Minimizing Energy Expenditure in Smart Devices

Soumya Kanti Datta, Christian Bonnet, Navid Nikaein
Mobile Communication Department
EURECOM
Sophia Antipolis, France
{dattas, bonnet, nikaeinn}@eurecom.fr

***Abstract*—The growing popularity of smartphones and tablets has highlighted several research issues. In this paper we focus on minimizing energy expenditure of Android devices. The energy dissipated by exotic hardware is explained in detail. The software development practices that result in high power consumption are also described. An application "Power Monitor" is developed to understand the usage pattern of smart devices. We have presented three usage patterns and have shown that how higher power consumption can be estimated from such patterns. This discussion sets the platform for power efficient application development. The paper then provides adequate road map to create such applications having reduced impact on battery life. Several guidelines for the end users are also provided to prolong the battery life. Finally the paper concludes with some future research directions on minimizing energy expenditure in Android devices.**

***Index Terms*—Android; energy expenditure; usage pattern; power optimized application development; battery life.**

I. INTRODUCTION

The smartphones and tablets are becoming dominant devices in the present market. Smartphone sales overtook the global PC sales in 2011 [6]. Apart from offering the same features of traditional mobile phones, the smart devices provide several other features as well. They include high speed access to internet using Wi-Fi and mobile data network, running several applications & games, capturing & sharing pictures to social platforms, audio-video capturing and sharing. The devices also boost array of sensors e.g. accelerometers, GPS, gyroscope, digital compass, light sensor, proximity sensor etc. A closer look to the hardware reveals that the devices contain powerful processor, RAM and bright colorful display. The application developers are exploiting the desktop-like features to provide sophisticated and engaging usage experience through mobile applications. But these hardware components are power hungry. Therefore continuous usage of them by various applications will shorten the battery life significantly as the capacity of the battery is limited and it has not increased drastically. For example, HTC Dream has battery capacity of 1150mAh while that of Samsung GT-I9100 is 1650mAh. The tablets have higher battery capacity (Google Nexus 7 has 4325mAh); still they also suffer from limited battery life. Research on improving battery capacity is ongoing but there is no reliable solution.

In the project Smart 4G Tablet [3, 7], we attempt to understand the power management issues of Android devices and research on ways to reduce power consumption. It has been observed that end users often complain about the battery life being low. Therefore a comprehensive understanding of how power is spent in the devices forms the crux of proposing solutions increased battery life. The objective of the paper is twofold. Firstly, it reports energy expenditure in smart devices. It is found that most energy is spent on the exotic hardware components, especially on the display hardware and networking interfaces. Also, the third party applications that transfer bulk data or keep waking in the background for automatic updates, consume much power. According to [2], most of the third party applications are not developed with power consumption as a priority. We have also developed "Power Monitor" to understand the usage pattern of smart devices and estimate power consumption from the pattern. We have deployed the application to several individuals and collected usage logs. It is seen that usage patterns can explain high power consumption of smart devices. The second part germinates from these observations and we attempt to answer how to develop power optimized Android applications while providing engaging user experience. At the same time, we put forward some measures that can be adapted by end users to reduce battery consumption further.

The remaining of the paper is organized as follows. Section II describes in detail energy expenditure in smart devices. Also several issues where there is energy waste are described. Section III estimates power consumption from individual usage patterns. Section IV provides several guidelines for the Android software developers to produce applications that are battery-aware and optimized accordingly. Section V puts forwards some guidelines for the end-users to control some features of their devices to increase the battery life. Section VI briefs future research directions.

II. ENERGY EXPENDITURE IN SMART DEVICES

Several researches have been undertaken to learn how energy is spent in smart devices. In the paper [5], the authors have presented a breakdown of power consumption by various hardware components. The results are summarized as below.

A. Energy dissipated at hardware components

- **Display hardware**: It consumes a major share of the total power. Higher the brightness of the touchscreen, higher is the power consumption of display hardware.

This work is sponsored by French Research project Smart 4G Tablet Pole SCS.

Thus reducing the brightness in smart devices would lower the power dissipation.

- **Network interfaces**: The network interfaces consume high amount of static and dynamic power. Figure 1 illustrates that even when the EDGE, Wi-Fi or 3G are idle, they consume power. Also when these technologies are being used by applications for data transfer, the power dissipation is higher [1].

