DELHI TECHNOLOGICAL UNIVERSITY

A Major Project-2 Report

on

"A Soft Computing Approach to Designing Energy Efficient Applications in Smartphone devices"

Guided By

Dr. Daya Gupta
Professor,
Department of Computer Engineering,
Delhi Technological University.

Dr. Vijay Rao
Scientist,
Institute for Systems Studies and Analyses (ISSA),
Defence Research and Development Organization, Delhi.

Submitted By :

Kuldeep Prakash
(2K11/SWT/11)
Delhi Technological University

D E C L A R A T I O N

I hereby declare that the Report of the Major project-2 Work entitled “A Soft Computing Approach to Designing Energy Efficient Applications in Smartphone devices” which is being submitted to the Delhi Technological University, in partial fulfillment of the requirements for the Sixth semester MAJOR PROJECT-2 course of the Master of Technology Degree in Software Technology in the Department of Computer Engineering, is a bonafide report of the study carried out by me. The material contained in this report has not been submitted to any University or Institution for the award of any degree.

____

Kuldeep Prakash

Department of Computer Engineering

Place: Delhi Technological University, Delhi.

Date:

CERTIFICATE

This is to certify that the Major Project-2 Report entitled "**A Soft Computing Approach to Designing Energy Efficient Applications in Smartphone devices**" submitted by **KULDEEP PRAKASH** (Roll Number: **2K11/SWT/11**) as the record of the work carried out by him, is a bonafide work which is not submitted to any other university or institution in best of my knowledge, is accepted as the major project-2 Report submission in partial fulfillment of the requirements for the award of degree of sixth semester **MAJOR PROJECT-2** course of the **Master of Technology** Degree in **Software Technology** in the **Department of Computer Engineering**, Delhi Technological University, Delhi.

Dr. Daya Gupta
Professor,
Department of Computer Engineering, DTU,
Delhi.

Dr. Vijay Rao
Scientist,
Institute for Systems Studies and Analyses (ISSA),
Defence Research and Development Organization, Delhi.

ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Software Technology.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Dr. Daya Gupta**, Professor, Department of Computer Engineering, Delhi Technological University and **Dr. Vijay Rao**, Scientist, Defence Research and Development Organization for their valuable guidance and support in all the phases from conceptualization to final completion of the project.

I humbly extend my grateful appreciation to my friends whose moral support made this project possible.

Last but not the least, I would like to thank all the people directly and indirectly involved in successfully completion of this project.

Kuldeep Prakash

Roll No. 2K11/SWT/11

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

This chapter gives introduction to various Soft Computing techniques and presents the motivation, scope and problem statement of the project. The chapter ends with a brief description of how this thesis is organized.

1.1 General Concepts

With increasing demand for smart phones as mobile handheld computing devices there is an imperative need for increasing the battery life, which is a very crucial resource for running the applications on the device. A conventional way of handling the situation typically has been to kill the applications based on usage, Context and switch to a core set of functionality that are minimally required to run the devices. As there is no limit on the number of applications running in a smart phone, finding the best combination of the applications become a NP kind of problem. This problem can be attempted using Soft Computing techniques.

Generally for NP-Complete set of problems, it is not possible to compute the exact solution in polynomial time. As a result Soft Computing techniques could find optimal solutions to these problems in a polynomial time. The major difference between Soft Computing and the conventional computing is that, unlike conventional computing, Soft Computing can tolerate the solutions with approximations, imprecision, uncertainty, and partial truth. Soft Computing Techniques try to model the human brain.

Now-a-days soft Computing techniques are becoming very popular in providing the optimal solution to the NP set of problems. Soft computing techniques are efficient in finding the optimal solution as they can limit the search space of the solution. They are being exhaustively used in the areas like Image Processing, Adhoc Network. Some of the popular Soft Computing techniques are

1. Genetic Algorithm
2. Neural Networks
3. Ant Colony Optimization
4. Particle Swarm Optimization

1.1.1 Genetic Algorithms

Genetic Algorithm (GA) is an optimization technique inspired by the biological evolution process. Aim of Genetic Algorithm is to discover one of the optimal, if not the best solution to an optimization problem. Genetic algorithms belong to a group of algorithms called evolutionary algorithms. These algorithms are used to find solutions to optimization problems. These algorithms are based on natural process of evolution and have similar steps such as inheritance, mutation, selection and crossover.

1.1.2 Artificial Neural Networks

Artificial Neural Networks in short **ANN** are the Soft computing technique which has been inspired from the human Nervous system. The computation models based on the **ANN** are capable of "**Machine Learning".** They can be used for designing "*if-then* rules" based decision system and pattern recognition systems. **ANN** is generally represented as system of interconnected neurons. The connections between various neurons have weights assigned to them. The output is influenced by the input along with the weights associated with the connections between various neurons.

1.1.3 Particle Swarm Optimization

Particle Swarm Optimization or PSO in short, is an optimization technique which stores global best solution. PSO simulates the social behavior of the fish school or the bird flock and methods by which they explore for finding food sources or other suitable habitats. The algorithm maintains potential of the population and each particle in the population represent the potential solution to a problem. During iteration the prospective solution is evaluated and its fitness is determined, maxima and minima of such solutions are determined.

1.1.4 Ant Colony Optimization

Ant Colony Optimization or ACO is a probabilistic technique for solving all those problems, which can be equated to finding a good path in the network. This algorithm aims to search for the shortest path in a connected graph. It is developed after observing the behavior of the ants searching for food. In real words, the ants wander randomly looking for the food. ACO as algorithm mimics this behavior. The real life ANTs are replaced by the simulated ants, which walk around the graph representing the problem under consideration. ACO is mostly used to find the solution to the NP complete set of problems. One example of such a problem is Travelling Salesman Problem.

1.2 Related Work

[11] Talks about use of the genetic algorithm for scheduling applications in large scale data centers. The aim of developing application scheduling method is to **reduce the energy consumption** in a large-scale data center. In this paper the large-scale data center is simulated and then scheduling algorithm based on the genetic algorithm is applied. In the paper, author aims to decrease the energy usage by scheduling the applications in such a way that the applications run on minimum number of nodes i.e. there are minimum number of transfer of applications between nodes.

A finite number of nodes and applications are considered. The resources such as CPU, Memory, I/O used by each application are also known. Gigabit Ethernet Switches are used to connect the various Computing Nodes in a data center.

The chromosome of the application is defined by its location vector. The object function is prepared using the Numbers of the Nodes used and amount of the data transferred. It has been found that total energy consumption in the large scale network is directly related to these two parameters. Single-point crossover, Two Point Crossover and Uniform Crossover were used.

[12] Proposes a genetic algorithm based solution to schedule various task or applications on a multi-core system. The scheduling is done by giving due regard to the concerns related to the power and thermal safety.

High temperatures causes hot spots, this paper discuss about genetic algorithm based temperature aware scheduling on a multi-core system. Embedded applications often consist of multiple concurrent tasks, which can be scheduled on difference processors. One of the approach is to schedule is Power aware scheduling, but Power aware scheduling cannot guarantee thermal safety, resulting in temperature of some chip spots might exceed the highest allowable temperature. The objective is to minimize the overall schedule time while not exceeding the maximum core temperature allowed for safety reason.

The chromosome Encoding is with the Processor number, once such chromosome is shown in the figure 3-2. The crossover chosen is Two Point crossover, and the Fitness function is the minimum schedule time. To ensure that local hot spot are not created, each Chromosome is evaluated for thermal safety, penalty is incorporated in the fitness function calculation.

[13] Proposes a way to improve the user overall satisfaction in distributed computing, using genetic algorithms. Distributed system is required to simultaneously handle a large number of requests. Hence a Distributed System needs a scheduling algorithm which meets both system's and user's requirement. This paper provides a distributed task scheduling model based on Genetic Algorithm. The main objective of this model is to keep the sum of user satisfaction as minimum.

Each gene in a chromosome is represented by a task resource pair (J_{ij}, $r_{k)}$ where J_{ij} is ID of a task and r_k is ID of a computing resource. The initial population is generated randomly.

The fitness value of a chromosome is comprised as satisfaction value and is also called user overall satisfaction and is shown by the following equation.

$$\text{Fitness} = \sum (\text{Satisfaction}_i)$$

Where Satisfaction_i represents the satisfaction of a particular user. The satisfaction of a particular user is calculated as follows

$\text{satisfaction} = \alpha * \text{Time}_{cons} / \text{Time}_i + \beta * \text{Cost}_{budget} / \text{Cost}_i$

In the before mentioned equation $\alpha + \beta = 1$;

Time_{cons} is the expected computing time and Cost_{budget} is the const constraints. α and β can be adjusted as per user's requirement. Single point crossover was used for reproduction.

1.3 Motivation

Now days, Mobile computing is becoming very popular. Increasing number of people are using mobile devices especially mobile phones to accomplish various computing tasks. Mobile technology especially the mobile devices have undergone a quantum of progress in last few years but the battery technology has failed to keep the pace with the progress in electronics used for designing mobile devices. Currently short Battery life is a major problem area faced by mobile device system engineers. To counter this problem the software engineers and system designers in the mobile space use many techniques to enhance the battery life. One such activity is to kill the applications which has been lying dormant or are the least used applications. Another approach is that in condition of low battery, user is not allowed to use applications with high energy. All these approaches have the potential of deeply impact the user experience for such devices.

One way to optimize energy usage is by using the intelligent application and Context-awareness in the applications can be used to make them smart. The Context-awareness will enable application to modify its behavior or structure so to be able use the available resource optimally. The context aware application is that application which is aware of the user requirement and can decide on which applications should be killed without degrading the user experience. One such example will be that suppose that user want to watch a video in a low battery condition so the application will kill all other applications other than video, so that user can continue watching video application even in the low battery condition which would not be allowed in normal condition. Most of the existing context aware frameworks have a big constraint in terms of energy requirements. They themselves have heavy resource requirements for deployment in term of battery life or computing ability, which severely limits the possibility of their deployment in the field. Our research as explained in the forgoing section establishes that use of Genetic Algorithm is a proper solution to reduce energy consumption or resource optimization. The purpose of this application is to optimize energy usage. We propose to change the way the low battery condition is currently applied in the existing Smart Phones.

1.4 Problem Statement

The battery consumption in a phone is directly related to the current being consumed at any instance of application. To enhance the battery life one needs to minimize the total current being consumed in the phone. Since there is no limit number of applications running in a smart phone, and if the number of application is too high, then finding the best combination of application to reduce the energy consumption becomes a NP kind of problem. Generally the applications running on smart phone can be broadly categorized in the following types of classes.

- Multimedia - All the video and music applications come under this class. Under this class watching Video is the most energy consuming application. All the Video and Music player will belong to this class

- Data Transfer - All the application which use packet based data service belong to this class. The physical bearer can be Wi-Fi or the RF. Some of the examples of the application in this class can be
    - Wi-Fi Direct.
    - tTorrent.
    - Email.

- Gaming - Android as a platform is very popular among gamers. Lots of the gaming applications have been developed and continue to get developed. In fact almost all the popular application on android "**playstore**" are some or other game only. Gaming is again a high energy consuming application. Some of the popular application in this section are
    - Temple Run
    - Subway Surf

- Communication Service - Communication Services are the basic service provided by the phone. Voice Calling and SMS are part of this service only. If the RF part is ignored this services do not consume much battery.

The objective of this thesis is to propose a soft computing approach to the designing of application that efficiently manages the applications running in a Smartphone with the aim of enhancing battery life of a smart phone. The aim is to enhance the user experience and Smartphone utility under the present constraint of the short battery life.

This thesis focuses on adapting the Genetic Algorithm to manage running of different applications in smart phone to enhance its battery life. The Genetic Algorithm is an established

Soft computing technique to solve optimization problem. This thesis also aims to optimize battery consumption giving equal regard to the user experience, that is the user should be allowed to run the application he or she desires. The problem statement for this thesis is

Develop a Genetic Algorithm based application that determines the optimal combination of the application to run on smart phone device in order to minimize battery usage and enhance the battery life.

1.5 Scope of Work

This thesis aims to optimize the battery consumption without affecting the user experience. To achieve the before mentioned aim the problem can be further divided in the following subparts. To conserve the energy one must be able to quantify the energy being used, so the first step towards solving the problem will be to estimate the Energy consumption in smart phone at any instance. Once the energy consumption has been quantified, the second step will be to choose a Soft Computing based intelligent technique to optimize the Battery consumption, and last but not the least step will be to verifying and Validate the Implementation on real devices.

1. **Energy Consumption Estimation**

In a smart phone the Energy Consumption can be measured in various ways, voltage of the battery, battery percentage, and current consumption by various applications running in the smart phone. This thesis will be focusing on the current consumption at any instance to quantify the energy consumption in the phone. The power consumption calculated from various hardware modules like CPU, Wi-Fi, Display (Screen), GPS, Audio in a Smartphone and then distributes to individual application as per their uses. The current consumption for the individual hardware modules is provided by their drivers. The total current consumption per hardware module is averaged among various applications using that module at that particular time.

2. Select optimal combination of application to optimize battery consumption

This thesis shall develop the Genetic Algorithm (GA) based approach to find out one of the best combination of the applications which can be allowed to run and remaining applications can be terminated, thus resulting in the saving of the battery life.

3. Validating the Implementation

As part of the Validating the approach, we will design two applications. One application will adapt the Genetic Algorithm to Smart Phone environment and second application to log the energy consumption in the smart phone. We plan to log the remaining battery percentage to log this information. We plan to test the hypothesis on a number of actual devices and observe the effectiveness of Genetic Algorithm on real devices.

1.6 Thesis Organization

The remaining sections of the thesis are organized as follows:

Chapter 2 provides a detailed description about various researches done for this thesis. This chapter contains the work done by other people to solve the similar problem in same or different area.

Chapter 3 presents the approach taken to solve the problem in detail.

Chapter 4 describes the implementation part of this research work. It describes the design on the implementation.

Chapter 5 shows the results obtained when the idea described in the thesis is implemented on the real Smart phones.

Chapter 6 concludes the thesis and presents the possible improvements in this research work in future.

Chapter 2 Literature Survey

I referred to a large number of papers published in journals like IEEE, ACM etc to study the work done so far and the summary of the some of the relevant papers is reproduced in this section. My study was focused mainly on two areas.

1. Application of Genetic Algorithms to various Optimization problems.
2. Study of the energy saving framework

2.1 Energy Saving Framework

One of the ways to conserve energy is to change the way applications behave when there are constraints on the total energy available. Most of the energy saving framework discussed in the following section use context aware information. Some of the papers discussed about Adaptation in the application to conserve energy. If the available energy is less, then the application will adapt itself to conserve the energy.

2.1.1 Adaptations in Applications

[1] Discusses about the framework to collect the various information related to context. It also discusses about the adaptations which can happen in the framework. The framework describes design of an application middle ware which can adapt itself to reduce the consumption of its own resource to a minimum. This paper describes two kinds of adaptations which can happen in the applications.

1. **Structural adaptation** – This is the kind of adaptation in which application will change its structure or composition depending on the context.
2. **Behavioral adaptation** – Behavioral self-adaptation is the adaptation which results in modification in applications behavior. Generally this will triggered by changes happening in the application context. Some examples:
    - Media player increases the volume automatically, when the listener's Favorite song starts to play.

• If the user is in a meeting, phone will sense the change in the context and will Automatically change the user's online status, to busy.

The framework proposed in the paper consists of three layers,

Layer 1 Application Layer

Applications are thought of as composition of components. These components can be either mandatory, or optional. The components are interconnected and can communicate with each other by sending asynchronous messages to each other. It is possible to dynamically add or remove a component dynamically, depending on the execution context.

Adaption in Application layer.

a. Behavioral Self Adaptation – Dynamically changing the application's behavior, like reducing the brightness automatically when available battery is low.
b. Structural Self Adaptation - Changes the composition, the components may be brought in or removed depending on the application needs.

Layer 2 The Context Layer This layer is responsible for the following

a) **Input** – Gathers the data from sensors and encapsulates the collected data.
b) **Filters** - Filter out the irrelevant or old data.
c) **Persistency** – Acts as a repository that stores the context value and the relation between various contexts.
d) **Transformation** – Changes the representation of a concept to another form e.g. deriving the city name from the GPS position.
e) **Reasoning** – Derives new information by combining known context and derivation rules.

Examples of Adaptation in the Context Layer

1. Behavioral

If the repository has a limited storage capacity, then the old data can be purged to free up the memory, another way can be to sense and retrieve the context at a slower rate so as to reduce the rate at which the information is being collected.

2. Structural

If a change in the application's context, forces that some components are no longer used during processing. Then all the components processing the information related to that context can be removed from the processing chain.

Layer 3 RunTime Layer The basic reason for this layer to exist is to provide the basic functionality of displaying and running the component compositions. This layer also acts as a hardware abstraction layer. It provides the following functions.

a) **Core Component management** – Deploys the components and also takes care of the inter communication between the various components.
b) **Self Evaluation and Adaptation control** - Evaluates and adapts to resource usage and controls on demand activation of the computational resources.
c) **Resource Access** – This module has the responsibility of providing the s Uniform access to platform specific I/O features like backlight, audio etc.
d) **Resource Monitors** – This particular Layer is responsible for keeping the track of the resource usage. The resources which are monitored by this layer can be memory usage, network bandwidth, battery status, CPU load, disk space allocation, etc.

Example of Adaptation in the RunTime layer

Behavioral

One way to reduce the energy usage is to reduce the CPU Clock frequency, when the system is not being used or in case the phone like device, if it is idle than energy can be also saved by decreasing the display brightness.

Structural

One example of Structural adaptation can be to Enable or Disable the hardware resources and I/O devices as per demands of the applications.

2.1.2 Context Aware Mobile Applications

[2] Describes the "**CAPIM**". CAPIM is the platform designed, with the aim of developing the context aware mobile applications. CAPIM stands for "Context Aware Platform Using Integrated Mobile Services". It consists of the following modules.

1. Container - Is the execution framework on which acts as base for all the layers. It provides the service for the discovery of availably monitoring services. Data is collected through monitoring services. Application context is then prepared from this collected data. Each monitoring service is executed as different process.

2. Second Layer – The basic functionality of this layer is to perform the aggregation, followed by the storing of the context data. A device having the server environment is generally required to run this layer. This layer supplements the Smart Phone computation capability without interfering with the Smart Phone user's activity.

3. The third layer is the Rule Action: - The rules which are stored in the remote repository are expressed in XML based format. These rules are dynamically loaded by the applications, who execute them on the local device.

This paper further describes following context models.

Generic Context Model - External data and internal information are included by this context model. This information is aggregated by this context model into a unique set of data. All the relevant and detectable attributes of a mobile wireless device are used to build the context of the application user and device.

Hierarchical Context Model – This context model is composed of various parts. The first layer is the device. The information which has been collected from various sensors is grouped together with the information like location, time and user's identity.

Context Acquisition Model - Remote repository stores the context model monitoring modules as packages, these packages can be downloaded from the remote repository. Discovery and loading of modules is the responsibility of the module loader. All the monitoring services are discovered dynamically, downloaded and executed when needed.

Semantic Based Model - The data that has been sensed and collected is further aggregated using a semantic model. The information is stored using context ontology. Server executes the aggregation and semantic services. The aggregation services receives the context information from the mobile devices, and manages it to keep it semantically organized. The aggregation module uses XML format to pack the received information.

Rule Engine: Rules are prepared from the context information. The lists of parameters obtained from the context information are used for preparing the conditions inside the rule configuration file. The engine will periodically evaluate the rules. The Rule is composed of conditions, Actions, and Action Parameters. New rules can be prepared by combining various Rules. Boolean operators can be used to combine various rules. These rules are used by the context aware applications to react and take smart decisions automatically as and when the context changes.

2.1.3 Energy Aware Application

[3] Describes the dynamically modification in the applications to conserve energy. The applications can continuously monitor the available energy supply and its demand, and is able to decide the correct balance between the quality of service desired by the user and energy conservation. This paper also describes the support which an application requires from the Operating system to make battery last for the time as desired by the user.

The Operating systems guides such an adaptation by continuously monitoring the energy supply and demand. When there is no constraint on the energy availability than the application can give a preference to the good user experience and where there are severe constraint on how much energy is available the application can take steps to conserve the energy.

The author describes the uses of tools such as PowerScope. This tool allows him to measure the energy consumption of the individual software components. PowerScope is a tool which can be used for associating the energy consumption to the specific software components. It derives its design and functionality from the existing and popular CPU profilers such as prof and gprof. These profilers are commonly used in Linux based system to profile the usage of processor cycles by various code components. This way PowerScope can help in exposing the system components, which are most responsible for energy consumption. Computer System's energy usage is profile by the Power Scope using statistical sampling. One such example of adaptation is that suppose a user is watching a color movie video. The video is being fetched from a server in real time. At a particular time network experiences delays or constraints in the bandwidth, than the device can switch to black and white video so as to conserve the bandwidth rather than user experiencing the delay or pauses in the video due to lost frames. Similarly another application is a GPS based map application. This application is used to fetch maps from some server in real time. If at a particular instance, the time to fetch the high quality map is very high, then the application can chose to fetch the not so detailed map, rather than keeping the user waiting for the best quality maps. The Fidelity is the attribute used to capture the data degradation. Energy aware adaptation can be summarized as first monitoring of the available resource levels, followed by adjustment of the fidelity by application to match the changes in the resource levels.

To check the effect of Fidelity on the energy consumption, the energy consumption of the applications was monitored. The applications monitored were the commonly used applications like video Player, map viewer, web browser and speech recognizer. Four different types of data were used for each application, like for different video clips for the video player, four maps of different quality for the map viewer and so on. The energy usage in the data with the highest quality or Fidelity was compared that in the data with lower fidelity. It was observed that Fidelity can affect or result in a reduction of 34 percentages in the energy usage.

<u>2.1.4 Content Adaptation</u>

[4] Introduces the concept of the static and dynamic adaptation. Now a day the people use a variety of the heterogeneous devices to access the World Wide Web and World Wide Web now includes the rich content applications such as video streaming, 3D games, Video conferencing and mobile TV. However most of these devices capabilities differ in terms of built in software and library, display size and battery supply. Content adaptation is required for the digital contents to fit the target device. This paper reviews some of the representative content adaption solutions.

With the advent of the availability of the rich content on the World Wide Web, it has become a very important challenge in the mobile computing, to efficiently support multimedia applications, given that these mobile devices have limited battery resources. One of the foremost requirements for the mobility is that size of the computing device should be as small as possible and weight should be as light as possible. In the modern computing devices the sizes of all other components have reduced considerably and now battery is what contributes most of the size and weight of the mobile devices. Battery life is also directly proportional to the Battery size, so it is almost impossible to increase the battery life without increasing its weight or size.

It is proven that content adaptation mechanisms have the capability to prolong battery life of multimedia streaming clients. Generally the following two types of the content adaptation can be considered.
<u>Static Adaptation</u> and **<u>Dynamic Adaptation</u>**.

<u>Static Adaptation</u> In static adaptation, different versions of the same content are prepared, and based on the user and user’s device capability the appropriate version is supplied to the user.

In contrast **<u>Dynamic Adaptation</u>** enables the adaptation of the content to happen in real time, once the server comes to know of the user mobile device capability, the adapted version for the specific context will be composed during user request. The main advantage of the Dynamic Adaptation is that it can be used to provide the most suitable adapted version.

Scenario of Content Adaptation

The three phases of Energy aware content adaptation the phases are

Collects,

Processes, and

Composes.

The user can request to view a particular content via the client devices, the content adaptation engine can collect the user preference and client device profile proactively. This can be done from the user as well as the content and Meta data from the content servers. The engine processes the collected data and produces an energy profile for the user. Energy Management Configuration can be produced using this prolife along with the user and device profile. Adapted content will be produced by applying the adaptation configuration to the content. Finally it will be transmitted and presented to the user.

The following fundamental areas in content adaptation that need to be well understood before any content adaptation system can be designed.

a) **Locality** - Where to perform the content adaptation.
b) **Strategy** - Who should perform the adaptation?
c) **Mechanism** - What should be adapted?
d) **Purpose** - Why perform adaptation.
e) **Context** - Adapt to what
f) **Method** - How to Adapt

The main concept in the Energy Aware Content Adaptation is to perform adaptation in the data or the content, before it is distributed to the user. Content can be adapted in several approaches such as

Appearance Adaptation,

Size Adaptation,

Format Adaptation,

Characteristics Adaptation and
Encapsulation Adaptation.

Appearance Adaptation: - This approach adapts or changes the content, so as to enable it to fit the device dimension.
Size Adaptation: - The content or the media is resized, it also minimizes bandwidth, processing and energy consumption during presentation
Format Adaptation: - The encoding technique and processing requirement, some content media formats consume more energy than other, this adaptation involves changing the media format to a more energy efficient format e.g. Bitmap to JPEG.
Characteristics Adaptation: - Fidelity of the contents also plays an important role in content adaptation. This approach adapts original content by varying the fidelity of the contents.
Encapsulation Adaptation: - Content usually come with various information. This approach extracts the most important aspect of the content and encapsulates the less important information. The energy is saved as the most important aspect is processed and presented.

Strategies :- Adaptation strategy is concerned with, who or which part should be responsible to carry out the energy aware content adaptation.

(a) **System –** In this strategy adaptation is performed by the underlying system. Requested content is downloaded by the mobile operating system, following which adaptation is carried out on the content before it is delivered to the user via mobile device.
(b) **Application** - In this strategy, the adaptation of the content is carried out by the application. One example of such a strategy is the browser adapts the web page content as per the target device.
(c) **System and Application** - In this strategy, both the system and application are responsible for carrying out the adaptation.

Localities: - refers to the location where the content adaptation operation takes place .It can be centralized or distributed. Centralized is further divided into Client Side, Server Side, and Proxy Side.

a. **Client Side** In client side approach the client itself needs to perform the content adaptation and then send the adapted version to the user's screen. This is suitable for the static adaptation.
b. **Server Side** Content adaptation is performed at the server where the original content resides.
c. **Proxy Side** The proxy is responsible for the adaptation in this approach.

Distributed: - In a distributed the Adaptation can also be done at different locations. This is achieved by breaking the adaptation into several tasks and passing the responsibility of completing each task to different locations. One example for such a scenario can be, Trans-coding of image data can be done at the server while adapting the web page based dimension can be done at client side with scarce resources.

Purposes : Every content adaptation is developed with the aim satisfying a particular purpose.

General Purpose: - Usually used to adapt the content properties, objects and some other common characteristics.

Content Specific: The content specific system is designed to handle specific task of adaptation in accordance to data type. For example is system goal can be to adapt image as it will only handle image adaptation.

Context : Refers to who should be considered for the content adaptation. In fact the content adaptation should be carried out based on priority. Content category can be divided into device **centric** and **user centric**.

Device Centric: - The adaptation process in this approach is mostly based on client device capabilities. The adapted content tailored to the specific profile includes energy states of the identified client device.

User Centric: - User preferences, its surroundings and users inferred interests are the main considerations in the adaptation process. These preferences are considered as high priority in dealing with the user centric approach.

Methods : Methods refer to the various techniques used to adapt media content, Some of the most used methods for content adaptation are adaptive streaming, distillation, content selection, scalable coding and color adaptation.

2.1.5 Context Awareness

[5] Describes the context and an intelligent framework. Applications and mobile services can be composed dynamically using this framework.

The context can be defined as the "*combination of information relevant to user's nearest environment*". Contextual information is generally used to prepare the user profile. In addition to the contextual information, the user profile may also have other details like user preferences, the terminal capabilities, network type and requested or required services. The profiles can be described using **XML**.

The features or the services which are required or wanted by the User are expressed in a user preference profile. The preferences of a particular user may be independent of the service being offered or may apply to all the services which are being used by the user or may apply to only the services offered by a particular application. The preferences of a User can have a wide range of attributes. Some of these attributes are

1. Perceived QOS requirements.
2. Preferred Language
3. Content and media presentation characteristics
4. Size of the Font
5. Tariff or billing
6. Privacy and Security
7. Favorite Geographical Zones
8. User History and Calendar

A **context aware service** is the service which can discover the current context of the user and following which it will change its behavior to satisfy the user's context. The services which are responsible for the context should continue to collect the information from the device sensors.
Service can be defined as information technology products, which are used directly by the end user. The Functionality and Content of the Service determines it value rather than the transport or network used by the Service. Services typically will generally run on the user device and may have a part which will interact with the server.

The following figure shows the framework which composes the mobile services considering consumers current context, and dynamically adapts service provision accordingly. When an authorized user requests the service for access, the service logic collects the information about the updated context, from the context-sensor components. This information is taken in the form of the profiles. The service requirement for the current context is generated by using the updated context information and the service profile requested by the user.

Next the component responsible for matching the profiles propagates the generated information to the "*Service Execution Engine*". The main responsibility of the "*Service Execution Engine*" is to compile and execute the commands which are mentioned in the incoming profile. This task may be performed as a series of several steps. Example of such steps is as follows.

1. The multimedia data or content stored in the remote content repositories is retrieved in the same format.
2. Interaction is done with the content adaptation Service with the purpose of performing the Trans-Coding or Translation of some of the media files.
3. The Media files are composed as per the indicated presentation format.
4. The resulting files and packaged and forwarded to the user, who had requested for them.

2.1.6 Sample ContextPhone architecture

[6] This paper describes architecture of *ContextPhone.*"Context Phone" helps the developer to create context aware applications. Such applications combine the existing technologies with user's everyday life. The human interaction was central to the design used to develop the ContextPhone. It is software platform which is composed of four modules. These modules are interconnected. These modules have been developed as a set of libraries coded in C++ . All the modules are part of the Open Source. ContextPhone can be executed on the Symbian OS based mobile phones.

The ContextPhone platform consists of following 4 modules:

• **Sensors** acquire context data from different sources, such as location (Cell ID and GPS) or phone use.

• **Communications** connect to external services via standard Internet protocols using General Packet Radio Service (GPRS), Bluetooth transfers, Short Message Service (SMS), and

Multimedia Messaging Service (MMS). The communication channels can, for example, share location information or obtain sensor data (using GPS over Bluetooth).

• **Customizable applications**—such as ContextLogger, ContextContacts, and ContextMedia these applications can be used to supplement or completely replace the native applications like contacts or call log in the device.

• **System services** This service is responsible for launching the background services, error logging and recovery, and the Status display.

2.1.7 Context Aware Multimedia Applications

[7] Focuses on mobile applications which are based on rich interactive multimedia content, which can be triggered by context information collected from the users, their environment, and a wealth of context-enabled content. This paper gives example of how the user experience can be enhanced by using the context information available in the multimedia contents. These types of applications are called mediascapes. A mediascape application is capable of playing multimedia content (image, audio, or video) on a mobile device in response to changes in the user context. Users may use any kind of sensors such as location Sensors, Radio Frequency Identification tags, and biometric sensors like heart rate monitor to collect the context information and trigger a mediascape application.

A mediascape application is a context aware application, which uses rich multimedia content and generally runs on a mobile device. It can be social or personal as well. The multimedia content is generally comprised of images, video, audio, and flash interactions. The logic that uses the physical situation or in other words person's context has the capability to add great value to the user media experience compared to the other rich media experience, generally available on mobile device.

Example: - Suppose an individual moves into a particular location, then the change in the user's location affect how the media content is played. The logic associated with that particular location may depend on the number of times a person has moved in to that particular location. The first time a person enters that location, the device may play the media which gives a detailed description about the location but on each subsequent visit to the location may result in much shorter description about the location, which may include only the details added to the location after the user’s last visit.

2.1.8 Context Aware Framework

[9] Presents a uniform mobile terminal software framework. This framework describes the methods which can be used for acquiring the useful context information available in a user's environment. The Framework gives the same context information to the application after processing it. The various entities belonging to the framework use a blackboard-based approach communication between them.

The context framework is composed of the following four main functional entities:

Context Manager,
Resource Server,
Context Recognition Service, and
Application

The central node is responsible for storing the context information, which may have been collected from any source available to the terminal and is served to the client in the following three ways.

•Clients can directly query the manager (as a context database) to receive the context data.

•Clients can subscribe to notification services for changes in the context.

•Clients can use higher-level (composite) contexts transparently, where the context manager contacts the required recognition services.

Each context expression is composed of many attributes. The minimum required are **Context type** and **Context value.** Optionally it may contain other attributes described below:

• **Context type** refers to the context category.

•**Context value** refers to the semantic or absolute "value" of context type and is usually used together with context type

•**Confidence**, an optional property, which is used to describe the un-certainty present in the context value. This value is typically stored as a probability or a fuzzy membership of the context.

• **Source** describes the source from which the context information has been collected. A client which may be only interested in contexts from a specific source can use this property.

• **Timestamp** Tells about the time as which the last changes occurred in the context.

• **Attributes** specify the context expression freely and might contain any additional details not included in the other properties.

2.1.9 Context Aware Middleware

[10] Describes propose an "*Acquisitional Context Engine*" (**ACE**). ACE is a middleware that supports the execution of the applications which are context-aware. Application can use the interface provided by ACE to collect the user's current context. In addition ACE has the intelligence to dynamically learn the relationship among various attributes of the user context. Example of such a relationship is that when user is driving, than he is not At Home. These automatically learned relationships are exploited for two powerful optimizations. First is the ACE can make the inferences about one context attribute (AtHome) from the value of another attribute (IsDriving), without having any need to collect any new information from any other sensor.

The second optimization is the "Speculative Sensing". This results in energy conservation as the value of an expensive (expensive from the point of view of computation required, or energy consumption) attribute (e.g. AtHome) can be sensed from value of a relatively cheaper attribute (e.g. IsDriving). The running of context aware applications, results in very high energy consumption. High energy consumption happens, as majority of these applications need to do continuous monitoring of the user's context. If care is not taken then, this cost of continuous monitoring of the context can be very high, especially in the cases where the application needs to monitor the computationally expensive attributes such as user location and god help us if multiple such applications are running in parallel.

ACE is the middleware which was designed with the purpose of supporting the context aware applications and at the same time reducing the energy consumption for collecting the various contexts. ACE collects the various context attribute of the user, and has an interface which can be used by the application to access these attributes. Applications can request for the information about these attributes from ACE. ACE has optimization inbuilt to save the energy which is required for collecting all this context information.

In experiments described in [10], it was seen that using ACE there was about 4.2 times reduction in the sensing energy cost. The experiment had three applications and context traces of 105 real users. The key observation that is exploited is that there are physical constraints which limit the human contexts, and hence, there is a high correlation between the values of various attributes of the user context. Thus, it is easy to deduce the value of an attribute, by observing the value of another seemingly unrelated attribute.

Consider an example, suppose a user named Peter is using ACE to run multiple applications. There is an application say App1 which is using the accelerometer to observe the transportation mode of the user. Transportation mode is whether Peter is Walking, Running or Driving. At the time there is another Application say App2, which is using the location sensors like GPS to observe whether Peter is "AtHome" or "InOffice". The ACE, by checking the history of the User's context, can discover various rules such as “When Driving is True, AtHome is False”.

2.1.10 Context Aware Browser

[8] Describes a context aware browser. This paper describes how applications like Browser can be made context aware and how the device usability for the user increases once this happens. Two scenarios described here to illustrate the point.

Scenario 1

Suppose the user gets up at 7:00 am. Now the user context has changed from sleeping to Awake. This change in context can trigger the downloading of content. This content may be news headlines, which user generally prefers to browse immediately after getting up in the morning. As the day passes, user is up and having his breakfast, which results in another context change and may trigger web browser to download his schedule for the day.

Scenario 2

The user is ready to leave for his work place; he enters his or her car to drive to work. CAB detects that the context has changed. Now the CAB can download the information about the traffic condition on the route which the user is about to take, or any other information which may interest user while he is driving to the work.

Chapter 3 Proposed Development Methodology

This chapter describes our proposed framework for energy conservation. The plan for implementation of various activities required to achieve the goal are explained in detail, in this chapter.

3.1 The Proposed Framework

The various phases required for optimal usage of Energy are described in the Figure 3-1. These consist of three main activities namely, first the User Context is fixed or the user is allowed to choose the application which he or she is interested in using on his mobile device. Energy Calculation is followed by cleansing of the mobile device memory space from undesired applications. The Genetic Algorithm is used to decide the applications which should be terminated. The output of this process is to find the applications which are consuming maximum energy. Once this is done, next step for optimizing the energy usage can be achieved.

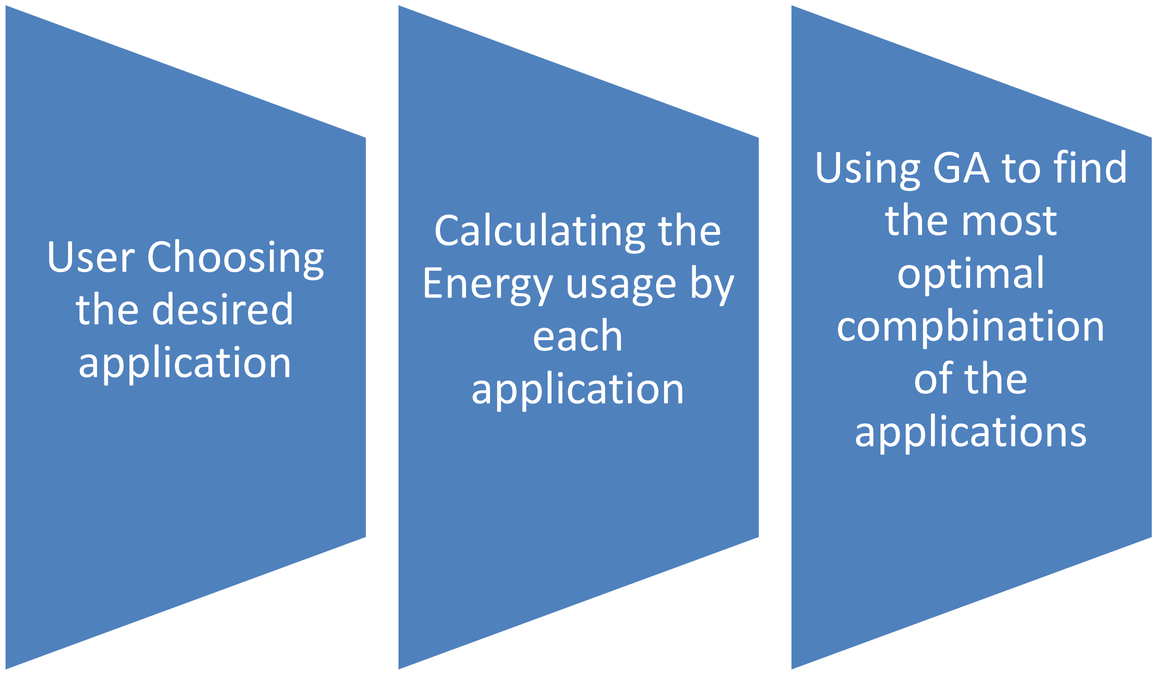

User Choosing the desired application
Calculating the Energy usage by each application
Using GA to find the most optimal compbination of the applications

Figure 3-1 The Framework for conserving the Energy

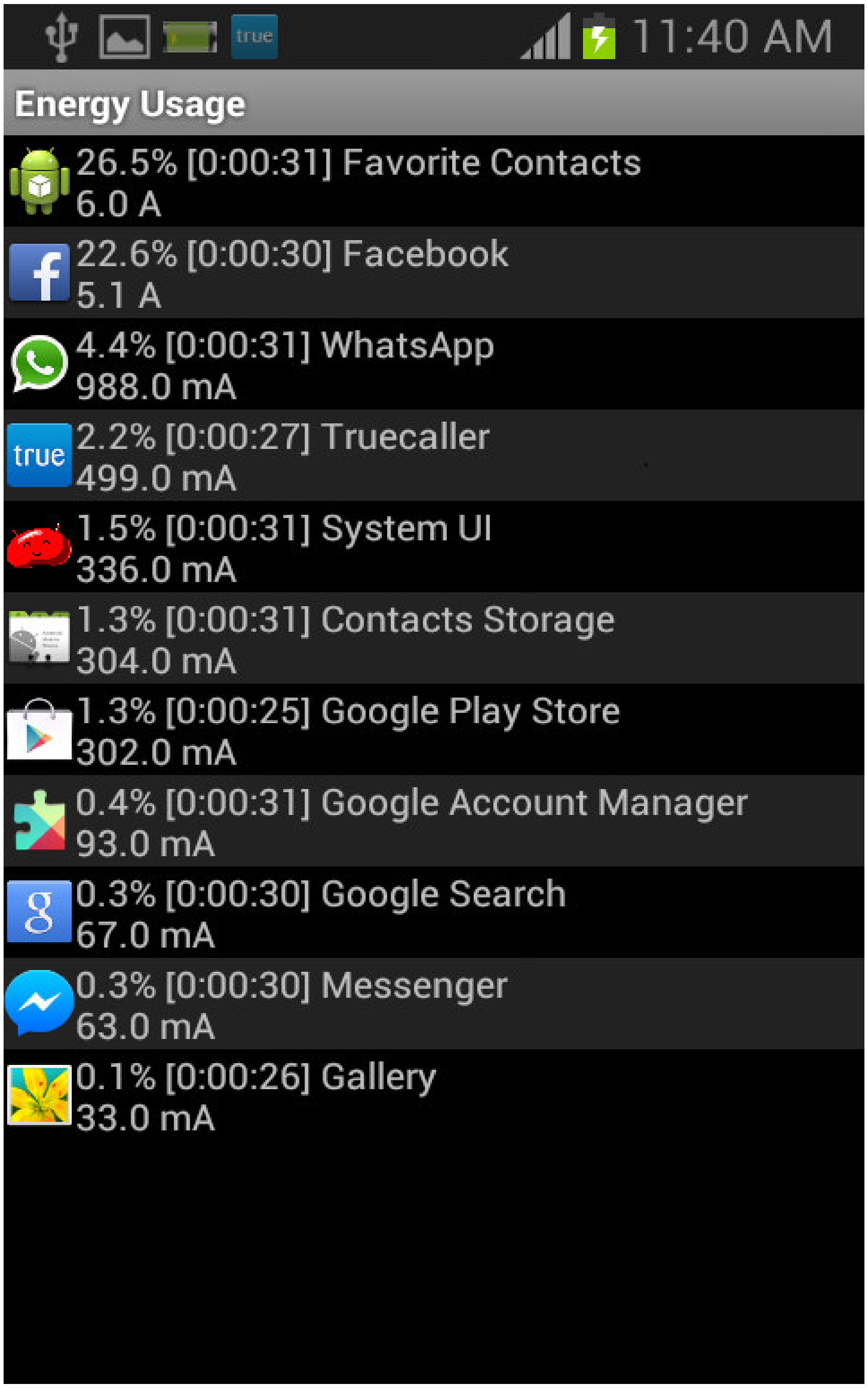
3.2 Framework to Estimate the Energy consumption of Applications

To conserve the energy, it is necessary to quantify the energy. The energy consumed at any instance is directly proportional to the current being consumed at any instance. The current being consumed by any application at any instance, can be used to represent the energy consumed by it at that instance. To estimate the current being consumed by any application, the total current consumption is calculated from below hardware modules in a Smartphone and then distributes to individual application as per their uses. The current consumption for the individual hardware modules is provided by their drivers. The total current consumption per hardware module is averaged among various applications using that module at that particular time.

