A Personalized Profile Based Learning System for Power Management in Android

by

ASHWIN ARIKERE

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
MAY 2015

Acknowledgements

I would like to thank the folks whose love and encouragement has made the this dissertation possible.

First, I would like to thank and express my deepest appreciation to Dr. Roger Walker whose direction, patience, support, inspiration and positive feedback made this project possible. I met Dr. Walker in my second semester here at the University of Texas at Arlington as a Master's student and I have been fortunate to work with him ever since. The passion with which he approaches his work is something I have never seen before and am happy to learn from. The vast amount of invaluable counsel and knowledge that I have been fortunate to receive from him will always be treasured in my life. Thank you Dr. Walker, for guiding me and more importantly believing in me.

Second, I would like to thank my team at Intel Corporation for providing an opportunity to work as an intern multiple times. The support given to pursue my research work while being an intern has been extraordinary . The work ethos, knowledge, and company of my colleagues were instrumental in getting me to this point. Thank you Tom, Sharad, Fernando and Yamini.

I would especially like to thank my committee members Dr. Ramez Elmasri, Dr. Hao Che and Mr. David Levine for their valuable guidance and feedback.

I would also like to thank all the colleagues at the Transportation Instrumentation Lab (TIL). Working with all of you over the course of our studies has been truly a pleasure.

I would like to thank my parents, Lakshmi and Ganapathi Arikere, for supporting me from half the world away. I thank you both for all the love you have showered up on

me, supported me through everything I have been through and can only hope I have made you proud .

Finally I would like to thank my wife Teju for being my rock. Your support and understanding right from the time I started this journey has been nothing but unwavering. I wouldn't have been able to complete this without you. You have always pushed me to be the best I can be. It has been a long trek to get here, but I think nothing can beat this view! I love you!.

April 15, 2015

Abstract

A Personalized Profile Based Learning System

for Power Management in

Android

Ashwin Arikere, PhD

The University of Texas at Arlington, 2015

Supervising Professor: Roger Walker

Mobile computing devices are becoming more ubiquitous everyday due to the phenomenal growth in technology powering them. With the amount of computing power available in these devices, users are capable of achieving a multitude of tasks that were only possible with a PC just a few years ago. However, these devices still face issues regarding power management. Battery technology has not kept pace with the development in other areas. With a limited supply of energy, the mobile device of today requires a fine balance of power management to provide adequate energy to support the heavy duty computing of the user while simultaneously enabling the device to stay alive for a long duration.

With such limitations, the onus is more on the user to limit his usage of the device and its features to conserve power, thus having a crippling effect on the user's operation of devices. This research effort analyzes how devices are used and explores the effect of demographics on power consumption. We also propose a solution which will adapt to the individual user and provide a customized power saving mechanism tailored to the user's usage of his/her device. The adaptive system will learn what type of apps

are used by the user and can intelligently make decisions to conserve power based on prior learnings. It is estimated that such a mechanism will have an improvement on battery life by 15%.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Introduction

Smartphones are fast becoming an integral part of everyday life. It is estimated a total of 6.9 billion devices/phones are present worldwide with increasing mobile adoption. [82] The availability of computing power and access to the internet in a form factor small enough to be carried in one's pocket makes the ownership of a device very attractive and irresistible. The developer community has risen to the occasion by creating a wide variety of applications for use on these smart devices. As of 2014, the total number of applications for Android on the Google Play store alone stands at 1.43 million [84]. With such impetus, smartphones are the new PCs of this generation.

As technology continues to develop at a rapid pace in the mobile arena, the load on energy storage mechanisms has increased. Driven by market, more powerful hardware such as multicore CPUs, larger displays etc. have been included on these mobile devices. These power hungry components, coupled with applications that are power hungry, exacerbate the power problem. While a variety of hardware optimizations have been proposed and developed [85],[86] to improve the power performance of the underlying silicon, an effort to approach the problem from a software perspective has also been undertaken. A combination of hardware and software optimizations can lead to optimal results from a power and a performance standpoint. Utilizing the "smartness" of the device to implement such innovative solutions is one such strategy.

1.2 Contributions of this dissertation

This research effort addresses an area of power management in mobile devices by considering the effects of the variety in use of the devices. A detailed analysis of how different people of different demographics use their smartphones and how this affects power consumption is also explored. Custom profiles were designed to be indicative of the type of applications used by people of different demographics. These profiles were also tuned to be representative of not just the applications used, but also how long each of those applications were used. These profiles were then tested and detailed power measurements were carried out in an attempt to understand how variation of system parameters affect power consumption, and how much of this impact is due to the variability of use across people of different demographic groups.

Based on the learnings above, a custom solution was designed and developed to maximize energy savings in mobile devices. This was achieved by having the device learn about the individual user, building a usage profile from the learned data and applying power management policies appropriate to the learnt pattern in order to maximize power savings. The learning system was also designed such that users see minimal impact from a performance standpoint. The learning system proposed utilizes a Naive Bayesian classifier to detect and learn the usage patterns. The system was then tested on the demographic representative profiles to demonstrate the feasibility of such an approach.

1.3 Outline

The rest of this thesis is organized as follows. Chapter 2 gives an introduction on power management in Android. A description of the overall architecture and design of the power management policies is followed by some of the enhancements introduced into Android as part of Android 5.0 (Lollipop). Chapter 3 covers the related work performed by both academia and the industry. First, the work done to improve power management in mobile devices from both a device and component level is described. A survey of some of the popular applications available on the Google Play Store designed for power management is then presented. This is followed by some of the innovative device specific solutions developed by the industry for power management. Chapter 4 discusses the impact of demographic groups on power usage in mobile devices and how variation in power consumption can be observed. Chapter 5 presents an overview of supervised learning algorithms and their applicability to this research effort. Chapter 6 describes the design and implementation details of a learning power management system. Chapter 7 discusses the results of the system described in Chapter 6. Finally, Chapter 8 concludes the dissertation. The appendix presents a comparison of the two runtimes developed for Android from a power standpoint.

Chapter 2

Introduction to Power Management in mobile space and Android

Power management is a crucial mechanism for controlling power use in computers. This feature allows for the system to be set to a low power state when inactive, as well as provides control on how to wake a device when needed. An early API specification for providing power management in IBM compatible personal computers called Advanced Power Management (APM), was developed by Intel and Microsoft in the year 1992. This allowed the operating system to work in cohesion with the BIOS to achieve power management. It provided CPU and device power management when system was in an idle state. The core control however rested with the BIOS and thus the OS had no knowledge of what APM was doing.

The APM standard was superseded in 1996 by the Advanced Configuration and Power Interface (ACPI). This open standard developed jointly by Intel, Microsoft, Toshiba and others brought power management under the control of the operating system as opposed to the BIOS centric APM specification.

This specification allowed the operating system to manage all aspects of power management and device configuration in a PC. Proper power management allowed for lower power consumption and reduced heat dissipation which helped increase system stability. Other benefits included lower operational costs and increased battery life in portable systems.

With the increase of laptops as primary computing devices, focus on saving power increased and this led to improved power control from the OS. For example, Linux machines started to support newer kernel settings to account for such devices. These

settings allowed the OS to check the source of the power supply i.e. main line or battery and accordingly set parameters such as CPU, video, display etc. Windows allows users to configure these parameters through the power management options in the Control Panel. Battery management was equally important.

These options however are not sufficient when dealing with the energy demands of newer technology. Computing power of mobile devices has increased exponentially leading to greater functionality. Current mobile devices support multiple functions ranging from the basic phone call to more advanced functions such as media players, GPS navigation, browsers and more. The constraints of size on these devices coupled with slower improvements in energy storage methods resulted in the available computing power becoming an expensive resource, one that needs to be carefully managed. This resulted in improved OS designs being employed along with more efficient hardware.

2.1 Power Management in Android:

Android, open source mobile operating system developed by Google, is currently one of the largest mobile OSes available in the market. Although Android is based on the Linux kernel, Android uses its own power management system. Android systems provide their own infrastructure for power management via an API called PowerManager. The basic is: if there are no applications or services requiring power, the CPU should not consume any power. Android's design requires that applications and services request CPU resources with "wake locks" through the Android application framework and native Linux libraries. If there are no active wake locks, Android will shut down the CPU. The image below illustrates the power management architecture in Android.

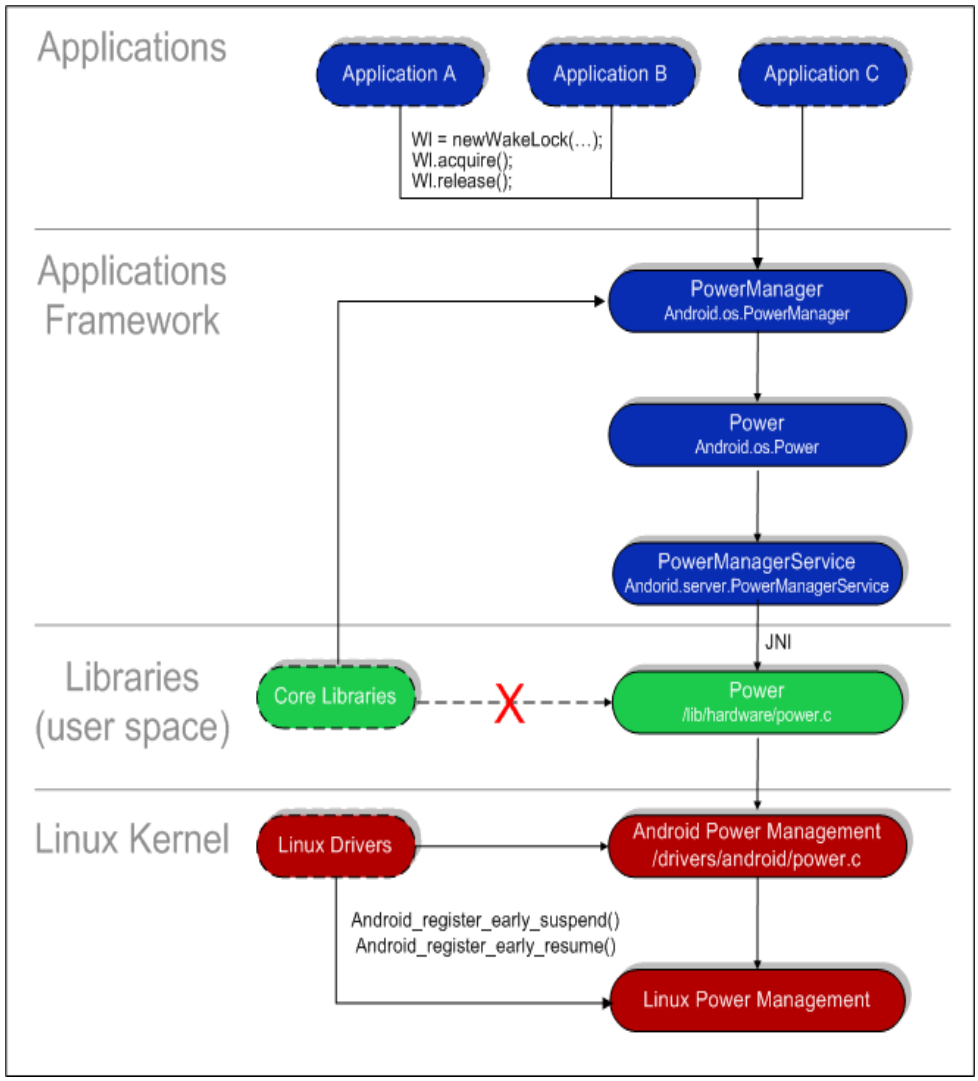

Figure 2-1 Android power management architecture

Wake locks are used by applications and services to request CPU resources.[79], [80] 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 causes large interrupt latencies or that which disable a set of interrupts, will not be entered from idle until the wake locks are released. [21]

An application wanting to use CPU resources will thus acquire a wakelock before it can consume CPU resources.

All power management calls follow the same basic format:

- Acquire handle to the PowerManager service.
- Create a wake lock and specify the power management flags for screen, timeout, etc.
- Acquire wake lock.
- Perform operation (play MP3, open HTML page, etc.).
- Release wake lock.

Because Android development has been an incremental cycle, better power management features have been introduced with the newer versions. Android is designed to manage memory (RAM) such that power consumption is at a minimum, in comparison to desktop operating systems. When an Android application is no longer in use, the OS will keep the application suspended in memory; while the application is still technically "open", applications in a suspended state consume no resources (battery power or processing power) and sit idle in memory until needed again. This increases the general sensitivity of Android devices, since applications do not need to be shut down and reopened each time and also ensures that background applications do not use unnecessary power. Android 2.3 (Gingerbread) took a more active role in managing apps that were keeping the device awake for too long or that were consuming CPU while running in the background. [9]

Android 4.4 (KitKat) saw the introduction of hardware sensor batching. This optimization can dramatically reduce power consumed by ongoing sensor activities. Android works with the device hardware to collect and deliver sensor events efficiently in batches, rather than individually as they are detected, letting the device's application processor remain in a low-power idle state until batches are delivered. KitKat also saw the introduction of newer Bluetooth profiles which support low-power interactions with peripherals. [10]

With Android 5.0 (Lollipop), Google introduced Project Volta, which is a suite of enhancements designed to improve battery life.

2.1.1 Android RunTime (ART)

The biggest change to Android since its inception, ART was introduced in Android 5.0 Lollipop. Android previously used Dalvik, a virtual machine (VM) designed to run on systems which are constrained in terms of memory and processor speed. Dalvik was designed to run programs in an interpreted manner i.e. it used a just-in-time compiler to run apps/programs. While great from a developmental perspective, using a JIT implies there needs to be some processing power spent purely to translate code to machine instructions. This from a power stand point is inefficient as every launch of the program requires a new compilation. ART is an improved VM that has many feature improvements over Dalvik, with the most significant being the ahead-of-time compilation of apps. This means that on install of an application it needs to be compiled only once, thus reducing the total amount of time the processor spends on compiling code. ART also includes an improved garbage collection mechanism which runs faster as well as lesser number of times. Therefore the introduction of ART has improved both power and performance in

mobile devices. An analysis of Dalvik v ART from a power perspective is included in the appendix

2.1.2 Battery Historian

Lollipop also saw the introduction of Battery Historian, a HTML visualization tool aimed at developers which generates statistical data about power-related events on a device. The tool runs on the output of a dumpsys command which gives details such as

- History of battery related events.
- Global statistics for the device.
- Approximate power use per UID and system component.
- Per-app mobile ms per packet.
- System UID aggregated statistics.
- App UID aggregated statistics.

Figure 2-2 HTML visualization of Battery historian

These statistics allow developers to visualize what is consuming the device's battery, to what extent and at what times, thus enabling developers to pinpoint and diagnose power related issues.

2.1.3 JobSchedulerAPI

In Lollipop, a new API was provided to enable developers to define conditions for background jobs for the system to perform. This allows the OS to batch together multiple tasks and run them together when those conditions are met. This prevents the processor from being woken up in the middle of its sleep for something that can be safely delayed for later, thus reducing overall power consumption. The scheduler is also smart enough

not to allow tasks to run when the resource required isn't available, like network updates when there is no network.

2.1.4 Battery Saver Mode

Android 5.0 also builds in a power saving mode called Battery Saver. This feature is turned on automatically once the battery drops below a threshold value. The saver lowers the processor's speed, reduces animations, dims the screen, and reduces radio usage.

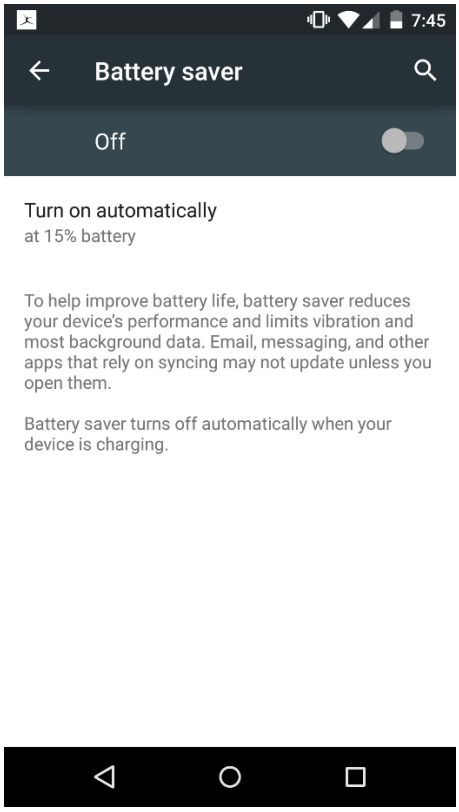

Figure 2-3 Battery Saver feature on the Nexus 5

Chapter 3

Related Work

The previous chapter introduced some of the power saving features of Android. While majority of the features introduced in Android 5.0 bode well from a power perspective, only 3 % of the devices have been updated with this latest version. A large number of devices (about 83%) still run either Jellybean (4.0-4.3) or KitKat (4.4) versions of Android [15]. The features introduced in Lollipop target only some of the issues plaguing power management. While mitigating a number of problems from an OS standpoint, the problem of mismanaged power will exist as long as developers prioritize performance in their apps. Ageing battery technology coupled with the advent of faster cellular networks such as 4G, data and location interfaces that are power heavy, larger and brighter displays only exacerbates the power management problem.

Consumers consistently rate battery life as one of the most important features looked at in a phone while making a purchase [16], [17], [18]. With such a premium placed on battery life, proper energy management in phones is vital. Researchers and developers alike have tried to tackle this issue with varying success. The works can be categorized as follows: 1. Work on profiling usage patterns and device components, 2. Work on improving energy efficiency of onboard components, 3. Developer solutions

3.1 Work on profiling usage patterns and device components

One of the ways researchers have approached the area of energy management is to understand how user-phone interaction in the real world. Knowing how various components behave helps developers build applications that are smarter in their resource

usage. Studying usage patterns can also highlight unintended ways an app or device component is being used.

The authors of [19] show how usage information collected from all smartphone users exhibit unique device usage patterns. They also describe a technique to utilize usage patterns to predict battery life and show that their results can be used to detect abnormal battery states as well as influence network selections. MyExperience [22] captures traces from users' smartphones by combining passive logging of device usage as well as context-triggered user experience sampling. It presents several case studies demonstrating how people use and experience mobile technology ranging from battery life and charging behavior to studies on SMS usage and mobility. In [20] the authors present a fine-grained energy profiler *eprof* for smartphones. The profiler reveals several wakelock bugs caused due to improper handling of wakelocks by developers. The profiler also confirms that majority of the energy spent by apps are by I/O interfaces such as 3G, WiFi and GPS. The authors of [23] evaluate how battery life is affected by network applications running in the background and can reduce energy efficiency of an iPhone by up to 72%. They show that using a lower energy data link such as the WiFi can increase efficiency by up to 59%. Wang et al. [31] develop a statistical method using coarse grained battery traces to estimate power consumption for each mobile application. This however requires a large data set to reduce error of direct estimation.

3.2 Work on improving energy efficiency of onboard components

Profiling device components and usage patterns allows us to understand how apps utilize power. Using this information, we can target specific components and improve their efficiency. The most commonly targeted components include data and location interfaces, CPU, and display.

Lee et al. [28] propose a new location logging mechanism to maximize battery life without compromising quality of logs. Their method involves use of a Variable Rate Logging (VRL) to reduce constant location logging by detecting if the user is indoors or at a standstill. The authors of [27] propose an intelligent duty cycling system for the GPS along with other sensors to aid location detection. Their approach utilizes a combination of geo-fencing and well-curated estimation to turn on sensors, thus reducing power. Ra et al. [29] suggest a method to decide which wireless interface to use for data transfer and tries to delay data transfer when possible until a lower energy connection is available. In [24] the authors develop a power estimation model to estimate system power consumption. They conclude that the CPU and screen tend to dominate active power components and propose a mechanism to modify these parameters over time. They also propose to reduce the screen brightness gradually using the concept of change blindness, thus reducing the power draw by the display without affecting user satisfaction. However, their approach is not application sensitive and can work detrimentally to lower QoS. Another approach proposes some form of user aware CPU frequency scaling which measures rate of change of pixel densities in the display.[26] However, this targets desktop and laptops and does not accounts for mobile devices.

Datta et al. discuss various research paths in smartphone power management. They propose a few methods to prolong battery life by incorporating some form of analysis of battery records and improve capacity by adding photovoltaic cells that can be integrated on top of displays that can charge the device [21]. In [30], the authors develop a user-guided tool called BatteryExtender, to enable device reconfiguration based on the workload. They claim to decrease energy consumption by about 10-20%. Other methods proposed include selective offloading of energy consumption hotspots to the cloud [32] as

determined by a runtime mechanism which can estimate energy savings and also decide on the offloading strategy.

3.3 Developer solutions

The previous two sections gave an overview of the research undertaken by the academic community. However, as the issue of power management in smartphones is of vital importance to the average consumer, multiple developers have tried to solve the problem in their own ways. A quick search of Google Play Store, the Android app market, for the key words "battery saver" brings up more than 100 apps. Each of these apps claim to extend battery life without any user interaction. Following is a brief look into some of the top apps and their features.

3.3.1 JuiceDefender

JuiceDefender, [33] a product developed by latedroid in early 2010, was one of the first apps developed for power management. Designed initially for Android 2.1 and updated till Android 4.0.3, the app has minimal need for user interaction. It has multiple automated profiles which give the user a fair amount of options in controlling a phone's features. JuiceDefender allows users to configure the toggling of the different data and location interfaces on a device based on certain criteria such as location. It also allows for CPU throttling if the phone is rooted and the specific kernel used on the device supports it. The various profiles can also target specific apps and set schedules for syncing. The app is available for free in the market, however limits the more advanced features such as app based settings, CPU control, location and time triggers to the paid version of the app. The custom mode in the app allows the user to have fine grained control over power management.

