Università
Ca'Foscari
Venezia

Master's Degree programme — Second Cycle
(D.M. 270/2004)
in Informatica — Computer Science
Final Thesis

—

Ca' Foscari
Dorsoduro 3246
30123 Venezia

Smartphone energy analysis in WiFi walkable areas by Abstract Interpretation

Supervisor
Prof. Agostino Cortesi

Candidate
Enrico Eugenio
Matriculation number 847564

Academic Year
2015/2016

... ai miei genitori.

Abstract

The huge increase in the usage of mobile devices has led to the need of sophisticated optimisation techniques in order to minimise energy wastage. In this thesis we analyse energy consumption of mobile devices during the exchange of data, while walking in a WiFi network area, in order to study the dynamic of the power absorption. This analysis can be used to develop suitable optimisations in case of poor signal. The analysis is obtained as an instance of the Abstract Interpretation framework for semantics-based software verification, and the results are validated by a preliminary experimental evaluation.

L'enorme aumento negli ultimi anni dell'utilizzo di dispositivi mobili ha portato ad uno studio sempre più approfondito delle tecniche di ottimizzazione per minimizzarne lo spreco di energia. In questa tesi abbiamo analizzato il consumo di energia di questi dispositivi durante lo scambio di dati, mentre si cammina in una zona coperta da rete WiFi, per studiare la dinamica della potenza assorbita. Questa analisi può essere usata per sviluppare possibili ottimizzazioni in caso di segnale debole. L'analisi è stata ottenuta utilizzando l'Interpretazione Astratta, una tecnica per la verifica del software basata sulla semantica, ed i risultati sono convalidati tramite una valutazione sperimentale.

CONTENTS

CHAPTER 1

INTRODUCTION

The prominent growth of the smart phone utilisation has driven to develop Apps that use a constant data stream, letting real-time system (Call and Videocall, as Skype, Facebook Massenger, Whatsapp...) keep a certain quality of service.When using these services it is recommended (when possible) to use a WiFi connection instead of LTE technology, because of the greater stability, the less energy consumption and to prevent data overuse, given the limitations of current phone plans.

Is it always necessary to keep a WiFi connection?

As we will analyse, the performance in terms of connection speed and energy consumption, gradually degrades as you walk away from centre of the cell of Access Point [AP]. When it gets to the connection lower bound, such that the speed does not allow to get an acceptable QoS there is an expensive, and therefore useless, aggressive use of battery resources. The limited life of battery is a very important limitation in modern devices, for this reason, to extend life is crucial to understand how to optimise the use of this resource.

Is it possible determine a point beyond which (compared to the App minimum QoS) maintaining the connection is useless?

The aim of the thesis is to study how the applying of a threshold point can ensure energy saving. Adopting this technique is possible to fix a point, which relates power consumption and intensity of the WiFi signal, to define a lower bound for the QoS. This analysis will show how much energy is expended in order to do common tasks for a user, such as sending a text message via a Internet messaging App. In order to implement this investigation we have to define more specifically our problem and the parts in our interest.

State of art

The development of more and more sophisticated applications on mobile devices has made the problem of energy minimisation and power consumption crucial. Since mobile devices like smart phones and tablets are battery driven, energy consumption issue must be taken into account in order to establish the best practices for saving energy and make the battery life as higher as possible. Nowadays, in this field, display hardware and network connections are considered the most power-hungry components in small devices, followed by third-part advertisements displayed in most free application [1]. Almost all the studies on energy minimisation and power consumption generated a specific tool with the goal of monitoring and, in some cases, minimising energy consumption on Android devices. At the state of art, various analyses have been performed on the use of different modules, integrated into modern mobile devices. As so far analysed can be summarised in the following points:

- **Network connections**: in particular is observed the use of energy caused by the transmission of data via WiFi by the Internet based applications that are popular in [19]; analysis and comparison of performance and consequent consumption among the new LTE networks and 3G networks / WiFi in [13];

- **WiFi module**: alternative proposals to the continuous use of WiFi sensor (e.g managing such situations in which the user is moving) for a set of applications that use real time stream (ex. Videochat) [14]

- **Benchmarking of the modules** that can adversely affect energy consumption (CPU, WiFi, bluetooth, GPS, accelerometers, cameras, antenna, displays) [4].

In the papers that we have analysed, no static and dynamic analysis methods were implemented on energy consumption and minimisation issue. For this, what we want to do is try to analyse two simple code snippets, an original and an adapted one, using static and dynamic analysis methods in order to conclude given a good model to minimising energy expenditure in Android devices.

Program Analysis

The program analysis refers to a set of techniques that allow to automatically analyse the instructions of a program to highlight relations among the program. These analysis methods are applied in the field of software verification and they are used to detect flaws in the implementation of the algorithms. The program analysis techniques can be divided into static analysis and dynamic analysis.

Dynamic Analysis

The dynamic analysis is an analytical methodology performed by executing the program and observing its behaviour. To make this process effective the program must be run using a series of test inputs made in order to get a suitable behaviour. The dynamic approach has the following properties:

- precision: that allows to observe the behaviour of a program when it is executed, making the results detailed and precise. For example, to infer the value of a certain variable at a certain time just run the program to the desired point and observe the value that this variable takes.

- specificity: the information obtained dynamically can not be generalised and include all the possible versions, since they depend on a particular input configuration. This means that the results obtained are related to a specific run.

Static Analysis

Differently, this analytical methodology is implemented without the effective execution of the programs. In most cases it is carried out on the source code of a given application. The sophistication of the static analysis process varies depending on whether we consider only the behaviour of individual instructions or the behaviour of the entire program analysed. The information obtained from this analysis process can be used both to highlight coding errors and for the creation of formal methods that mathematically demonstrate the properties of a given program. The formal methods are analysis techniques which results are exclusively obtained through rigorous mathematical methods. It has been shown that finding all possible coding errors, or more generally any type of violation is an undecidable[1] problem. In other words, there is no mechanical method always capable of determining whether a given program may or may not generate runtime errors (Rice theorem [12]). However, there are some formal static analysis techniques that attempt to provide solutions which approximate the results.

- correctness: the results describe accurately the behaviour of the program, independently from the input and the specific execution.
- generalisation: the properties derived from the analysis are, in principle, weaker than they really are.

[1] All non trivial questions on the concrete program semantics are undecidable

Combining Analyses

Both analyses were made to take advantage of all strengths and fill the gaps that could result from the use of only one of the two. Initially we have statistically analysed the snippet of code under investigation, showing how the addition of a `IF` statement served as switch, inhibits the sending of data in certain signal conditions. Once we have a positive theoretical response for our thesis, was performed dynamic analysis, showing how the device really behaves in case of lack of signal.

General Power Consumption

The fundamental part of the analysis is the power consumption, which defines the total amount of energy expenditure of each component, better identified as *module*, in a specific system. We will give a general overview on the power consumption in a mobile system and then we will try to formalise the property better and give a reasonable definition of power consumption.

$$Pc = P_{CPU} + P_{DISPLAY} + P_{3G} + P_{GPS} + P_{AUDIO} + P_{MIC} + ... + P_{WIFI}$$

Power consumption in a specific system defines the total amount of energy expenditure of each module. Initially, we are not going to empirically measure the energy expenditure, but, once detected a module to analyse, it is interesting to identify those instructions that involve energy expenditure, ignoring those that do not bring any energy expenditure. For instance, for the WiFi module, we can consider only the instructions that change the state of the module itself, that are the instructions which involve a certain expenditure of energy. In this way we can isolate the power consumption of a single module, without the rest of the device may interfere in the analysis. In order to better understand the energy expenditure of a single instructions, we have studied the possible methods and possible scenarios in which the module can occur. We have taken into account the WiFi module and we have created a schema (Figure 1-1) for summarise all the states and the

significant methods (the ones that allows to switch through the states).

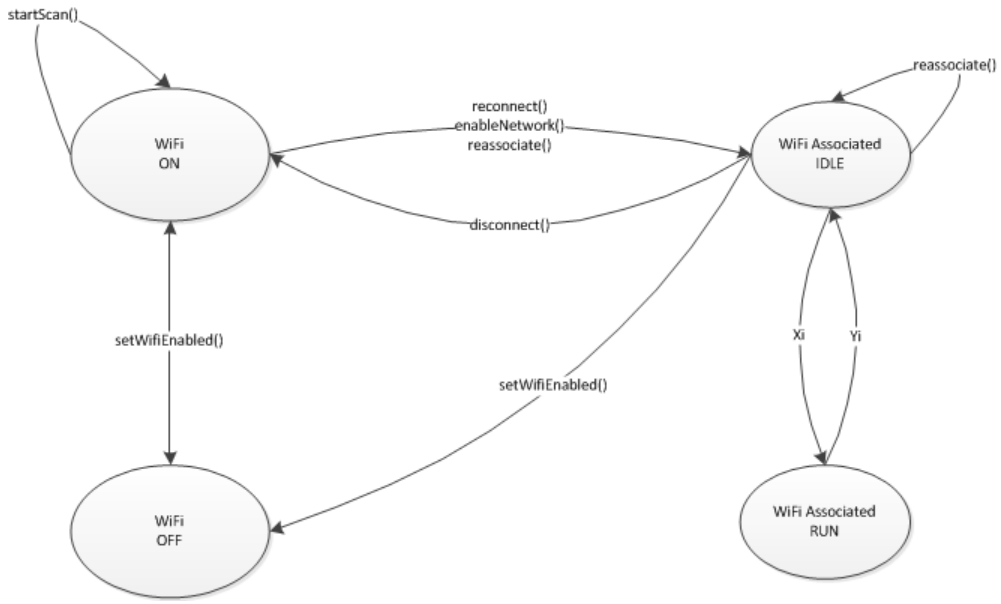


Figure 1-1: Wifi Schema States

A description of each state is given below.

- `Wifi_off` it's the state in which the WiFi module is turn off
- `Wifi_on` this state represent the WiFi module turn on. We can enter in this state considering the *setWifiEnabled()* method[2]. When we are in the state in which the WiFi is on, we can call the *startScan()* method in order to scan all possible network available (through the Access Point) and choose one of them in order to being associate with this. From this state can be also performed other methods like: *reconnect()* trying to hang up the currently active AP, if we are disconnected; *reassociate()* to reconnect at the currently active AP, even if we are already connected and the *enableNetwork()* method, to associate at one of the previously configured networks;
- `Wifi_idle` this represent the state in which the association with a certain network is established, but no data transferring has begun yet. We reach

[2]We consider all the methods in this section as instructions of Android.

this state through some methods: *reconnect()*, *reassociate()* and *enableNetwork()*. The *reassociate()* method allows us to remain in the `Wifi_idle` state;

- `Wifi_run` this represent the state in which the data transferring begins. We can reach this state whenever an instruction that allow the WiFiManager to initialise a channel and start the data transferring. Otherwise, we can came back to the `Wifi_idle` through all the instructions that allows to stop the data transferring and destroy the channel.

Let us to define the cost inside each state and the cost when a state transition happens. [18] Android measure the power use during those operations and compute approximately the values. As the goal of a power profile is to estimate battery drain appropriately, power profile values are given in current (mA). We emphasise the fact that these values are estimated based on an average of more devices to give a dimension to the energy consumption of the various methods and are not intended as precise data.
For better understanding of the units of measurement that will be used in the study we refer to Appendix B.

Table 1.1: Energy Costs

Method	Cost	Description
setWifiEnabled()	2mA	Turn on the Wifi module
startScan()	100mA	WiFi is scanning for access points
reconnect()	31mA	Since switch from `Wifi_on` to `Wifi_idle` involve energy consumption, we consider the energy expenditure of the previous state `Wifi_on` plus the reconnect() value.
reassociate()	31mA	Since switch from `Wifi_on` to `Wifi_idle` involve energy consumption, we consider the energy expenditure of the previous state `Wifi_on` plus the reassociate() value.
enableNetwork()	31mA	Since switch from `Wifi_on` to `Wifi_idle` involve energy consumption, we consider the energy expenditure of the previous state `Wifi_on` plus the enableNewtork() value.
disconnect()	0mA	0mA because we don't add any energy expenditure when we dis-connect from a network.
[RUN]	? mA	The current draw during the data exchange varies according to external factors, which we will study in the next section.
[STOP]	0mA	0mA because we don't add any energy expenditure when we stop data transferring

Power consumption in Android

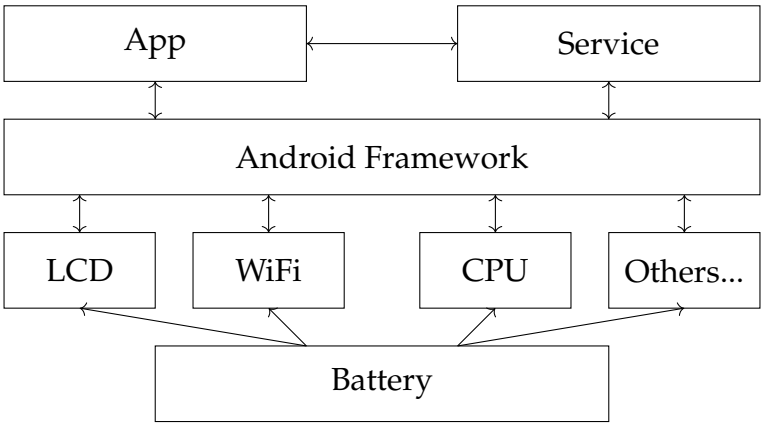


Figure 1-2: Battery View Schema

As said, we want to do an analysis of an App regarding the use of a specific module in Android environment; in the Figure1-2 above we can see an Android layer configuration oriented on the power consumption. The Android framework, together to the Linux kernel provide basic features to the applications and the services [3]. The Android Framework encapsulates the components below, to make programming easier, but this makes it difficult to know how much energy consumes each module.

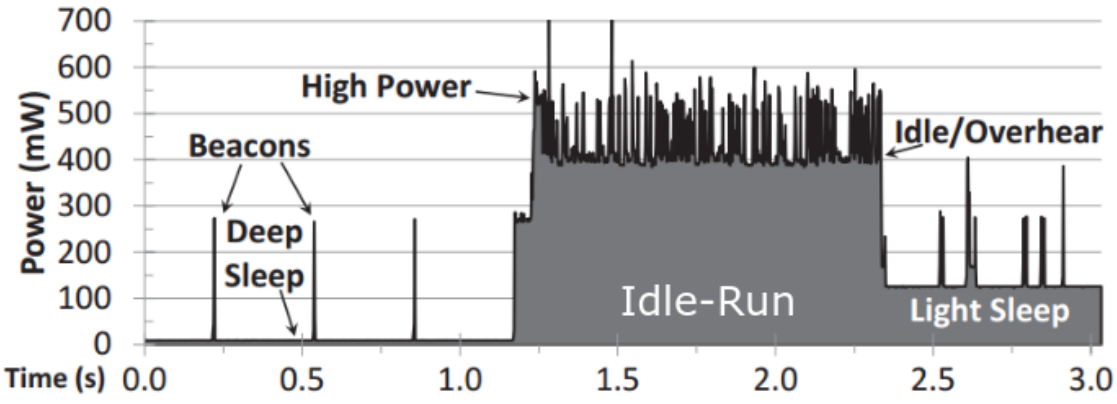


Figure 1-3: Power consumption of the WiFi module over time per each state [15]

[3]The main difference between these two types of programs is given by the fact that an application presents a GUI, while a service is a background process without a GUI

Focusing to the energy consumption of the Wifi module, we can see from the Figure 1-3 taken from [15], the profiling of the power consumption of the module involved. As previously mentioned, an empirical measure of a single module may not be precise, but concentrating on the state 'Idle - Run', we can see frequent power spikes. These power peaks of different intensity, are directly related to the WiFi signal strength. The components that affect on the rise of the energy absorbed during the exchange of the data are many and of various nature, we can list the major[5]:

- Channel width
- Limited bandwidth
- Channel noise
- Channel capacity

Everything is also attributable to a weak wireless signal strength, which force the module to compensate by increasing its transmission and reception power which can significantly increase the energy drain of apps that perform network activities. A weak wireless signal strength can be seen as a possible root cause for unusual battery drain spikes. Moreover, recalling the results of the research made by [11], a wireless channel can be noisy, and this noise can go to affect the capacity of the channel and consequently affects the achievable network performance. Furthermore, an estimated signal strength equal to or less than -80dBm (poor signal strength) gives negative effects not only on the speed of data transfer but also on energy consumption. Therefore, the distance between communicating endpoints will affect a given radio performance in term of throughput, due to higher latencies which depends to channel noise, this implies more retransmissions for reliable communication. So the channel capacity can change due to distance from the base station and this effect contribute to the higher power consumption of WiFi networks. It should be added that a state of very low signal, which leads to a frequent disconnection and reconnection to a wifi network, brings to a very intense current absorption. Insights in Appendix A.

Summary

The results we present in this work are the following:

1. we formalise energy consumption analysis in the Abstract Interpretation framework,

2. we show the effectiveness of the analysis on the messaging-like application,

3. we validate the results of the analysis with empirical experiments, by computing the power consumption related to WiFi connection activity in a walkable area.

The rest of the thesis is structured as follows: Chapter 2 gives the fundamental mathematical concepts used to develop this work, Chapter 3 formalises the analysis, focusing on the semantics of the application and in Chapter 4 the formalisation is applied in case of our interest. In Chapter 5 a mathematical software is used to simulate an ideal environment, to be used by the analysis and shows the results of the empirical validation of our work. Chapter 6 concludes.

CHAPTER 2

MATHEMATICAL NOTIONS

In this chapter we will define the mathematical concepts, related to the Set and Order theories, which will help us to understand better the study that we will do in the next chapter.

Sets

A set is a collection of elements, which can be considered as a single element, consequently can belong to other sets. The collections of items that interest us are well known sets as the set of natural numbers $\mathbb{N}$, the set of integers $\mathbb{Z}$, the set of booleans $\mathbb{B}$. We can define also an non-empty finite set of characters, called alphabet, and define string a finite sequence of character. We use the standard notation $n \in \mathbb{N}$ to say that n is an element of the $\mathbb{N}$ set.The cardinality of a set is defined as $|\mathbb{N}|$ and it represents the number of elements contained in the set. We can represent finite sets by listing their elements between curly brackets, like $true, false$. We write $A \subseteq B$ to indicate that it A is contained in B, or that A is subset of B. We also define $\emptyset$ as an empty set, and $\wp(A)$ as a power set of A i.e. the set of all subsets of A, including the $\emptyset$ and A itself.

