A POWER MANAGEMENT MODEL FOR ANDROID OPERATED SMART PHONES

Aisha Awal[1], Amit Mishra[2], Elijah Joseph[3], A AbdulG[4]

[1,3,4]Department of Computer Science, IBB University, Nigeria

[2]Department of Computer Science, Baze University, Abuja Nigeria

Abstract:

Power management (PM) of smartphones has evolved since the introduction of Advanced Power Management (APM) and Advanced Configuration & Power Interface (ACPI). 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, 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 [2]. 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. This comes at the cost of high drain of battery. It is studied that Wi-Fi, GPS, colourful bright display of Organic LED (OLED) and some uncontrollable background running applications consume very high power. 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. Therefore, prolonged use of these hardware components and free apps displaying advertisements will increase the power dissipation and battery life will be reduced considerably [3]. 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. Historically, power management techniques rely on the ability of certain components to be turned on and off dynamically, thus enabling the system as a whole to save energy when those components are not being used. Only more recently, techniques have been introduced to enable some components to operate at different energy levels along the time. Multiple operational modes and Dynamic Voltage Scaling (DVS) are examples of such techniques that are becoming commonplace for microprocessors. Unfortunately, microprocessors are seldom the main energy drain in embedded systems so traditional on/off mechanisms are still of great interest.

Keywords: Power Management, Smart Phone, Android OS

STATEMENT OF THE PROBLEM

Rapid power dissipation in android smartphones that is addressed from energy aware design to energy efficient implementation, 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. Wi-Fi, GPS, colourful bright display of Organic LED

(OLED) and some uncontrollable background running applications has been the main causes of high power consumption in smartphones whose optimal management is yet to be achieved.

OBJECTIVE OF STUDY

The main objective of this research work is to develop and design efficient client server architecture (app) and applying wake locks techniques that will depend on statically defined power saving profiles to manage the smartphone features such as Wi-Fi, GPS, Colourful bright display of Organic LED (OLED) and some uncontrollable background running applications.
Each profile will have a predefined control on several features of android smartphones which include: Turning off GPS, Reducing the brightness level and, dynamically shut down of some background running applications to make optimal use of the battery life.
In addition, the client-server (app) concept will be applied to collect usage information of clients and sends them to a remote server that generates the power saving profiles as the usage of smartphones varies from user to user.

LITERATURE REVIEW

This section highlights on the existing schemes of reducing the power dissipation rates on android smartphone devices along with their various strength and weaknesses.
Advanced Power Management APM is an API which was developed by Intel and Microsoft and was proposed and implemented on android devices by android operating system developers starting on earlier android operating system versions such as Android alpha, android beta, Cupcake, Doughnut, Éclair, Froyo and Gingerbread. APM consists of one or more layers of software that support power management in android devices with power manageable hardware. APM defines the hardware independent software interface between hardware-specific power management software and an operating system power management policy driver. It masks the details of the hardware, allowing higher-level software to use APM without any knowledge of the hardware interface. The APM software interface specification defines a layered cooperative environment in which applications, operating systems, device drivers and the APM BIOS work together to reduce power consumption. In brief, APM extends the life of system batteries and thereby increases productivity and system availability. As the power demand failed to meet the need of several android users, a new improved technique was proposed and implemented which improved on the power management of APM on smartphones this technique is called Advanced Configuration Power Interface (ACPI) [7].
Advanced Configuration & Power Interface ACPI this power management technique allows control of power management from within the operating system unlike APM where power management is done at BIOS level. With ACPI the user can specify at what time a device, such as a display monitor, is to turn off or on. The user can specify a lower level of power consumption when the battery starts running low so that essential applications can still be used while other less important applications are allowed to become inactive. The operating system can reduce motherboard and peripheral device power needs by not activating devices until they are needed. With ACPI, the computer can power itself down to a deep sleep state but still be capable of responding to an incoming phone call or a timed backup procedure. In short ACPI gives both the user and the operating system the power to customize the system to suit their personal power needs [8].

