Power Guru: Implementing Smart Power Management on the Android Platform

Hans Fredrik Unelsroed
Computer Science
UCSB
Email: hfu@cs.ucsb.edu

Per Christian Roeine
Computer Science
UCSB
Email: roeine@cs.ucsb.edu

Fahad Ghani
Computer Science
UCSB
Email: fahad@cs.ucsb.edu

***Abstract*—Improving battery life in mobile devices has become a very important area of research. Hardware components like the CPU, screens and radios has evolved quickly while battery technology has lacked the same progression. Our goal is to implement a smart power management system for the Android OS that makes multi-tasking more battery friendly. We do this by looking at the currently running applications and giving the user suggestions on which applications to kill in order to optimize the battery life of the device.**

I. Introduction

Mobile devices have progressively become more powerful with power consuming technologies like GPS, WiFi, 3G and 3GS now inherent to them. Furthermore, more power hungry applications, for example video streaming applications like YouTube, are being built and used frequently by mobile phone users thereby putting more stress on battery life. However, the battery technology has not been able to keep up with this rapid progress and there is a need for mechanisms that manage the usage of battery in smart ways to improve battery life, at least till the battery technology catches up.

Power Guru is an application for the Android 1.6 Platform that attempts to optimize the battery life of the device it is running on. It does this by asking the user which applications he/she wants to prioritize and then makes suggestions on which applications should be killed in order to maximize the remaining battery life. This idea is novel compared to previous work done in the field because it gives the user suggestions, other solutions that deal with task management often give the user the information needed to make a decisions, but doesn't use that information in a way that automates a lot of the decision making.

The Android OS claims to suspend/kill unused background processes, but many users are complaining that the OS is not doing a good enough job (we don't have a reference for this, but there is lots of anecdotal evidence to suggest this is true). After heavy usage, Android devices often become sluggish because of unnecessary background processes running, this also causes drains the battery. That is why an application like Power Guru is so important for for the Android OS. Balancing the trade-off between giving the user the ability to run as many applications in the background as he/she wishes and conserving resources is hard on mobile devices with very limited resources.[5]

In this paper we will present a solution we believe is more user-friendly than existing solutions, while still having comparative performance (direct comparisons of performance is hard since traditional task managers depend on how the users uses them). We will also show how application usage affects the battery life and how Power Guru helps reduce the number of unnecessary running applications.

The rest of this paper will be structured as follows. In Section 2 we will look at the motivation for doing power management and look at previous work done in the field. Section 3 gives an insight into how our solution works. And Section 4 shows how well our solution is performing by doing quantitative evaluation.

II. Background

Multi-tasking on a mobile device can be solved in several different ways. The Apple iPhone OS only supports multi-tasking of a few of the applications that Apple has provided with the OS, no other applications are allowed to run in the background. This has enabled the iPhone to achieve better battery life and less crashes/sluggishness than competing devices. Palm's WebOS supports multi-tasking of all applications and gives the user a graphical way to view all currently running applications[7]. This makes it easy for the user to visualize which applications are running, this in turn makes it easier for the user to decide which applications to kill. Android OS allows multi-tasking in the same way that WebOS does, but does not include a graphical way for the user to view/kill running applications. Google claims this is not needed because the OS suspends/kills applications when they are no longer necessary, anecdotal evidence suggests this is not the case. Many users report that battery life is low and the phone often becomes sluggish after having been used for a few days. Some users have solved this problem by installing a task manager that enables them to manually decide which applications to kill. This solution is not optimal for any user, especially for novice users who might not recognize each application running by it's name. Maybe write more details about how Android OS does power management

Most of the research done on power management has focused on hardware solutions and software that turns off hardware features (for example turning off the HDD, screen, etc.)[10][9][8][11][12][13][14][15]. Previous research has ex-

plored ideas at multiple levels of a phone's usage: at the OS level, MAC layer, network layer (the non-application layers) and application layer. The solutions that are in the non application layer are very unobtrusive to the user, they tend to be fine grained and use low level optimizations. Examples of OS level power management are optimizing the use of each CPU cycle, optimizing memory usage and can even include turning down screen brightness after a set amount of idle time. For example, CoolSpots [3] focuses on dynamically (based on bandwidth requirement) switching between bluetooth and wifi to conserve power, whereas the authors of the paper 'Integrated power management for video streaming to mobile handheld devices'[9], focus on integrating several optimizations at the OS level and middleware techniques. There have also, as previously mentioned, been a lot of work at the hardware level, for example reducing the power consumed by the WiFi modes (eg. Wifi PSM) and the bluetooth adapter (eg. BlueCore3).