1. **CPU:** CPU utilization and frequency level in a particular time.
2. **Display:** For Displays like LCD screen and OLED screen the application uses the screen brightness to calculate the current consumption.
3. **WIFI:** In case of Wi-Fi, It calculates current consumption for Uplink channel and data rate and also the rate at which packets are transmitted in a second.
4. **Network:** In case of network uses, it calculates current consumption for packets transmitted per second and power states.
5. **GPS:** In case of GPS, it calculates the energy consumption on the basis of power states of the GPS device (active, sleep, off states).
6. **Audio:** It also calculates the energy consumption by checking the power states of the Audio device (on, off).

When calculating the energy consumption, it is assumed that as if only that particular application is running. For example, consider a scenario where there are two applications running. Both the applications are using the Wi-Fi connection to transmit some data. The energy consumed by the Wi-Fi device is averaged over the two applications.

The figure 3-2 gives current consumption of some of the applications running in the Smart Phone.



Energy Usage

26.5% [0:00:31] Favorite Contacts
6.0 A

22.6% [0:00:30] Facebook
5.1 A

4.4% [0:00:31] WhatsApp
988.0 mA

2.2% [0:00:27] Truecaller
499.0 mA

1.5% [0:00:31] System UI
336.0 mA

1.3% [0:00:31] Contacts Storage
304.0 mA

1.3% [0:00:25] Google Play Store
302.0 mA

0.4% [0:00:31] Google Account Manager
93.0 mA

0.3% [0:00:30] Google Search
67.0 mA

0.3% [0:00:30] Messenger
63.0 mA

0.1% [0:00:26] Gallery
33.0 mA

Figure 3-2 the Current Consumption in Various Applications

3.3 Framework for Implementation of Genetic Algorithm

3.3.1 Genetic Algorithm

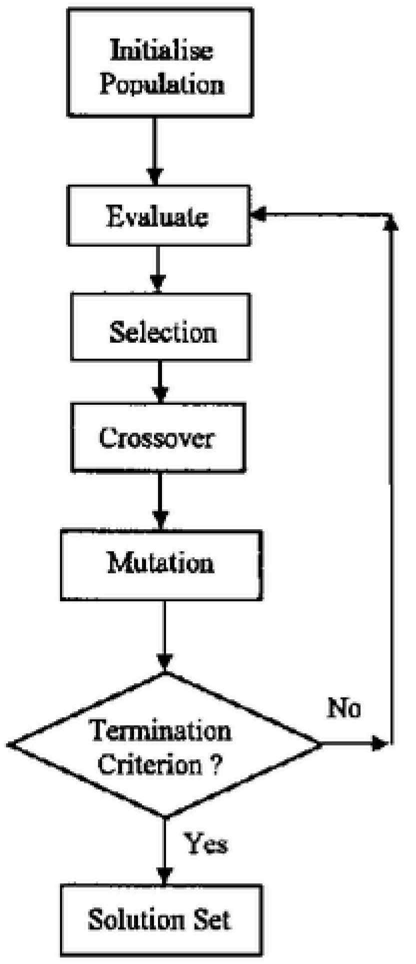


Figure 3-3 Flow for Genetic Algorithm

Genetic Algorithm (GA) is an optimization technique inspired by the biological evolution process. Genetic Algorithm is generally used to discover one of the optimal, if not the best solution to an optimization problem. Genetic algorithms belong to a group of algorithms called evolutionary algorithms. These algorithms are used to find solutions to optimization problems.

These algorithms are based on natural process of evolution and have similar steps such as inheritance, mutation, selection and crossover.

Steps in a Genetic Algorithm

The various steps of the Genetic Algorithms are shown in the Figure 3-2 and described in the following section.

(i) Initialization - The first step is to prepare an initial population of possible candidate solutions to the optimization problem under consideration. This population is called individuals, creatures, or phenotypes. The Selection Function selects that chromosome from the present generation which can become the base for preparation of the next generation. The selected chromosomes are referred to as Parents. The Parents are selected in pairs. There are various ways to select parents

a) Uniform or Random Selection.
b) Roulette Wheel Selection.
c) Tournament Selection.

(ii) Encoding - Encoding is the process of computationally suitable representation of the problem.

(iii) Fitness Function - The Fitness function is the base of the GA. The Fitness of a chromosome defines the way to calculate the quality of that chromosome. Or in other words higher the Fitness value, more optimal is the solution. The Fitness is the value returned by the Fitness function, which represent the merit of the solution.

(iv) Selection - The fitness of the each chromosome in the population is calculated, and best chromosomes among the existing population are selected and rests are discarded. The selected chromosomes are used to generate the new generation of the population.

(v) Crossover - Crossover is the way new population is generated from the existing population. Crossover mimics the biological mating. The Crossover can be defined as per requirement of the solution, the most common types of the Crossover used are

a. Single Point Crossover.
b. Two Point Crossover.

c. Uniform Crossover

(vi) Mutation - Mutation is used to give a shock to the algorithm incase the algorithm gets stuck in a local minima. Mutation can be used to change the genes in the chromosome arbitrarily so as to widen the solution search space to include other points of minima.

3.3.2 Genetic Algorithm Adaptation

The Genetic Algorithm based approach has been used to determine the processes or applications which should be terminated to optimize the battery performance. The context is simulated by allowing the User to choose an application which will not be terminated. The constraint is given this application is running the energy consumption should be kept minimum. Following Section describes the adaptation of the Genetic Algorithm to our specific problem.

Encoding

This framework uses the binary encoding is used. Length of the chromosome is same as the number of applications running in the phone. Each bit in the chromosome represents a particular application. Each chromosome is a series of '0' and '1'. If that chromosome is selected as the most optimal, then the applications, whose corresponding bit is 0, will be terminated and the applications whose corresponding bits are 1 will be allowed to execute.

Example

Suppose the following applications are running in the phone

[0] - System UI

[1] - Cricket

[2] - Newspapers of India

[3] - Context Service

[4] - Media Storage

[5] - TouchWiz Home

[6] - Music

[7] - Truecaller

[8] - WhatsApp

An example chromosome may look like

[111000000]

The first bit represents System UI, Second bit represents Cricket and so on.

If this chromosome represents the most optimized combination, then the first three applications i.e. System UI, Cricket, Newspapers of India will continue running and rest all will be terminated.

Selection

The initial population is generated randomly, the population size is customizable, and for our experiment we chose 8 as initial population size.

Fitness Function

The Fitness Function for a chromosome is the total current being consumed that particular chromosome. Lower is this value more fitter is the chromosome.

Example if the following were the current consumption for the following applications running in the phone.

Serial Number	Application Name	Current Consumption (mA)
1	System UI	0.7
2	Cricket	0.1
3	Newspapers of India	10.1
4	Context Service	0.2
5	Media Storage	1.2
6	TouchWiz Home	0.9
7	Music	25.9

8	Truecaller	0.1
9	WhatsApp	0.1

Table 3-1 Sample Fitness Values

Then fitness value of the sample chromosome [111000000]

will be calculated as

Fitness Value = 0.7 + 0.1 + 10.1 + 0 + 0 +0 + 0 + 0 + 0

= 10.9

<u>Crossover Types</u>

We experimented with three types of crossovers.

<u>1. Crossover Type 1</u>

If length of the chromosome is n bits than the first n/2 bits of each chromosome is interchanged.

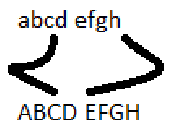
Example

Consider the following two chromosome of length 8.

Chromosome 1 abcdefgh

chromosome 2 ABCDEFGH

During crossover



After crossover

Chromosome 1' abcdEFGH

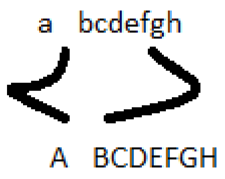
Chromosome 2' ABCDefgh

2. Crossover Type 2

In this crossover we generate a random number and only the bit corresponding to that random number is exchanged.

Example

Suppose we get 1 as the random number.



After Crossover

Chromosome 1' Abcdefgh

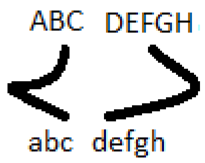
Chromosome 2' aBCDEFGH

3. Crossover Type 3

In this we find the point of the exchange by generating a random number.

Example

Suppose the random number generated come as 3



After Crossover

Chromosome 1' ABCdefgh

Chromosome 2' abcDEFGH

After experimenting with all the Crossover Types, it was seen that the Crossover Type1 gives the best result in most of the cases.

Termination

Each iteration of the application is terminated when the difference between the last iteration minimum current value and present iteration minimum current value decrease than 10 %. The chromosome with minimum current is chosen and applications are either kept running or terminated depending whether in the chromosome selected, that particular application is represented by 0 or 1.

3.3.3 Implementation of Genetic Algorithm

Step 1:- Initialize Population - Initial population of 8 chromosomes is generated. The Random selection method was used for the same. The Population size was limited to 8. Suppose the Chromosomes generated are C1, C2.....C8.

Step 2:- Evaluate - The fitness of solution is calculated. In our case Fitness is defined by the current being consumed for a chromosome. Suppose the fitness value obtained for each Chromosome is F1, F2....F8.

Step 3:- Selection - All the Fitness value are arranged in the ascending order and the four chromosomes with the minimum Fitness are selected. Suppose F1,F2,F3 and F4 are values representing the minimum fitness values, than C1,C2,C3 and C4 are selected.

Step 4 :- Crossover - Crossover is the step when the next generation is generated. The Chromosomes are paired with each other and they produce four more chromosomes according to the chosen crossover type. C1 and C2 are paired to generate C5 and C6. Similarly C3 and C4 are paired to generate C7 and C8.

Step 5:- Fitness of all the Chromosomes is calculated once again. This is quite similar to Step 2. Once the Fitness has been evaluated, the minimum Fitness value is compared with that obtained in last iteration. If the difference between two is less than 10 % than the algorithm is terminated otherwise the iteration continues from step 2.

Step 6:- If the difference between the current minimum Fitness value and the minimum Fitness value is less than 10 %, then the Chromosome with the minimum Fitness value will be chosen and the algorithm will terminate for this iteration.

Step 7:- Depending on the chosen Chromosome, the applications will be terminated as described in the earlier section.

3.3.4 Case Studies

This section describes the running of Genetic Algorithm using examples.

The initial applications assumed to be running are

1. System UI
2. Cricket
3. Newspapers of India
4. Context Service
5. Media Storage
6. TouchWiz Home
7. Music
8. Truecaller
9. WhatsApp

We refer to the Table 3-2 for current being consumed by each application.

The population size is 8, The Initial population is generated at random and looks like the following.

0. [111000000]

1. [011111110]

2. [000000001]

3. [111100000]

4. [000011100]

5. [111111111]

6. [111110001]

7. [111110000]

Fitness of each chromosome is calculated by adding the current being consumed by each application.

0. 10.9

1. 38.5

2. 39.2

3. 11.1

4. 28

5. 39.3

6. 12.4

7. 12.3

After iteration the 4 Chromosomes with best fitness value are chosen.

The chosen chromosomes are 0, 3, 6 and 7.

Now the crossover will be applied to following chromosomes.

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

1. <u>Crossover Type 1</u>

The population after first iteration will become chromosomes 0 and 1 mate with each other to produce chromosomes 4 and 5, chromosomes 2 and 3 mate with each other to produce chromosomes 6 and 7.

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

4. [111010001]

5. [111100000]

6. [111110000]

7. [111100000]

Since the chromosomes number 2 and 3 are duplicated, 2 chromosomes are generated at random.

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

4. [111010001]

5. [111100010]

6. [111110100]

7. [111100100]

Calculating the fitness again

0. 10.9

1. 12.4

2. 11.1

3. 12.3

4. 12.2

5. 11.2

6. 38.1

7. 13.3

The minimum Fitness value is 10.9, the difference between the minimum Fitness value and the Fitness value in the last iteration is 0, so the algorithm is terminated and Chromosome 0 is chosen as the most optimal one. Following which the following applications which will be terminated.

4. Context Service

5. Media Storage

6. TouchWiz Home

7. Music

8. Truecaller

9. WhatsApp

2. Crossover Type 2

The population after first iteration will become chromosomes 0 and 1 mate with each other to produce chromosomes 4 and 5, chromosomes 2 and 3 mate with each other to produce chromosomes 6 and 7.

Suppose the random number generated is 5.

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

4. [111010000]

5. [111100001]

6. [111110000]

7. [111100000]

After crossover the Chromosomes 2 and 7 are duplicated so the chromosome 7 is discarded and a new chromosome is generated.

So the next generation is

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

4. [111010000]

5. [111100001]

6. [111110000]

7. [101000000]

Calculating the Fitness value again.

0. 10.9

1. 11.1

2. 12.4

3. 12.3

4. 12.1

5. 12.5

6. 13.6

7. 10.8

The minimum Fitness value is 10.8, the difference between the minimum Fitness value and the Fitness value in the last iteration is 0.1, which is less than 10%, so the algorithm is terminated and Chromosome 7 is chosen as the most optimal one. Following which the following applications which will be terminated.

2. Cricket

4. Context Service

5. Media Storage

6. TouchWiz Home

7. Music

8. Truecaller

9. WhatsApp

3. <u>Crossover Type 3</u>

The population after first iteration will become chromosomes 0 and 1 mate with each other to produce chromosomes 4 and 5, chromosomes 2 and 3 mate with each other to produce chromosomes 6 and 7.

Suppose the random number generated is 5.

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

4. [111110000]

5. [111000001]

6. [111110000]

7. [111100000]

Since the Chromosomes number 2 and 3 are duplicated so two chromosomes are generated at random.

0. [111000000]

1. [111110001]

2. [111100000]

3. [111110000]

4. [111110010]

5. [111000001]

6. [111110100]

7. [011000000]

The fitness value for the current population is

0. 10.9

1. 12.4

2. 11.1

3. 12.3

4. 12.4

5. 11.0

6. 38.2

7. 10.2

The minimum Fitness value is 10.2, the difference between the minimum Fitness value and the Fitness value in the last iteration is 0.7, which is less than 10%, so the algorithm is terminated and Chromosome 7 is chosen as the most optimal one. Following which the following applications which will be terminated.

1. System UI

4. Context Service

5. Media Storage

6. TouchWiz Home

7. Music

8. Truecaller

9. WhatsApp

Chapter 4 Implementation

This chapter describes the implementation of the proposed methodology in detail. We describe the details of two applications "Smart Energy Saver" and "Battery Info".

4.1 Smart Energy Saver

Smart Energy Saver is the application developed to adapt the Genetic Algorithm to the Smart Phone environment. The Application is comprised of two parts, the first part measures and assigns the current being consumed to the applications running in the Smart Phone. The second part applies Genetic Algorithm to find the best combination of the applications which should be run for best battery life.

We used the following tools for developing this application.

1. Editor: Eclipse (Indigo Version)
2. Android SDK
3. Android ADT
4. Android Device
5. Android Emulator

4.1.1 Architecture

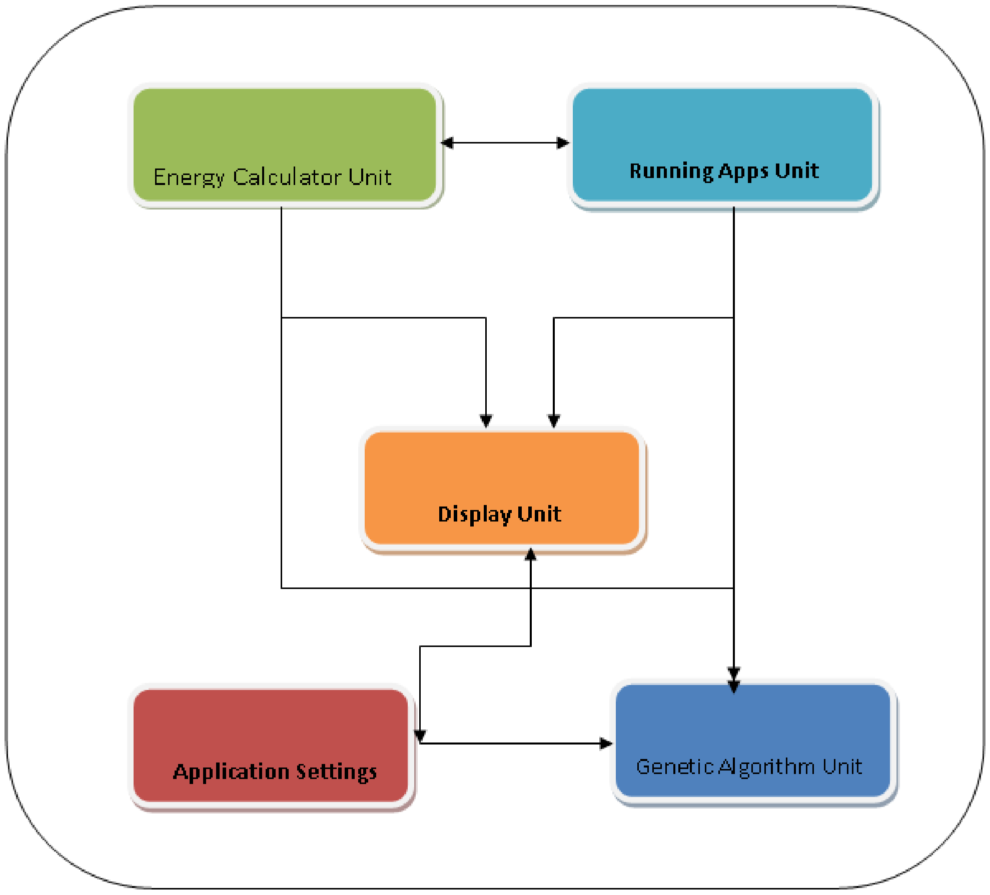


Figure 4-1 Smart Energy Saver Architecture

4.1.2 Energy calculator Unit

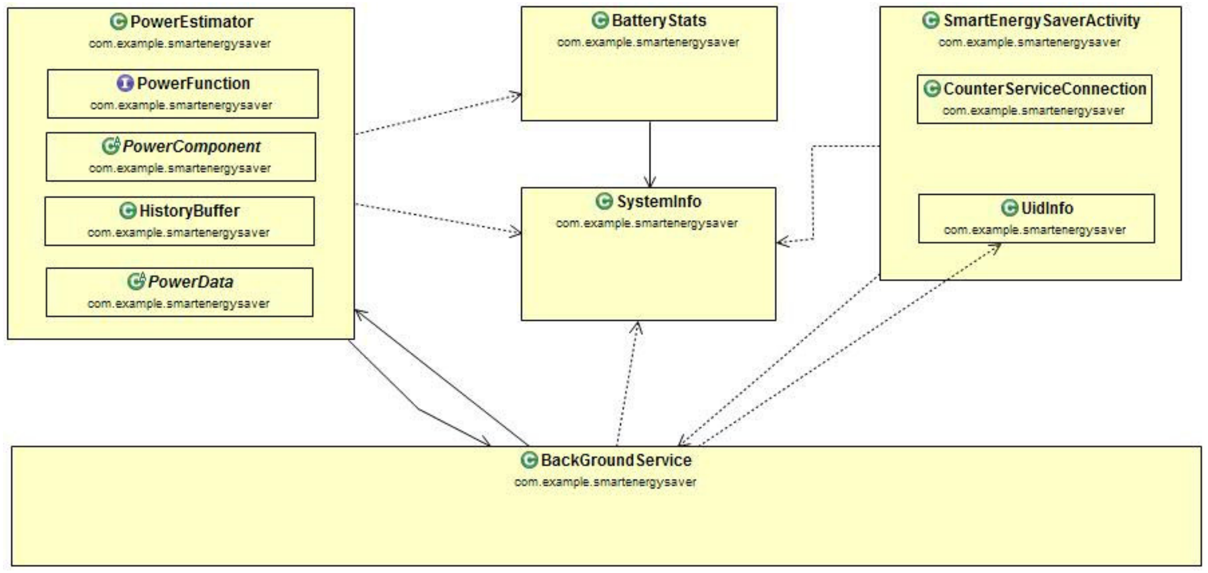

PowerEstimator
com.example.smartenergysaver
PowerFunction
com.example.smartenergysaver
PowerComponent
com.example.smartenergysaver
HistoryBuffer
com.example.smartenergysaver
PowerData
com.example.smartenergysaver
BatteryStats
com.example.smartenergysaver
SystemInfo
com.example.smartenergysaver
SmartEnergySaverActivity
com.example.smartenergysaver
CounterServiceConnection
com.example.smartenergysaver
UidInfo
com.example.smartenergysaver
BackGroundService
com.example.smartenergysaver

Figure 4-2 Class Diagram - Energy Calculator Unit

4.1.3 SmartEnergySaverActivity:

This the first GUI screen for the Smart Energy Saver application. It contains all menu options like settings, start/stop the service and different types of Genetic Algorithm options. This screen also contains the option for list of running applications and their energy consumption reports.

- OnCreate-This function is the entry point of SmartEnerySaver application. Inside this function, all the GUI components are initialized and start the background service.
- OnCreateOptionsMenu-This function populates all option menus for this application (Like Settings, Exclude App List, population Size, Interval, Cross over Type etc).

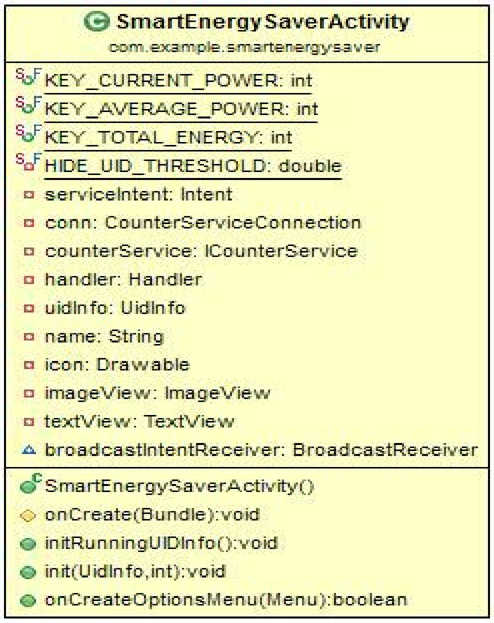


Figure 4-3 Class Diagram - SmartEnergySaverActivity

4.1.4 BackGroundService

This is the background thread which keeps track of all running applications and calculates their individual energy consumption for a particular point of time. This class starts the actual operation and maintains the session for this application.

- OnCreate-This function is an overridden function, which is invoked when SmartEnergySaverActivity starts the background service. It starts PowerEstimator module and register listeners for 3G/WIFI/LCD modules.
- OnStart-This function starts a new thread and all the energy calculation takes place inside this thread. This function also initializes all notifications on the notification bar.
- OnDestroy-This class automatically called when user stops or exits the application. It cleans up all consumed resources and free required memories back to OS. Also it stops all running threads and allows the application to exit normally.

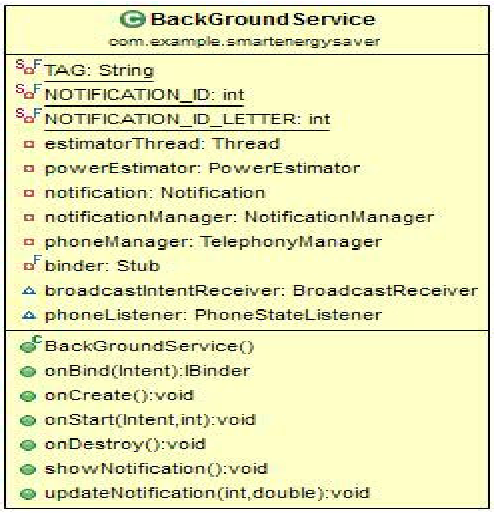
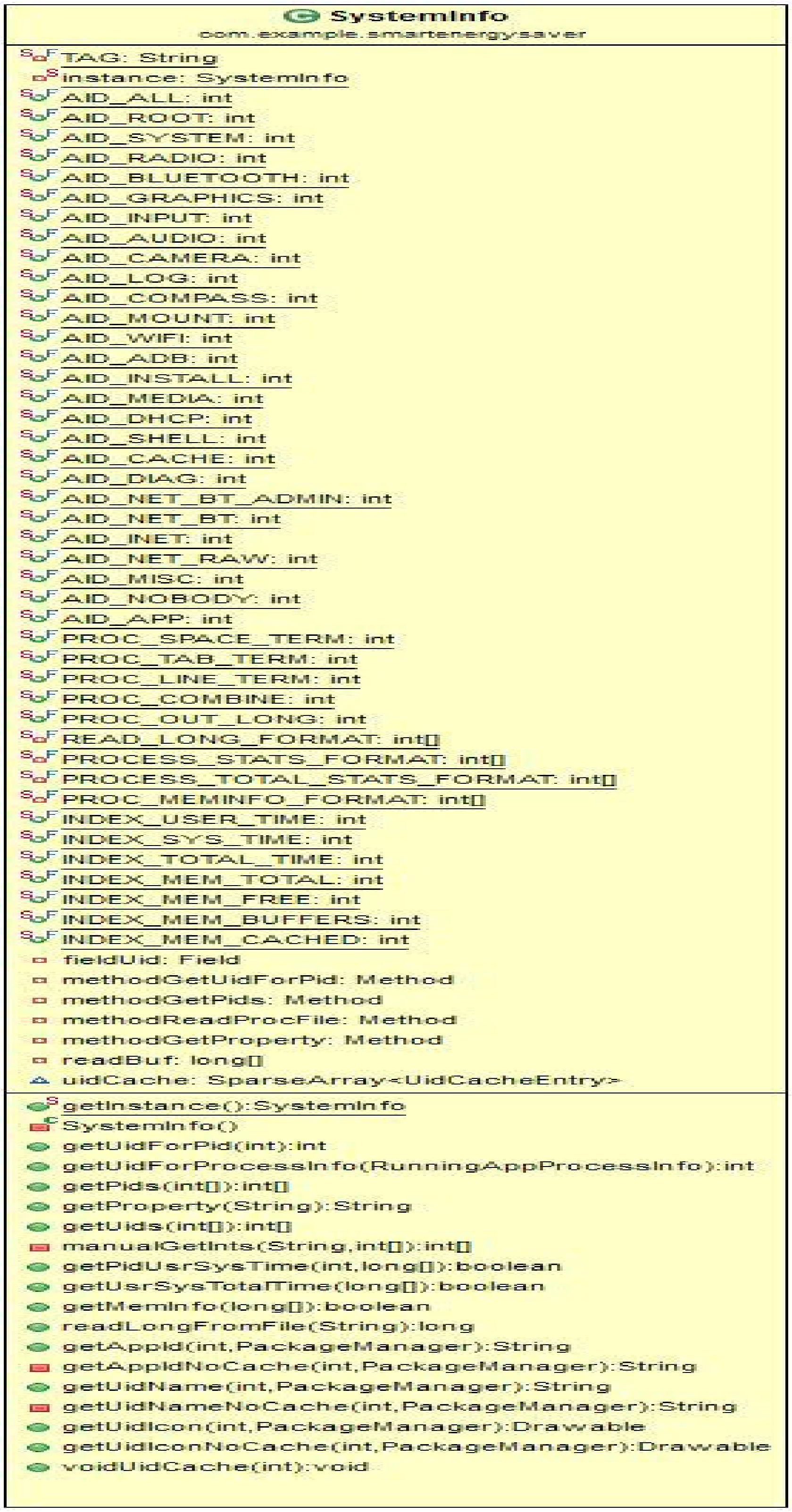

BackGround Service
com.example.smartenergysaver
TAG: String
NOTIFICATION_ID: int
NOTIFICATION_ID_LETTER: int
estimatorThread: Thread
powerEstimator: PowerEstimator
notification: Notification
notificationManager: NotificationManager
phoneManager: TelephonyManager
binder: Stub
broadcastIntentReceiver: BroadcastReceiver
phoneListener: PhoneStateListener
BackGroundService()
onBind(Intent):IBinder
onCreate():void
onStart(Intent,int):void
onDestroy():void
showNotification():void
updateNotification(int,double):void

Figure 4-4 Class Diagram - BackGroundService

4.1.5 SystemInfo

This class contains UID of different kind of services running in the Android device. It also keeps tracks on system level drivers and utility functions to get system related information.

- getInstance-This is a static function, which is called from other modules to get the running instance of this class. It returns pointer of the SystemInfo active object.
- getUidForProcessInfo - This function gets the uid field. If the field is not available it converts the pid of the running application to the uid.
- getAppId-This function returns the unique ID of the running application. It requests the package manager to get the ID, which is used for calculate current consumption for the application.



```
SystemInfo
com.example.smartenergysaver

TAG: String
instance: SystemInfo
AID_ALL: int
AID_ROOT: int
AID_SYSTEM: int
AID_RADIO: int
AID_BLUETOOTH: int
AID_GRAPHICS: int
AID_INPUT: int
AID_AUDIO: int
AID_CAMERA: int
AID_LOG: int
AID_COMPASS: int
AID_MOUNT: int
AID_WIFI: int
AID_ADB: int
AID_INSTALL: int
AID_MEDIA: int
AID_DHCP: int
AID_SHELL: int
AID_CACHE: int
AID_DIAG: int
AID_NET_BT_ADMIN: int
AID_NET_BT: int
AID_INET: int
AID_NET_RAW: int
AID_MISC: int
AID_NOBODY: int
AID_APP: int
PROC_SPACE_TERM: int
PROC_TAB_TERM: int
PROC_LINE_TERM: int
PROC_COMBINE: int
PROC_OUT_LONG: int
READ_LONG_FORMAT: int[]
PROCESS_STATS_FORMAT: int[]
PROCESS_TOTAL_STATS_FORMAT: int[]
PROC_MEMINFO_FORMAT: int[]
INDEX_USER_TIME: int
INDEX_SYS_TIME: int
INDEX_TOTAL_TIME: int
INDEX_MEM_TOTAL: int
INDEX_MEM_FREE: int
INDEX_MEM_BUFFERS: int
INDEX_MEM_CACHED: int
fieldUid: Field
methodGetUidForPid: Method
methodGetPids: Method
methodReadProcFile: Method
methodGetProperty: Method
readBuf: long[]
uidCache: SparseArray<UidCacheEntry>

getInstance():SystemInfo
SystemInfo()
getUidForPid(int):int
getUidForProcessInfo(RunningAppProcessInfo):int
getPids(int[]):int[]
getProperty(String):String
getUids(int[]):int[]
manualGetInts(String,int[]):int[]
getPidUsrSysTime(int,long[]):boolean
getUsrSysTotalTime(long[]):boolean
getMemInfo(long[]):boolean
readLongFromFile(String):long
getAppId(int,PackageManager):String
getAppIdNoCache(int,PackageManager):String
getUidName(int,PackageManager):String
getUidNameNoCache(int,PackageManager):String
getUidIcon(int,PackageManager):Drawable
getUidIconNoCache(int,PackageManager):Drawable
voidUidCache(int):void
```

Figure 4-5 Class Diagram - SystemInfo

4.1.6 BatteryStats:

This class provides information about the Android device battery. It provides the details for the physical battery which is used in your Android device.

- getInstance-This is a static function, which can be called from other modules to get current battery status for the device.
- getVoltage-This function returns the physical battery voltage for the device.
- getCurrent-This function returns the physical battery current for the device.
- getTemp-This function returns temperature of the physical battery for the device.
- getFullCapacity-This function returns full capacity of the physical battery for the device.

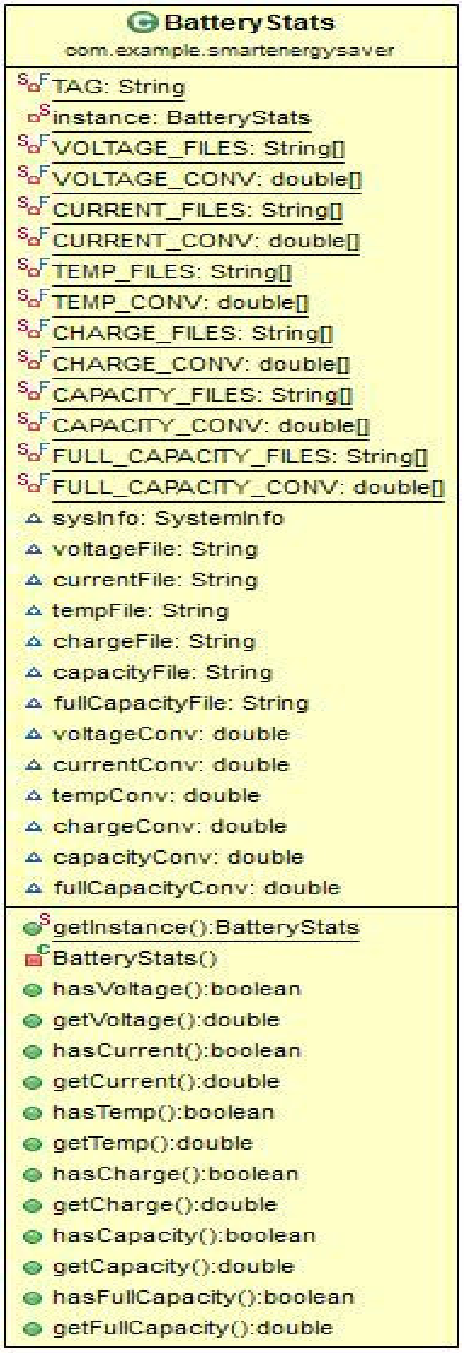


Figure 4-6 Class Diagram - BatteryStats

<u>**4.1.7 PowerEstimator**</u>:

This is the main class which calculates energy consumption for all H/W drivers and individual application. To make the application very modular, this class has association with 3 other classes and one interface as below.

- PowerEstimator - This is the constructor of the PowerEstimator class, It initialize all power components for the application.
- run - This function is called by background thread, which runs in background continuously. This function searches for all running applications and calculates their current consumption. It calculates and stores the details for each application in a predefined time interval. It retrieves the current consumption for each H/W module.

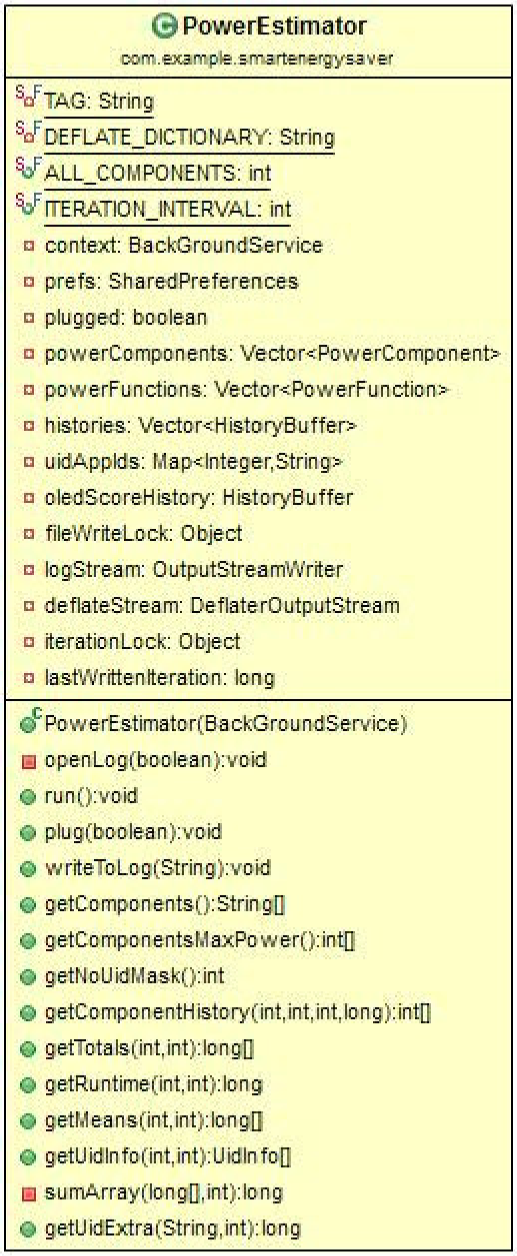

PowerEstimator
com.example.smartenergysaver
TAG: String
DEFLATE_DICTIONARY: String
ALL_COMPONENTS: int
ITERATION_INTERVAL: int
context: BackGroundService
prefs: SharedPreferences
plugged: boolean
powerComponents: Vector<PowerComponent>
powerFunctions: Vector<PowerFunction>
histories: Vector<HistoryBuffer>
uidAppIds: Map<Integer,String>
oledScoreHistory: HistoryBuffer
fileWriteLock: Object
logStream: OutputStreamWriter
deflateStream: DeflaterOutputStream
iterationLock: Object
lastWrittenIteration: long
PowerEstimator(BackGroundService)
openLog(boolean):void
run():void
plug(boolean):void
writeToLog(String):void
getComponents():String[]
getComponentsMaxPower():int[]
getNoUidMask():int
getComponentHistory(int,int,int,long):int[]
getTotals(int,int):long[]
getRuntime(int,int):long
getMeans(int,int):long[]
getUidInfo(int,int):UidInfo[]
sumArray(long[],int):long
getUidExtra(String,int):long

Figure 4-7 Class Diagram - PowerEstimator

4.1.8 PowerComponent :- This class calculates real power from different H/W components

- init - This function is called at the beginning of the daemon loop. It initializes the iteration and begins time for each application.
- run - This application runs the daemon loop that collects data for each component.
- getData - This function returns the data point for the given iteration.

4.1.9 HistoryBuffer :- This class helps to keep track of total current consumption from the time when the application start till exit of the application. It manages a custom array which holds all detailed data for an application.

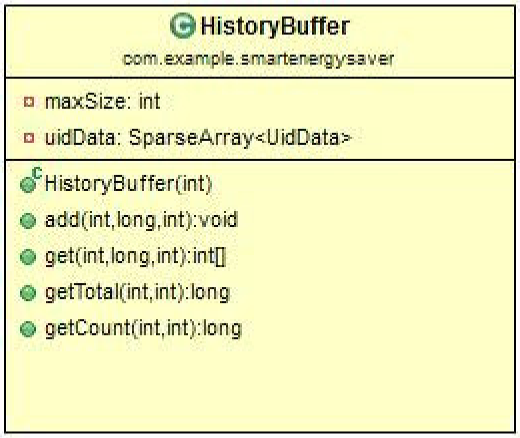


Figure 4-8 Class Diagram - HistoryBuffer

4.1.10 PowerData :- This class is used for initialize the power data for each running application. Also it recycles the data once the application gets closed.

- getCachedPower -This function is called to cache the details stored in the PowerData object.

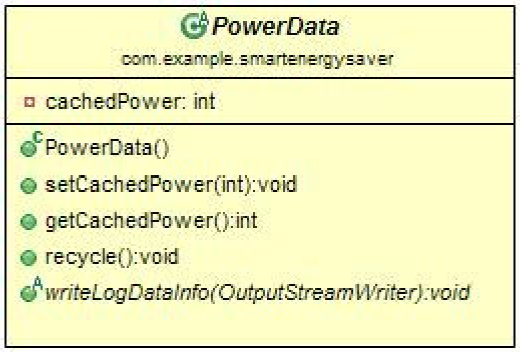


Figure 4-9 Class Diagram - PowerData

4.1.11 Genetic Algorithm Unit

This particular module implements the Genetic Algorithm for the Smart Energy Saver.