In relation to the two above mentioned power management techniques in android smartphones which are the inbuilt power management techniques from the android mobile manufacturers, hundreds of applications which serve as addition to managing power on android devices and as client server has been developed by computer programmers which are available for download on android app market (Play store) some of which are reviewed below based on their efficiency and number of user downloads.

DU Battery Saver & Widgets app by DU apps Studio is a free battery-saving app that makes your android battery last longer, users get about 50% more battery life on their android phones and tablets. The DU Battery Saver has a smart pre-set battery power management modes and easy one-touch controls that solves battery problems and extends the battery life. This power saving app is designed with "Optimize" home screen widget that allows users to stop power consumptive background apps with one tap to boost the battery life [9].

Battery Doctor (Battery Saver) app by Cheetah Mobile (Formerly known as KS Mobile) The Battery Doctor app provide a longer battery life to Android devices, it provides users with detailed battery information and helps android devices to charge healthily with a unique 3 state charging system function while extending users battery life up to 50% by finding apps and settings that drains power on users' android devices [10].

Easy Battery Saver app developed by 2Easy Team is an energy saving application that helps to extend users battery life and optimizes user's android devices hand-on experience. This app saves users battery by intelligently dealing with phone's network connectivity, screen timeout and screen brightness. It's designed with considerable sleep schedule setting function that saves battery when a user is asleep [11].

JuiceDefender – Battery Saver app developed by LATEDROID is a powerful yet easy to use power manager app specifically designed to extend the battery life of users' android devices. The app is packed with smart functions that automatically and transparently manage the most battery draining components like 3G/4G connectivity and WiFi. It allows users complete customization through a clean and intuitive user interface that when once configured runs by itself improving battery life in a fully automated manner, it also integrates seamlessly with power control widgets and shortcuts without interfering with the manual settings [12].

Battery by MACROPINCH this app is a small, sleek and elegant app that will help you follow the current battery percentage on client Android device and even serve as a battery saver plus tells if the battery is charged enough to play a game, a movie, or to browse the web designed with an intuitive, neat and gorgeous interface as a battery has. The UI is as simple as possible but tremendously practical [13].

Table 1: Below is a table showing the strengths and weaknesses of the literature reviewed

Author/Literature	Strengths	Weaknesses
Intel and Microsoft. Advanced Power Management APM	Makes power-management decisions without informing OS or individual applications	Uses activity timeout to determine when to power down an app and device
Intel and Microsoft. Advanced Configuration & Power Interface ACPI	Decisions managed through the OS. Enables sophisticated power policy	No knowledge of device-specific scenarios (e.g. Need to provide

	general-purpose computers with standard usage patterns and hardware	predictable response times or to respond to critical events over extended period)
DU apps Studio. DU Battery Saver & Widgets app	- Accurate battery status - Smart pre-set mode - One click optimization - Better battery details	Wrong battery information on Android 2.3 (Gingerbread)
Cheetah Mobile (Formerly known as KS Mobile) Battery Doctor (Battery Saver)	- Disables unnecessary apps that drains battery - Kills tasks with one click - Gives accurate battery and charging remaining time - Unique 3 stage charging system function	- Compatibility issues with android 4.4 (Kitkat) - App often crashes after sometime on use
2Easy Team. Easy Battery Saver app	- Visualizes battery status on main page - Gives battery consumption list of real time running items - Gives notice on how to use the battery better	- Barely improves battery life to about 20% - Background running apps still runs even when killed from the app
LATEDROID. JuiceDefender – Battery Saver app	- 5 pre-set profiles (from default mode to full customization) - Home screen battery widgets - 2G/3G toggle automation - WiFi toggle automation plus Auto-Disabling option - Location aware WiFi control	Slow in processing user requests
MACROPINCH. Battery by MACROPINCH	- Supports the lock screen widgets introduced in Android 4.2 - Offers quality support for Android devices from Android 1.5 to Android 4.3 - Supports all known screen resolution	Consumes more memory upon installation

RESEARCH METHODOLOGY

The limitations mentioned above in the earlier section on the strength and weaknesses of the various power management schemes leave enough room for improvements. This research

methodology however proposed four dimensions of reducing the power dissipation rate on android smartphone devices.