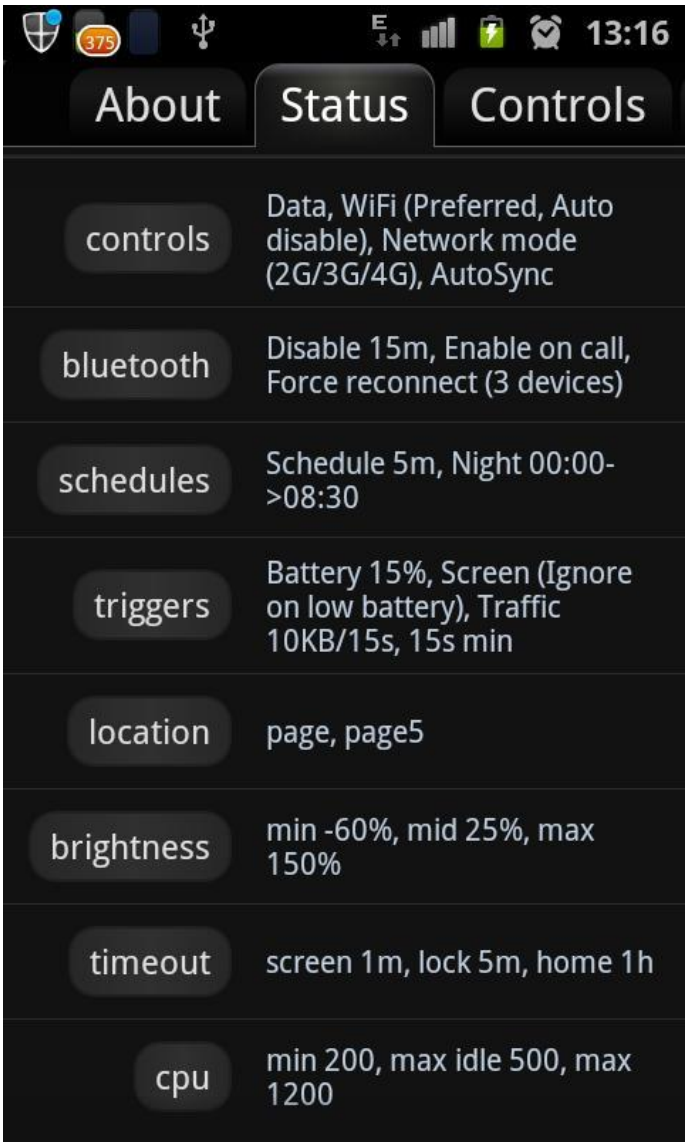

Figure 3-1 Status page on the JuiceDefender app

3.3.2 DU Battery Saver

Another app similar to JuiceDefender, DU Battery Saver [35] offers simple options to the user to toggle between a few set modes. These modes generally work by

limiting the data interfaces as well as setting the screen brightness to pre-determined levels. The app also allows the user to build a custom mode and configure certain settings such as data interfaces. The app has a single optimize button which basically kills apps running in the background which it considers to be causing a battery drain. It differs from other apps as it claims to also promote battery health by only requiring the user to charge the battery after a set level. The app also has a reminder feature to indicate when power draining apps are left running in the background. Protection lists are also included to allow the user to override certain apps from being shut down by DU Battery Saver. Overall, once the app is installed, the amount of interaction needed by the user is kept to a minimum. While the DU Battery Saver tries to save power by killing power hungry apps, the inherent structure of Android actually makes this more of a drain on the system as it (the system) tries to recreate the killed apps, thus consuming more power.

Figure 3-2 DU Battery Saver app

3.3.3 Battery Doctor

The top ranked battery app in the Play Store, Battery Doctor [34] has a similar interface as DU Battery Saver. It differs from DU Battery Saver by including an app list it considers to be power hogs. Clicking on any of the apps in the list allows it to be pushed to an ignore list. The mechanism of optimizing remaining charge on the phone is also similar as it works by killing the apps. Battery Doctor also fails at understanding the native structure of Android by using an app killing mechanism. It also includes a history graph and a summary section indicating where the majority of power has been spent since the last full charge. Another feature is a charge history section which notes the various charging states for the device i.e. fast charge and trickle charge states. The app also

includes a scheduling mechanism to automatically switch between modes based on time of day.

Figure 3-3 Settings on the Battery Doctor app

3.3.4 Tasker

While the previous apps had an explicit focus on power management, Tasker [36] is actually a task automation application based on contextual information such as date, time, location etc. The onus on task creation is on the user. The task is then executed based on the context using an "if this then that" model. The level of customization allows for an infinite number of tasks to be done based on user defined

contexts. The drawback is that the amount of time the user needs to spend to achieve a specific task is large. The highly customizable nature of Tasker's interface enables complex tasks to control various onboard sensors, data interfaces etc. based on contextual information to be developed. Once a task has been set, power hungry sensors and interfaces can be toggled without active user input as required.

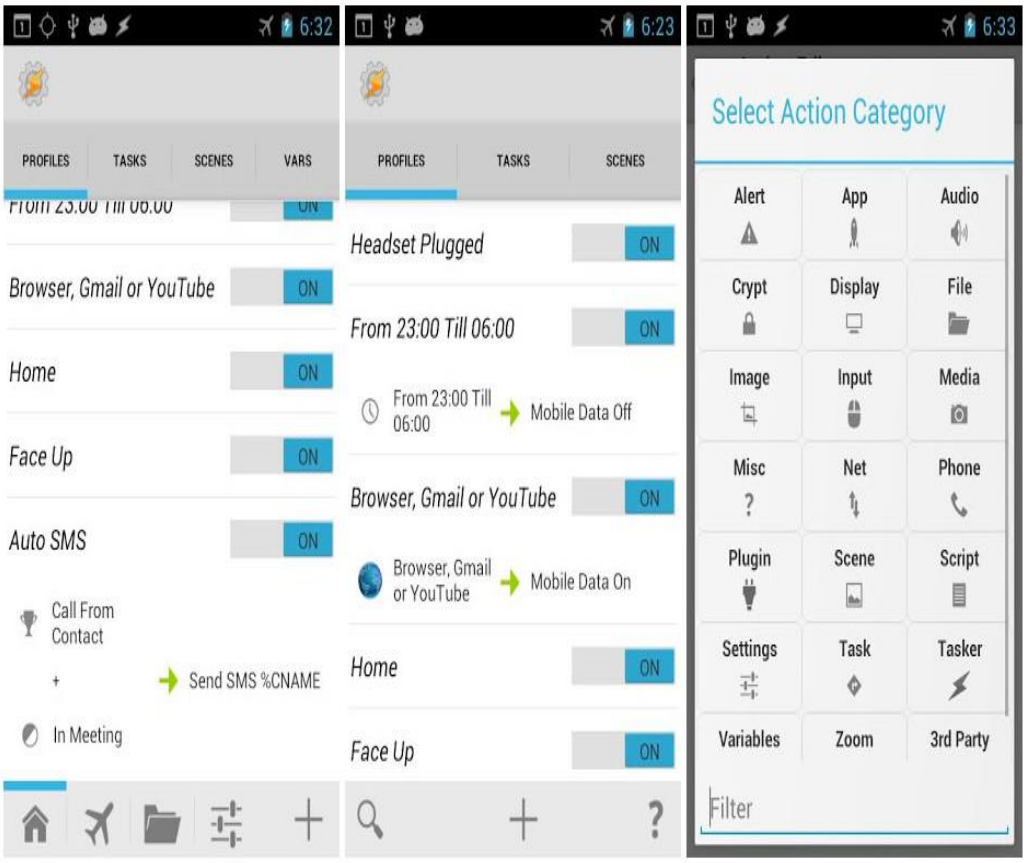

Figure 3-4 Steps to create a task in Tasker

3.3.5 Device Specific Solutions

Before Android 5.0, there existed no native power saving application or setting. The system would behave exactly the same way as it did, whether the battery level was at 100 or 10. Device manufacturers have now included some native settings in software which work in conjunction with the hardware to achieve better power savings than possible with pure software. These settings can be activated at preset battery levels and minimize power draw further.

3.3.5.1 Samsung S5

Samsung included an ultra-power saving mode on their Galaxy S5 [37] phones. The mode when turned on manually by the user, switches the phone into a black and white mode. The device uses an AMOLED display, therefore switching to a black and white mode saves some power as all pixels displaying the color black are completely turned off. The phone limits the total number of applications which can be used in this mode to six. The device also changes the mobile data connection to 3G only which is active only if the screen is turned on. Other data interfaces are turned off as well. The mode also disables 2 cores of the quad core CPU and down clocks the remaining 2 cores. The display refresh frequency is also reduced. The smartphone thus is economical in its power consumption at the cost of restricting the user to a limited feature set.

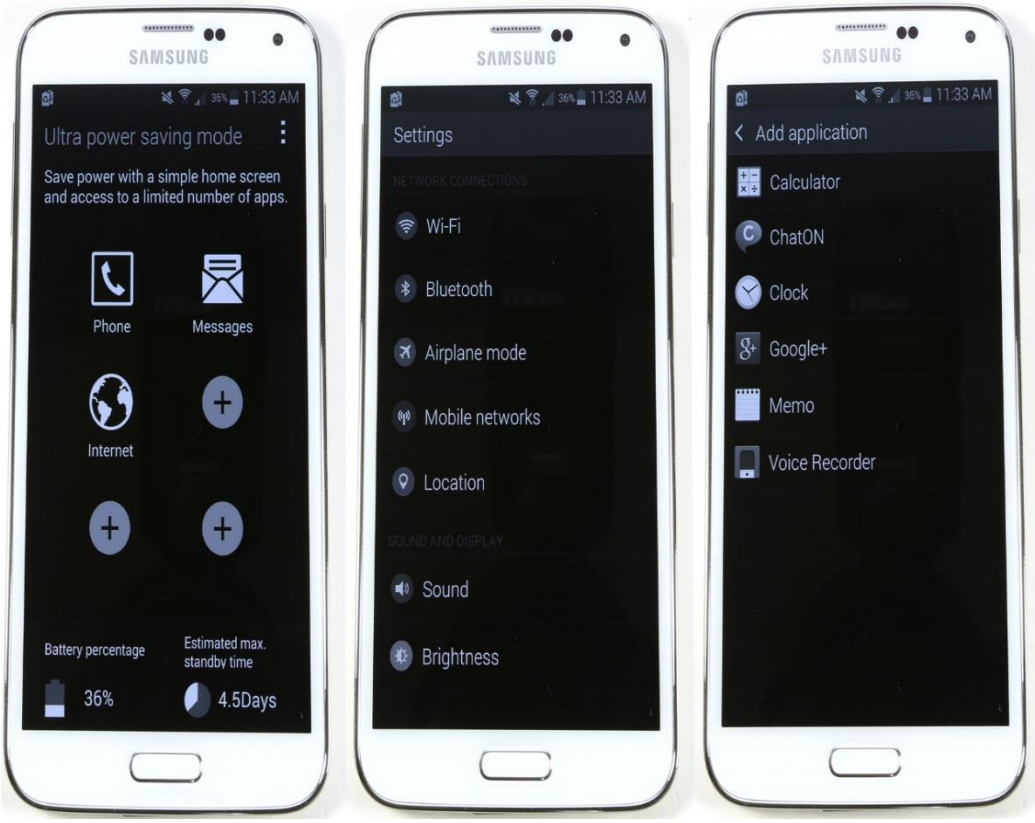

Figure 3-5 Ultra Power Saving Mode on the Samsung S5

3.3.5.2 HTC One M8

HTC also incorporated a dual level power saving mode on their One (M8) phones [40]. While the first level conserves battery by reducing brightness and conserves CPU usage, the extreme mode goes a couple of steps further. It restricts the device to five stock apps only. The feature can be automatically turned on once the battery drops below a set level. The mode turns off all data interfaces when the screen is on. Additionally it throttles the CPU and turns off haptic feedback.

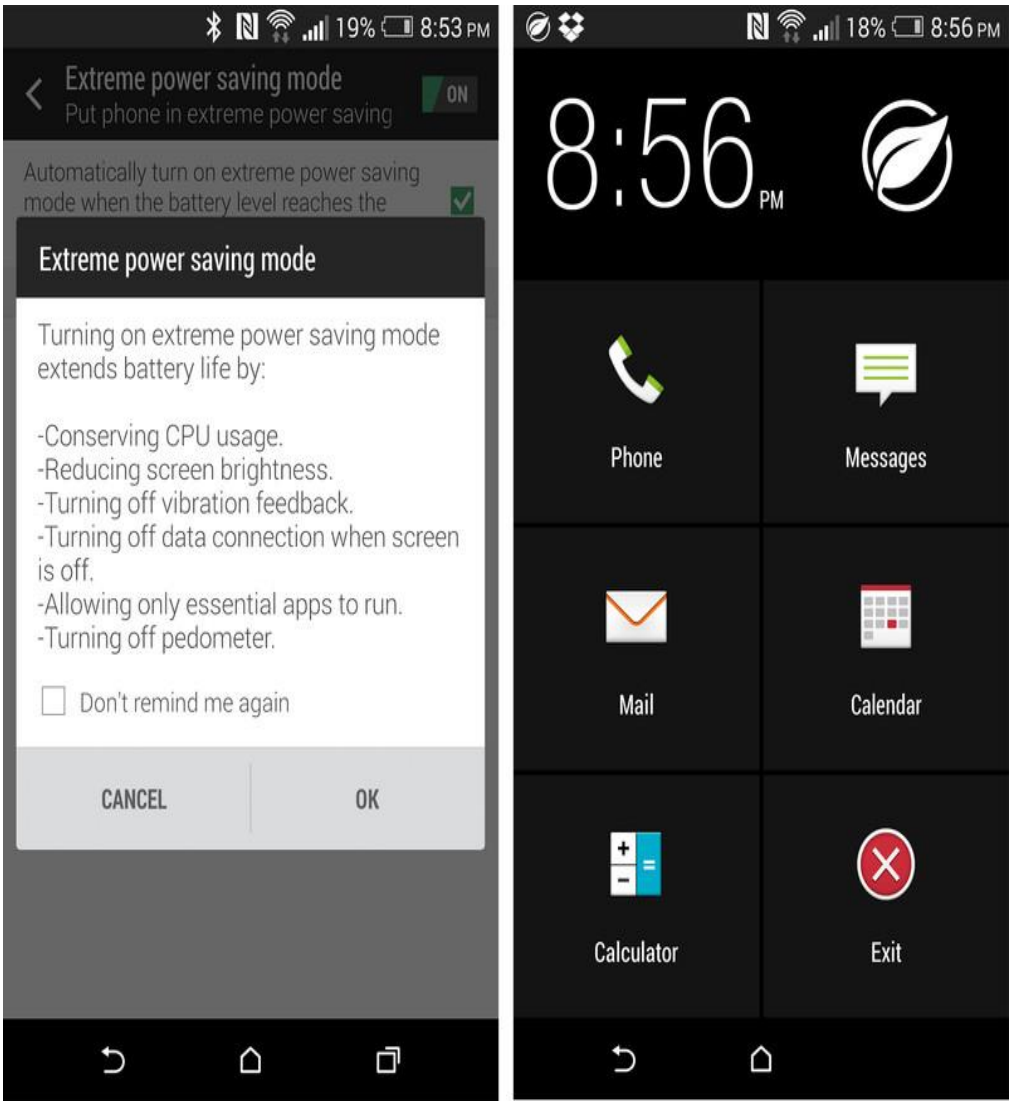

Figure 3-6 Extreme power saving mode on the HTC One (M8)

While both devices incorporate a drastic power saving mode, neither phone uses any form of contextual information to switch to the power saving mode. The modes function as an additional tool for the user in the goal to conserve battery. The interfaces presented to the user in these modes are similar and minimalistic, including just the basic phone and messaging apps. The extreme saver modes on these devices are successful

in minimizing battery use and essentially reduce the functionality of the smartphone to basic communication functions only, thus affecting the quality of service

Chapter 4

Effects of mobile usage on power in mobile devices

This chapter presents the work done on determining how mobile usage varies across users and effects of the varied usage on power. Kang et al. tackle this issue by logging data from 20 or so users and draw statistical conclusions about those individual users based on the various states the device was in. Their observations lend weight to the argument that smartphone users do have unique usage patterns. However their work makes no effort to understand the behavior of the average user and how that type of usage affects power consumption.

Multiple surveys have been conducted asking users how they utilize their mobile devices. Google commissioned many global studies to gain insight about how smartphones are being adopted and used. The data from these studies has been analyzed to understand the impact of demographics on usage. Using this knowledge, multiple workload profiles characteristic of the average user of a particular age group was constructed. These profiles were then used to study their effects on smartphone power. The following sections will first present the information regarding the data collected in the survey, draw out some of the conclusions used to build the profiles and finally show the impact of these profiles on power in a mobile device.

4.1. The survey data

"The smartphone revolution has touched every corner of the world, and that in turn is changing how any digital marketer does business." [41] Google realized that understanding how smartphones are used and adopted is crucial in understanding the mobile user. This information can be used to determine what strategies are appropriate to expand their core business. To this end they commissioned a research effort called Our Mobile Planet. This was achieved by surveying private smartphone users in multiple countries. The survey was administered in three waves, thus accumulating data from 3 different years i.e. 2011, 2012 and 2013, with at least 1000 respondents in each wave. The collected sample data was also weighted using variables of smartphone population. The data collected was then made public by hosting it online, allowing users to download the raw data or use the chart creation tool to pull data relevant to them.

This data was used to understand how mobile behavior is categorized by demography. It was then used to construct profiles representative of the mobile behavior exhibited by the different categories. The data set comprises information collected from multiple countries. For this effort, only data from US users was considered For example, Figure 4-1 shows the video viewing trends for different age groups for the 3 surveyed years. The increased availability of streaming videos has led to a continually increasing trend of viewing videos on the smartphone, a behavior which is CPU intensive and battery heavy.

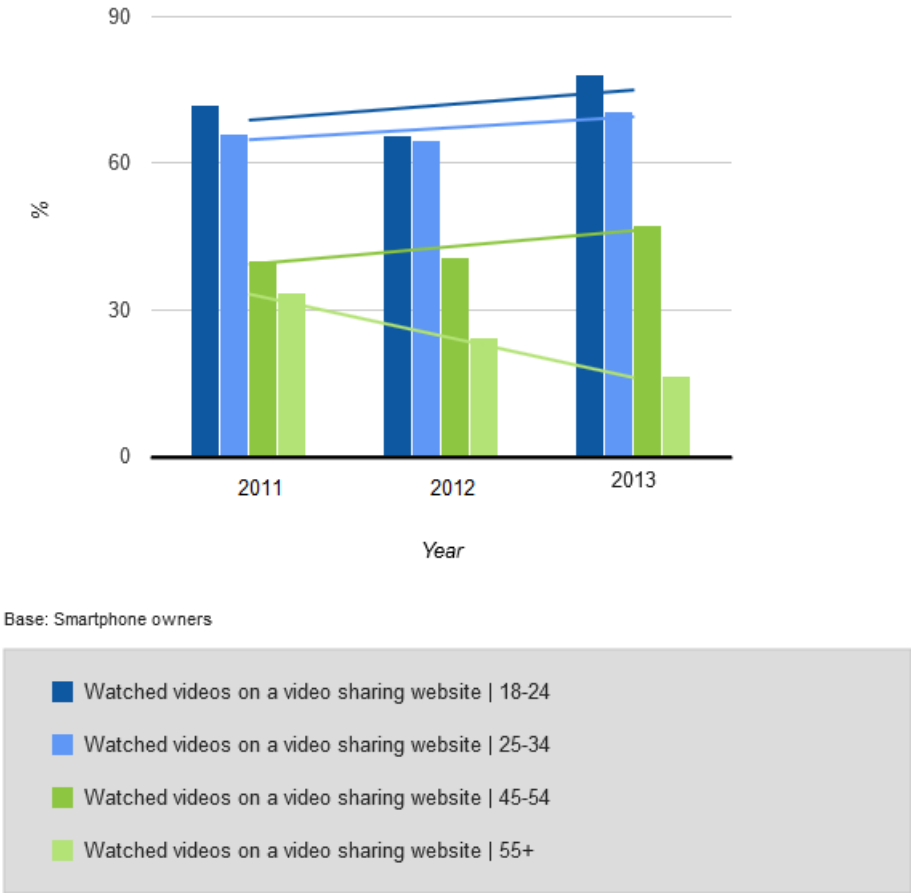

Figure 4-1 Video viewing trends year to year

Thus, an understanding of battery heavy behaviors will help identify power hungry components and power consumption can then be reduced by optimizing those components. Along with the results of this survey, Google also made the results of the "Consumer Barometer" [42] available to the public. The data from this was used to validate some of the conclusions made to derive profiles.

4.2. Workload profiles using demographic data

To build a workload profile, the most common smartphone use cases were considered. The most common uses other than making phone calls were considered in building these profiles.

As seen from Figure 4-2, each of the activities show an increasing trend from year to year. This implies that as the devices become more powerful and access to high speed internet increases, users tend to rely more on their mobile devices to perform various tasks. The quick and convenient nature of access to information and entertainment has led to increased adoption of more powerful mobile devices. With the increase in processing power and power draw, management of power becomes even more important.

The figure also shows that users have a wide range of activities for which they use their devices. Any form of power management will need to take in to account the varied nature of the activity set and try to understand the nuances of how different components of the device are stressed by the activities. Based on the activities shown, workloads representative of those activities were chosen to build a workload profile tailored to the different demographic groups. The overall data set was broken down based on age and gender into 6 different groups. Some of the activities were combined into a single workload to better represent the activity. For example, the activities for "browsed the internet" and "accessed a social network" were combined and incorporated in a single workload. The following sections detail how the various demographic groups are affected

by the type of activity and also describes the representative workload chosen.

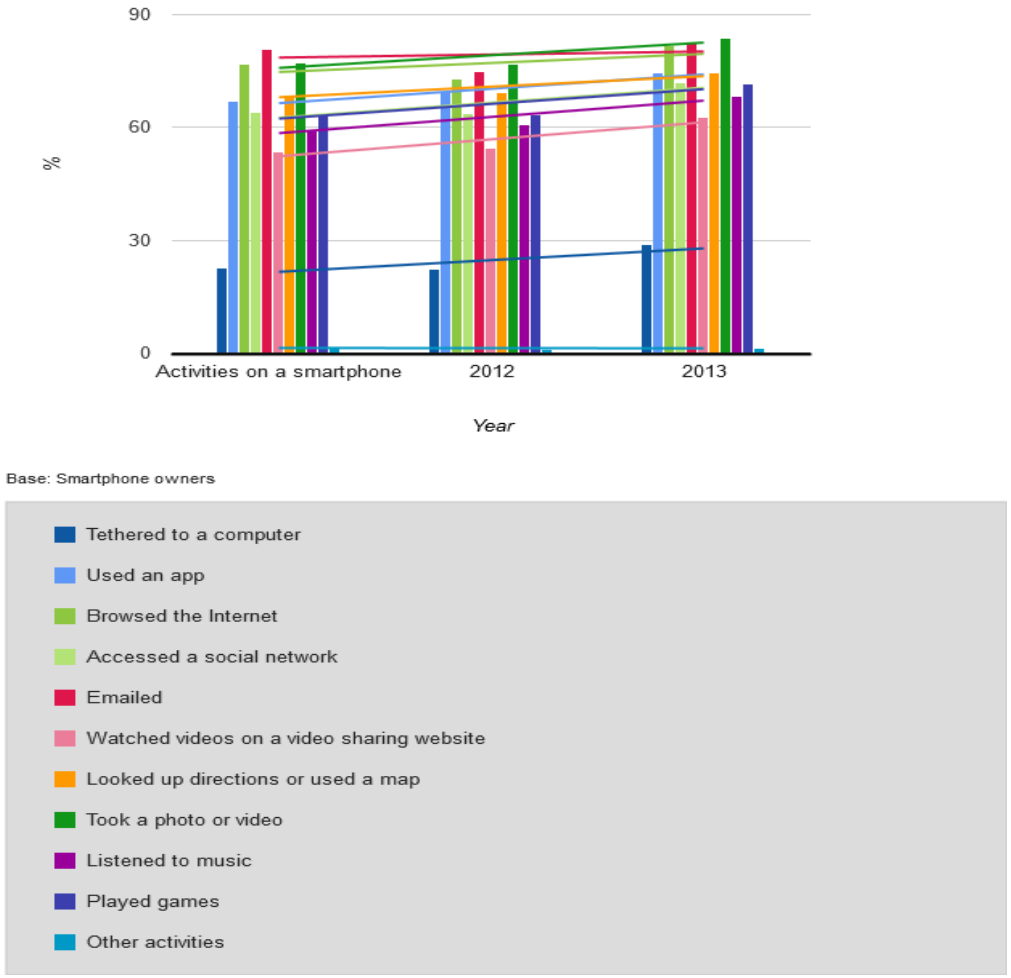

Figure 4-2 Key activities of Smartphone users

4.2.1 Video Playback

One of the most common activities on a mobile device is to watch videos. The availability of sufficiently large displays on mobile devices coupled with access to the internet allows users to view videos easily. Figure 4-3 shows the video viewership broken

down by age and gender. The graph shows that people between the ages of 18-34 have similar viewing habits. The numbers between men and women are also similar, differing by just a couple of percentage points with women watching more than men. In the next age group of 35-55 however, there is a marked difference in viewing habits from the previous age group. The trend of more women watching videos on their devices than men continues. The trend however is reversed in the 45-54 category with more men than women using their devices to watch videos. The 55+ age category also has men viewing videos on the mobile devices more than women, but when compared to the other age groups, both sexes use their devices less frequently to view videos.