Relations and Functions

Definition 2.1.1 (Cartesian product)**.** Let A and B sets. Their Cartesian product $A \times B = \{(a, b) \,|\, a \in A \wedge b \in B\}$ is the set of pairs with the first element of A and second element of B.

Definition 2.1.2 (Relation)**.** A relation between two sets A and B is a subset of $A \times B$.

Definition 2.1.3 (Binary Relation)**.** Let ρ be relation over A, we can say that ρ is a binary relation, or (A, ρ) is an ordered set if:

- $\forall a \in A \rightarrow (a, a) \in \rho$
- $\forall a, b \in A, \exists (a, b) \in \rho \wedge (b, a) \in \rho \rightarrow a = b$
- $\forall a, b, c \in A, \exists (a, b) \in \rho \wedge (b, c) \in \rho \rightarrow (a, c) \in \rho$

An example of binary relation is the set of natural numbers $\mathbb{N}$ with the relation $\rho = \{(x, y) \in \mathbb{N} \times \mathbb{N} \,|\, x \leq y\}$. An order relation ρ over A, is called *total order* if $\forall a, \forall b \in A$ holds $(a, b) \in \rho$ or $(b, a) \in \rho$; otherwise it is called *partial order*.

Definition 2.1.4. Given a binary relation $\leq$ over the set A and let $a \in A$ and $B \subseteq A$. Then

- a is *upper bound* for B if $\forall b \in B,\ b \leq a$
- a is *least upper bound* for B if a is *upper bound* for B and $a \leq c$ where $c \in A$

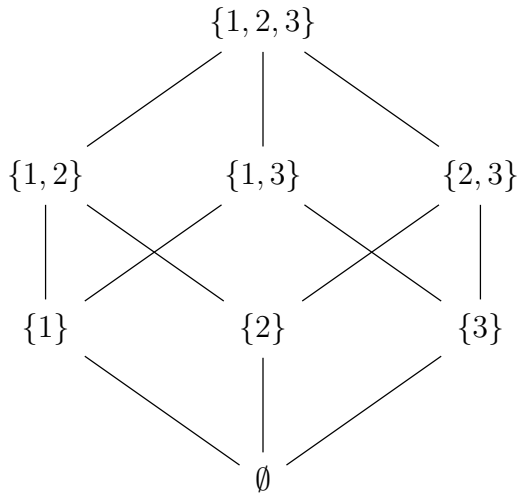
Dually we define the concepts of *lower bound* and *greatest lower bound*.

Algebraic Concepts

A fundamental part of the thesis are the algebraic structures, in fact we will use them in order to define the items that then there will be used to analyse the states of the software. In this section we report definitions taken from [3, 10]

Definition 2.2.1 (Poset)**.** A partially ordered set is a set A with a binary relation $\leq$ that indicates that, for certain pairs of elements in the set, one of the elements precedes the other, written as $(A, \leq)$

By using the Hasse diagram, we can provide a graphic example of a poset:



The figure in example is a graphical representation of poset $(A, \subseteq)$ where $C = \{1, 2, 3\}$, as we can see, the top element is the major element with respect to the binary relation, vice versa the bottom element is the empty set. In particular, we can define *chain* all the elements of the poset that are in relation, two by two.

Definition 2.2.2 (Lattice)**.** Given a poset $(A, \leq)$, it's also a lattice if, for every pair $x, y \in A$, has:

- a least upper bound $(x \vee y)$ and
- a greatest lower bound $x \wedge y$.

Furthermore, is called *complete lattice* is a partially ordered set in which all subsets have both a least upper bound and a greatest lower bound.

Definition 2.2.3 (Fixed Point)**.** Given a function $f : A \to A$ over a poset $(A, \leq)$ and $a \in A$ then a is a fixed point of f if $f(a) = a$. Written as $fix(f)$

Theorem 2.2.1 (Tarski Theorem)**.** Given a complete lattice $< C, \leq, \vee, \wedge, \top, \bot >$ and a monotonic function $f : C \to C$, then the $fix(f)$ is a complete lattice .

Galois connection

Galois connections are algebraic structures which form the basis of abstract interpretation, and they allow to transfer properties from a domain to another. They are composed of a concrete and an abstract posets, and two functions called respectively concretization and abstraction functions [9].

Definition 2.2.4 (Galois connection)**.** Given two poset $(C, \leq_C)$ and $(A, \leq_A)$ and two functions $\alpha : C \to A$ and $\gamma : A \to C$, the tuple (C, γ, α, A) is a Galois connection if:

- the α and γ are monotonic
- $\forall c \in C$ and $\forall a \in A$ so $\alpha(c) \leq_A a$ iff $c \leq_C \gamma(a)$

written as $(C, \leq_C) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (A, \leq_A)$

In our work we also need to use the *interval analysis* [17] which is defined as:

Definition 2.2.5 (Interval)**.** the interval $[l, u] | l, u \in \mathbb{Z} + \{-\infty, +\infty\}, l \leq u$ and where the lattice of the interval is $I = \{\bot, [l, u], (-\infty, u], [l, +\infty), \top\}$. It is a complete lattice:

$$[l, u] \sqcup [l', u'] = [min(l, l'), max(u, u')]$$
$$[l, u] \sqcap [l', u'] = [max(l, l'), min(u, u')]$$

It tracks the interval of values that each variable might hold.

Widening is another method for enforcing termination of the abstract interpretation. General idea is to eliminate unstable components through consecutive iterates and find a more approximate but sound upper bound of the iteration sequence. In case of infinite ascending chain, the *widening operator* [7] shorten the chain to a finite height.

Definition 2.2.6 (Widening Operator)**.** Given a poset $(L, \sqsubseteq)$ is a binary operator $\nabla \in L \times L \to L$ such that

$$\forall x, y \in L : x \sqsubseteq x \nabla y$$

$$\forall x, y \in L : y \sqsubseteq x \nabla y$$

and for all increasing chains $x^0 \sqsubseteq x^1 \sqsubseteq ...$, the increasing chain defined by $y^0 = x^0, ..., y^{i+1} = y^i \nabla x^{i+1}, ...$ is not strictly increasing.

The purpose of the *widening operator* is to compress the infinite chain to a finite length, considering the most recent elements in the chain.

Finally we can give a summary of this chapter in the following table, the element of the interval domain is denoted as $[a, b]$ where $a \in \mathbb{Z} \cup -\infty, b \in \mathbb{Z} \cup \infty$

$\sqcup$	$[l, u] \sqcup [l', u'] = [min(l, l'), max(u, u')]$
$\sqcap$	$[l, u] \sqcap [l', u'] = [max(l, l'), min(u, u')]$
$\sqsubseteq$	$[l, u] \sqsubseteq [l', u'] = l \geq l' \wedge u \leq u'$
∇	$[l, u] \nabla [l', u'] = \begin{cases} \text{if } l' < l \text{ then } -\infty & \text{else } l \\ \text{if } u < u' \text{ then } \infty & \text{else } u \end{cases}$
$\top$	$[-\infty, \infty]$
$\bot$	$\emptyset$
α	$\alpha(k) = [k, k]$
γ	$\gamma([a, b]) = \{x \in Z \mid a \leq x \leq b\}$

Table 2.1: Math Summary

CHAPTER 3

SEMANTICS

In this chapter we introduce the formal semantics of programming languages and explains briefly the main approach to the *operational semantic*, as it will be a key tool for our work. It describes the meaning of programs such as sequences of steps of computation performed on a machine, in this case we are interested in how computing is done. Indeed, to check the connection status and signal level we have to understand how information flows in the program semantics. The semantics of a program describes the run-time behaviour, i.e. the set of all its possible behaviours when it is executed for all possible input data. The description of the semantics can be made using the concept of a transition system whose function is that of interpreter of the programming language.

Language

Knowing that a program is a sequence of statements we define `Var` to be the set of variables in the *program*. Let V be the set of values that each variable can be assigned to. We define $\perp \in V$, which means that variable is uninitialised. The syntax of a language defines the way in which the various components of the language can be associated to form legal instructions and programs. On the application side, we analyse the sending data according to its type (text, audio, video) and the its

dimension(dim). On the other hand, by the device side, we analyse the connection status ($conn$), battery status ($batt$) and signal level($sign$). We define the tuple SysApp, which groups the variables $< type, dim >$ that represent the status of the application; the tuple SysDev $< conn, batt, sign >$ representing the state of the device. The pair $< SysApp, SysDev >$ represents the state of the entire system of our interest. Both systems have the methods `Get` and `Put`, which return or set the values of each tuple. The application and the Android layer are implemented in Java, to simplify our work we will introduce a smaller version of this language, composed only by those constructs required in the analysis, we must be emphasised that not all the aspects and details of the possible behaviour of a program during its execution should be mandatory considered. Indeed, the handling of a program is simplified if we consider "shaped" semantic, i.e. obtained by abstracting from issues that are in that context and in that moment irrelevant. This is a simple sequential non-deterministic programming language with no procedures, no pointers and no recursion.

Syntax

We introduce the syntax composed of different constructs, called *syntactic categories*, and particular variables which change in each category, called *meta syntactic variables*[1]:

- n refers to an element belonging to the set of numbers, Num
- z refers to an element belonging to the set of string, $String$
- $x,\ y$ refer to elements belonging to the set of variable, `Var`
- a refers to the set of arithmetic expressions, $Aexp$
- b refers to the set of boolean expressions, $Bexp$
- w refers to the set of string expressions, $Wexp$

[1] A meta syntactic variable refers to a generic element within a given category

- s refers to the set of statements

There are also sub-domains related to the free variables `Var` that we analyse

$type \in$ { "text", "audio", "video"}

$dim \in \mathbb{N}$

$conn \in$ { "on", "off", "idle", "run"}

$batt \in [0, 100] \subseteq \mathbb{N}$

$sign \in [-100, 0] \subseteq \mathbb{Z}$

Below we define the elements that characterise each syntactic category as follows:

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 - a_2$$

$$b ::= true \mid false \mid a_1 == a_2 \mid a_1 \Leftarrow a_2 \mid w_1 == w_2$$

$$w ::= z \mid w_1 \; concat \; w_2$$

$$s ::= \texttt{skip} \mid s_1; s_2 \mid x{=}a \mid \texttt{if(}\, b \,\texttt{)\{}\, s_1 \,\texttt{\}}\; \texttt{else\{}\, s_2 \,\texttt{\}} \mid \texttt{while(}\, b \,\texttt{)\{}\, s_1 \,\texttt{\}}$$

The semantics of our language is defined in terms of semantic functions corresponding to each syntactic category. So, a semantic function takes as argument a syntactic construct and returns its semantic meaning.

Semantic

We start defining the environment, a total semantic function which uniquely determines the value assigned to a variable; the values assigned by a map:

$$\sigma : \texttt{Var} \to V.$$

This map $\sigma \in \Sigma$ can be seen as a *memory* that stores the values of the identifiers where Σ is the set of all possible `Environments`. We denote with $q \in Q$ a program point, if it associated to Σ is called *state*. A state is represented as $\langle \sigma, q \rangle$ and the set of all *states* is $\texttt{States} = \Sigma \times Q$. As we mentioned, each environment σ

specifies a value $\sigma(x)$ for each variable x in the program, e.i. we can represent the environment as pair variable - value, where $\sigma[x \to 1, w \to$ "Hello"], such that in the environment σ the variable x worths 1 and the variable w worths $Hello$. Given an arithmetic expression a and a environment σ we can determine its meaning with a semantic function

$$A[\![P]\!] : Aexp \to (\Sigma \to \mathbb{N})$$

which returns the value of the expression. Here we define the meaning of the arithmetic expressions inductively on their syntactic structure.

$$A[\![n]\!]\sigma = n$$

$$A[\![x]\!]\sigma = \sigma(x)$$

$$A[\![a_1 + a_2]\!]\sigma = A[\![a_1]\!]\sigma + A[\![a_1]\!]\sigma$$

$$A[\![a_1 - a_2]\!]\sigma = A[\![a_1]\!]\sigma - A[\![a_1]\!]\sigma$$

We emphasize that the operators on the right are the common arithmetic operators for integer operations (addition and subtraction), while on the left are pure syntax.

We now define the meaning of Boolean expressions,

$$B[\![P]\!] : Bexp \to (\Sigma \to \{TRUE, FALSE\})$$

which return a Boolean value $\{TRUE, FALSE\}$. Here we define the meaning of the boolean expressions inductively on their syntactic structure.

$$B[\![true]\!]\sigma = TRUE$$

$$B[\![false]\!]\sigma = FALSE$$

$$B[\![a_1 = a_2]\!]\sigma = \begin{cases} TRUE & if\ A[\![a_1]\!]\sigma = A[\![a_1]\!]\sigma \\ FALSE & if\ A[\![a_1]\!]\sigma \neq A[\![a_1]\!]\sigma \end{cases}$$

$$B[\![a_1 <= a_2]\!]\sigma = \begin{cases} TRUE & if\ A[\![a_1]\!]\sigma \leq A[\![a_1]\!]\sigma \\ FALSE & if\ A[\![a_1]\!]\sigma > A[\![a_1]\!]\sigma \end{cases}$$

Similarly the two previous cases, now we're going to define the meaning of the strings expressions. Given an non-empty set, called alphabet Ψ, we can build the set of $String$ combining the elements from each subset of $\wp(\Psi)$, so

$$W[\![P]\!] : Wexp \to (\Sigma \to String)$$

Here we define the meaning of the string expressions inductively on their syntactic structure.

$$W[\![z]\!]\sigma = z$$

$$W[\![w_1\ concat\ w_2]\!]\sigma = W[\![w_1]\!]\sigma W[\![w_2]\!]\sigma$$

Taking the definition of environment, now we focus on the program point, a piece of code composed by a statement and/or a expression it represents the smallest standalone element of an imperative programming language that expresses some action to be carried out. It is an instruction written in a high-level language that commands the computer to perform a specified action. Here we define the **transition system** that interprets each command according to the following rules:

$$< skip, \sigma > \to \sigma$$

$$\frac{< s_1, \sigma > \to \sigma_1, < s_2, \sigma_1 > \to \sigma_2,}{< s_1; s_2, \sigma > \to \sigma_2,}$$

$$\frac{a \in Aexp}{< x = a, \sigma > \to \sigma[x \to A[\![a]\!]\sigma]}$$

$$\frac{b \in Bexp}{< x = b, \sigma > \to \sigma[x \to B[\![b]\!]\sigma]}$$

$$\frac{w \in Wexp}{< x = w, \sigma > \rightarrow \sigma[x \rightarrow W[\![w]\!]\sigma]}$$

$$\frac{B[\![b]\!]\sigma = TRUE, < s_1, \sigma > \rightarrow \sigma_1}{< \texttt{if(}\ b\ \texttt{)\{}\ s_1\ \texttt{\}else\{}\ s_2\ \texttt{\}}, \sigma > \rightarrow \sigma_1}$$

$$\frac{B[\![b]\!]\sigma = FALSE, < s_1, \sigma > \rightarrow \sigma_2}{< \texttt{if(}\ b\ \texttt{)\{}\ s_1\ \texttt{\}else\{}\ s_2\ \texttt{\}}, \sigma > \rightarrow \sigma_2}$$

$$\frac{B[\![b]\!]\sigma = FALSE}{< \texttt{while(}\ b\ \texttt{)\{}\ s\ \texttt{\}}, \sigma > \rightarrow \sigma}$$

$$\frac{B[\![b]\!]\sigma = TRUE, < s_1, \sigma > \rightarrow \sigma_1, < \texttt{while(}\ b\ \texttt{)\{}\ s\ \texttt{\}}, \sigma_1 > \rightarrow \sigma_2}{< \texttt{while(}\ b\ \texttt{)\{}\ s\ \texttt{\}}, \sigma > \rightarrow \sigma_2}$$

Transition Relation

The aim of semantics is to find all possible *environments* for each program point, while the program is running; knowing that each statement belongs to the program and a point of the program can be represented by a statement, the semantics can be described as a transition system which map a set of states to a new one; this passage from one state to another can be formalised as:

$$\tau : \texttt{States} \rightarrow \texttt{States}$$

$$\tau = \langle s_n, s_{n+1} \rangle$$

where τ is the transition relation between a state and its possible successors, $s_n, s_{n+1} \in \texttt{States}$ [8]. A serie of transitions is called **trace** and we call *finite partial execution trace* a trace which starts from a state and then moves on through transitions to another state. For instance, $s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_n$ denote a trace that begin with a state s_0, it is followed by s_1 and it end with s_n. We denote by $\texttt{States}^*$ the set of all finite traces composed of elements of `States`.

$$\texttt{States}^* = \{s_1 \to s_2 \to ... \to s_n \mid n \geq 1 \wedge \forall i \in [0, n-1] :< s_i, s_{i+1} >\in T\}$$

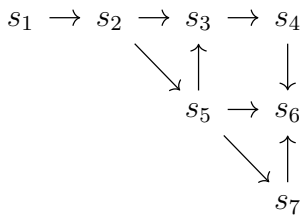
Now we can take two partial traces $\pi_0, \pi_1 \in PT \subseteq \texttt{States}^*$ and we can define the following operator:

- $\pi_0 \preceq \pi_1$ iff π_0 is a subtrace of π_1

We can define the complete lattice of partial execution traces as:

$$\langle \wp(\texttt{States}^*), \subseteq, \emptyset, \texttt{States}^*, \cap, \cup \rangle$$

Given the following set of traces



if

- $\pi_1 = s_1 \to s_2 \to s_5 \to s_3 \to s_4$
- $\pi_2 = s_2 \to s_5 \to s_3$
- $\pi_3 = s_1 \to s_2 \to s_3 \to s_4$

By the set above we denote that $\pi_2 \preceq \pi_1$ but $\pi_3 \not\preceq \pi_1$

Example

Let us consider the following program:

```
1 x = 2
2 while (x > 0 ){
3  if( x > 1)
4  { s = "Hello"; } else
5  { s = "Hell" ; }
6  x = x - 1; }
7  ;
```

The states of the program are $\texttt{States} = \{1,2,3,4,5,6,7\} \times \Sigma$, and the transitions are:

$$\tau_0 = \{< 1, \sigma > \rightarrow < 2, \sigma[x \rightarrow 2] >\}$$
$$\tau_1 = \{< 2, \sigma > \rightarrow < 3, \sigma >\}$$
$$\tau_2 = \{< 3, \sigma > \rightarrow < 4, \sigma >\}$$
$$\tau_3 = \{< 4, \sigma > \rightarrow < 6, \sigma[s \rightarrow \text{"Hello"}] >\}$$
$$\tau_4 = \{< 6, \sigma > \rightarrow < 3, \sigma[x \rightarrow \sigma(x) - 1] >\}$$
$$\tau_5 = \{< 3, \sigma > \rightarrow < 5, \sigma >\}$$
$$\tau_6 = \{< 5, \sigma > \rightarrow < 6, \sigma[s \rightarrow \text{"Hell"}] >\}$$
$$\tau_7 = \{< 3, \sigma > \rightarrow < 7, \sigma >\}$$

Now we consider the variables x and s, we can see in the table as they evolve during the execution of the snippet.