MODELING POWER MANAGEMENT SCHEMES IN ANDROID SMARTPHONES

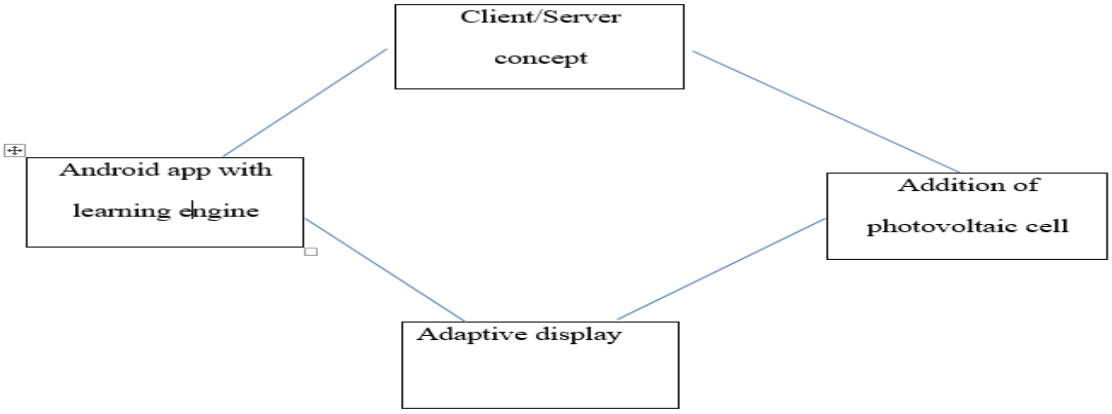

Figure 1: Research model

Client-server concept to generate power saving profiles

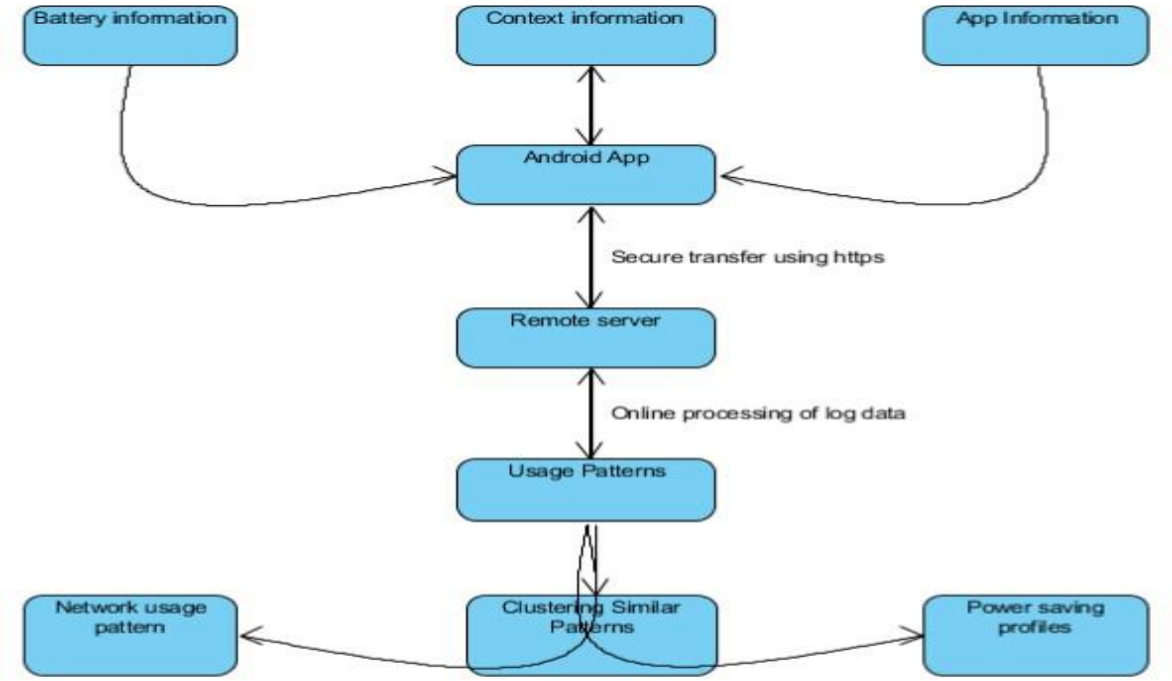

Figure 2: Client-server concept to generate power saving profiles [1].