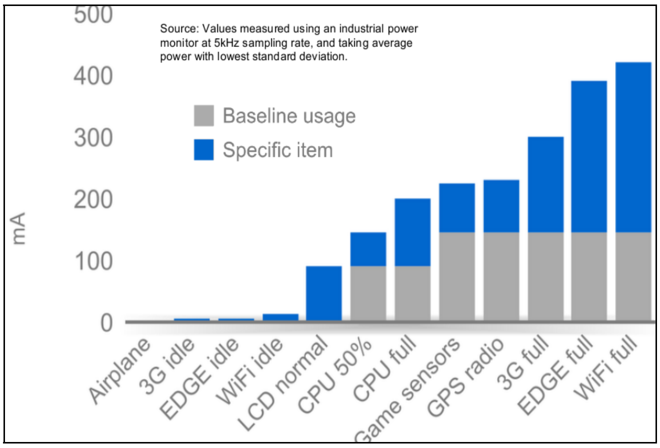

Fig. 1. Power consumption by various hardware components of smart devices.

- **Other hardware:** Among the other hardware components it is shown that RAM, audio and flash subsystems dissipate relatively lower power. Even for video playback which is quite intensive job, the power drawn by SD cards is below 1% of total power spent.
- **CPU:** The authors report that CPU operating with higher frequency draws more power. But they also argue that dynamic scaling of frequency may not be effective solution in this case as it will increase the execution time of applications and other tasks.

B. Power consumption by from Android applications

It is obvious that if Android applications do not use the hardware judiciously, the battery life will decrease considerably. In this subsection we present how the applications increase the power consumption.

- **Frequent waking up in background**: Consider an application that performs some tasks in the background as a service [9]. If the service wakes up to monitor some features and update, it draws significant amount of battery. It should also be noted that, the service occupies memory and consumes power even when it is not performing any task.
- **Power consumption for 3G connection**: Almost all the smartphones contain the hardware for 3G connection. The applications that depend on internet to fetch data from server(s) or run updates drain battery very quickly. Browsing web or performing similar tasks from applications over 3G would require typically 225mA. At this rate, a Samsung Galaxy S2 smartphone will be able to support 3G operations only for 7hours and HTM dream for 5hours [1, 18].
- **Static power dissipation:** Even if the mobile device is idle but connected to network using Wi-Fi, Edge or 3G, it consumes power to maintain the access to the network. Edge consumes about 5mA when completely idle [1].
- **Bulk data transfer:** Several applications (e.g. social media apps, YouTube, Dropbox) transfer bulk data or stream audio/video over internet and in turn account for high energy expense. In terms of battery consumption, 3G takes 225mA, Edge 300mA and Wi-Fi about 330mA. But in terms of speed, Edge offers around 90kbps, 3G offers 300kbps and Wi-Fi offers 1Mbps. Thus bulk data transfer over Edge drains battery the most [1].
- **Mobility:** Because of mobility when a smart device moves from a cell to another cell in the network, the radio link of the device has to strengthen itself for the new cell. For this purpose, broadcast intents are fired in Android system. This has additional impact on battery life.
- **Auto-sync:** The auto-sync feature is quite useful to remain up-to-dated with news, email, Facebook or Twitter notifications. But the always-on connectivity model limits the battery life to a few hours.

C. Detecting buggy behavior of Android applications

The bugs present in the Android applications also account for energy waste. The research [4] presents a statistical approach to detect such bugs. The paper classifies the applications as bugs and hogs. Bugs are defined as applications that use a lot of energy on small number of devices whereas hogs are applications using higher energy across several smart devices. A mobile application, Carat is developed to gather information about running applications, operating system version, memory usage etc and they are sent to a server for analysis. According to the paper, Carat is deployed to 883 users and the process has identified 5434 instances of applications with energy waste. Some of the Hogs detected by Carat in Samsung Galaxy S2 are Facebook, WhatsApp, AndroSensor and AVG Antivirus.

D. Power consumption by in-app advertisements

Another recent research suggests that free Android applications displaying in-app advertisements sometimes spend up to 75% of their total power (consumed by the applications) to display the in-app ads [2]. The authors have developed an energy profiler that highlights the wakelock bugs present inside applications. Other significant findings of the research are - (i) termination of TCP sockets can consume up to 50% of the total energy spent by an application and (ii) tracking data related to users consume up to 30%. Also the authors point out that most of the energy is spent at the I/O networking interfaces.

