Android power management: Current and future trends

Conference Paper · June 2012

DOI: 10.1109/ETSIoT.2012.6311253

CITATIONS
30

READS
745

3 authors, including:

Soumya Kanti Datta
Institut Mines-Télécom
65 PUBLICATIONS **422** CITATIONS
SEE PROFILE

Christian Bonnet
Institut Mines-Télécom
205 PUBLICATIONS **2,868** CITATIONS
SEE PROFILE

Android Power Management: Current and Future Trends

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

Abstract—**Saving power of Android enabled devices have become a significant issue with 400,000 such devices being activated daily. Android smartphones and tablets offer several power hungry hardware components and the app developers are exploiting these components at disposal to provide revolutionary user experience. But the battery life has not increased at the same pace to support the power demand. Thus many researches have been carried out to investigate how to minimize the power consumption in smartphones. This paper reports four different research themes towards the reduction of smartphone power consumption. Efforts have been made to survey Android power saving apps available in Google play Apps store as the basis to find out different power saving approaches, operations and limitations. Then we present four different avenues to prolong the batter life of Android devices. The first two approaches include usage pattern analysis to generate power saving profiles. The advantage being that the power saving profiles are customized to actual user behavior. Integrating a photovoltaic film on top of the smartphone and tablet touch screen to generate electricity is also mentioned. The usage pattern based power saving profile generation has several privacy concerns which are discussed and countermeasures are proposed. Some emerging security attacks are also briefed.**

Keywords - Android power management; usage pattern; SetCPU; JuiceDefender; power monitoring; privacy; security attack.

I. INTRODUCTION

Power management (PM) of computers has evolved since the introduction of Advanced Power Management (APM) and Advanced Configuration & Power Interface (ACPI) [1]. These are primarily aimed at personal computers. Although Android is based on Linux kernel, Android has put forward its own power management system. During Google I/O 2011 [2], it was reported that 400,000 Android devices are being activated per day and the proper power management of these devices is becoming an issue. With more sophisticated hardware components being available on the smartphones and the tablets, the developers are exploiting them to provide state-of-the-art user experience. These come at the cost of high drain of battery. It is studied that Wi-Fi, GPS and colorful bright display of Organic LED (OLED) consume very high power [7]. Another study shows that the third party advertisements shown in free Android apps consume up to 30% of the total power consumed by the app [12]. Therefore, prolonged use of these hardware components and free apps displaying advertisements will increase the power dissipation and battery life will be reduced considerably. Android employs an aggressive policy for power saving using wake locks but that is not sufficient to conserve the battery lifetime. Thus developers wrote many power saving apps which are available in Google Play Apps store (previously known as Android market). To understand the operating principle, several of these apps are studied in depth. It is found that they aim at controlling different smartphone features like Wi-Fi, 2G and 3G connections, brightness level, CPU frequency, GPS and more to prolong the battery life. But in-depth study of these apps reveals that they depend on statically defined power saving profiles to control the smartphone features. Each profile has predefined control on several smartphone features which include turning off GPS, autosync and reducing the brightness level. Since smartphone usage pattern varies from user to user, the power dissipation pattern will also vary. Therefore, these static profiles might not suite several users.

Then four research avenues are presented to prolong the battery life. It is obvious that usage of smartphones and tablets vary from user to user. Thus power consumption behavior is different. To propose power saving profiles that better suite the need of users, we have to analyze the usage pattern. A client-server concept is proposed in which an app (client) collects usage information and sends them to a remote server that generates the power saving profiles. Another approach employs a learning engine within an Android app, that based on usage pattern generates the profiles. The privacy concerns are addressed. Integration of a photovoltaic cell to generate electricity and adaptive display is also discussed briefly.

The rest of the paper is as follows. Section II discusses the addition of a power driver to Linux kernel and basic Android power management architecture. Section III summarizes the related works on Android PM while section IV is the outcome of our survey on power saving apps. Section V provides the outline of improvements that could be done to prolong the battery life. Section VI discusses some privacy concerns and emerging security attacks based on the usage pattern analysis.