Similarly, there have also been several applications developed lately which focus on power management on these devices. Such solutions tend to be more coarse and use more high level information. For example one such application made for iPhone is 'Battery Magic'[6], which just provides the user the time left for a fixed number of activities that they can do (like game play). There are such applications for Android platforms too: 'Battery Booster'[1], which focuses on managing the hardware features by autonomously turning on/off 3G, WiFi, etc. However, it doesn't deal with process level management at all. There are process managing applications available too, for example ProcessManager[2]. Similar examples are: Task Panel, Advanced Task Killer, Task Manager and Fpt System Manager. All these applications more or less list all currently running applications with information about their memory consumption and give the user the ability to kill or suspend applications. But these applications contain no autonomous decision making or suggestions on which applications would be the best to kill in order to increase battery life. There are also applications that can modify CPU settings, this can be used to improve battery life by under-clocking the CPU, making the peak power consumption lower. But this solution naturally has some drawbacks, such as decreased peak performance.

III. Implementation

The main screen, shown Figure 1, is presented to the user. Running applications are displayed in groups according to their power rating, making it easy for the user to make informative decisions of which applications he/she really wants to prioritize. The remaining battery time with the current load is also displayed. When the user is done prioritizing, Power Guru estimates what the battery time would be if certain applications are killed, and presents these applications and the new, potential remaining battery time. The user may now choose to kill the suggested applications or go back to make changes.

Our application is implemented on Google Android Dev phone G1, which runs the Android 1.6 firmware and is one

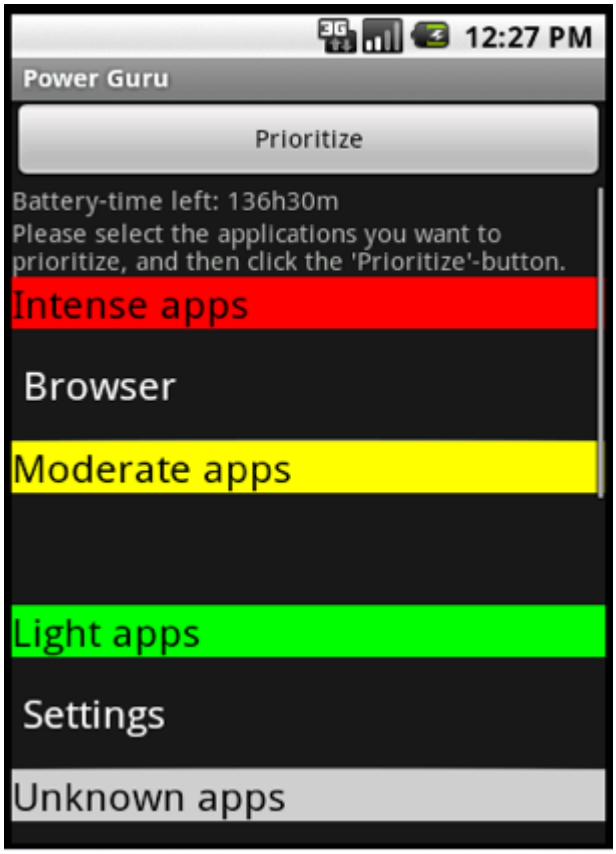

Fig. 1. Power Guru main screen

of the first android powered mobile phones. To develop our application we are using the public APIs exposed by Google. These APIs expose services for getting information about the battery status of the device, but unfortunately the API does not provide any information about the CPU, screen, radio or other resource usage.