From a power standpoint, any workload mix will need to account for these differences. The solution presented here is to play videos of length in proportion to the viewing habits as shown above. Another factor to be considered is the wide variation in the types of videos viewed. This is accounted for by using an open source video called Elephant's Dream made with open source tools. [43] The video adequately stresses all components of the video render process and is commonly used in the industry for video testing. While there exist multiple video playback apps, this effort utilizes the YouTube app, available on all Android devices. By using a streaming video app, both network and video decoding blocks can be stressed and measured for power.

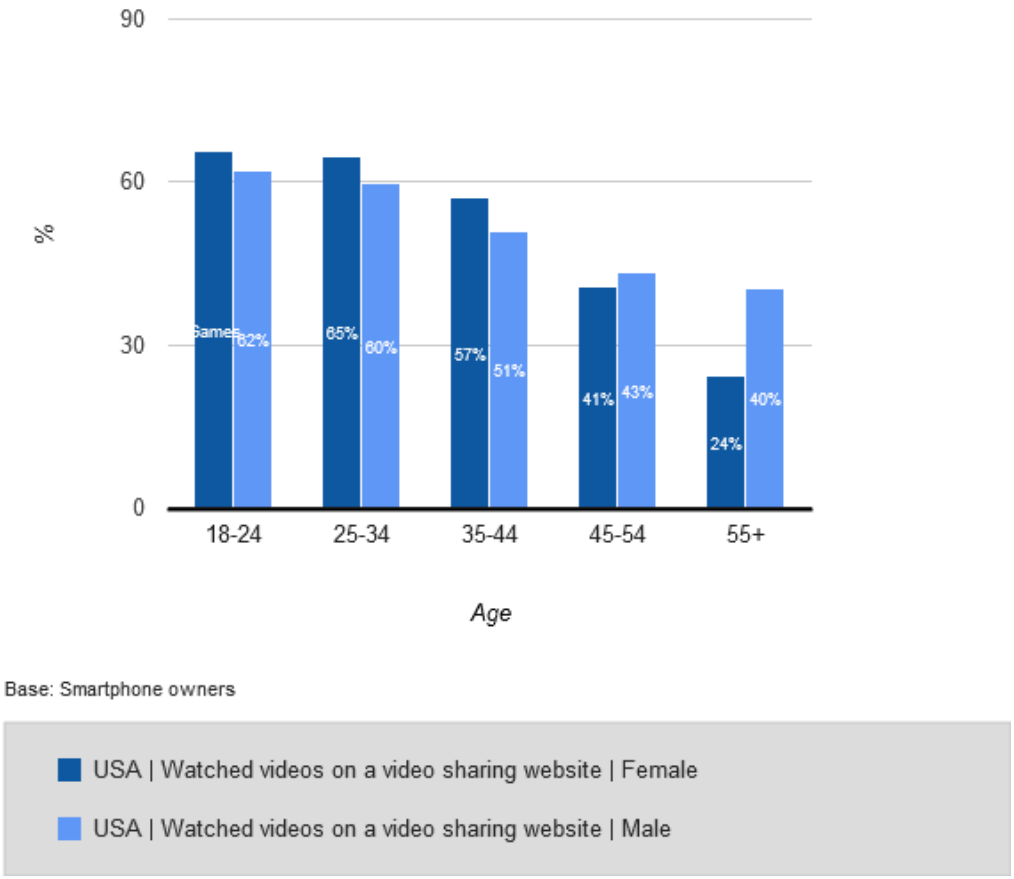

Figure 4-3 Video viewing statistics by age and gender

4.2.2 Audio/Music Playback

Another extremely common use case of a smartphone is listening to audio or music. The portability and ease of being able to carry large collection of music and listen to it on a device which can also perform other tasks is invaluable. Figure 4-4 shows the popularity of the activity of listening to music on a mobile device. The graph shows that younger people in the 18-34 age group quite frequently use their devices to listen to music. The older the person, the less likely they are to use their smartphones to listen to music. The drop off is greater in women aged 55 and over. Based on this trend, the

workload mix appropriately weights this activity for shorter durations than other age groups.

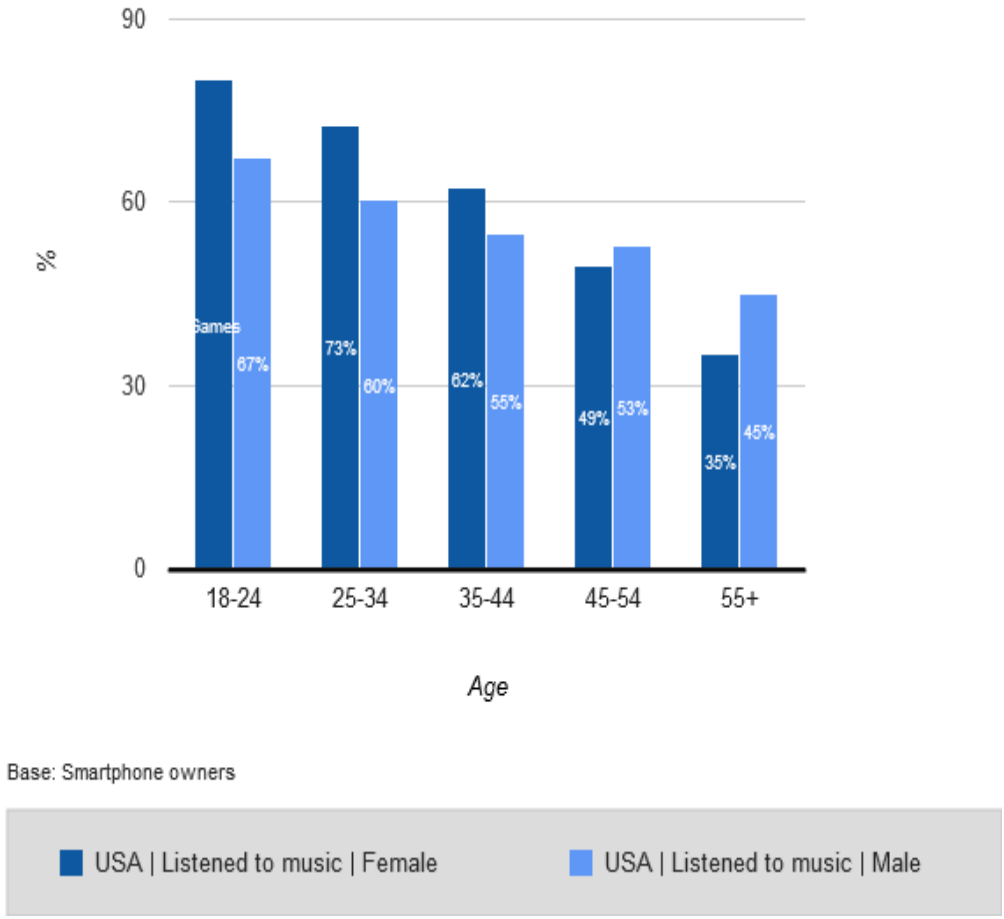

Figure 4-4 Music habits by age and gender

Audio playback is one of the most optimized components on a mobile device. Nonetheless, an analysis into the activity can provide potential opportunities for identifying any issues into power from an overall system view. The workload chosen to represent audio playback is a MP3 soundtrack and is stored on the device's memory, i.e.

the audio file is played back locally thus isolating the playback from any impact of streaming it over the network. The player used is the default Android music player.

4.2.3 Email

While the previous two use cases dealt with entertainment, email deals with communication, a vital feature in a smartphone. This feature has been included in all mobile devices, even in legacy devices which were designed primarily for voice calls. Figure 4-5 shows the breakdown of email usage across various demographic groups. The numbers stay roughly the same regardless of age or gender, indicating that this is a very important activity for all users. Another trend to note is that women tend to use email more than men. This has also been accounted for and is reflected in the final workload mix.

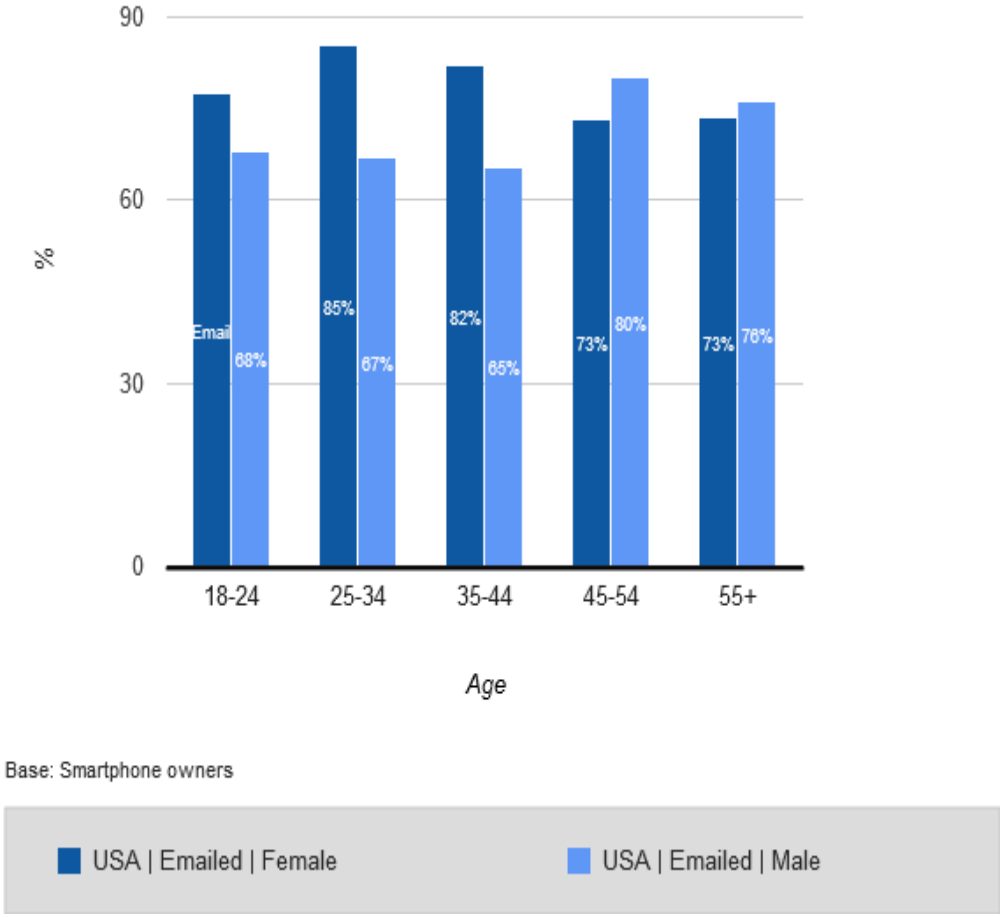

Figure 4-5 Demographic breakdown of email usage

A standard email app i.e. GMail was used as the workload of choice for this activity. The workload focused on scrolling through existing emails, composing an email and swiping through the app to simulate a standard use case.

4.2.4 Camera

Cameras are one of the more widely used features in a smartphone. The ease and flexibility in taking digital photographs using a mobile device implies this is a use case that needs to be considered for power studies as well. The graph in Figure 4-6 shows that men tend to use less of the camera than women. A wider variation in usage amongst men of different ages can also be observed. Conversely, women of all ages tend to use the camera approximately the same.

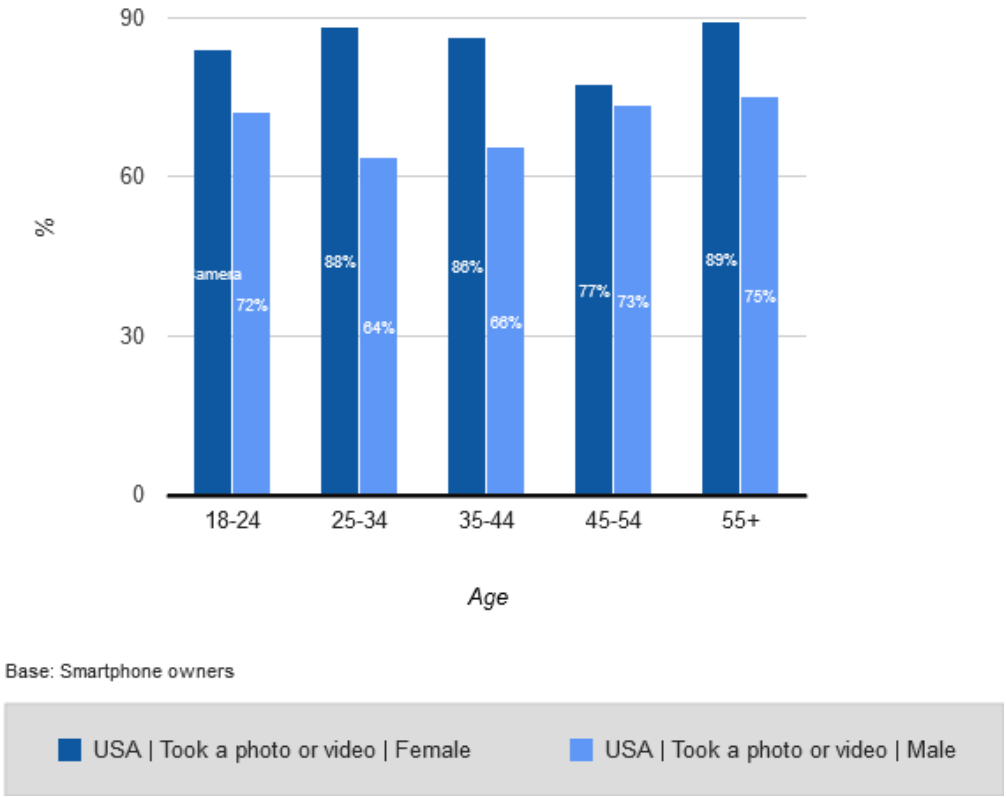

Figure 4-6 Camera use statistics by demographics

The default camera app is used to simulate the use case. The camera app was used to take a few photographs and then is switched into video recording mode. A short clip is then recorded before ending the use case.

4.2.5 Browsing the internet

The accessibility of internet on a mobile device has led to a new revolution. More people access the internet today on a mobile device than with a computer. [44] Probably the most popular activity on mobile devices, this is an activity that must be analyzed from a power perspective. Figure 4-7 shows that older men tend to browse more than women of the same age. Women of the younger age group also browse the internet for a longer time than men.

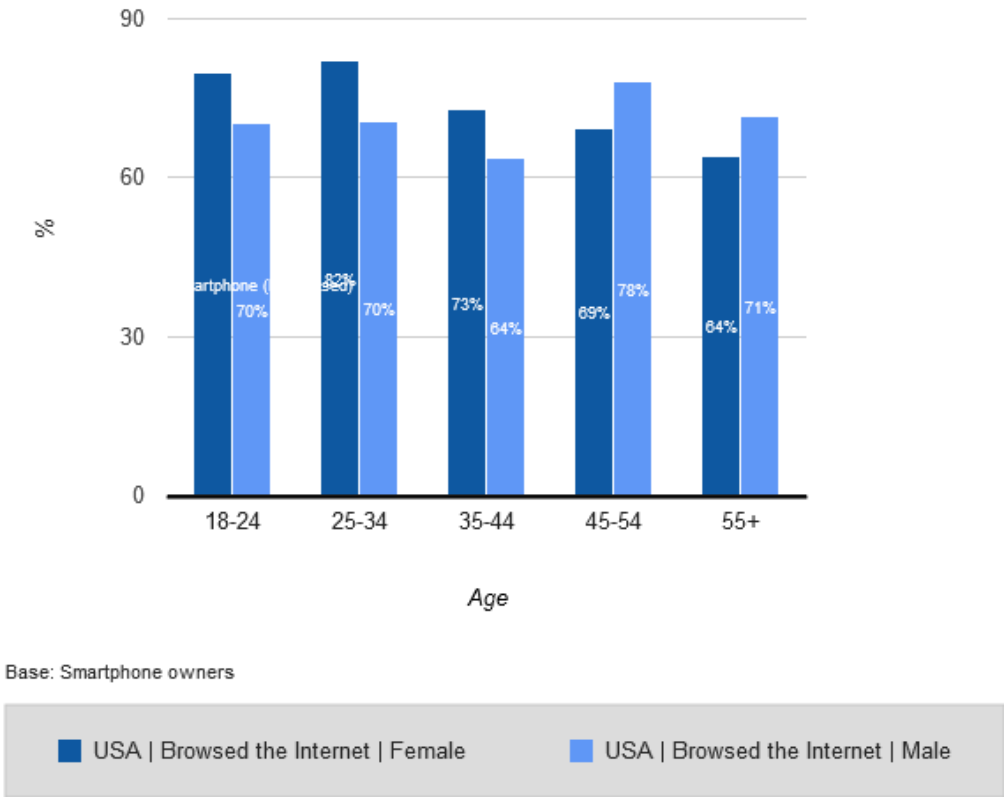

Figure 4-7 Browsing habits by demographics

Figure 4-8 shows the impact of social networks. Women tend to access social networks far more often than men across all age groups. The graph also shows that social networking is more popular among the younger crowd and by a significant margin.

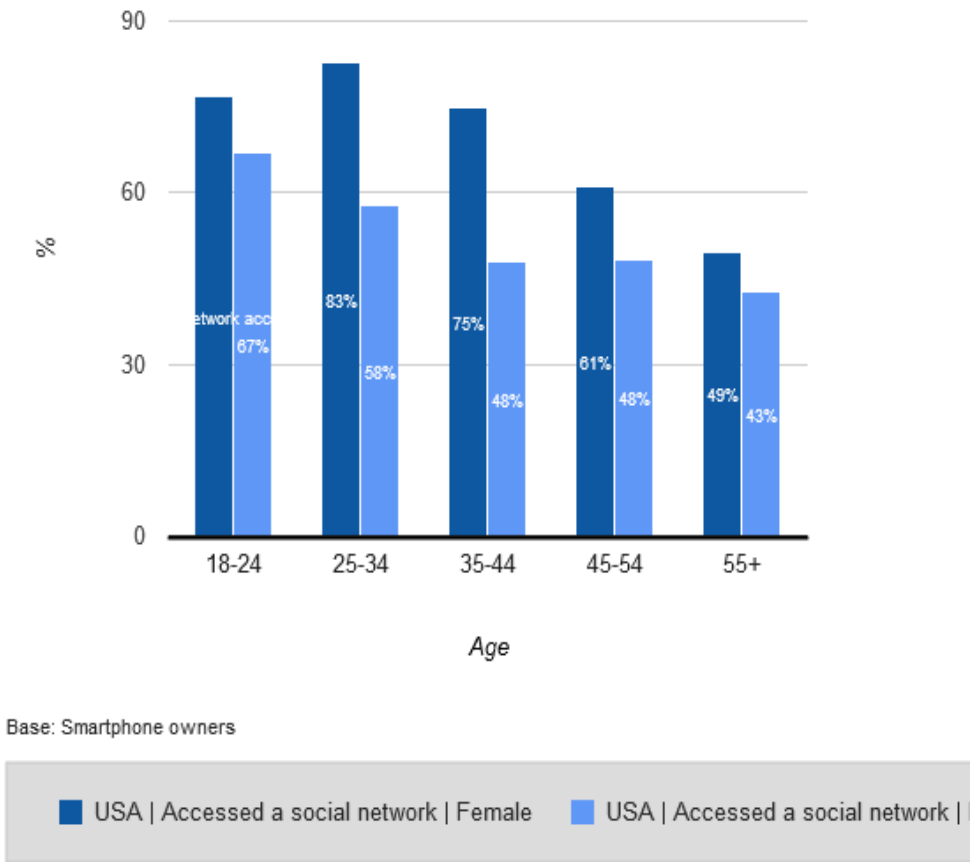

Figure 4-8 Breakdown of social networking habits by demography

Since the workload incorporates both browsing and social networking, it has been internally weighted to account for the differences. The browser used is the Chrome browser. The websites chosen for this effort include the more commonly used websites such as Amazon and the BBC. Accessing Facebook was used to simulate the social networking aspect of the workload.

4.2.6 Games

Games are another commonly used form of entertainment on mobile devices. The mobile gaming industry is rapidly gaining ground on other forms of games such as

PCs and consoles.[45] The wide variety of games available coupled with the fact that majority of games available on mobile devices are free all contribute to its popularity. Figure 4-8 shows how popular games are across the different demographic groups. The graph shows that games tend to be evenly popular among men regardless of age whereas a drop off in popularity is seen in older women. Another point to note is that women tend to play more games than men of the same age.

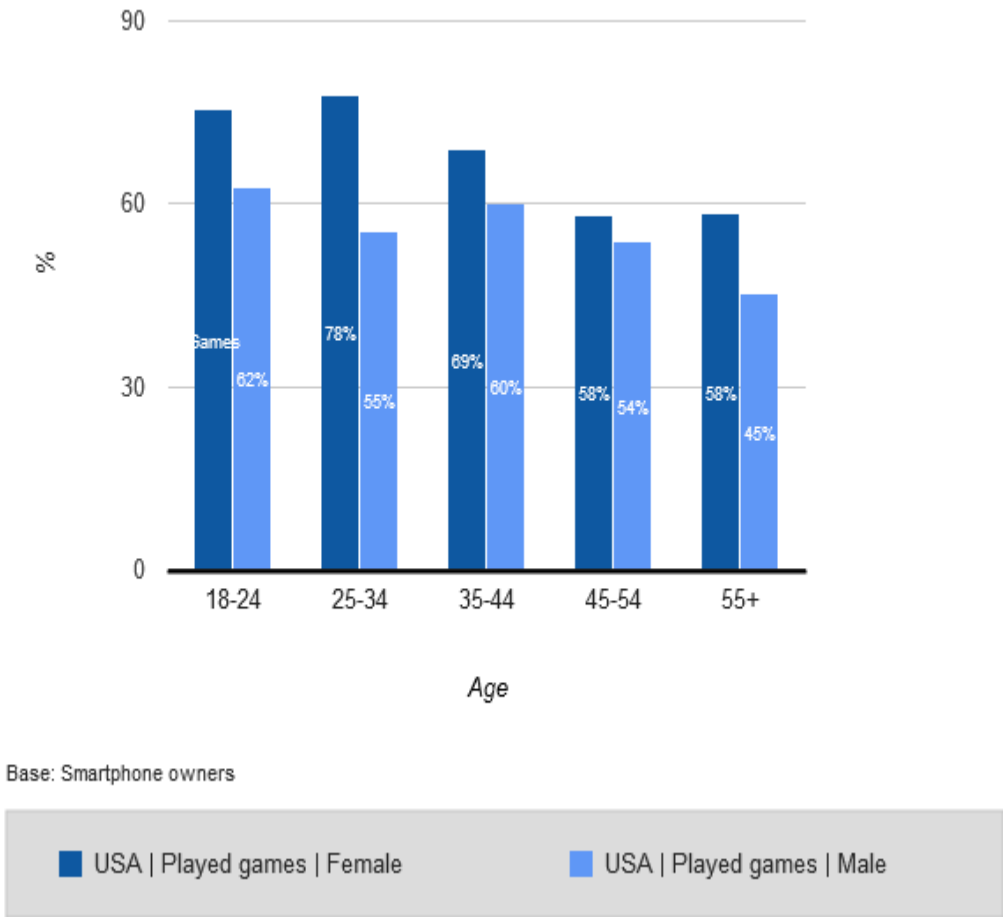

Figure 4-9 Games by age and gender

For the workloads, the workload included both casual and main stream games. Angry Birds [46] was chosen as a commonly used casual 2D game while AngryBots [47], a 3D game incorporating the Unity engine is used as the representative workload for a main stream game.