The analysis of a system often consists in the enumeration of all its possible executions. This example is not totally accurate for our problem, because the variables that we analyse will not have always precise values. An exact enumeration is difficult, because the set of reachable states maybe very large or even infinite. Instead of an exact analysis, in abstract interpretation the set of states in each location is

τ	**q**	**x**	**s**
τ_0	1	$\perp$	$\perp$
τ_1	2	2	$\perp$
τ_2	3	2	$\perp$
τ_3	4	2	$\perp$
τ_4	6	2	Hello
τ_1	2	1	Hello
τ_5	3	1	Hello
τ_6	5	1	Hello
τ_4	6	1	Hell
τ_1	2	0	Hell
τ_7	7	0	Hell

Table 3.1: Execution

abstracted as an upper approximation of the exact set. An upper approximation may be more efficient to manipulate than the exact set of values.

Let us recall the program from the Example:

$$q_1 : \{x \in [2, 2], s \in \emptyset\}$$

$$q_2 : q_1 \cup q_6$$

$$q_3 : q_2 \cap (1, \infty] \cap (0, \infty]$$

$$q_4 : q_3 \cup \{s \to \text{"Hello"}\}$$

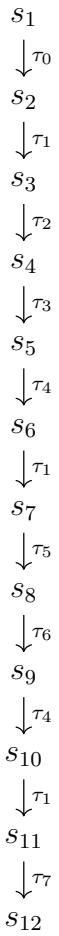
$$q_5 : q_2 \cap [-\infty, 1] \cup \{s \to \text{"Hell"}\}$$

$$q_6 : q_5 \cup q_4 \cup \{x \to (x - 1)\}\}$$

$$q_7 : q_2 \cap [-\infty, 0]$$

τ	**q**	**x**	**s**
τ_0	1	$[-\infty, \infty]$	$\emptyset$
τ_1	2	[2 , 2]	$\emptyset$
τ_2	3	[2 , 2]	$\emptyset$
τ_3	4	[2 , 2]	$\emptyset$
τ_4	6	[2 , 2]	{Hello}
τ_1	2	[1 , 2]	{Hello}
τ_5	3	[1 , 2]	{Hello}
τ_6	5	[1 , 2]	{Hello}
τ_4	6	[1 , 2]	{Hell, Hello}
τ_1	2	[0 , 2]	{Hell, Hello}
τ_7	7	[0 , 2]	{Hell, Hello}

Table 3.2: Execution



New environment

Recalling the previous set Σ, which define the set of the memory environments, we denote that this is not enough for a complete definition of the system we are going to analyse. For this we will define $\varepsilon \in \mathcal{E}$, which represents the status of identifiers by the device side. Now our environment is $\Phi = \Sigma \times \mathcal{E}$ where $< \sigma, \varepsilon >= \phi \in \Phi$. To address the analysis of the whole system in a more precise and deeper manner, now we will define the functions that allow us to read the application status.

- `read_type` : this *get*-function returns the value of $type$ in SysApp, reading from the σ. This function gives back the type of message from the definition set given before.

$$\frac{\text{SysApp.}\textit{getType} = \text{t, t} \in \{\text{ "text", "audio", "video"}\}}{< \texttt{read\_type}, < \sigma, \varepsilon >> \rightarrow < \sigma[type \rightarrow t], \varepsilon >}$$

- `read_dim` : this *get*-function returns the value of dim in SysApp, reading from the σ. This function gives backs the dimension of message in bytes.

$$\frac{\text{SysApp.}\textit{getDim} = \text{d, d} \in \mathbb{N}}{< \texttt{read\_dim}, < \sigma, \varepsilon >> \rightarrow < \sigma[dim \rightarrow d], \varepsilon >}$$

To analyse the operation of the application we need to be able to access memory locations extern of the application itself. To do this we also require the functions `read_conn`, `read_sign`, `read_batt`. These functions belong to the set of *statements*, but they don't affect the application store σ since they are functions related to the mobile device variables.

- `read_conn` : this *get*-function returns the value of $conn$ in SysDev, reading from the ε.

$$\frac{\text{SysDev.}\textit{getConn} = \text{c, c} \in \{\text{ "on", "off", "idle", "run"}\}}{< \texttt{read\_conn} \ , < \sigma, \varepsilon >> \rightarrow < \sigma, \varepsilon[conn \rightarrow c] >}$$

- `read_batt` : this *get*-function returns the value of $batt$ in SysDev, reading

from the ε.

$$\frac{\text{SysDev.}\textit{getBatt} = \mathsf{b},\ \mathsf{b} \in [0, 100]}{< \texttt{read\_batt}\ , < \sigma, \varepsilon >> \rightarrow < \sigma, \varepsilon[batt \rightarrow b] >}$$

- `read_sign` : this *get*-function returns the value of $sign$ in SysDev, reading from the ε.

$$\frac{\text{SysDev.}\textit{getSign} = \mathsf{s},\ \mathsf{s} \in [0, 100]}{< \texttt{read\_sign}\ , < \sigma, \varepsilon >> \rightarrow < \sigma, \varepsilon[sign \rightarrow s] >}$$

We also need to implement a set of functions that interact with the device and application environments. These *set*-function are going to change the values in the σ and ε:

- `write_app` : this *set*-function write the value in var in SysApp.

$$\frac{\text{SysApp.}\textit{set(var, } \sigma\textit{(v))}}{< \texttt{write\_app}(var, v), < \sigma, \varepsilon >> \rightarrow < \sigma, \varepsilon >}$$

- `write_dev` : this *set*-function write the value in var in SysDev.

$$\frac{\text{SysDev.}\textit{set(var, } \varepsilon\textit{(v))}}{< \texttt{write\_dev}(var, v), < \sigma, \varepsilon >> \rightarrow < \sigma, \varepsilon >}$$

CHAPTER 4

ABSTRACTION

As explained in Chapter 1, the signal scanning and the exchange of data in certain signal condition entail an increase in energy consumption. For this reason, it will create an algorithm that blocks momentarily sending data in non-optimal network conditions. In this chapter, we define the abstract properties focused on energy consumption, so we will have a more accurate view of our field of interest, ignoring variables outside of our analysis.

According to a standard definition due to its founder [8]: Abstract interpretation is a general theory to approximate the semantics of discrete dynamical systems. The basic idea are very simple: we know that programs are functions that perform over a C domain, called concrete domain. Abstract interpretation consists to run the program in an simpler A domain, called abstract domain that contains less information, only that necessary to be able to answer the question of interest. The study of these notions need to determine a partial order on the elements of the domain. To do this we assume that both the concrete and abstract domain are complete lattice. In our case we define Concrete domain as $\langle \wp(\texttt{States}^*), \subseteq \rangle$ and the property that more accurately describes the behaviour of our system is $\phi \in \Phi$. Set of traces are approximated and represented by suitable abstract elements, which capture interesting properties while disregarding other execution properties that

are out of the scope of interest. Abstract properties belong to the A domain and they are ordered according to $\sqsubseteq$, as said, $\langle A, \sqsubseteq \rangle$ is a complete lattice. Furthermore, the relationship between the concrete and the abstract domain domain is formalised through the added functions α and γ. The function $\alpha \in [\wp(\texttt{States}^*) \to A]$ called the abstraction function, and $\alpha(\texttt{States}^*)$ represents the best approximation in A of the set of traces $\texttt{States}^*$. The function $\gamma \in [\wp(A \to \texttt{States}^*]$ called the concretization function, returns the set of traces that are captured by an abstract property. These functions are monotonic, indeed, the monotonicity reflects the precise relationship in the corresponding domain, i.e. having $c_1, c_2 \in C$ where $c_1 < c_2$, their abstraction through α preserves the relationship in the abstract domain $\alpha(c_1) < \alpha(c_2)$. Obviously we can apply the same property to γ. So, given $c \in C$ and $a \in A$, where $\alpha(c)$ is the abstract representation of c, and $\gamma(a)$ is the concrete representation of a. The relation $\alpha(c) \sqsubseteq a$, or vice versa, $c \leq \gamma(a)$ shows that a is a correct approximation of c in A, namely the abstraction process has led to the loss of information and consequently, the possible loss of precision. The abstraction and concretization functions must satisfy the following property:

$$\forall \mathbf{S} \in \wp(\texttt{States}^*).$$

$$\forall a \in A.$$

$$\alpha(\mathbf{S}) \sqsubseteq a \iff \mathbf{S} \subseteq \gamma(a)$$

so α and γ form a Galois connection between abstract and concrete domain:

$$\langle \wp(\texttt{States}^*), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \sqsubseteq \rangle$$

we can also define the abstract semantics of the system $S^\sharp[\![P]\!]$, over an abstract domain whom is linked to the concrete domain by a Galois connection

$$\alpha(S[\![P]\!]\phi) \sqsubseteq S^\sharp[\![P]\!]\alpha(\phi)^1$$

[1]Where $S[\![P]\!]$ is a generic semantic

This imposes that, when the properties encoded by a given abstract domain are considered, the abstract semantics $S^\sharp[\![P]\!]$ captures all the behaviours of the application P.

As shown in [6], given an abstract property $p^\sharp$, which explains a system specification, it can be apply to the system as $S^\sharp[\![P]\!]\alpha(\phi) \sqsubseteq p^\sharp$, and using the transitivity of $\sqsubseteq$ we will have $\alpha(S[\![P]\!]\phi) \sqsubseteq p^\sharp$.

Then, p is also true in all concrete execution $S[\![P]\!]\phi$ and this means that P respect the property p. In other words, the Galois connections helps us to transform the concrete domain, composed by the program traces, in the abstract domain, composed by the abstract properties defined to adapted the program.

Application of Power Consumption Analysis

Now we need to transform the analysis of energy consumption made in the previous chapter in abstract property. The trend of the energy consumption is proportional to the distance between the mobile device and the access point; this is translated as the level of the signal received from the smart phone. In order, to make the program efficient than the battery resource we have to define a threshold level beyond which the sending data is too power-hungry.

First, we have to analyse how an application sends a message. We started analysing Telegram[2], a cross-platform instant messaging application. Telegram was chosen because it is released under an open source license, and especially since the software was created with the intent to be fully customizable. To perform the analysis, we have shrunk the original code to adapt it to our needs; this our version of the application allows only sending text messages to simplify the analysis of the code. In order to automatically verify requirements, we must find a lower bound to the signal level during the sending operation of a system.

Definition 4.1.1. Let P be a program and $strength \in [\wp(\texttt{States}^*) \to \mathbb{N}]$ be the intensity of the Wifi signal, and $\langle \wp(\texttt{States}^*), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathbb{N}, \sqsubseteq \rangle$ a Galois connection where:

$$\alpha = \lambda\tau.\, min(\{strenght(\tau) \,|\, \tau \in T\})$$
$$\gamma = \lambda n.\, \{\tau \in \wp(\texttt{States}^*) \,|\, strenght \leq n\}$$

We can say that the system respect the power efficienty requirement $threshold$ if

$$\forall \phi_0 \subseteq \Phi. \quad \alpha(S[\![P]\!]\phi) \leq threshold$$

[2]https://telegram.org/

Using the previous definitions we get:

$$\alpha(A[\![\texttt{read\_sign}]\!]\varepsilon) \leq threshold$$

This definition can be used to set a threshold, beyond which the system inhibits the exchange of packets with the WiFi network. In order to automatically verify signal strength requirements, we must find a lower bound to signal read during the execution of a system.

Now we show the two algorithms that perform data sending. We notice in the 6th row, of the Adapted Algorithm, the `IF` statement, which allows us to decide whether to forward the packet according to the signal level.

Algorithm 1 Original algorithm

```
1: procedure SEND_MESSAGE
2:     if read_dim > 0 then
3:         count = read_dim / 4096
4:         a = 0
5:         while a ⇐ count do
6:             sendMessage(text[a])
7:             a = a + 1
```

Algorithm 2 Adapted algorithm

```
1: procedure SEND_MESSAGE
2:     if read_dim > 0 then
3:         count = read_dim / 4096
4:         a = 0
5:         while a ⇐ count do
6:             if read_sign > threshold then
7:                 sendMessage(text[a])
8:                 a = a + 1
```

The Analysis

Now we analyse the variables of our interest, predicting the trend of the intervals, where possible :

- `read_type` : as already said, in our version of the software the only type of message is textual
- `read_dim` : the message size, in bytes, will be gradually decreased each submission, it follows that the interval will have a fixed upper bound dim_{MAX} and a variable lower bound dim_{min}
- `read_conn` : assuming that it is always connected to a WiFi network, the state of connection can switch between idle and run
- `read_batt` : assuming that initial battery state is fully charged, we have a fixed upper bound $batt_{MAX}$ and a decreasing lower bound $batt_{min}$. How we are going to analyse, the rate of decrease of charging depends of throughput and signal strength [16]
- `read_sign` : the signal level is inversely proportional to the distance between the device and the access point. In the case in which it moves, the interval can be seen as a stochastic process, which varies randomly according to the displacement of the device relative to the WiFi network.

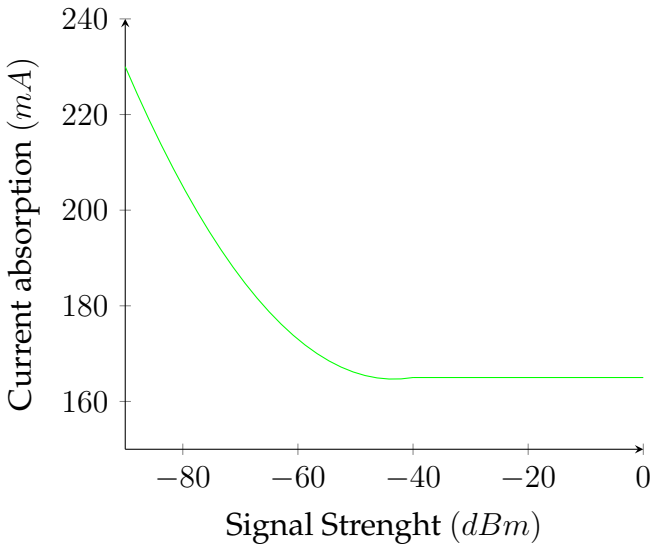


Figure 4-1: Current consumption related to the WiFi signal strength

Adapting the results of [11] to our study, we discover that the energy consumption of the data transfer does not follow a linear trend compared to the level of the signal strength. So, from 0 dBm to -40 dBm, we define the absorption of current constant at 165 mA. With a signal level lower than -40 dbm the consumption will be quadratic; approximated to $(0.03x^2 + 2.6x + 221)$ mA, where x is the signal strength in dBm.

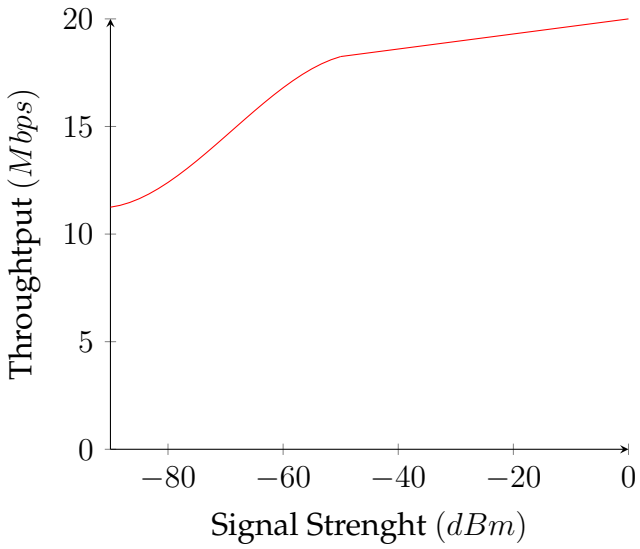


Figure 4-2: Transfer rate using a 20Mbps connection

At this point, we know how much is the current consumption for the transmission of data, now we have to understand how long the transmission of the data is in the wireless channel, in order to determine the electric charge so that it can be directly subtracted to the value of the battery charge. Thanks to the data provided by to [2] we are able to interpolate a function $-0.00015x^3 - 0.031x^2 - 1.9x - 18$ from -90dBm to -50dBm and $0.035 * x + 20$ from -50dBm to 0dBm that allows us to approximate the transfer rate compared to the wifi signal strength.

It is important to say that the results found here are not universal, but as we will see in the experimental tests, they may vary from device. The models defined are guides that show us how will be the trend of the values that we are analyzing.

