding solutions for optimizing the energy consumption became a very important research topic. Most of the research that is done for the optimizing the energy consumption is oriented towards techniques that try to identify where the energy is lost and which is the amount of the energy that is lost.

From our point of view the research should be oriented towards identifying the development guidelines and best practices that can lead to the development of energy-efficient applications. In this paper we have identified some of the development guidelines that help mobile application developers to write applications that do not produce energy bug. We have also identified some development guidelines that help mobile application developers to implement applications that will be more energy-efficient. For future work we will investigate more development guidelines that could lead to an improvement of energy consumption of mobile application. We will also focus more on the development guidelines for Blackberry OS, iOS or Windows Phone OS. We also would like to implement a tool to check that the implementation of a mobile application complies to the development guidelines that should be used for the energy consumption optimization. We would also like for this tool to offer solutions for improving the pieces of code that are not energy-efficient.

References

[1] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, *Bootstrapping Energy Debugging on Smartphones: A First Look at Energy Bugs in Mobile Devices*, Proceedings of the 10th ACM Workshop on Hot Topics in Networks, New York, USA, 2011, Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.232.9209

[2] Abhinav Pathak, Abhilash Jindal, Y. Charlie Hu and Samuel P. Midkiff, *What is keeping my phone awake? Characterizing and Detecting No-Sleep Energy Bugs in Smartphone Apps*, Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, Lombard, IL, USA, April 2013, Available: https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final198.pdf

[3] Ding Li, William G. J. Halfond, *An investigation into Energy-Saving Programming Practices for Android Smartphone App Development*, Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS), Hyderabad, India June 2014, Available: http://www-bcf.usc.edu/ halfond/papers/li14greens.pdf

[4] Ding Li, William G. J. Halfond, *An Investigation into Energy-Saving Programming Practices for Android Smartphone App Development*, Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS), Los Angeles, USA June 2014, Available: http://www-bcf.usc.edu/ halfond/papers/li14greens.pdf

[5] Jack Zhang, Ayemi Musa, Wei Le, *A Comparison of Energy Bugs for Smartphone Platforms*, 1st International Workshop on the Engineering of Mobile-Enabled Systems, San Francisco, USA 2013, Available: http://www.cs.iastate.edu/ weile/docs/le-mobs13-1.pdf

[6] Panagiotis Vekris, Ranjit Jhala, Sorin Lerner and Yuvraj Agarwal, *Towards Verifying Android Apps for the Absence of No-Sleep Energy Bugs*, Proceedings of 2012 Workshop

on Power-Aware Computing and systemsHotPower 2012, Hollywood, CA, Available: http://dl.acm.org/citation.cfm?id=2387872

[7] Shuai Hao, Ding Li, William G. J. Halfond, Ramesh Govinda, *Estimating Mobile Application Energy Consumption using Program Analysis*, Proceedings of the 2013 International Conference on Software Engineering, Pages 92-101, Piscataway, NJ, 2013, Available: http://www.cs.binghamton.edu/ millerti/cs680r/papers/EstimatingMobileApplication Energy.pdf

[8] Xiao Ma, Peng Huang, Xinxin Jin, Pei Wang, Soyeon Park, Dongcai Shen, Yuanyuan Zhou, Lawrence K. Saul and Geoffrey M. Voelker, *eDoctor: Automatically Diagnosing Abnormal Battery Drain Issues on Smartphone*, Proceedings of the 10th ACM/USENIX Symposium on Networked Systems Design and Implementation, Lombard, IL, April 2013, Available: http://cseweb.ucsd.edu/ voelker/pubs/edoctor-nsdi13.pdf

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: `diana.zoicas@cs.ubbcluj.ro`