4.3. Power impacts of profiles

To study the effect of demographics, workload profiles representative of their age and gender were built. A workload profile is a combination of the different workloads, along with the duration each workload is run for. The profiles were constructed to reflect the use of the workloads by the respective demographic group. The profiles were constructed such that the time taken to run all workloads is 600s i.e. 10 minutes. 60s of the 10 minutes was used to transition between the various activities.

The profiles were created using the survey information presented in the previous section. The runtimes for each workload thus varies based on the demographic group. For example, it was observed that games were more popular among people of the age group 18-34, with women playing games more often than men. The workload profile for the average 18-34 man thus reflects this by spending more time on games than any other workload. Overall, younger people tend to use their phones more for gaming and video than older people who tend to spend more time browsing the internet or using email. Table 4-1 shows the breakdown of the time spent on a particular workload for each workload profile.

Table 4-1 Workload Profile breakdown (in seconds)

	Video	Email	Browser	Music	Camera	Game
18-34 Male	84	94	98	87	88	86
18-34 Female	75	99	94	84	100	85
35-54 Male	72	104	103	81	96	82
35-54 Female	62	109	101	74	106	85
55+ Male	55	124	113	60	111	73
55+ Female	47	125	101	57	125	83

4.3.1 Experiment Setup

The 6 workload profiles constructed above were then run while simultaneously measuring the power consumed by the mobile device. For this experiment, an Intel x86 based Android device running Android 4.0.4 was used. The device was instrumented by connecting the battery terminals to a Monsoon Solutions power monitor [71] and measuring the current. By measuring the current consumed, the amount of energy consumed by the device and therefore the system level power can be measured.

To capture the default behavior of the device, the system's frequency governor was set to "ondemand". The default behavior of the ondemand frequency governor is to change the CPU frequency as and when a demand for higher frequency exists. It rapidly lowers the frequency when there is no load on the CPU. The workload profile was then run at different system frequencies to understand the effects of frequency scaling on power consumption as well the effects on user experience. The device used for this experiment supported CPU frequencies of 600MHz, 1200MHz and a max frequency of

1600MHz. To force the device to run under the above mentioned frequencies, the governor was set to "performance" mode.

4.3.2 Results

4.3.2.1 Frequency Scaling - Energy

The effect of frequency scaling on the different workload profiles was observed by measuring the energy consumed by each workload profile. Figures 4-10 through 4-15 show the actual energy consumed by the device for the different workload profiles when run at 600MHz, 1200MHz, 1600MHz and the default "ondemand" frequency. Figure 4-16 compares the energy consumed values for all profiles at all run frequencies.

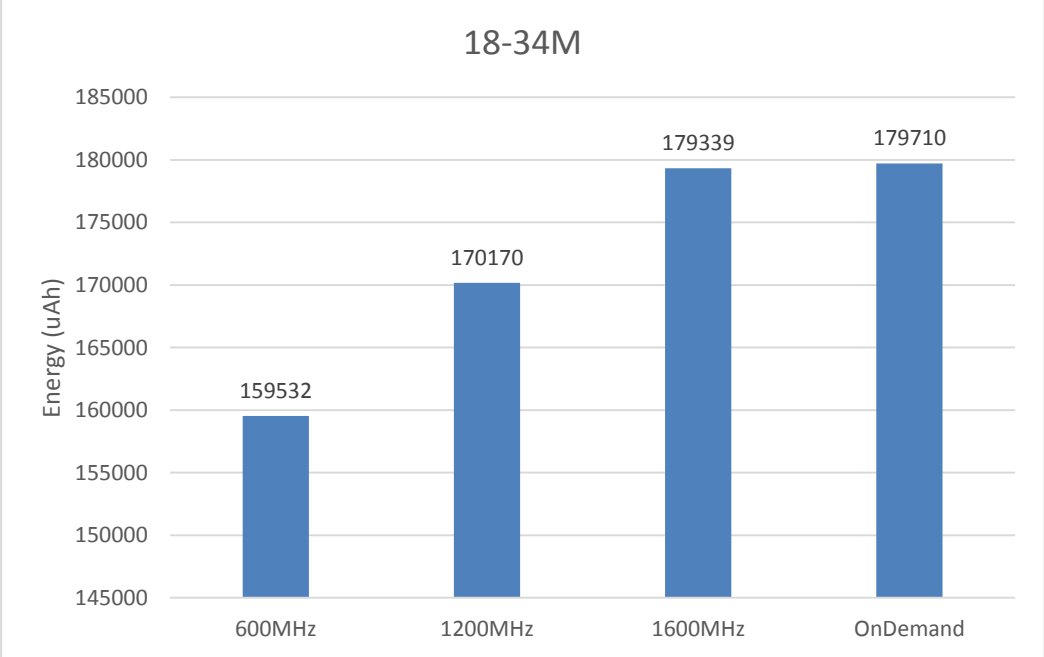

Figure 4-10 Energy consumed for 18-34 male profile

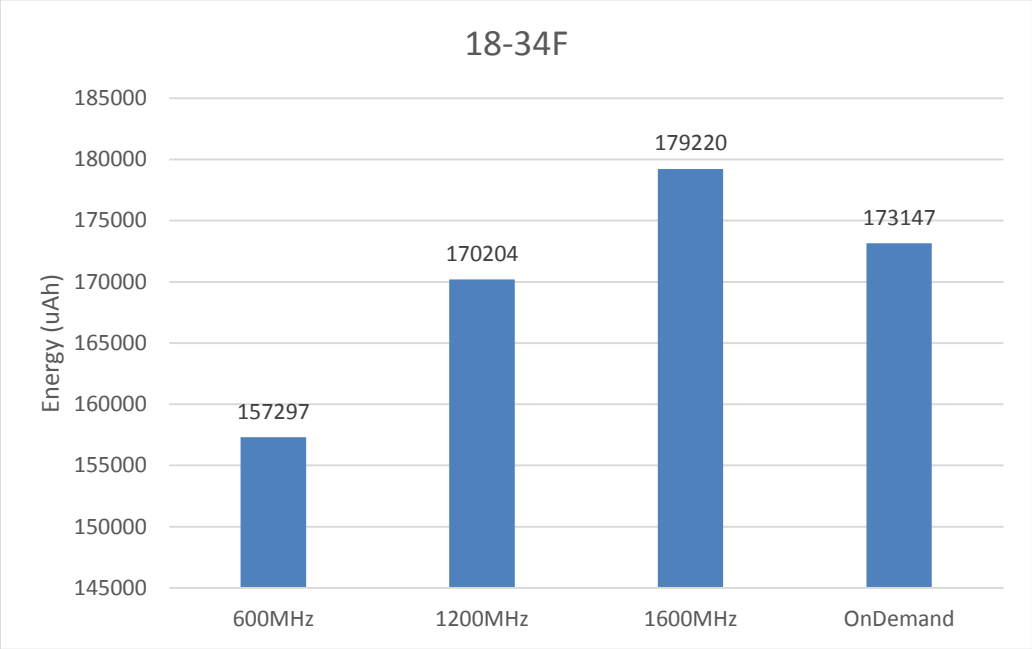

Figure 4-11 Energy consumed for 18-34 female profile

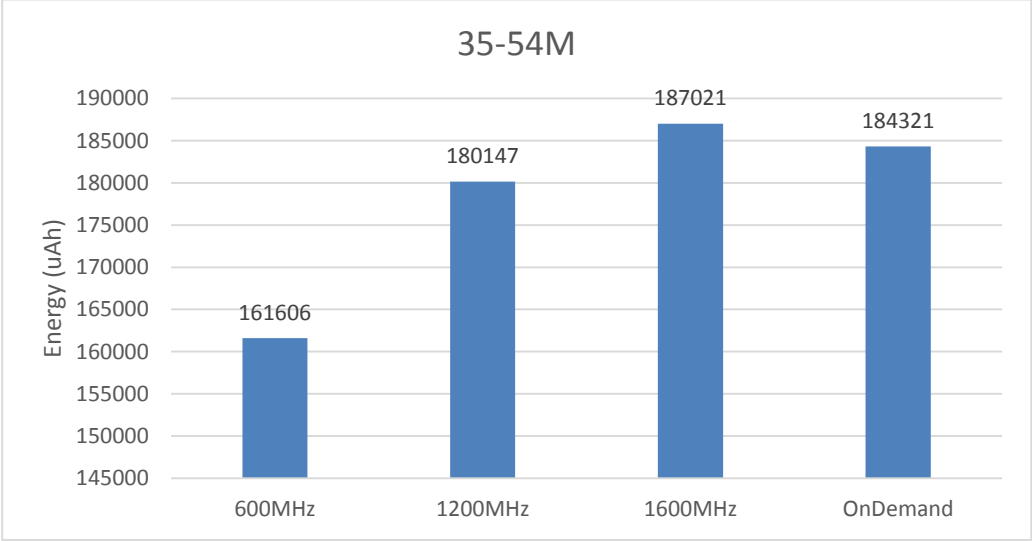

Figure 4-12 Energy consumed for 35-54 male profile

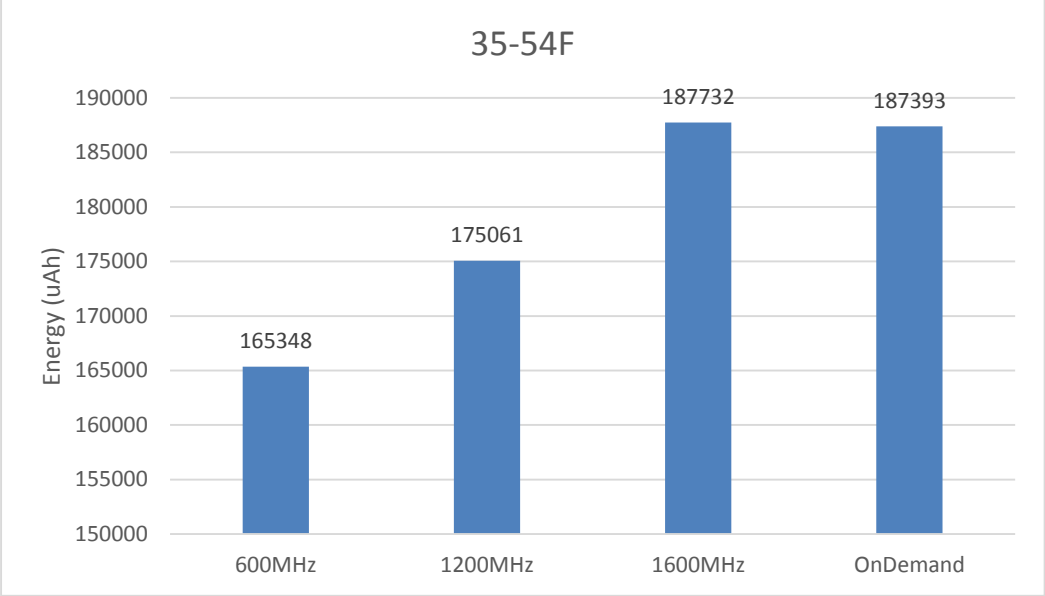

Figure 4-13 Energy consumed for 35-54 female profile

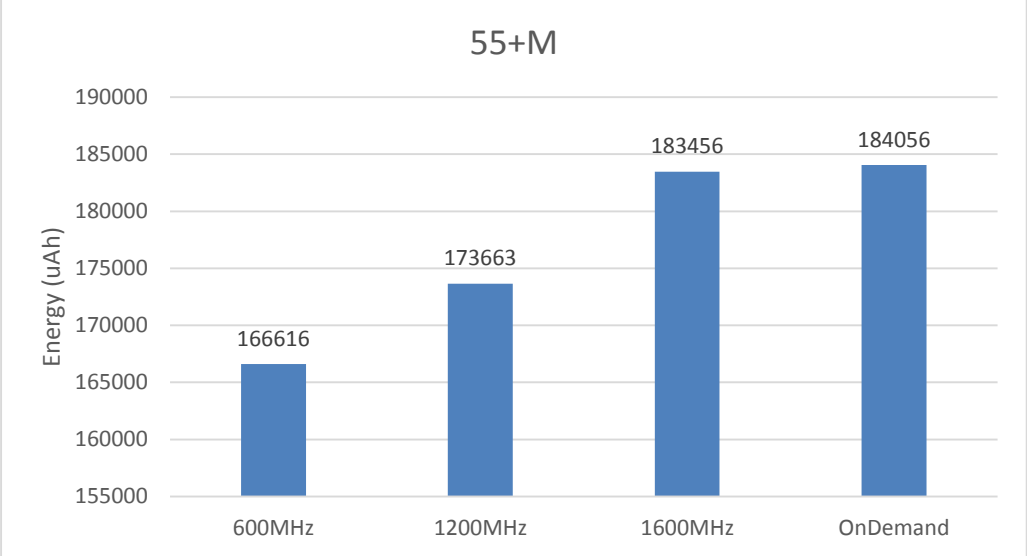

Figure 4-14 Energy consumed for 55+ male profile

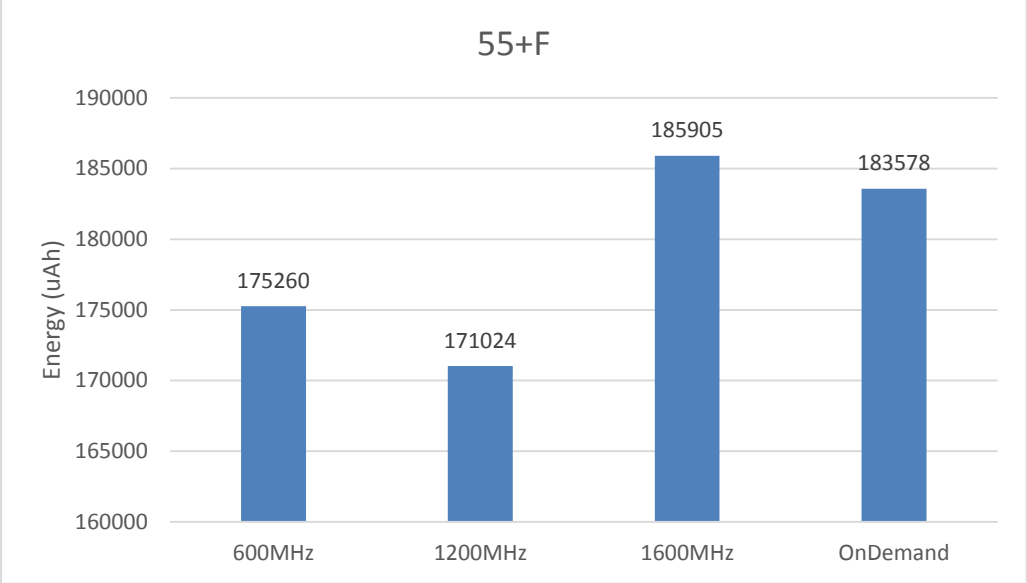

Figure 4-15 Energy consumed for 55+ female profile

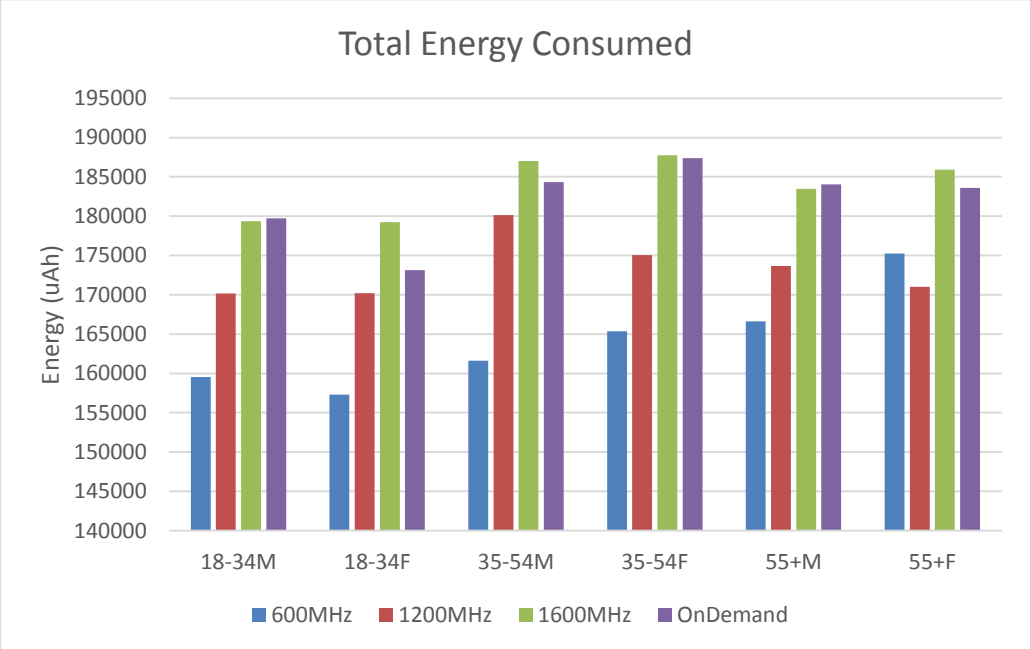

Figure 4-16 Comparison of energy consumed for all profiles and frequencies

It is seen all profiles when run at 600MHz consume lesser energy than other frequency runs except for the 55+ female category. The profiles run at 1200MHz also show that the energy consumed is lesser than the default ondemand frequencies. When run at 1600MHz, the energy consumed is similar to the ondemand energy values. 18-34M and 55+M run at 1600MHz have lesser energy consumed than the OnDemand governor suggesting that, running at a higher frequency is more suitable than the OnDemand for these 2 workload profiles.

4.3.2.2 Frequency Scaling - Perceived Performance

Another method of determining the effect of frequency scaling is to measure the actual runtimes of the workloads and also observe the reactions of the user. The default design of the workload mix is to complete the execution in 600 seconds. This also requires the device to be run under default settings. By running the workloads at set frequencies, it was observed that runtimes varied from the designed 600s. Figure 4-17 shows the runtimes of the workload mix at the different frequencies. Running the workload at 600MHz uniformly affects all workloads negatively by atleast 12%. More importantly, all workloads exhibited increased app loading times, sluggish responsiveness and increased page load times, all of which are noticeable to the user.

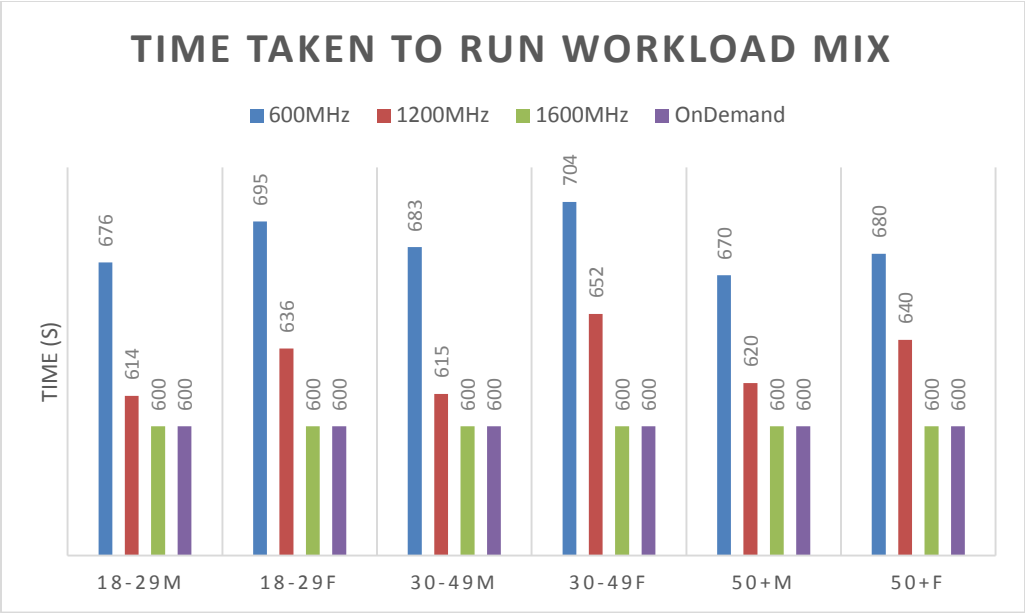

Figure 4-17 Comparison of runtimes for all workload profiles

All workloads run at 1200MHz also exceed the 600s runtime. However the impact is muted compared to the 600MHz scenario. All workload profiles take longer than the designed 600s, but even the slowest workload profile takes only 7% longer to run. The responsiveness of the workload profiles when run at 1200MHz are better than the 600MHz, but are not as smooth as the default frequency setting.

The 1600MHz scenario provides the best possible performance and matches the run time design of the workload profile, with no impact on any perceived performance.

Four of the profiles running at 1200MHz have their runtimes within 5% of the designed total 600s. If this is acceptable in terms of user experience, it would be advantageous to run the workloads at a 1200MHz than use the default ondemand frequency setting. Setting an appropriate performance bound can allow for improved power saving on the mobile device.

4.3.2.3 Hyper Threading (HT)

An experiment was carried out was to observe the effect of hyper threading on the workload profiles. For this experiment, the different workload profiles were executed with the hyper threading mechanism of the CPU disabled and the results were compared to the default behavior. Figure 4-18 compares the energy consumed by the device for the different profiles with HT enabled and disabled.

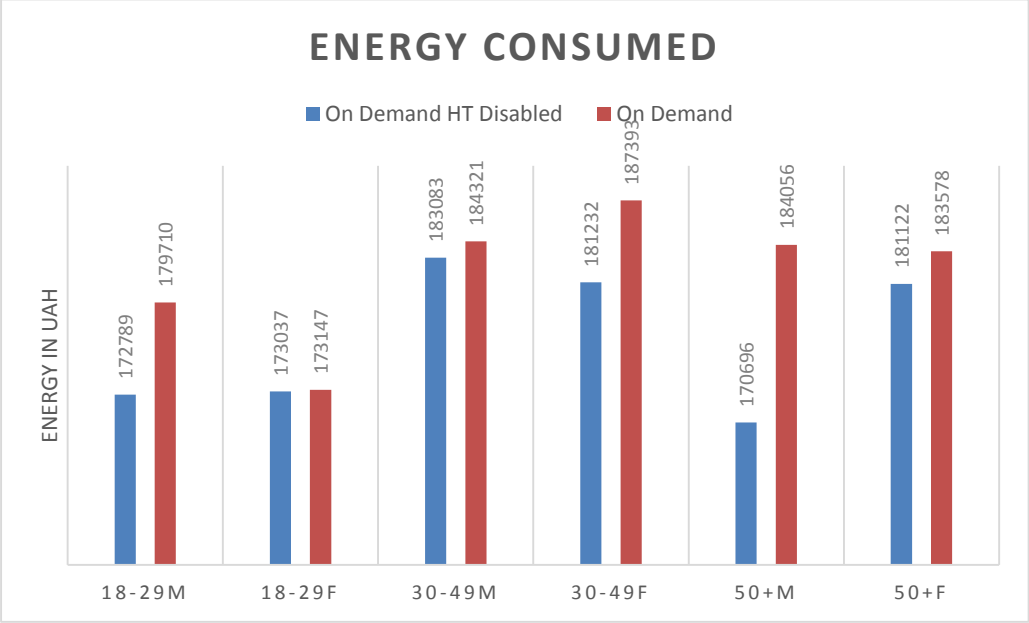

Figure 4-18 Comparison of workload profiles with HT enabled and disabled

While 2 groups show almost no difference with HT disabled, the most significant difference is seen with the 50+ male group where disabling HT consumes approximately 7% less energy. This is caused by some workloads not taking advantage of the HT feature, resulting in idle units consuming power and thus causing a larger energy draw with HT enabled.