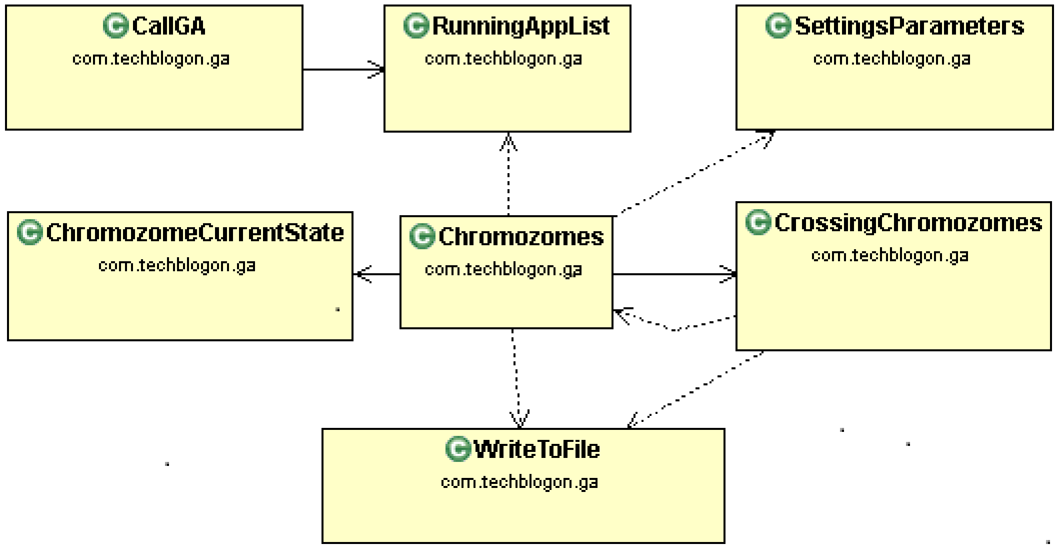


Figure 4-10 Class Diagram - Genetic Algorithm Unit

4.1.12 CallGA:

It is the entry point for GA module. This class stores running application list and its current consumption value for the next level of processing.

- addToRunningAppList - This function receives current application name and unique ID for each application. Then it passes the data to next level for further processing.

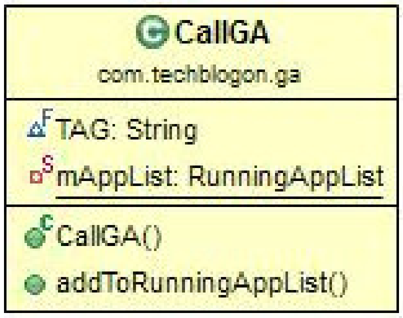


Figure 4-11 Class Diagram - CallGA

<u>4.1.13 Chromozomes</u> :

This class generates the initial population (random chromosomes list) for all running applications. This class adds current consumption value for each chromosomes series. Then it stores all required values in a vector for further processing. This class also calculates the minimum current list from the list of all chromosomes series.

- genChromozomesList :- This method generates the chromosomes
- sumCurrentValue :- This method is recursive method where we recursively call it till we land up with our best value
- updateChromozomeList :- This method will update the list by removing half of the chromosomes.

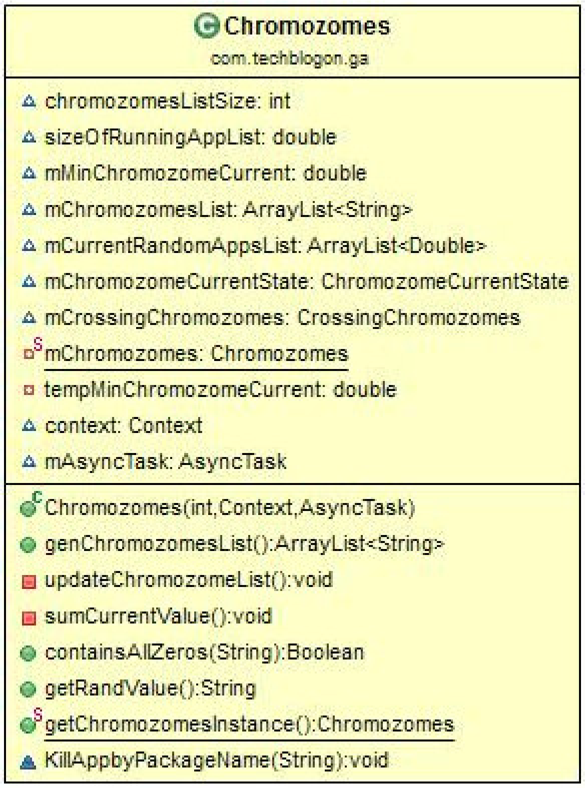

Chromozomes
com.techblogon.ga
chromozomesListSize: int
sizeOfRunningAppList: double
mMinChromozomeCurrent: double
mChromozomesList: ArrayList<String>
mCurrentRandomAppsList: ArrayList<Double>
mChromozomeCurrentState: ChromozomeCurrentState
mCrossingChromozomes: CrossingChromozomes
mChromozomes: Chromozomes
tempMinChromozomeCurrent: double
context: Context
mAsyncTask: AsyncTask
Chromozomes(int,Context,AsyncTask)
genChromozomesList():ArrayList<String>
updateChromozomeList():void
sumCurrentValue():void
containsAllZeros(String):Boolean
getRandValue():String
getChromozomesInstance():Chromozomes
KillAppbyPackageName(String):void

Figure 4-12 Class Diagram - Chromozomes

<u>4.1.14 CrossingChromozomes</u> :

This class receives chromosomes series from the above said Chromozomes Class and perform crossover in the available population. There are different types of crossover which can be performed . The crossover type can by chosen from the settings page. Based on the settings parameters, this class performs the appropriate crossover and then updates the chromosome population. This process runs recursively till the best match found.

- performCrossing - This method performs crossing between a pair of chromosomes series, which are stored in Chromosomes Array List. This method uses selected crossover type to perform crossing between two chromosomes. After crossing it updates the chromosomes list values for further processing.
- clearAll - This class clears all date (clear memory) after calculation.

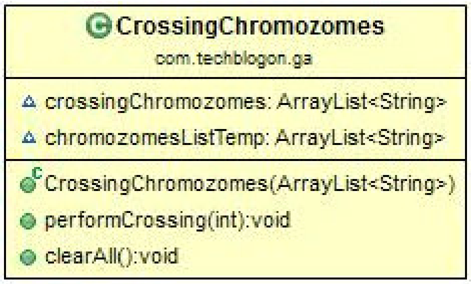

CrossingChromozomes
com.techblogon.ga
crossingChromozomes: ArrayList<String>
chromozomesListTemp: ArrayList<String>
CrossingChromozomes(ArrayList<String>)
performCrossing(int):void
clearAll():void

Figure 4-13 Class Diagram - CrossingChromozomes

4.1.15 ChromozomeCurrentState:

This class gathers current consumption data for each application and find the fittest chromosome by summing the current being consumed by each chromosomes. It also calculates minimum current and sort the current list for getting best chromosome series. This updated series is used for new updated reproductions.

- getMinCurrent - This function calculates minimum current value of each series and return the value for further processing.

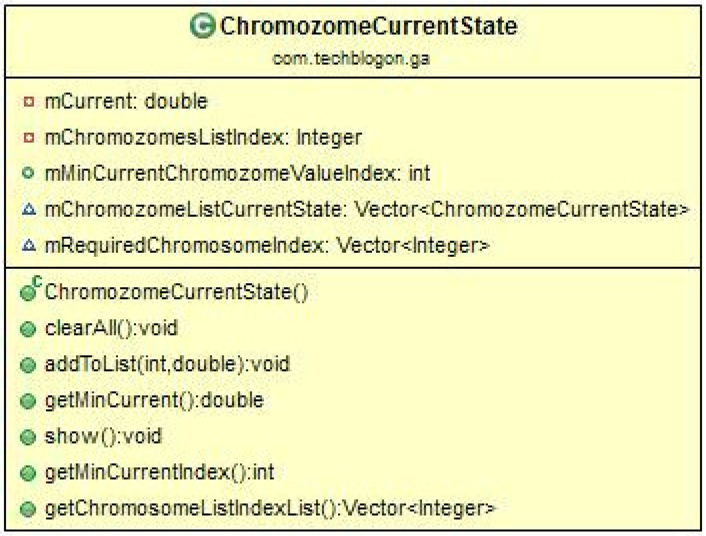

ChromozomeCurrentState
com.techblogon.ga
mCurrent: double
mChromozomesListIndex: Integer
mMinCurrentChromozomeValueIndex: int
mChromozomeListCurrentState: Vector<ChromozomeCurrentState>
mRequiredChromosomeIndex: Vector<Integer>
ChromozomeCurrentState()
clearAll():void
addToList(int,double):void
getMinCurrent():double
show():void
getMinCurrentIndex():int
getChromosomeListIndexList():Vector<Integer>

Figure 4-14 Class Diagram - ChromozomeCurrentState

4.1.16 SettingsParameterss:

This class contains all required setting for the application. This class contains the following user options, which are used for executing the GA algorithm.

1. Population Size
2. Application Launch Frequency
3. Crossover Type
4. Excluded Application List, this represents the hardcoded context. This is the applications which user wants to continue executing.

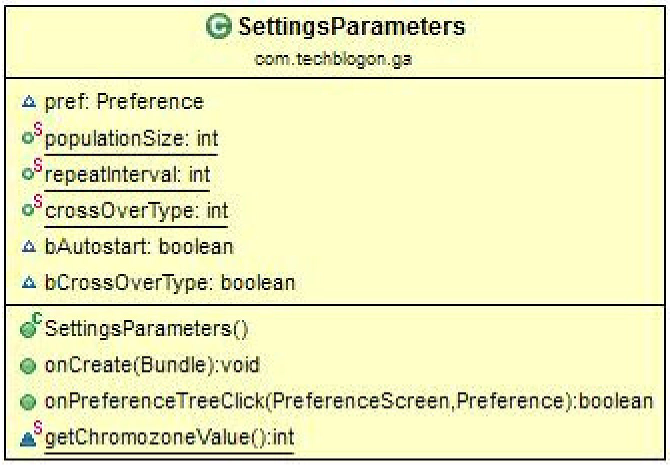


Figure 4-15 Class Diagram - SettingsParameters

4.1.17 Util:

This is a utility class where some common functions are defined and used across the application. It also holds functions to save permanent settings parameter data for the application.

- LoadConfig - Load all default parameters for the application.
- setExcludeFlag - This function excludes applications from including in the termination process. Suppose user wants that the application should not terminate Facebook or WhatsApp application, then GA algorithm will not include these apps while preparing the chromosomes.

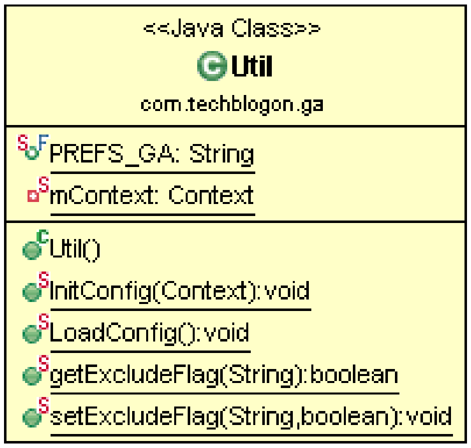
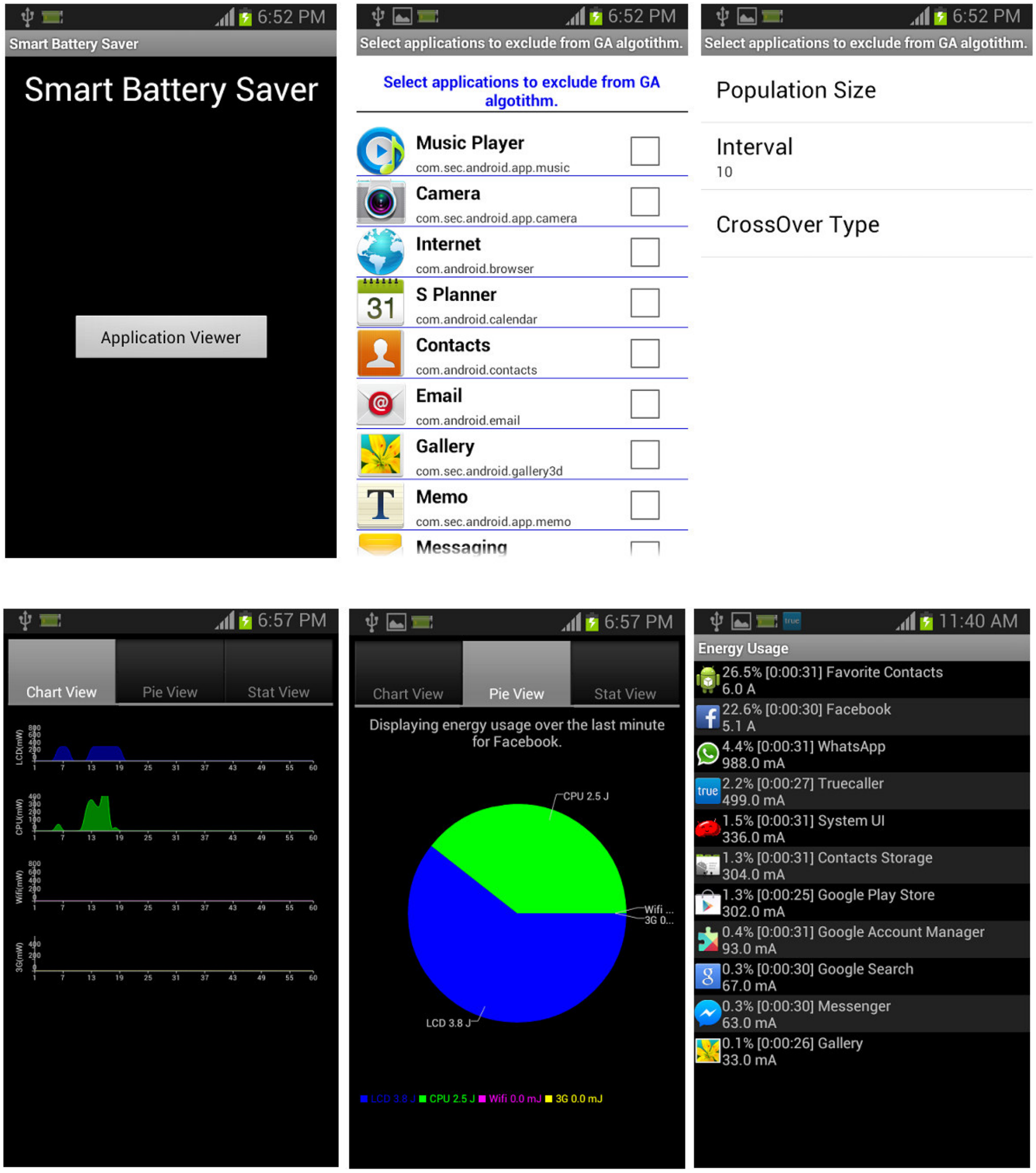


Figure 4-16 Class Diagram - Util


6:52 PM

Smart Battery Saver

Smart Battery Saver

Application Viewer

6:52 PM

Select applications to exclude from GA algotithm.

Select applications to exclude from GA algotithm.

Music Player com.sec.android.app.music	☐
Camera com.sec.android.app.camera	☐
Internet com.android.browser	☐
S Planner com.android.calendar	☐
Contacts com.android.contacts	☐
Email com.android.email	☐
Gallery com.sec.android.gallery3d	☐
Memo com.sec.android.app.memo	☐
Messaging	☐

6:52 PM

Select applications to exclude from GA algotithm.

Population Size

Interval
10

CrossOver Type

6:57 PM

6:57 PM

Displaying energy usage over the last minute for Facebook.

■ LCD 3.8 J ■ CPU 2.5 J ■ Wifi 0.0 mJ ■ 3G 0.0 mJ

11:40 AM

Energy Usage

26.5% [0:00:31] Favorite Contacts
6.0 A
22.6% [0:00:30] Facebook
5.1 A
4.4% [0:00:31] WhatsApp
988.0 mA
2.2% [0:00:27] Truecaller
499.0 mA
1.5% [0:00:31] System UI
336.0 mA
1.3% [0:00:31] Contacts Storage
304.0 mA
1.3% [0:00:25] Google Play Store
302.0 mA
0.4% [0:00:31] Google Account Manager
93.0 mA
0.3% [0:00:30] Google Search
67.0 mA
0.3% [0:00:30] Messenger
63.0 mA
0.1% [0:00:26] Gallery
33.0 mA

Figure 4-17 Smart Energy Saver Screenshots

4.2 Battery Info

Battery Info is a simple application to log the battery level every ten minutes. This application runs in the background and stores the log in a text file.

We used the following tools for developing this application.

1. Editor: Eclipse (Indigo Version)
2. Android SDK
3. Android ADT
4. Android Device
5. Android Emulator

4.2.1 MainActivity:

This is the entry point for battery log application.

- registerReceiver function register the BroadcastReceiver for battery status with intent Intent.ACTION_BATTERY_CHANGED

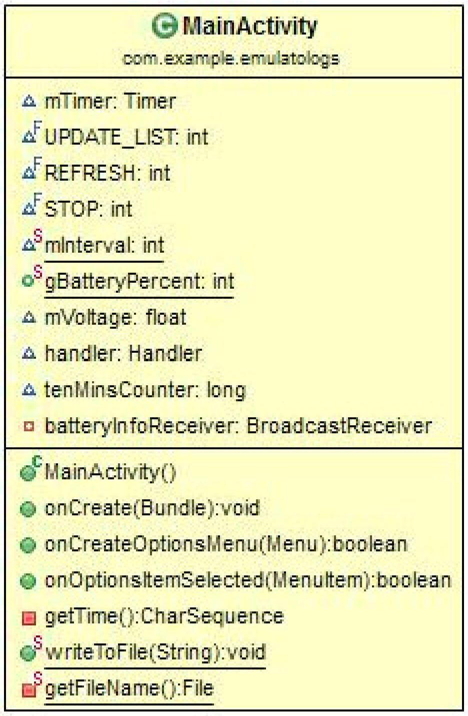


Figure 4-18 Class Diagram - MainActivity

4.2.2MyService Class:

This class is used for logging the battery level at predefined interval of 10 minutes.

- onStartCommand - This is the starting point of the service class.
- writeToFile - This class writes the battery percentage along with the time and date in the log file.

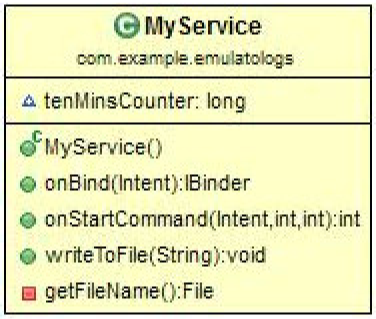


Figure 4-19 Class Diagram - MyService

Chapter 5 EVALUATION AND RESULTS

In this chapter, we present the results obtained by applying the proposed Genetic Algorithm technique to efficiently manage the applications to enhance the battery life. We designed and developed an app called "**Smart Energy Saver"** to select the user's app to be terminated using the Genetic Algorithm technique. Another app called the "**Battery Info"** is designed to log the battery levels of the device at a user defined set time interval.

The following three smart phones have been chosen for conducting the experiments.

1.**Samsung Galaxy Grand2**

Chipset - Qualcomm 8226

RAM - 1.5 GB

2.**Samsung Note2**

Chipset - Exynos 54412

RAM - 2 GB

3. **Samsung Note 8.0**.

Chipset - Exynos 54412

RAM - 2 GB.

The following three classes of applications have been considered in our study:

a. **Video** - In this class of applications, a video was continuously running on the selected hardware. This case was chosen as watching video on a smart phone is one of the most common activities. This is also one of most energy consuming activity.

b. **Music** - In this case music was continuously played on the selected hardware. This case was chosen as listening music on a smart phone is one of the most common activities with most users.

c. **Video + Wi-Fi** - In this case, video was continuously played on the selected device and at the same time device Wi-Fi was switched on and connected to a Wi-Fi hotspot. This case was chosen as most of the users have data connectivity always on.

For each case (video, music , video+Wi-Fi) we used 13 different user-samples to conduct the experiment.

1. Benchmark case (Normal case) - One sample was running under normal conditions, that is using the existing approach for terminating apps. This is the case where our proposed GA app was not used on the mobile.

2. Four Samples for each crossover type (Type 1, Type 2, and Type 3 as already explained in the previous section).

For each crossover type (Type 1, Type 2 and Type 3) the periodic times for the launch of application "Smart Energy Saver" is varied. We used four variations in the study:

a. 10 minutes :- Smart Energy Saver becomes active every 10 minutes.

b. 20 minutes:- Smart Energy Saver becomes active every 20 minutes.

c. 40 minutes:- Smart Energy Saver becomes active every 40 minutes.

d. 60 minutes:- Smart Energy Saver becomes active every 60 minutes.

In all the samples, the percentage of remaining battery-life in the phone will be logged at interval of ten minutes and graphs of battery loss against time will be generated for each use case. At the start of the experiments the phones will be charge completely (100 % battery) and all the phones for an experiment will have same applications running in background.

5.1 Results

The graphs for all the experiments are shown in this section. Each graph shows loss in battery percentage on y axis and time in minutes on x axis. From the graphs, one can conclude the percentage rate of loss in battery by choosing the apps using the proposed GA technique. In other words the graph for a particular use case shows the energy consumption rate for that particular use case.

The slope of the battery loss curves in various cases can be compared with each other to conclude the effectiveness of a particular solution relative to other benchmarked (Normal) solution.

5.1.1 Note 2

On the Galaxy Note 2 the following results have been obtained.

Case 1 - Class of music application

The following graphs are plotted with different use cases (Use Case 1-2) for music class of application, Other use cases are shown in the Appendix A.

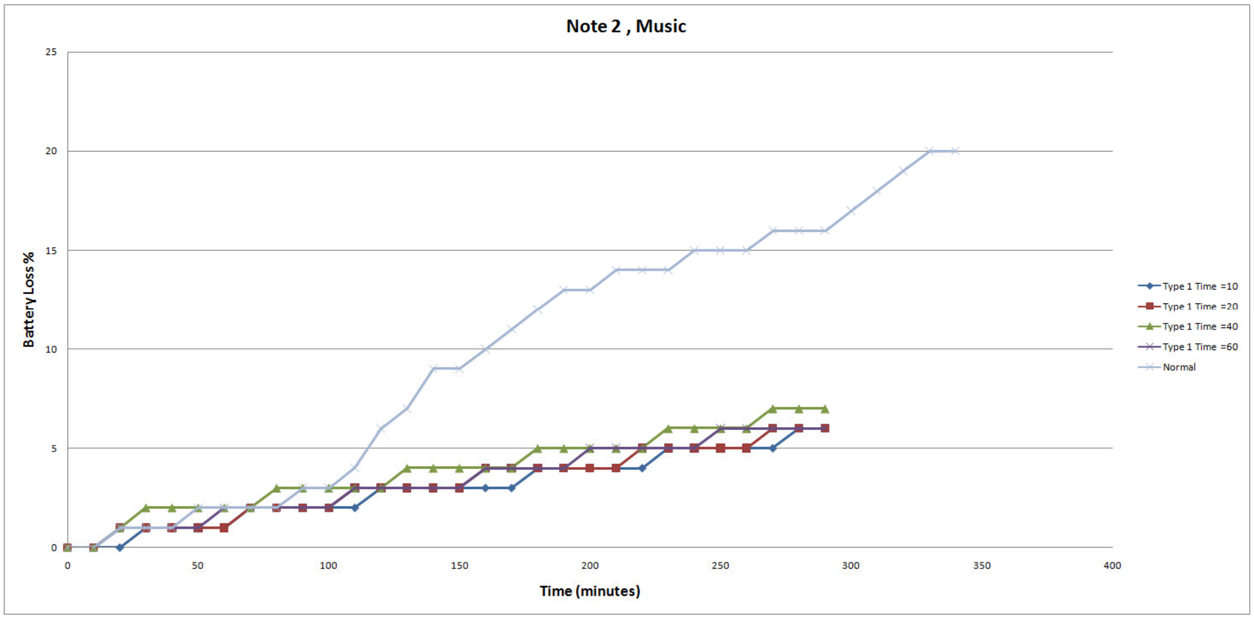


Figure 5-1 Use Case 1

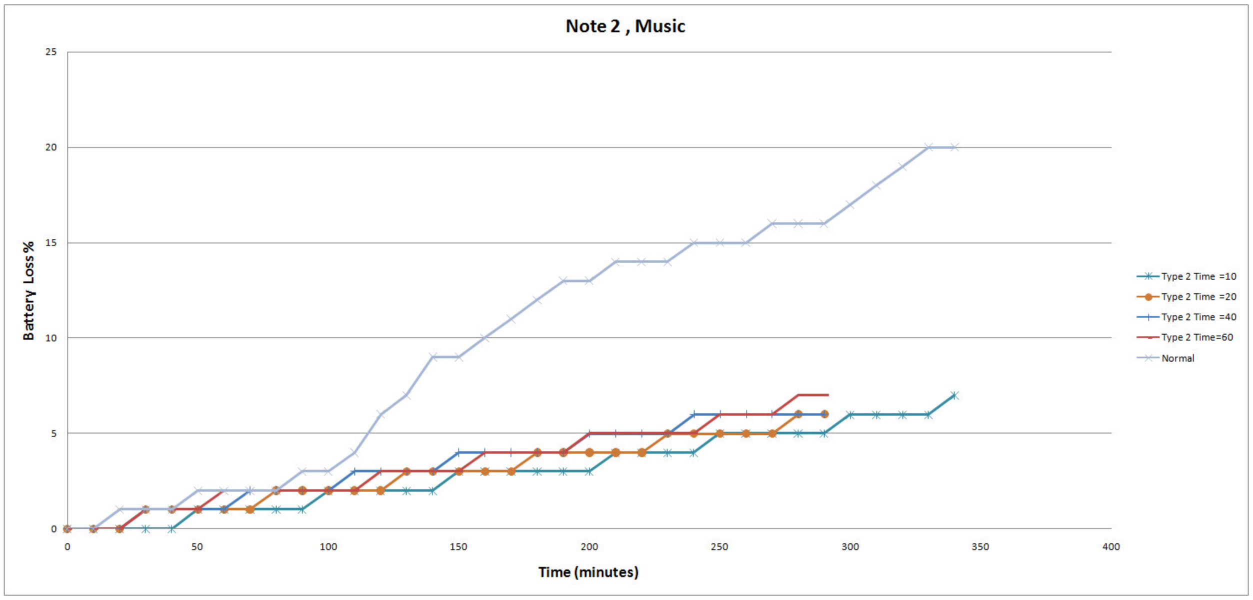


Figure 5-2 Use Case 2

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The GA solution with crossover Type 2 and Application launch frequency of 10 minutes gives the best results.
- During the Initial times till 100 minutes, all the devices show similar battery consumption.
- At the later stages, the devices with Genetic Algorithm implementation show much less battery consumption as seen from the graphs.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig. 5-2) we observe that a given time say 250 minutes, the existing approach shows a battery loss percentage as 15 % (remaining battery life of 85 %) , where as the proposed Genetic Algorithm methodology is 6 % (remaining battery life of 94 %) . So the Genetic Algorithm based approach is 9 % better compared to the existing implementation.

<u>Case 2 - Class of video application</u>

The following graphs are plotted with different use cases (Use Case 3-4) for video class of application, Other use cases are shown in the Appendix A.

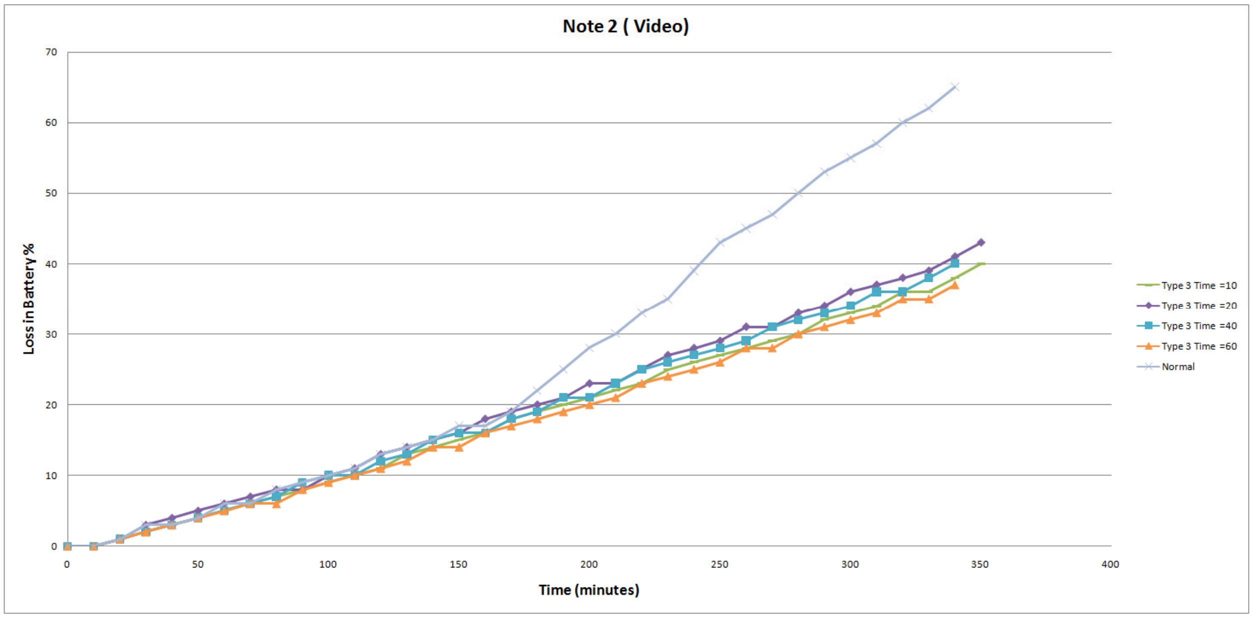


Figure 5-3 Use Case 3

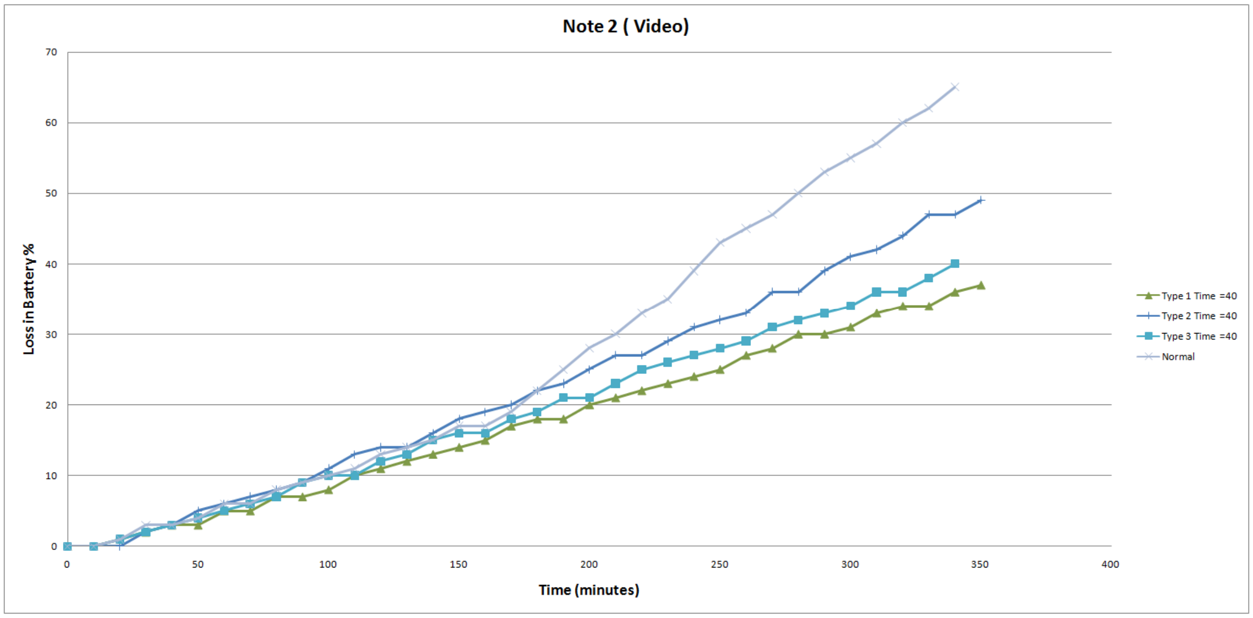


Figure 5-4 Use Case 4

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The GA solution with crossover Type 1 and Application launch frequency of 40 minutes gives the best results
- During the Initial times (till about 70 minutes), all the devices show similar battery consumption.
- At the later stages, the devices with Genetic Algorithm implementation show much less battery consumption.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig. 5-4) we observe that a given time say 300 minutes, the existing approach shows a battery loss percentage as 55 % (remaining battery life of 45 %) , where as the proposed Genetic Algorithm methodology is 40 % (remaining battery life of 60 %) . SO the Genetic Algorithm based approach is 15 % better compared to the existing implementation.

Case 3 - Class of Video + Wi-Fi application

The following graphs are plotted with different use cases (Use Case 5-6) for video + Wi-Fi class of application, Other use cases are shown in the Appendix A.

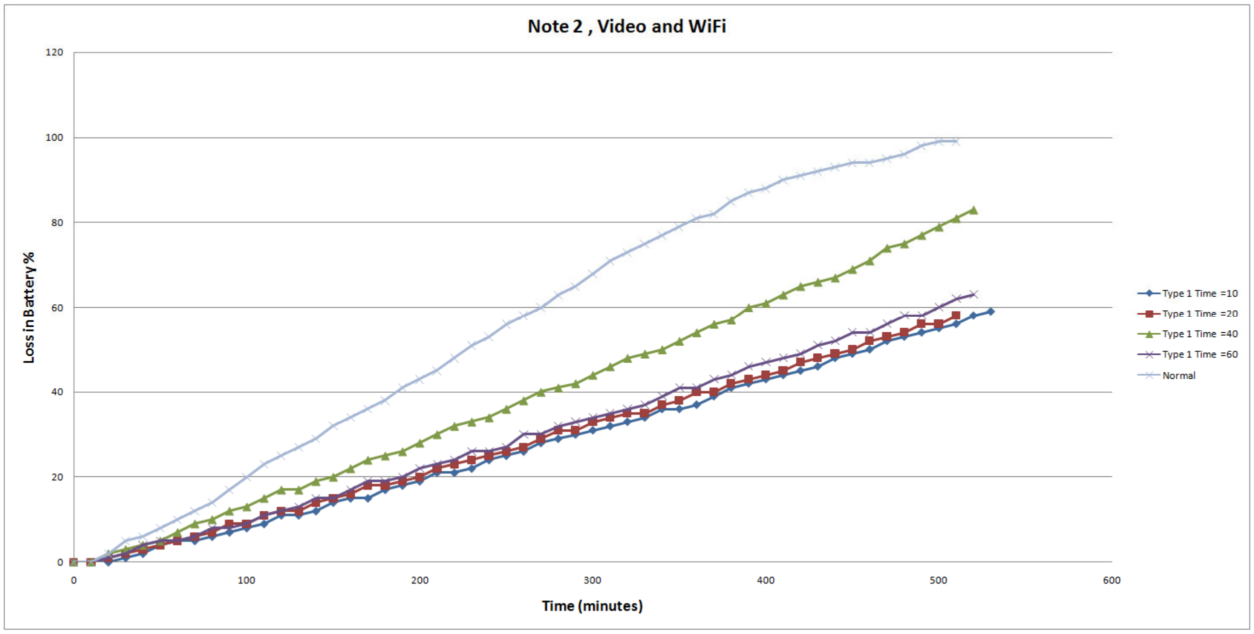


Figure 5-5 Use Case 5

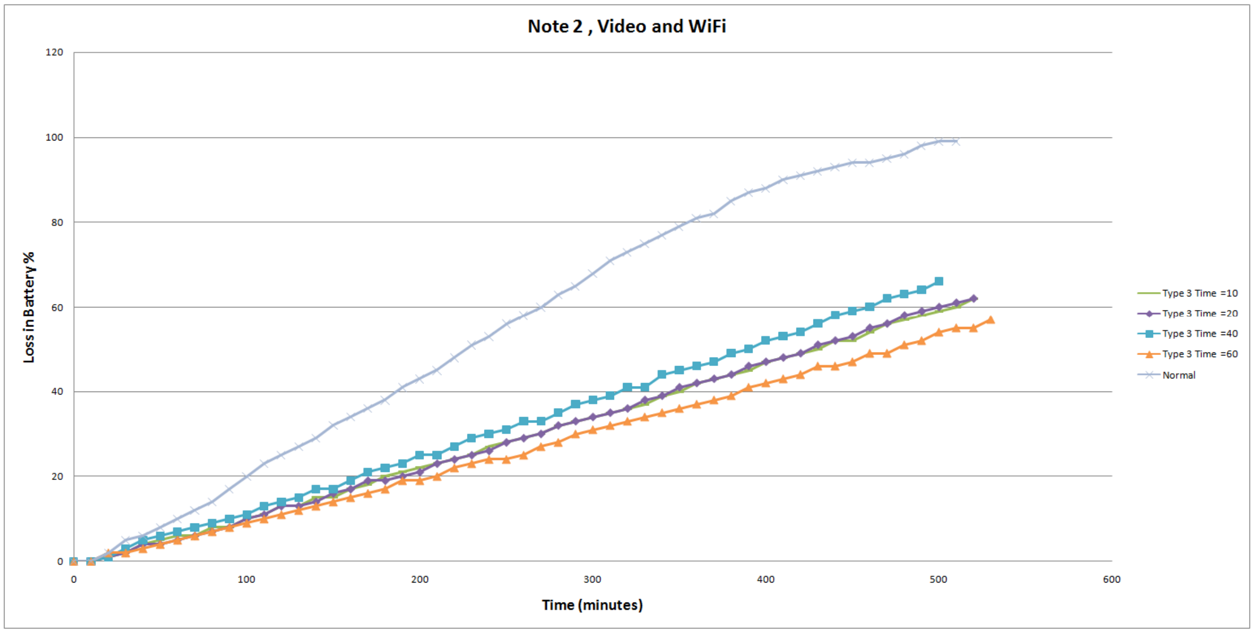


Figure 5-6 Use Case 6

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The GA solution with crossover Type 3 and Application launch frequency of 60 minutes gives the best results
- At all the stages, the devices with Genetic Algorithm implementation show much less battery consumption.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig.5-6) we observe that a given time say 400 minutes, the existing approach shows a battery loss percentage as 90 % (remaining battery life of 10 %) , where as the proposed Genetic Algorithm methodology is 57 % (remaining battery life of 43 %) . SO the Genetic Algorithm based approach is 33 % better compared to the existing implementation.

5.1.2 Grand 2

Case 1 - Class of music application

The following graphs are plotted with different use cases (Use Case 7-8) for music class of application, Other use cases are shown in the Appendix A.

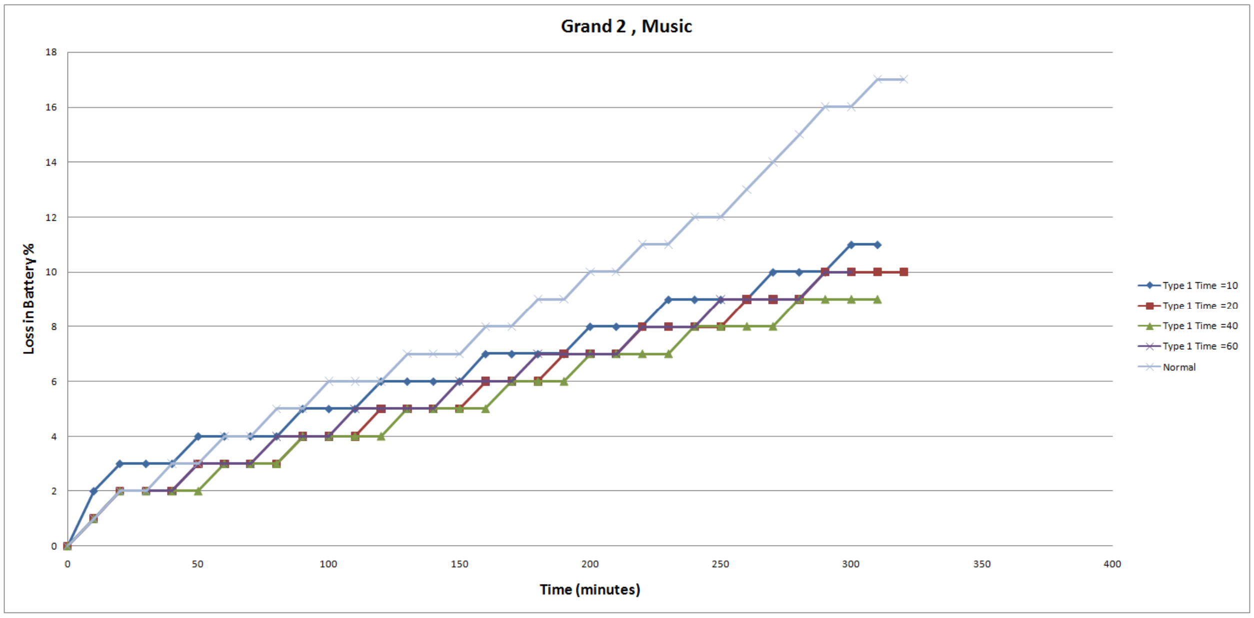


Figure 5-7 Use Case 7

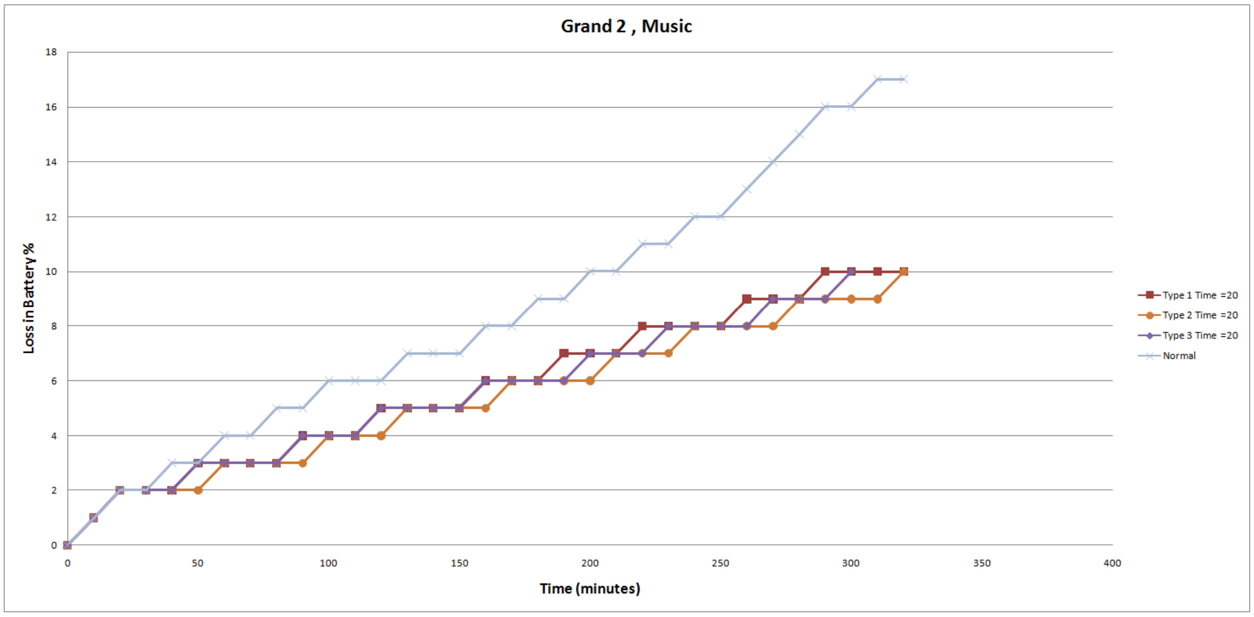


Figure 5-8 Use Case 8

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The performance of all the GA solutions is almost same, but the GA solution with crossover Type 3 and Application launch frequency of 60 minutes gives marginally better results.
- During the Initial times (till about 50 minutes), all the devices show similar battery consumption.
- At the later stages, the devices with Genetic Algorithm implementation show much less battery consumption.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig.5-8) we observe that a given time say 250 minutes, the existing approach shows a battery loss percentage as 12 % (remaining battery life of 88 %) , where as the proposed Genetic Algorithm methodology is 8 % (remaining battery life of 92 %) . SO the Genetic Algorithm based approach is 4 % better compared to the existing implementation.