Power Guru consists of three parts, GUI, a data collector and the algorithms. GUI: We tried to make the GUI as simple and user-friendly as possible. The applications are divided into groups, based on their power rating, which is represented by lists. This makes it easy for the user to get an overview over how power hungry the applications are. As prioritized applications are selected, they are dynamically moved to a list for prioritized applications for the user to review and unprioritize if necessary. During the whole process, remaining battery time is displayed in top of the screen. Data Collector: The data needed for Power Guru to work is battery status, CPU usage and preferably information about radio, screen and other hardware usage. Our current implementation only collects battery and CPU info. In order to get the battery information we register our application with the OS as a receiver of "ACTION_BATTERY_CHANGED" notifications, this means that every time the battery status changes (for instance from 50% to 49%) we receive a notification with the updated information. Getting the CPU information was not as easy, there is no OS notification support for CPU usage

of an individual application. Our solution to this problem was a background service that parses the ”/proc/1234/stat” file, the ”stat” file is a standard UNIX construct that contains information about all running processes, including how much CPU time each process has used. This background service is currently set to run every minute, this results in a roughly 3% decrease in overall battery time for the device.

Algorithms: There are two main algorithms in Power Guru, the battery estimation algorithm and the application to kill suggestions algorithm. Battery estimation is done by assuming a fully charged device with no load will last for 300 hours (which is the standby time HTC provides for the device), we subtract time from that estimate as the CPU load on the system goes up. This linear correlation between battery time and CPU load is inaccurate, firstly there are other things that impact the battery life than the CPU and secondly the battery might not discharge linearly. But what this estimate does give us is a view of how much the battery life improves when we look at which applications to kill, to decide which applications should be killed in order to conserve the battery we have another algorithm. This application-suggestion algorithm works by looking at the current battery life, it then selects the most power hungry application currently running and estimates what the battery life would be if this application was killed, we then repeat this until the additional improvement in battery life by killing another application is less than 5%.

IV. Evaluation

In this section we present the results of running some battery tests with and without the use of Power Guru. Here we attempt to find out how battery life is affected in different scenarios and why power managers are needed. Next we demonstrate how the use of Power Guru changes the battery usage pattern for the good. Specifically there are several goals of our evaluation, mainly: evaluating the effect of number of running applications and the type of those applications on battery life. We also present results which prove that CPU time utilized by an application is a fair indicator of the amount of battery used by that application, with some exceptions, which will be discussed later. Finally we present the battery usage pattern with Power Guru running, showing how well it identifies the power hungry applications and how is the battery life affected after the suggested applications are killed.

A. Issues with testing

The metrics used in our evaluation are quite simple and intuitive: Battery discharge and Total CPU time used by an application. However, there are several issues associated with using battery time as a metric. First of all, it is a time consuming effort with the large amount of time it takes to drain the battery once and run a new test. Secondly, the emulator does not have any support for battery, so an Android device is needed to perform battery related tests. Thirdly, battery discharge is not a very accurate indicator of how the battery is being used, mainly because of the inconsistent testing conditions: The display might be kept on much longer

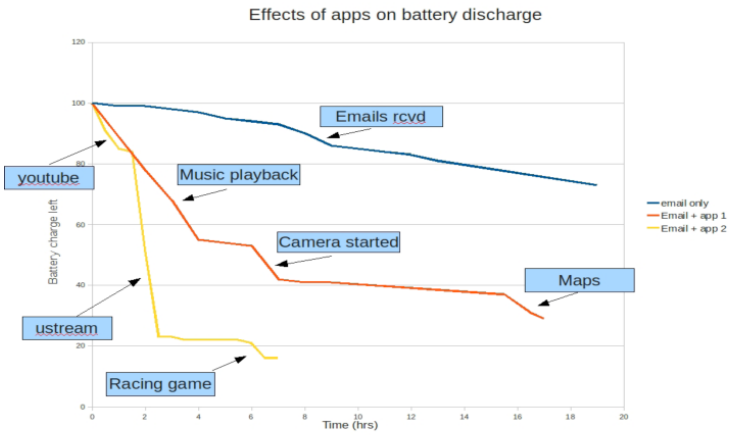

Fig. 2. Results for Cases 1 to 3

in some cases than others, incoming email (which contributes to battery consumption a lot) is not consistent, with a period of email burst following a period of no emails. Also, the phone falls into idle state on an irregular basis depending upon the applications that run (or do not run) in the background. The idle state reduces the power consumption of the phone to the minimal level. Lastly, these tests are not representative of the actual usage of a phone by real users since battery usage pattern changes tremendously with phone calls, which we are not evaluating in our tests.