4.3.3 Power analysis of workloads

The previous sections focused on exploring how different sections of the population used their mobile device and how the varied usage affects power consumption on the device. It was shown that workload profiles did have a varied effect on power, therefore accounting for how the device is utilized by the user is equally important as optimizing components on the device. The previous scenario however explored only the custom workload profiles created. The following section explores the effects of some of the workloads used to create the workload profiles.

In order to optimize power consumption on a mobile device, we must understand how commonly used applications affect the system from a power point of view. To this end, choosing workloads that pressure the system the most allows us to get information about the system under those conditions. The workloads chosen must also be representative of commonly used applications. Based on the knowledge of what users do on their devices, the following apps/workloads were chosen:

1. Audio
2. GLBenchmark 2.7
3. YouTube app
4. Browsing

The chosen workloads were executed and accurate system power measurements for different CPU frequencies were collected. The workloads were also run on multiple Android devices to observe how power consumption varies across different Android devices.

The experiments for frequency scaling were run on an Intel x86 device running Android 4.2, and the power measurements were made using a NI DAQ system. To understand if the hardware configuration of the system contributes significantly to power

consumption, the workloads were also run on a Samsung S4 and a LG Nexus 4. Measurements on these 2 devices were carried out by instrumenting the battery terminals using a Monsoon Solutions Power Monitor.

4.3.3.1 Audio

Smartphones are frequently used to listen to music. The workload chosen to represent this use case was the same as the one used in the workload profile experiment, i.e. a local MP3 file being played using the default onboard media player. The variation in power consumed by the device as frequency is varied is shown in Figure 4-19. The overall power drawn stays relatively the same regardless of the set CPU frequency because audio processing is primarily done by specialized onboard DSP processors.

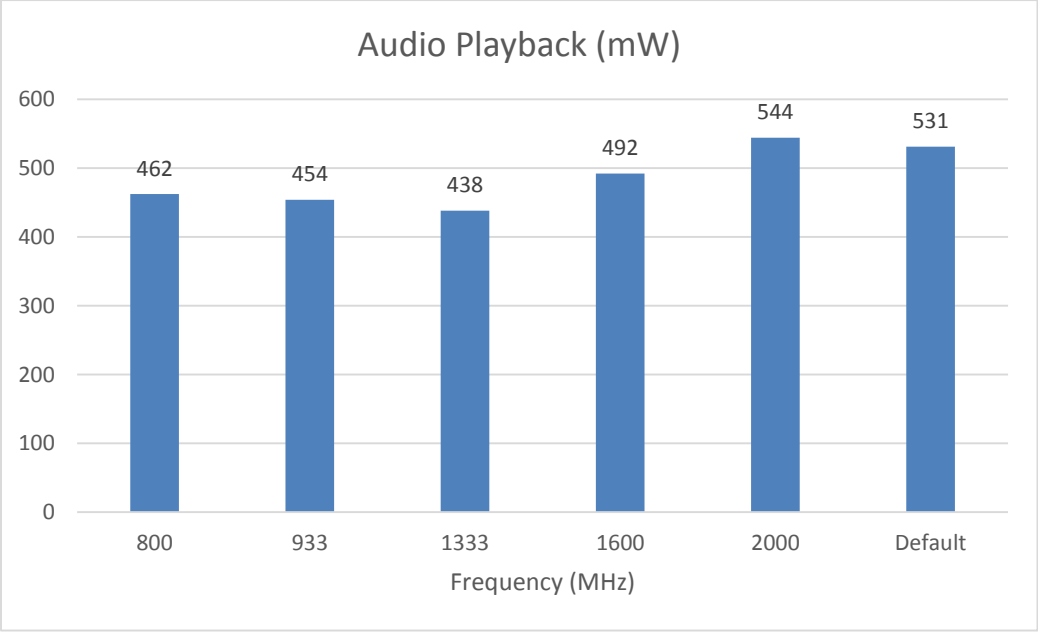

Figure 4-19 Power consumption for audio workload

Figure 4-20 shows the power consumed by the workload across the different test devices. A wider variation in power consumption is observed here, but this can be attributed to the different size of the displays as the primary power consuming component in this workload is the display.

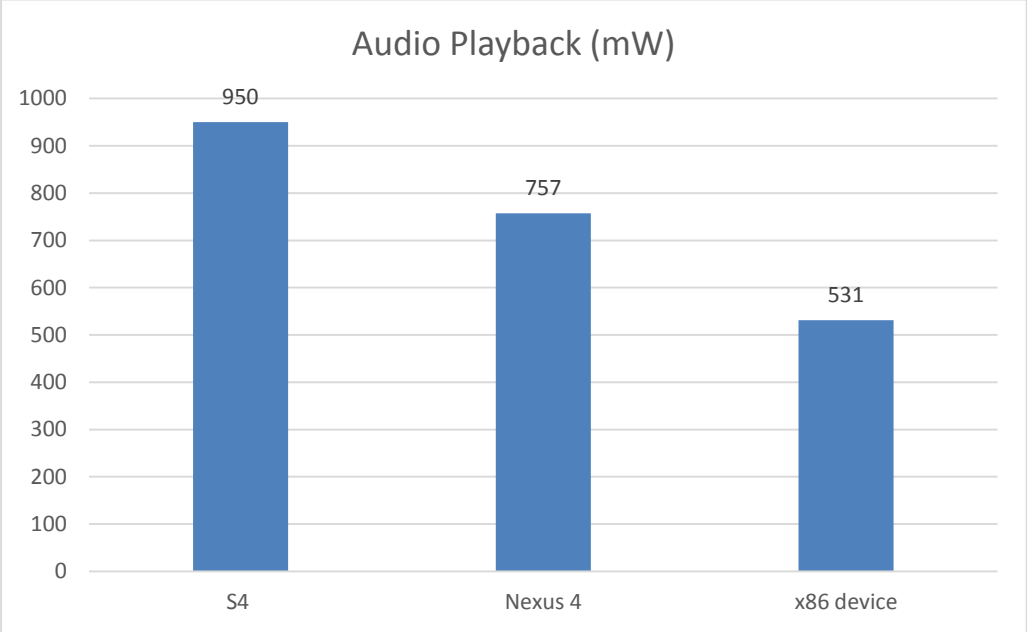

Figure 4-20 Audio workload power comparison across devices

4.3.3.2 GLBenchmark 2.7

Another common use case of smartphones is to play games on them. GLBenchmark[48] is a free, commercially available tool, which is used to benchmark mobile devices. The work done in the benchmark process is similar to the type of work the CPU and GPU would do in a regular 3D game. The sub-workload Egypt was chosen for this experiment. Figure 4-21 shows the effect of frequency scaling on the power consumed by the device when running this workload. Being a graphics and computation

heavy workload, frequency scaling has a significant effect on the workload. From a performance perspective, lowering the frequency led to degradation in performance.

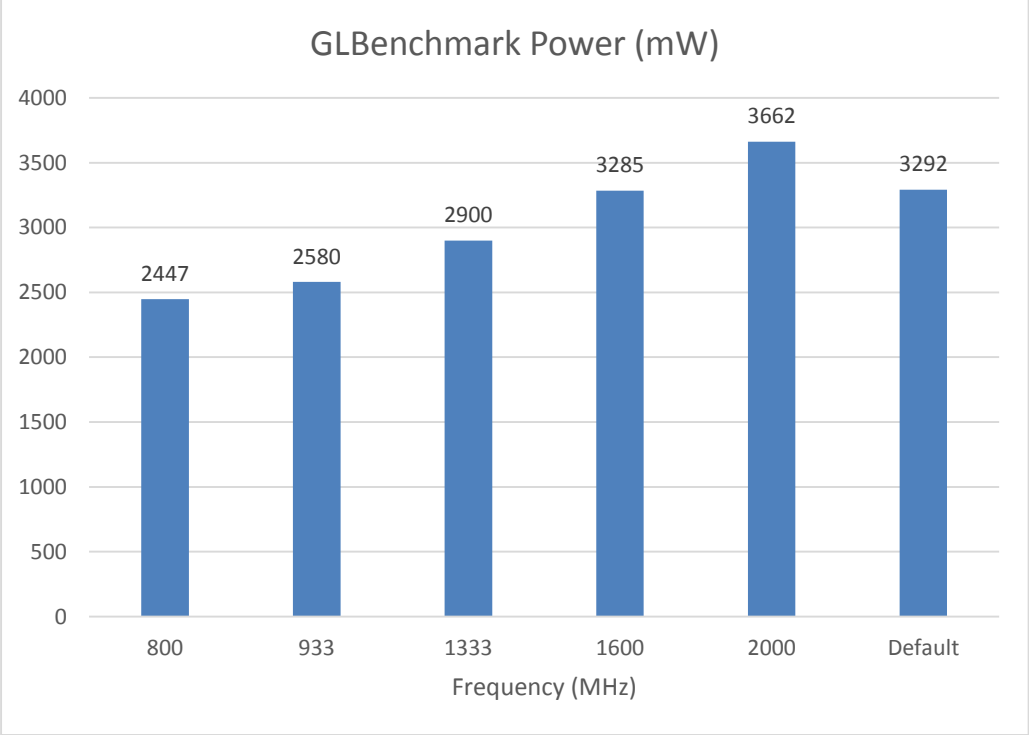

Figure 4-21 Power consumption for GLBenchmark

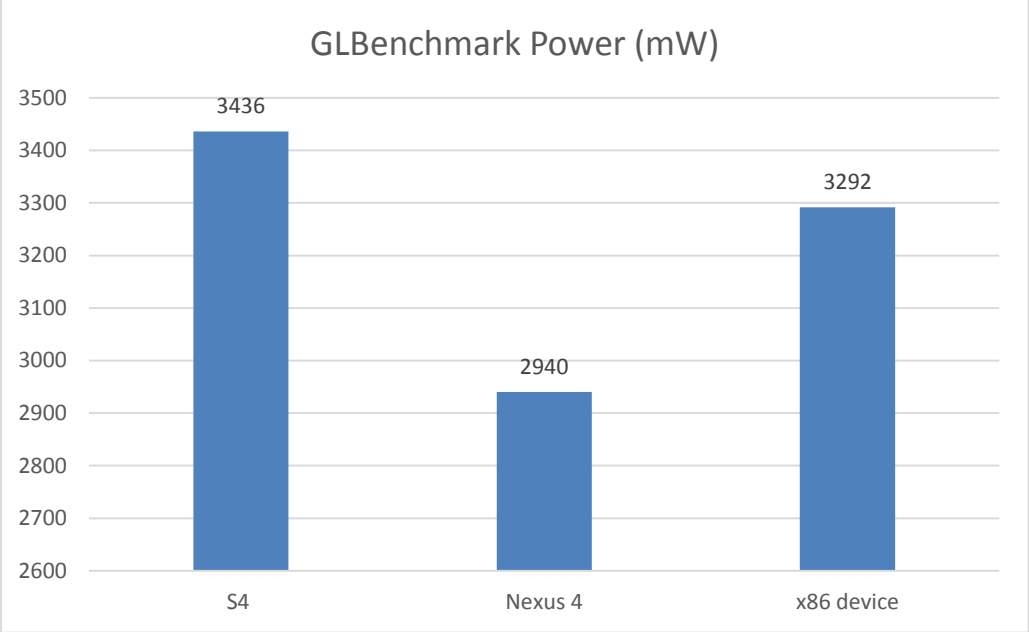

Figure 4-22 GLBenchmark power comparison across devices

Figure 4-22 shows how the workload behaved across the devices. The power consumption numbers are similar across devices. The size of the screen does not affect this workload significantly as the workload is computationally heavy.

4.3.3.3 YouTube app

Video playback is another common use case in smartphones. YouTube is a commonly used source for playing videos on mobile devices, therefore this was chosen as a workload targeting the video playback use case.

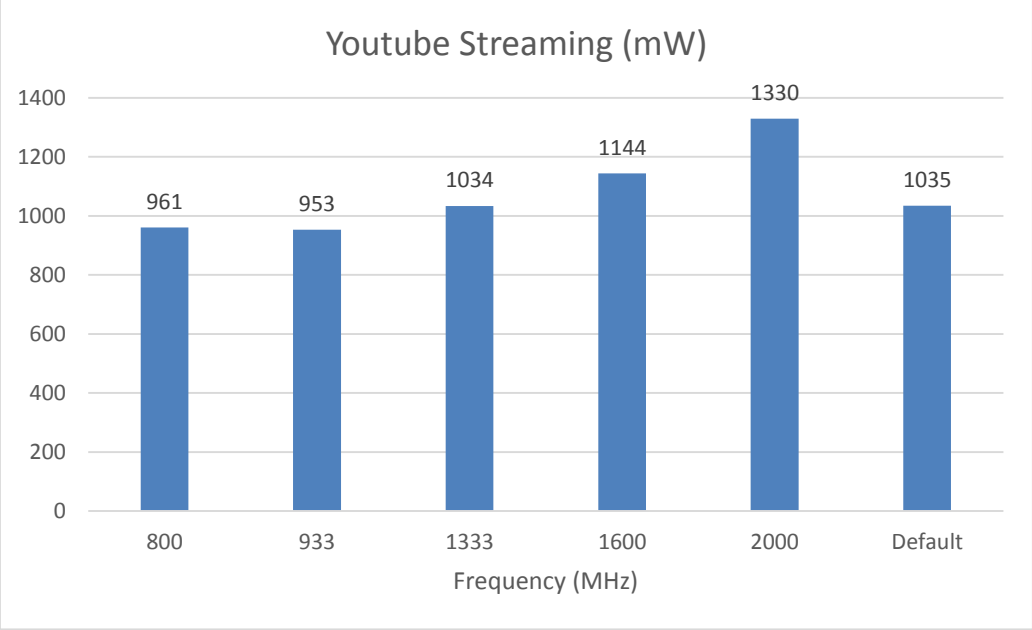

Figure 4-23 Power consumption for YouTube streaming

Figure 4-23 shows that while the highest power consumption was observed at the highest frequency, all other frequencies had power values similar to the default frequency governor. This indicates that the workload is less affected by variations in CPU frequency.

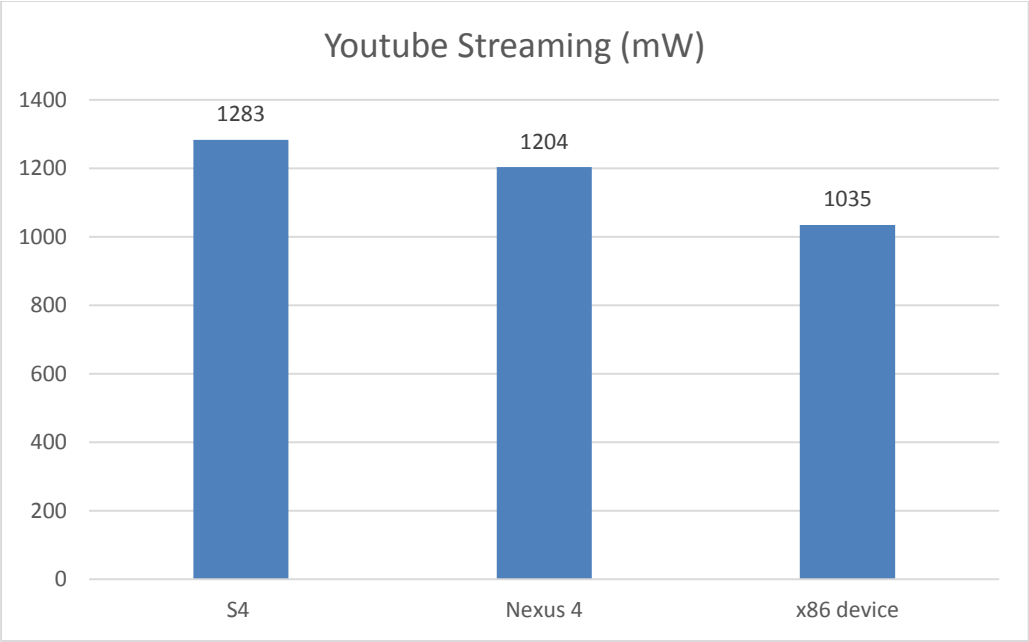

Figure 4-24 YouTube streaming power comparison across devices

Figure 4-24 shows the comparison of the workload run on different devices. The power consumed across the devices for the YouTube app is similar, with a slightly smaller power draw in the x86 device due to a smaller screen.

4.3.3.4 Browsing

The most popular activity on a mobile device is to browse the internet. This is an extremely common use case and therefore was chosen as a workload for this experiment. The scenario involves simulation of browsing the internet by visiting websites of a diverse nature.

Figure 4-25 shows the power consumption numbers for different frequencies. It was observed that ramping the frequency resulted in increasing power consumption, with increase in performance as well.

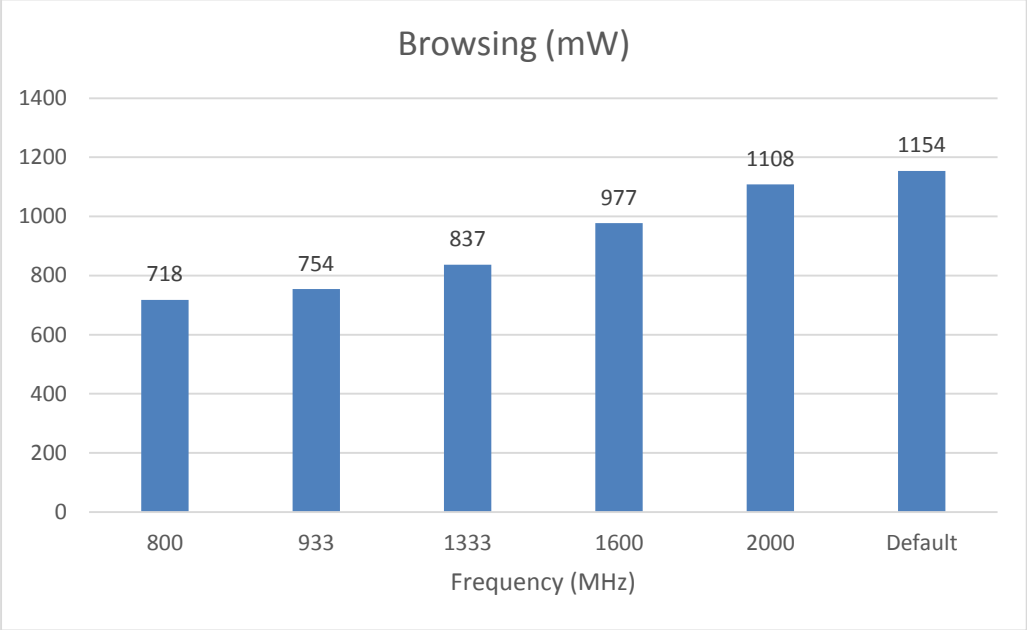

Figure 4-25 Power comparison for browsing

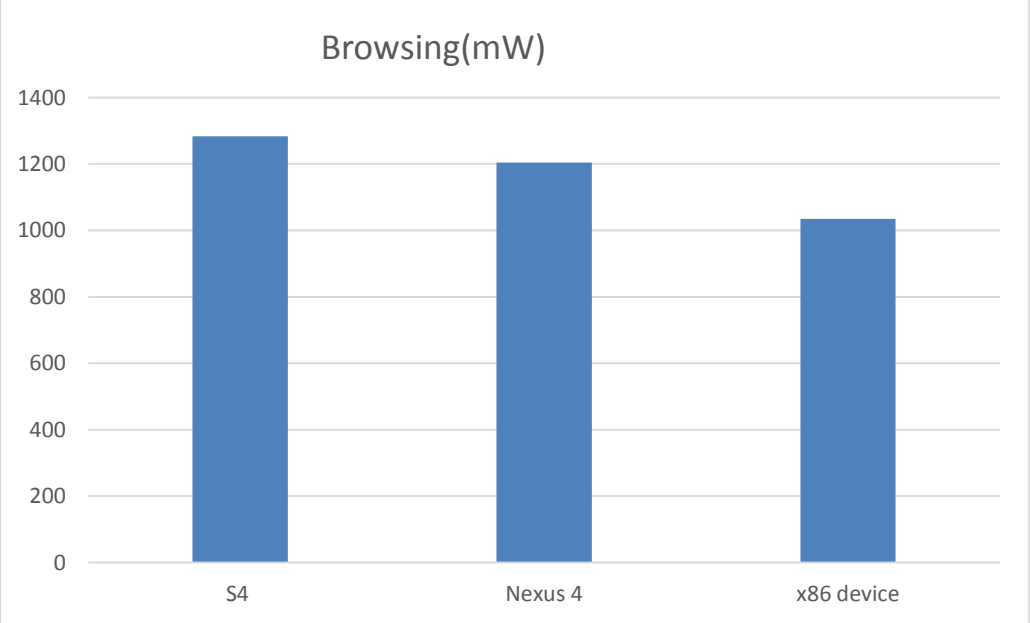

Figure 4-26 Browsing power across devices

The power consumption numbers of executing this workload on different devices is shown in Figure 4-26. The Samsung S4 device draws more power than the other 2 devices. This can again can be attributed to the larger screen size compared to the other 2 devices.

Chapter 5

Machine Learning Algorithms

The study and development of algorithms that can enable computers to learn and act based on data is termed machine learning. Identification of complex patterns in data can help understand why these occur and suitable actions can then be taken to build models from the data based on these understandings. Machine learning can be classified into three broad categories based on the nature of the feedback available to the learning mechanism [81]. They are:

1. Supervised Learning - a situation in which both inputs and outputs of component can be perceived [49]

2. Unsupervised learning - learning when there is no hint about correct outputs [49]

3. Reinforcement learning - learning with feedback about the process but none about the correctness of the process. [49]

For this project, the applications are being classified on the basis of usage, supervised learning algorithms are best suited for this problem. The following sections discuss the aspects of some machine learning algorithms that can be applicable to this effort. Finally a discussion of WEKA, a tool used for machine learning is presented.

5.1 Naive Bayes

Naive Bayesian methods are a set of supervised learning algorithms which work by applying Bayes' theorem. It makes a "naive" assumption that the input features under consideration are independent of each other. The basic principle is to determine the probability that a given input feature set maps to a specific output class. This is computed

based on knowledge of the prior probability of the mapping and how well it matches with the observed evidence. The data with which this classifier is trained, also known as the training set, is the evidence previously seen. The classifier learns by finding a reliable method to map the set of input features to the output measure in the training data set.