<u>Case 2 - Class of video application</u>

The following graphs are plotted with different use cases (Use Case 9-10) for video class of application, Other use cases are shown in the Appendix A.

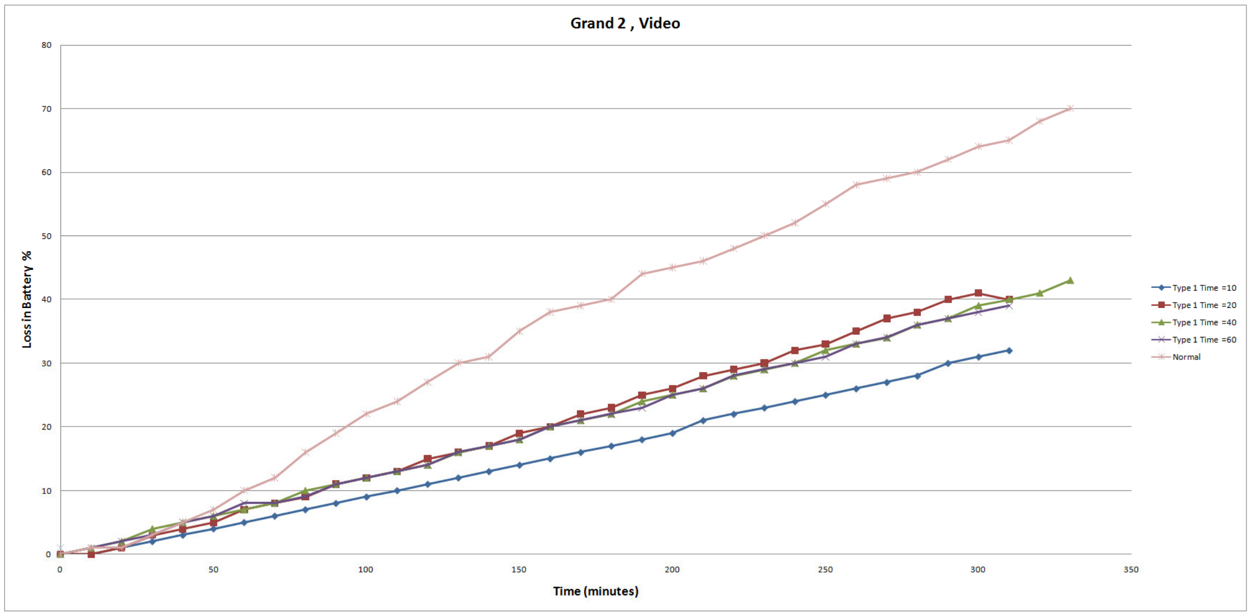


Figure 5-9 Use Case 9

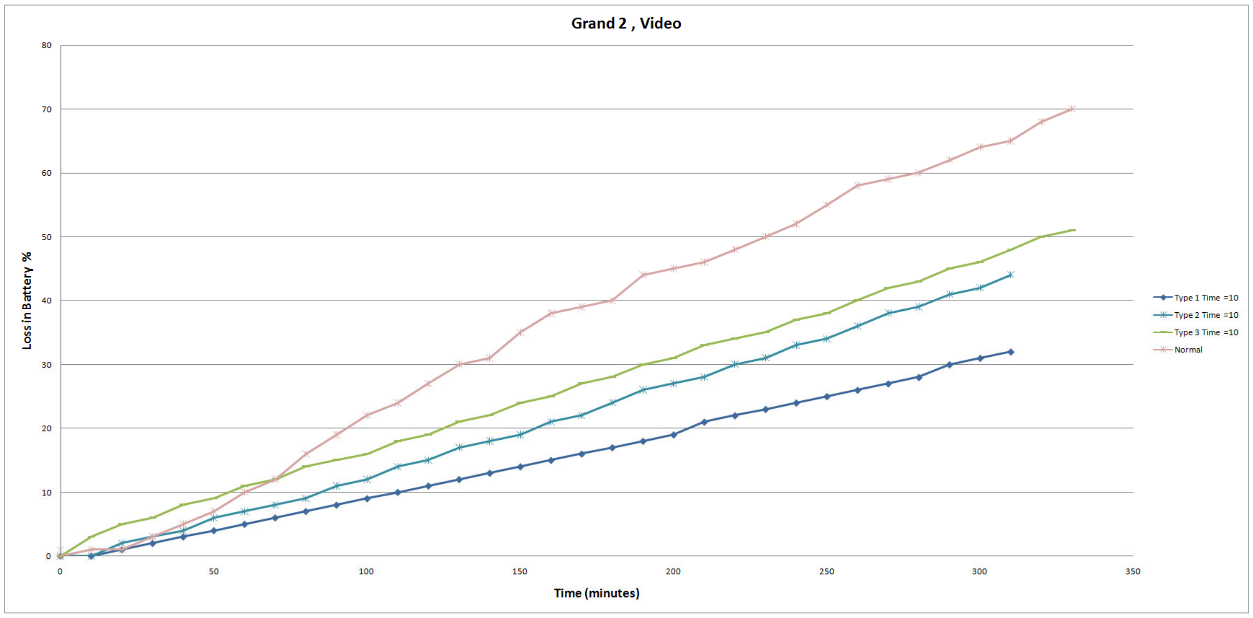


Figure 5-10 Use Case 10

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded:

- The GA solution with crossover Type 3 and Application launch frequency of 40 minutes gives the best results.
- During the Initial times (till about 70 minutes), all the devices show similar battery consumption.
- At the later stages, the devices with the proposed Genetic Algorithm technique shows much less battery consumption and hence a longer battery life.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig. 5-9) we observe that a given time say 300 minutes, the existing approach shows a battery loss percentage as 45 % (remaining battery life of 55 %) , where as the proposed Genetic Algorithm methodology is 27 % (remaining battery life of 73 %) . So the Genetic Algorithm based approach is 18 % better compared to the existing implementation.

<u>Case 3 - Class of video+ Wi-Fi application</u>

The following graphs are plotted with different use cases (Use Case 11-12) for video+Wi-Fi class of application, Other use cases are shown in the Appendix A.

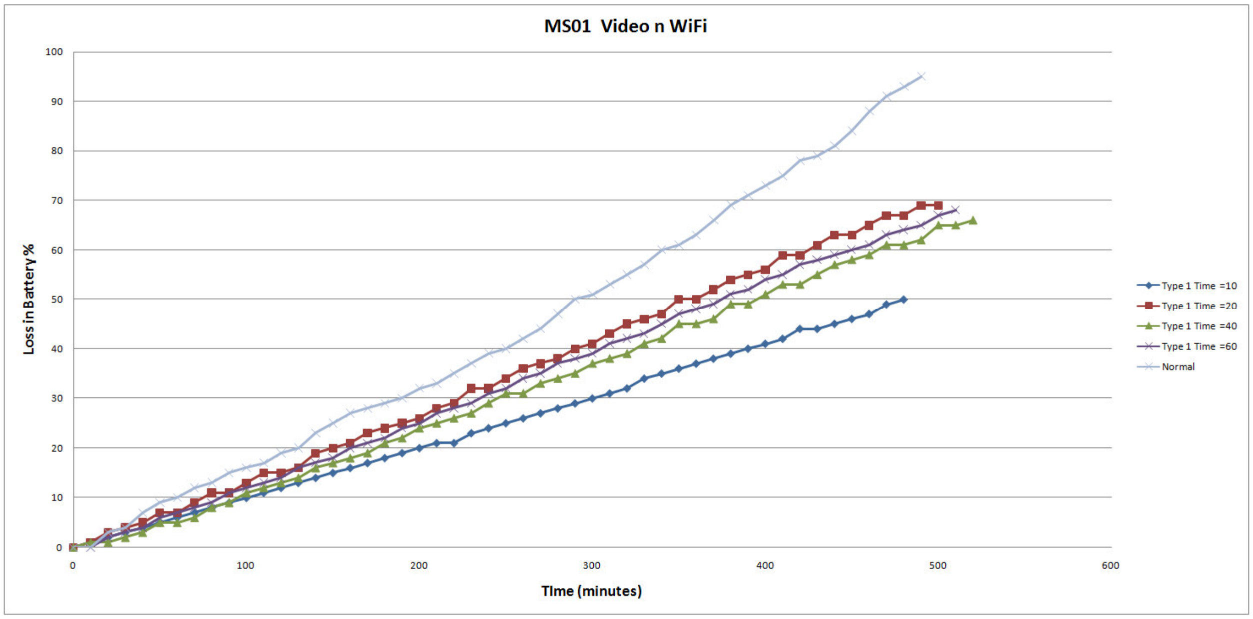


Figure 5-11 Use Case 11

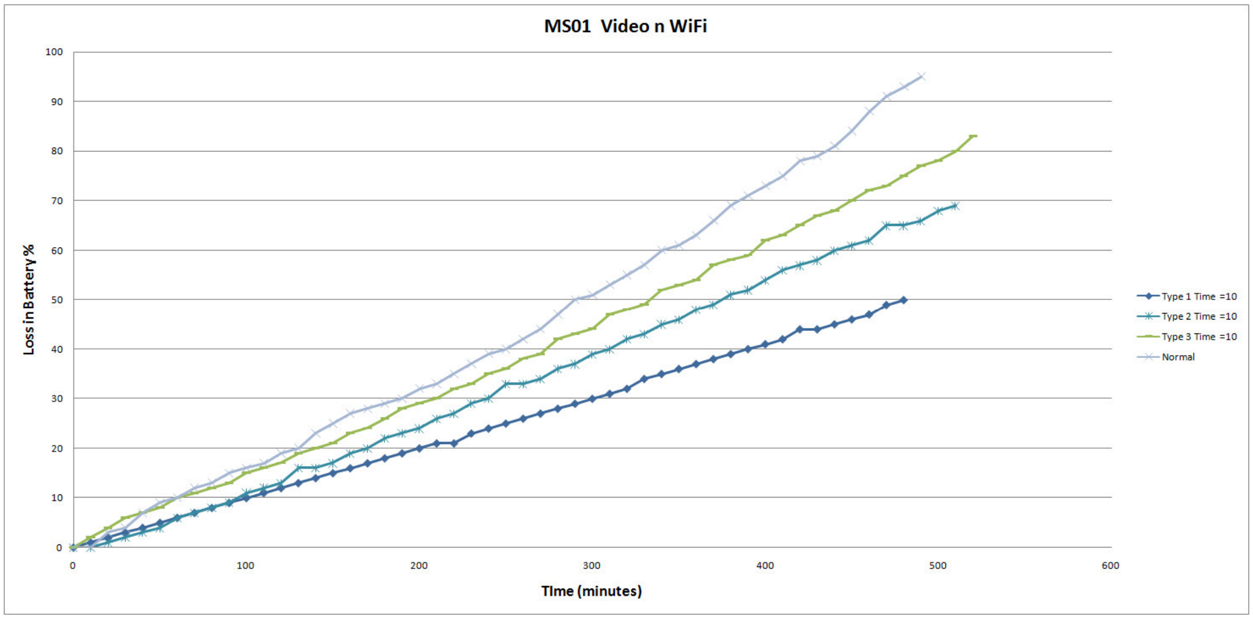


Figure 5-12 Use Case 12

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded:

- The GA solution with crossover Type 1 and Application launch frequency of 10 minutes gives the best results
- At all the stages, the devices with Genetic Algorithm implementation show much less battery consumption as the apps are terminated using this logic.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig. 5-12) we observe that a given time say 400 minutes, the existing approach shows a battery loss percentage as 72 % (remaining battery life of 28 %) , where as the proposed Genetic Algorithm methodology is 62 % (remaining battery life of 38 %) . Thus, the Genetic Algorithm based approach is 10 % better compared to the existing implementation.

5.1.3 Note 8.0

Case 1 - Class of music application

The following graphs are plotted with different use cases (Use Case 13-14) for music class of application, Other use cases are shown in the Appendix A.

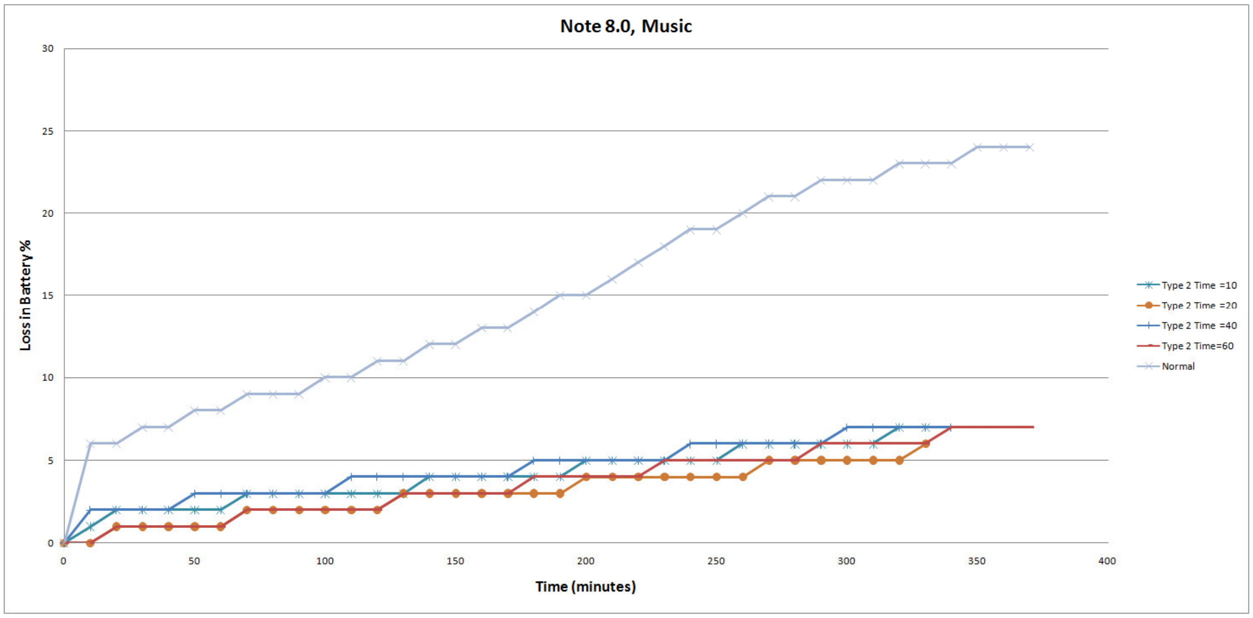


Figure 5-13 Use Case 13

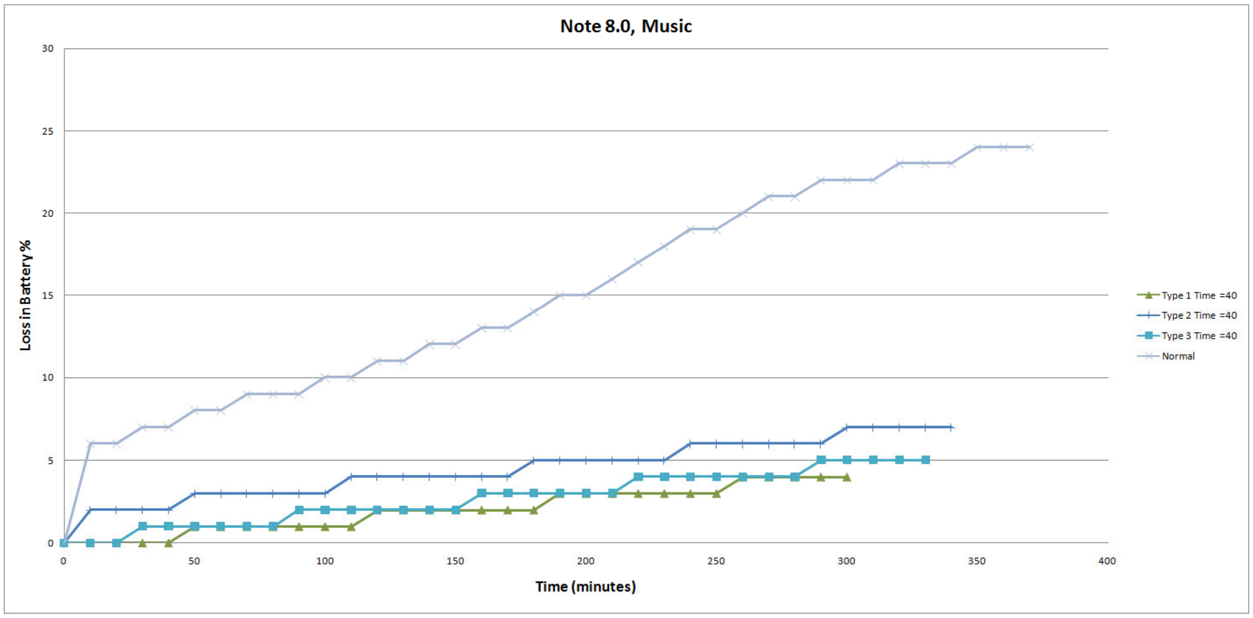


Figure 5-14 Use Case 14

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The performance of all the GA solutions is almost same, but the GA solution with crossover Type 1 and Application launch frequency of 40 minutes gives marginally better results.
- From the start, the devices with Genetic Algorithm implementation show much less battery consumption.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig.5-14) we observe that a given time say 300 minutes, the existing approach shows a battery loss percentage as 22 % (remaining battery life of 78 %) , where as the proposed Genetic Algorithm methodology is 7 % (remaining battery life of 93 %) . The Genetic Algorithm based approach is 15 % better compared to the existing implementation.

<u>Case 2 - Class of video application</u>

The following graphs are plotted with different use cases (Use Case 15-16) for video class of application, Other use cases are shown in the Appendix A.

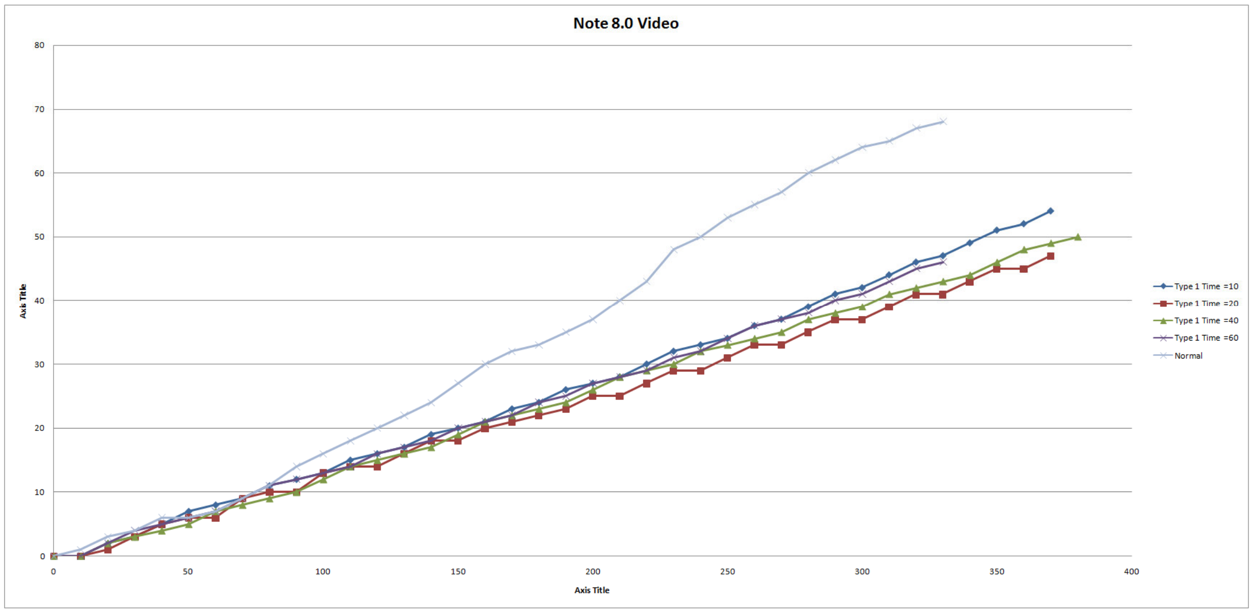


Figure 5-15 Use Case 15

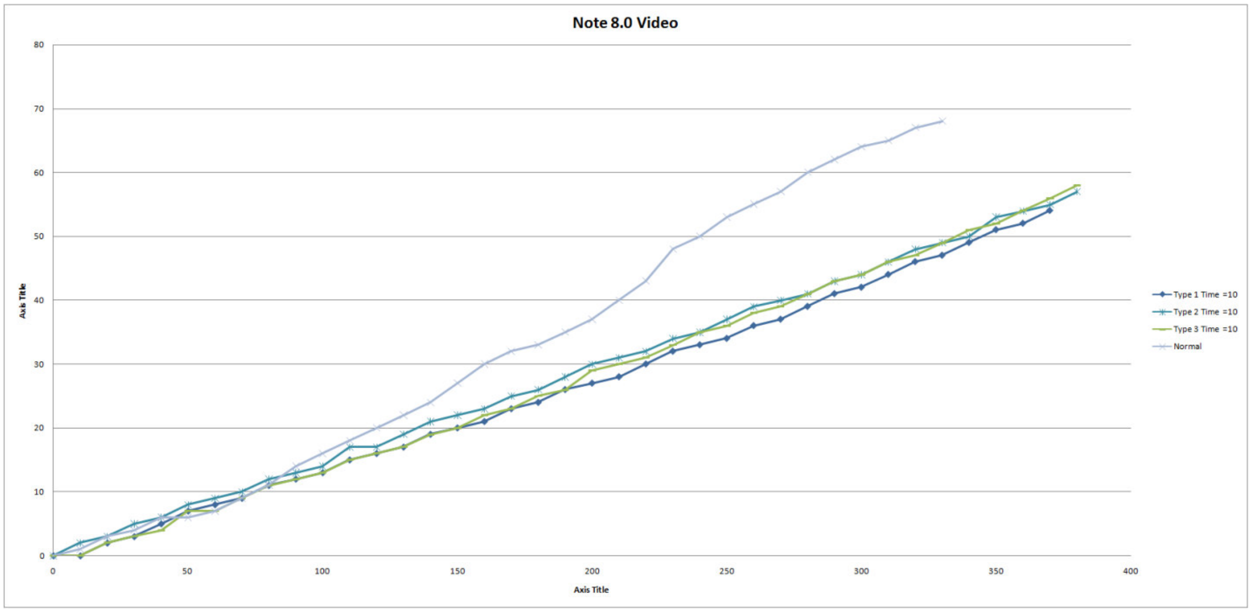


Figure 5-16 Use Case 16

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The GA solution with crossover Type 1 and Application launch frequency of 20 minutes gives the best results.
- During the Initial times (till about 60 minutes), all the devices show similar battery consumption.
- At the later stages, the devices with Genetic Algorithm implementation show much less battery consumption.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig. 5-16) we observe that a given time say minutes, the existing approach shows a battery loss percentage as 65 % (remaining battery life of 35 %) , where as the proposed Genetic Algorithm methodology is 45 % (remaining battery life of 55 %) . SO the Genetic Algorithm based approach is 20 % better compared to the existing implementation.

<u>Case 3 - Class of Video + Wi-Fi application</u>

The following graphs are plotted with different use cases (Use Case 17-18) for video+Wi-Fi class of application, Other use cases are shown in the Appendix A.

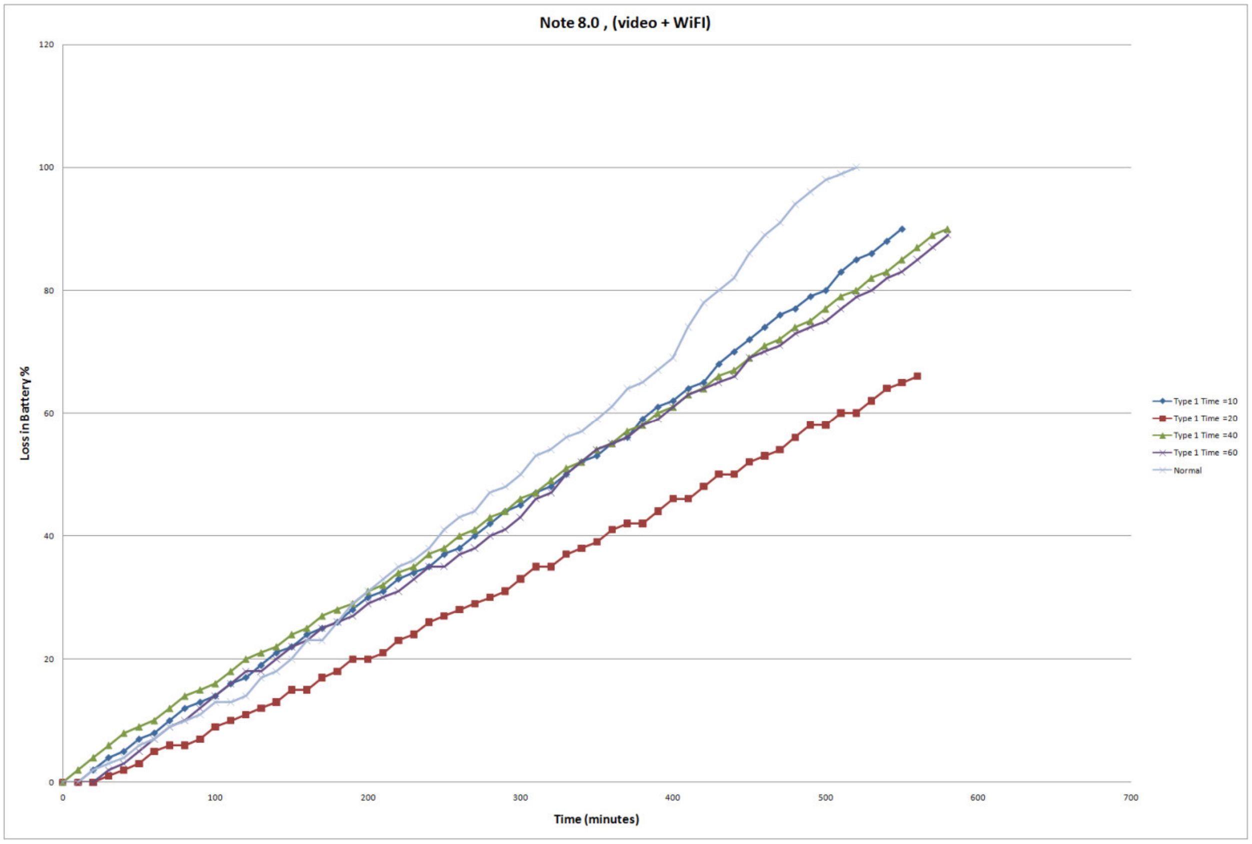


Figure 5-17 Use Case 17

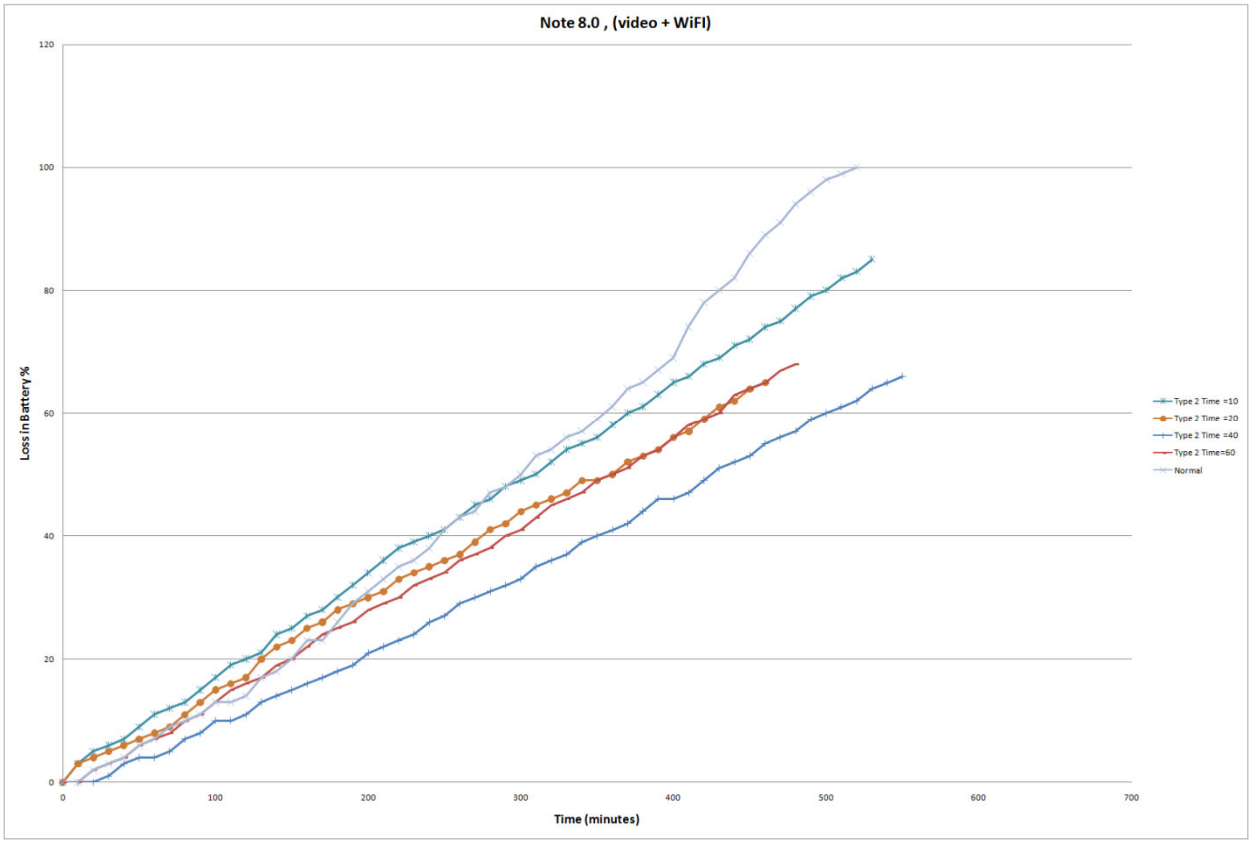


Figure 5-18 Use Case 18

When the graphs for all the use cases under this category are analyzed, the following observations can be concluded

- The GA solution with crossover Type 3 and Application launch frequency of 10 minutes gives the best results
- During the Initial times (till about 200 minutes), all the devices show similar battery consumption.
- At the later stages, the devices with Genetic Algorithm implementation show much less battery consumption.
- The performance is largely independent of the crossover chosen, it also does not depend much on the frequency, at which the application is launched.
- In the graph (Fig.5-17) we observe that a given time say minutes, the existing approach shows a battery loss percentage as 98 % (remaining battery life of 2 %) , where as the proposed Genetic Algorithm methodology is 80 % (remaining battery life of 20 %) . SO

the Genetic Algorithm based approach is 18 % better compared to the existing implementation.

5.3 Result Summary

The following table summarizes the results, giving which is the best and worst performer for the chosen use cases. Please refer to the Appendix A for details of all the use cases.

Use Case	Device	Use Case	Frequency	Best case	Worst Case
				Crossover Type	Crossover Type
4	Note 2	Music	10	2	3
5	Note 2	Music	20	2	3
6	Note 2	Music	40	3	1
7	Note 2	Music	60	1	3
11	Note 2	Video	10	1	2
12	Note 2	Video	20	2	1
13	Note 2	Video	40	1	2
14	Note 2	Video	60	3	1
18	Note 2	Video + Wi-Fi	10	1	2
19	Note 2	Video + Wi-Fi	20	1	3
20	Note 2	Video + Wi-Fi	40	3	1
21	Note 2	Video + Wi-Fi	60	3	2
25	Grand 2	Music	10	2	3
26	Grand 2	Music	20	2	1
27	Grand 2	Music	40	1	2
28	Grand 2	Music	60	2	1
32	Grand 2	Video	10	1	3
33	Grand 2	Video	20	1	2
34	Grand 2	Video	40	3	2
35	Grand 2	Video	60	1	2

39	Grand 2	Video + Wi-Fi	10	1	3
40	Grand 2	Video + Wi-Fi	20	3	2
41	Grand 2	Video + Wi-Fi	40	1	3
42	Grand 2	Video + Wi-Fi	60	1	2
46	Note 8.0	Music	10	2	3
47	Note 8.0	Music	20	2	3
48	Note 8.0	Music	40	1	2
49	Note 8.0	Music	60	3	2
53	Note 8.0	Video	10	1	3
54	Note 8.0	Video	20	1	3
55	Note 8.0	Video	40	1	3
56	Note 8.0	Video	60	1	3
60	Note 8.0	Video + Wi-Fi	10	3	2
61	Note 8.0	Video + Wi-Fi	20	3	2
62	Note 8.0	Video + Wi-Fi	40	2	3
63	Note 8.0	Video + Wi-Fi	60	3	1

Table 5-1 Summarized Results

Chapter 6 Conclusion and Future Work

6.1 Conclusion

Mobile handheld computing is gaining momentum as more and more users are using their Smart Phone as their primary computing and communication device. Smart Phone and Tablet sales have already exceeded the PC and notebook sales, but the battery life continues to be a major constraint towards effective use of the mobile devices by the user. The irony here is that most of the battery is consumed in running or supporting those background applications, which were neither started nor are desired by the user. One of the ways to enhance the battery life is to terminate some of these applications thus reducing the energy requirement of the Smart Phone. This can be one way to enhance the battery life before one can see significant progress in the battery technology itself.

One way to optimize the battery life is to terminate all the applications, other than those desired by the user. Though it may appear trivial but this approach is not practical, as there are many mandatory services running in the background which are essential for normal working of the phone. The best approach will be to find a combination of the application which consumes battery most efficiently. With this aim in mind, the application "Smart Energy Saver" was designed, and executed on three different hardware: Galaxy Grand 2, Note 2 and Note 8.0 .

The graphs from the results have been discussed in chapter 5. From these results we can conclude the following.

Barring a few exceptions (5%), in almost (95 %) all the cases GA based decision making gave better results. In the energy guzzling applications like Video, the difference in the energy consumption is quite prominent. Both the total energy consumption and rate of energy consumption gives much better result in the GA based approach compared to the existing methodologies.

When the results for same scenario on different hardware (Note 2, Galaxy Grand 2 and Note 8.0) are compared , it is noticed that result are almost similar, so it can be concluded that the

Smart Phone chipset and RAM do not affect the effectiveness of GA based technique in any manner.

When the results for same scenario are compared for the frequency at which the application is launched, it can be observed that barring few exceptions, the results are same for all the frequencies. It can be concluded that application itself does not consume much energy.

When the crossover (Type 1, Type 2, and Type 3) for the same scenario is compared, it can be observed that the results do not vary considerably. As expected crossover method chosen in GA does not affects the output considerably. In most of the cases the GA with Type 1 crossover is giving the better results. It can be concluded that though the present implementation performs satisfactorily but may perform better if a more appropriate type of Crossover can be devised.

The current approach validates the hypothesis that Soft Computing based approach is possible for optimizing the battery life but current implementation has one limitation from the fact is it is implemented as an application. The result is that current implementation needs to do a lot of processing for calculating the current being consumed by each application. It will be much more efficient had the same feature be implemented in the native platform, thus the results would have been better compared to the present results.

6.2 Future Work

During the initial days of the preparation of the thesis, substantial effort was spent on understanding the term "context" and how it can be stored. Traditionally most of the energy saving framework rely on the user context to be able to save energy. All these frameworks rely on the application's ability to either change its composition or vary its behavior to conserve the energy.

Moving forward in the same direction, it will be very beneficial if the user context or information about the usage patter for a particular user can be incorporated in the decision making. This way the decision making can be more smart and can enhance the user experience. Most of the context framework described in the literature suffers from the following limitations, First they are very heavy to be implemented in the Smart Phone. Quite many of them require server functionality, to be able to function efficiently. Second User Context is difficult to define, and even more difficult is to store it. The storage required for storing context is too high to fit in a Smart Phone's memory.

In the current implementation the user context is hardcoded and chosen by the user. The application gives option to the user to choose, the activity it wants to perform. Suppose the user wants to watch Video, than the current application gives this choice to the user. Once the user have chosen the Video as the preferred activity, than the application can terminate all the applications but the video player.

For the future evolution of the current work, a way to store and use the context in an efficient way in a Smart Phone need to be devised. In the present study, the fitness function is driven totally by the current or energy consumption by an application but more practical and efficient approach will be to consider the past usage of the Smart Phone by the user along with the energy consumption by an application to determine the fitness value of the application.