We define three thresholds of signal strength:

- **Good** : higher than $-60dBm$
- **Fair** : between $-60dBm$ and $-80dBm$
- **Poor** : lower than $-80dBm$

To make our job easier, the Java code has been faithfully converted to pseudo-code, maintaining consistency with the original one and respecting the semantics previously defined; moreover, statements were added to help us conduct our analysis

Algorithm 3 Original algorithm

```
1:  procedure SEND_MESSAGE(text)
2:      read_batt
3:      if read_dim > 0 then
4:          count = read_dim / 4096
5:          a = 0
6:          while a ⇐ count do
7:              read_sign
8:              sendMessage(text[a])
9:              read_batt
10:             a = a + 1
11:             write_app(dim, read_dim − 4096)
```

From a first informal analysis we see that the variables that we are going to analyse are $batt$, dim and $sign$; $count$ and a are not so relevant to be analysed in this context. The original algorithm divides the message into substring of 4kBytes, so in our simulation we use a text message of 10kBytes, in order to have more than one cycle within the `while`-block. The states of the program are $\texttt{States} = \{2,3,4,5,6,7,8,9,10,11\} \times \Sigma$, and the transitions are:

$$\tau_0 = \{< 1, \phi > \rightarrow < 2, \phi >\}$$

$$\tau_1 = \{< 2, \phi > \rightarrow < 3, < \sigma, \varepsilon[batt \rightarrow b] >>\}$$

$$\tau_2 = \{< 3, \phi > \rightarrow < 4, \phi >\}$$

$$\tau_3 = \{< 4, \phi > \rightarrow < 5, \sigma[dim \rightarrow d], \varepsilon >\}$$

$$\tau_4 = \{< 5, \phi > \rightarrow < 6, \phi >\}$$

$$\tau_5 = \{< 6, \phi > \rightarrow < 7, \phi >\}$$

$$\tau_6 = \{< 7, \phi > \rightarrow < 8, < \sigma, \varepsilon[sign \rightarrow s] >>\}$$

$$\tau_7 = \{< 8, \phi > \rightarrow < 9, \phi >\}$$

$$\tau_8 = \{< 9, \phi > \rightarrow < 10, < \sigma, \varepsilon[batt \rightarrow b] >>\}$$

$$\tau_9 = \{< 10, \phi > \rightarrow < 11, \phi >\}$$

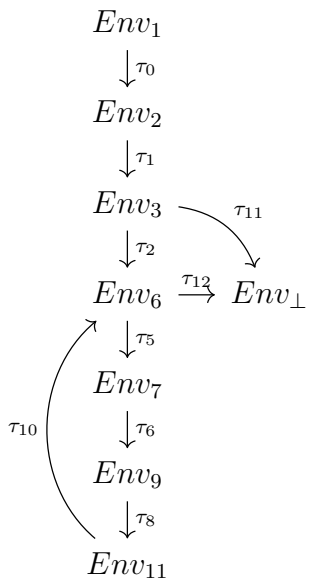
$$\tau_{10} = \{< 11, \phi > \rightarrow < 6, \sigma[dim \rightarrow dim - 4096], \varepsilon >\}$$

$$\tau_{11} = \{< 3, \phi > \rightarrow <\perp, \phi >\}$$

$$\tau_{12} = \{< 6, \phi > \rightarrow <\perp, \phi >\}$$

Now, we define the evolution of the program in terms of the domain of the variables taken into account, specifying Env_i as the set of environments at program point i and $Env_\perp$ is the environment at the end of the procedure.

$$
\begin{aligned}
Env_1 &: \perp \\
Env_2 &: Env_1 \sqcup \varepsilon[batt \to b] \\
Env_3 &: Env_2 \sqcup (\sigma[dim \to d] \cap (0, \infty]) \\
Env_4 &: Env_3 \\
Env_5 &: Env_4 \\
Env_6 &: Env_5 \sqcup Env_{11} \\
Env_7 &: Env_6 \sqcup \varepsilon[sign \to s] \\
Env_8 &: Env_7 \\
Env_9 &: Env_8 \sqcup \varepsilon[batt \to b] \\
Env_{10} &: Env_9 \\
Env_{11} &: Env_{10} \sqcup \sigma[dim \to dim - 4096] \\
Env_\perp &: Env_6 \sqcap (Env_3 \sqcap (\sigma[dim \to d] \sqcap [-\infty, 0]))
\end{aligned}
$$



On the right side there is the representation, in the form of graph, of the program seen from the side of the environment; besides the states, which do not alter the status of the variables taken into consideration, have been simplified.

Analysing a simulated execution, it follows that, given that the message to be sent is 10kBytes, the `IF` statement is true, and the `WHILE` loop run for 3 iterations. Using a 20 Mbps connection, we define three different scenarios, receiving an intensity of signal equal to -40 dBm, -65 dBm and -85dBm. In this way we obtain, thanks to the previously defined functions, three electric charge consumptions (EC) every time 4096 bytes are sent.

- $Throughput_{-40dBm} = 16.42Mbps$ and $EC_{-40dBm} = 3.3\mu Ah$
- $Throughput_{-65dBm} = 10.17Mbps$ and $EC_{-65dBm} = 5.5\mu Ah$
- $Throughput_{-85dBm} = 8.7Mbps$ and $EC_{-85dBm} = 8\mu Ah$

Although there is a significant difference between the different areas, the battery capacity is much greater than the amount of charge absorbed in worst transmission to not make clearly perceptible the variation of the battery charging percentage. To bypass this problem, assign a discharge percentage point in the case of good signal sending, two point in case of fair signal and finally 3 point in case of poor signal.

Now we proceed to analyse a performance in the case of good signal.

Env	τ	**dim**	**sign**	**batt**
Env_1	τ_0	$\perp$	$\perp$	$\perp$
Env_2	τ_1	$\perp$	$\perp$	$[100, 100]$
Env_3	τ_2	$[10000, 10000]$	$\perp$	$[100, 100]$
Env_6	τ_5	$[10000, 10000]$	$\perp$	$[100, 100]$
Env_7	τ_6	$[10000, 10000]$	$[-40, -40]$	$[100, 100]$
Env_9	τ_8	$[10000, 10000]$	$[-40, -40]$	$[99, 100]$
Env_{11}	τ_{10}	$[5904, 10000]$	$[-40, -40]$	$[99, 100]$
Env_6	τ_5	$[5904, 10000]$	$[-40, -40]$	$[99, 100]$
Env_7	τ_6	$[5904, 10000]$	$[-40, -40]$	$[99, 100]$
Env_9	τ_8	$[5904, 10000]$	$[-40, -40]$	$[98, 100]$
Env_{11}	τ_{10}	$[1808, 10000]$	$[-40, -40]$	$[98, 100]$
Env_6	τ_5	$[1808, 10000]$	$[-40, -40]$	$[98, 100]$
Env_7	τ_6	$[1808, 10000]$	$[-40, -40]$	$[98, 100]$
Env_9	τ_8	$[1808, 10000]$	$[-40, -40]$	$[97, 100]$
Env_{11}	τ_{10}	$[-2288, 10000]$	$[-40, -40]$	$[97, 100]$
Env_6	τ_{12}	$[-2288, 10000]$	$[-40, -40]$	$[97, 100]$
$Env_\perp$	$\tau_\perp$	$[-2288, 10000]$	$[-40, -40]$	$[97, 100]$

Table 4.1: Execution at -40 dBm

Now we proceed to analyse a performance in the case of fair signal.

Env	τ	**dim**	**sign**	**batt**
Env_1	τ_0	$\bot$	$\bot$	$\bot$
Env_2	τ_1	$\bot$	$\bot$	$[100, 100]$
Env_3	τ_2	$[10000, 10000]$	$\bot$	$[100, 100]$
Env_6	τ_5	$[10000, 10000]$	$\bot$	$[100, 100]$
Env_7	τ_6	$[10000, 10000]$	$[-65, -65]$	$[100, 100]$
Env_9	τ_8	$[10000, 10000]$	$[-65, -65]$	$[98, 100]$
Env_{11}	τ_{10}	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env_6	τ_5	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env_7	τ_6	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env_9	τ_8	$[5904, 10000]$	$[-65, -65]$	$[96, 100]$
Env_{11}	τ_{10}	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env_6	τ_5	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env_7	τ_6	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env_9	τ_8	$[1808, 10000]$	$[-65, -65]$	$[94, 100]$
Env_{11}	τ_{10}	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$
Env_6	τ_{12}	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$
$Env_\bot$	$\tau_\bot$	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$

Table 4.2: Execution at -65 dBm

Now we proceed to analyse a performance in the case of poor signal.

Env	τ	**dim**	**sign**	**batt**
Env_1	τ_0	$\perp$	$\perp$	$\perp$
Env_2	τ_1	$\perp$	$\perp$	$[100, 100]$
Env_3	τ_2	$[10000, 10000]$	$\perp$	$[100, 100]$
Env_6	τ_5	$[10000, 10000]$	$\perp$	$[100, 100]$
Env_7	τ_6	$[10000, 10000]$	$[-85, -85]$	$[100, 100]$
Env_9	τ_8	$[10000, 10000]$	$[-85, -85]$	$[97, 100]$
Env_{11}	τ_{10}	$[5904, 10000]$	$[-85, -85]$	$[97, 100]$
Env_6	τ_5	$[5904, 10000]$	$[-85, -85]$	$[97, 100]$
Env_7	τ_6	$[5904, 10000]$	$[-85, -85]$	$[97, 100]$
Env_9	τ_8	$[5904, 10000]$	$[-85, -85]$	$[94, 100]$
Env_{11}	τ_{10}	$[1808, 10000]$	$[-85, -85]$	$[94, 100]$
Env_6	τ_5	$[1808, 10000]$	$[-85, -85]$	$[94, 100]$
Env_7	τ_6	$[1808, 10000]$	$[-85, -85]$	$[94, 100]$
Env_9	τ_8	$[1808, 10000]$	$[-85, -85]$	$[91, 100]$
Env_{11}	τ_{10}	$[-2288, 10000]$	$[-85, -85]$	$[91, 100]$
Env_6	τ_{12}	$[-2288, 10000]$	$[-85, -85]$	$[91, 100]$
$Env_\perp$	$\tau_\perp$	$[-2288, 10000]$	$[-85, -85]$	$[91, 100]$

Table 4.3: Execution at -85 dBm

We analyse now the adapted algorithm. It is similar to the original one, the only difference is the `IF` statement contained in the `WHILE` loop, which checks that the signal level is above a certain threshold. It was kept the same notation used in the previous algorithm.

Algorithm 4 Adapted algorithm

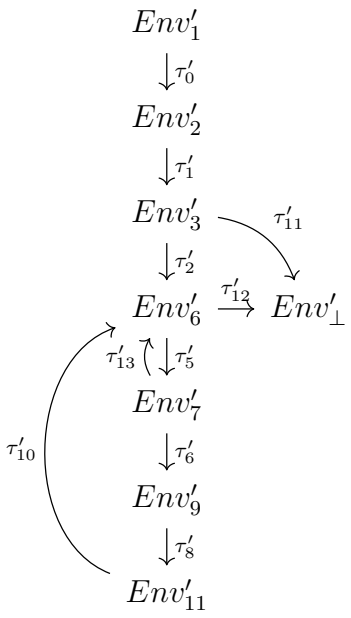
```
1:  procedure SEND_MESSAGE(text)
2:      read_batt
3:      if read_dim > 0 then
4:          count = read_dim / 4096
5:          a = 0
6:          while a ⇐ count do
7:              if read_sign > Threshold then
8:                  sendMessage(text[a])
9:                  read_batt
10:                 a = a + 1
11:                 write_app(dim, read_dim − 4096)
```

The states of the adapted program are $\texttt{States} = \{2,3,4,5,6,7,8,9,10,11\} \times \Sigma$, and the transitions are:

$$\tau'_0 = \{< 1, \phi > \rightarrow < 2, \phi >\}$$
$$\tau'_1 = \{< 2, \phi > \rightarrow < 3, < \sigma, \varepsilon[batt \rightarrow b] >>\}$$
$$\tau'_2 = \{< 3, \phi > \rightarrow < 4, \phi >\}$$
$$\tau'_3 = \{< 4, \phi > \rightarrow < 5, \sigma[dim \rightarrow d], \varepsilon >\}$$
$$\tau'_4 = \{< 5, \phi > \rightarrow < 6, \phi >\}$$
$$\tau'_5 = \{< 6, \phi > \rightarrow < 7, \phi >\}$$
$$\tau'_6 = \{< 7, \phi > \rightarrow < 8, < \sigma, \varepsilon[sign \rightarrow s] >>\}$$
$$\tau'_7 = \{< 8, \phi > \rightarrow < 9, \phi >\}$$
$$\tau'_8 = \{< 9, \phi > \rightarrow < 10, < \sigma, \varepsilon[batt \rightarrow b] >>\}$$
$$\tau'_9 = \{< 10, \phi > \rightarrow < 11, \phi >\}$$
$$\tau'_{10} = \{< 11, \phi > \rightarrow < 6, \sigma[dim \rightarrow dim - 4096], \varepsilon >\}$$
$$\tau'_{11} = \{< 3, \phi > \rightarrow < \perp, \phi >\}$$
$$\tau'_{12} = \{< 6, \phi > \rightarrow < \perp, \phi >\}$$
$$\tau'_{13} = \{< 7, \phi > \rightarrow < 6, < \sigma, \varepsilon[sign \rightarrow s] >>\}$$

The evolution of the adapted program is the following.

$$
\begin{aligned}
&Env'_1 : \bot \\
&Env'_2 : Env'_1 \sqcup \varepsilon[batt \to b] \\
&Env'_3 : Env'_2 \sqcup (\sigma[dim \to d] \sqcap (0, \infty]) \\
&Env'_4 : Env'_3 \\
&Env'_5 : Env'_4 \\
&Env'_6 : Env'_5 \sqcup Env'_{11} \\
&Env'_7 : Env'_6 \sqcup (\varepsilon[sign \to s] \sqcap (Threshold, \infty]) \\
&Env'_8 : Env'_7 \\
&Env'_9 : Env'_8 \sqcup \varepsilon[batt \to b] \\
&Env'_{10} : Env'_9 \\
&Env'_{11} : Env'_{10} \sqcup \sigma[dim \to dim - 4096] \sqcap (\varepsilon[sign] \sqcap [-\infty, Threshold]) \\
&Env'_{\bot} : Env'_6 \sqcap (Env'_3 \sqcap (\sigma[dim \to d] \sqcap [-\infty, 0]))
\end{aligned}
$$



The threshold is a fundamental parameter in our analysis, in fact a too low threshold level would allow the sending of data in case of low signal level, thus increasing the battery power consumption and making the optimisation useless. On the other hand, a too high threshold level would require a high signal level in order to send data, reducing the user experience in areas where the signal is weak.

In this version the `if` statement inhibits sending data if it doesn't respects the condition, so it is interesting to analyse a simulated use with the moving device. To highlight the effectiveness of the algorithm, the study is conducted with a user that sends data while passes from a state of **fair** signal ($-65dBm$), to a **poor** signal ($-85dBm$), then returns to a state of **good** signal to complete the sending. In order to do this, the threshold is set to -75 dBm.

Env	τ'	**dim**	**sign**	**batt**
Env'_1	τ'_0	$\perp$	$\perp$	$\perp$
Env'_2	τ'_1	$\perp$	$\perp$	$[100, 100]$
Env'_3	τ'_2	$[10000, 10000]$	$\perp$	$[100, 100]$
Env'_6	τ'_5	$[10000, 10000]$	$\perp$	$[100, 100]$
Env'_7	τ'_6	$[10000, 10000]$	$[-65, -65]$	$[100, 100]$
Env'_9	τ'_8	$[10000, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_{11}	τ'_{10}	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_6	τ'_5	$[5904, 10000]$	$[-85, -65]$	$[98, 100]$
Env'_7	τ'_{13}	$[5904, 10000]$	$[-85, -65]$	$[98, 100]$
Env'_6	τ'_5	$[5904, 10000]$	$[-85, -65]$	$[98, 100]$
Env'_7	τ'_{13}	$[5904, 10000]$	$[-85, -65]$	$[98, 100]$
Env'_6	τ'_5	$[5904, 10000]$	$[-85, -65]$	$[98, 100]$
Env'_7	τ'_{13}	$[5904, 10000]$	$[-85, -65]$	$[98, 100]$
Env'_6	τ'_5	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_7	τ'_6	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_9	τ'_8	$[5904, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_{11}	τ'_{10}	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_6	τ'_5	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_7	τ'_6	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_9	τ'_8	$[1808, 10000]$	$[-65, -65]$	$[94, 100]$
Env'_{11}	τ'_{10}	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$
Env'_6	τ'_{12}	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$
$Env'_\perp$	$\tau'_\perp$	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$

Table 4.4: Execution with moving mobile

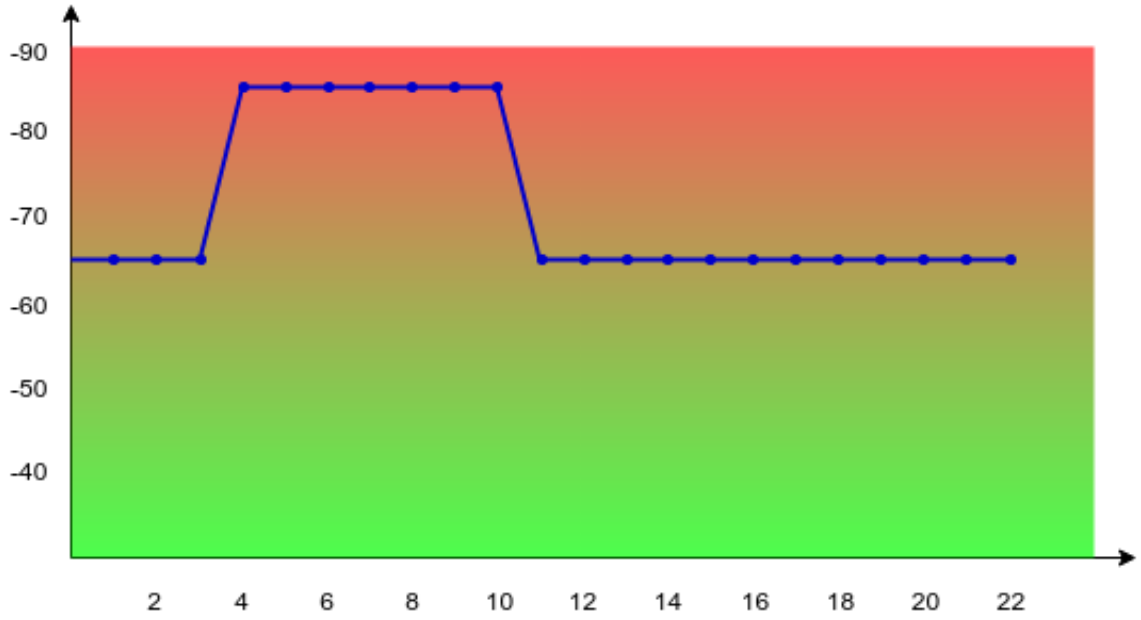


Figure 4-3: Signal strength for each Env

However, in the previous simulation, it was defined a priori path made from the mobile device, but in the real world the device is not always located beyond the defined signal threshold. So it is useful to define a lattice of infinite height. One way to ensure that the analysis ends is to find situations in which the lattice respects the ascending chain condition (and no-strictly ascending chain condition) at a given point in the program, so as to shorten the chain to a finite length. We can do so with a *widening operator*. Recalling the theory previously defined, we have that the widening operator (∇) is defined like:

$$[l_0, u_0]\nabla[l_1, u_1] = \begin{cases} \text{if } l_1 < l_0 \text{ then } -\infty & \text{else } l_0 \\ \text{if } u_0 < u_1 \text{ then } \infty & \text{else } u_0 \end{cases}$$

In example:

$$[-65, -65]\nabla[-85, -65] = [-\infty, -65]$$

We can use a smart version of this widening operator, with the insight that the bounds of a lattice are related to constants in the program.