The client-server architecture which is an application (that acts as client) could be developed to record 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 will be 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 would collect 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 behaviour 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 behaviour.
- 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 [14].

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 behaviour. There would be two comprehensive kinds of power saving profiles which are listed and explained in detail.

PROPOSED POWER SAVING PROFILES

Power saving profile is a collection of power saving modes that a user can select at a certain percentage of the battery level. This will optimize the battery for optimum use and prolong the battery power. It will give users the opportunity of selecting a power plan amongst the two featured kinds of power saving modes. These power saving profiles includes:

- ✓ *Power plan*
- ✓ *Stamina mode*

Power PLAN: this power saving profile of power monitoring would help user select the best usage plan according to their needs. It will give the user the opportunity of selecting a power plan amongst the three featured kinds of power plans at certain battery percentage level. This power plan includes:

- ✓ Normal power plan
- ✓ Smart power plan
- ✓ Ultra smart power plan

NORMAL POWER PLAN*:* this power plan takes the default android device power plan, enabling all the social functionalities connectivity at user discrete. It slightly adjusts the CPU

and network usage to maximize performance, this plan is recommended for gaming and online videos and maintains it status only when the battery level is above 30%.

SMART POWER PLAN: this plan automatically adjusts the CPU and network usage for balanced performance which is recommended for daily use.

ULTRA SMART POWER PLAN*:* this plan would only keep basic call and message functions available and is automatically activated when the battery is very low say at 10% to keep the user in contact for twice the time when in either of the two kinds of power plan. When activated it would switch the phone colourful GUI and optimize it for only call and messaging functions with a single like GUI colour although the user could exit the Ultra Smart power plan function whenever he/she so desires.

Stamina Mode

Unlike the other kind of power saving profile that requires the user attention to activate the various kinds of power plan, the stamina mode automatically disables the social functionality connectivity such as Data services, background running applications, WiFi, GPS, and also reduces the colourful bright light display to its lowest minimum when the user is unaware of his/her battery level at 30%. The device would restore it previous state when the user exits the stamina mode.

However, this approach of Client-server concept to generate power saving profiles can lead to several privacy and security issues if not efficiently managed as usage pattern is being transferred to a remote server for data mining. Some of these issues are addressed below:

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 behaviour-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 [14].

Privacy concerns

It is obvious that the generation of power saving profiles through user pattern analysis has to be privacy aware. It was 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 behaviour 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 [14].

Security attacks

There could be several possible cases of security attacks on the Android devices based on usage pattern. Consider a case where the battery monitoring app published in Google Play 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 use 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 [14].

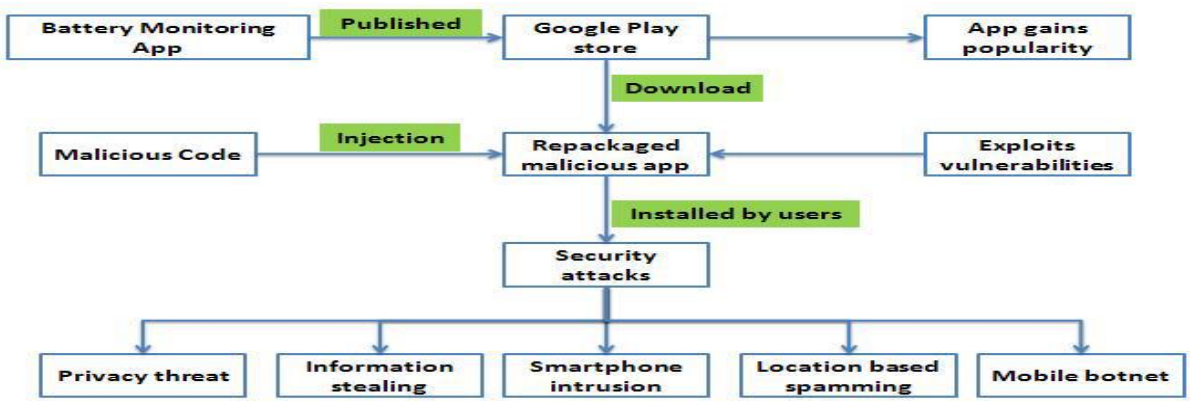

Figure 3: Privacy threat and security attacks based on usage pattern of smartphone and tablet users [14].