B. Testing conditions

All the tests have been performed on Android OS 1.6 on an HTC G1 phone (google dev phone). In order to gather data, the in-built Settings application has been used. Since it is an in-built application, it is more reliable than any third party applications. Secondly, using a third party application will probably draw more power itself and hence may give some inaccurate results. However, there are certain drawbacks too while using Settings app: not all the applications are shown in the list, for example, both Camera and Music applications are clubbed under a single name of ’Media Server’. However, this drawback can be worked around easily. WiFi was kept turned on all the time during the evaluation so that the applications that need internet connection may run in the background (like email). Lastly, in our results we use a category called ’Others’ which includes applications like Display, Core apps, Android OS, cell standby, etc. which any power manager cannot kill and thus are not relevant from the point of view of our discussion. For the evaluation purposes, following test cases were run: Case 1: Email is the only running application Case 2: Email and some other application (Camera, Music and Google maps) Case 3: Email and some other application (YouTube, UStream and Gaming) Case 4: Email and some other application along with Power Guru

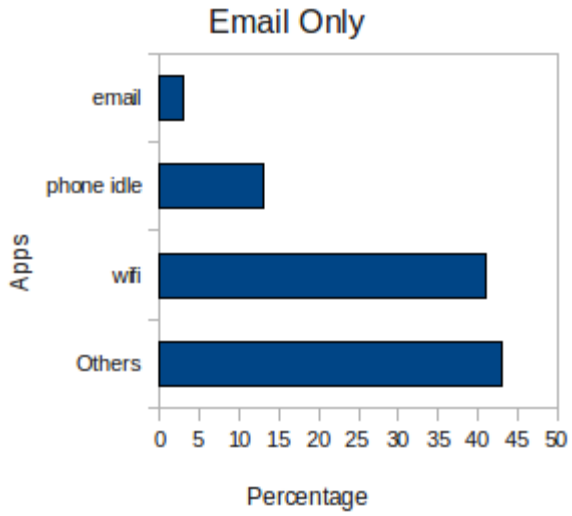

Fig. 3.

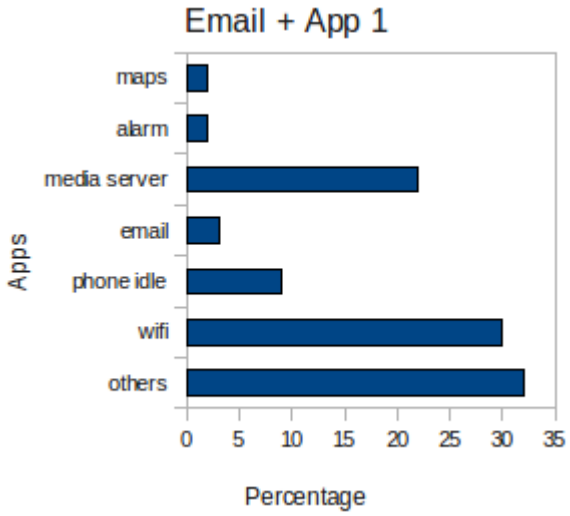

Fig. 4.