References

[1] Davy Preuveneers, Yolande Berbers, “Towards Context-Aware and Resource Driven Self Adaptation for Mobile HandHeld Applications “,ACM1-59593-480-4/07/0003

[2] Ciprian Dobre ,”A Platform to Support Context-Aware Mobile Applications”, Control Systems and Computer Science (CSCS), 19th International IEEE Conference, pp 29-32, May 2013

[3] Jason Flinn ,M. Satyanarayanan ,” Energy-aware adaptation for mobile applications”, 17th ACM Symposium on Operating Systems Principles (SO SP ’99),pp 48-63,Dec 1999

[4] Mohd Norasri Ismail, Rosziati Ibrahim, Mohd Farhan Md Fudzee, “A survey on Content Adaptation Systems towards Energy Consumption Awareness”, http://www.hindawi.com/journals/am/2013/871516/ ,pp 1-8,2013

[5] Spyros Pangiotakis, Athanassia Alonistioti,” Context Aware Composition of Mobile services”,IEEE Computer Society, pp 38-43 , 2006

[6] Mika Raento, Antti Oulasvirta,Renaud Petit, and Hannu Toivonen, “ ContextPhone:A Prototyping Platform for Context-Aware Mobile Applications”, *IEEE CS and IEEE ComSoc , pp 51-59,2005*

[7] Stuart P. Stenton, Richard Hull, Patrick M. Goddi, Josephine E. Reid, Ben J. Clayton,Tom J. Melamed, Susie Wee ,” Multimedia at work Mediascapes: Context-Aware Multimedia Experiences “,IEEE Computer Society,pp 98-105,2007

[8] Paolo Coppola, Vincenzo Della Mea, Luca Di Gaspero, Davide Menegon, Danny Mischis, Stefano Mizzaro, Ivan Scagnetto, and Luca Vassena, “ The Context-Aware Browser” , IEEE Computer Society , pp 38 – 47,2010

[9] Panu Korpipää, Jani Mäntyjärvi,Juha Kela, Heikki Keränen, and Esko-Juhani Malm , “Managing Context Information in Mobile Devices”, *IEEE CS and IEEE ComSoc, pp 42 -51 , 2003*

[10] Suman Nath, “ACE: Exploiting Correlation for Energy-Efficient
and Continuous Context Sensing”,IEEE transactions on mobile computing ,Vol 12 , NO. 8, pp 1472-1486 , 2013

[11] Gaojin Wen, Shengzhong Feng, Yanyi Wan Pingchuang, Jiang Senlin Zhang , " Energy-aware Application Scheduling based on Genetic Algorithm", Seventh International (IEEE) Conference on Natural Computation,pp 2050-2053,2011

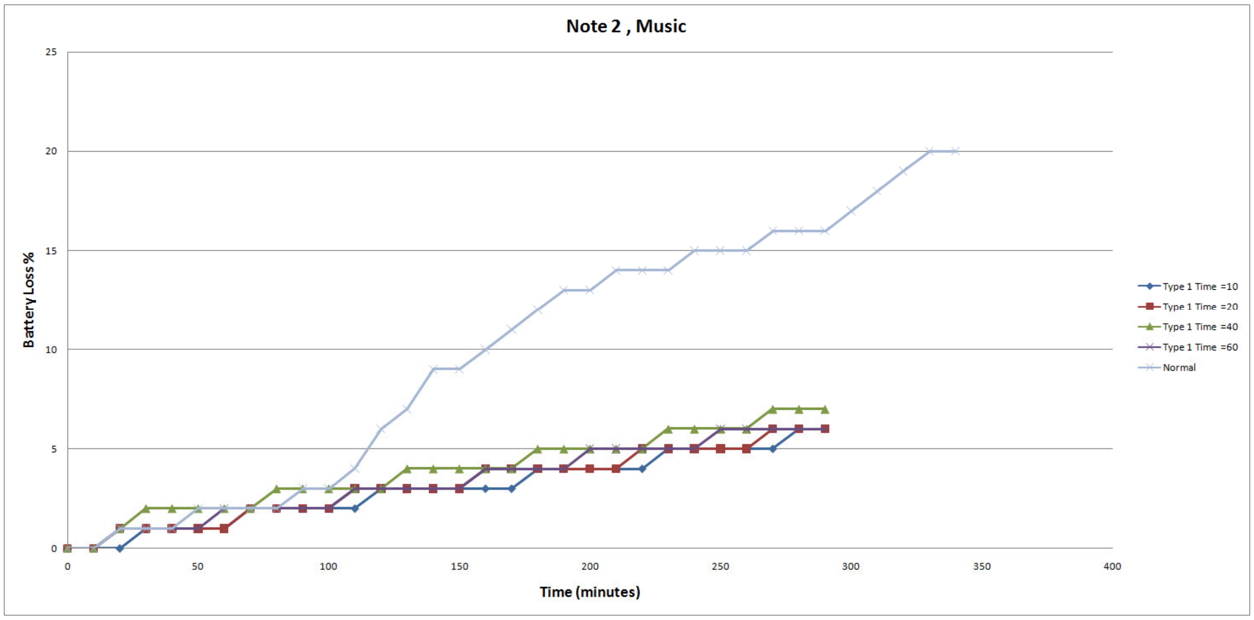
[12] Hassan Salamy, Semih Aslan," Energy-Aware Schedule Optimization on Multicore Systems" , IEEE 56th International Midwest Symposium, pp 109-112, 2013

[13] Xiaoguang Wang," A Genetic Algorithm for Task Scheduling Based on User Overall Satisfaction",IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discover,pp 527- 530, 2012

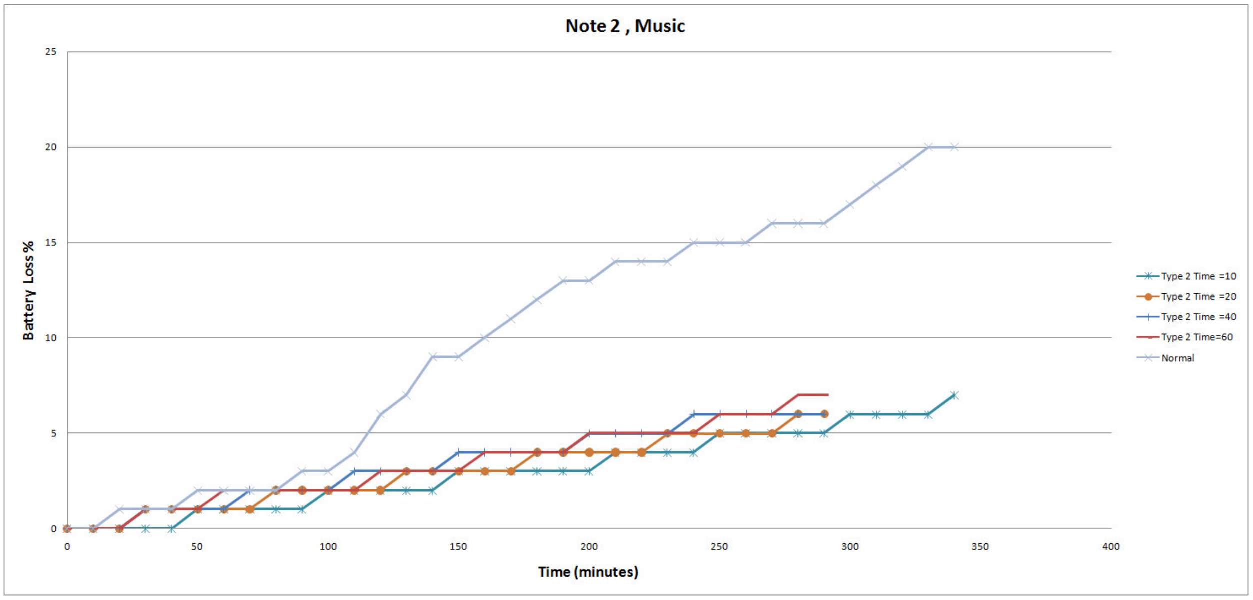
Appendix A

Case 1 Note 2

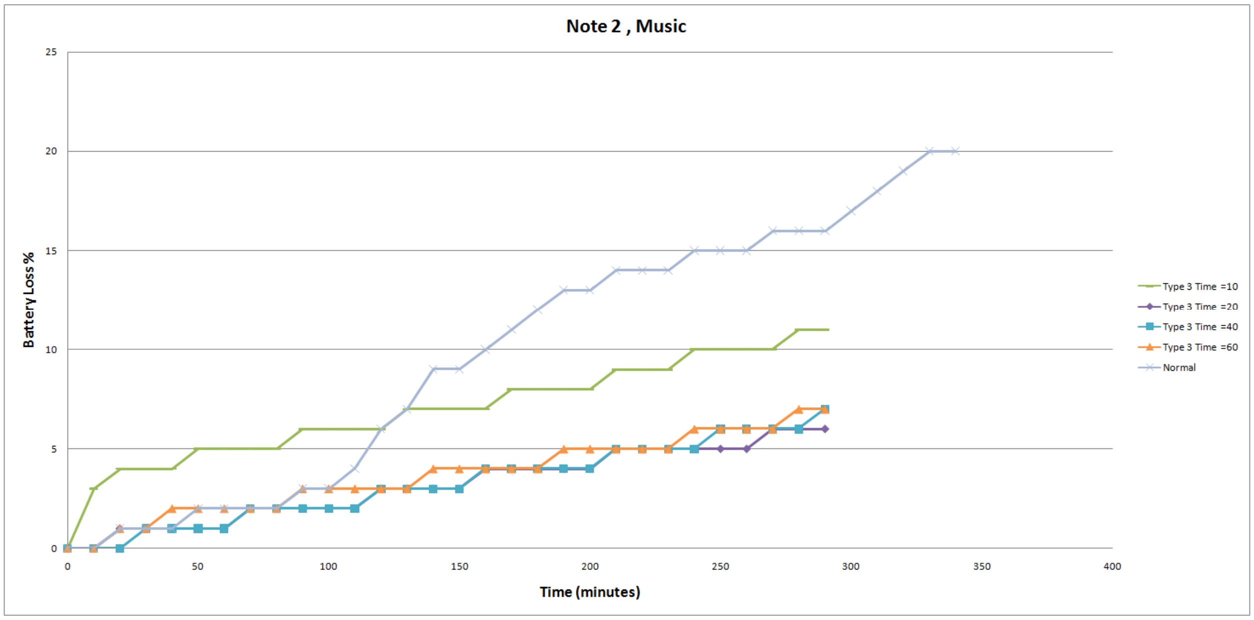
Case 1.1 - Class of music application On Note2



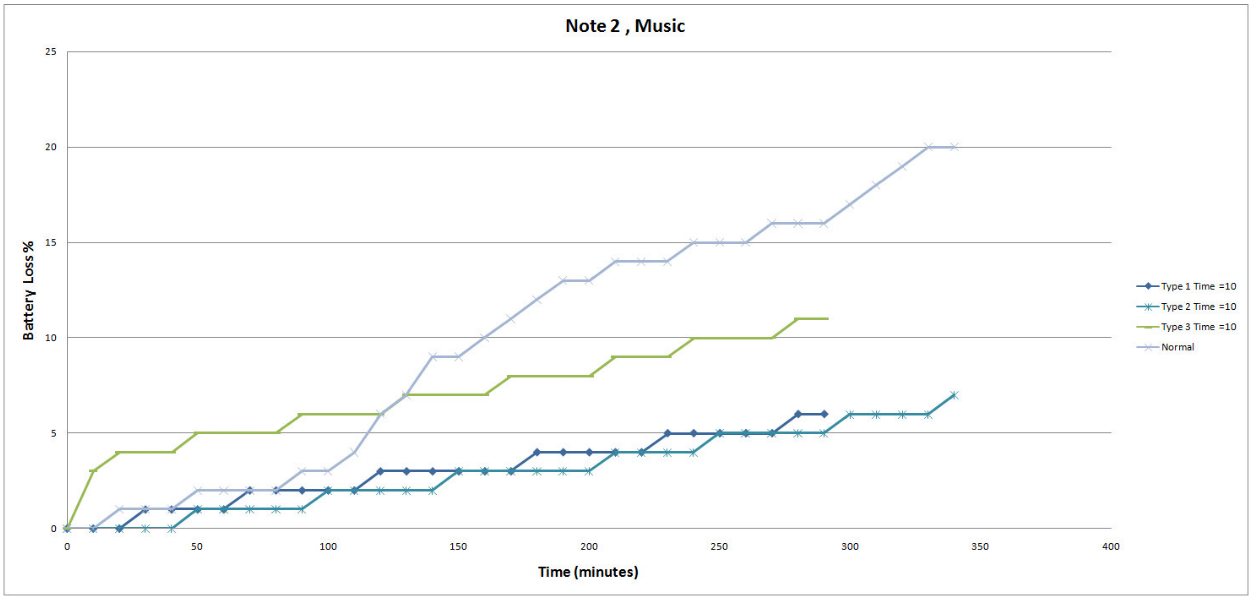
Use Case 1



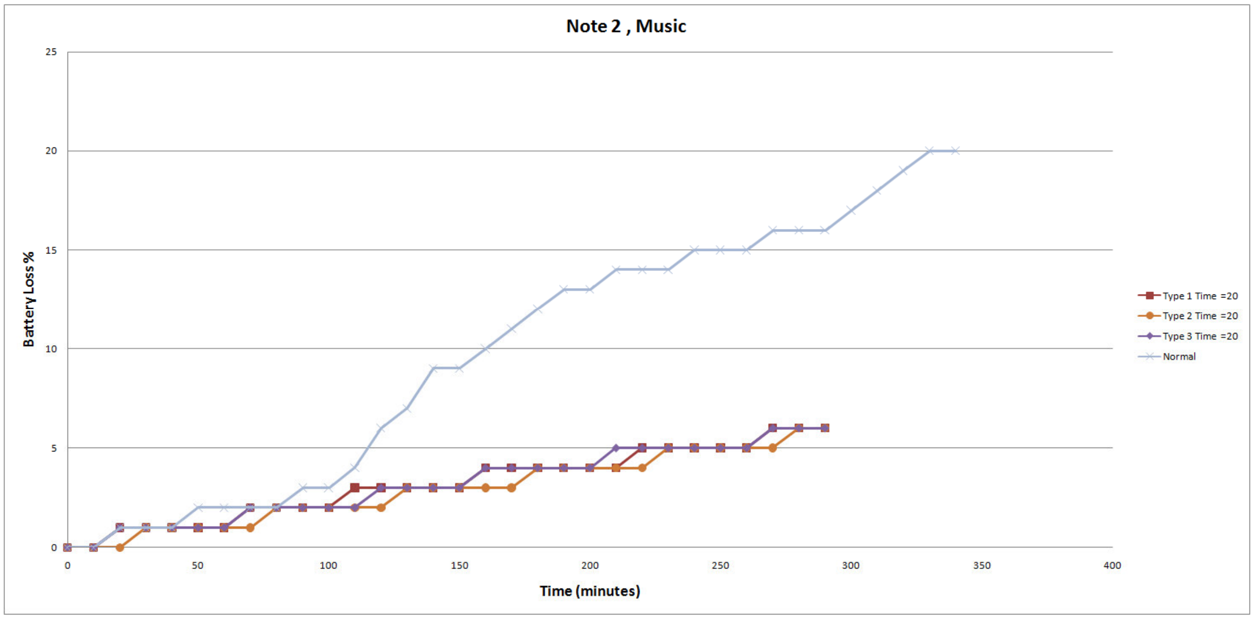
Use Case 2



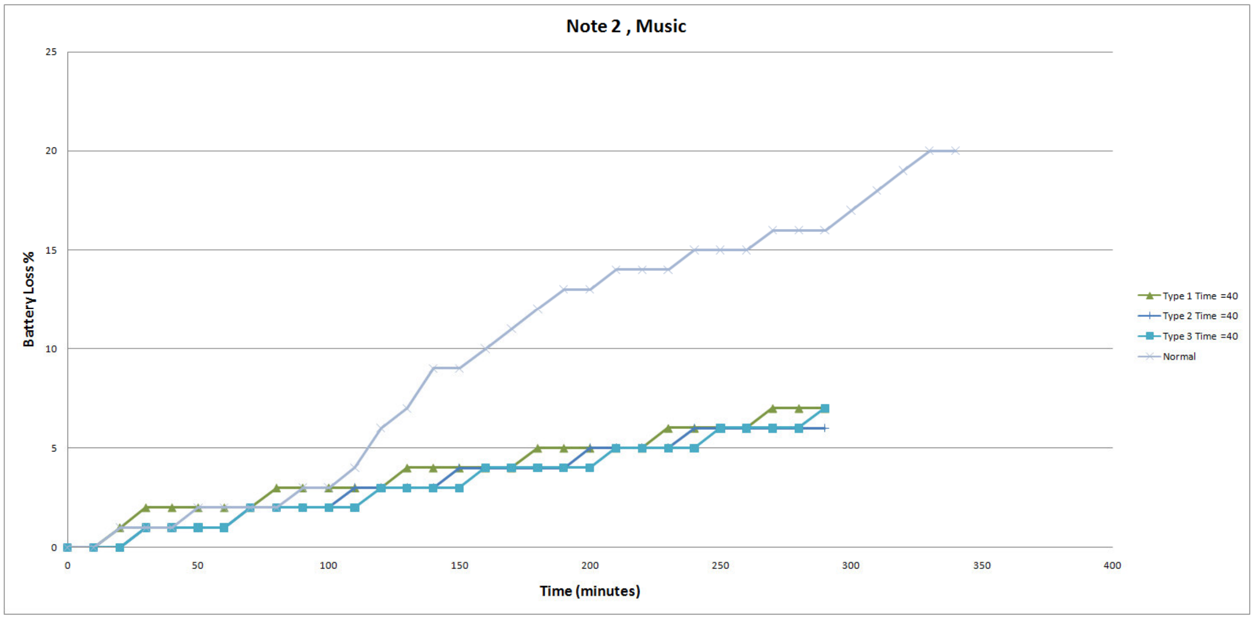
Use Case 3



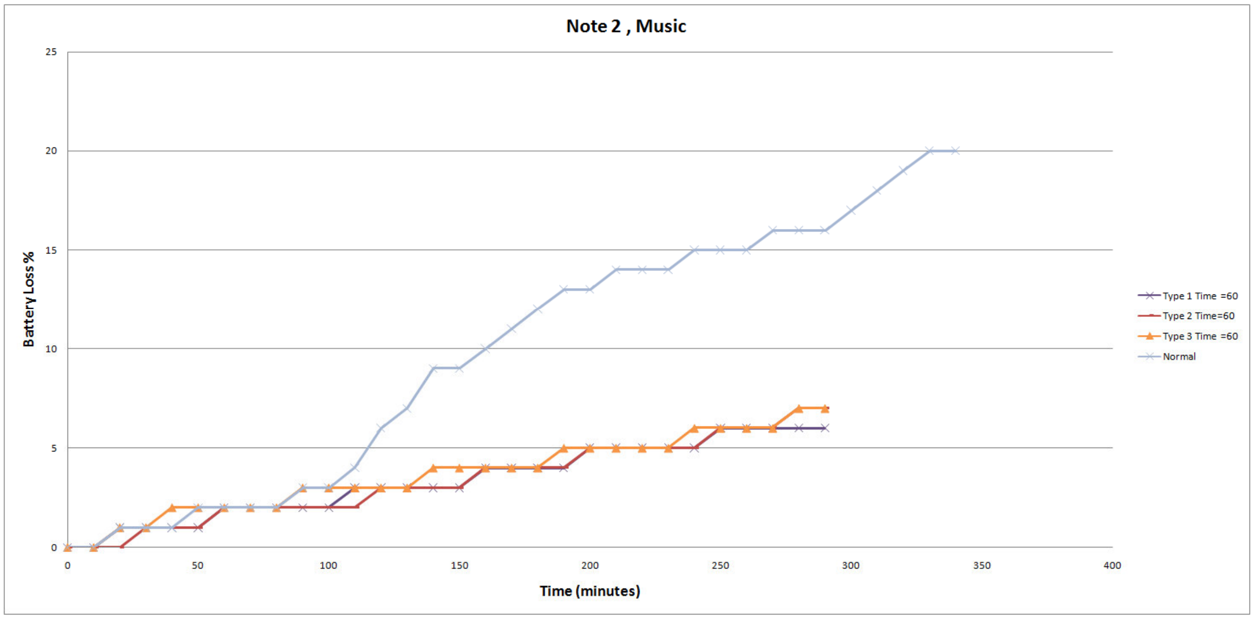
Use Case 4



Use Case 5

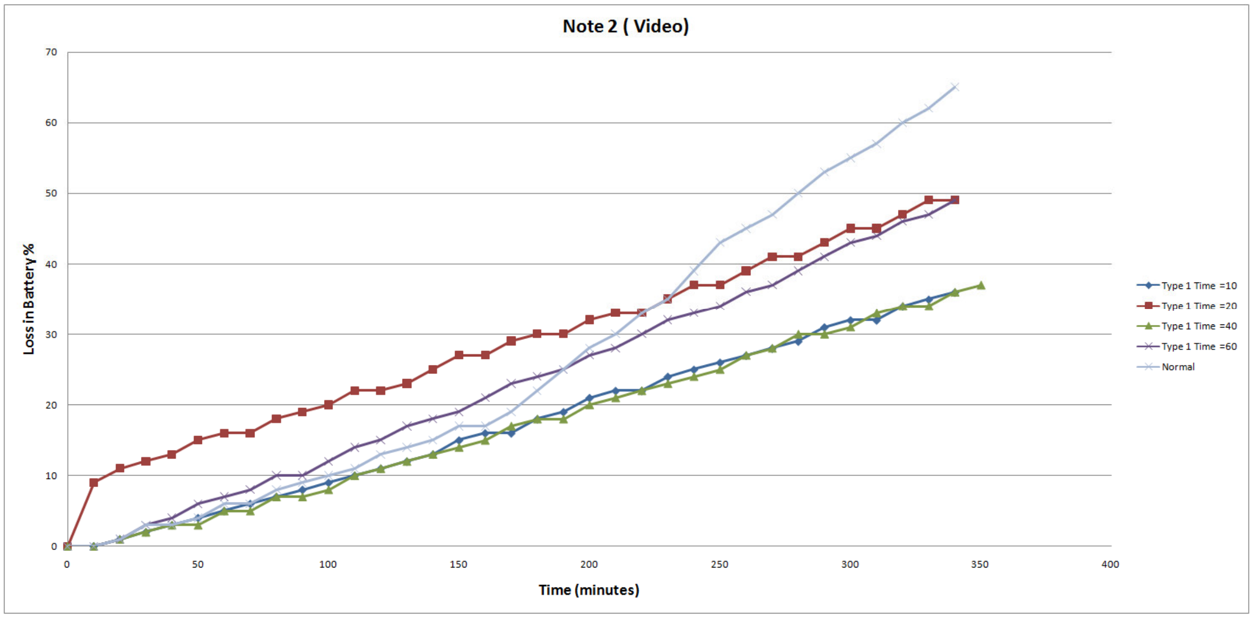


Use Case 6

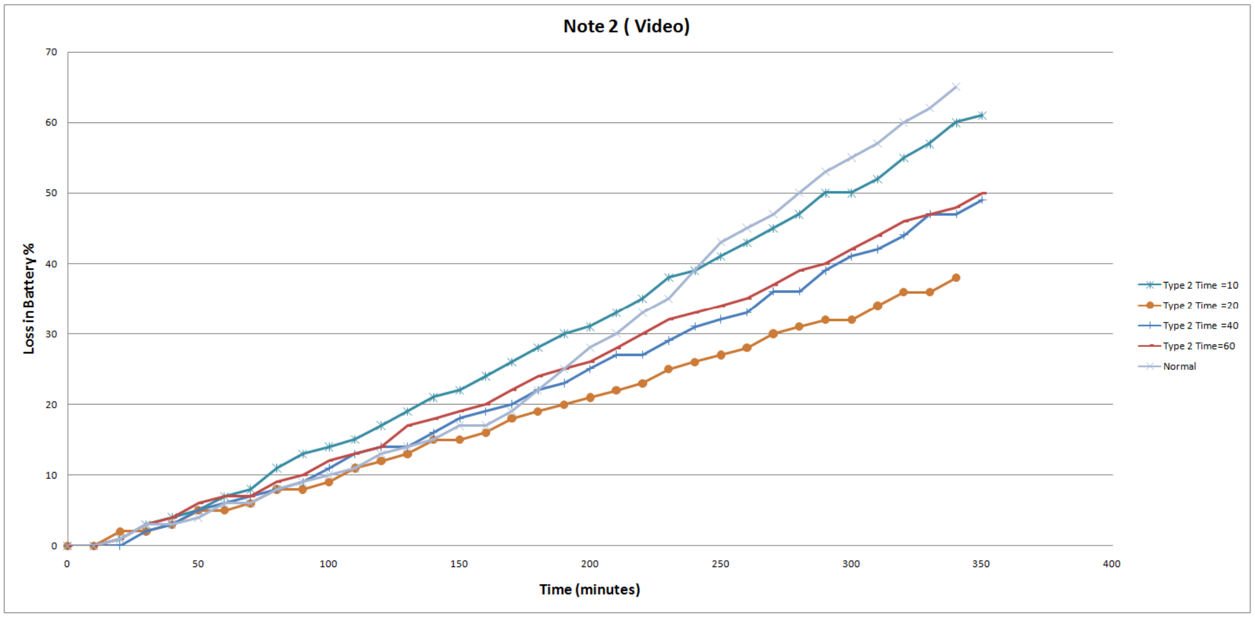


Use Case 7

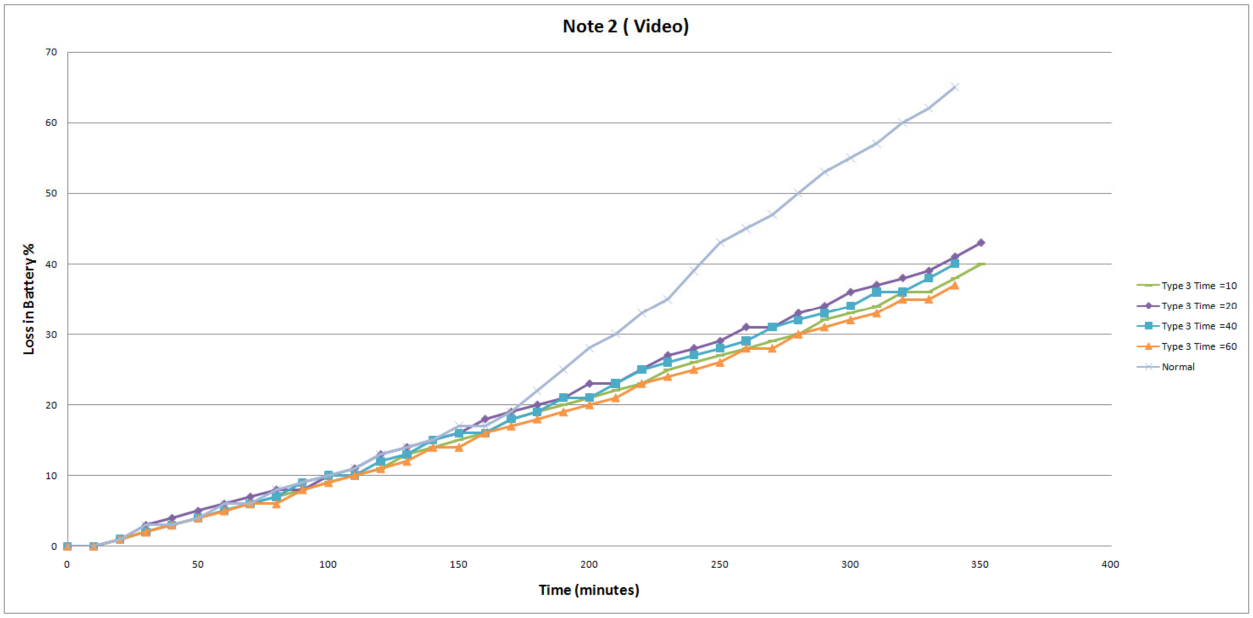
Case 1.2 - Class of video application on Note 2



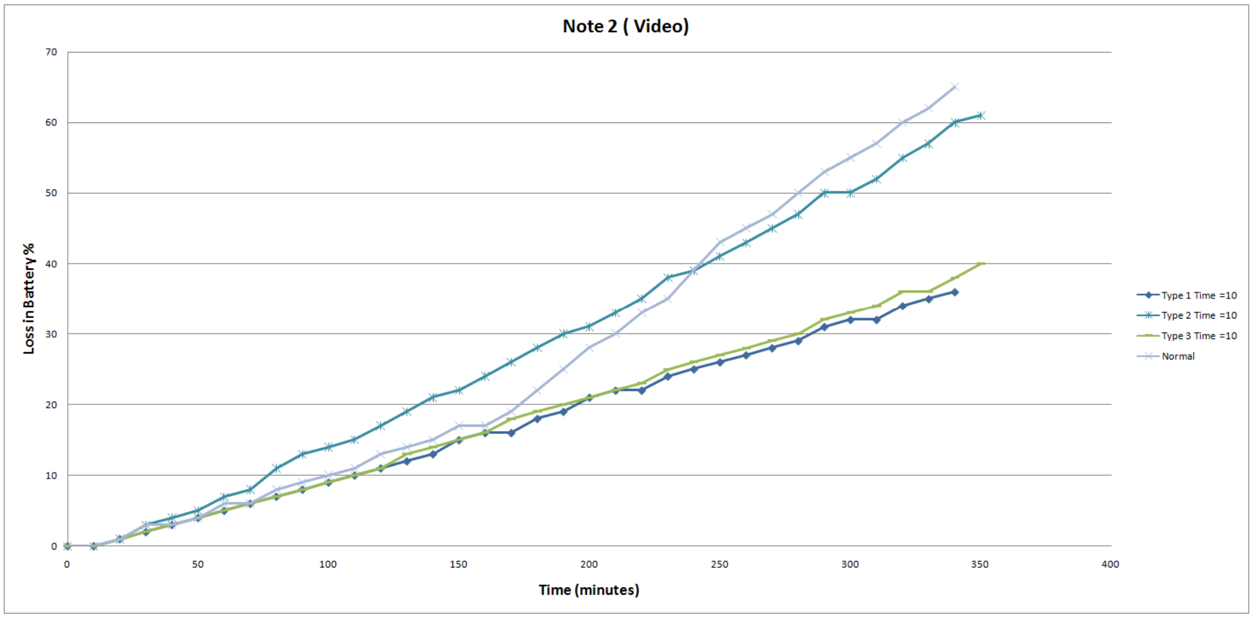
Use Case 8



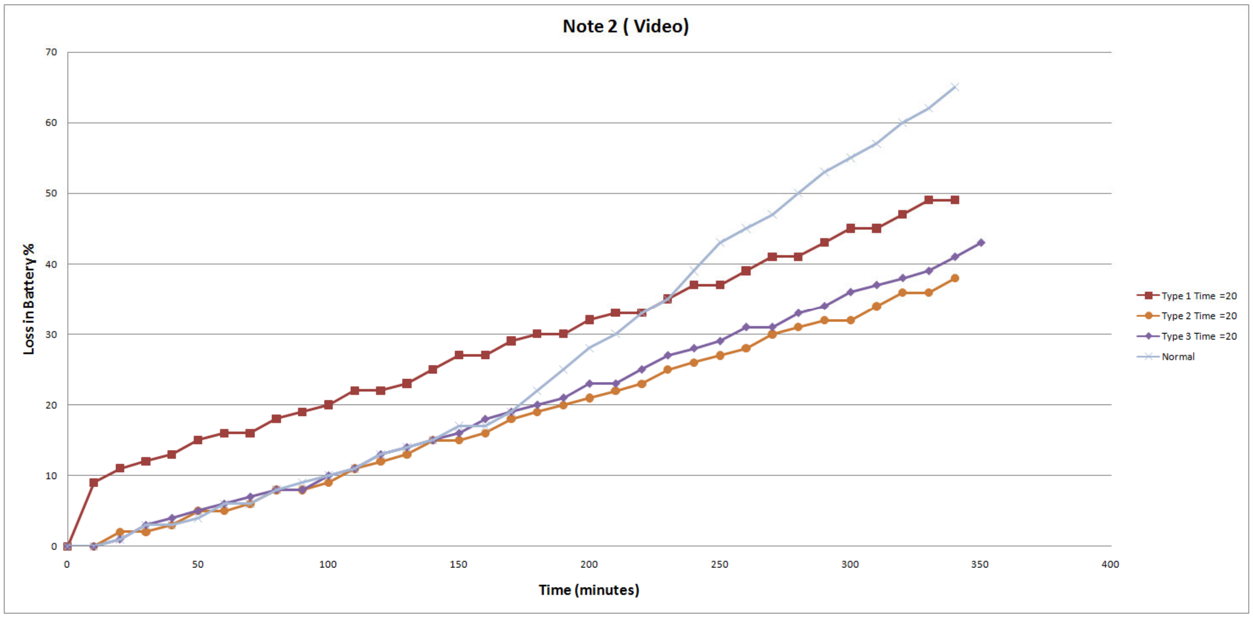
Use Case 9



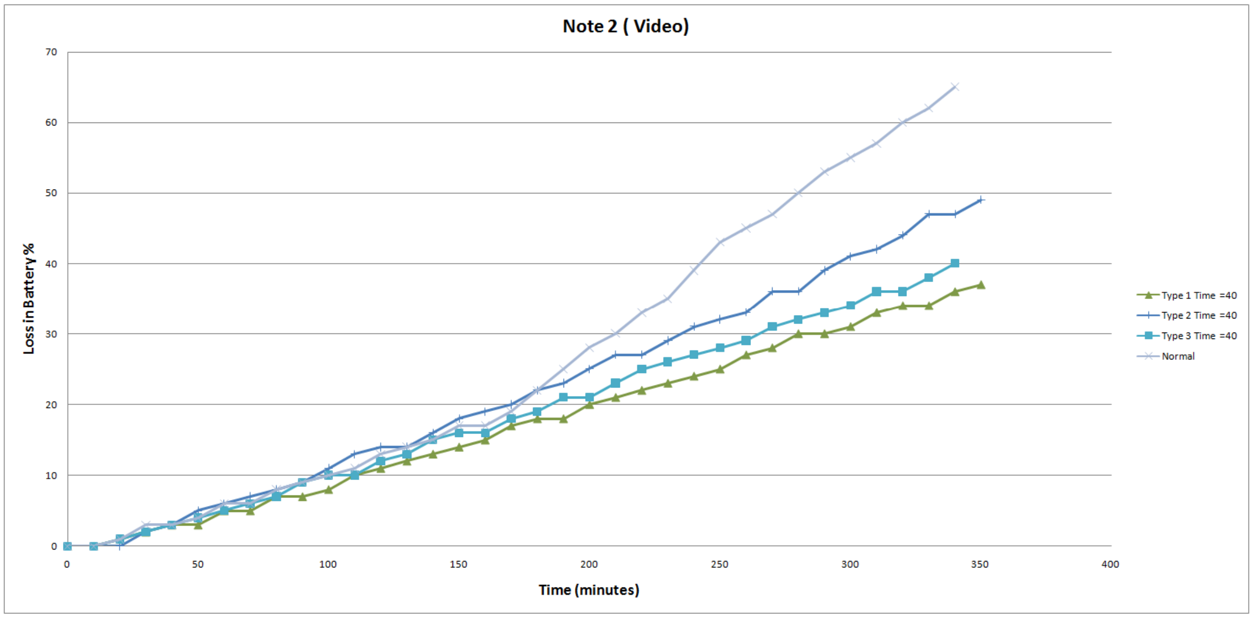
Use Case 10



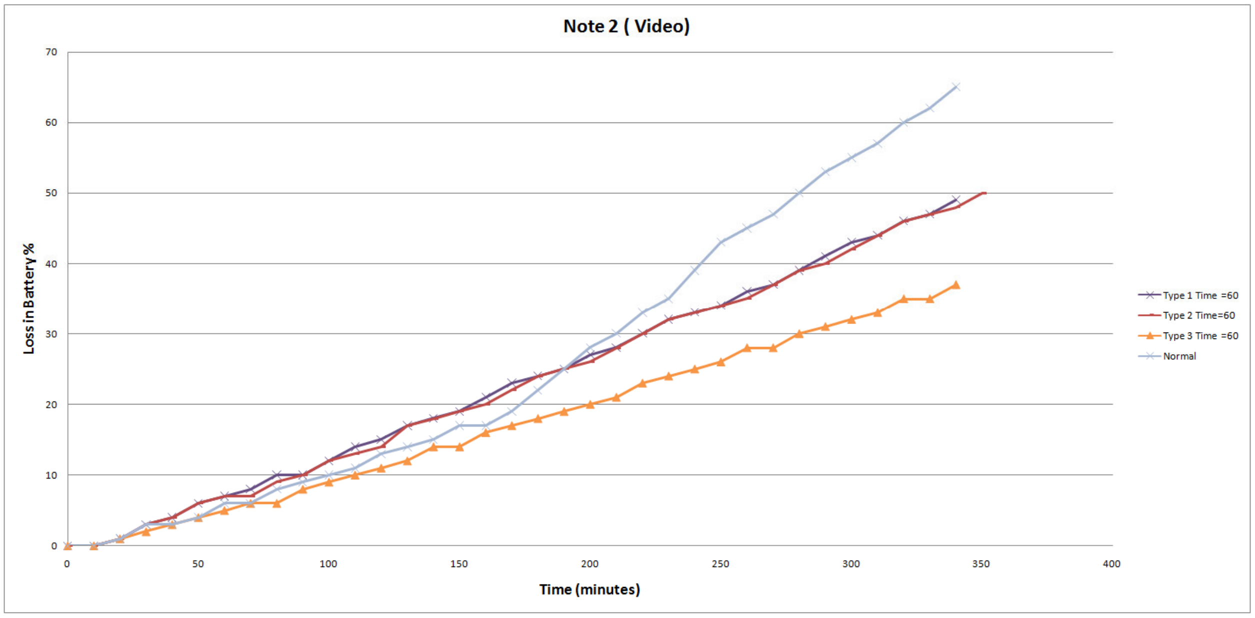
Use Case 11



Use Case 12

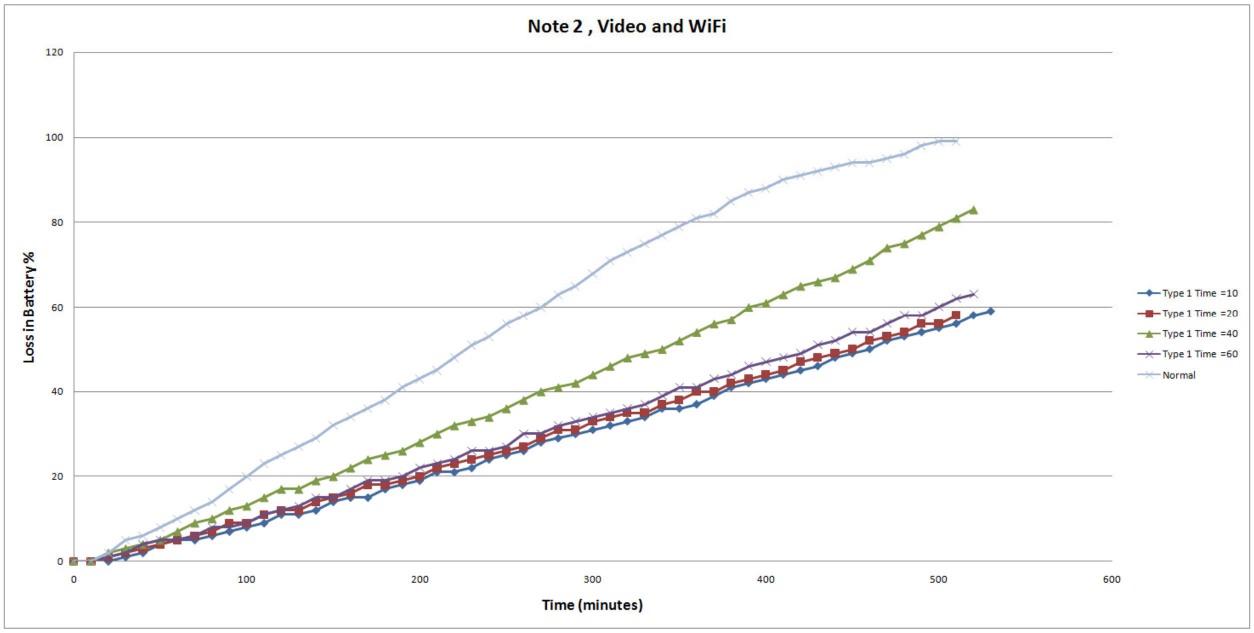


Use Case 13

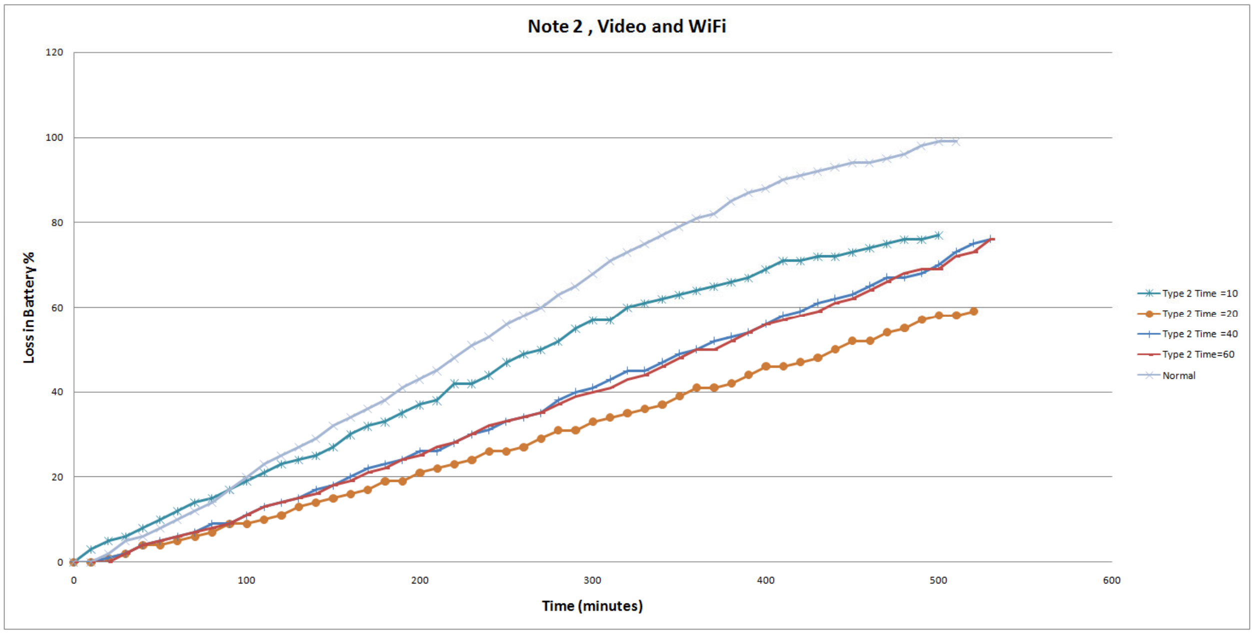


Use Case 14

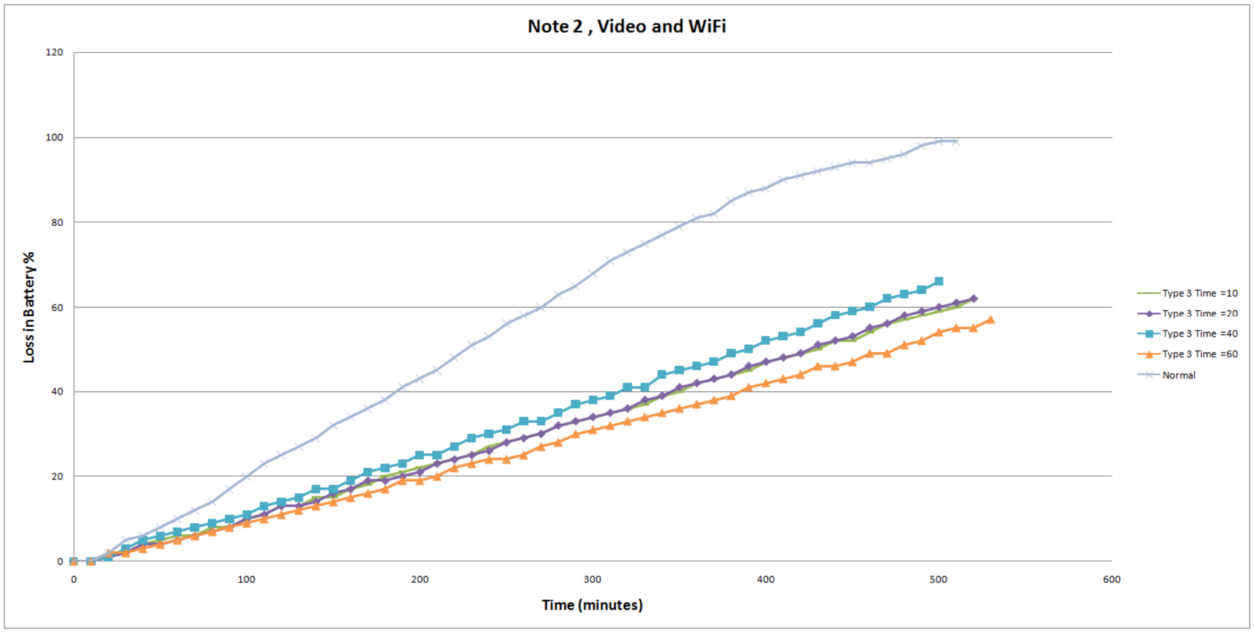
Case 1.3 - Class of video+Wi-Fi application on Note 2