Android app with learning engine

This approach would 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 behaviour 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 behaviour. Since, the entire processing is done on the Android device, there is no privacy concern. But the demerit will be that the service providers might not get network usage pattern [15].

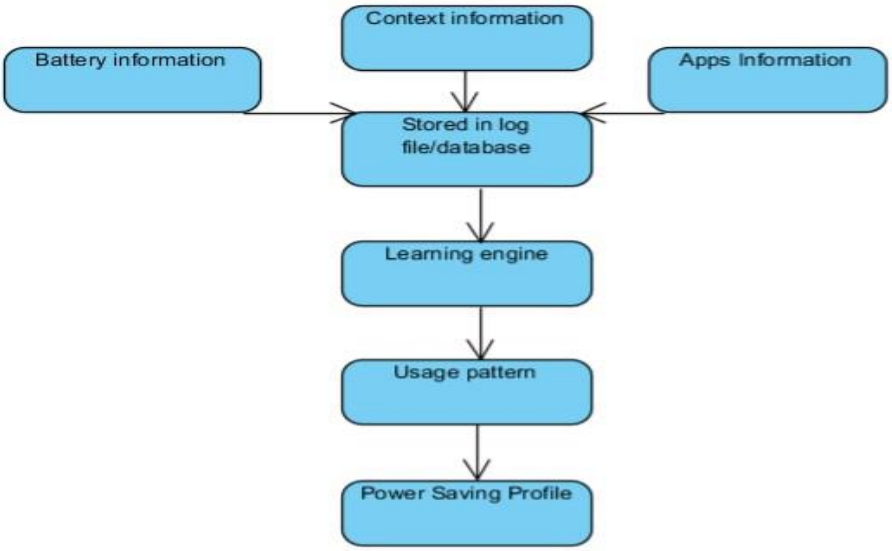

Figure 4: Android app with learning engine [15]

Adding another power source – a photovoltaic cell

In order to add another power source in Android devices, a photovoltaic cell could be integrated. WYSIPS (What You See Is Photovoltaic Surface) 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. This concept could potentially increase the battery life [16].

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 [16].

RESEARCH ANALYSIS AND IMPLEMENTATION

For a proper modelling description and clarity of the proposed research methodology of this research study, state machine diagram of the client server for android power management was

designed and transformed into finite automata (DFA and NFA) to verify and validate the power states of android smartphone. This can help to ensure the correctness of the functions proposed for each state and reduce the defect that might lead to the failure of the software when developed. However, a model is an abstract representation of a system, constructed to understand the system prior to building or modifying it.

UML STATE DIAGRAM

State diagram is used to describe the dynamic behaviour of an individual object as a number of states and transitions between the states; it describes all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system. State diagrams are used to demonstrate the behaviour of an object through many use cases of the system [23]. A state is a condition during the life of an object which satisfies some conditions, perform some activities, or wait for some external events. Below is the state diagram of the proposed research methodology on power management on android smartphones:

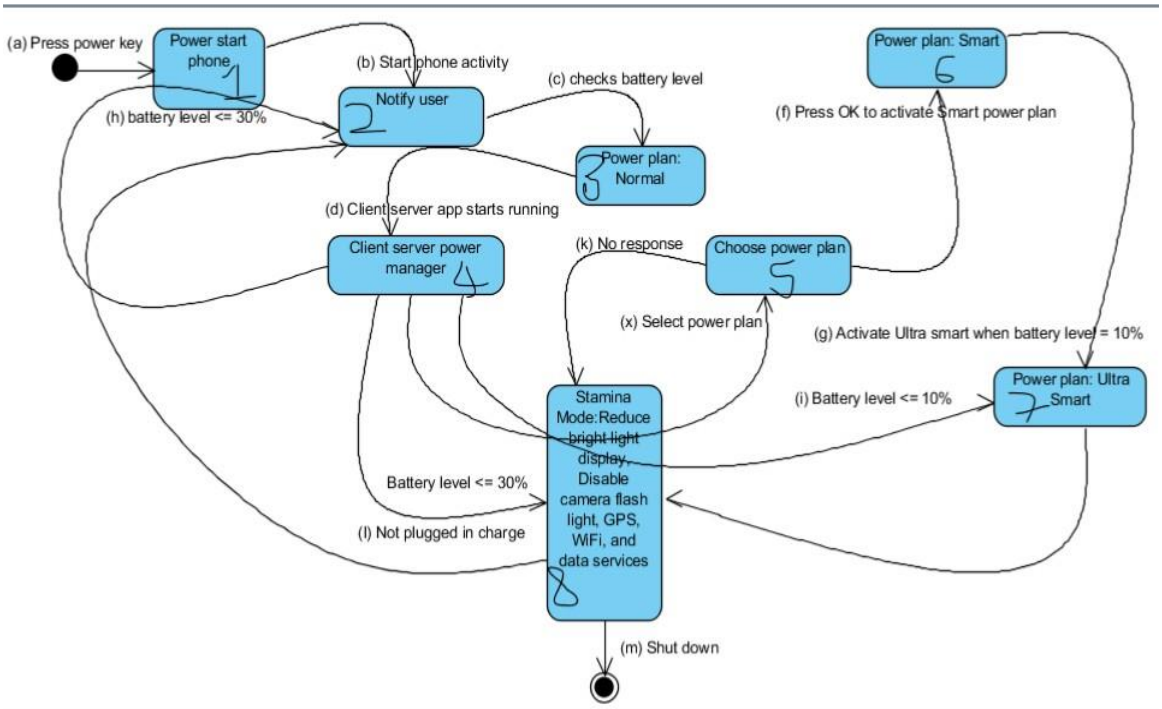

Figure 4: State machine diagram of the proposed client server for android power management

Table 2: ***Description of states and Inputs***

S/No	States and Inputs	Name of states	Description of inputs
1	q_o / a	User end	User executes "input a" by pressing device power key which leads to state "q_1"
2	q_1 / b	Power start phone	Device executes "input b" and starts to boot, loads phone activities and moves to the next state "q_2"

3	q_2 / c	Notify user	Device executes “input c” and notifies the user on running activities plus checks the battery percentage level and goes to the next state “q_3”
4	q_3 / d	Power plan: Normal	Device executes “input d” and activates normal power plan when client server starts running and moves to state “q_4”
5	q_4 / h, j, i, x	Client server power manager	This state checks device power functions and takes control. If “input h” is True it goes to state q_2 and checks again if no response it executes “input j” If True it goes to state q_8 Else it executes “input i" and checks if True it moves to state q_7 and automatically activates the Ultra Smart power plan Else it executes “input x” to select a desired power plan at state q_5
6	q_5 / f, k	Choose power plan	At this state the device executes “input f” and if OK key is pressed it activates the Smart power plan at state q_6 else if no response it executes “input k” which lead to state q_8
7	q_6 / g	Power plan: Smart	At state q_6, as activities keeps running on the device, it executes “input g” and if true at any point in time it moves to state q_7
8	q_7 / Ɛ	Power plan: Ultra Smart	While the device activities are still running at this state with little battery percentage, it executes no input (empty) and hence goes to state q_8 to ensure all the device social functionalities are disabled before the device is plugged in DC outlet and powering off at 0%
9	q_8 / l, m	Disable functions	As the stamina mode automatically disables the social functionality connectivity such as Data services, background running applications, WiFi, GPS, and also reduces the colourful bright light display to its lowest minimum when the user is unaware of his/her battery level at 30% the device would execute “input l” which would lead to state q_2 to notify the user to select a more proficient power plan else the phone

			would power off by executing “input m” which leads to state q_9
10	q_9	Device shuts down	