C. Results

Regarding fig 2, the Email application running in all three cases is configured to check for emails every 5 minutes (this is done to keep the phone from falling into idle state as often as possible). Clearly, multimedia applications are the most power hungry applications that can eat the battery in no time. It took roughly 14 hours for the battery to fall a level of 20% with only Email running (Case 1). However, with music playback, the battery level fell about 45% in a period of about 4 hours and similar pattern was observed when running the camera application (Case 2). Ustream was even worse with battery level falling from 85% to 23% in a matter of just 30 minutes! Ustream is a live streaming application where users can stream live from the phone to the Ustream server. Thus it involves both a lot of graphics usage (live video) and network usage. The application uploaded about 67MB of data in the 30 minutes of running. The graphics intense car racing game had a similar battery footprint. Clearly, multimedia applications, as expected, are the most power hungry.

The results can be corroborated by the percentage of battery consumed by each application as depicted by the settings app. Figures 3 to 5 show the contribution of each application at the end of each test case run. In the first test case, Email application takes up only 3% of the total battery eaten up till the end of the test case. Furthermore, the phone's idle state consumed about 13% of the battery. Clearly, email is not a very resource intense app. On the other hand however, in Case 2, media server (music and camera apps) used up a lot of battery power (23%) and correspondingly, the idle state of the phone came down to 8% (from 13%). Similar results are seen in Case 3, were Ustream and a Car racing game combined together eat roughly 20% of the battery. The idle state of the phone comes down to roughly 4%! This is in accordance with our earlier conclusions that multimedia apps are the most power hungry.

Clearly, with such plethora of applications around and applications drawing out such power from the battery, it is

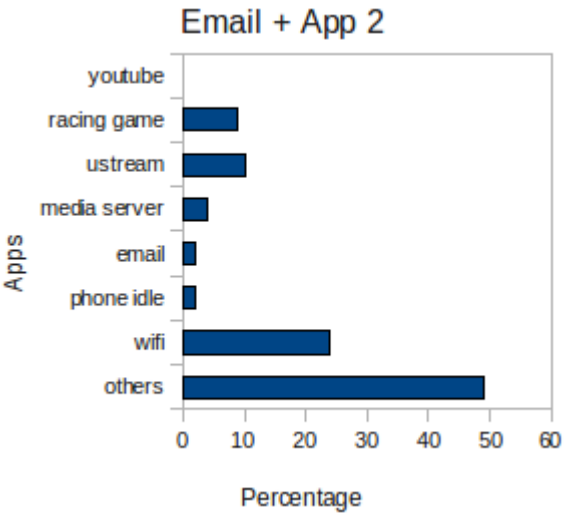

Fig. 5.

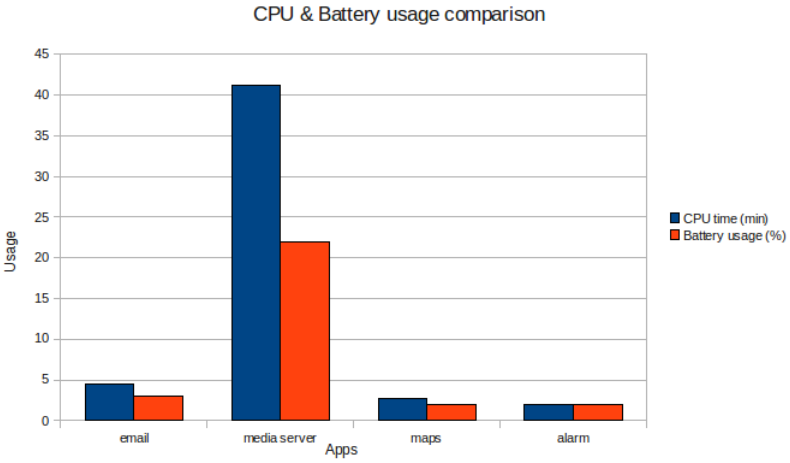

Fig. 6.

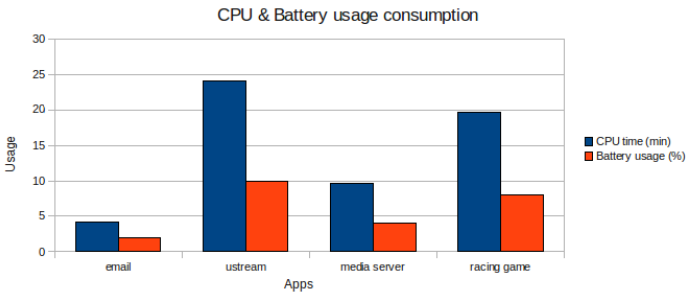

Fig. 7.