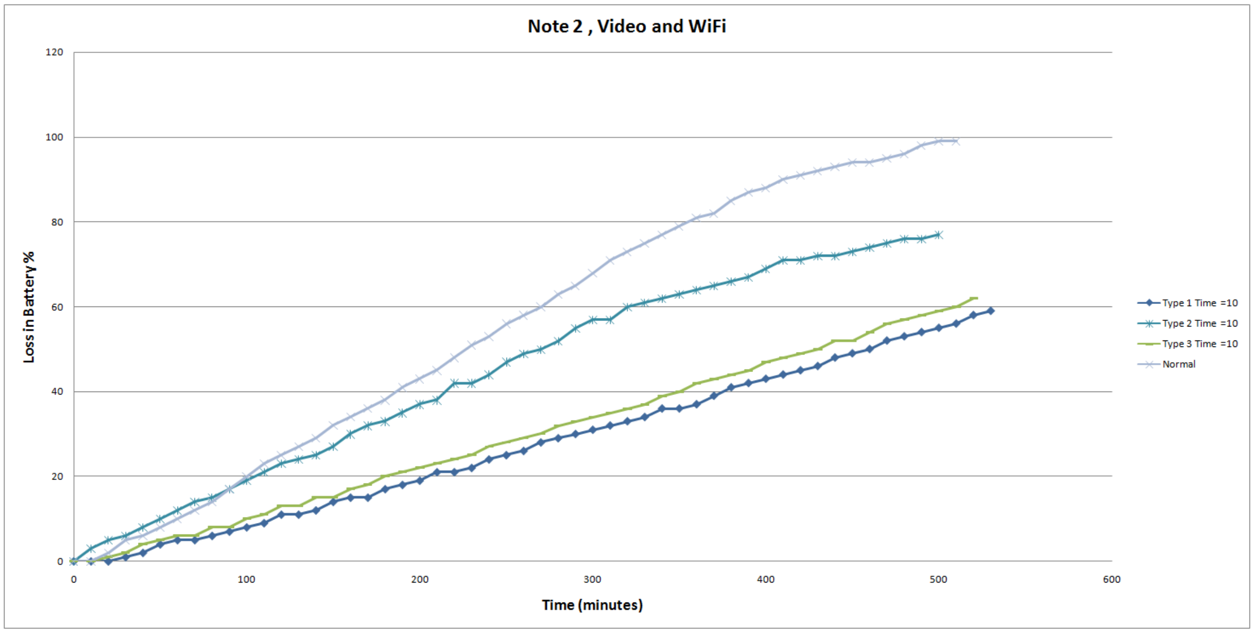
Use Case 15



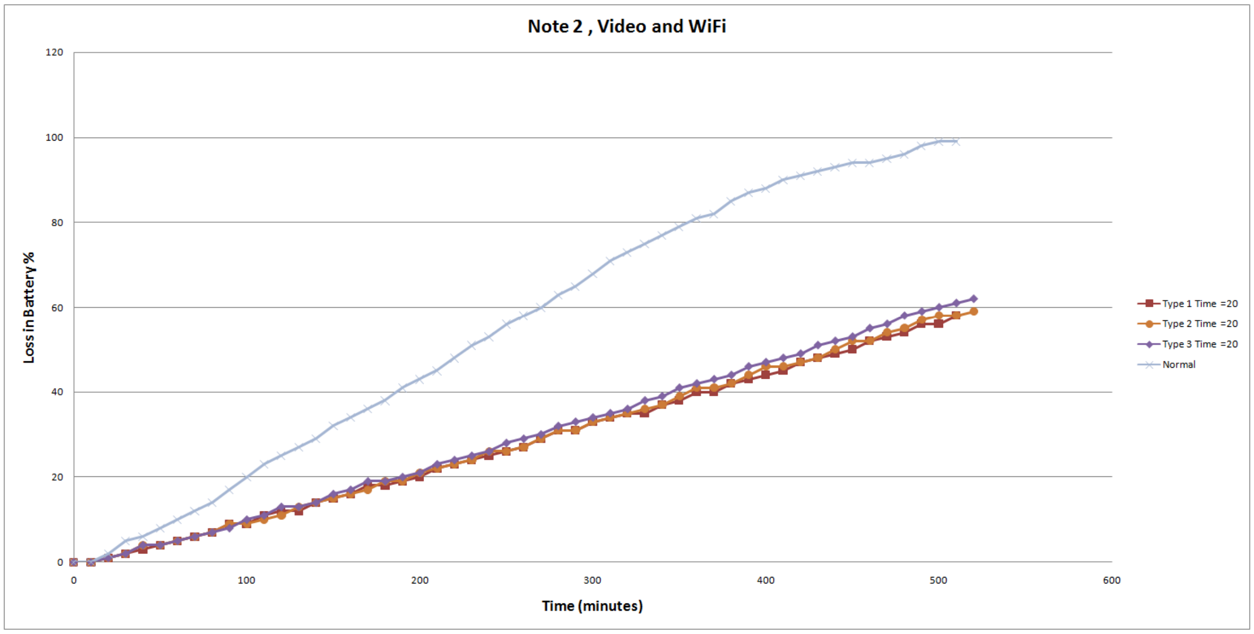
Use Case 16



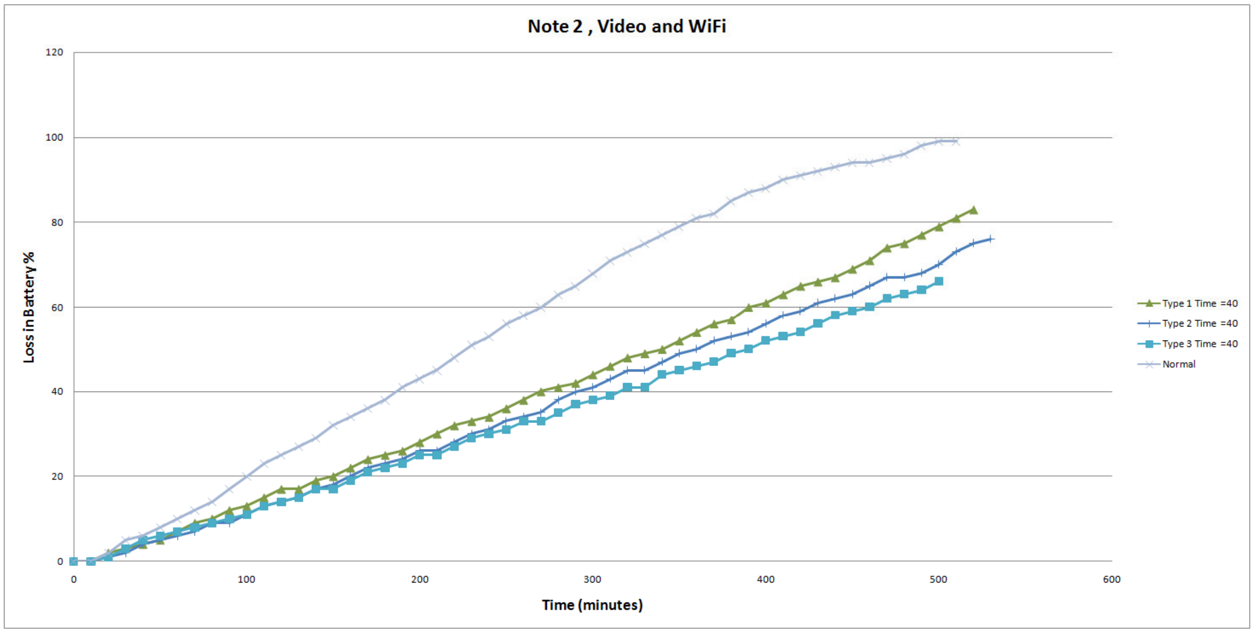
Use Case 17


Note 2 , Video and WiFi
Loss in Battery %
Time (minutes)
120
100
80
60
40
20
0
0
100
200
300
400
500
600
Type 1 Time =10
Type 2 Time =10
Type 3 Time =10
Normal

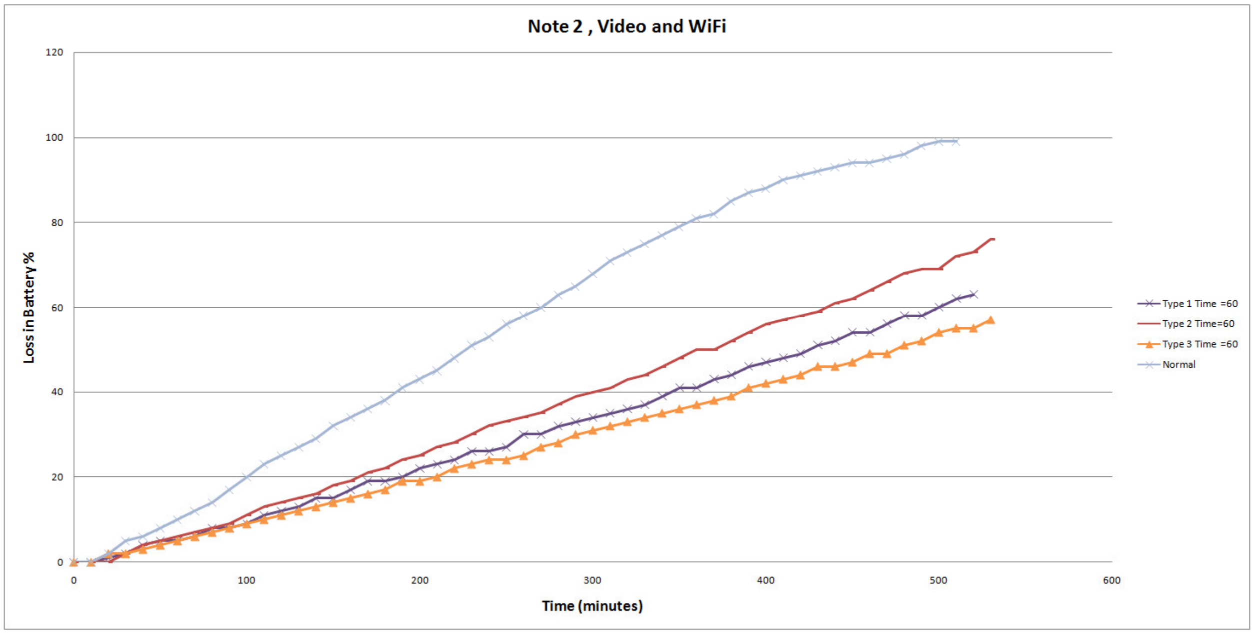
Use Case 18



Use Case 19



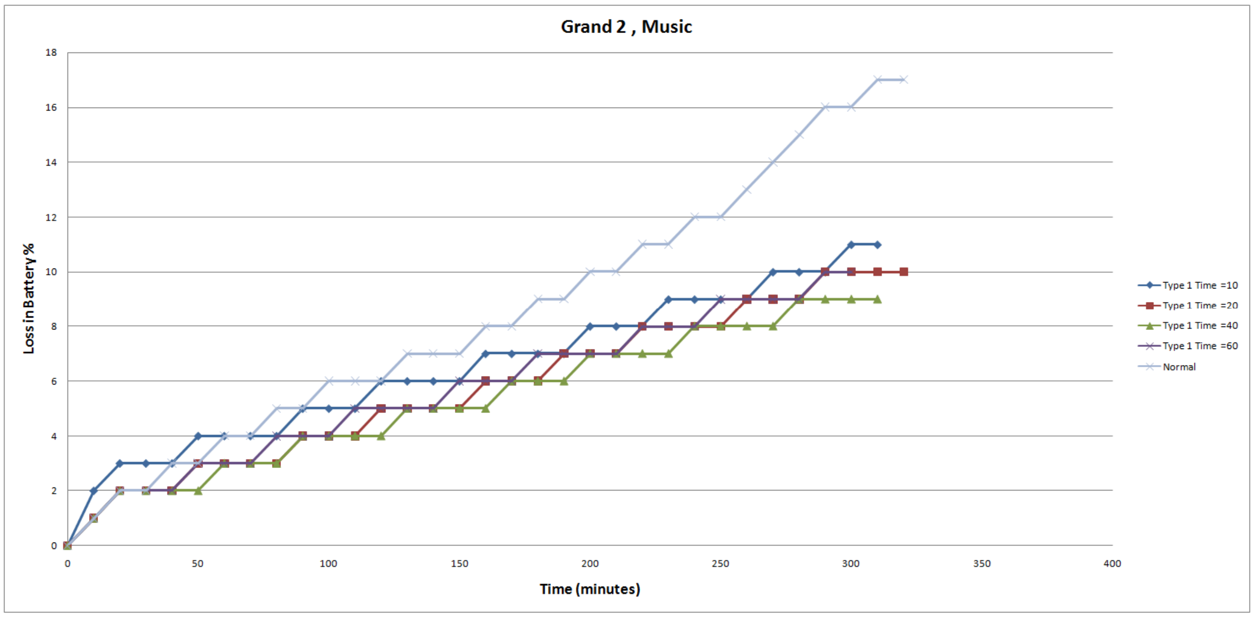
Use Case 20


Note 2 , Video and WiFi
Loss in Battery %
Time (minutes)
0
20
40
60
80
100
120
0
100
200
300
400
500
600
Type 1 Time =60
Type 2 Time=60
Type 3 Time =60
Normal

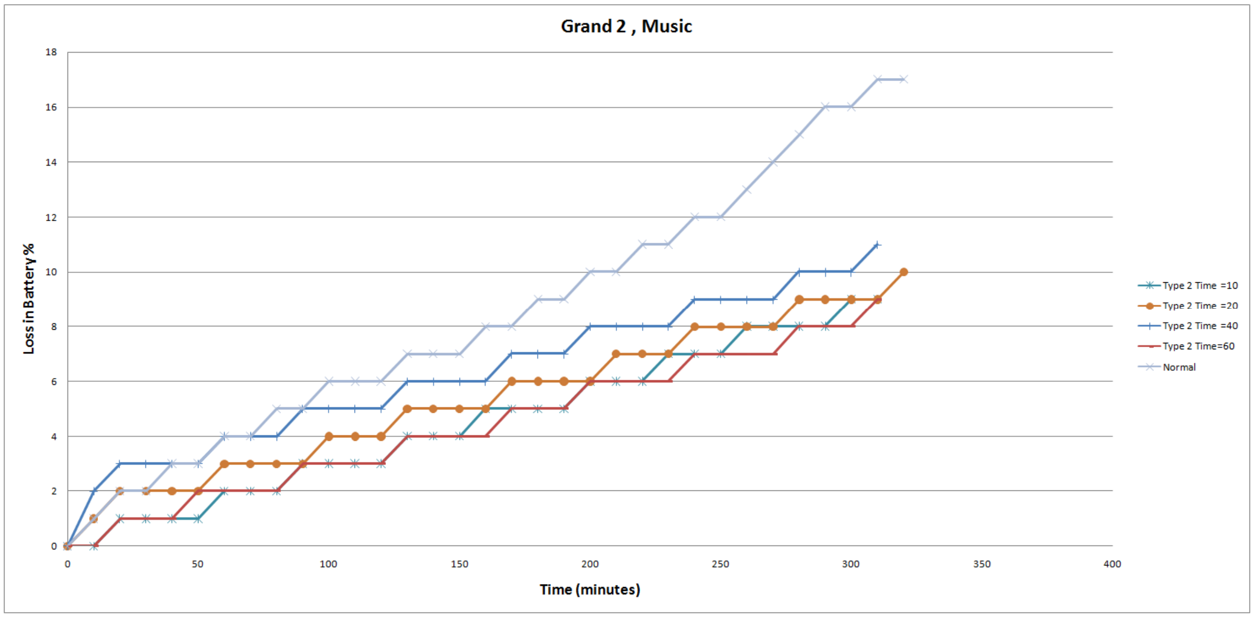
Use Case 21

Case 2 - Galaxy Grand2

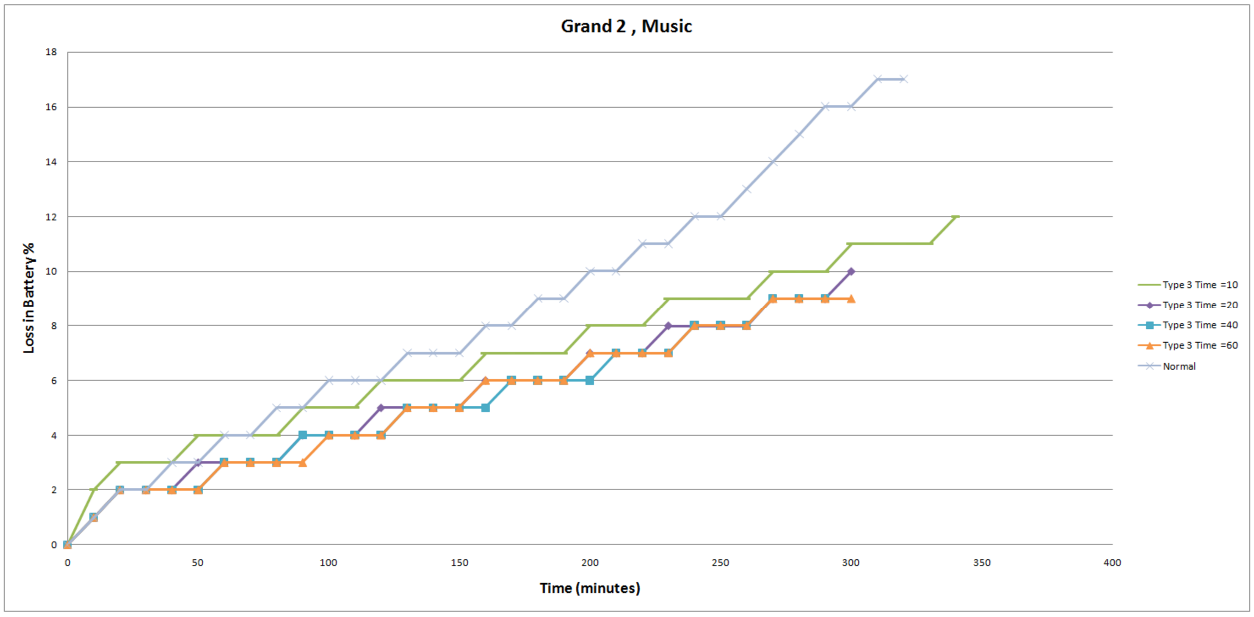
Case 2.1 - Class of music application on Grand 2



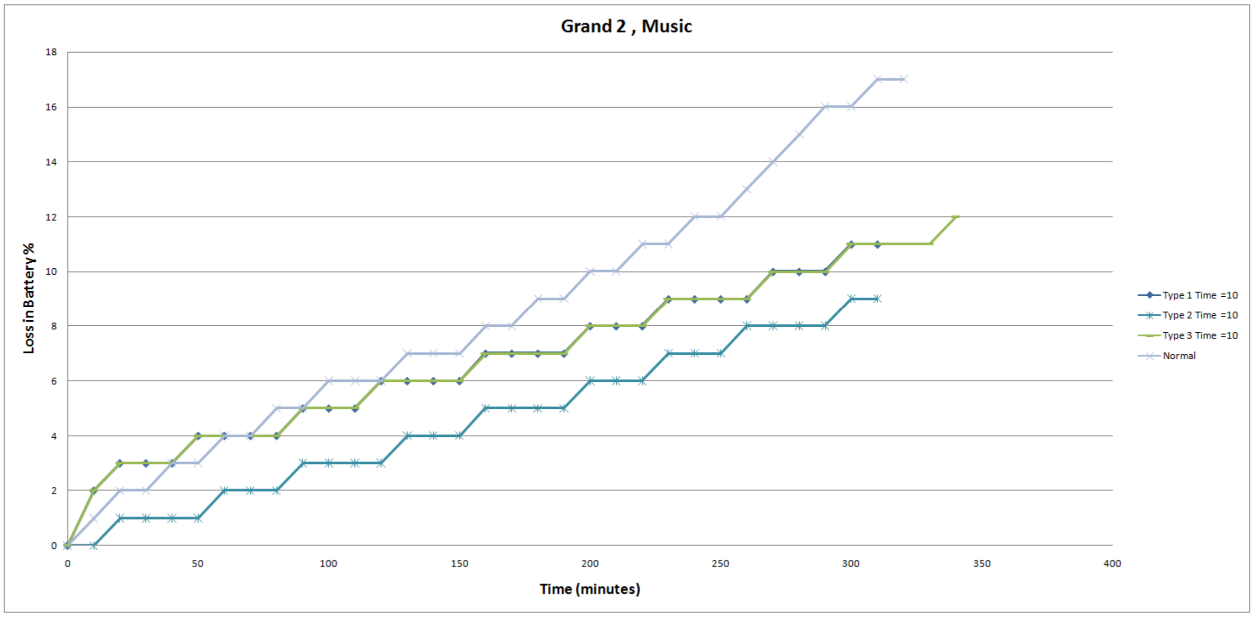
Use Case 22



Use Case 23

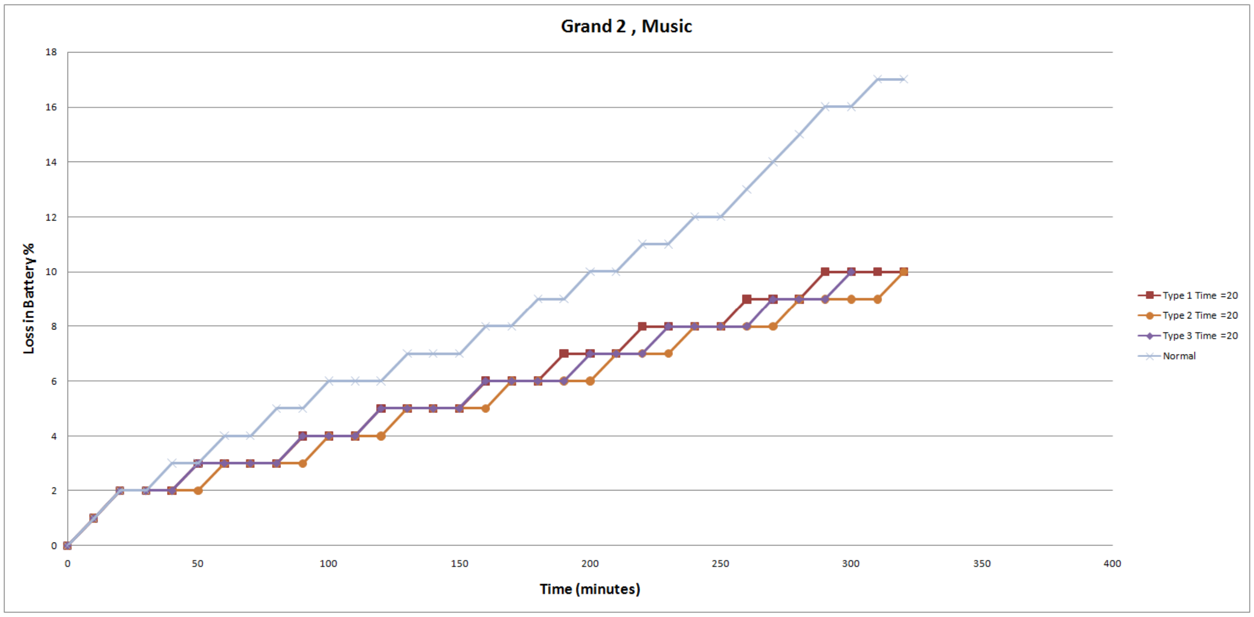


Use Case 24

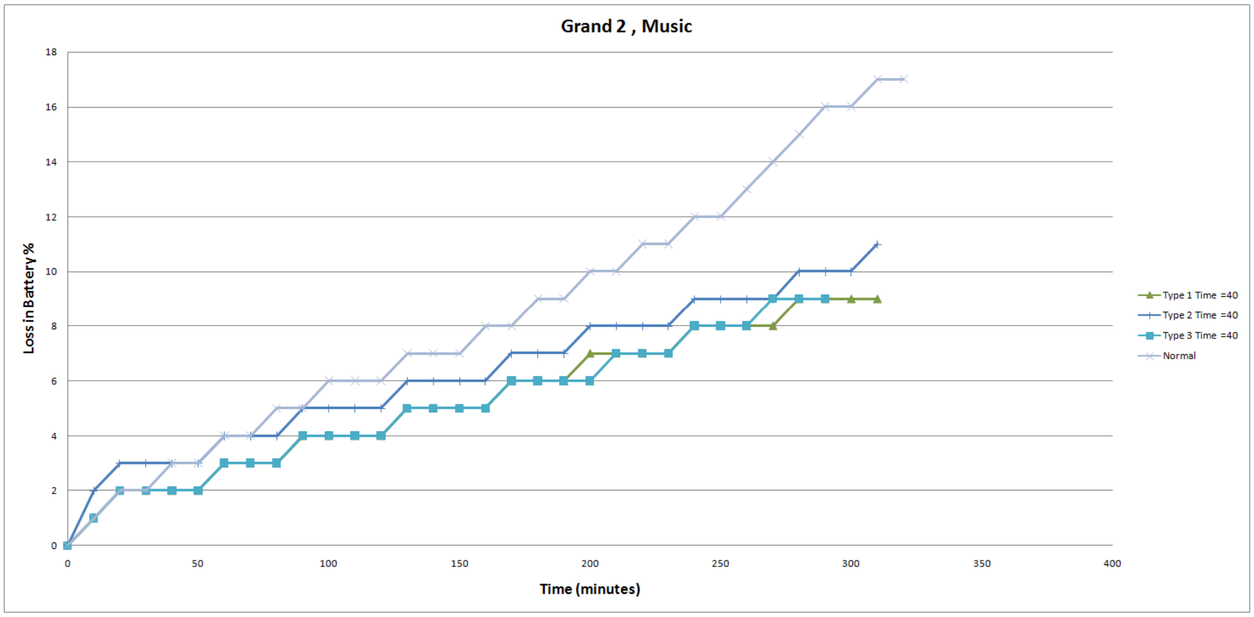


Grand 2 , Music
Loss in Battery %
Time (minutes)
0
2
4
6
8
10
12
14
16
18
0
50
100
150
200
250
300
350
400
Type 1 Time =10
Type 2 Time =10
Type 3 Time =10
Normal

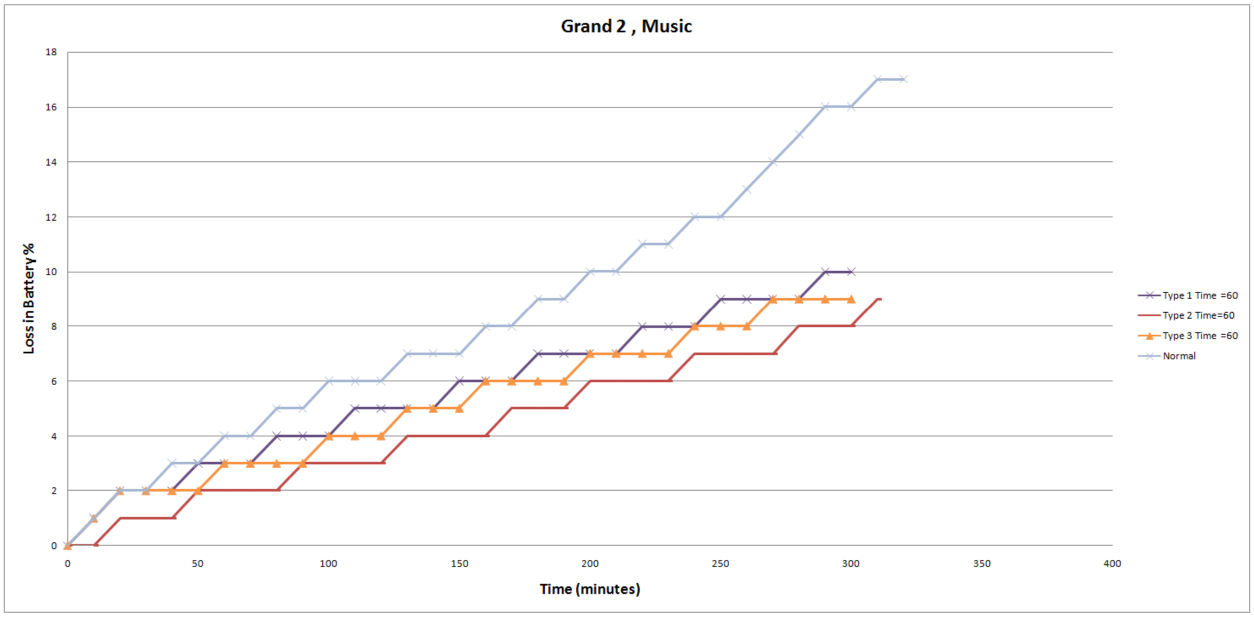
Use Case 25



Use Case 26

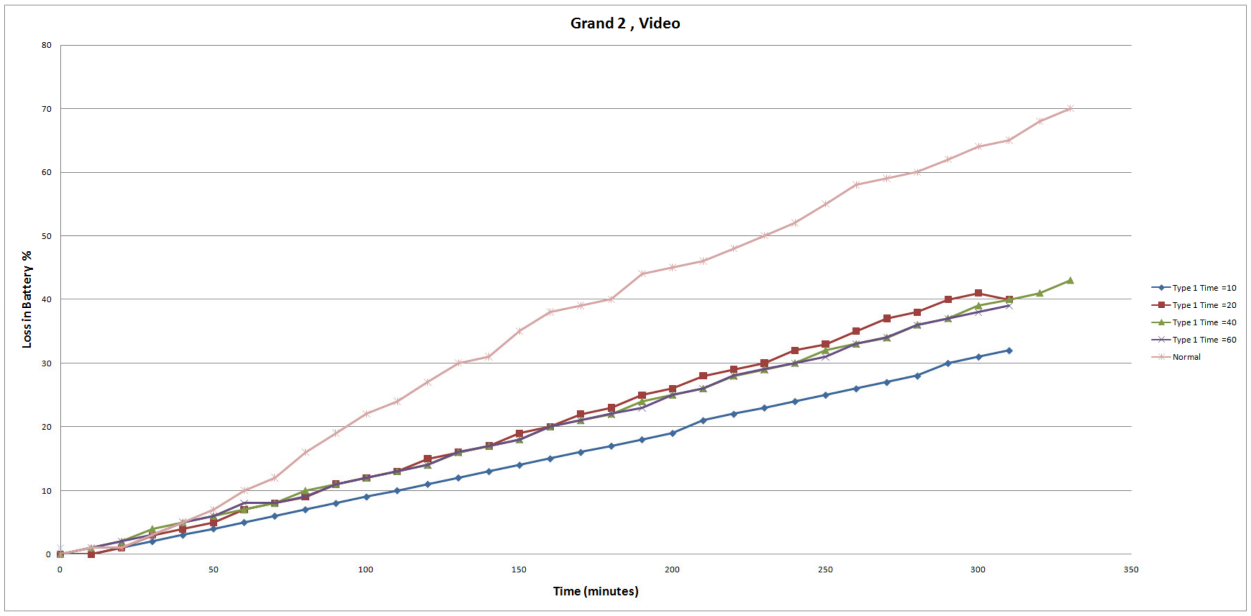


Use Case 27

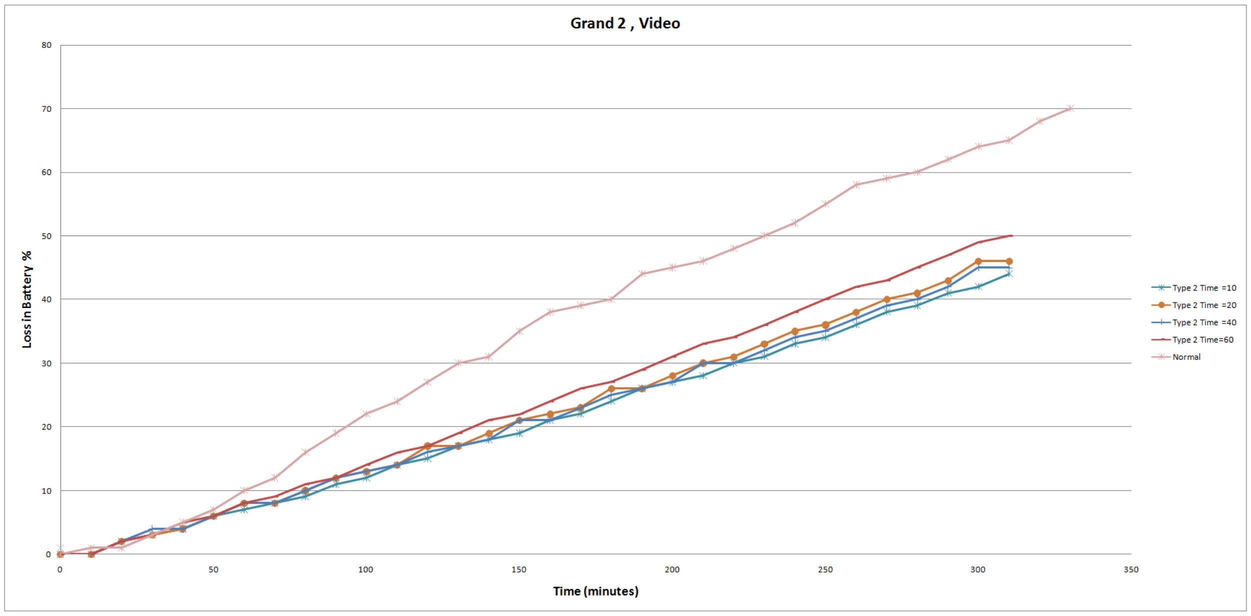


Use Case 28

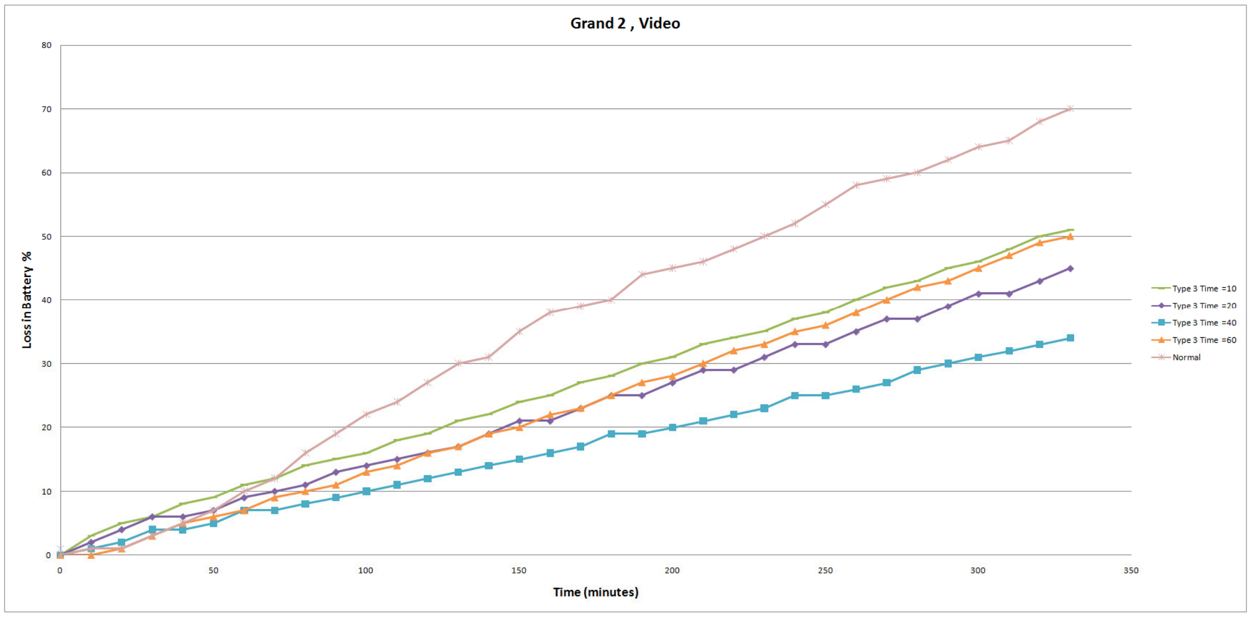
Case 2.2 - Class of video application on Galaxy Grand2