E. Sensor related energy expenditure

Smart devices contain wide range of embedded sensors like accelerometer, light sensor, proximity sensor, gyroscope and more. The authors of [20] have argued that applications ineffectively using the sensors and the output data waste much energy. They have reported 33 applications suffering from the problem. To use a sensor, applications must register a sensor listener [19]. This listener must be unregistered when the purpose is over otherwise energy would be wasted on sensor operations. Ineffective use of the output of sensor also causes energy waste.

All the factors mentioned above calls attention to efficient development of Android applications. They should use the hardware resources and wireless technologies judiciously to avoid energy waste. The remaining battery level should be taken into account before performing computationally intensive jobs and bulk data transfer. In the following section we put forward several important guidelines for power optimized application development.

III. Power Consumption Estimation from Smart device Usage Pattern

Usage pattern of smart devices reveal much information about how energy is consumed by the individual devices. To gather the usage information we developed an Android application called "Power Monitor". The application employs several monitoring segments to collect information as mentioned below and is deployed into several devices.

- **Application monitor:** It retrieves the running applications.
- **Battery monitor:** It records remaining battery level and charging/discharging status.
- **Context monitor:** Context information like date, time, location, luminosity is collected by this module.
- **CPU monitor:** It is responsible for registering the CPU load and operating frequency.
- **Display monitor:** It measures the total interaction time with a smart device and determines the brightness level of the touch screen.
- **Network monitor:** It records the status of Wi-Fi, Bluetooth, GPS, mobile data and the amount of network traffic.

We present three different usage patterns in this section to show the energy expenditure in the devices. All the devices are monitored for a week. To preserve the privacy of users, detailed usage patterns and location information are not revealed in the paper.

A. Samsung GT-I9100 with Android version 2.3.4

The user of the smartphone has the following pattern. The battery capacity is 1650mAh.

- The phone interaction time is 127 minutes/day on average and the brightness level is set to 30 which is the minimum for the phone.
- Wi-Fi and Bluetooth are not used.
- 3G is actively used for 105 minutes and idle for 1335 minutes resulting in 394mAh and 67mAh power consumption respectively. The network usage is also quite high, ~50MB/day.
- GPS is actively used for 30 minutes which dissipates 70mAh.
- Several applications (Facebook, Gmail, Live Score, Gtalk, WhatsApp) are running in the background.

Thus it is evident that around 32% of the battery capacity is spent on the networking operations and GPS. Additionally the interaction time is higher although the brightness is considerably low.

B. Samsung GT-I9070 with Android version 2.3.6

The pattern is outlined below.

- The interaction time with the device is 30 min/day and the brightness level is 35.
- 3G, Bluetooth and GPS are never used.
- Wi-Fi is actively used only for about 18 min/day. The power spent is about 60mAh and network usage is 8 – 10 Megabytes.
- The applications running in the background are contacts, Hindi dictionary and the launcher.
- The phone is charged every alternate day.
- The CPU frequency is always on the higher side irrespective of the operating frequency.

The battery life of the phone is estimated to be close to 30 hours since there is not much activity. The device interaction is very limited along with the brightness level is less. Wi-Fi is rarely used and traffic generated is not much high. The only anomaly with the device is that the CPU operating frequency is always on the higher side irrespective of the load on the CPU.

C. Samsung GT-P3100 with Android version 4.1.1

The following pattern is derived for the mentioned Android tablet.

- Tablet interaction time is five hours/day and the brightness is set to 107.
- Wi-Fi, Bluetooth and GPS are not used.
- 3G is actively used for around 78 minutes and is idle for rest of the day. This accounts for 361mAh of power consumption. 15MB of network traffic is generated.
- Several applications are running in the background including Facebook, Gtalk, Gmail and Chrome.

In case of this device, very high interaction time, brightness and continuous 3G connection are accounted for low battery life.

IV. Power Optimized Android Application Development

The smart devices require more applications that are engineered keeping in mind battery life. Developers may argue that sophisticated user experience come at the cost of high energy drain. But there are several ways to reduce the impact on battery from the codes [1]. These are described below.