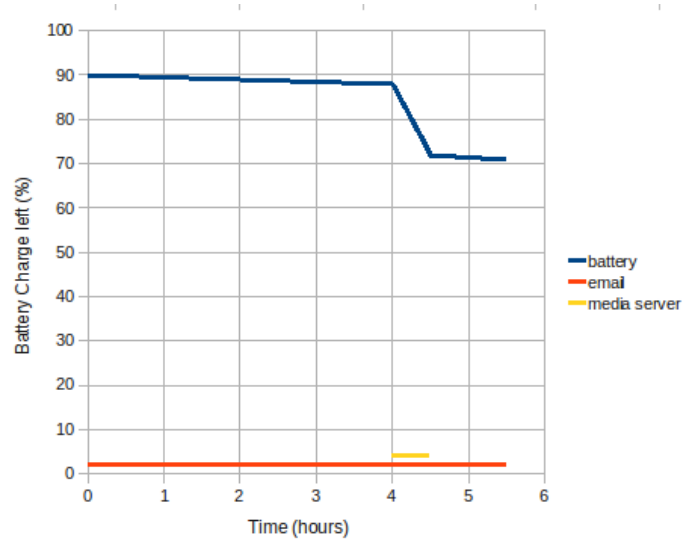

Fig. 8.

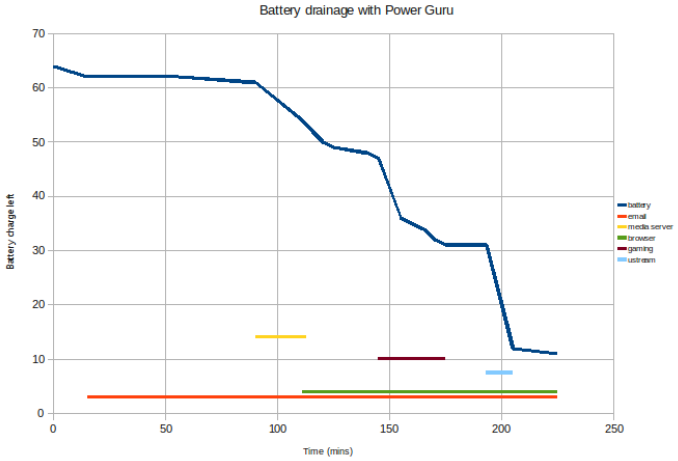

Fig. 9.

important to use a power manager which can find out which applications are hurting the battery the most and help the user to prolong their battery life. In Power Guru, we used CPU time consumed by an application as a factor to determine how resource intensive an application is. It was therefore important to find out how well correlated the CPU time and amount of battery consumed are. Figures 6 and 7 shows the CPU time taken by the applications (in Case 2 and 3) along with the percentage of battery consumed by those applications. The results clearly show that the two are closely related and therefore CPU time is a good indicator of the intensiveness of an application. However, YouTube did not feature in the list of applications in the settings app which initially led us to wrongly conclude that the application YouTube itself was not eating up much resources, but the Display instead was consuming it. However, after running a set of tests again, we found out that YouTube was included in the Media Server application and indeed did bring the power levels down considerably in a matter of less than 20 minutes (Figure 8).

Thus, our results prove that there is a need for a power manager to effectively manage the battery power of a mobile device running on Android OS. We also prove that total CPU time is a good measure to find out how much power an app consumes. Next we show the results of running PowerGuru on the device. Figure 9 shows the battery discharge pattern with the same apps running. First, we ran the Camera application (85th minute), and after the app had been running till the 114th minute, PowerGuru suggested that the app be killed and upon killing the app, the battery fall was arrested. Similarly, after the gaming app was running for 35 minutes, it came up in PowerGuru's suggested kill list and was then killed. Lastly, Ustream was killed after running for less than 25 minutes. Notice that since Ustream consumed battery most rapidly, it came up in the list quicker. Furthermore, the browser application which is consuming quite less battery, is never suggested to be killed. Hence, our results clearly show that PowerGuru performs well and rightly points out the applications that need to be killed.