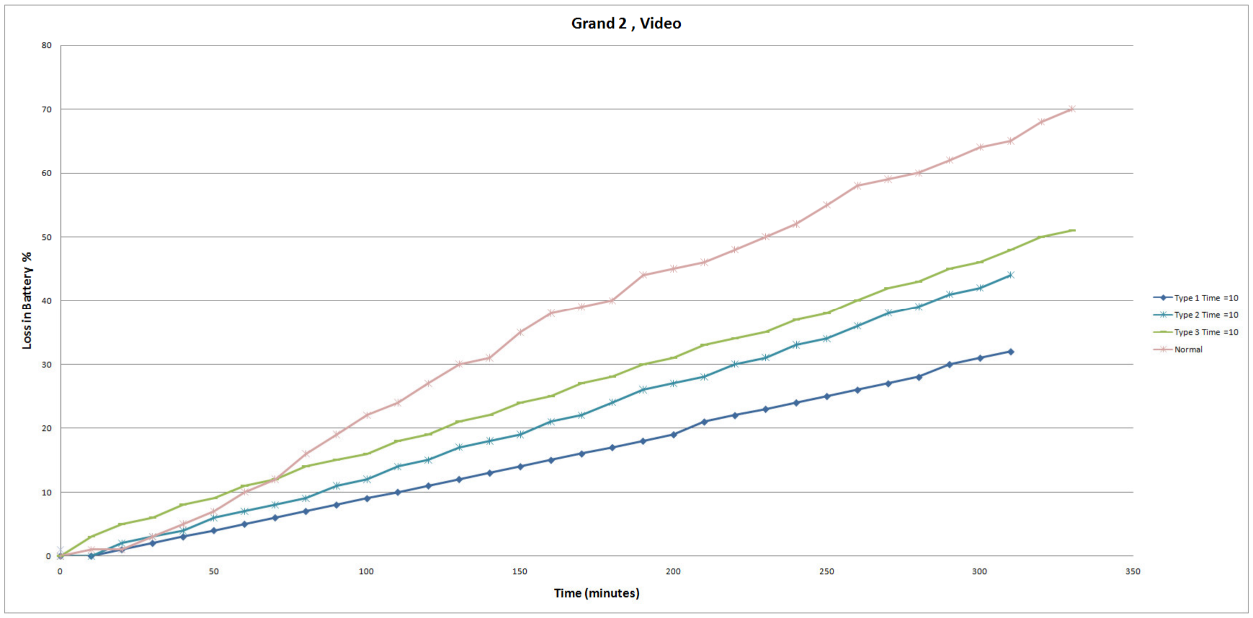
Use Case 29



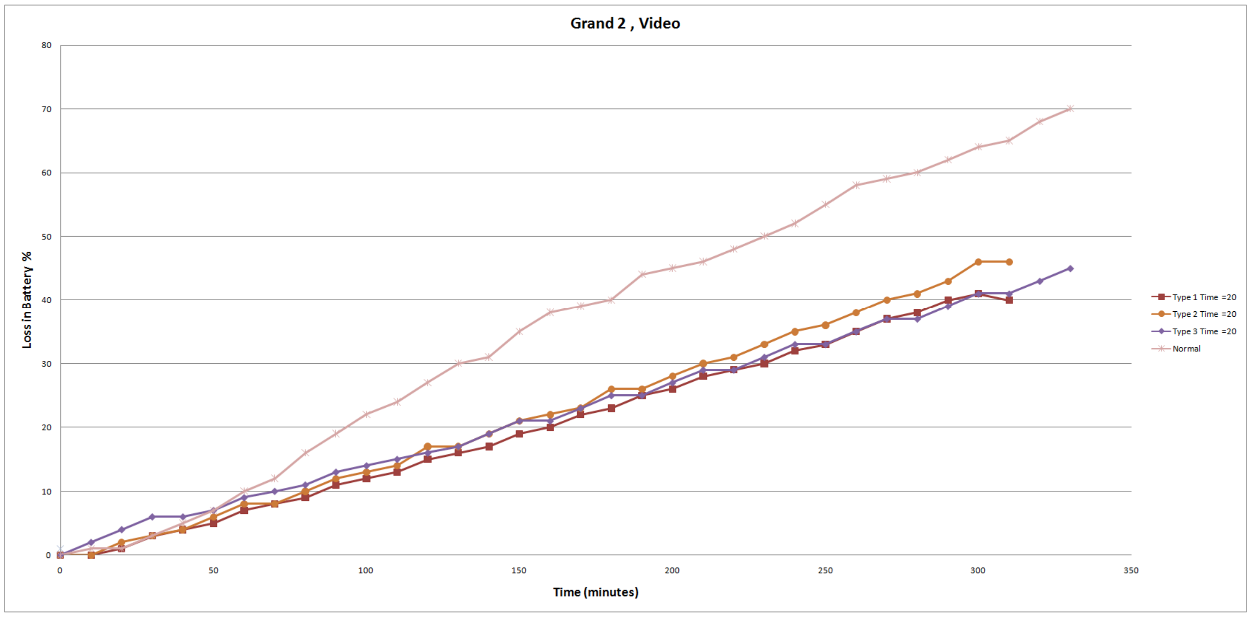
Use Case 30



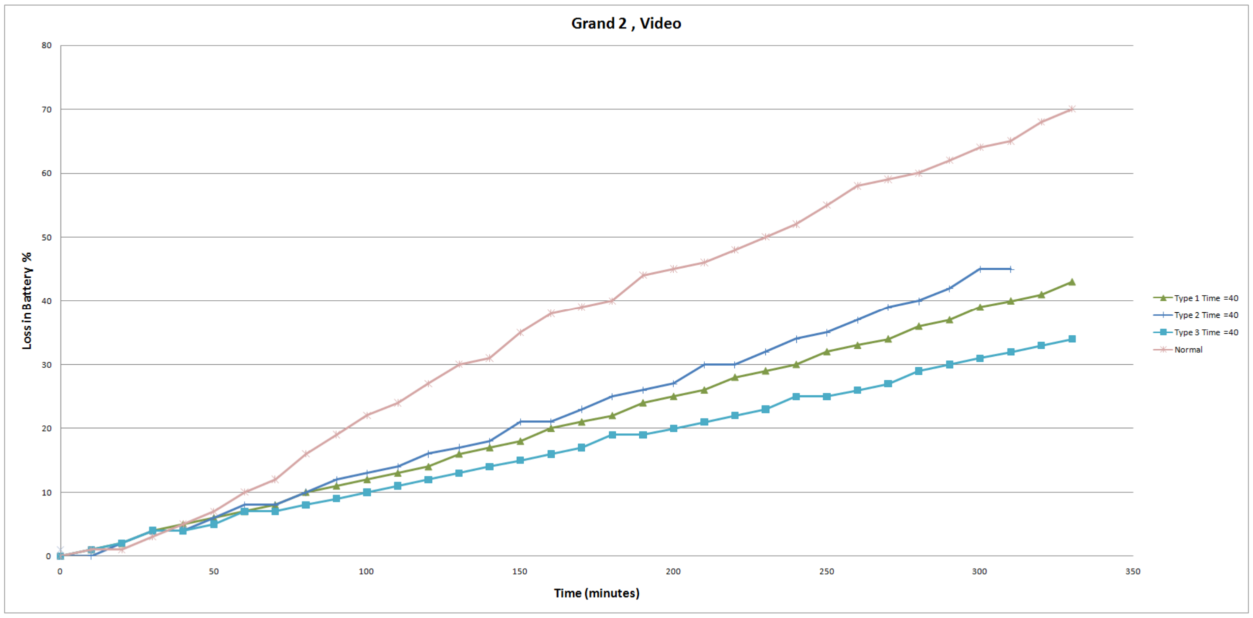
Use Case 31



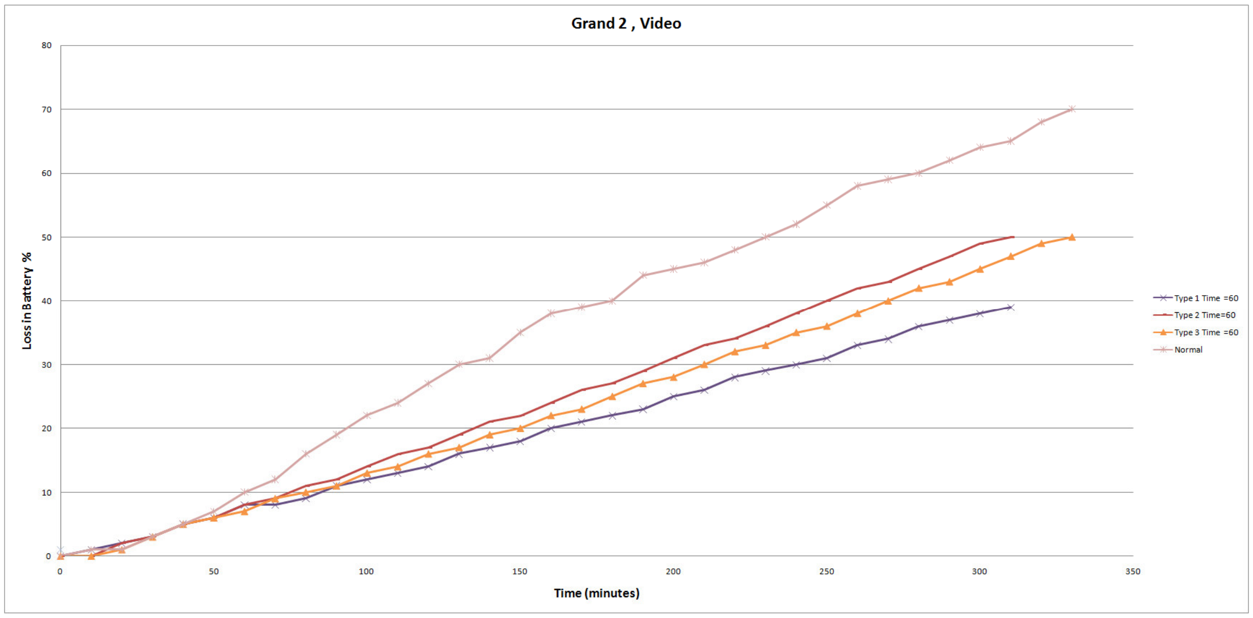
Use Case 32



Use Case 33

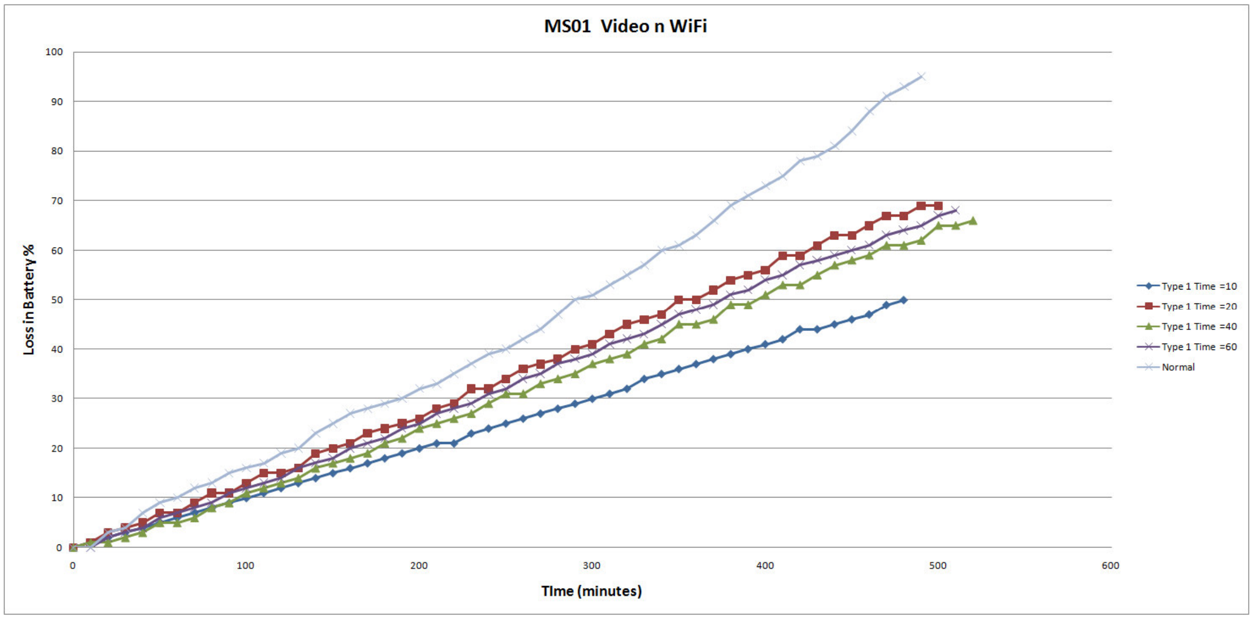


Use Case 34

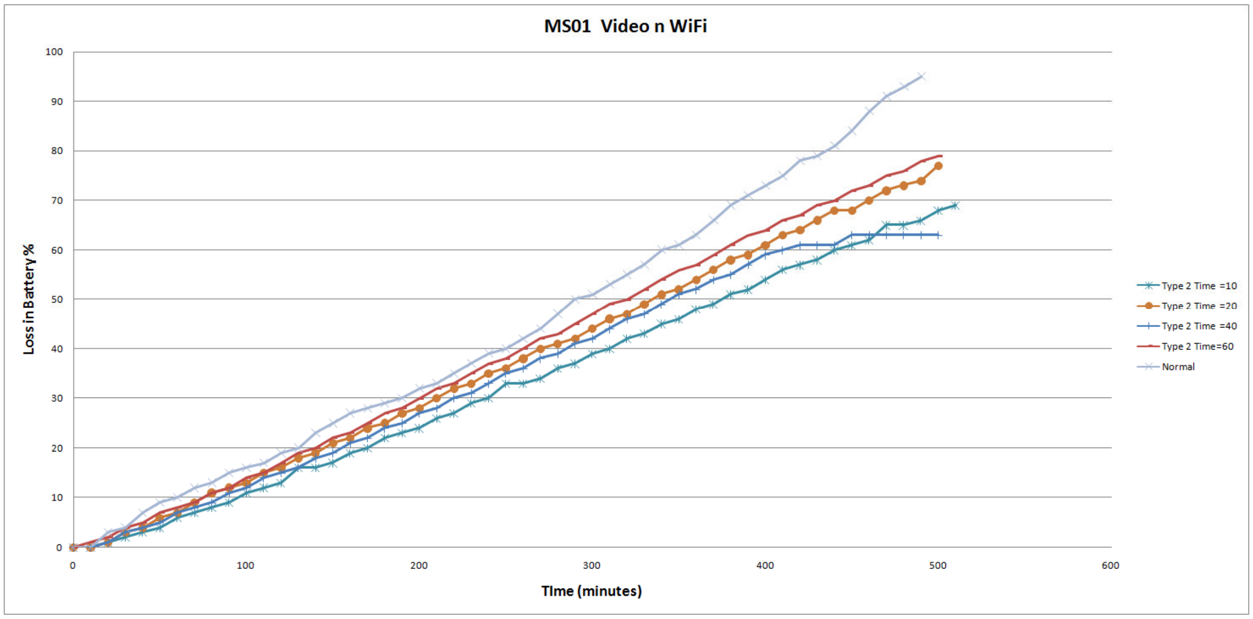


Use Case 35

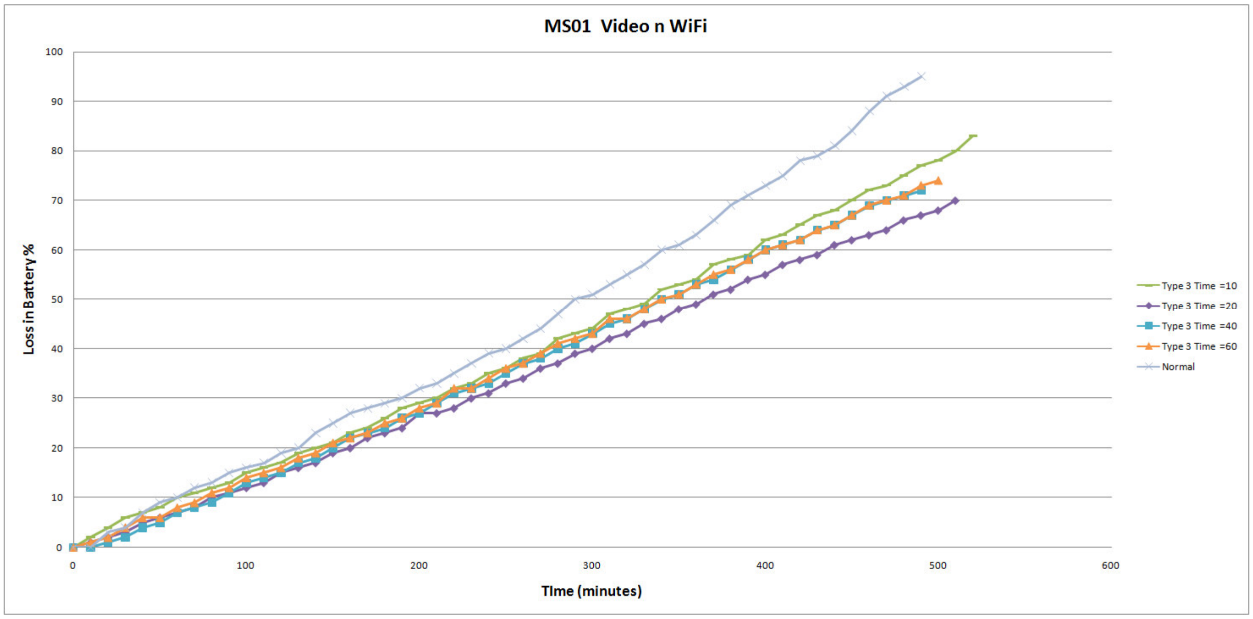
Case 2.3 - Class of video+Wi-Fi application on Galaxy Grand2



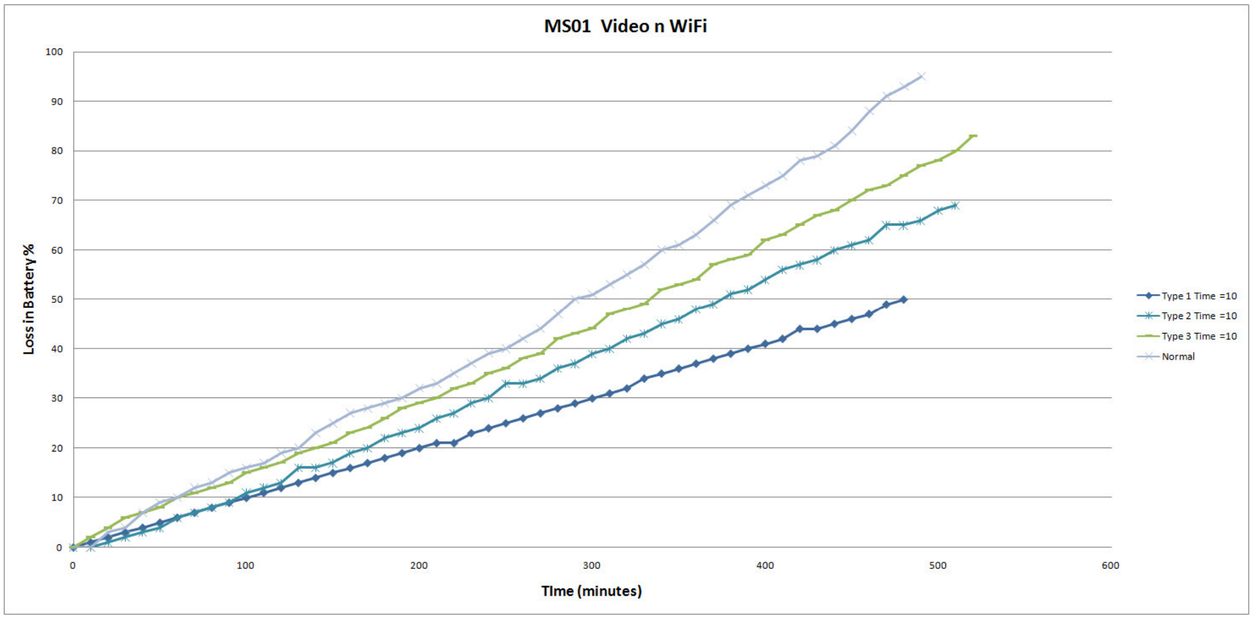
Use Case 36



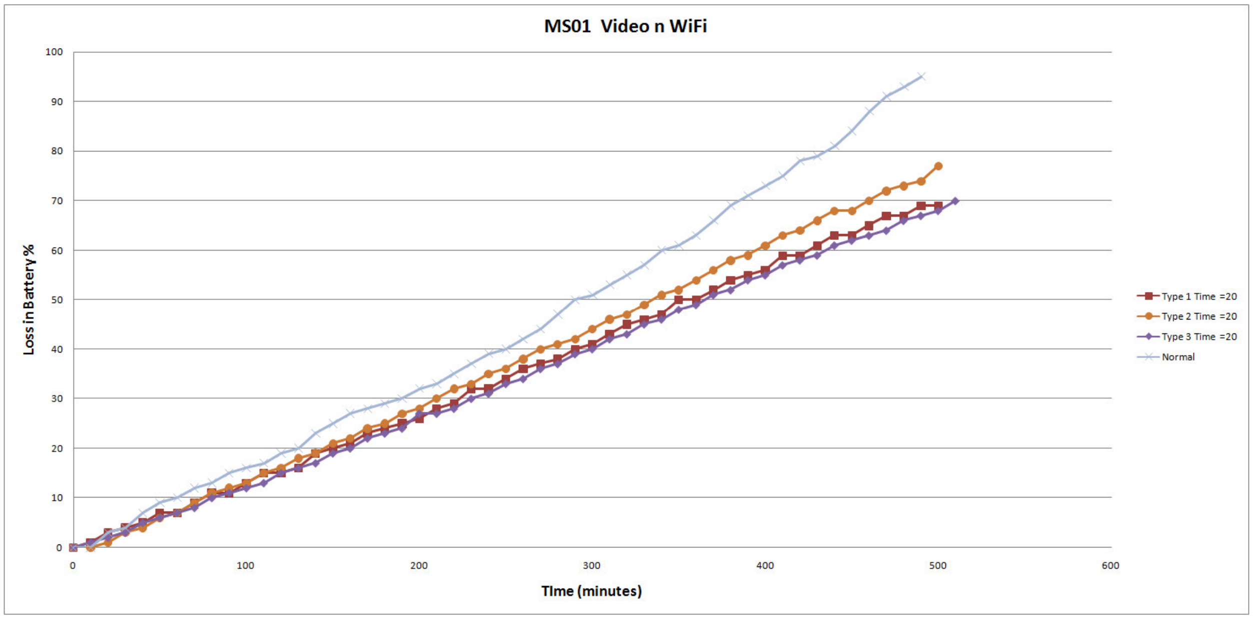
Use Case 37



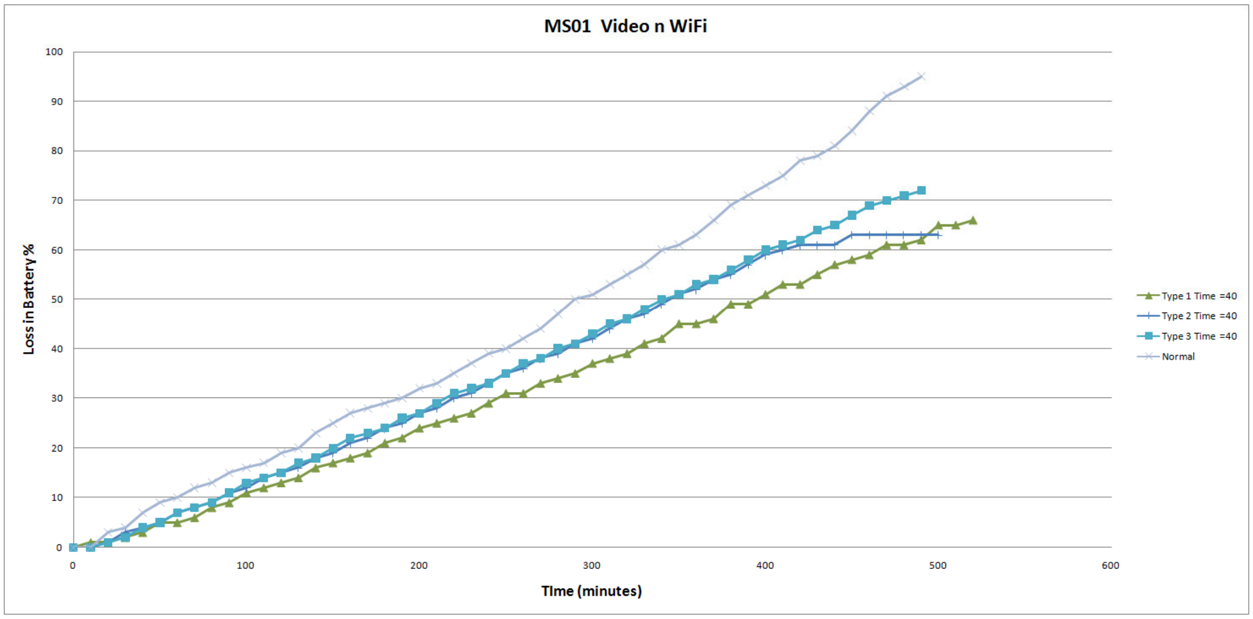
Use Case 38



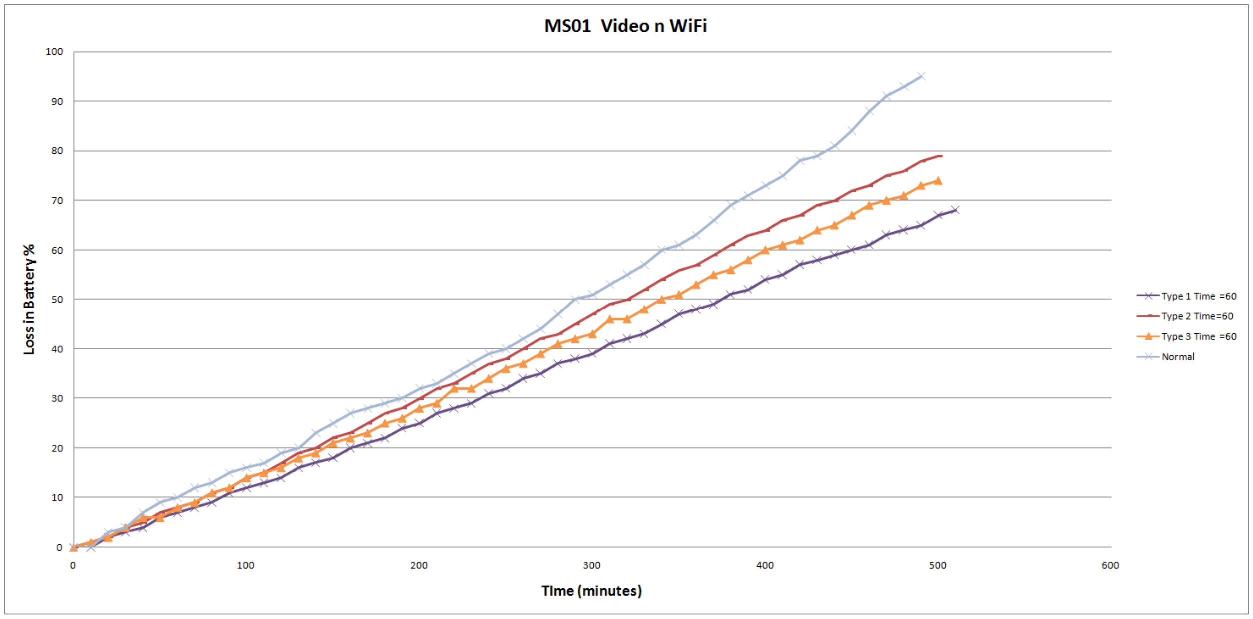
Use Case 39



Use Case 40



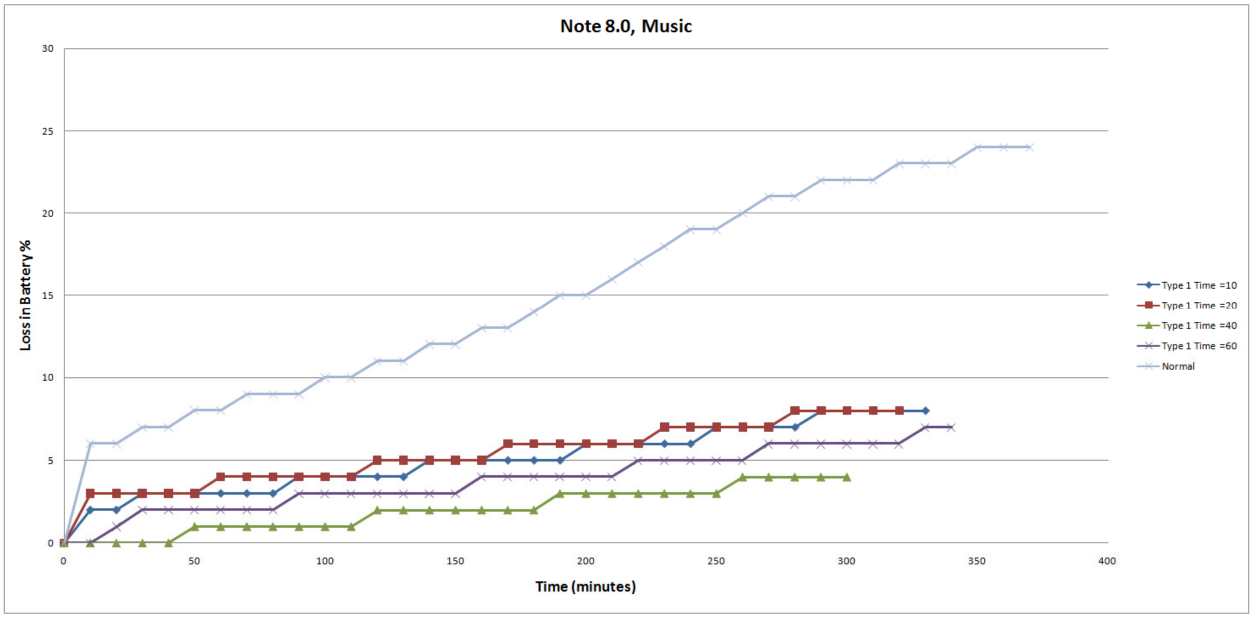
Use Case 41



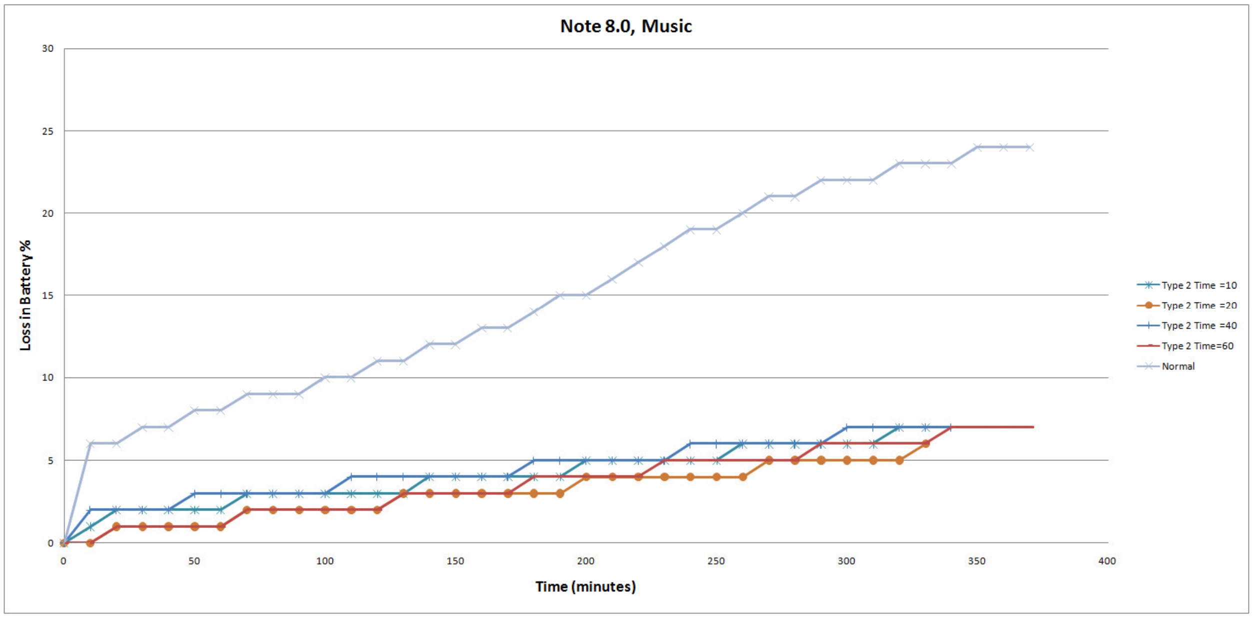
Use Case 42

Case 3 - Note8.0

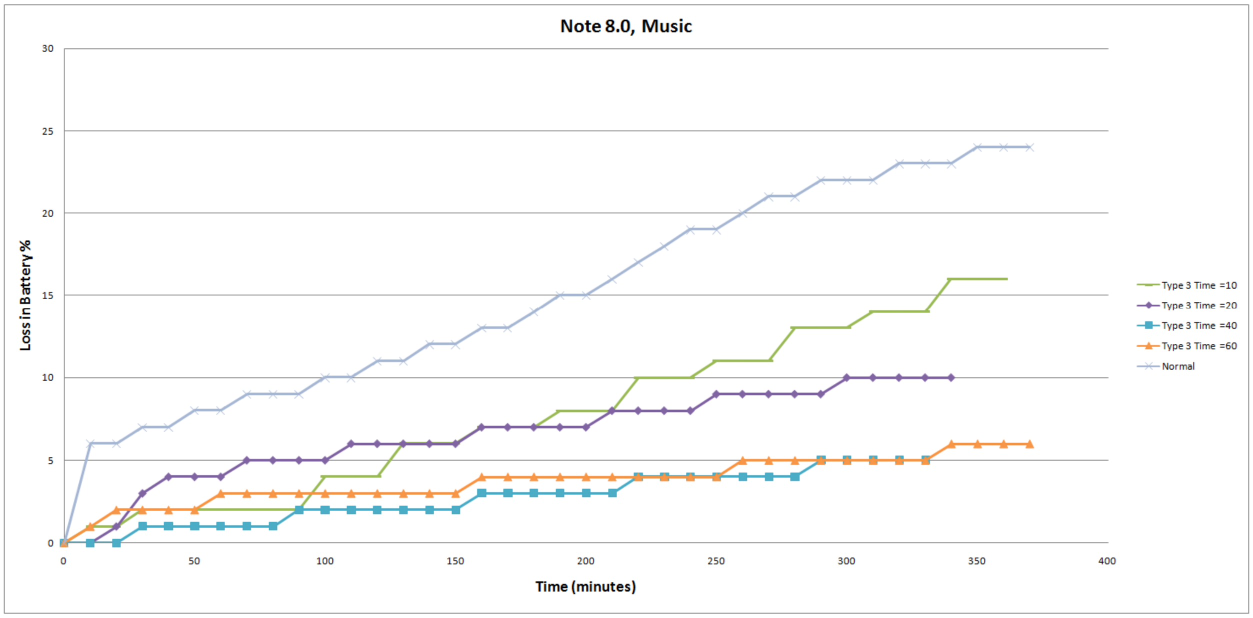
Case 3.1 - Class of music application on Note 8.0