Mathematically, given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{5.1}$$

Assuming the independence between features, we have:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y), \tag{5.2}$$

This is used to reduce equation (5.1) to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{5.3}$$

Now since the denominator is the marginal probability that an observation x is seen for all possible scenarios, it is not relevant for the decision process as it is same for all class assignments [51]. Using that, the posterior probability of the class can be computed as:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \tag{5.4}$$

Naive Bayes learners and classifiers are extremely fast when compared to other classification algorithms. They work well when the independence assumption is appropriate and are highly scalable. Based on the probability distribution of the features under consideration, different types of Naive Bayes Classifier can be designed.

5.2 Decision trees

Decision tree learning utilizes a decision tree as a predictive model, mapping observations about an item to conclusions about the item's target value [54]. It is a non-parametric supervised learning method, whose goal is to create a model that can predict the value of a target variable by learning decision rules from training data sets. Target values taking finite discrete values are called classification trees while target values which are continuous are called regression trees.

The trees built are relatively simple to understand and interpret, but can create over-complex trees by over fitting the data. For classification, an assumption that all features have finite discrete domains and that there is only one target class. The cost of using the tree is logarithmic in the number of data points used to train the tree. However, the problem of learning an optimal decision tree is an NP complete problem.

There are multiple decision tree algorithms such as the ID3, C4.5 and C5.0

The ID3 algorithm creates a multiway tree, determining the minimum entropy for each node. The training data set is used to build a decision tree which is stored in memory. At runtime, this is used to classify new test data, by working down the decision tree using the values of the new test data.

The C4.5 tree is an extension of the ID3 algorithm. The decision tree is built from the training data in the same manner as the ID3 algorithm. It then converts the trained trees into sets of if-then rules. The accuracy of the rules are then evaluated to determine the order in which the rules must be applied to new data. [56]

The C5.0 algorithm is an improved version of the C4.5 algorithm. It runs more efficiently by utilizing lesser memory and running faster. The decision trees built can be modified by winnowing attributes or by weighting them as necessary.

5.3 Support Vector Machines

Support Vector Machines are supervised learning methods used for classification. The SVM takes a set of training examples marked as one of two categories and builds a model that assigns new data into one category or the other. It is a form of a linear classifier. The training set is used to construct a hyper-plane (aka decision surface) such that the separation between the plane and the nearest training points of any class is maximal.

The speed of classification is high. The model built while linear is also discriminative as an attempt is made to maximize the quality of the output from a given training data set.

Mathematically, given some training data $\mathcal{D}$, a set of n points of the form

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1, 1\}\}_{i=1}^{n} \tag{5.5}$$

the hyper plane can be written as the set of points x satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0, \tag{5.6}$$

where $\cdot$ denotes dot product and w the normal vector to the hyperplane. Minimizing the $||w||$ such that there are no data points between the margin hyper planes described by equations 5.7 and 5.8 allows for maximal separation between the two classes

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{5.7}$$

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{5.8}$$

.

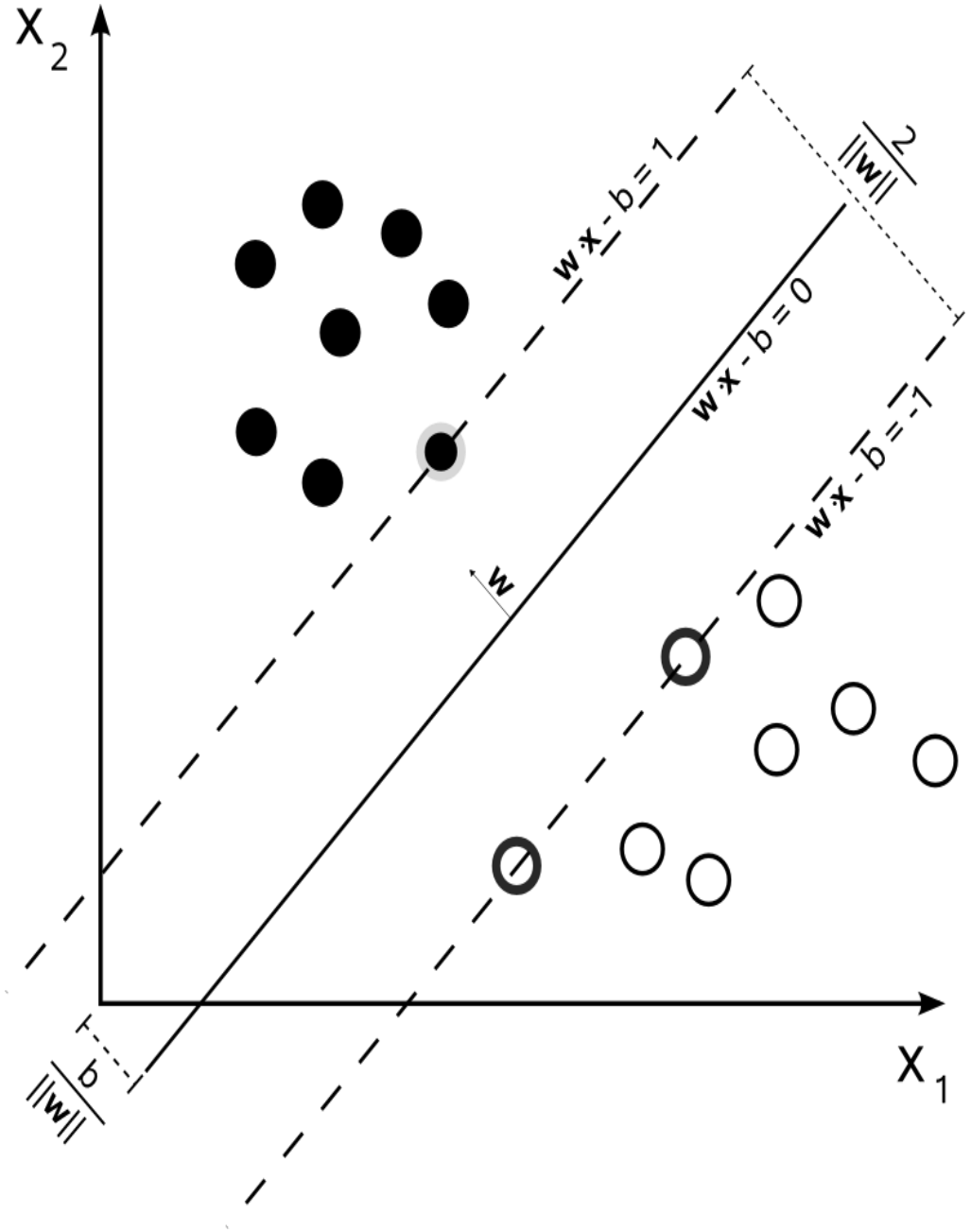

Figure 5-1 Hyperplane dividing a set of points

Support vectors are the data points closest to the decision surface, which are hardest to classify and have a direct bearing on the optimal location of the hyper plane. Thus the support vectors of the linear classifier is the set of training points that would change the position of the hyper plane if removed from the data set. [57]

SVMs are directly applicable to 2 class tasks. Extensions to SVM to tackle multi-class classification is generally performed by reducing the problem to several binary problems. The parameters of the maximum margin hyper plane are derived by solving the optimization problem. This is implemented commonly by using the Platt's sequential minimal optimization algorithm [58]. Situations where a nonlinear region can better separate the classes are handled by using kernel functions.

5.4 k-Nearest Neighbor

k-NN or k Nearest Neighbors algorithm is a non-parametric method used in classification. The algorithm effectively classifies an object based on the class of the nearest neighbor(s) i.e. it stores all training data and their respective classes and classifies new objects based on a similarity measure. The similarity measure commonly used is the Euclidean distance between the variables. The distance between two points is calculated as shown in equation 5.9

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2} \tag{5.9}$$

Other distances such as Hamming distances are used when the variables are discrete.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

(5.10)

The ideal or optimal value for K is determined based on the data. Larger values of K reduce the effect of noise on the classification but at the cost of less distinct boundaries between the classes. Figure 5-2 shows a classification example using kNN.

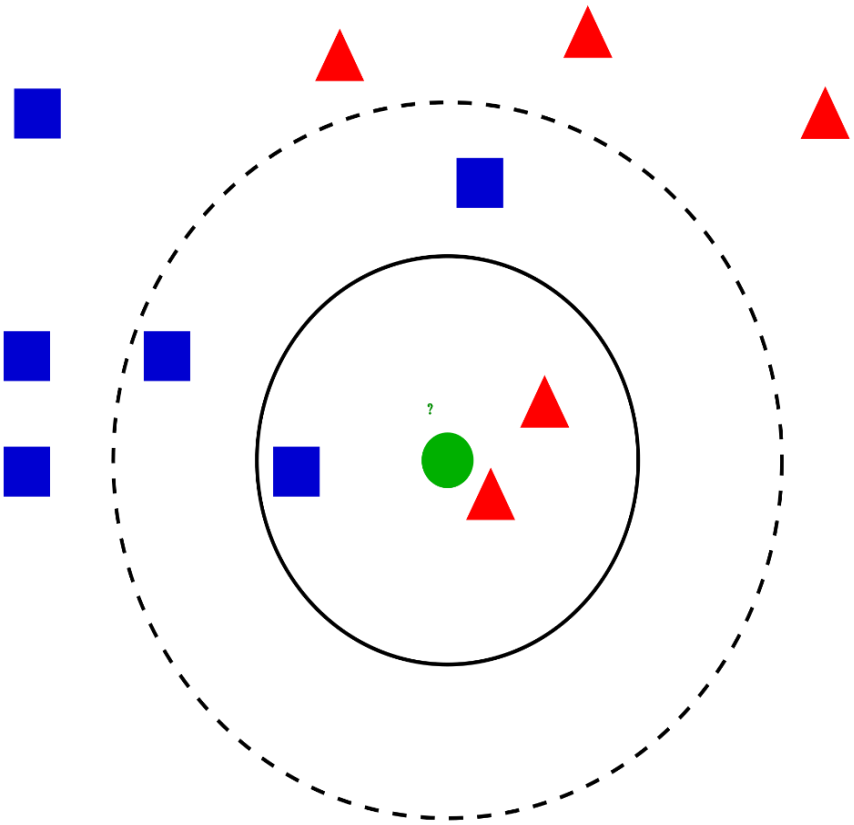

Figure 5-2 Example of k-Nearest Neighbor classification

Depending on the value of k, the sample to be classified (green circle) can be classified as a triangle or a square. The general algorithm is a type of instance based learning as no general model is constructed, rather classification is computed by a vote of the nearest neighbors of the data point. The problem is further complicated by the size of the data set. Appropriate choice of nearest neighbor search algorithms can make kNN computationally viable even for large data sets.

5.5 Neural Networks

Artificial neural networks are a family of statistical learning algorithms inspired by the way the brain is believed to function[60]. They are presented as a crude approximation of the network of cells in the brains called neurons, which can compute values from multiple inputs and are capable of machine learning due to their adaptable nature. The word neuron in the term 'neural network' refers to a function that sums a set of input values and associated weights. The word network in the term 'neural networks' refers to the interconnections between the different neurons of the system and are also referred to as synapses. Figure 5-3 shows an example neural network.

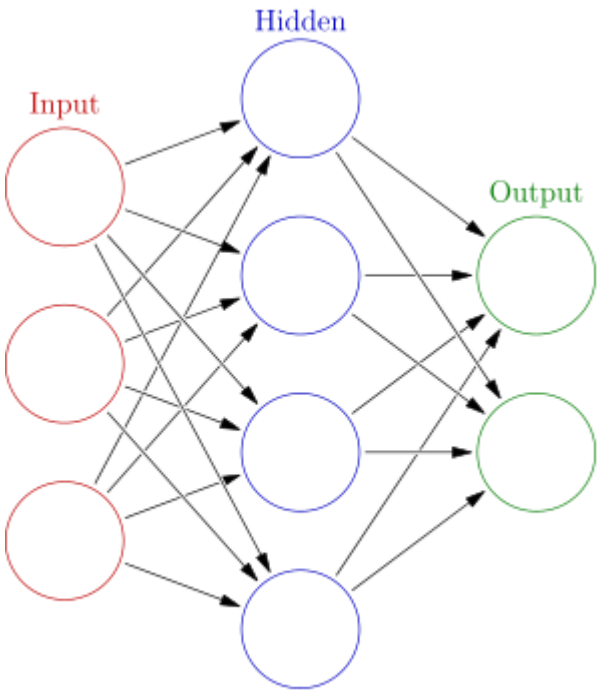

Figure 5-3 Neural Network with 1 hidden layer

Neurons are arranged into multiple layers: input, hidden and output. The input layer comprises of the inputs to the next layer. The next layer is the hidden layer. There can be multiple levels of hidden layers, all interconnected. Finally, the output layer consists of nodes indicating the various output classes.

In a supervised setting using multilayer perceptrons, a set of example pairs (x,y) are fed to the network with the goal of finding a mapping from x to y as shown in equations 5.11 and 5.12.

$$(x, y), x \in X, y \in Y \tag{5.11}$$

$$f : X \to Y \tag{5.12}$$

Each neuron in the multilayer perceptron has a nonlinear activation function, usually a sigmoid function (hyperbolic tangent or a logistic function). (ref68)

In the training phase, since the correct output class for each set of inputs is known, the output of the neural network is checked for correctness. Using this comparison, an error term is calculated for each output class which is then propagated back all the way to the input node, appropriately updating the weights used for each of the synapses. This is known as the back propagation algorithm. As more training data is presented, the weights are continually adjusted and refined to better represent the classification function. To adjust the weights, the effects of which input contributed the most to an incorrect output must also be accounted for. A commonly used cost, i.e. error term calculation used in neural networks is the mean-squared error, which tries to minimize the average squared error between the output of the network and the expected output.

5.6 Weka

Weka is a collection of machine learning algorithms for data mining tasks. Developed as a library, the algorithms can both be applied directly to data as well as called from Java applications. It also contains various tools for data pre-processing, classification, regression, clustering, association rules and visualization. [62]

The dataset on which the various algorithms are run on, is implemented in Weka by the Instances class. The external representation of this data is the Attribute-Relation File Format (ARFF). An example ARFF file is shown in figure 5-4

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Figure 5-4 Example ARFF file

An ARFF file is a text file that describes a list of instances sharing a set of attributes. The file is divided into 2 sections. The first section is the Header, which

contains a list of the attributes and their data types. Attribute declarations take the form of an ordered sequence of @attribute statements. The order in which the attributes are declared indicates the column position of that attribute in the data section. The format for the @attribute statement is:

```
@attribute <attribute-name> <datatype>
```

Each attribute can be any of the following four data types:

- Numeric,
- Nominal,
- String
- Date

The second section of the file is the data section containing the actual instance lines. Each instance is represented on a single line, with attribute values delimited by commas. Attribute values must appear in the order they were declared in the header section. Missing values can be represented by a single question mark. The values of string and nominal data types are case sensitive and any value containing a space must be quoted.

The Instances class also allows for basic validation of ARFF files. Other routines included allow data present in comma/tab separated files to be imported into ARFF files.

Weka also includes a weka.filters package that can transform datasets. It contains useful methods for resampling the dataset, removing examples, data preprocessing etc.

Weka includes algorithms for many different classification algorithms. By supplying the appropriate training and test files, the algorithms can be used to develop and use various classification algorithms. Table 5-1 shows some of the supported

classification algorithms. The output of any classification algorithm can also provide statistics regarding the classification process such as time taken to classify given samples. It also develops the confusion matrix which can be used to evaluate the performance of the classification algorithm.

Table 5-1 Classifiers in Weka tool

Classifier	**Description**
trees.J48	open source java implementation of C4.5 decision tree learner
bayes.NaiveBayes	A Naïve Bayesian learner
functions.Logistic	Logistic Regression based classifier
functions.SMO	Support Vector Machine with Sequential Minimal optimization algorithm
lazy.Kstar	Instance based learner using an entropy-based distance function
lazy.Ibk	Another instance based learner with fixed neighborhood (kNN)
Functions.MultilayerPerceptron	Multilayer Perceptron based learning algorithm

Chapter 6

Design of a learning power management system

The previous chapters have shown that different users have varied usages and power consumption varies for each of those users. Different components in the system may also be active when they are not being utilized. The varied nature of mobile device usage also shows that no single power management scheme is suitable to target all users. The following chapter details a system that can learn information about the users and intelligently manage power onboard the device. By using patterns identified in the usage of a mobile device, ideal settings for sensors can be determined and tweaked as necessary to save power. Any system that tries to identify patterns in usage and adaptively manage power must include some form of system to log data about the user. It must have a mechanism to learn from the logged data and must also have a controlling mechanism to actively control system parameters. A user specific approach allows for a customized and individualized solution to power management in mobile devices.

Section 6.1 gives a brief overview of the whole system. Section 6.2 describes the basis of the log collection system. Section 6.3 and 6.4 give a description on how the learning module functions. Section 6.5 elaborates on the power manager system.

6.1 Learning system overview

A high level overview of the learning system is described below. Figure 6.1 shows the block diagram of the system. It comprises of a logging system which also monitors for change in applications. When an application switch is detected, the relevant information regarding the system is passed on to the application classifier. The result of the classifier allows the power management and control system to invoke an application specific power management strategy and changes the system parameters as necessary.

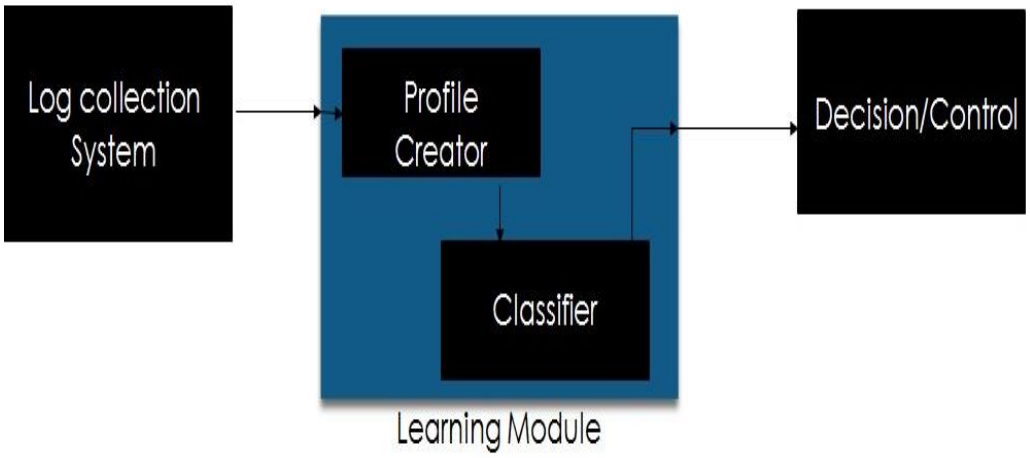

Figure 6-1 Block diagram of system

Any changes made by the user in the system parameters is also logged for the system to learn from. If the application has never been classified before, a default power management strategy is followed.

The system learns and builds a new classification model based on the logged historical data. The learning process occurs only when the device is plugged in so that the impact of the learning process is nonexistent on battery.

6.2 Data Collection system (Logger)

To manage power on a device in a manner adapted to the individual user, data about how that user utilizes their device must be captured. This is accomplished by using a custom logging application that runs as a background Android service. A combination of polling and event based logging is used to gather data about the system. A list of the various data points logged are listed in Table 6-1.

Table 6-1 List of sensors and data sources being logged

Sensor	Sensor Type	Polling frequency	Information
Audio	poll	low	Capture audio stream info
Battery Capacity	poll	low	Battery information and status
Bluetooth Radio	poll	low	Bluetooth radio information and status
Cellular Info	poll	low	Cellular network information
Data Traffic	poll	low	aggregated information on network traffic
Telephony	poll	low	information regarding the phone
Wi-Fi Radio	poll	low	Wi-Fi radio information and status
CPU Intensive Processes	poll	med	top 5 CPU intensive processes
Screen	poll	med	Screen status On/Off
Background Apps	poll	high	List of apps running in the background
Foreground App	poll	high	foreground application
Battery Charging	event	n/a	Indicates if phone is being charged
Location	event	n/a	GPS and Network location
Network	event	n/a	Information regarding the current network status

A series of threads focuses on capturing the data from specific sensors. Each thread is scheduled to wake at a set frequency so as to capture the data from the sensors and write to file. To minimize the impact of polling, different sensors and data sources are polled at different frequencies. For example, it is not necessary to poll information about the Wi-Fi Radio every second, however knowledge about which application is currently running in the foreground is. A separate thread manages the logging of various event based activities such as identifying when the phone has been connected to a power source. Therefore only when a change of state, i.e. an event, occurs is the thread invoked. Both polling and events are logged by using standard Android APIs available to access various sensors and data sources. When a new application gains focus, i.e. it is the application in the foreground, data about the application along with current system settings are then saved to file. There are three different threads being called every 10, 20, 120 seconds to poll the various sensors. Figure 6-2 shows the structure of the logging system.

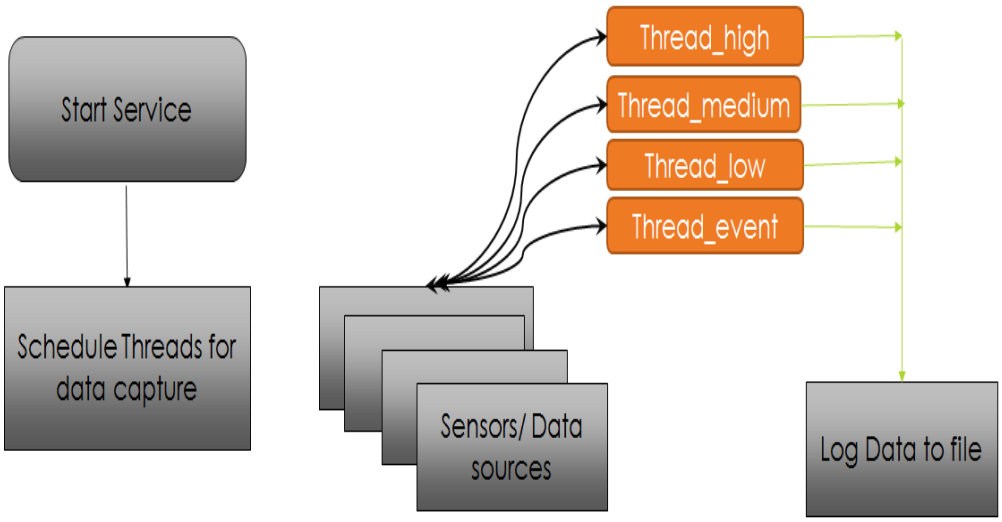

Figure 6-2 Block diagram of logging system

6.3 Learning module (Profile Creator)

The learning module comprises of 2 parts, the Profile creator and the Classifier module. The following section explains the working of the Profile creator. The profile creator takes in data being logged by the Logger and formats it into a form consumable by the classification module. In this effort, the classification module utilizes the collection of machine learning algorithms called WEKA described in Section 5.6. Thus for any classification effort, the data must be formatted into a form consumable by WEKA algorithms. The external representation of the data set to be fed into the classifier is an ARFF file. It consists of a header describing the attribute types and the data presented as a comma separated list.

```
@RELATION POWER
@ATTRIBUTE application {"com.android.launcher", "com.google.android.gallery3d", "com.android.settings", "com.microsoft.office.lync15",
"com.mobileiron", "com.android.vending", "com.google.android.googlequicksearchbox", "com.android.chrome", "org.zwanoo.android.speedtest",
"com.whatsapp", "com.viber.voip", "org.telegram.messenger", "com.skype.raider"}
@ATTRIBUTE time {morning, noon, evening, night}
@ATTRIBUTE wifi {on, off}
@ATTRIBUTE screen {on, off}
@ATTRIBUTE bluetooth {on, off}
@ATTRIBUTE screenbrightness {low, med, high}
@ATTRIBUTE powerclass {1,2,3,4,5,6,7,8,9,10,11,12}
@DATA
"com.android.chrome",evening,on,on,off,low,3
"com.android.chrome",evening,on,on,off,low,3
"com.skype.raider",night,on,on,off,high,4
"com.mobileiron" ,morning ,on,off,off,low,3
"com.whatsapp" ,noon ,on,on,on,med,8
"com.viber.voip" ,noon ,on,off,on,high,9
"com.viber.voip" ,noon ,on,off,on,high,9
"com.android.launcher" ,night,on,on,off,low,9
"com.google.android.gallery3d",noon ,off,on,on,high,3
```

Figure 6-3 Sample ARFF file created by profile creator system

Figure 6-3 shows a sample ARFF file used in our system. The list of attributes are retrieved from the logging system and then written to an ARFF file. The classifier utilizes this ARFF file to build a new and updated classification model when necessary. The files are updated as and when new information is retrieved by the logging module.