A. Best practices for network operations

- **Detect network type for download:** The application must detect the availability of Wi-Fi or mobile data network before attempting to connect to networks. Schedule the high volume downloads for Wi-Fi as the speed is way higher than 3G. Thus even if Wi-Fi consumes more energy than 3G, the download time is significantly less in the case of former and the total power dissipation is also less.

- **Avoid multiple connections:** Creating and maintaining multiple network connections are common practices for several Android applications, particularly which stream HD videos or download high volume data. But multiple connections lead to more power consumption as the device has to change the state of networking interfaces frequently.
- **Prefetch data:** To avoid increased network operations over Wi-Fi, EDGE or 3G, it is a good practice to prefetch data and to store them locally in the smart devices. In this case, it should be kept in mind too much download will increase battery drain. According to Google recommendation, applications should prefetch data of 1-5 MB that would be potentially used in next couple of minutes. Also cache data locally to avoid download same content repeatedly.
- **Batch upload:** If it is necessary to upload data to servers, several such data must be zipped with GZIP and then uploaded. Such operations would also conserve the bandwidth and have less impact on battery. GZIP libraries are available in Android and should be used whenever possible.
- **Inexact timers for updating:** When scheduling network operations using timers in background services, the best way is to use inexact timers. In this case, the network operation will take place definitely but they can be performed together with some other network operations.

B. Best practice for getting location information

Location based applications (e.g. Google Maps, Navigation, Four Square) are very popular among users. In smart devices, fine-grained location co-ordinates can be obtained from GPS directly. But its use draws around 140mA and takes around 10 – 25 seconds to report the location. The smart devices also provide location service using the Wi-Fi or mobile data network. If that service is used the location is retrieved in 2 – 5 seconds with power consumption being around 180mA. Thus it is economical for the location based applications to use network technologies to determine the user's location. It is important to note that location services are often implemented as background service and location information may continue to update. Therefore, it is necessary to unregister so that the update does not run when the information is not being used for any purpose.

C. Best practices for display hardware

As display hardware draws high amount of battery, special care should be taken from the code while showing contents to the end users. The most important part is to reducing the brightness. It is possible to set the brightness level from Android applications. Developers could benefit from this and set the brightness to a lower value during the operation of the application. Also using darker colors for the screen can reduce impact on battery.

D. Monitoring battery level

Applications should be aware of remaining battery level. Operations such as intense computations, HD video streaming over internet, synchronizing data & pictures should be done when battery level is sufficiently high. When it is critically low, the applications must provide a choice to the user to either carry out the tasks at low battery or start them when battery level is higher than a predefined level. This is particularly applicable for applications like YouTube and Dropbox.

E. Best practices for CPU

CPU resources are requested through wake locks in Android. The wake locks must be carefully used in Java code otherwise they might result in very high power consumption. A simple solution is to use "android:keepScreenOn" property from the layout xml file.

Another important point in this case is the operating frequency that decides the power consumption. It is not possible to dynamically scale the CPU frequency from an application without rooting a smart device but Android OS can do so based on the load on CPU. Attempts should be made to distribute intense tasks, perform several tasks at parallel, avoid floating point mathematics and recycle Java objects. The floating numbers could be converted to integers for computation and the results could be converted back after that. This would allow the CPU to function at lower frequency and the power consumption will be significantly less.

F. Using efficient data format and parser

Browsing applications must use efficient data format and parser for faster UI response and less power drain. Figure 2 illustrates that using a stream parser over tree parser can speed up the parsing and consequently reduce energy expenditure [1].

G. Best practice for background processes

Antivirus applications and several others initiate background services that are always running. It is to be noted that each of the services occupy 2MB of memory even if they are not doing anything meaningful. The best practice in this case is to make the services short-lived as possible. If frequent monitoring of the device is necessary, the AlarmManager [11] should be used so that the service is offline between two successive calls. It is also possible to correlate the service with broadcast receivers [10] so that the service is not running all the time consuming critical resources of smart devices. Another aspect is to dynamically switch on/off manifest elements to reduce energy consumption.