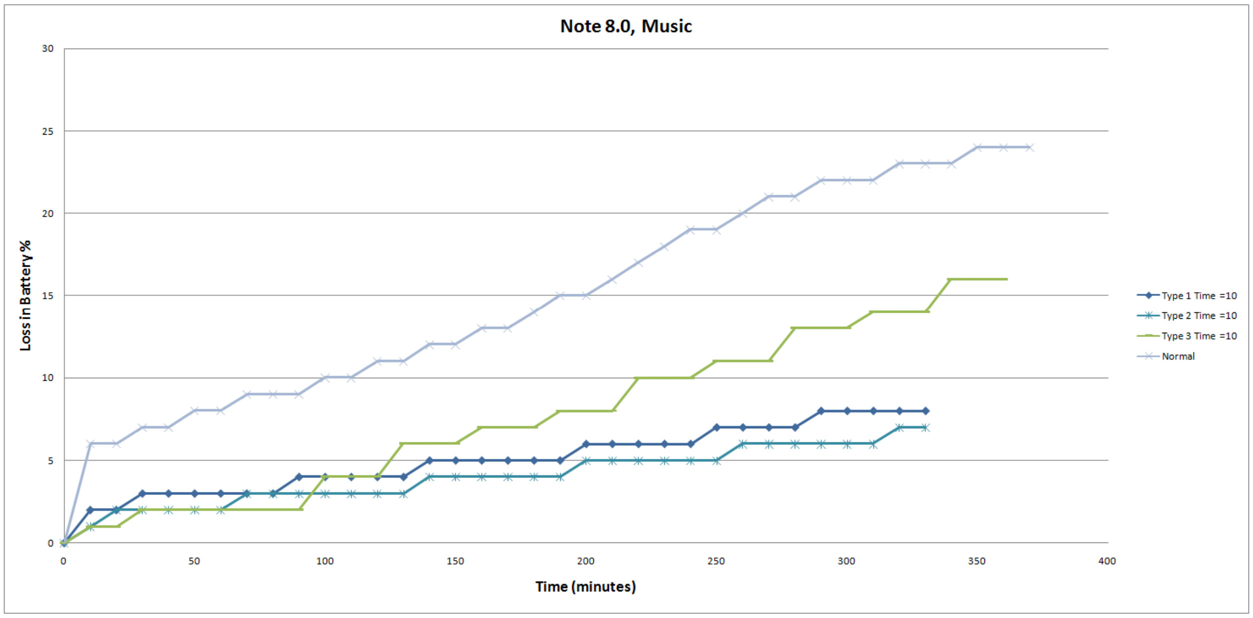
Use Case 43



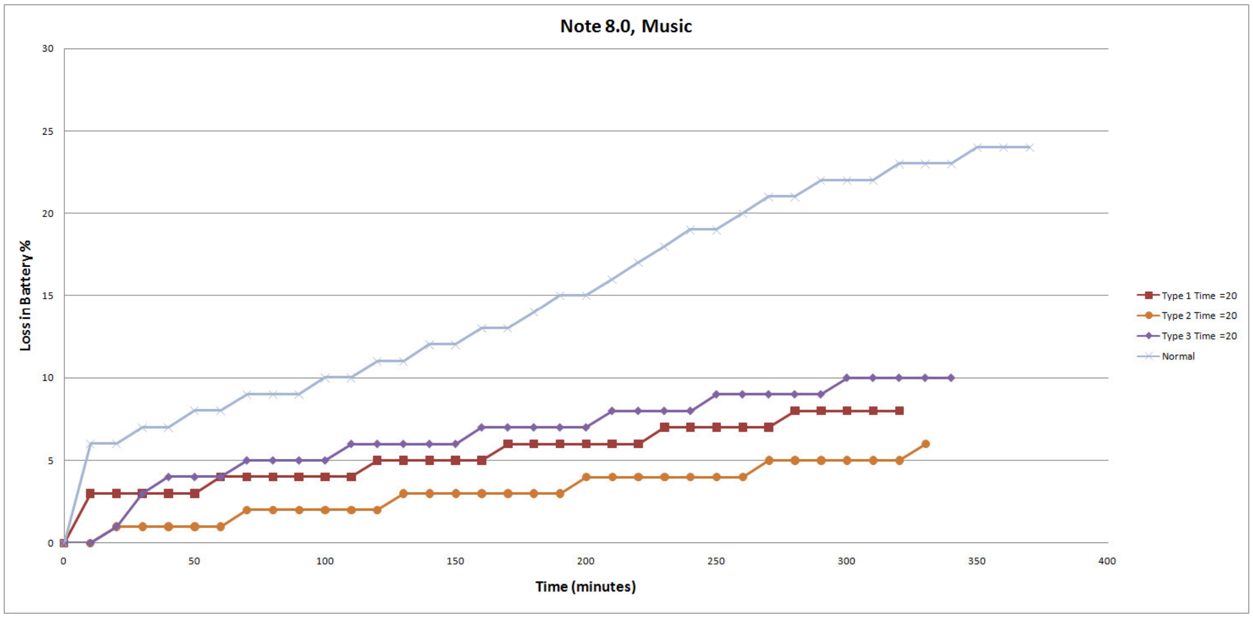
Use Case 44



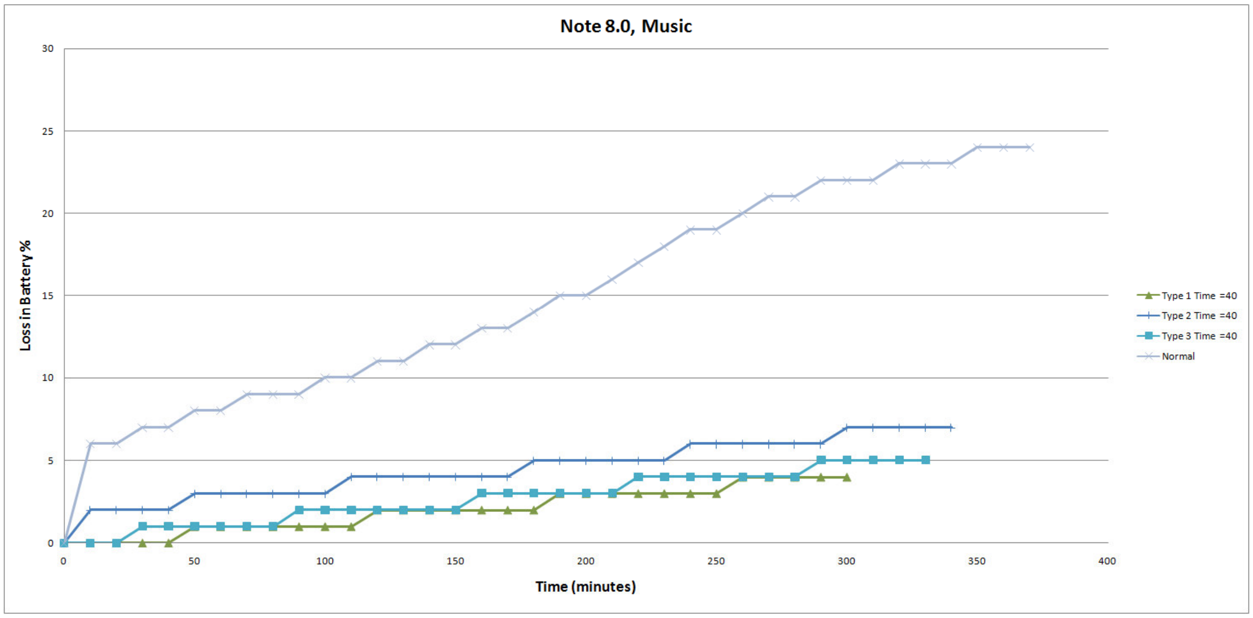
Use Case 45



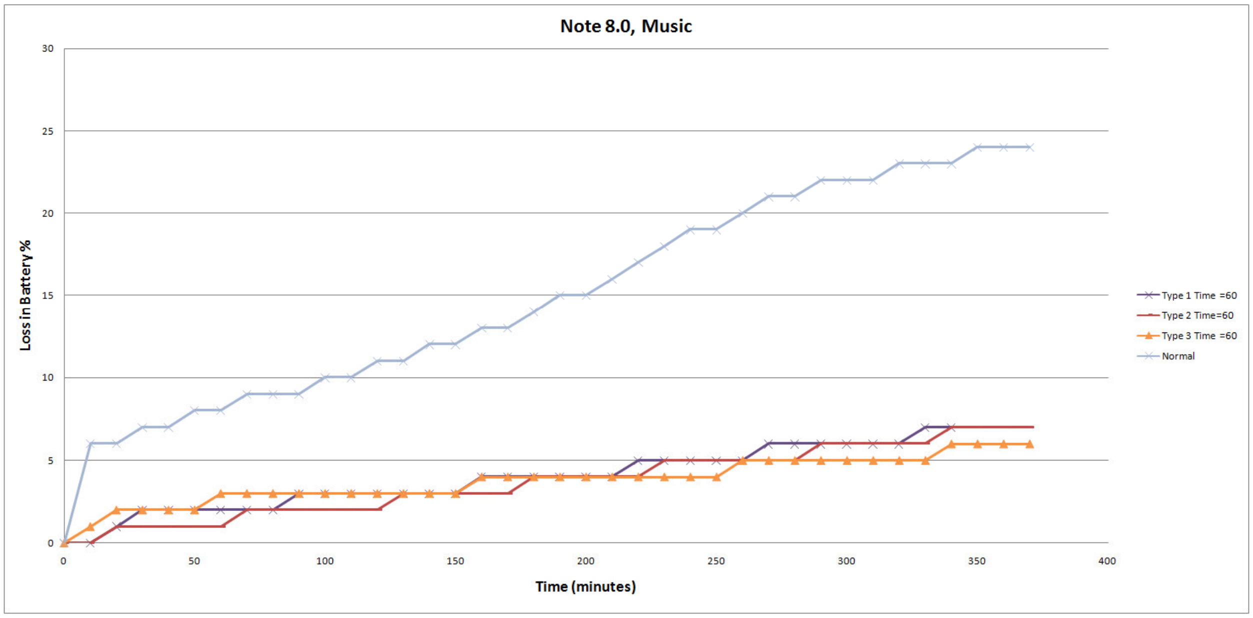
Use Case 46



Use Case 47

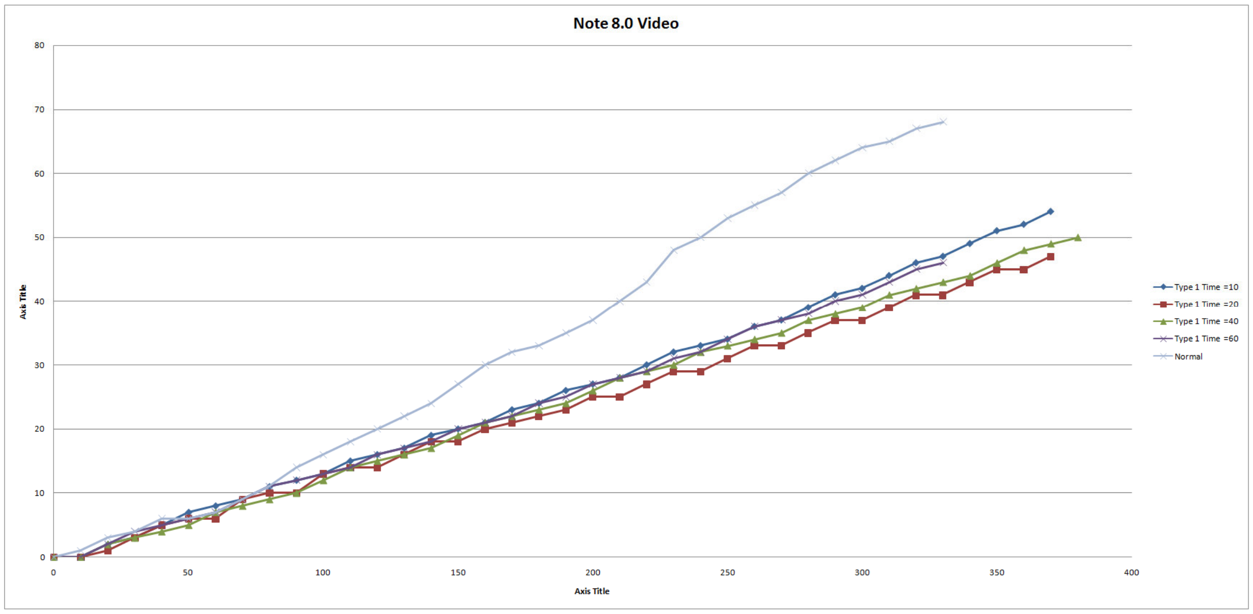


Use Case 48

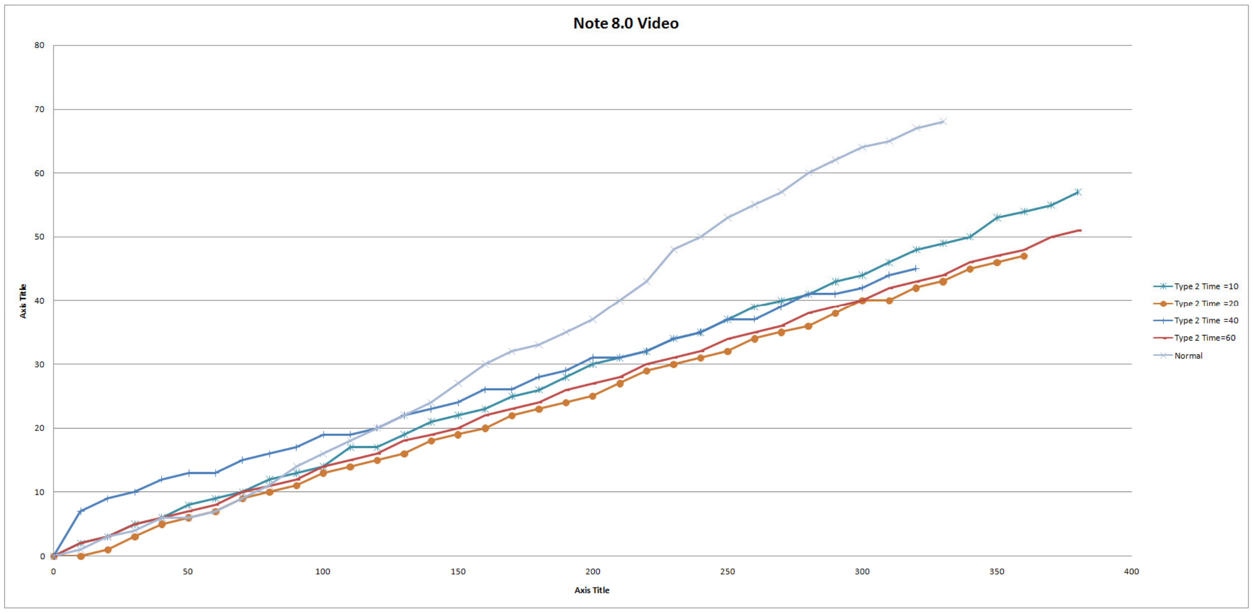


Use Case 49

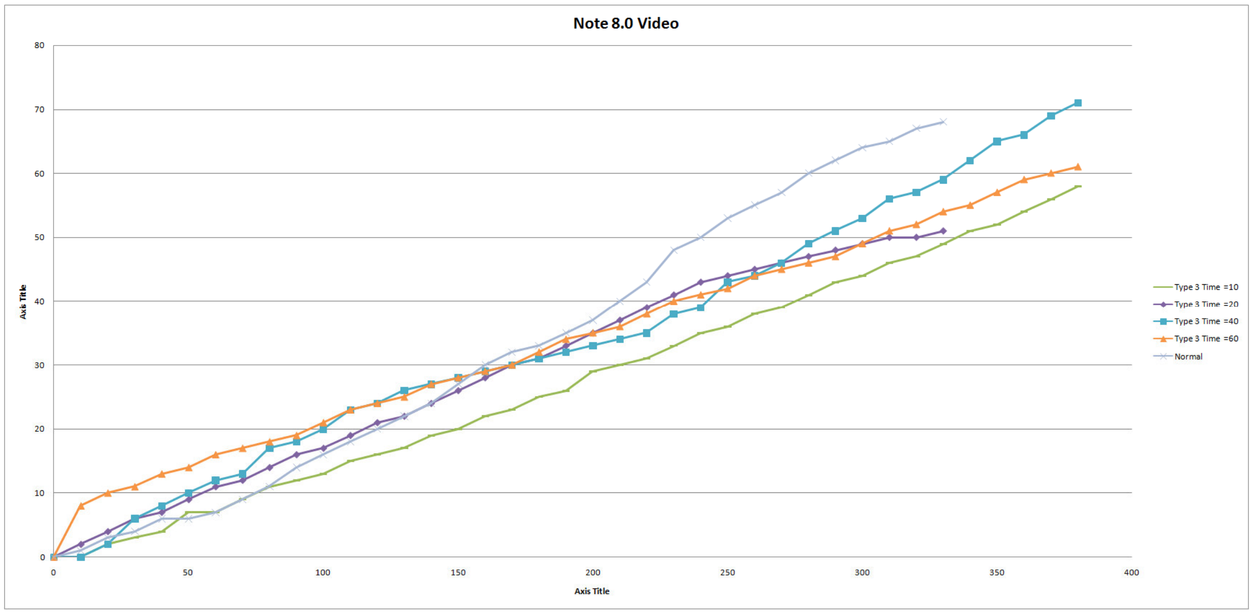
Case 3.2 - Class of video application on Note 8.0



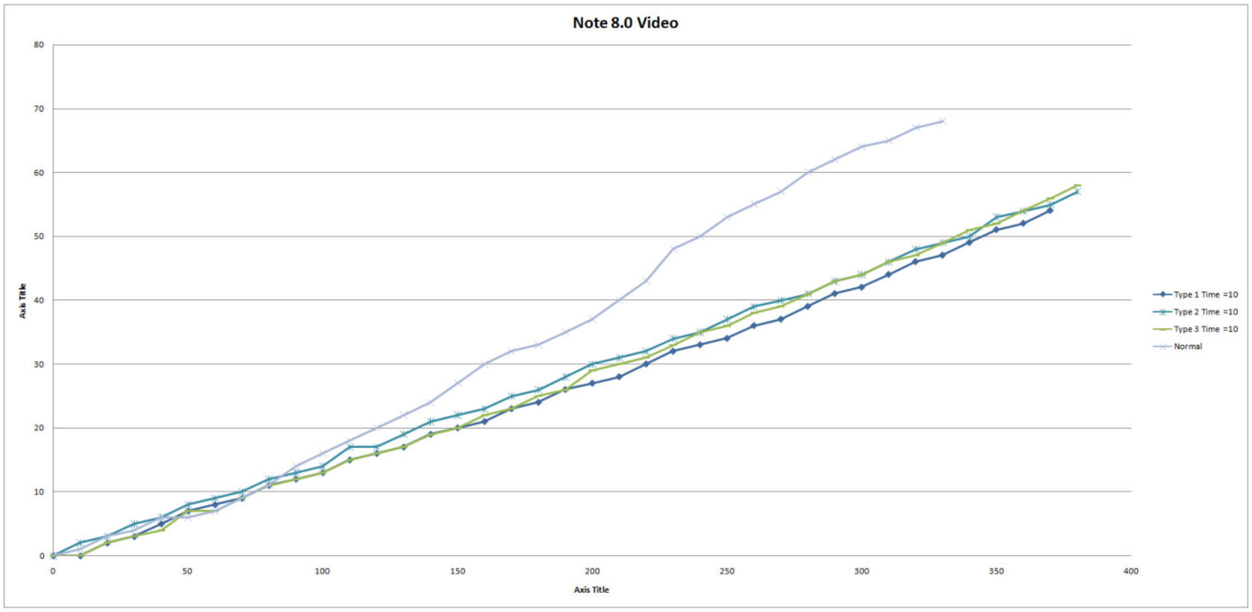
Use Case 50



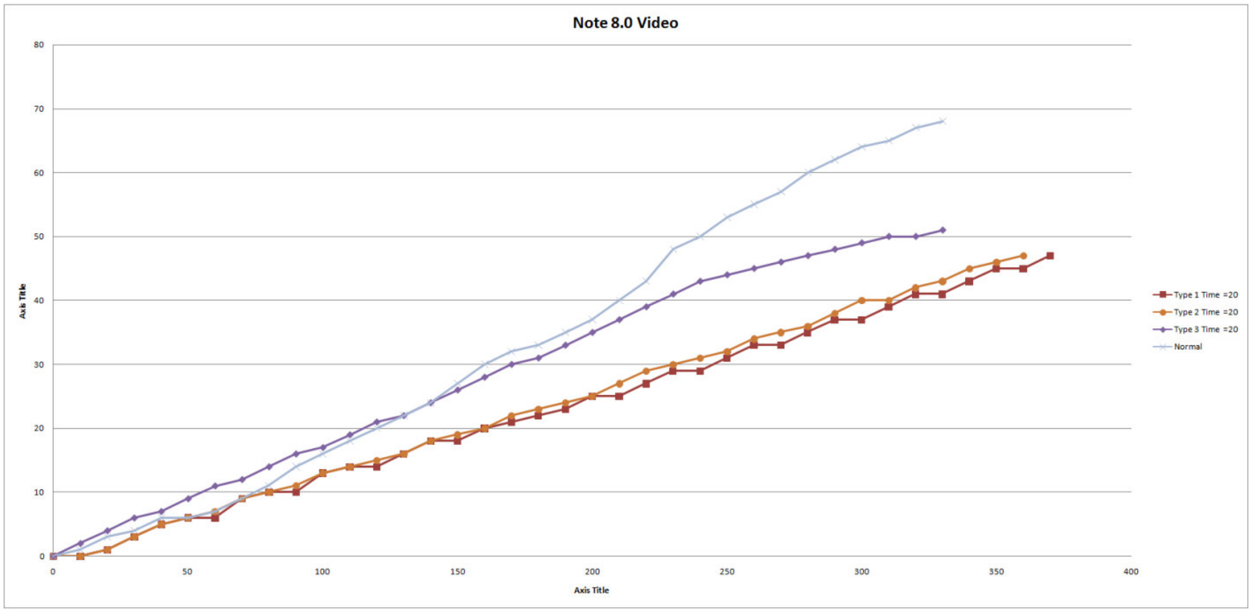
Use Case 51



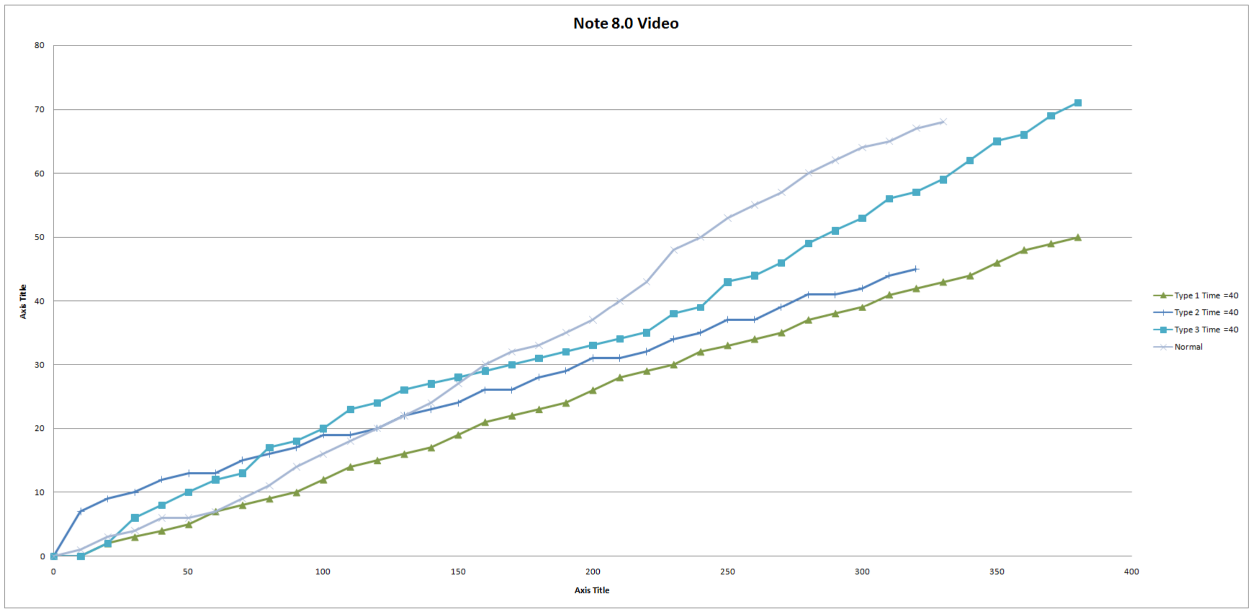
Use Case 52



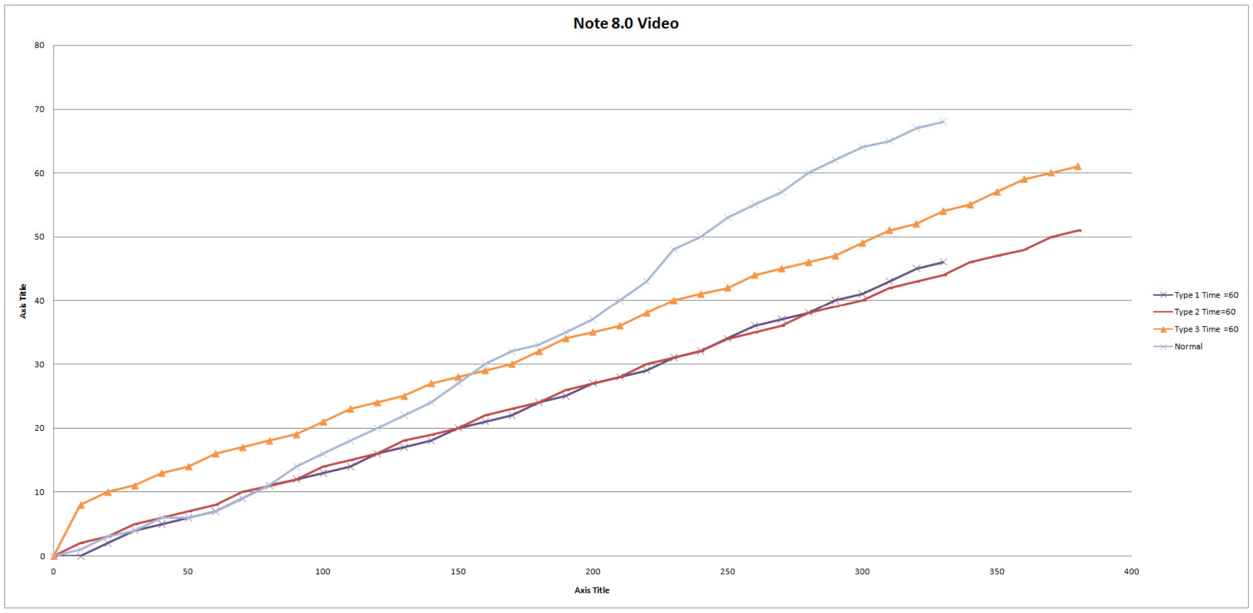
Use Case 53



Use Case 54

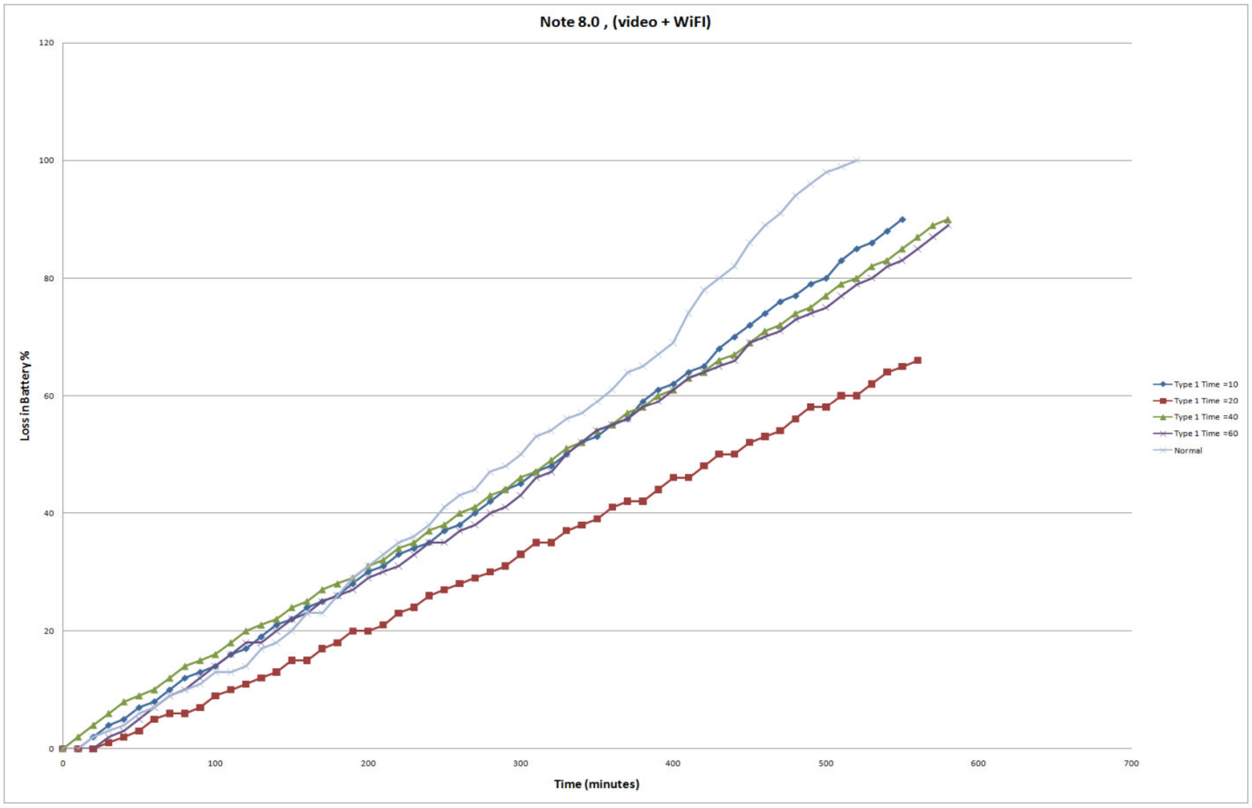


Use Case 55

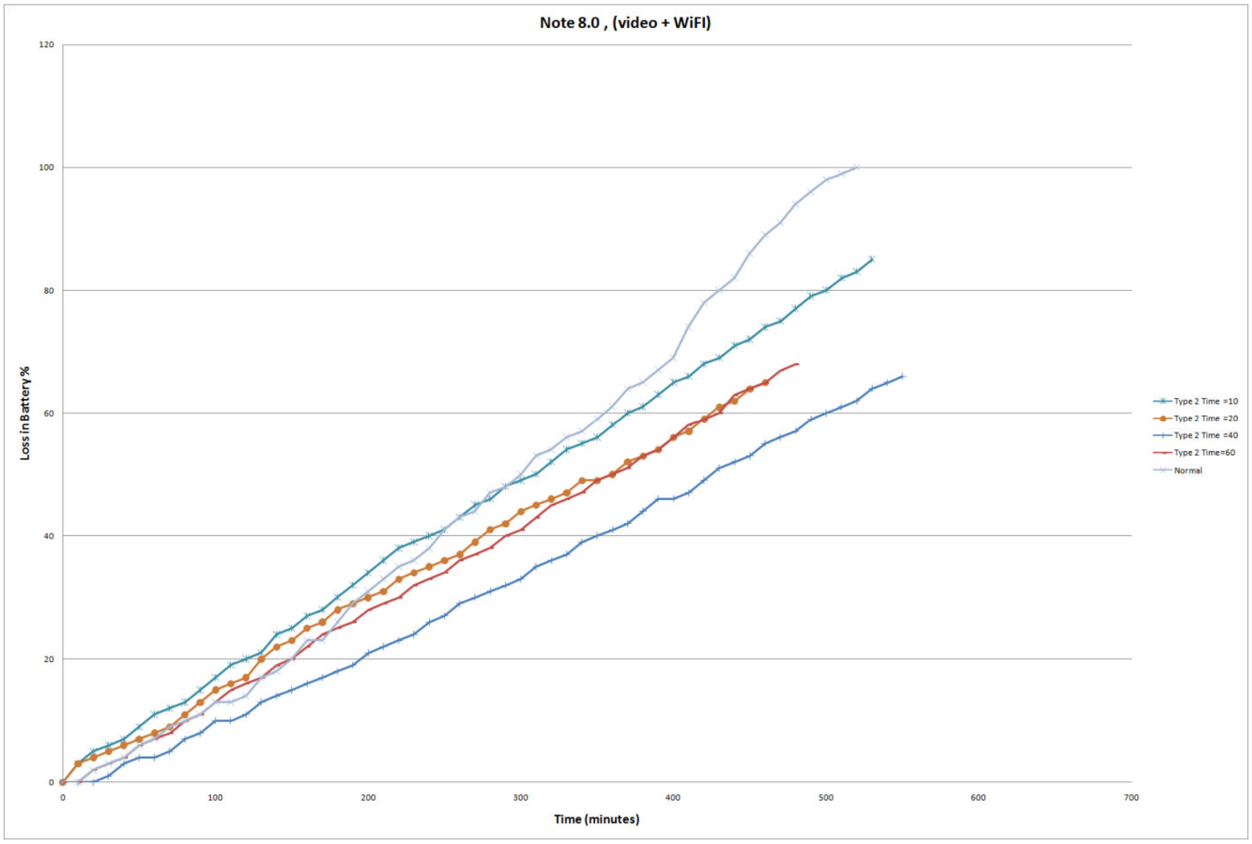


Use Case 56

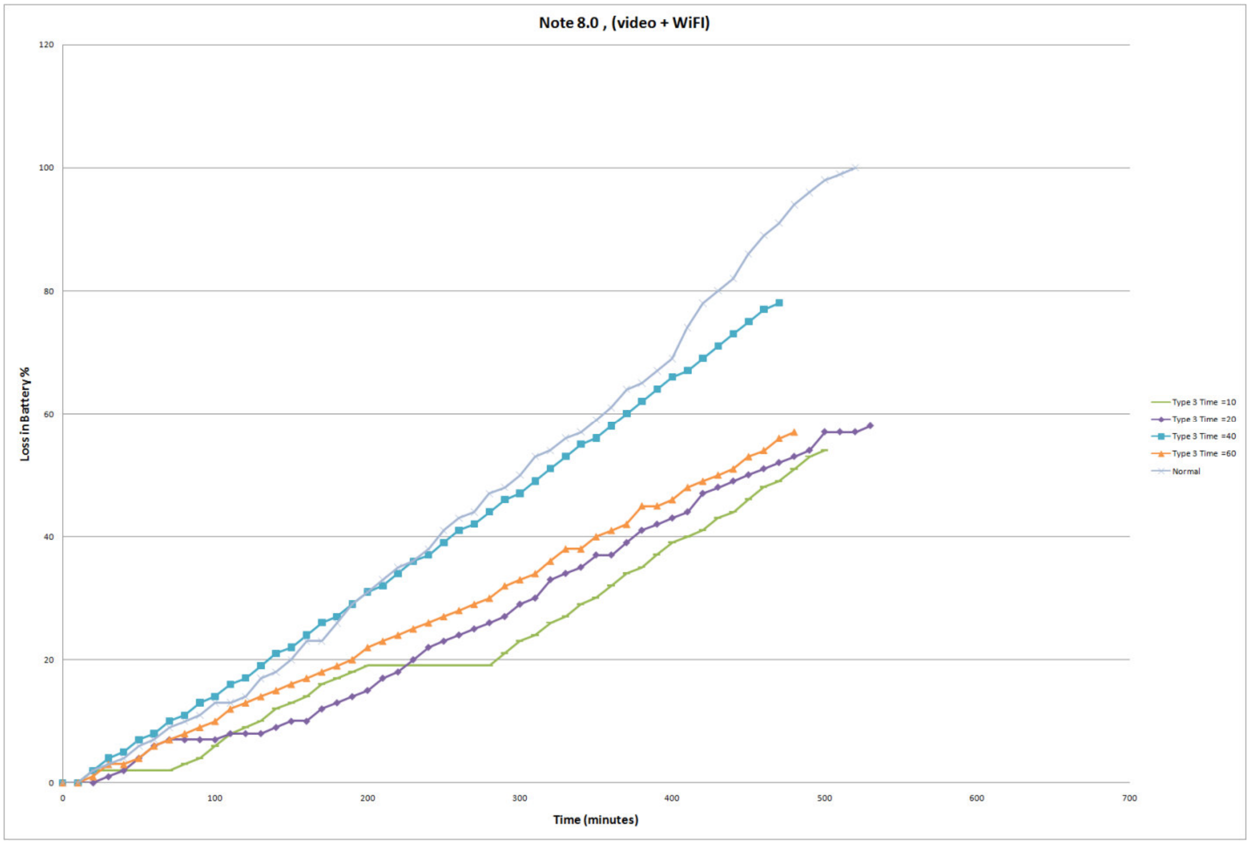
Case 3.3 - Class of video+Wi-Fi application on Note 8.0



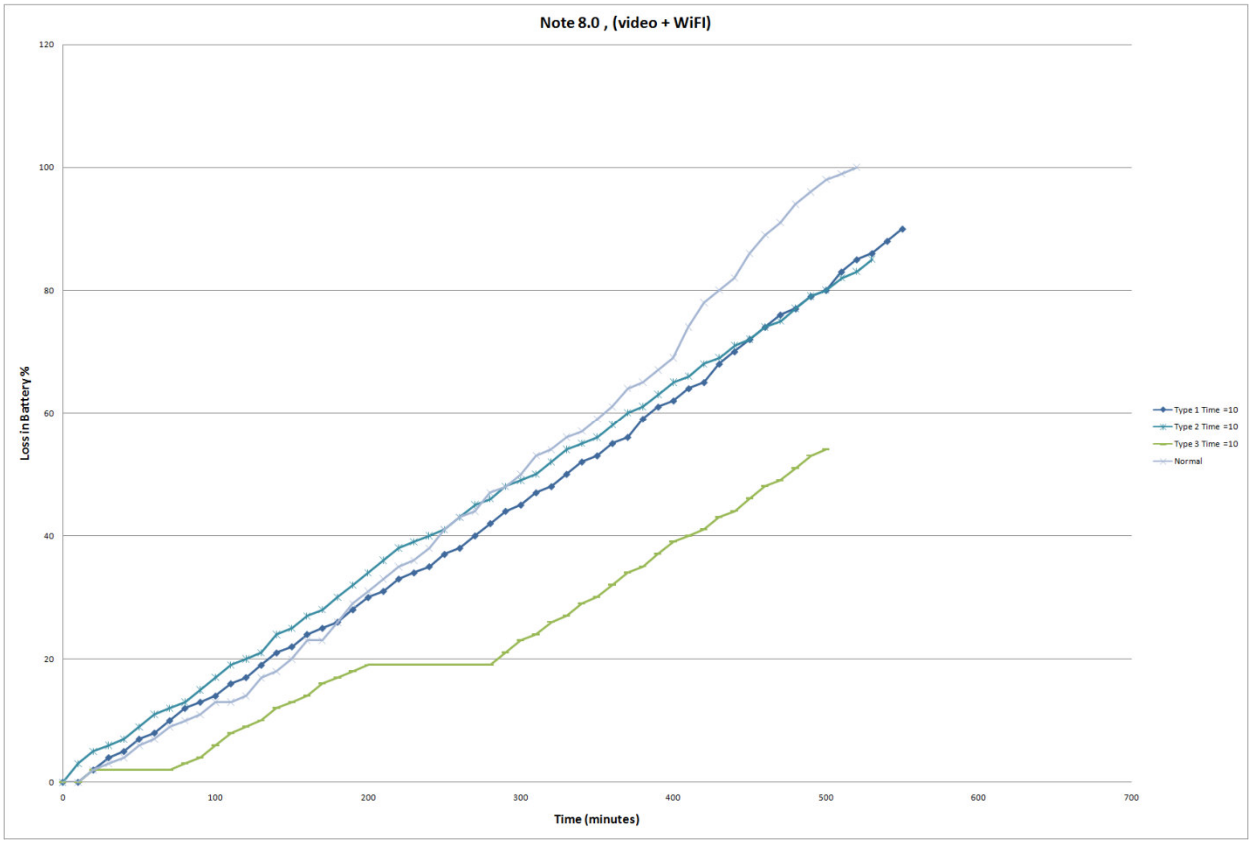
Use Case 57


Note 8.0 , (video + WiFI)
Loss in Battery %
Time (minutes)
Type 2 Time =10
Type 2 Time =20
Type 2 Time =40
Type 2 Time=60
Normal
0
20
40
60
80
100
120
0
100
200
300
400
500
600
700

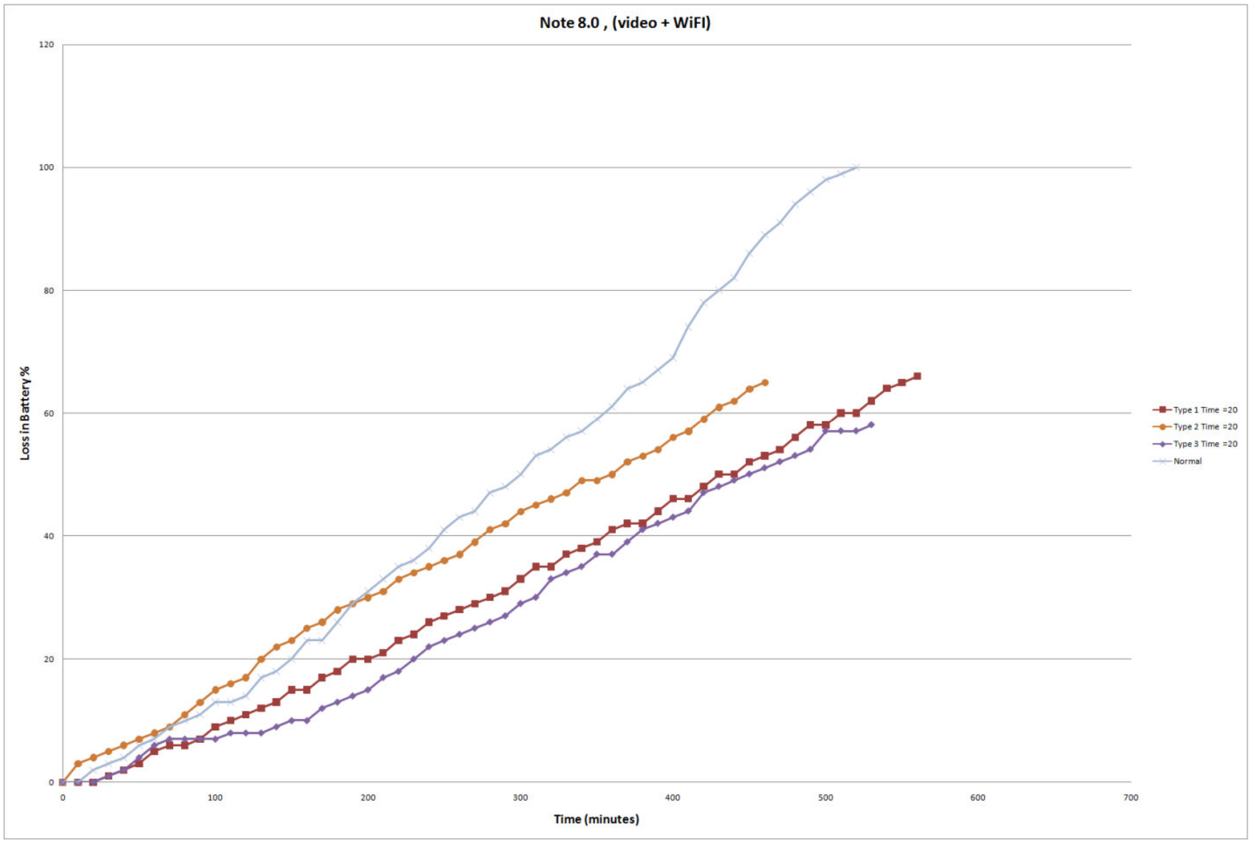
Use Case 58


Note 8.0 , (video + WiFI)
Loss in Battery %
Time (minutes)
0
20
40
60
80
100
120
0
100
200
300
400
500
600
700
Type 3 Time =10
Type 3 Time =20
Type 3 Time =40
Type 3 Time =60
Normal

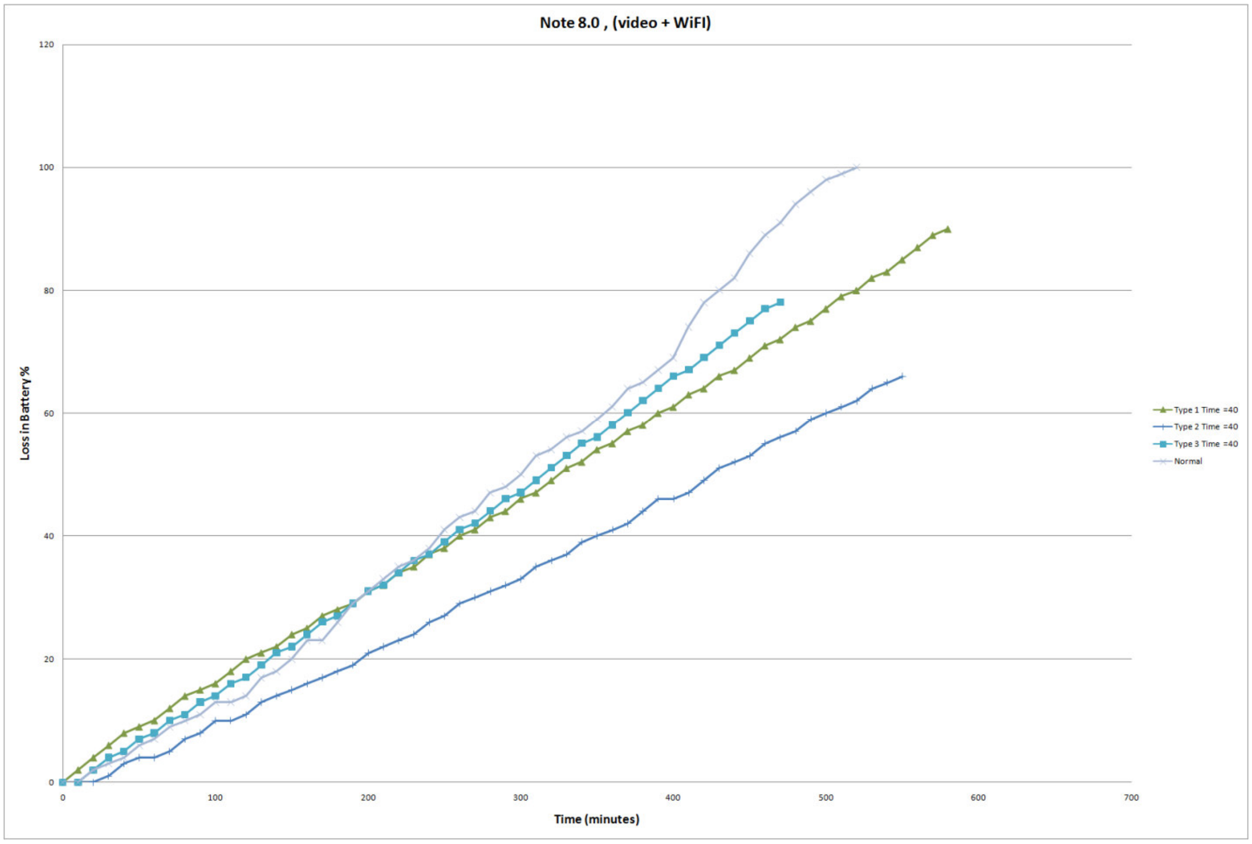
Use Case 59


Note 8.0 , (video + WiFI)
Loss in Battery %
Time (minutes)
Type 1 Time =10
Type 2 Time =10
Type 3 Time =10
Normal
0
20
40
60
80
100
120
0
100
200
300
400
500
600
700

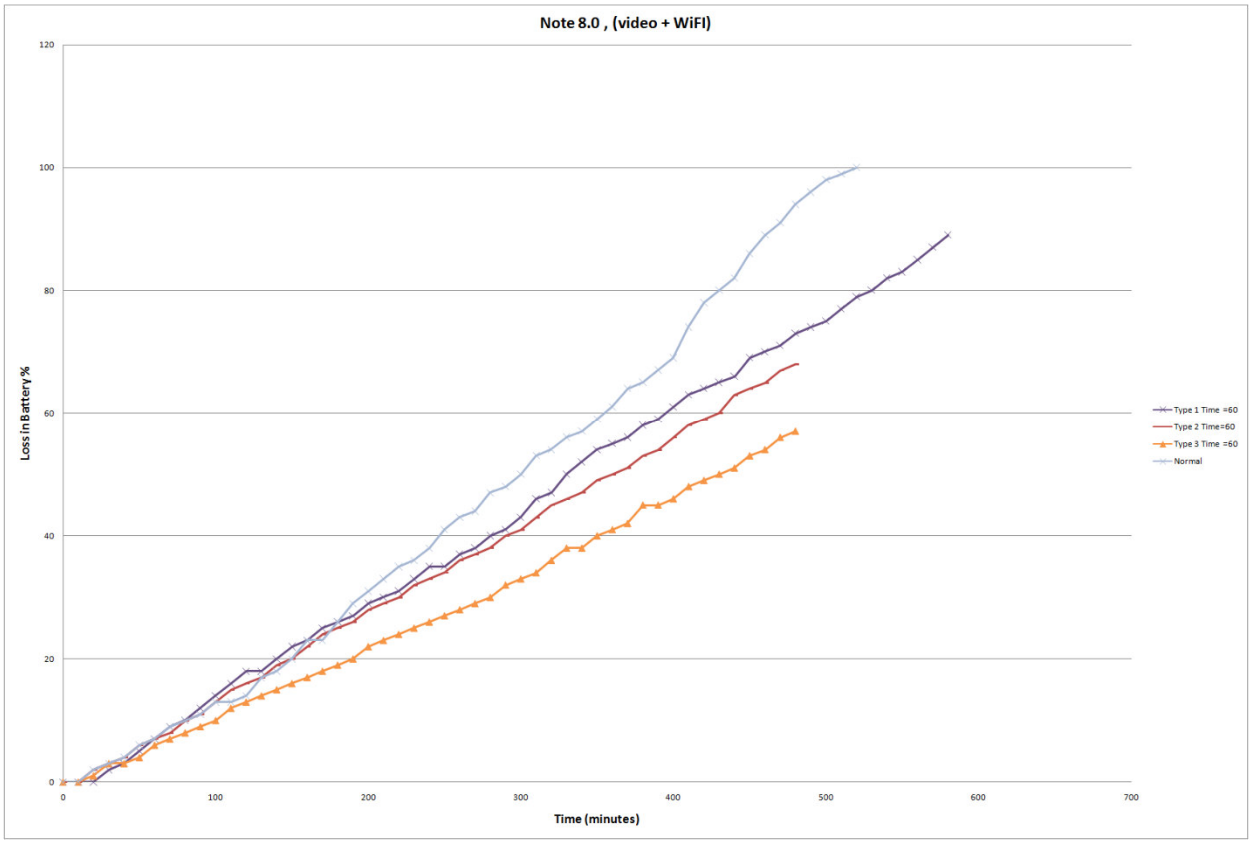
Use Case 60


Note 8.0 , (video + WiFI)
Loss in Battery %
Time (minutes)
Type 1 Time =20
Type 2 Time =20
Type 3 Time =20
Normal
0
20
40
60
80
100
120
0
100
200
300
400
500
600
700

Use Case 61


Note 8.0 , (video + WiFI)
Loss in Battery %
Time (minutes)
0
20
40
60
80
100
120
0
100
200
300
400
500
600
700
Type 1 Time =40
Type 2 Time =40
Type 3 Time =40
Normal

Use Case 62


Note 8.0 , (video + WiFI)
Loss in Battery %
Time (minutes)
Type 1 Time =60
Type 2 Time=60
Type 3 Time =60
Normal
0
20
40
60
80
100
120
0
100
200
300
400
500
600
700

Use Case 63