II. ANDROID POWER MANAGEMENT

Android stack is based on Linux kernel and Google added several new features to the kernel to support Android [3]. One such addition is a power driver to manage the device peripherals. As of kernel 3.3, some of the Android changes are merged with it [18]. The power driver and Android PM Architecture are discussed based on Android Gingerbread.

A. Power driver

Although Android inherits the power management of Linux, the former added its own power driver to the kernel

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

2.6.33. This driver [3] is added keeping in mind Android devices have limited battery life and the power saving features are different than personal computers. The driver controls the peripherals which include screen display & backlight, keyboard backlight and button backlight.

B. Android power management architecture and wake locks

A dedicated PM API is written in Applications Framework layer. Android apps are required to request CPU resources with wake locks through the application framework and native Linux libraries. The architecture is depicted in Figure 1. If there is no wake lock active, CPU is shut down [1]. Wake locks are used by applications and services to request CPU resources. A locked wake lock, depending on its type, prevents the system from entering suspend or other low-power states. There are two settings for a wake lock. WAKE_LOCK_SUSPEND prevents a full system suspend while WAKE_LOCK_IDLE is a low-power state, which often cause large interrupt latencies or that disable a set of interrupts, will not be entered from idle until the wake locks are released.

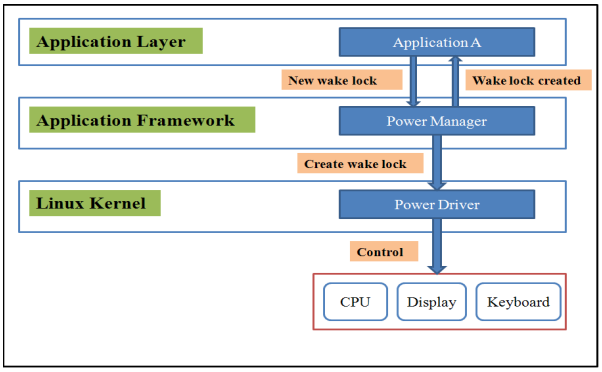

Figure 1. Android power management architecture.

In Figure 1, when Application A is launched, it needs to use CPU resources. Thus it sends a request to Power Manager API in applications framework which in turn transfers (using Java Native Interface) the request to the power driver present in the Linux kernel. Power Manager also reports back to the Application A that the wake lock is created. And depending on the wake lock, resources are consumed. Table 1 lists different wake lock settings. For application which does not interact with the user and run in the background, PARTIAL_WAKE_LOCK is used.

TABLE I. DIFFERENT WAKE LOCK SETTINGS

Wake Lock	CPU	Screen	Keyboard
PATIAL_WAKE_LOCK	On	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

III. RELATED WORKS

Several researches have been carried out in relation to Android power management and increasing the lifetime of the battery. The state of the art is presented in four separate parts each highlighting significant research themes.

A. Research on power consumption of connectivity features

Here, we mention some related works that have focused on power saving in one of the power hungry features. It is well established fact that the Wi-Fi, 3G and GPS hardware are some of the most power consuming components in a smartphone [7]. Reference [6] implemented a system WiFisense that uses mobility information obtained from sensors and apply adaptive Wi-Fi sensing algorithm. This method improves Wi-Fi usage while maintaining the battery life. The work applies user movements and density of access points for the efficient Wi-Fi sensing and the experimental results validate the work.

Many researchers have aimed at reducing the power consumption of GPS [4], [5]. Location Based Apps (LBA) for social networking, local weather and traffic need continuous location update of user. These apps also use the location information to display third party advertisements. But frequent usage of GPS shortens the smartphone battery considerably. To address this issue, the authors of [4] present an adaptive location sensing framework. The framework employs four principles: substitution, suppression, piggybacking and adaptation. Substitution uses different positioning systems instead of GPS when high accuracy of location is not necessary. When the user is in static mode, suppression uses accelerometer-less power sensors to position the user. The location requests of several LBAs pass through Piggybacking which is responsible for caching and distributing the location information to all the LBAs. Adaptation manages the position information when the battery is low. The experimental result in [4] proves that using the framework battery life could be extended up to 75%.