$$[l_0, u_0]\nabla_{th}[l_1, u_1] = [min_{th}(l_0, l_1), max_{th}(u_0, u_1)]$$

where:

$$min_{th}(l_0, l_1) = \begin{cases} l_0 & \text{if} \quad l_0 \leq l_1 \\ max(Threshold, l_1) & \text{otherwise} \end{cases}$$

$$max_{th}(u_0, u_1) = \begin{cases} u_1 & \text{if} \quad u_1 \leq u_0 \\ min(u_1, Threshold) & \text{otherwise} \end{cases}$$

So if we have an ascending chain could bound the chain values within the values of the constants, so in our case the value of the signal threshold

$Env'_1 : \bot$

$Env'_2 : Env'_1 \sqcup \varepsilon[batt \to b]$

$Env'_3 : Env'_2 \sqcup (\sigma[dim \to d] \sqcap (0, \infty])$

$Env'_4 : Env'_3$

$Env'_5 : Env'_4$

$Env'_6 : Env'_5 \sqcup Env'_{11}$

$Env'_7 : Env'_6 \sqcup (\varepsilon[sign \to s] \sqcap (Threshold, \infty])\nabla_{th}(Env'_6)_{n-1}$

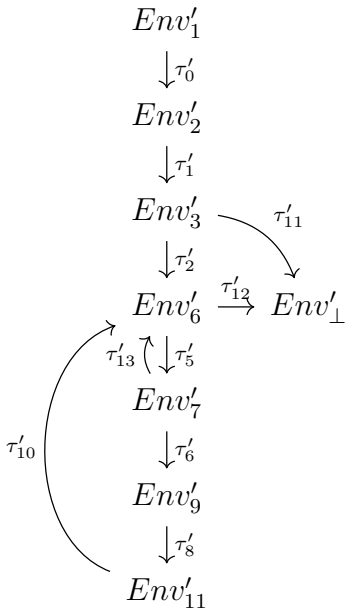
$Env'_8 : Env'_7$

$Env'_9 : Env'_8 \sqcup \varepsilon[batt \to b]$

$Env'_{10} : Env'_9$

$Env'_{11} : Env'_{10} \sqcup \sigma[dim \to dim - 4096] \sqcap (\varepsilon[sign] \sqcap [-\infty, Threshold])$

$Env'_\bot : Env'_6 \sqcap (Env'_3 \sqcap (\sigma[dim \to d] \sqcap [-\infty, 0]))$



Env	τ'	**dim**	**sign**	**batt**
Env'_1	τ'_0	$\bot$	$\bot$	$\bot$
Env'_2	τ'_1	$\bot$	$\bot$	$[100, 100]$
Env'_3	τ'_2	$[10000, 10000]$	$\bot$	$[100, 100]$
Env'_6	τ'_5	$[10000, 10000]$	$\bot$	$[100, 100]$
Env'_7	τ'_6	$[10000, 10000]$	$[-65, -65]$	$[100, 100]$
Env'_9	τ'_8	$[10000, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_{11}	τ'_{10}	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_6	τ'_5	$[5904, 10000]$	$[-75, -65]$	$[98, 100]$
Env'_7	τ'_{13}	$[5904, 10000]$	$[-75, -65]$	$[98, 100]$
Env'_6	τ'_5	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_7	τ'_6	$[5904, 10000]$	$[-65, -65]$	$[98, 100]$
Env'_9	τ'_8	$[5904, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_{11}	τ'_{10}	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_6	τ'_5	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_7	τ'_6	$[1808, 10000]$	$[-65, -65]$	$[96, 100]$
Env'_9	τ'_8	$[1808, 10000]$	$[-65, -65]$	$[94, 100]$
Env'_{11}	τ'_{10}	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$
Env'_6	τ'_{12}	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$
$Env'_\bot$	$\tau'_\bot$	$[-2288, 10000]$	$[-65, -65]$	$[94, 100]$

Table 4.5: Execution with narrowing operator with Threshold = -75

In this way we can shrink down the chain, ignoring the steps in which signal strength is below the threshold level.

We can see how static analysis has brought positive results for our research, shown a significant difference between the two snippets of code.

CHAPTER 5

SIMULATION

In this chapter we will define two models: the theoretical one, which creates a simulation that let us define our ideal system. On the other hand, the Empirical one uses the data collected from the device via an App, developed ad-hoc for our purpose. The final aim will be to compare the two models in order to verify whether there is a correlation between the intensity of the WiFi signal and the current consumption, and to be able to quantify the effective energy saving data from the analysis of the code of the previous chapter.

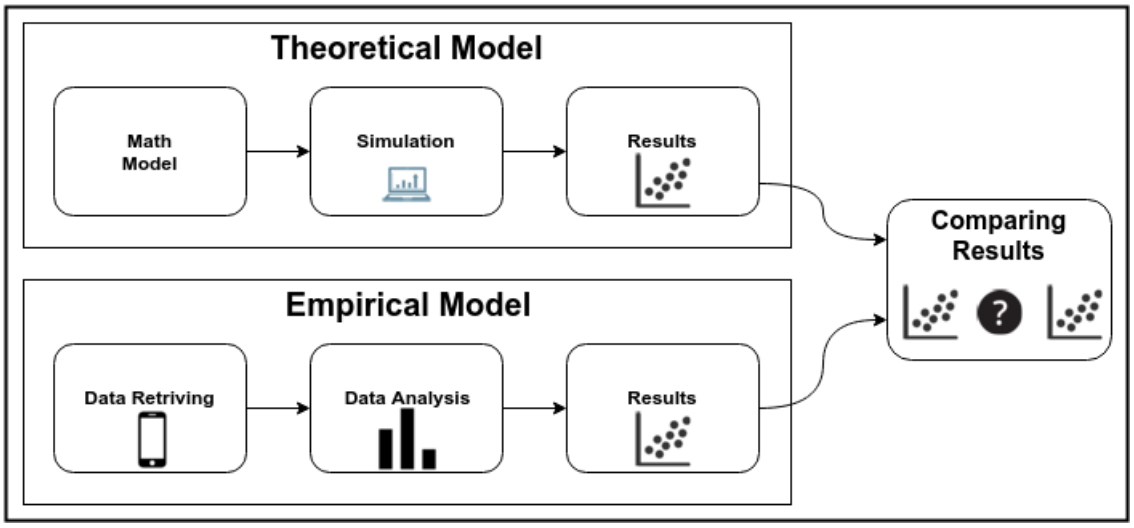


Figure 5-1: Work Flow Chart

Theoretical simulation

First of all, we need to define how the WiFi signal strength degenerate as you walk away from the Access Point. In network theory, the ratio between signal strength and distance from transmitter is governed by the FSLP law (free-space path loss); i.e., the attenuation of the power of an electromagnetic signal with respect to the end point without obstacles that may cause reflections or diffractions.

$$FSPL = \left(\frac{4\pi df}{c}\right)^2$$

where

1. d is the distance between transmitter and receiver
2. f is the frequnecy of the signal
3. c is constant of the speed of light in vacuum

Since the distance is measured in meters and the Wifi frequency is of the order of GHz, the formula is simplified into:

$$20log_{10}(d) + 20log_{10}(f) + 32,45$$

The final result of the attenuation is expressed in dB, to be consistent with the signal strength expressed in dBm. As an example, suppose you have a WiFi router $-20dBm$ power with $2.5GHz$ channel frequency, a 10 meter remote receiver suffer an attenuation of $20log_{10}(10) + 20log_{10}(2,5) + 32,45 = 60dB$; therefore, the device receives a signal intensity equal to $(-20dBm) - (60dB) = -80dBm$.

Once defined the model of the Access Point in the free space, assuming we know the route of a device in space, we can define the spatial trajectories. These trajectories define the path, inside of which the signal received by the device is variable. To simulate this process we use the random walk. The random walk method is a Markov process that formalises the idea of taking a series of steps in

pseudo-random directions. We define our case study as a two-dimensional random walk, where we will study the movements of a particle in space with stochastic behaviour with the following probabilities:

$$p_{EE} = 0.1875 \qquad p_{WW} = 0.0625 \qquad p_{NN} = 0.125$$
$$p_{SS} = 0.125 \qquad p_{NE} = 0.25 \qquad p_{SE} = 0.25$$

These probabilities were chosen to give a gait as much realistic as possible to simulate the walk of our user and, using the MatLab software, we implement a model that simulates the evolution in the space. Here an example of a simulation where the graph in question shows the path done by a user:

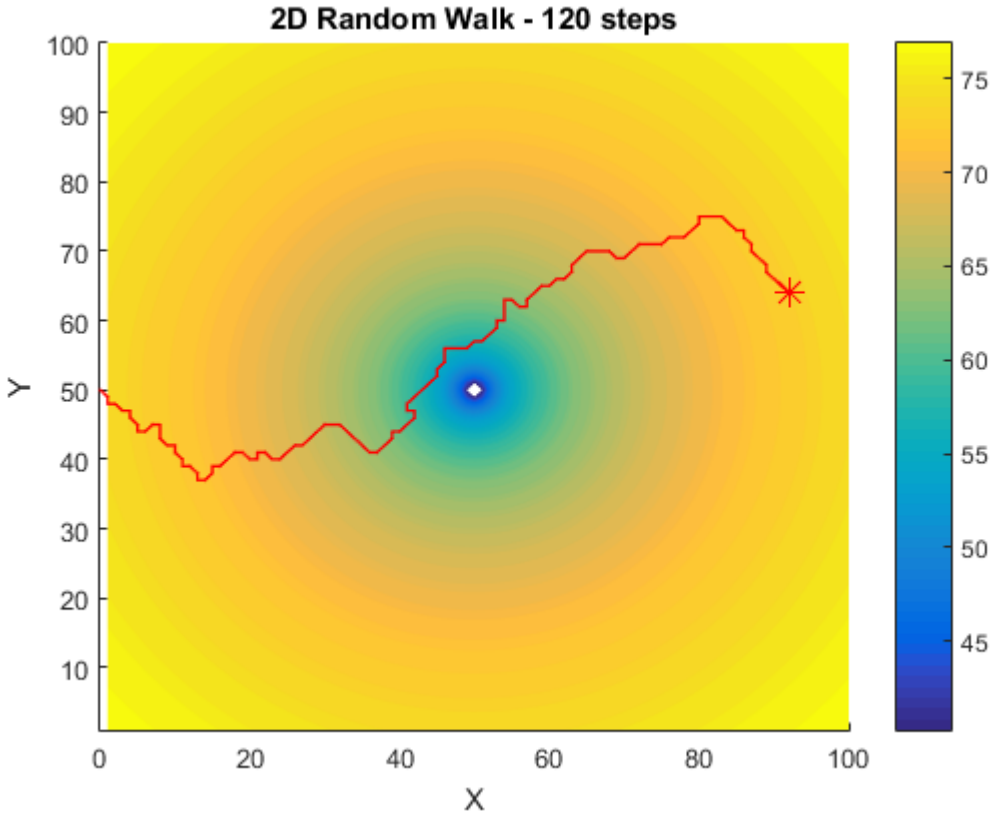


Figure 5-2: Example of a user's path

It is represented a square area, with side of 100 meters; in the central point (coordinates [50,50]) there is an ideal Access Point, simulated using the previously

defined model with a power equal to -10dBm and frequency of 2.5Ghz. We can see the continuous vectorial field that represents the different signal intensity, with the color gradient fade, starting with the most intense case at the center and degenerating as it moves away from it. Assuming that the user travels 120 steps from 1 meter each, starting from the point with coordinate [0,50] [1] and every step is sent a 4096 byte packet.

[1] the simulator does not allow to go beyond the limits of the defined space in question

Here we can see part[2] of the analysis of the simulated date made using the static analyser,

Env	τ'	**step**	**sign**	**batt**
Env'_1	τ'_0	1	$\perp$	$\perp$
Env'_2	τ'_1	1	$\perp$	$\perp$
Env'_3	τ'_2	1	$\perp$	$\perp$
Env'_6	τ'_5	1	$\perp$	$[100, 100]$
Env'_7	τ'_{13}	1	$[-75, -75]$	$[100, 100]$
Env'_6	τ'_5	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_7	τ'_6	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_9	τ'_8	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_{11}	τ'_{10}	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_6	τ'_5	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_7	τ'_6	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_9	τ'_8	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_{11}	τ'_{10}	47	$[-74, 7, -74, 7]$	$[96, 100]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Env'_9	τ'_8	81	$[-73, 7, -64, 9]$	$[28, 100]$
Env'_{11}	τ'_{10}	81	$[-73, 7, -64, 9]$	$[28, 100]$
Env'_6	τ'_5	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_7	τ'_6	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_9	τ'_8	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_{11}	τ'_{10}	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_6	τ'_5	83	$[-75, -75]$	$[26, 100]$
Env'_7	τ'_{13}	83	$[-75, -75]$	$[26, 100]$

Table 5.1: Part of analysis of the simulation with narrowing operator with Threshold = -75

We have we have an energy saving of about 63% compared to the case without a threshold.

[2]for the complete analysis please refer to Appendix C

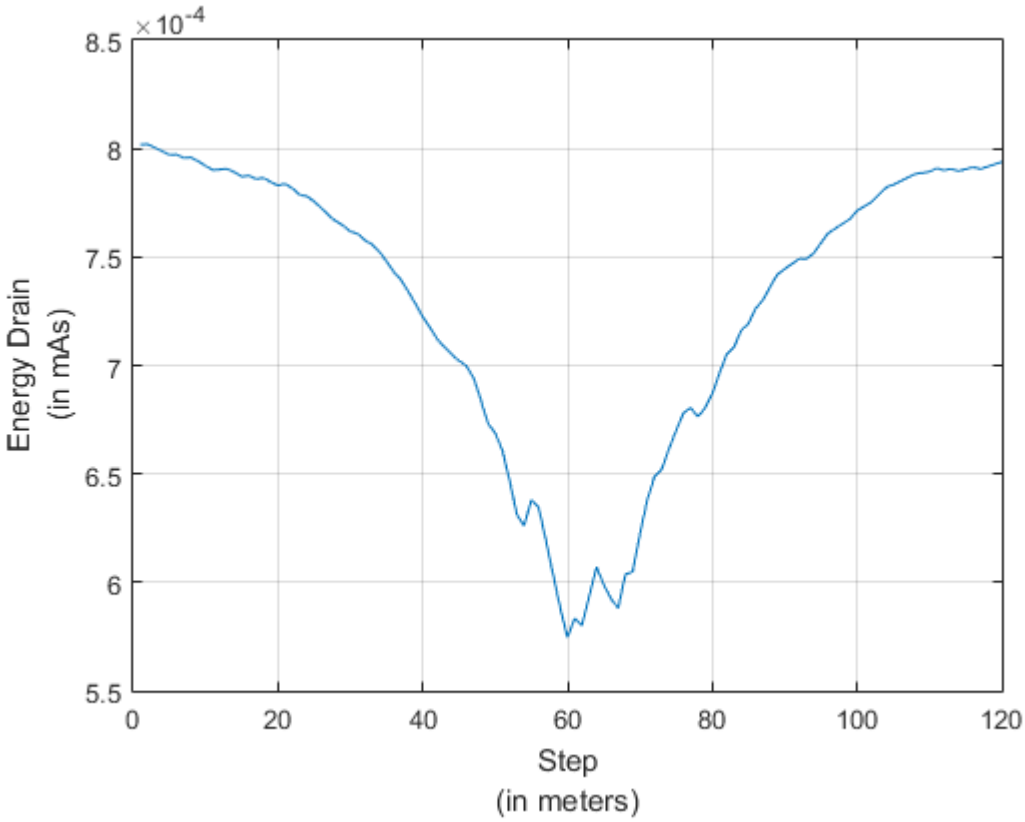


Figure 5-3: The energy drained for each packet sent in the previous example

The second plot correlates the intensity of punctual signal strength with the sending of the package, so as to obtain a curve that shows the energy absorbed for each submission, highlighting the difference in energy absorbed by varying the distance from the Access Point and consequently the intensity of signal. Based on of this result, we repeat the analysis applying a threshold and sweeping this from -90dBm to -50dBm (with step 1 dBm), iterating 100 times every value. By analyzing the samples obtained, the results are the following:

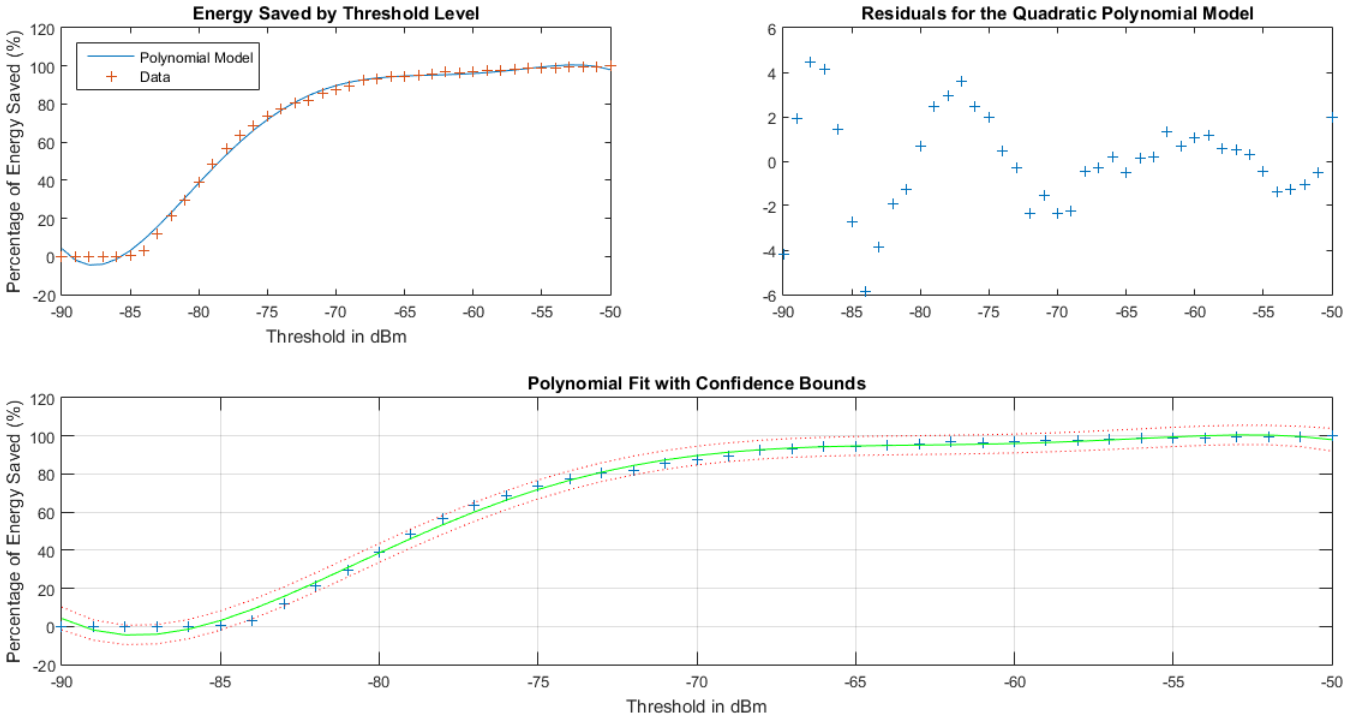


Figure 5-4

With $R^2 = 0.996$ and a correlation matrix of:

$$\begin{bmatrix} 1 & 0.9004 \\ 0.9004 & 1 \end{bmatrix}$$

It is possible to see how R^2 is very high (very near to the absolute maximum equal to 1), this implies that the results remain focused around their regression curve, minimizing the errors of it. So, we can use the curve

$$-6.304x^5 + 8.743x^4 + 18.95x^3 - 38.21x^2 + 24.27x + 89.69$$

where x is normalised by mean $\mu = -70$ and standard deviation $\sigma = 11.98$. Define a threshold to $-75dBm$ which will lead us to an energy saving of approximately 72% during a random walk over the same case without threshold, with a maximum error of about 5%.