NONDETERMINISTIC FINITE AUTOMATA (NFA)

A non-deterministic finite automaton has the power to be in several states at once. This ability is often expressed as an ability to guess something about its input. NFA may have multiple or no following transition for a given state and input. A finite automaton is nondeterministic if each state is not unique. NFA also has a fixed number of states but can be in multiple states at one time. A Nondeterministic finite automaton A is often abbreviated as NFA. It is defined as a “five tuple” notation: A = (Q, $\sum$, δ, q_0, F) where

- A is the name of the NFA
- Q is its set of states
- $\sum$ is its input symbols
- q_0 is its start/initial state
- F is its set of accepting states
- δ is its transition function, it is a function that takes a state in Q and an input symbol in $\sum$ as arguments and returns a subset of Q. The difference between an NFA and a DFA is in the type of value that δ returns: a set of states in the case of an NFA and a single state in the case of a DFA [17].

NFA generated using visual automata simulator tool for the proposed research methodology of power management on android smartphone:

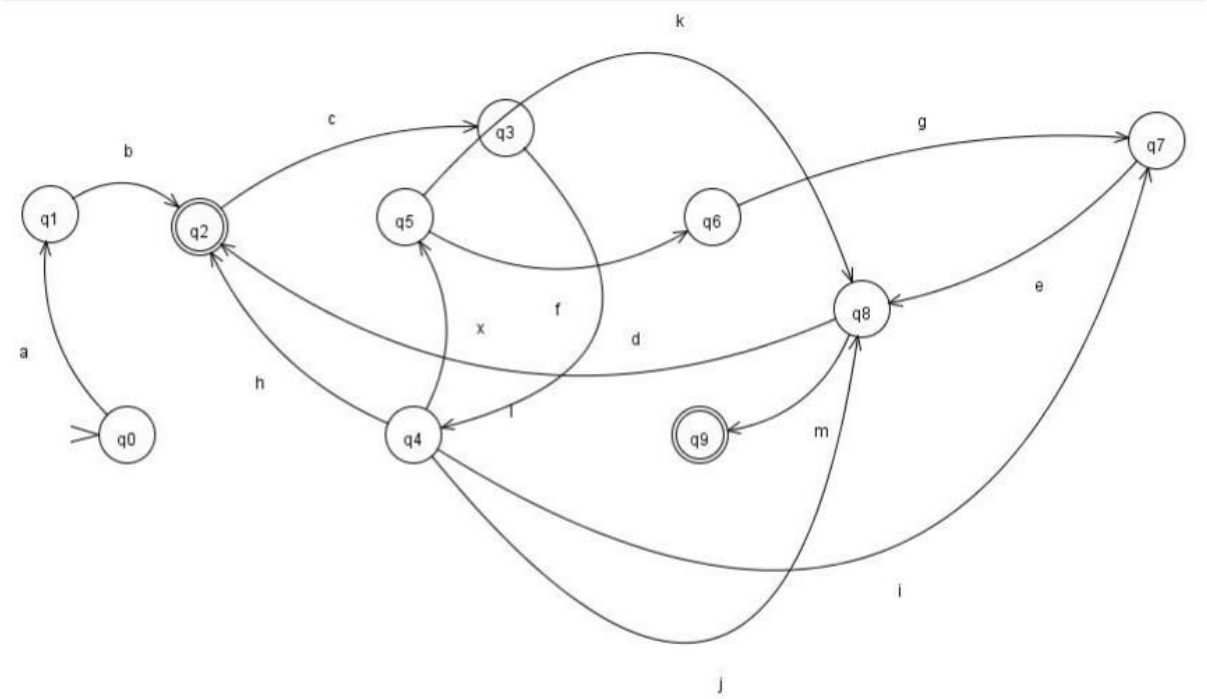

Figure 5: NFA for the proposed research methodology of power management on android smartphone