B. Context aware power management

The "context-aware" concept has already been used in relation to power management of assisted living [10]. The authors of [9] approached power management using the context information derived from mobile phone usage. They proposed context-aware battery management architecture (CABMAN). It records context information (e.g. location) using a context monitor, battery level using a battery monitor, processes running using a process monitor and incoming outgoing communication using a call monitor. CABMAN is also equipped with algorithms that can predict the next charging opportunity and battery lifetime.

C. Power Model Generation

To understand the power dissipation in smartphones, it is necessary to generate a power model. Researchers have been successful to generate such a model of smartphones. The paper [7] introduces an automated power model construction technique called PowerBooter. This technique makes use of the smartphone sensors to monitor the battery consumption and records the battery discharge behavior. An Android app PowerTutor is also developed that uses PowerBooter technique to display power consumed by each app running in a smartphone. The authors have correlated the power consumption with the hardware elements of the smartphones. Each hardware element has several states and power consumption depends on the states [8]. For CPU, the power

state depends on CPU utilization and CPU frequency. The main advantage of PowerTutor is that, it does not require any external power measuring devices to show power consumption.

D. *Towards the analysis of smartphone usage patterns*

The authors of [11] have presented an approach to gather usage information of smartphone users and analyze them to reveal usage pattern. An application called battery logger is developed to store the smartphone usage log of real users and the log is collected. Based on the received information the authors have analyzed the average usage time and related power consumption, smartphone usage pattern, network usage and battery usage. The paper also describes a method to predict the battery life based on the usage pattern.

Apart from these approaches, [12] investigates internal energy consumption of apps. An energy profiler is developed to measure such power consumption. The difficulties arising from asynchronous power behavior is explained in terms of tail energy [7], wake locks and exotic components like GPS and sensor hardware. Interestingly it is found that, popular free apps (e.g. Angry Birds free version) dissipate high amount of power in displaying 3rd party advertisements, location based user tracking and clean termination of long lived TCP connection. The algorithm written to achieve the purpose of the app consumes only 10-30 percent of the total power consumed by the app.

Another approach [19] introduces the concept of human-battery interaction. The authors performed an international survey and interviewed mobile phone users to understand charging behavior, user interface for power saving settings and more. The paper then tries to interpret how users cope with limited battery life.

IV. SURVEY ON POWER SAVING APPS

In order to prolong the battery life of Android powered devices, developers have written many apps available in the Google Play Apps store. Several such apps are studied during the survey to understand the power saving approaches, how they increase power efficiency, their operating principles and limitations that pave way for improvements. It is observed that these power saving apps have two distinct approaches for controlling power consumption. These approaches are portrayed in Table II. The subsequent sections discuss power efficiency increase, operation and limitations of SetCPU for Root Users [13], CPU tuner (Rooted phones) [14], JuiceDefender [15]. These apps are chosen based on their popularity, high user rating and positive user feedback.

TABLE II. POWER SAVING APPROACHES OF APPS

Primary approach	Secondary Approach	Example of apps
CPU frequency scaling	Controlling smartphone features	SetCPU, CPU tuner
Controlling smartphone features	CPU frequency scaling	JuiceDefender

A. *Increasing power efficiency*

Several hardware components like GPS, Wi-Fi of smartphones and tablets consume very high power [7]. Thus power can be saved by switching them off when not being used. There are some other features like autosync, notification frequency which use the connectivity and other hardware. Lowering such notification frequencies (mainly Facebook, Gmail) will reduce the usage of smartphone components and increase power efficiency. Following are the features which are controlled by these power saving apps to increase battery life. The list is not exhaustive.

- Toggle control on Wi-Fi, Bluetooth, GPS, auto sync, airplane mode, auto screen lock, USB mass storage, screen-always-on, torch, 2G, 3G, 4G/Wimax (if present) and mobile data (APN).
- Change brightness level of display.
- Volume and vibration control.
- Alter screen timeout value.
- Scheduling – night, weekend, peak.
- Setting Wi-Fi timeout.
- Setting dark home screen wallpaper for OLED display.

The mentioned three apps use all or a subset of these features in their power saving profiles.