Empirical simulation

Let as look at the real energy savings using a material device. Initially, we used OnePlus One smartphone, a latest generation device that uses the Cyanogen OS 13.0 operating system, based on Android 6.0.1. The approach used in this analysis is not the same as the theoretical method, in fact, we can not detect the energy absorbed from a single sending because of precision limitations of the methods made available by Android for this type of analysis, and also because of poor accuracy of the sensors of the same device. The technique used is to send a series of packets, with variable size, a large number of times, by saving for each sending the signal strength, the current consumed at that instant and the average frequency of 4 Core of the device in the moment of sending. In order to retrieve this data, we have developed an App that allows us to save the data in CSV format for more efficient analysis. In order to have a description as detailed as possible, the app in question has been implemented using the multi-threading technique; the main thread will execute sending packets, the second thread will be responsible of saving the data in a CSV file. It has been chosen to use this approach in order to synchronise the GET information in the exact instant of the data sending, increasing the accuracy. In order to obtain refined data, all services which are not necessary for the regular operation of the device were deactivated; it is important to add that it is not possible to kill all the background services (many of which are not even visible in the developer console). The remaining active services use CPU resources during the analysis, adding an error variable inside information obtained, moreover the errors are aleatory and not constant, so not removable. A first analysis was done by sending packets from 512 KB, 1024 KB, 1536 KB, 2048 KB, each one repeating sending 25 times [3]. This procedure was performed 2 times, the first time the device is located very close to the access point, with a fixed signal intensity of about -10dBm; the second time the device far from it, with a signal strength of about

[3]The packeges in question were chosen after a first analysis that showed how a package smaller than 512 KB and bigger 2048 KB would lead to a greater dispersion of data collected and did not highlight a markedly trend

-80dBm.

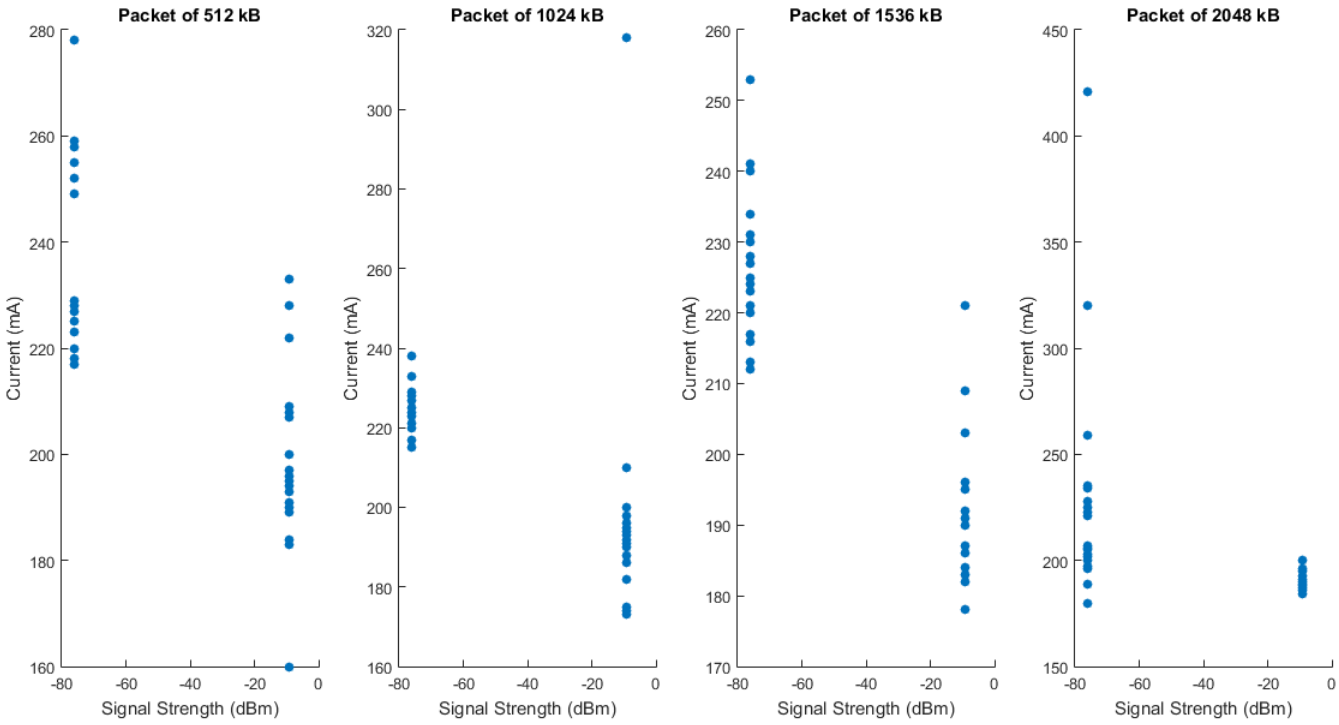
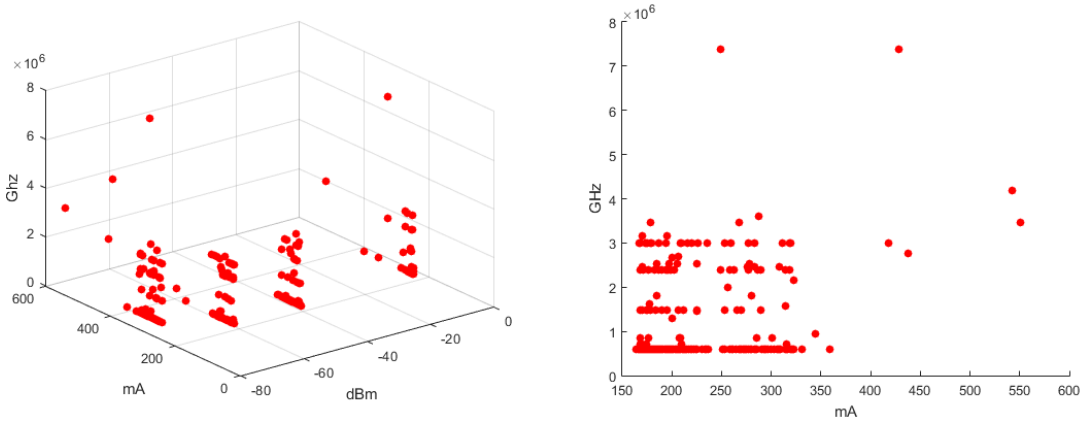


Figure 5-5: The sending of data divided by packet size

One can already see from these graphs a trend between the signal power and the current consumption for each packet size, however, we also see a large dispersion of data along the Y-axis, which denotes a error component. We choose to take the package to 1024kb which keeps data close to its average. We repeat the data retrieve procedure, this time keeping the packet size to 1024kB and increasing the number of iterations to 100. This time we repeat the procedure for 4 different signal intensities: approximately -80dBm, approximately -65dBm, approximately -45dBm and approximately -10dBm.

We analyze these packages to see if there is a correlation between current consumption and frequency of CPU.



(a) Frequency and Current by Signal Strength (b) Frequency and Current

Figure 5-6: Correlation between Current consumption and CPU Frequency

From the graph on the left we notice that there is no direct correlation between GHz and mA, to the varying of dBm. From the graph on the right we realise that there is no obvious trend. This does not determine a no direct correlation between CPU frequency and the current consumption, but it highlights the fact that we are not able to isolate the various components of the device so as to minimise the measurement error.

Then we analyze the relationship between signal strength and current consumption for each CPU frequency.

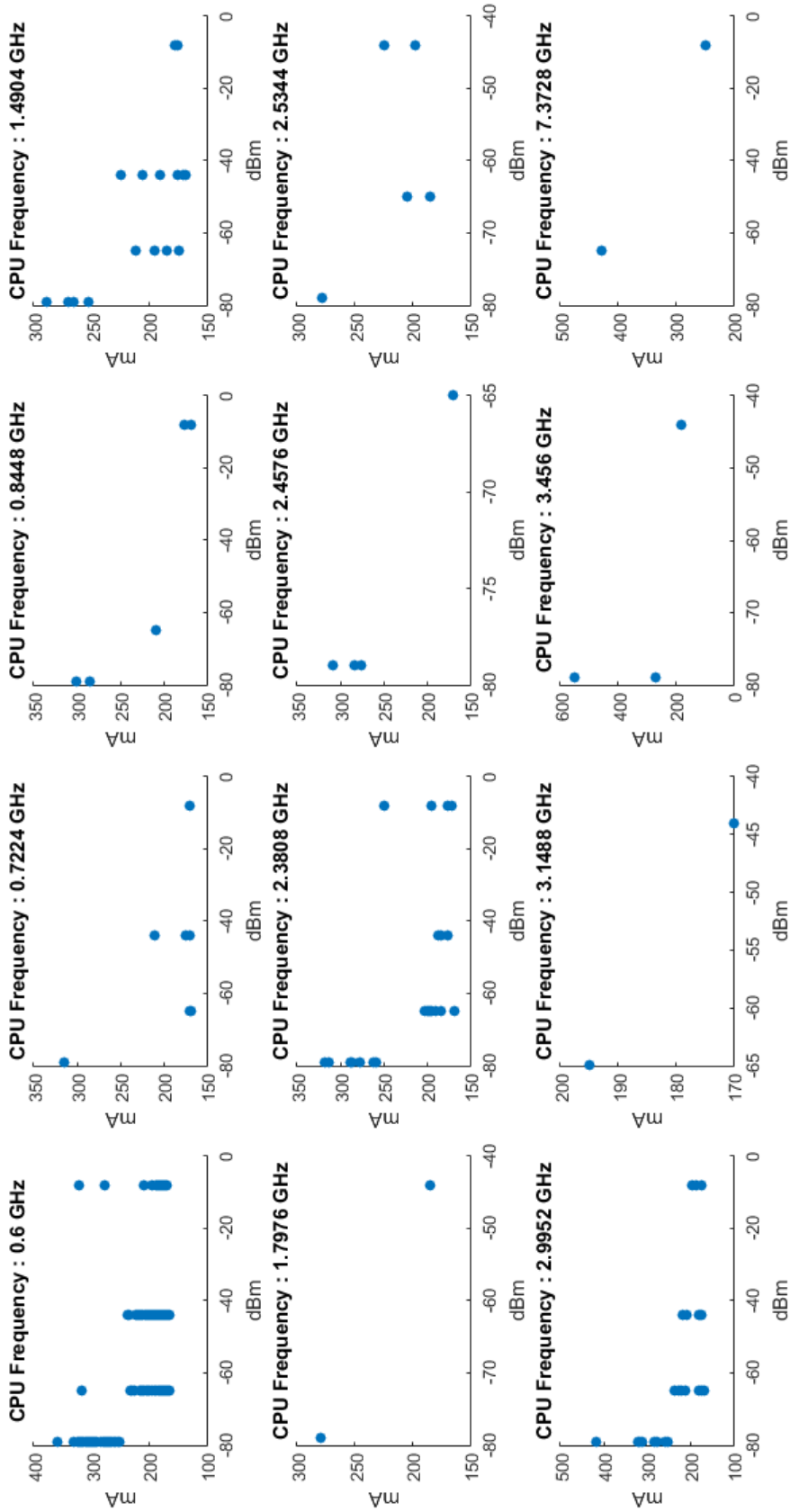


Figure 5-7: Current vs. Strength for each Frequency

We notice, mainly from charts with more measurements, an obvious trend which underlines the increasing of current delivered as the received signal decrease. Therefore, we are going to take the plot that has more occurrences of the same frequency and analyse it. The outliers (practically : the current peaks), that are below the 15 % and more than 85 % of the estimators distributed around a normal curve, were also removed.

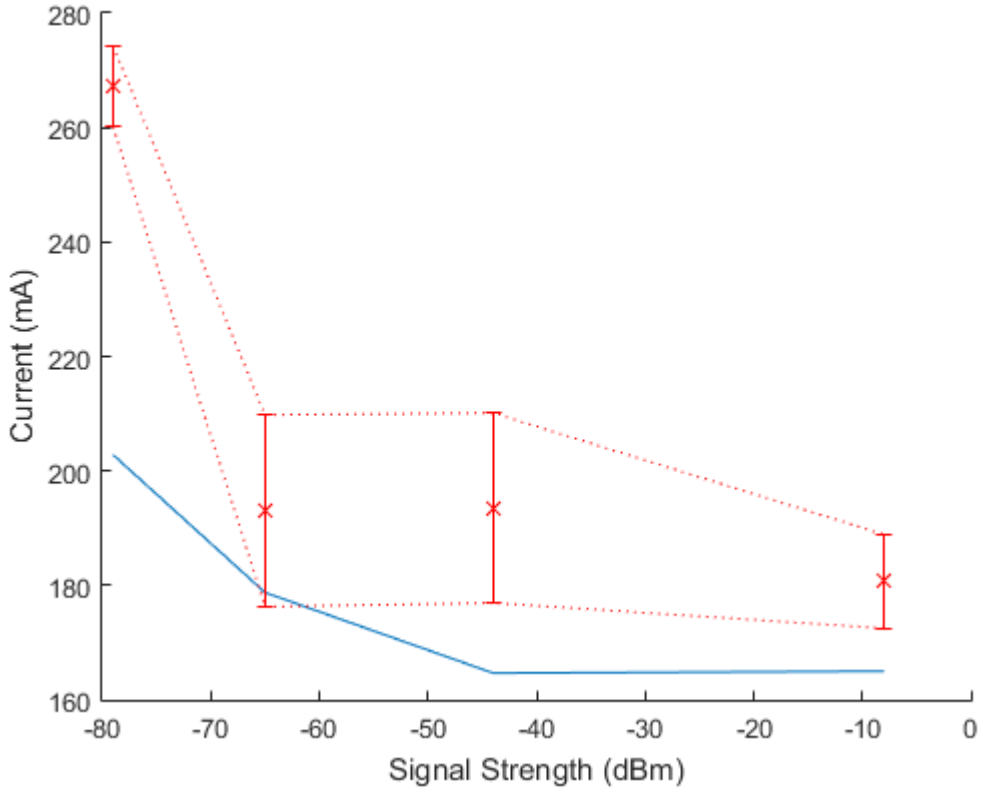


Figure 5-8: Refined data with standard deviation and theoretical curve

We can notice the blue curve represents the interpolation of the theoretical data, and the one in red represents the empirical data with their range of error. Let as see how the theoretical model is similar to the curve of the data, unless there is an additional constant given by the factors external to our analysis (ex. Screen).

Comparing Results

Once we have the two curves we can fit them between -90 and -10 dBm,

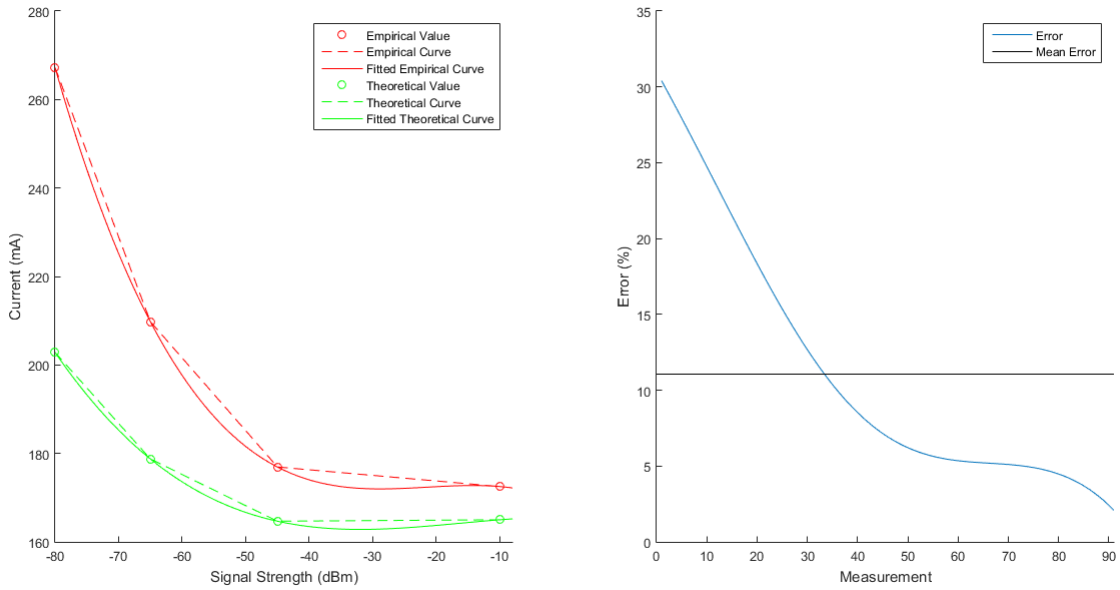


Figure 5-9: Comparing the Theoretical and Empirical curves

From the graph on the left we can see a similarity between the theoretical and empirical curve, they not entirely equal because, as pointed out, the real data above are affected by measurement errors which cannot be eliminated with the means at our disposal. From the graph on the right we can see how the error between the 2 curves increase when signal strength decrease; with an average error of circa 11%.

We can say that the results obtained, despite being affected by a great error component, have allowed us to highlight the trend which makes real sense of our research. Although we could not quantify the amount of energy expended by the device under low WiFi signal level conditions, we still found a strong correlation between lower received signal level and increase in current consumption. We are therefore satisfied with the results obtained which allow us to justify positively our analysis.