6.4 Learning Module (Classifier)

The classification module is the second part of the learning mechanism. The classification algorithm used in this system is a form of the Naive Bayesian classifier. For this module, the classification system uses the following parameters as input:

1. Application name (foreground application)
2. Time of day - this is categorized into 4 sections as morning (6AM-12PM, afternoon (12-6PM), evening (6PM-12AM) and night (12-6AM).
3. Current Wi-Fi Module Status - On/off
4. Screen status - On/Off
5. Bluetooth status – On/Off
6. Current screen brightness - low, medium and high

The various inputs are then categorized into one of 12 output classes based on the input parameters. This classification allows the Decision module i.e. power manager, to adjust system parameters as necessary to maximize power savings. The current classifier module output is based exclusively on the different data interfaces and the screen brightness of the device.

Initially a default ARFF file loaded with information about some commonly used applications is used to build the classifier model. This acts as our training data set. All classifications then made are based on this constructed model. As new information is captured about the various applications and the system settings, the Profile Creator updates the ARFF file. The default classifier model is then updated using the newly captured information stored in the ARFF file. The classifier model is updated at least

every 3 days when the device is plugged in to a power source and has been in an idle state for at least 10 minutes.

Real time classification of applications is made every time an application change is detected. A simplified classification example based on the sample training ARFF file shown in fig 6-4 is given below.

```
@RELATION POWER
@ATTRIBUTE location {home, office, other}
@ATTRIBUTE time {morning, noon, evening, night}
@ATTRIBUTE wifi {on, off}
@ATTRIBUTE application {netflix, youtube, camera, game}
@ATTRIBUTE wififlip {yes, no}
@DATA
other,evening,on,game,no
other,evening,off,netflix,yes
other,night,off,camera,no
other,night,on,game,yes
office,morning,on,game,no
office,morning,on,game,no
office,morning,off,netflix,yes
home,night,on,netflix,no
home,night,on,youtube,no
home,noon,off,game,no
home,evening,on,netflix,no
home,morning,on,game,no
office,noon,off,netflix,yes
```

Figure 6-4 Simplified training file

As seen in the above file, we have 4 input attributes under consideration and a single binary output class - whether the Wi-Fi module should be flipped on/off (Wf). Equation 5.4 can then be used to classify any new instance.

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \tag{5.4}$$

Consider a new input instance to be classified as the following tuple:

{other, night, on, netflix}

From the training data, we can calculate the following probabilities:

$$P(Wf = yes) = \frac{4}{13} \tag{6.1}$$

$$P(Wf = no) = \frac{9}{13} \tag{6.2}$$

$$P(location = other | Wf = yes) = \frac{2}{4} \tag{6.3}$$

$$P(location = other | Wf = no) = \frac{2}{9} \tag{6.4}$$

$$P(application = netflix | Wf = yes) = \frac{3}{4} \tag{6.5}$$

$$P(application = netflix | Wf = no) = \frac{2}{9} \tag{6.6}$$

$$P(Wifi = on \mid Wf = yes) = \frac{1}{4} \tag{6.7}$$

$$P(Wifi = on \mid Wf = no) = \frac{7}{9} \tag{6.8}$$

$$P(TOD = night | Wf = yes) = \frac{1}{4} \tag{6.9}$$

$$P(TOD = night | Wf = no) = \frac{3}{9} \tag{6.10}$$

Therefore, for the hypothesis that the output class Wf = yes, the posterior probability is calculated as:

$$P(location = other, TOD = night, Wifi = on, application = netflix | Wf = yes)$$

$$\frac{4}{13} * \frac{2}{4} * \frac{3}{4} * \frac{1}{4} * \frac{1}{4} = \frac{6}{832} = 0.00721 \quad (6.11)$$

$$P(location = other, TOD = night, Wifi = on, application = netflix | Wf = no) =$$

$$\frac{9}{13} * \frac{2}{9} * \frac{2}{9} * \frac{7}{9} * \frac{3}{9} = 28/3159 = 0.00886 \quad (6.12)$$

Therefore, the output class of the new instance will be Wf = No.

6.5 Power Manager (Decision Controller)

The output of the classifier stage is a single value categorizing the input variables into one of 12 classes. Each of these classes is a combination of settings for Wi-Fi, Bluetooth, Screen brightness and Mobile data. The power manager utilizes standard Android APIs to control each of the above parameters. Table 6.2 shows all the possible output combinations possible. By having application specific settings, a customized power saving setting can be applied for each application, to maximize the power savings. As shown earlier, each user has a unique usage pattern. The different patterns affect power consumption on the device in ways unique to each user. By targeting the specific application mix, a unique power profile can be determined and savings can be achieved.

Table 6-2 Power Manager output classes

Class	Wifi	Screen Brightness	Bluetooth
1	On	High	On
2	On	High	Off
3	On	Medium	On
4	On	Medium	Off
5	On	Low	On
6	On	Low	Off
7	Off	High	On
8	Off	High	Off
9	Off	Medium	On
10	Off	Medium	Off
11	Off	Low	On
12	Off	Low	Off

Additionally, as the service is running in the background, the service is capable of determining how and when various data interfaces can be turned off based on the type of application being used. Further power savings can be achieved by turning off the haptic feedback for the applications as this prevents power being used by the vibration motor.

6.6 Analysis of various classification algorithms

Table 6-3 shows a brief comparison of the different classification algorithms and the applicability of each of them to this project.

Table 6-3 Comparison of various classification algorithms

Classifier	Data Size	Learning Speed	Accuracy	Comments
Naïve Bayesian	Small	Fast	Medium	cannot understand correlation between features; assumption of independence between features
kNN	Medium	Slow	High	memory intensive
Decision Tree	Medium	Medium	Medium	easier to handle than SVMs
SVM	Large	Slow	High	High accuracy and fast classification
Neural networks	Large	Slow	Very High	High computation complexity.

Given that the primary goal of this effort is to increase power savings on mobile devices, any algorithm being run to identify patterns in usage must not be a burden on the system. With limited computing and energy resources at hand, the algorithm needs to be computationally efficient and fairly accurate.

As shown in Table 6-3, the Naive Bayes classifier has a fast learning speed, as the essential algorithm boils down to the calculation of probabilities from observed data and application of these to new unknown input vectors. The main assumption made for this algorithm is the independence between features. Therefore, the algorithm cannot

learn and adapt to any dependencies that may exist between features. From both learning and classification points of view, the Naive Bayes is a fast classifier. The other advantage is that to build the classifier, it needs a relatively small training data set.

Decision Trees are fairly accurate class of nonparametric classifiers, i.e. they do not make assumptions about the distributions of the attributes or the class. The speed of classification is extremely fast as the worst case complexity is $O(w)$ where the depth of the tree is w. The algorithm however can suffer from creation of trees that do not generalize well from training data, i.e. overfits the data.

k-NN learning algorithms are a class of instance based learning algorithms. They are also non parametric, hence no assumptions are made on the distribution of underlying data. However, k-NN algorithms are highly memory intensive as they store all training data in memory. Thus larger training data sets can stress the memory adversely. The computational costs are also high as classification of any new instance requires comparison of the instance to the complete data set stored in memory. This is highly unsuitable for use in mobile devices which are constrained in terms of processing power and memory, even though they can be fairly accurate.

Support Vector Machine classification algorithms suffer from large training times. This is because of the time taken to solve the optimization problem while determining the hyper plane to divide the data into classes. This is a computationally intensive step and therefore generally unsuited for use in mobile devices. This can however be overcome by training the classifier either offline or only when the device has sufficient resources. Proper training also requires a large training data set. Once trained however, the

classification speed of a new instance is fast. The accuracy of classification is also very high.

Neural networks are not particularly suited for this particular task because of the resource constraint. The training of neural networks is highly complex and computationally intensive. The time taken to train is also very high and usually training requires a large training data set. Thus the high training time and complexity make the implementation of the algorithm on a mobile device infeasible.

Chapter 7

Results

The following chapter describes the results of the learning classifier. Results regarding impacts of the classifier on power as well as the classification step in mobile devices are presented.

7.1 Methodology

To determine the effect of the power management scheme implemented by the classifier, two separate experiments were done. First, a profile representative of the average smartphone user was built based on research data. Using this profile, a baseline power measurement was made utilizing a combination of the on board battery statistics and PowerTutor [68], [69]. The baseline power numbers were then compared to the learning power management system. Section 7.2.1 shows the design of the profile as well as power consumption values for this profile.

Additionally, the system was also tested against each of the 6 demographic profiles built in Chapter 4. Power measurements were made using the same setup used for the average smartphone user profile. Section 7.2.2 shows the detailed power breakdown for these 6 profiles.

The classifier performance was also recorded by using data from a real user to train and test the implemented Naïve Bayesian Classifier. This was done by using previously stored ARFF files which are fed into the Naive Bayesian Classifier implemented as part of the Weka package. The ARFF file consists of 3000 records. The attributes present in the ARFF files are identical to the input attributes expected by the

classifier on the mobile device. Section 7.3 compares the performance of various classifiers using the stored ARFF file.

7.2 Power Measurements

7.2.1 Measurements for the average smartphone user

To measure the impact of the learning power management system, a profile was developed mirroring the use of a smartphone by the average user. Research has shown that the average smartphone user utilizes their device for approximately an hour each day.[70] Figure 7-1 shows the breakdown of time of use for each of these applications.

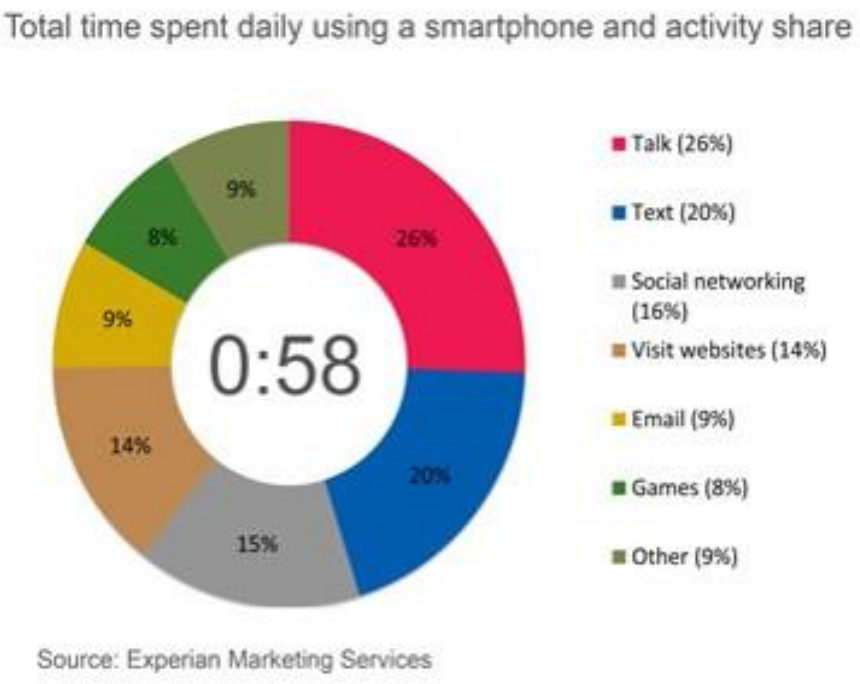

Figure 7-1 Smartphone time of use breakdown

Figure 7-2 shows the frequency of use of each of the applications and the time spent in each app. Using this data, a profile consisting of multiple applications was developed, with different varied number of invocations and different runtimes.

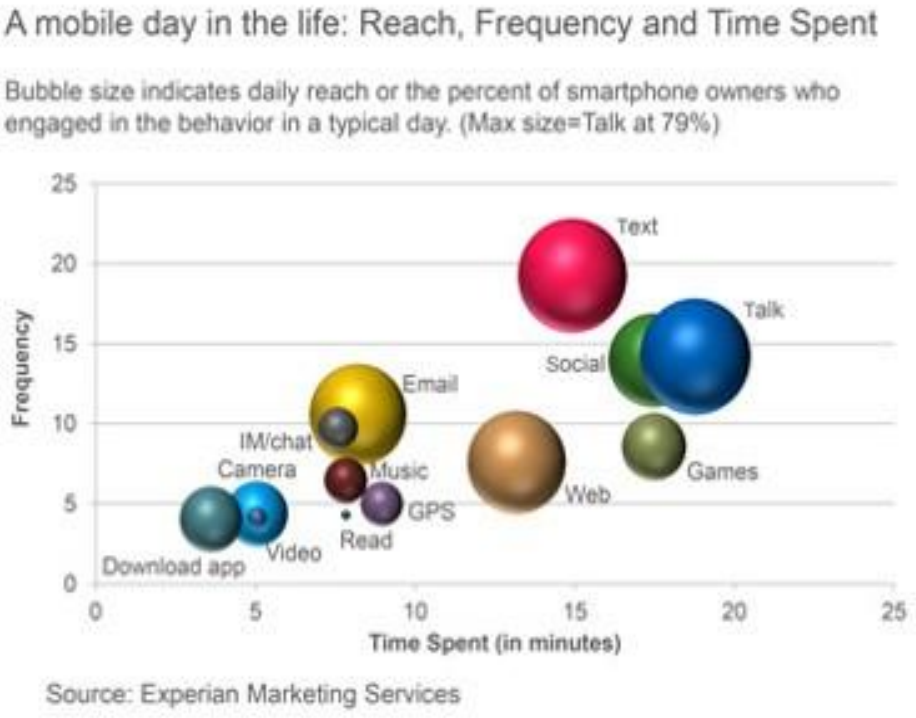

Figure 7-2 Frequency of use of various applications

Table 7-1 shows the time splits for the runtime of each application. The profile was then run multiple times on an x86 based Android device, to obtain both the baseline system power consumption values and the power consumption values with an idealized learning power management system in place. Only the various data interfaces and screen brightness settings were varied as part of the idealized system. Table 7-2 shows the average difference in energy used across multiple runs.

Table 7-1 Time splits of various applications

Activity	Time of Use in minutes
Talking on Phone	14.5
Texting	12
Social Networking	10
Browsing the Web	8.5
using Email	5
Gaming	5
Camera	1
Maps	1
Other miscellaneous apps	3

Table 7-2 Power consumption comparision for average smartphone user profile

Average Power Draw(mW)	Default settings	Idealized learned settings	Delta %
Run 1	1658	1403	18.18%
Run 2	1702	1487	14.46%
Run 3	1675	1408	18.96%
Run 4	1423	1263	12.67%
Run 5	1598	1421	12.46%
Average	1611.2	1396.4	15.38%

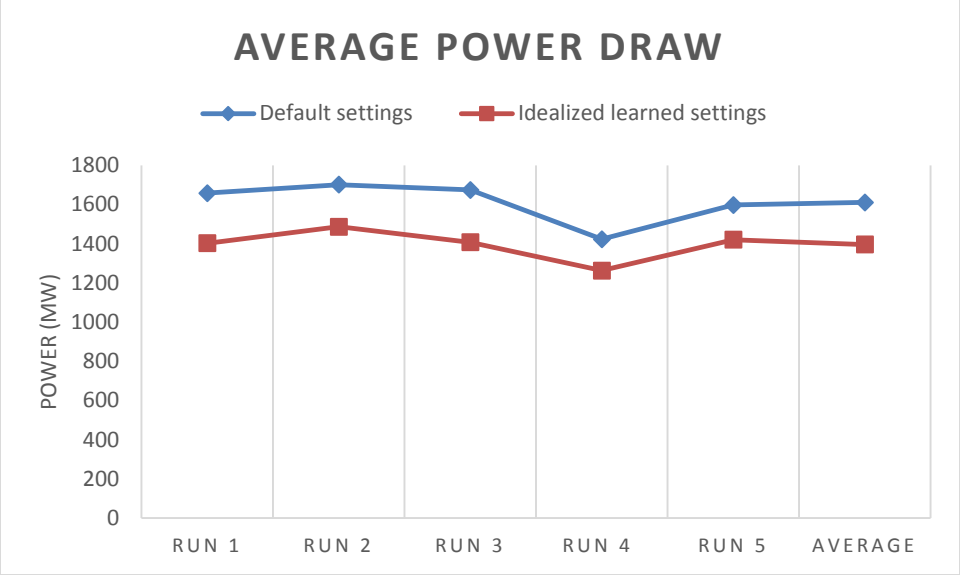

Figure 7-3 Power Consumption for average smartphone user profile

7.2.2 Measurements for various demographic profiles.

In addition to measuring the impact of the learning power management system on an average smartphone user, the results of the system on various demographic profiles developed in chapter 4 were also tested. Power measurements were made by the use of PowerTutor, an online power estimation tool developed at University of Michigan.

Since each of the demographic profiles were developed to run for a time period of 600s, the same profile/ application mix was executed to determine the total power consumed by the device with the learning system active. All baseline measurements were made with the Wi-Fi and Bluetooth modules turned on and screen brightness set to Auto.

Table 7-3 Power Consumption values for the 6 profiles (mW)

Power Consumed (mW)	Default	With learning system active	Default	With learning system active	Default	With learning system active	Default	With learning system active	Default	With learning system active	Default	With learning system active
	18-34 M		18-34F		35-54M		35-54F		55+M		55+F	
Run 1	1180	1081	1234	1099	1232	1043	1228	1031	1279	1045	1260	1025
Run 2	1235	1125	1189	1100	1304	1091	1285	1108	1408	1101	1184	1003
Run 3	1164	1093	1277	1158	1277	1168	1203	1127	1164	1026	1247	1054
Run 4	1182	1065	1300	1206	1310	1121	1111	1095	1252	1069	1293	1039
Run 5	1276	1067	1198	1081	1249	1084	1375	1127	1243	1051	1162	1011