V. Conclusion

In this paper we have presented a smart power management solution for the Android platform. Our solution, called Power Guru, is successful in giving the user suggestions on which applications needs to be killed.

Preliminary results suggest that proactively using a task manager on an Android device leads to less power usage. Power Guru makes task management easier to do for novice users, by having them select the applications they know they want to use instead of asking them to select all the applications they don't want to use. One drawback with this approach is that some users might not know the name of the application they want to use, in that case implementing a graphical icon/screenshot of the applications might make it easier for the user. The best solution would be if the OS managed background applications better, but as it stands today, a task manager is needed.

VI. Future work

It would have been a good idea to run the tests for at least two different mobile phones. However, because of the lack of availability of another android phone (and sufficient time), we were not able to do this. The algorithms for estimating remaining battery time is not that very accurate, an idea for future work here would be to use machine learning to improve the estimate for each battery cycle. The algorithm used to select which applications to suggest for killing could also be improved, but our evaluation showed that performs as expected, the main thing to improve is the data used by this algorithm. By adding information about screen and radio usage in addition to the already present CPU information would lead to more accurate selections and battery time estimations.

References

[1] http://www.androlib.com/android.application.com-sangodroid-batteryboosterfull-zAjw.aspx

[2] http://curvefish.com/apps/processmanager.htm

[3] CoolSpots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces, Trevor Pering, Y.Agrawal, R.Gupta, Roy Want

[4] Integrated power management for video streaming to mobile handheld devices, S.Mohapatra, R. Cornea, Nikil Dutt, Alex Nicolau, N. Venkata-subramanian

[5] http://developer.android.com/

[6] Battery Magic, www.myNewApps.com, Apple appstore

[7] http://developer.palm.com/

[8] Software controlled power management. Lu, Y.-H; Simunic, T.; De Micheli, G.; Comput. Syst. Lab., Stanford Univ., CA. Hardware/Software Codesign, 1999. (CODES '99) Proceedings of the Seventh International Workshop.

[9] System-level dynamic power management. Benini, L.; Bogliolo, A.; De Micheli, G.; Dipt. di Elettronica Inf. e Sistemistica, Bologna Univ. Low-Power Design, 1999. Proceedings. IEEE Alessandro Volta Memorial Workshop.

[10] OS Strategies for the Next Generation of Green Devices. Stephen Olsen, 2009 Embedded Systems Conference.

[11] Unified Power Management Framework for Portable Media Devices. Iyenggar, A.; Tripathi, A.; Ajit Basarur; Roy, I.; Ittiam Syst. Pvt. Ltd, Bangalore. Portable Information Devices, 2007. PORTABLE07. IEEE International Conference.

[12] System-Level Power Management for Mobile Devices. Bosch i Creus, G.; Niska, P.; Nokia Res. Center, Nokia. Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference.

[13] Adaptive power management using reinforcement learning. Ying Tan; Wei Liu; Qinru Qiu; Dept. of Electr. & Comput. Eng., SUNY - Binghamton Univ., Binghamton, NY, USA. Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference.

[14] Efficient power management in real-time embedded systems. Zuquim, A.L.A.P.; Vieira, L.F.M.; Vieira, M.A.; Vieira, A.B.; Carvalho, H.S.; Nacif, J.A.; Coelho, C.N., Jr.; da Silva, D.C., Jr.; Fernandes, A.O.; Loureiro, A.A.F.; Dept. of Comput. Sci., Univ. Fed. de Minas Gerais, Belo Horizonte, Brazil. Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference.

[15] PowerNap: an efficient power management scheme for mobile devices. Olsen, C.M.; Narayanaswarni, C.; IBM Syst. & Technol. Group, Hopewell Junction, NY, USA. Mobile Computing, IEEE Transactions July 2006.