Further Tests

New tests were done to give more robustness to the thesis, using two Galaxy Tab 2 tablet with the original Android 5.0.2 operative system, a Huawei P8 Lite and a Samsung Galaxy S Advance. Initially this model was mounting Android 2.3 operating system, which can not support our application. It was then updated with Resurrection Remix, a custom ROM based on Android 5.1.1. This update not only allowed to install the application for testing, but it has also gives more accurate results; This due to the fact that this ROM is born with the intention to decrease the load on the CPU, eliminating all third-party applications and services not directly useful to the proper functioning of the mobile device.

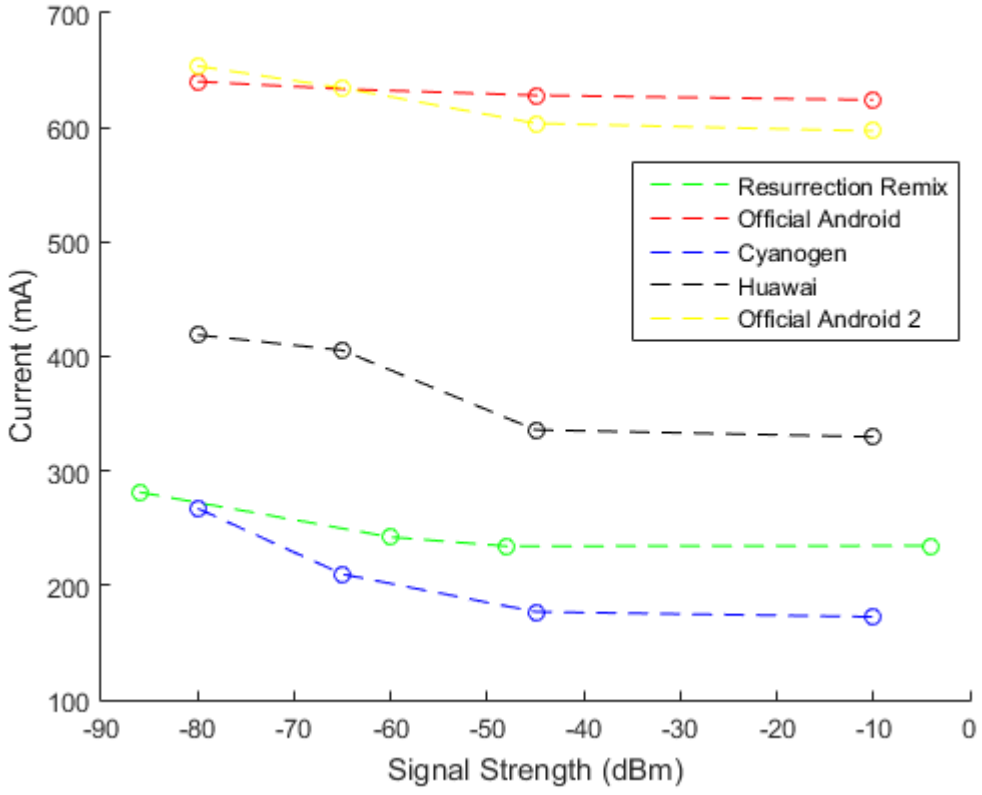


Figure 5-10: Current Consumption trends of each device

In Figure 5-10 we compare all 5 devices to show in an explicit manner the dif-

ferent trends of current absorption. We notice that the results obtained from two the Galaxy Tab 2 show a slight trend compared to the data obtained from the other devices; we highlight a high offset current values, due to the screen, which, being larger than the other (10.1 inch vs. 6 inch for the One Plus and Huawei, and ad 5 inch for the Samsung S Advance), will draw more current than the other devices. In the middle of the plot we can see the curve of the Huawei P8 Lite, this is the most power-hungry smartphone in our test, most likely due to the fact that this device is the most performant and the only one with a 4 core CPU working at 1.2GHz. Although the Resurrection Remix ROM is optimised to have a reduced energy consumption, in the graph it shows a higher current offset with respect to the Cyanogen OS curve. Most likely this higher current consumption is due to the forced update, so not all the hardware-components are optimised for this operating system.

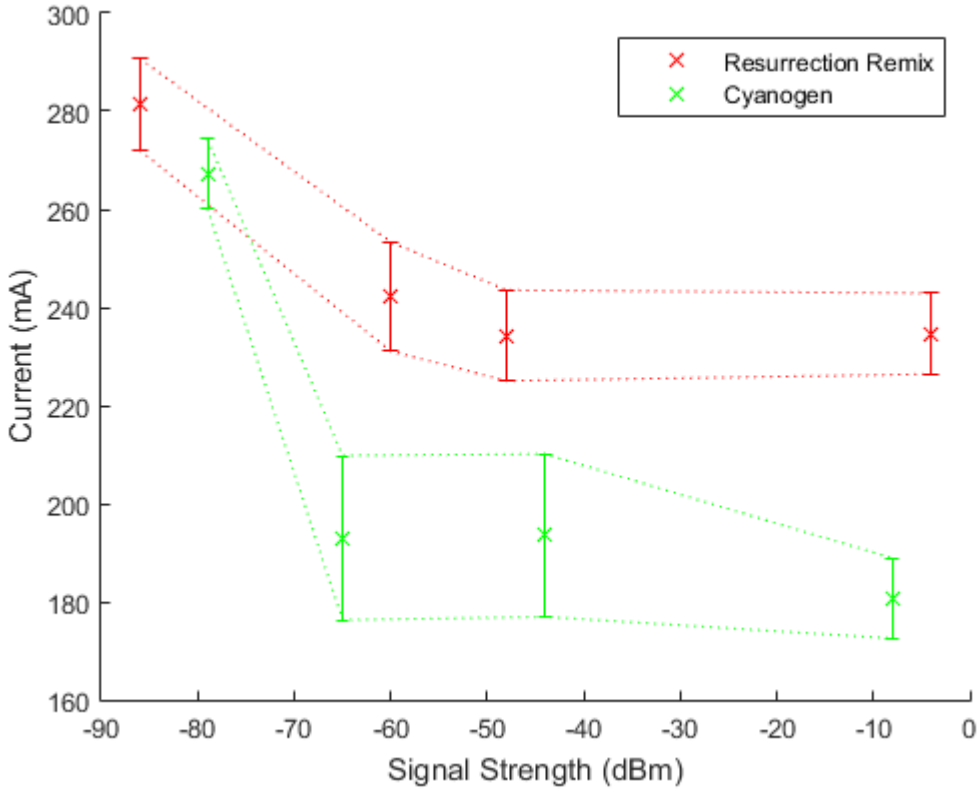


Figure 5-11: Value dispertion

Nevertheless, as seen in the Figure 5-11, the fewer applications on the background the lower the dispersion of values, with the measurements closer to the average for each value of signal intensity. Although the more accuracy Advance Galaxy S device in the data retrieving, we preferred use the One Plus device as example to study in deep the trend of power consumption since it comes closest to a use of an average user device. To improve the solidity of the thesis, they have been also used two equal tablet devices, in this way we are able to check the similarity of values between two device of the same model.

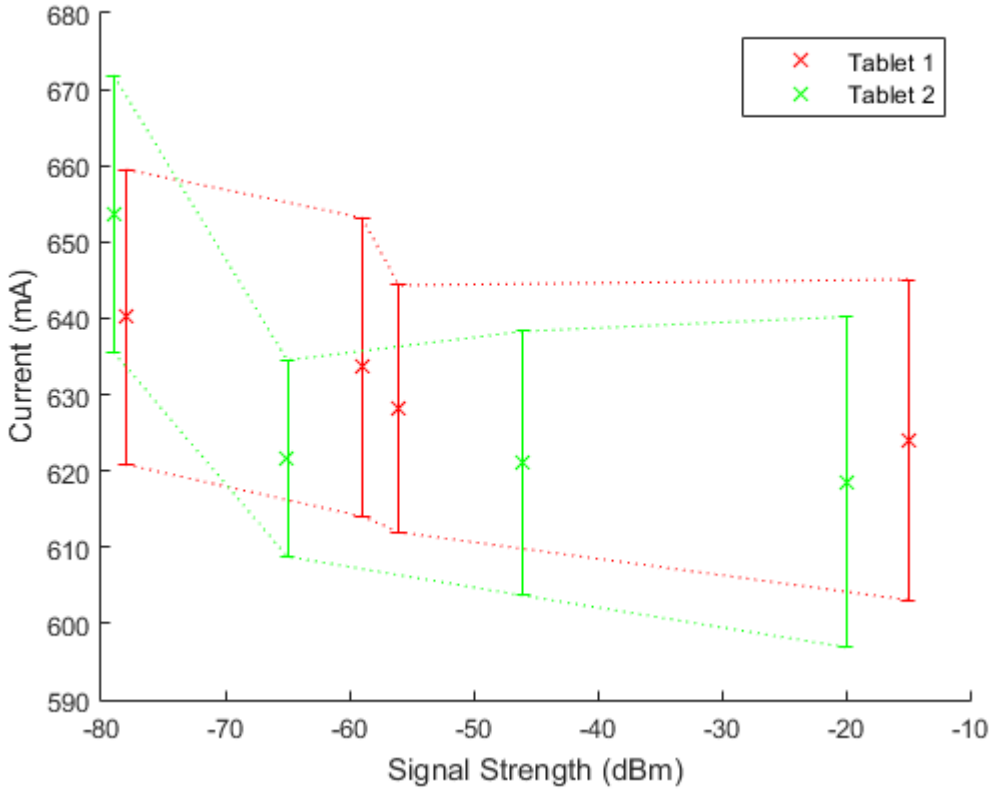


Figure 5-12: Comparing the values of the two tablets

In Figure 5-12, we can see how the dispersion, related to signal levels, is very similar for the two models; also we can see that the offset between the two devices is very similar, respectively 631 mA and 628 mA. We can assert that the two tablets have a very similar trend, ergo we can define the problem of specificity, not related to the individual device, but in a more extensive way, related to the device model.

Combining Analyses

In the final part of the simulation we are going to analyse, in a combined way, a sample path made by a user, using data collected from the device and we are going to analyse them by static analysis. In this example we are going to check the correctness of the adapted algorithm, using real data as input; after performing a series of 100 measurements into 8 points, we used the averages of the measurements to associate a current consumption for each signal level, discretized in 4 areas.

Signal (dBm)	**Current (mA)**	**Value**
-30	152	*GOOD*
-40	153	
-49	156	ALMOST
-54	161	GOOD
-60	167	FAIR
-67	177	
-75	189	*POOR*
-80	205	

Table 5.2: Signals strength per area

The table 5.2 defines the area in which the levels of the measured signals are contained. Now we can define the actual path performed in order to be able to analyze the power consumption.

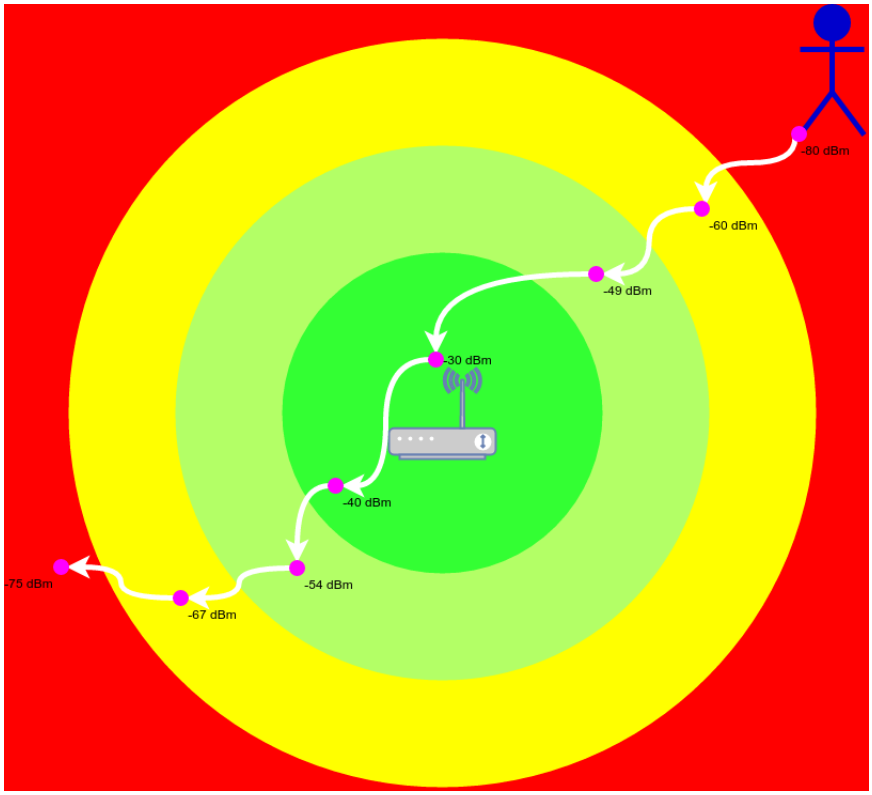


Figure 5-13: Real path

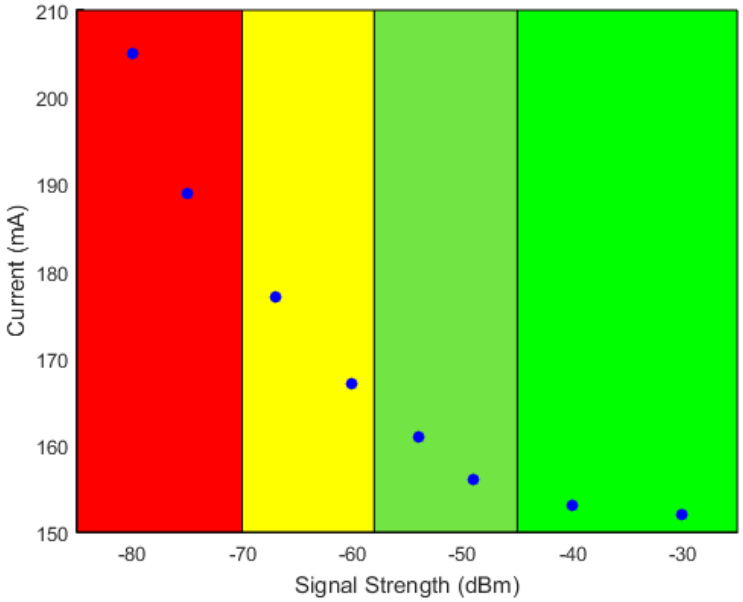


Figure 5-14: Current absorbed for each point.

The Figure 5-13 displays the path done to retrieve the data in the device, showing how the user moves from north east to south west, staying 2 times for each area, highlighted by concentric circles of different colour. We can associate, to the 4 levels of power absorption, equal number of signal levels to emphasise the level of absorption of the signal in the various areas. In Figure 5-14 there is the plot which shows the current absorbed for each point; in this way we can use this values to perform the static analysis as shown in the next table. Using as example a device with a battery of 3000mAh, we need to define the time needed to send the packet to define how much charge capacity is drained for each transmission. Using the plot inf Figure 4-2 and its relative function we can define the transfer rate of each packet sent, the transfer time using packet of 1024 kb and transforming later in percentage points.

Signal (dBm)	**Transfer Rate (Mbps)**	**Transfer Time (s)**	**Charge Capacity (mAh)**
-30	18,95	0,0540	0,2737
-40	18,60	0,0550	0,2807
-49	18,28	0,0560	0,2912
-54	17,52	0,0584	0,2912
-60	16,80	0,0609	0,3393
-67	15,25	0,0671	0,3960
-75	13,40	0,0763	0,4812
-80	12,40	0,0825	0,5643

Table 5.3: Charge capacity per signal strength

Env	τ'	**sign**	**batt**
Env'_1	τ'_0	$\perp$	$\perp$
Env'_2	τ'_1	$\perp$	$\perp$
Env'_3	τ'_2	$\perp$	$\perp$
Env'_6	τ'_5	$\perp$	$[100, 100]$
Env'_6	τ'_5	$[-80, -80]$	$[99.4357, 100]$
Env'_7	τ'_6	$[-80, -80]$	$[99.4357, 100]$
Env'_9	τ'_8	$[-80, -80]$	$[99.4357, 100]$
Env'_{11}	τ'_{10}	$[-80, -80]$	$[99.4357, 100]$
Env'_6	τ'_5	$[-60, -60]$	$[99.0964, 100]$
Env'_7	τ'_6	$[-60, -60]$	$[99.0964, 100]$
Env'_9	τ'_8	$[-60, -60]$	$[99.0964, 100]$
Env'_{11}	τ'_{10}	$[-60, -60]$	$[99.0964, 100]$
Env'_6	τ'_5	$[-49, -49]$	$[98.8052, 100]$
Env'_7	τ'_6	$[-49, -49]$	$[98.8052, 100]$
Env'_9	τ'_8	$[-49, -49]$	$[98.8052, 100]$
Env'_{11}	τ'_{10}	$[-49, -49]$	$[98.8052, 100]$
Env'_6	τ'_5	$[-30, -30]$	$[98.5314, 100]$
Env'_7	τ'_6	$[-30, -30]$	$[98.5314, 100]$
Env'_9	τ'_8	$[-30, -30]$	$[98.5314, 100]$
Env'_{11}	τ'_{10}	$[-30, -30]$	$[98.5314, 100]$
Env'_6	τ'_5	$[-40, -30]$	$[98.2506, 100]$
Env'_7	τ'_6	$[-40, -30]$	$[98.2506, 100]$
Env'_9	τ'_8	$[-40, -30]$	$[98.2506, 100]$
Env'_{11}	τ'_{10}	$[-40, -30]$	$[98.2506, 100]$
Env'_6	τ'_5	$[-54, -30]$	$[97.9371, 100]$
Env'_7	τ'_6	$[-54, -30]$	$[97.9371, 100]$
Env'_9	τ'_8	$[-54, -30]$	$[97.9371, 100]$
Env'_{11}	τ'_{10}	$[-54, -30]$	$[97.9371, 100]$
Env'_6	τ'_5	$[-67, -30]$	$[97.5410, 100]$
Env'_7	τ'_6	$[-67, -30]$	$[97.5410, 100]$
Env'_9	τ'_8	$[-67, -30]$	$[97.5410, 100]$
Env'_{11}	τ'_{10}	$[-67, -30]$	$[97.5410, 100]$
Env'_6	τ'_5	$[-75, -30]$	$[97.0598, 100]$
Env'_7	τ'_6	$[-75, -30]$	$[97.0598, 100]$
Env'_9	τ'_8	$[-75, -30]$	$[97.0598, 100]$
Env'_{11}	τ'_{10}	$[-75, -30]$	$[97.0598, 100]$