H. In-app advertisements

Free Android applications often integrate an SDK for downloading and displaying advertisements. It has been already reported that up to 75% of total power drawn by an application goes to the third-party ads. It is advisable to the developers that the SDKs should download several ads at a time and store them locally. When there is network connection present, the ads should be shown and if the user clicks on ads, it should open the respective content. Thus, several connections to the ad server could be avoided just to load and display ads. This would minimize the energy use and prolong the battery life.

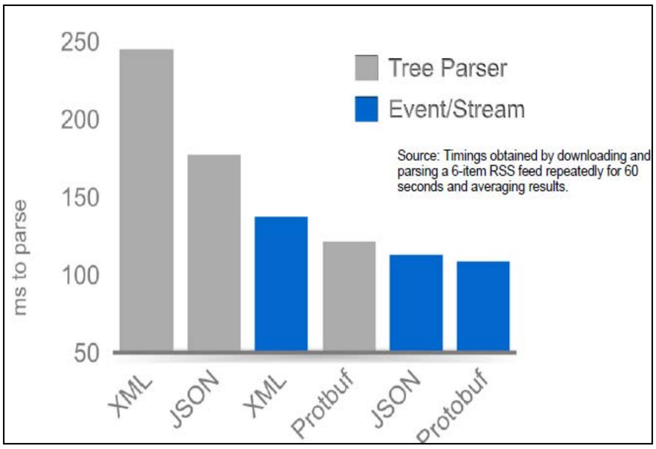

Fig. 2. Time to parse data for well known parser

I. Efficient use of smart device sensors

The sensors provide important context related information used by applications. The sensor listener should always be unregistered after the purpose of the sensor is served. At the same time, the sensor output values should be effectively used. Otherwise the power drawn by the sensors to provide the output are not justified.

The above mentioned are the road maps to develop applications that are aware of battery level and makes conscious attempts to minimize the battery drain. The interaction between end users and smart device is also important to increase battery life. Following section provides several such ways that must be adapted by users desiring longer battery life.

V. User Guidelines

Users must observe some guidelines to further reduce the power expenditure as explained below. These are necessary as a wide number of applications are not optimized for power. However the list is not exhaustive.

- **Reducing brightness:** Toning down the brightness manually will reduce the overall power spent in the display hardware.
- **Setting a lower timeout:** Android OS automatically switches off the screen in case of inactivity. The duration of the inactivity can be chosen and selecting a lower value for screen timeout is better.
- **Automatic orientation:** The accelerometer sensor reading is used to orient the screen automatically. And the sensor typically consumes 80 – 90 mA. Thus frequent changing of orientation between landscape and portrait will silently minimize battery life.
- **Turning off wireless technologies:** The wireless technologies spend energy to just maintain connection to the network. Thus if the smart device is not actively being used, Wi-Fi, 3G and Bluetooth must be switched off to conserve the battery.
- **Location:** Use of GPS should be limited to cases where highly fine-grained location information is needed. In other cases, the location update can be obtained from the network and GPS can be switched off.
- **Examining battery usage:** Android devices show a breakdown of how much power is used by the running applications. It is possible that some applications are not visible to the user as they are running in background but consuming higher power. Such applications must be studied with attention and killed or uninstalled to reclaim higher battery life.
- **Auto-sync feature:** This is suitable to stay updated with social media app notifications, Emails, news and more. But applications using this feature open and maintain multiple network connections leading to more energy consumption at the networking interfaces. This feature should be switched off most of the time and switched on manually only when needed.
- **Modifying update settings of applications:** Several applications (like news and weather category) frequently update the contents to be shown to the users. This requires more network operations and consequently battery life suffers. The update interval should be set to maximum time interval from the applications. The same applies for social media applications like Facebook and Twitter.
- **Airplane mode:** When moving through areas with very poor signal, the smart devices emit signal at quite high power. In such cases airplane mode could be activated which disables all connectivity and saves battery. During the night when there is practically no user interaction with his/her devices, airplane mode could be activated. It consumes about 1-2 mA.
- **Update applications:** Developers must update their applications and make them battery aware so that end users can benefit from increased battery life.
- **Using battery saving applications:** Several Android applications are available in Google play store that save battery by switching off unused features of the smart devices [14, 15].
- **Using Power Tutor:** This application provides how much energy is being consumed by the running applications in Android devices [12, 13]. It can point out if any application is using high amount power for long time.