B. *Operation of power saving apps*

It is important to understand the operating principle of the apps to investigate their limitations. We present the working diagram of SetCPU and CPU tuner in Figure 2.

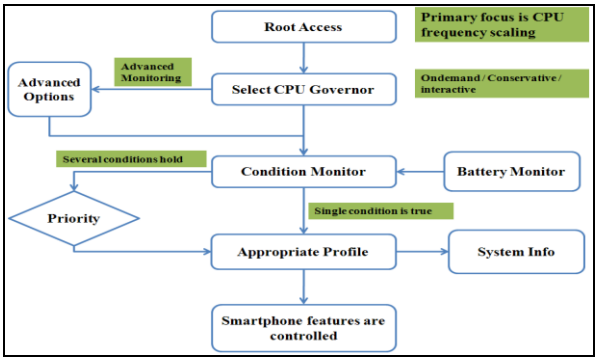

Figure 2. Operation of SetCPU and CPU tuner.

Once the app is installed in a rooted phone and root permission is granted, there are sliders that allow controlling the CPU frequency manually [17]. Then CPU governor must be selected. It controls how the CPU frequency should be scaled between the maximum and minimum set frequencies. Most of the kernels (consequently smartphones) have "ondemand" and "performance". When the CPU load reaches a threshold, ondemand scales up the frequency rapidly and scales down the frequency when the load is lesser. Other available CPU governors are listed in [17]. Some of them have advanced condition monitoring features. The profiles configure the app to set the CPU frequency under certain conditions. There is a "condition monitor" which continuously monitors the conditions set in profiles. If such a condition is true, the respective profile is triggered. For example, the profile "Battery <" is set when battery level falls below a given threshold. The "Time" profile is activated for a particular duration of time. Each profile has a priority. If conditions of several profiles are true, then the priority of the profiles is checked. The profile with highest priority is activated. Each

profile is also able to provide the system information i.e. the battery level, memory status etc.

In case of CPU tuner, the concept of prioritizing the power saving profile does not exist. Rest of the operation is same as SetCPU.

JuiceDefender on the other hand focuses on controlling the smartphone features, like disabling connectivity and reducing brightness to reduce power consumption. There are several versions (both free and paid) of JuiceDefender available in Google play store. One version of the app requires root permission to scale CPU frequency and toggle 4G/Wimax. But that is not the prime concern in this power saving app. The operational diagram is given in Figure 3.

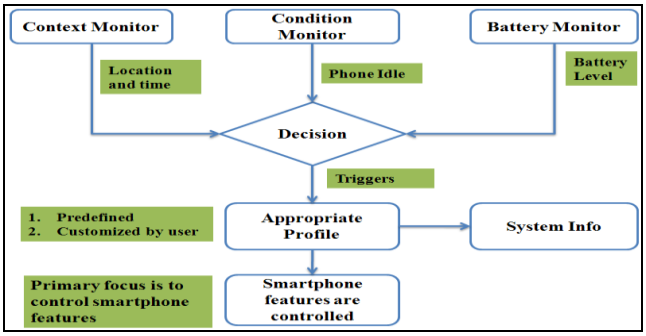

Figure 3. Working diagram of JuiceDefender

The app employs context monitor (which extracts contexts such as location and time), condition monitor and batter level monitor. The condition monitor constantly looks for if conditions set in any profile is met and consequently triggers appropriate profile. Each profile has different control over the smartphone features. Most of the profiles are statically defined during the development of the app and there is one profile that can be customized by the users. In this app, the frequency scaling of CPU is secondary focus.

C. *Limitation of these apps*

- The profiles are defined statically during the app development and are not customized for user specific behaviors.
- The controlling part of these apps is not very intelligent. If proper condition is met, the associated profile is activated to control the connectivity for example. But after charging when the battery is full, most of the apps do not restore the previous states. There is no dynamic decision making and the profiles do not evolve.
- To use SetCPU and CPU tuner, the smartphones have to be rooted. These apps need root permission to perform CPU frequency scaling.
- It is understood from [12] that much energy is dissipated displaying advertisements and tracking users which does not consume CPU resources. Thus CPU frequency scaling should not be the primary focus of power saving apps.
- The context information (in JuiceDefender) is not used for any usage pattern generation process.
- These apps do not focus on learning the power consumption pattern of user. It is obvious that, different users install different apps and consequently the power consumption will vary. Thus learning the power consumption pattern and then building power saving profiles to control the smartphone features need to be done.