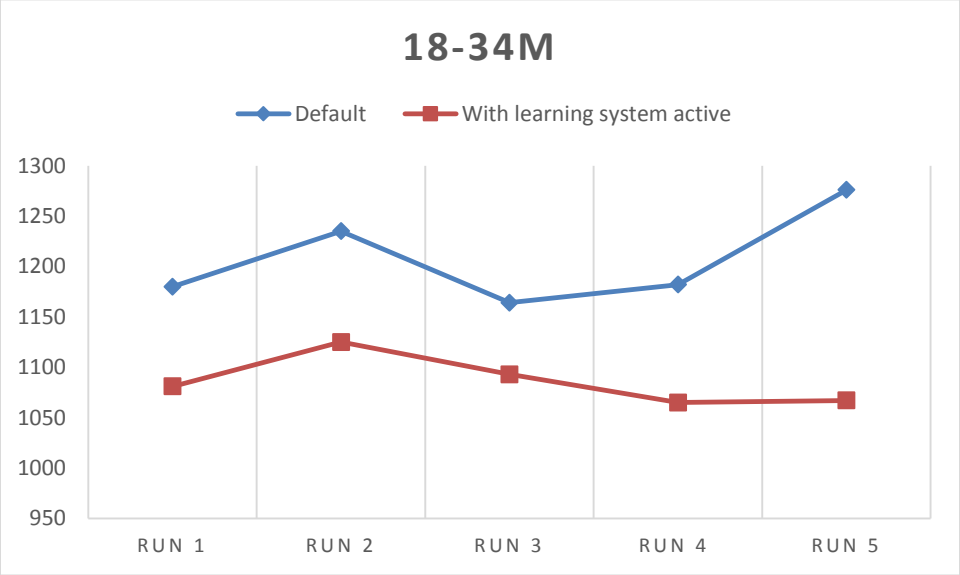

Figure 7-4 Comparison of power consumption for 18-34 M profile

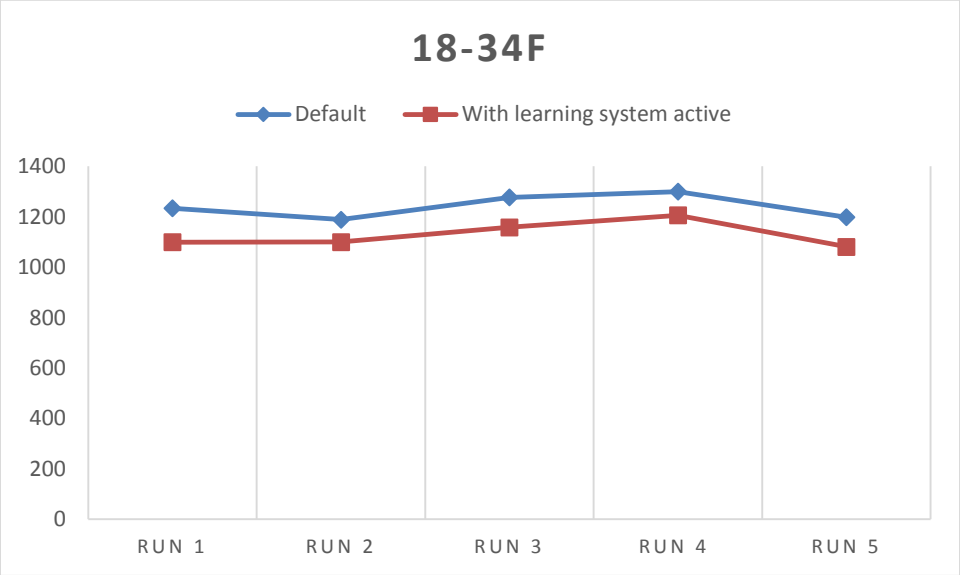

Figure 7-5 Comparison of power consumption for 18-34 F profile

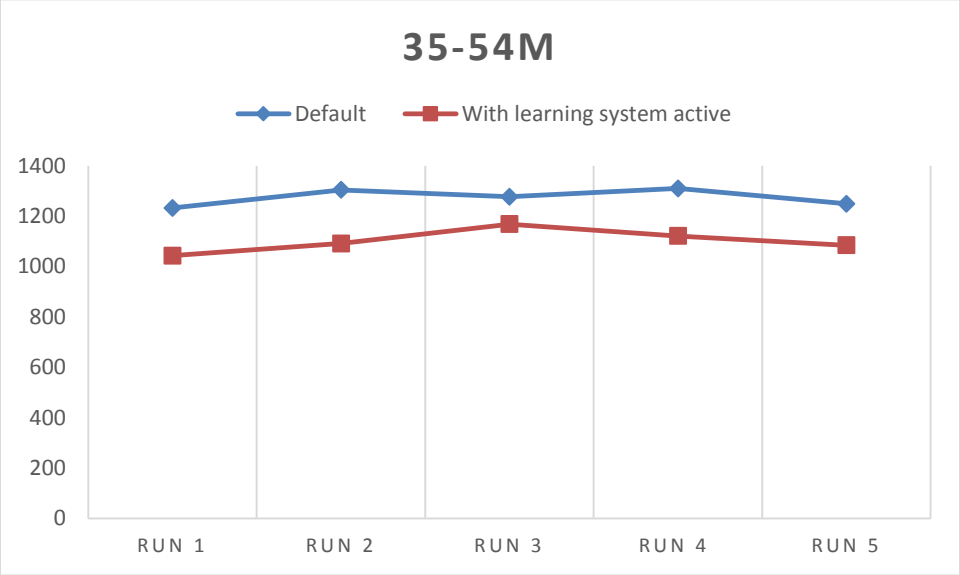

Figure 7-6 Comparison of power consumption for 35-54 M profile

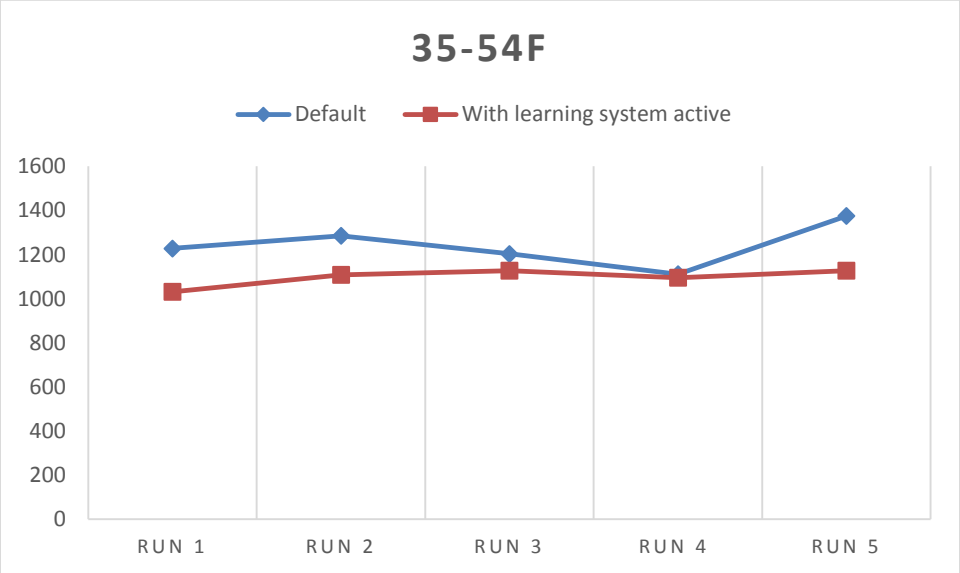

Figure 7-7 Comparison of power consumption for 35-54 F profile

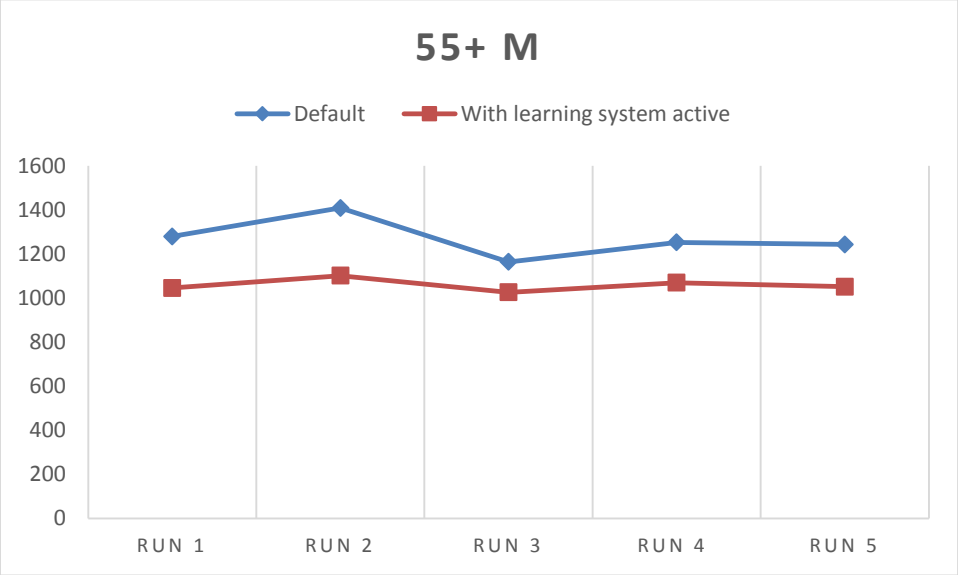

Figure 7-8 Comparison of power consumption for 55+ M profile

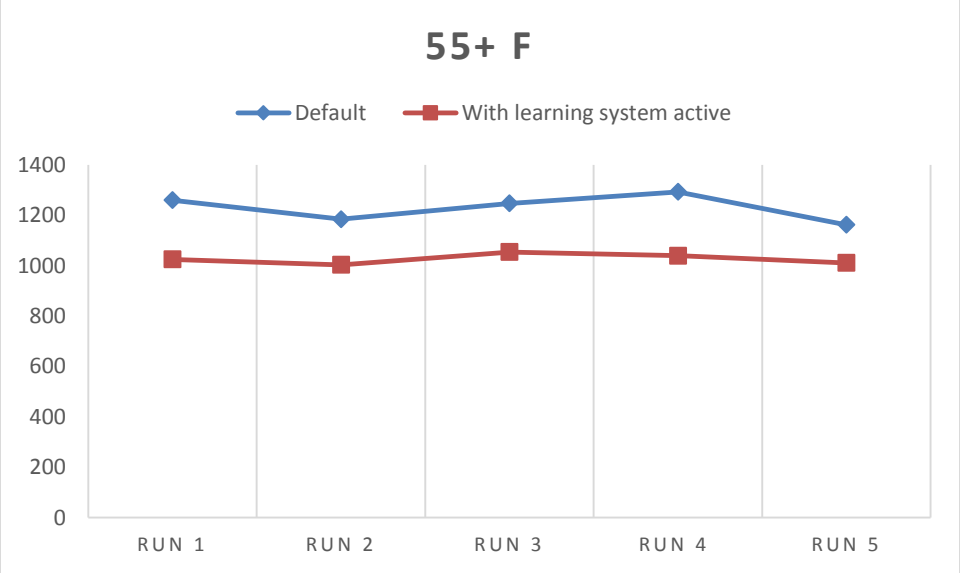

Figure 7-9 Comparison of power consumption for 55+ F profile

7.3 Performance of the classifier

To compare the performance of the classifiers, a trained dataset matching the usage collected from a real user was taken and an 80/20 split was used to train and test the classifier on a personal computer. Table 7-4 shows the various computed statistics for the different classifiers.

Table 7-4 Classifer Performance Statistics

	Correctly Classified %	Incorrectly Classified %	Kappa Statistic	RMS Error	Weighted Average ROC	Time Taken to learn (s)
Naïve Bayesian	60.33	39.667	0.4847	0.2002	0.897	0
kNN	63.5	36.5	0.5304	0.2135	0.858	0
MultiLayer Perceptron	72	28	0.6485	0.1998	0.922	39.85
SVM using SMO	70	30	0.6208	0.2578	0.89	2.32
J48 Decision tree	70	30	0.6235	0.176	0.907	0.03

Chapter 8

Conclusion and Future Work

In today's fast paced world, mobile devices of all types have developed into full blown personal computing devices, compressed into a small form factor. The combination of computing power along with connectivity of these devices to the internet allows people to perform a wide range of activities that was not possible just a few years ago. These devices allow people to communicate, entertain, connect with others both professionally and socially. As more people use such devices which increase in computing power year to year, developers have come up with unique methods to utilize the smartphone to solve various problems. This however comes at a cost as the small form factor of these devices limits the total battery capacity. With high performance expected by users at all times, applications grow more power hungry. While the rate of change in computing technology has been high, battery technology has not kept pace. Therefore managing power on these mobile computing and communication devices is a critical task. These methods must be able to minimize power consumption without any impact on QoS to the user.

However, due to the varied nature of the mobile user, no single power management strategy can successfully tackle this challenge. This research effort shows that even a simple demographic breakdown of users have varied power use profiles. The mix of applications used by each user varies and each profile or mix of applications has a unique power consumption profile.

This unique power consumption pattern can be used to develop an individualized power management scheme, one that can adapt to the individual user. Based on the application mix, it was shown that a power saving scheme customized to the user can provide greater power savings.

This effort provides a solution by implementing an on device scheme that learns about the mix of applications used by the user, the various sensors and interfaces required and manages them, improving the overall power consumed by the device by minimizing unnecessary power losses. It uses a Naive Bayesian learning and classification algorithm to analyze, understand user profiles and manage power accordingly. The developed strategy is tested on the different demographic profiles, obtaining a maximum of 19% and an average of 12-15% in energy savings over the default power management scheme present in Android smartphones.

8.1 Future Work

The effort presented in this effort tackled the solution by implementing a Naive Bayesian learning algorithm to manage power. While the solution does show promising results, much work can be done to improve the scheme.

One such solution is to utilize a better learning algorithm to learn and classify the application mix. Potential solutions include implementing a Support Vector Machine based learning system. While such algorithms are expensive from a computation and power perspective, potential solutions include running the algorithms only when the device is being charged. Another such solution is to offload the computation to a server, thus minimizing the impact of using computationally heavy algorithms.

Another solution is to control more system parameters than presented as part of this effort. It was shown that better CPU frequency scaling can improve overall power consumption statistics of the device. Targeting CPU scaling in conjunction with other sensors and interfaces can help improve the overall system.

Other future work would include designing power management strategies to be a part of the operating system, thus gaining more fine-grained control over the multitude of components present in a mobile device.

Appendix A

Comparison of Android Runtime (ART) and Dalvik VM from a power perspective

Google introduced a new runtime called Android runtime (ART) with the 4.4.4 KitKat release of Android. It was included as a separate option, along with the default option, i.e. Dalvik. Since then, ART has undergone further development and with the release of Android 5.0, it has completely replaced Dalvik as the runtime.

The big change in ART compared to Dalvik is the introduction of AOT, i.e. Ahead-of-Time compilation. Here all application code is compiled to native code at app install time. Dalvik employed the JIT (Just-in-Time) compiler to perform bytecode to native code compilations. With the AOT principle, all compilation is done only once. As there exists no need for the interpreter or the JIT compiler to perform any tasks in the ART environment, a direct result is a major reduction in CPU cycles consumed when running any application, which translates to savings in power consumption.

The following section compares the Dalvik and ART from a power perspective. As Android 4.4.4 KitKat supports both Dalvik and ART, it was chosen as the operating system version to perform experiments on power consumption. The device used for this experiment is the Google Nexus 5.

To test the power consumption across the two runtimes, a fully charged device was made to run the (IcyRocks workload, a workload developed by Intel Corp,) workload while measuring the battery capacity and battery current. A script was developed to capture the above values along with all other battery related statistics every 5 minutes until the device was completely discharged. Figure A-1 shows the logged battery capacity over time.

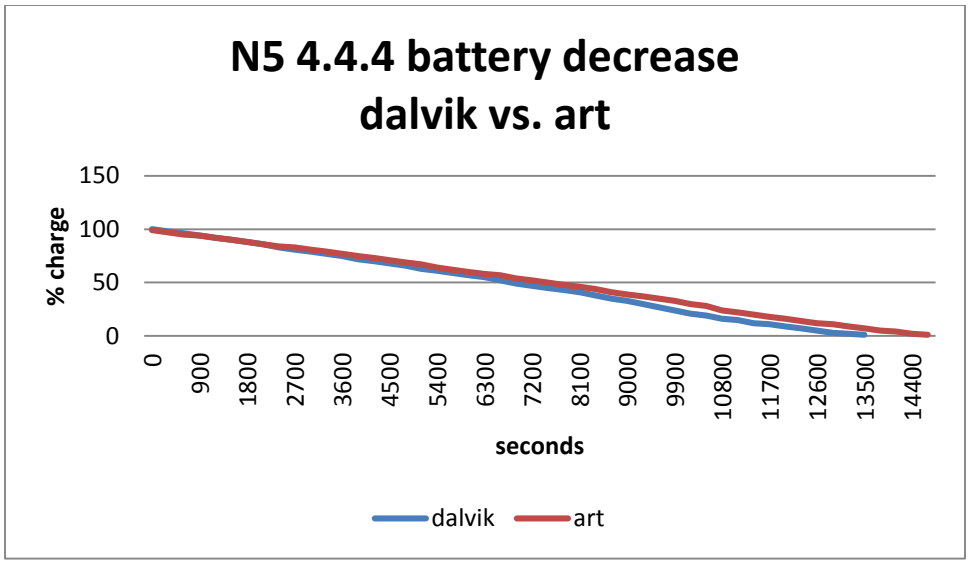

Figure A-1 Battery discharge curve Dalvik v ART

As seen, the Dalvik VM executes the workload for a total of 13495 seconds before the phone is completely discharged. For the same workload, with only the runtime being changed to ART, the phone executes the workload for 14699 seconds before a complete discharge. Therefore, the net runtime has increased by approximately 9%.

Table A-1 Performance Statistics, Dalvik v ART

	Animation/s	FPS
Dalvik	4681.206255	23.9036873
ART	7889.440923	29.2820734
% diff	68.5%	22.5%

Table A-2 shows some of the performance related statistics reported by the workload. It is seen that the Frames drawn per second statistic has increased by 22.5%, while the total number of animations drawn per second has increased by 68.5%. Thus,

the power consumption of ART is shown to be better than Dalvik, with a simultaneous increase in performance.

References

[1] http://en.wikipedia.org/wiki/Power_management

[2] http://en.wikipedia.org/wiki/Advanced_Power_Management

[3] http://en.wikipedia.org/wiki/Advanced_Configuration_and_Power_Interface

[4] http://www.acpi.info/presentations/ACPI_Overview.pdf

[5] www.ieee802.org/3/eee_study/public/mar07/chalupsky_01_0307.pdf

[6] http://www.gartner.com/newsroom/id/2944819

[7] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[8] http://www.kandroid.org/online-pdk/guide/power_management.html

[9] http://grail.cba.csuohio.edu/~matos/notes/cis-493/lecture-notes/Android-Chapter03-Life-Cycle.pdf

[10] https://developer.android.com/about/versions/kitkat.html

[11] https://developer.android.com/about/versions/android-5.0.html

[12] http://www.greenbot.com/article/2449812/android-l-features-revealed-project-volta-boosts-battery-life.html

[13] http://source.android.com/devices/tech/dalvik/

[14] http://wear.techbrood.com/preview/images/battery_historian.png

[15] https://developer.android.com/about/dashboards/index.html

[16] http://www.forbes.com/sites/chrisversace/2013/08/21/what-do-consumers-want-in-a-new-smartphone/ --- April 2013

[17] http://www.theguardian.com/technology/2014/may/21/your-smartphones-best-app-battery-life-say-89-of-Britons --- May 21, 2014

[18] https://www.surveymonkey.com/blog/en/blog/2014/04/02/really-want-smartphone/ --- April 2014

[19] 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.s

[20] 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

[21] 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

[22] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, J. A. Landay, "MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones," in MOBISYS, pp. 57-70, 2007

[23] Metri, G., Agrawal, A., Peri, R., & Shi, W. (2012, December). What is eating up battery life on my smartphone: A case study. In Energy Aware Computing, 2012 International Conference on (pp. 1-6). IEEE.

[24] A. Shye, B. Scholbrock, G. Memik, "Into the Wild: Studying Real User Activity Patterns to Guide Power Optimizations for Mobile Architectures," in MICRO-42 '09, pp. 168-178, Jan. 2009

[25] K. Choi, R. Soma, M. Pedram, "Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Trade-Off Based on the Ratio of Off-Chip Access to On-Chip Computation Times," in TCAD '04, 2004

[26] Mallik, J. Cosgrove, R. Dick, G. Memik, P. Dinda, "PICSEL: Measuring User-Perceived Performance to Control Dynamic Frequency Scaling," in ASPLOS '08, Mar. 2008

[27] Y. Man, Y. Liu, "Towards an energy-efficient framework for location-triggered mobile application," Telecommunication Networks and Applications Conference (ATNAC), 2012 Australasian , vol., no., pp.1,6, 7-9 Nov. 2012

[28] C. Lee, M. Lee, D. Han, "Energy efficient location logging for mobile device," in SAINT '11, pp. 84, Oct. 2010

[29] M. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, M. J. Neely, "Energy-Delay Tradeoffs in Smartphone Applications," in MOBISYS '10, pp. 255-270, Jun. 2010.

[30] Metri, G., Shi, W., Brockmeyer, M., Agrawal, A., "BatteryExtender: an adaptive user-guided tool for power management of mobile devices." in UBICOMP '14, pp. 33-43, Sep. 2014.

[31] C. Wang; F. Yan; Y. Guo; X. Chen, "Power estimation for mobile applications with profile-driven battery traces," in ISLPED '13 , pp.120,125, 4-6 Sept. 2013

[32] Y.-W. Kwon and E. Tilevich. Reducing the energy consumption of mobile applications behind the scenes. In 29th IEEE International Conference on Software Maintenance, 2013

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

[34] https://play.google.com/store/apps/details?id=com.ijinshan.kbatterydoctor_en

[35] https://play.google.com/store/apps/details?id=com.dianxinos.dxbs

[36] http://tasker.dinglisch.net/

[37] http://www.samsung.com/global/microsite/galaxys5/features.html

[38] http://www.androidcentral.com/using-ultra-power-saver-samsung-galaxy-s5

[39] http://www.androidcentral.com/using-extreme-power-saving-mode-htc-one-m8

[40] http://blog.htc.com/2014/05/extreme-power-saving-mode/

[41] https://www.thinkwithgoogle.com/tools/our-mobile-planet.html

[42] https://www.thinkwithgoogle.com/tools/consumer-barometer.html

[43] https://orange.blender.org/

[44] http://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began

[45] http://techcrunch.com/2014/11/02/is-mobile-gaming-the-new-core-gaming/

[46] https://play.google.com/store/apps/details?id=com.rovio.angrybirds

[47] https://play.google.com/store/apps/details?id=com.iUnity.angryBots&hl=en

[48] https://play.google.com/store/apps/details?id=com.glbenchmark.glbenchmark27&hl=en

[49] Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall. ISBN 978-0137903955

[50] http://scikit-learn.org/stable/modules/naive_bayes.html

[51] H. Jung, M. Pedram, "Improving the Efficiency of Power Management Techniques by Using Bayesian Classification," in ISQED '08, pp. 178-183, Mar. 2008.

[52] Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 2010

[53] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[54] http://en.wikipedia.org/wiki/Decision_tree_learning

[55] http://en.wikipedia.org/wiki/Naive_Bayes_classifier

[56] J.R. Quinlan. C4. 5: programs for machine learning. Morgan Kaufmann, 1993.

[57] Berwick, Robert. "An Idiot's guide to Support vector machines (SVMs)." Retrieved on October 21 (2003): 2011.

[58] Platt, John (1998), Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines

[59] http://www.solver.com/xlminer/help/neural-networks-classification-intro

[60] http://en.wikipedia.org/wiki/Artificial_neural_network

[61] http://en.wikipedia.org/wiki/Multilayer_perceptron

[62] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[63] Zheng, Fei, and Geoffrey I. Webb. "A comparative study of semi-naive Bayes methods in classification learning." Proceedings of the fourth Australasian data mining conference (AusDM05). 2005.

[64] Martin, J. Kent, and D. S. Hirschberg. "On the complexity of learning decision trees." International Symposium on Artificial Intelligence and Mathematics. 1996.

[65] courses.cs.tamu.edu/rgutier/cs790_w02/l8.pdf

[66] http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/

[67] www.cs.columbia.edu/~mcollins/em.pdf

[68] 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.

[69] https://play.google.com/store/apps/details?id=edu.umich.PowerTutor

[70] http://www.experian.com/blogs/marketing-forward/2013/05/28/americans-spend-58-minutes-a-day-on-their-smartphones/

[71] http://www.msoon.com/LabEquipment/PowerMonitor/

[72] Barton, John J., Shumin Zhai, and Steve B. Cousins. "Mobile phones will become the primary personal computing devices." Mobile Computing Systems and

Applications, 2006. WMCSA'06. Proceedings. 7th IEEE Workshop on. IEEE, 2005.

[73] Carroll, Aaron, and Gernot Heiser. "An Analysis of Power Consumption in a Smartphone." USENIX annual technical conference. 2010.

[74] Maloney, Sean, and Ivan Boci. "Survey: Techniques for Efficient energy consumption in Mobile Architectures." Power (mW) 16.9.56 (2012): 7-35.

[75] http://anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l/

[76] Donohoo, Brad K., Chris Ohlsen, and Sudeep Pasricha. "AURA: An application and user interaction aware middleware framework for energy optimization in mobile devices." Computer Design (ICCD), 2011 IEEE 29th International Conference on. IEEE, 2011.

[77] Donohoo, B., et al. "Context-Aware Energy Enhancements for Smart Mobile Devices." (2013): 1-1.

[78] Donohoo, Brad Kyoshi. Machine learning techniques for energy optimization in mobile embedded systems. Diss. Colorado State University, 2012.

[79] http://elinux.org/Android_Power_Management

[80] http://www.kandroid.org/online-pdk/guide/power_management.html

[81] http://en.wikipedia.org/wiki/Machine_learning

[82] http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536

[83] http://www.nielsen.com/us/en/insights/news/2014/smartphones-so-many-apps--so-much-time.html

[84] http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/

[85] Z. Guo, S. Balasubramanium, R. Zlatanovici, T.-J. King and B. Nikolic. FinFET-based SRAM design. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, August 2005, pp. 2"7

[86] B. Yu, L. Chang, S. Ahmed, H. Wang, S. Bell, C.-Y. Yang, C. Tabery, C. Ho, Q. Xiang, T.-J. King, J. Bokor, C. Hu, M-R. Lin and D. Kyser. FinFET scaling to 10 nm gate length. Electron Devices Meeting, 2002. IEDM '02 Digest. International, pp. 251"254, 2002.

Biographical Information

Ashwin Arikere has completed his PhD in Computer Engineering under the guidance of Dr. Roger Walker at the University of Texas at Arlington. Prior to this, he received his master's degree in Computer Engineering from the University of Texas at Arlington in the year 2011. His Master's thesis dealt with the Performance Tuning of Embedded Template Analysis Tool which he successfully defended under the guidance of Dr. Roger Walker.

While working on his PhD, Ashwin served as a Graduate Teaching Assistant for the Embedded Systems and Real Time Embedded Systems courses. He has also been interning at Intel Corp's Software Services Group since the year 2012.

He wishes to continue working in Android Performance Analysis at Intel Corporation.