VI. Future Directions

Considering the current landscape of Android devices, effective power management by Android applications is highly desired. We are developing another version of Power Monitor which employs a learning engine that automatically generates the usage pattern analyzing user behavior. The application will also dynamically derive power saving profiles based on the pattern. This approach has two distinct advantages: (i) the overall user experience remains intact as the power profiles are aware of user behavior and (ii) if the usage pattern changes over the time, the application will learn the change and modify the power profiles.

We are also examining several strategies to develop battery-conscious applications. We propose three types of profiling to be done based on the battery level. When the battery level is higher than 60%, the application should provide sophisticated user experience, network access and more. When the battery level is between 20-60%, some power preserving features should be activated from the application like reducing the brightness, bulk data transfer on high speed internet (e.g. Wi-Fi, 3G). Finally, if the battery level is below 20%, then user should be notified that due to low battery the brightness is set to minimum, data transfer should be kept to minimum, update features should be switched off etc. This type of application development makes an attempt reduce power consumption when battery is low and there is no immediate charging opportunity. Such development should be encouraged among developer

VII. CONCLUSION

In a nutshell the paper describes how energy is being spent in today's smartphones and tablets. Display hardware and networking interfaces are reported to be the most power hungry components. Two other significant research studies are mentioned out of which Carat classifies applications as bug and hogs. The other work reports significant power consumption by the third-party advertisements displayed in popular free applications. We also described three usage patterns derives using Power Monitor and estimated the power consumption of the respective devices. From a usage pattern it is found that around one-third of power is spent in the networking interfaces. Then the paper illustrates how to incorporate power saving tricks into Android development to create applications that are power optimized. Several useful tips are provided to minimize power drain by the display hardware, networking interfaces, keeping CPU frequency at the minimum and more. Moreover detail end user guidelines are also provided to further reduce energy expenditure. The future research scopes are also mentioned.

REFERENCES

[1] http://www.google.com/events/io/2009/sessions/CodingLifeBatteryLife.html

[2] A. Pathak, Y. C. Hu and M. Zhang. "Where is the energy spent inside my app? Fine grained energy accounting on smartphones with eprof." In Proc. of ACM EruoSys'12, Bern, Switzerland, 2012.

[3] http://investincotedazur.com/en/newsletter/wysips-gemalto-archos-and-eurecom-to-develop-the-first-self-powered-4g-tablet&artid=act10269

[4] Adam J. Oliner, Anand Iyer, Eemil Lagerspetz, Sasu Tarkoma, and Ion Stoica. 2012. Collaborative energy debugging for mobile devices. In Proceedings of the Eighth USENIX conference on Hot Topics in System Dependability (HotDep'12). USENIX Association, Berkeley, CA, USA, 6-6.

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

[6] http://www.gartner.com/newsroom/id/1924314

[7] http://www.pole-scs.org/projet/smart-4g-tablet?popup=1

[8] Datta, S.K.; Bonnet, C.; Nikaein, N.; , "Android power management: Current and future trends," Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on , vol., no., pp.48-53, 18-18 June 2012.

[9] http://developer.android.com/reference/android/app/Service.html

[10] http://developer.android.com/reference/android/content/BroadcastReceiver.html

[11] http://developer.android.com/reference/android/app/AlarmManager.html

[12] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. 2010. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES/ISSS '10). ACM, New York, NY, USA, 105-114.

[13] https://play.google.com/store/apps/details?id=edu.umich.PowerTutor&hl=en

[14] https://play.google.com/store/apps/details?id=com.latedroid.juicedefender

[15] https://play.google.com/store/apps/details?id=com.mhuang.overclocking&hl=en

[16] Joon-Myung Kang; Sin-seok Seo; Hong, J.W.-K.; , "Usage pattern analysis of smartphones," Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific , vol., no., pp.1-8, 21-23 Sept. 2011.

[17] Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. 2011. Identifying diverse usage behaviors of smartphone apps. In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference (IMC '11). ACM, New York, NY, USA, 329-344.

[18] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. 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). ACM, New York, NY, USA, 280-293.

[19] http://developer.android.com/reference/android/hardware/SensorListener.html

[20] Y. Liu, C. Xu, S.C. Cheung. Finding sensor related energy black holes in smartphone application. http://cs.nju.edu.cn/changxu/publications/PERCOM13.pdf [Accessed online on 25th January, 2013]