V. FUTURE DIRECTION IN POWER MANAGEMENT

The limitations mentioned above leave enough room for improvements. It is obvious that defining static profiles which do not evolve based on the user behavior, is not ideal way of approaching power saving. In this paper, we propose four directions to prolong battery life.

A. *Client-server concept to generate power saving profiles*

The client-server architecture is motivated from [11] and is depicted in Figure 4. An app (that acts as client) could be developed that records the battery consumption of the apps running at certain interval of time along with the context information (location, date, time, environmental information etc.). The record log is sent to a remote server over a secured connection. This is done to protect the personal information being carried by the log file. The remote server collects several such logs and processes them offline. Such processing will reveal much useful information as mentioned below:

- Power consumption pattern of users, which could be worked out by associating battery consumption with the Android apps.
- The context information and associated battery consumption will also lead to understanding of the context(s) in which most of the power is spent. It could be predicted based on the usage history that if the battery will last during the most power consuming interval.
- It is possible to find several users with similar smartphone/tablet usage behavior and these usage patterns could be clustered. These clusters will contain data related to the features which are used the most, in which context and what period of time. From these data, power saving profiles could be generated that control smartphone and tablet features based on user behavior.
- As the number of databases grows, the mentioned clusters will evolve. Thus the power saving profiles will also evolve.
- An important aspect of this client-server concept is the network usage pattern. Since the number of Android devices continue to increase, it is necessary of the service providers to understand the pattern of network usage. In that way they could be able to extrapolate the network usage and how to cope with the increasing demand.

Apart from the mentioned capabilities, another useful functionality could be to periodically connect to the remote server to fetch the power saving profiles. These profiles will be able to control the Android device features based on the knowledge of device usage. Thus the user is presented with several profile choices and can select the best profile to minimize overall power consumption. This approach will give intelligent control over the Android device feature as it depends on the user behavior.

This approach can lead to several privacy and security issues as usage pattern is being transferred to a remote server for data mining. They are addressed in Section VI.

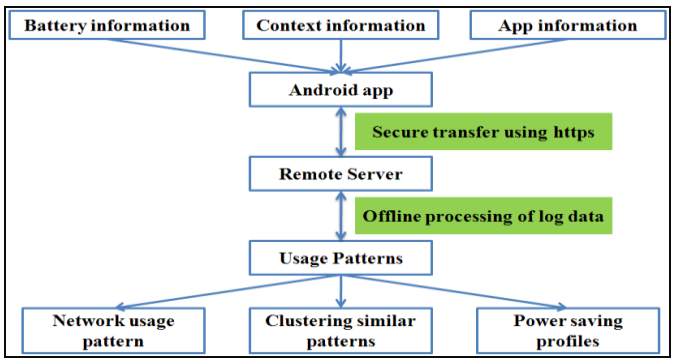

Figure 4. Client-server concept to generate power saving profiles.

B. *Android app with learning engine*

This is an advanced approach and does not require the client-server architecture. The working of the app is depicted in Figure 5. In this case, the developed app will include a learning engine. The app will monitor the user behavior in terms of battery consumption, apps used and contexts. The information will be collected for a period of time and then fed to a learning engine. Artificial neural network can be used to implement the learning engine. It will develop the usage pattern and based on the pattern intelligently decide how to control the smartphone/tablet features. The app will not contain any statically defined power saving profiles but will build the profiles after learning the user behavior. Since, the entire processing is done on the Android device, there is no privacy concern. But the demerit is that the service providers could not get network usage pattern.