Table 3: Transition table derived from the NFA above

δ / ∑	A	B	C	D	X	F	G	H	I	J	K	L	M	Ɛ
⇨**q0**	q_1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
q1	ϕ	q_2	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
q2	ϕ	ϕ	q_3	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
q3	ϕ	ϕ	ϕ	q_4	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
q4	ϕ	ϕ	ϕ	ϕ	q_5	ϕ	ϕ	q_2	q_7	q_8	ϕ	ϕ	ϕ	ϕ
q5	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	q_8	ϕ	ϕ	ϕ
q6	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	q_7	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
q7	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	q_8
q8	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	q_2	q_9	ϕ

DETERMINISTIC FINITE AUTOMATA (DFA)

The term "deterministic" refers to the fact that for each input there is one and only one state to which the automaton can transition from its current state and each of the state is unique. DFA has a fixed number of states and can only be in one state at a time. A deterministic finite automaton A is often abbreviated as DFA and is defined as a "five tuple" notation: A = (*Q, ∑, δ, q_0, F)* where

- A is the name of the DFA
- *Q* is its set of states
- ∑ is its input symbols
- *δ* is its transition function
- q_0 is its start state
- *F* is its set of accepting states [23]

As explained above, the state of a computer system can be described with DFA. It is supposed that there are p states in set Q and there are q transition conditions in set ∑. Then there are p state nodes at most in the corresponding DFA. Each state node can be transferred to q neighbor nodes at most. The whole state transition procedure of a computer system can be described directly with the state transition diagram [17].

Corresponding DFA automatically generated from the NFA by Visual Automata Simulator tool:

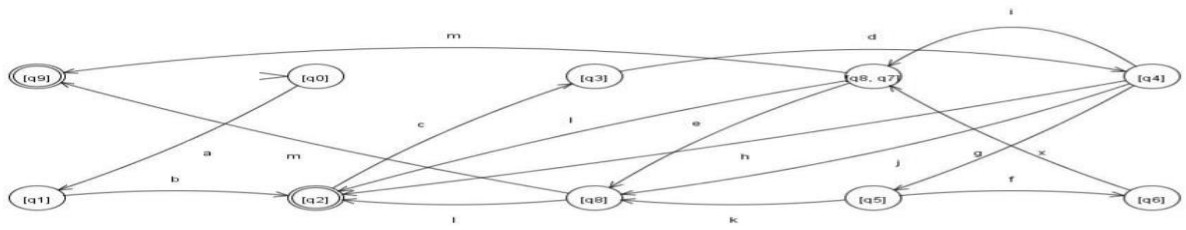

Figure 6: DFA automatically generated from the NFA by Visual Automata Simulator tool

CONCLUSION:

Smartphones are changing people's daily lives and the Android-based smartphone has become one of the most populous mobile telecommunication devices in the world. To make best use of such smartphones, battery efficiency is critical. Although Android devices have aggressive power management capabilities derived from the standard Linux kernel, software applications and services are optimized to run in an energy-efficient way. CPU power optimization approaches are needed to tune software on Android mobile systems. Tuned software applications and services request CPU resources only when necessary so devices remain at low power states as much as possible thereby increasing energy efficiency and ultimately battery life. The development of longer lasting batteries and low power consumption components is undoubtedly at its highest peak.

References:

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

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

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

[4] Steve Guo, "Android Power Management" 2011.

[5] 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.

[6] Rabaey, J. "*Low power design essentials" 2010.*

[7] APM V1.2 spec.

[8] Advanced Configuration and Power Interface (ACPI), http://www.acpi.info/

[9] https://play.google.com/store/apps/details?id=com.dianxinos.dxbs&hl=en

[9] https://play.google.com/store/apps/details?id=com.ijinshan.kbatterydoctor_en&hl=en

[11] https://play.google.com/store/apps/details?id=com.easy.battery.saver&hl=en

[12] https://play.google.com/store/apps/details?id=com.latedroid.juicedefender&hl=en

[13] https://play.google.com/store/apps/developer?id=MacroPinch&hl=en

[14] 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.

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

[16] 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.

[17] 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.

[18] N. Vallina-Rodriguez and J. Crowcroft, "*Energy Management Techniques in Modern Mobile Handsets*", IEEE COMMUNICATIONS SURVEYS, February 2012.

[19] 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.

[20] 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.

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

[22] 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.

[23] http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/FiniteAutomata.pdf