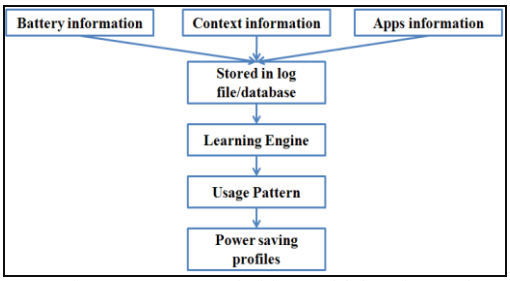

Figure 5. Android app with learning engine.

C. *Adding another power source – a photovoltaic cell*

In order to add another power source in Android devices, a photovoltaic cell could be integrated. WYSIPS has introduced a transparent photovoltaic film that can be integrated on top of the touchscreen of smartphones and tablets. This film generates electricity from solar power and can charge the battery [16]. This concept could potentially increase the battery life.

D. *Adaptive display*

Since Android device display consumes high power, the display screen dimension could be shortened if battery level falls below some predefined threshold. In case of streaming video, the resolution could be decreased. Thus the concept of adaptive display could actually make the dying battery last longer.

VI. SECURITY & PRIVACY CONCERNS IN USAGE PATTERN ANALYSIS

This section discusses the security and privacy concerns relating to the usage pattern analysis. Today's devices contain user sensitive information and collecting usage logs to interpret usage pattern raises the issue of privacy. Also it could lead to user behavior based security attacks on the Android devices. This could be an emerging attack directed towards the Android smartphones and tablets. The privacy concerns and some related security attacks are described below.

A. *Privacy concerns*

It is obvious that the generation of power saving profiles through user pattern analysis has to be privacy aware. In [11] it is mentioned that the data collected by the Android app are stored in a log. Another Android app could be written to access the log and send it to another server. The server can process such logs and easily generate usage patterns. Since the logs contain location information, it becomes easy to spam the device with location based advertisements. The location privacy of users is also compromised.

Such scenario could be extended where an app collects user centric information like MAC address of the device, IP address, email account credentials and phone number and sends them to a particular server along with usage logs. Then user behavior can be tracked and it compromises the privacy of user.

To eliminate the privacy concerns, following steps are to be taken.

- Instead of writing all the collected information into a log file, they could be stored in a database which could only be accessed by the Android app.
- The app randomly decides when to send the database to the remote server. The app creates a database dump, opens a secure connection to the remote server and sends the dump file. Then the dump file is deleted so that other apps could not access it.
- Fetching the power saving profiles will also be done via a secure connection.
- The user should be given some control over how much usage information is to be sent to the remote server.
- The remote server must employ some privacy preserving data mining algorithm to generate the

usage patterns. Such patterns should never be revealed to any third party.

B. *Security attacks*

There could be several possible cases of security attacks on the Android devices based on usage pattern. In this work, repackaging attack is described with the help of Figure 6. Consider a case where the battery monitoring app mentioned in Section V-A is published in Google Play Apps store and over time the app has gained popularity. An attacker can simply download the app; reverse engineer to obtain the codes. Then the same attacker can exploit any existing vulnerability or inject malicious codes and repackage the app. Now the malicious app could be published in Google play store. When the app is installed in an Android device, the app can pose various security threats to user.

The app could silently steal user information including account credentials, location and credit card details and send them to a malicious server. This will lead to several attacks such as location based attack and financial loss. Attackers can also leave a backdoor open in the malicious app so that when the apps connect to the remote server and attackers can send remote commands to gain control over the device. Then the device could be controlled using a command and control server and perform spamming. If the remote server knows the duration when a user is using network connections maximum during a day, the attacker can user the knowledge to spam other devices. This will hide the fact that an app is spamming as it intelligently works during the busiest time of the user. Thus a device could become mobile botnet.

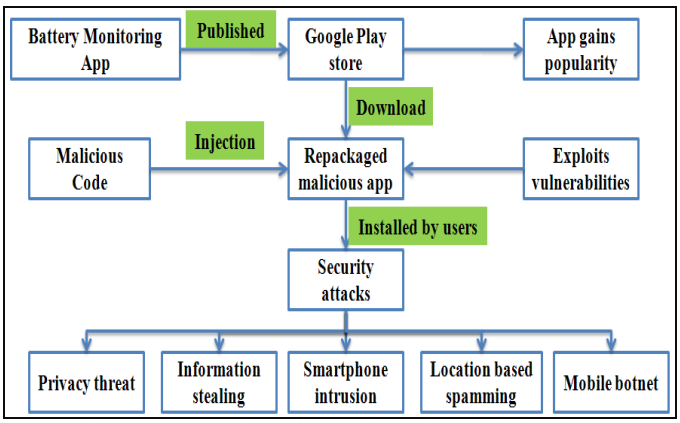

Figure 6. Privacy threat and security attacks based on usage pattern of smartphone and tablet users.

VII. CONCLUSION

In a nutshell, the paper highlights the need of efficient battery management for Android smartphones and tablets. The Android power management architecture is briefly discussed. Different research directions of smartphone power consumption are discussed. The operation of SetCPU and JuiceDefender are explained along with their power saving approaches. The limitations are identified and it is found that the profiles for power saving are statically defined. Thus controlling the smartphone features like connectivity, brightness of display is not very intelligent.

Then four research avenues are presented to prolong the battery life of Android devices. The client-server based approach is motivated from [11] although the offline processing in the remote server could be extended to a great extent. Similar usage patterns could be clustered to generate power saving profiles that takes into account actual usage pattern. But this approach raises privacy and security concerns. Another idea is to employ a learning engine to generate the usage pattern and develop power saving profiles. This approach will minimize the privacy concerns as all the processing is done inside the app. A photovoltaic cell could be added to smartphones and tablets to produce electricity from solar power. Adaptive display could also increase the battery lifespan. The privacy issues concerning usage pattern analysis is explained and some countermeasures are proposed. Finally some emerging security attacks are discussed.

REFERENCES

[1] www.cs.uwc.ac.za/~mmotlhabi/apm2.pdf

[2] http://www.google.com/events/io/2011/index-live.html

[3] S. K. Datta, "Android stack integration in embedded systems," in International Conference on Emerging Trends in Computer & Information Technology, Coimbatore, India, 2012.

[4] Z. Zhuang, K. Kim and J. Pal Singh. "Improving energy efficiency of location sensing on smartphones." In Proc. Of ACM MobiSys'10, San Francisco, California, 2010, pp. 315-329.

[5] J. Peak, J. Kim and R. Govindan. "Energy-efficient rate adaptive GPS-based positioning for smartphones." In Proc. Of ACM MobiSys'10, San Francisco, California, 2010, pp. 299-314.

[6] K. Kim, A. Min, D. Gupta, P. Mohapatra and J. P. Singh. "Improving energy efficiency of wi-fi sensing on smartphone." In Proc. Of IEEE INFOCOM, 2011, pp. 2930-2938.

[7] L. Zhang, et al. "Accurate online power estimation and automatic battery behavior based power model generation for smartphones." In Proc. Of ACM CODES+ISSS'10, Arizona, USA, 2010, pp. 105-114.

[8] http://ziyang.eecs.umich.edu/projects/powertutor/index.html.

[9] N. Ravi, J. Scott, L. Han and l. Iftode. "Context-aware battery management for mobile phones." In 6th Annual IEEE International Conference on Pervasive Computing and Communications, 2008, pp. 224-223.

[10] A. D. Wood, et al. "Context-aware wireless sensor networks for assisted living and residential monitoring." In IEEE Network, vol. 22, issue 4, 2008.

[11] J.M. Kang, S. Seo and J. Hong. "Usage pattern analysis of smartphones." In 13th Asia-Pacific Network Operations and Management Symposium, 2011, pp. 1-8.

[12] 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.

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

[14] https://play.google.com/store/apps/details?id=ch.amana.android.cputuner&hl=en

[15] http://www.juicedefender.com/

[16] http://www.arctablet.com/blog/featured/wysips-turning-your-device-display-into-a-solar-cell/

[17] http://www.setcpu.com/documentation.html

[18] http://www.pocketables.net/2012/03/linux-kernel-33-update-merges-androids-changes.html

[19] A. Rahmati, A. Qian and L. Zhong. "Understanding human-battery interaction on mobile phones." In Proc. of ACM MobileHCI'07, Singapore, September 9-12, 2007.