Table 5.4: Analysis of the simulation with real data with no Threshold

Env	τ'	**sign**	**batt**
Env'_1	τ'_0	$\perp$	$\perp$
Env'_2	τ'_1	$\perp$	$\perp$
Env'_3	τ'_2	$\perp$	$\perp$
Env'_6	τ'_5	$\perp$	$[100, 100]$
Env'_7	τ'_{13}	$[-80, -80]$	$[100, 100]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Env'_6	τ'_5	$[-60, -60]$	$[99.6607, 100]$
Env'_7	τ'_6	$[-60, -60]$	$[99.6607, 100]$
Env'_9	τ'_8	$[-60, -60]$	$[99.6607, 100]$
Env'_{11}	τ'_{10}	$[-60, -60]$	$[99.6607, 100]$
Env'_6	τ'_5	$[-49, -49]$	$[99.3695, 100]$
Env'_7	τ'_6	$[-49, -49]$	$[99.3695, 100]$
Env'_9	τ'_8	$[-49, -49]$	$[99.3695, 100]$
Env'_{11}	τ'_{10}	$[-49, -49]$	$[99.3695, 100]$
Env'_6	τ'_5	$[-30, -30]$	$[99.0957, 100]$
Env'_7	τ'_6	$[-30, -30]$	$[99.0957, 100]$
Env'_9	τ'_8	$[-30, -30]$	$[99.0957, 100]$
Env'_{11}	τ'_{10}	$[-30, -30]$	$[99.0957, 100]$
Env'_6	τ'_5	$[-40, -30]$	$[98.8149, 100]$
Env'_7	τ'_6	$[-40, -30]$	$[98.8149, 100]$
Env'_9	τ'_8	$[-40, -30]$	$[98.8149, 100]$
Env'_{11}	τ'_{10}	$[-40, -30]$	$[98.8149, 100]$
Env'_6	τ'_5	$[-54, -30]$	$[98.5014, 100]$
Env'_7	τ'_6	$[-54, -30]$	$[98.5014, 100]$
Env'_9	τ'_8	$[-54, -30]$	$[98.5014, 100]$
Env'_{11}	τ'_{10}	$[-54, -30]$	$[98.5014, 100]$
Env'_6	τ'_5	$[-67, -30]$	$[98.1053, 100]$
Env'_7	τ'_6	$[-67, -30]$	$[98.1053, 100]$
Env'_9	τ'_8	$[-67, -30]$	$[98.1053, 100]$
Env'_{11}	τ'_{10}	$[-67, -30]$	$[98.1053, 100]$
Env'_6	τ'_5	$[-75, -30]$	$[98.1053, 100]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Env'_7	τ'_{13}	$[-75, -30]$	$[98.1053, 100]$

Table 5.5: Analysis of the simulation with real data with Threshold = -75dBm

As the tables 5.4 and 5.5 show the analysis using the 2 algorithms, highlighting a final battery level of 97.0598% in case of original algorithm and 98.1053% in case of adapted algorithm; the analysis confirm the information concerning the proper operation of the algorithms, validating the previous analysis.

CHAPTER 6

CONCLUSIONS

Battery life is a basic aspect of smartphones; the increasing number of applications which work with Internet have made it more difficult for the smartphones to last longer due to the amount of energy involved to keep the connection. In this thesis we presented an approach to the analysis of energy consumption programs in a mobile environment, applied to Android OS. We have analised it statically, using Abstract Interpretation, getting data which were then compared with the real data, obtained from an app developed ad-hoc. In the first part is been developed a theoretical framework in order to define a static analysis sound with respect to the Android environment. In particular we define in every point of the execution which values energy is involved to execute a sending operation in a certain level of signal strength. We defined the concrete semantics of the system in a fixpoint form, and then we abstracted it with a computable semantics proving formally the soundness of our approach. After made a theoretical approach, we had developed a getter app which procures us the expediture of energy for a sending of data, related to the level of the WiFi Signal. The comparation of the data gives us the practice confirmation of what we had previously defined, validating our research. It should add that many improvements are possible, since the data produced from the app has a significant measurement error. With most accurate tools, we can

not only confirm the trend between signal intensity and energy consumption, but also give a much more accurate evaluation of power used in different records. Furthermore, we should repeat the analysis on other devices, even different, in order to give more solidity to the research.

Future developments

The main challenge of this thesis is to give a statical analysis for what concerns the energy saving, consequently to improve the user experience. A possible way to enhance it is to analyse those applications which use of a continuous stream of data, and that require a minimum bandwidth to be able to guarantee the minimum service status to the user. In this manner the application of the study can be used in a total way, highlighting strongly the energy saving, given the accentuated use of the energy resources of the device. We can bring such as streaming video and audio applications, such as Skype, WhatsApp and Facebook. Taking in example Skype, it needs 1.2Mbps in download and upload to guarantee an high definition one to one video calling, but it is recommended to have a 1.5Mbps to have an audio and video stream reliable [1]. It follows that video calling while moving implies an increasing battery consumption moving away from the access point, and degenerating the quality of service. Accordingly, we will come to a point where the quality of the streaming will no longer decent and simultaneously the energy consumption will be more intense. So, it can be definable a specific threshold to avoid useless energy expenditure.

[1] https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need

CHAPTER 7

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Agostino Cortesi, for having supported me throughout all my work for this thesis; his suggestions were very helpful and they allowed to me to find the right path. Thanks to Arianna, who helped me to translate and correct the thesis: she deserve my deepest thank you for the time she spent reading it. A special thanks to my mother Michela, my father Eugenio, my brother Edoardo, and my grandmother Santina, who have been supporting me for all of my life. Thanks to Barbara, for having been beside me all these years. Thanks to Gianluca, Alberto and Antonio for the beautiful days spent at the university over the past two years. Last but not least, many thanks also go to all my friends, and in particular to Federico, Massimo, Francesco, Silvia, Simone, Marica, Elisa, Marco, Davide, Anna, Enrico... all of my other friends of course.

APPENDIX A

AP BORDERLAND ZONE ANALYSIS.

A particular attention goes to the case where there is no overlapping among coverage of cells provided by two or more access points that normally guarantee a certain continuity of connection to the connected devices. Normally, when the user leaves the AP coverage area, a scanning function is called in order to search new networks to connect and this behavior can be repeated in different situations involving inevitably energy expenditure. In the WiFi roaming behavior when a device is disconnected from the AP, it immediately performs a scan in order to search for available networks. So doing, in the case in which the next AP is far from the current position, we have an high energy expenditure without possibility to connect.

APPENDIX B

DIMENSIONAL ANALYSIS

This study used multiple units of measurement and definitions in order to give practical meaning to the research; in this appendix we will give a full explanation of how it is used. We analyze first of all the WiFi system, which is a technology that allows user terminals to connect with each other through a local network wirelessly based on the IEEE 802.11 specifications. Being a radio protocol its electromagnetical radiation signal intensity is expressed in the form of energy average intensity of the wave, or average power per crossed area. The power is expressed in watts [W], but since the power in play is relatively low, for matters of convenience is used the decibel-milliwatts [dBm], a logarithmic units equal to $10log_{10}P+$ 30 We now define the electric current [A] such as moving amount of electric charge that passes through a given surface in a unit of time, then it's a instantaneous measurement unit. Since the electric power [W] is defined as the electric current to the electric voltage [V], being it constant in the mobile device, we will have that also the power will be defined as instantaneous. Vice versa, the device's battery has a charge capacity that is measured in milliAmpere - Hour [mAh], so in order to know how much energy [Wh] is absorbed by the battery, you have to know how much current is delivered in a certain amount of time.

APPENDIX C

STATIC ANALYSIS ON THE SIMULTATION DATA

Env	τ'	**step**	**sign**	**batt**
Env'_1	τ'_0	1	$\perp$	$\perp$
Env'_2	τ'_1	1	$\perp$	$\perp$
Env'_3	τ'_2	1	$\perp$	$\perp$
Env'_6	τ'_5	1	$\perp$	$[100, 100]$
Env'_7	τ'_{13}	1	$[-75, -75]$	$[100, 100]$
Env'_6	τ'_5	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_7	τ'_6	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_9	τ'_8	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_{11}	τ'_{10}	46	$[-74, 9, -74, 9]$	$[98, 100]$
Env'_6	τ'_5	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_7	τ'_6	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_9	τ'_8	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_{11}	τ'_{10}	47	$[-74, 7, -74, 7]$	$[96, 100]$
Env'_6	τ'_5	48	$[-74, 3, -74, 3]$	$[94, 100]$
Env'_7	τ'_6	48	$[-74, 3, -74, 3]$	$[94, 100]$
Env'_9	τ'_8	48	$[-74, 3, -74, 3]$	$[94, 100]$
Env'_{11}	τ'_{10}	48	$[-74, 3, -74, 3]$	$[94, 100]$
Env'_6	τ'_5	49	$[-73, 5, -73, 5]$	$[92, 100]$
Env'_7	τ'_6	49	$[-73, 5, -73, 5]$	$[92, 100]$
Env'_9	τ'_8	49	$[-73, 5, -73, 5]$	$[92, 100]$

Env'_{11}	τ'_{10}	49	$[-73, 5, -73, 5]$	$[92, 100]$
Env'_6	τ'_5	50	$[-72, 6, -72, 6]$	$[90, 100]$
Env'_7	τ'_6	50	$[-72, 6, -72, 6]$	$[90, 100]$
Env'_9	τ'_8	50	$[-72, 6, -72, 6]$	$[90, 100]$
Env'_{11}	τ'_{10}	50	$[-72, 6, -72, 6]$	$[90, 100]$
Env'_6	τ'_5	51	$[-72, 2, -72, 2]$	$[88, 100]$
Env'_7	τ'_6	51	$[-72, 2, -72, 2]$	$[88, 100]$
Env'_9	τ'_8	51	$[-72, 2, -72, 2]$	$[88, 100]$
Env'_{11}	τ'_{10}	51	$[-72, 2, -72, 2]$	$[88, 100]$
Env'_6	τ'_5	52	$[-71, 6, -71, 6]$	$[86, 100]$
Env'_7	τ'_6	52	$[-71, 6, -71, 6]$	$[86, 100]$
Env'_9	τ'_8	52	$[-71, 6, -71, 6]$	$[86, 100]$
Env'_{11}	τ'_{10}	52	$[-71, 6, -71, 6]$	$[86, 100]$
Env'_6	τ'_5	53	$[-70, 5, -70, 5]$	$[84, 100]$
Env'_7	τ'_6	53	$[-70, 5, -70, 5]$	$[84, 100]$
Env'_9	τ'_8	53	$[-70, 5, -70, 5]$	$[84, 100]$
Env'_{11}	τ'_{10}	53	$[-70, 5, -70, 5]$	$[84, 100]$
Env'_6	τ'_5	54	$[-69, 3, -69, 3]$	$[82, 100]$
Env'_7	τ'_6	54	$[-69, 3, -69, 3]$	$[82, 100]$
Env'_9	τ'_8	54	$[-69, 3, -69, 3]$	$[82, 100]$
Env'_{11}	τ'_{10}	54	$[-69, 3, -69, 3]$	$[82, 100]$
Env'_6	τ'_5	55	$[-68, 9, -68, 9]$	$[80, 100]$
Env'_7	τ'_6	55	$[-68, 9, -68, 9]$	$[80, 100]$
Env'_9	τ'_8	55	$[-68, 9, -68, 9]$	$[80, 100]$
Env'_{11}	τ'_{10}	55	$[-68, 9, -68, 9]$	$[80, 100]$
Env'_6	τ'_5	56	$[-69, 8, -68, 9]$	$[78, 100]$
Env'_7	τ'_6	56	$[-69, 8, -68, 9]$	$[78, 100]$
Env'_9	τ'_8	56	$[-69, 8, -68, 9]$	$[78, 100]$
Env'_{11}	τ'_{10}	56	$[-69, 8, -68, 9]$	$[78, 100]$

Env'_6	τ'_5	57	$[-69, 6, -68, 9]$	$[76, 100]$
Env'_7	τ'_6	57	$[-69, 6, -68, 9]$	$[76, 100]$
Env'_9	τ'_8	57	$[-69, 6, -68, 9]$	$[76, 100]$
Env'_{11}	τ'_{10}	57	$[-69, 6, -68, 9]$	$[76, 100]$
Env'_6	τ'_5	58	$[-68, 4, -68, 4]$	$[74, 100]$
Env'_7	τ'_6	58	$[-68, 4, -68, 4]$	$[74, 100]$
Env'_9	τ'_8	58	$[-68, 4, -68, 4]$	$[74, 100]$
Env'_{11}	τ'_{10}	58	$[-68, 4, -68, 4]$	$[74, 100]$
Env'_6	τ'_5	59	$[-67, 2, -67, 2]$	$[72, 100]$
Env'_7	τ'_6	59	$[-67, 2, -67, 2]$	$[72, 100]$
Env'_9	τ'_8	59	$[-67, 2, -67, 2]$	$[72, 100]$
Env'_{11}	τ'_{10}	59	$[-67, 2, -67, 2]$	$[72, 100]$
Env'_6	τ'_5	60	$[-66, -66]$	$[70, 100]$
Env'_7	τ'_6	60	$[-66, -66]$	$[70, 100]$
Env'_9	τ'_8	60	$[-66, -66]$	$[70, 100]$
Env'_{11}	τ'_{10}	60	$[-66, -66]$	$[70, 100]$
Env'_6	τ'_5	61	$[-64, 9, -64, 9]$	$[68, 100]$
Env'_7	τ'_6	61	$[-64, 9, -64, 9]$	$[68, 100]$
Env'_9	τ'_8	61	$[-64, 9, -64, 9]$	$[68, 100]$
Env'_{11}	τ'_{10}	61	$[-64, 9, -64, 9]$	$[68, 100]$
Env'_6	τ'_5	62	$[-65, 6, -64, 9]$	$[66, 100]$
Env'_7	τ'_6	62	$[-65, 6, -64, 9]$	$[66, 100]$
Env'_9	τ'_8	62	$[-65, 6, -64, 9]$	$[66, 100]$
Env'_{11}	τ'_{10}	62	$[-65, 6, -64, 9]$	$[66, 100]$
Env'_6	τ'_5	63	$[-65, 3, -64, 9]$	$[64, 100]$
Env'_7	τ'_6	63	$[-65, 3, -64, 9]$	$[64, 100]$
Env'_9	τ'_8	63	$[-65, 3, -64, 9]$	$[64, 100]$
Env'_{11}	τ'_{10}	63	$[-65, 3, -64, 9]$	$[64, 100]$
Env'_6	τ'_5	64	$[-66, 4, -64, 9]$	$[62, 100]$

Env'_7	τ'_6	64	$[-66, 4, -64, 9]$	$[62, 100]$
Env'_9	τ'_8	64	$[-66, 4, -64, 9]$	$[62, 100]$
Env'_{11}	τ'_{10}	64	$[-66, 4, -64, 9]$	$[62, 100]$
Env'_6	τ'_5	65	$[-67, 4, -64, 9]$	$[60, 100]$
Env'_7	τ'_6	65	$[-67, 4, -64, 9]$	$[60, 100]$
Env'_9	τ'_8	65	$[-67, 4, -64, 9]$	$[60, 100]$
Env'_{11}	τ'_{10}	65	$[-67, 4, -64, 9]$	$[60, 100]$
Env'_6	τ'_5	66	$[-66, 8, -64, 9]$	$[58, 100]$
Env'_7	τ'_6	66	$[-66, 8, -64, 9]$	$[58, 100]$
Env'_9	τ'_8	66	$[-66, 8, -64, 9]$	$[58, 100]$
Env'_{11}	τ'_{10}	66	$[-66, 8, -64, 9]$	$[58, 100]$
Env'_6	τ'_5	67	$[-66, 3, -64, 9]$	$[56, 100]$
Env'_7	τ'_6	67	$[-66, 3, -64, 9]$	$[56, 100]$
Env'_9	τ'_8	67	$[-66, 3, -64, 9]$	$[56, 100]$
Env'_{11}	τ'_{10}	67	$[-66, 3, -64, 9]$	$[56, 100]$
Env'_6	τ'_5	68	$[-66, -64, 9]$	$[54, 100]$
Env'_7	τ'_6	68	$[-66, -64, 9]$	$[54, 100]$
Env'_9	τ'_8	68	$[-66, -64, 9]$	$[54, 100]$
Env'_{11}	τ'_{10}	68	$[-66, -64, 9]$	$[54, 100]$
Env'_6	τ'_5	69	$[-67, 2, -64, 9]$	$[52, 100]$
Env'_7	τ'_6	69	$[-67, 2, -64, 9]$	$[52, 100]$
Env'_9	τ'_8	69	$[-67, 2, -64, 9]$	$[52, 100]$
Env'_{11}	τ'_{10}	69	$[-67, 2, -64, 9]$	$[52, 100]$
Env'_6	τ'_5	70	$[-67, 3, -64, 9]$	$[50, 100]$
Env'_7	τ'_6	70	$[-67, 3, -64, 9]$	$[50, 100]$
Env'_9	τ'_8	70	$[-67, 3, -64, 9]$	$[50, 100]$
Env'_{11}	τ'_{10}	70	$[-67, 3, -64, 9]$	$[50, 100]$
Env'_6	τ'_5	71	$[-68, 6, -64, 9]$	$[48, 100]$
Env'_7	τ'_6	71	$[-68, 6, -64, 9]$	$[48, 100]$

Env'_9	τ'_8	71	$[-68, 6, -64, 9]$	$[48, 100]$
Env'_{11}	τ'_{10}	71	$[-68, 6, -64, 9]$	$[48, 100]$
Env'_6	τ'_5	72	$[-69, 8, -64, 9]$	$[46, 100]$
Env'_7	τ'_6	72	$[-69, 8, -64, 9]$	$[46, 100]$
Env'_9	τ'_8	72	$[-69, 8, -64, 9]$	$[46, 100]$
Env'_{11}	τ'_{10}	72	$[-69, 8, -64, 9]$	$[46, 100]$
Env'_6	τ'_5	73	$[-70, 7, -64, 9]$	$[44, 100]$
Env'_7	τ'_6	73	$[-70, 7, -64, 9]$	$[44, 100]$
Env'_9	τ'_8	73	$[-70, 7, -64, 9]$	$[44, 100]$
Env'_{11}	τ'_{10}	73	$[-70, 7, -64, 9]$	$[44, 100]$
Env'_6	τ'_5	74	$[-70, 9, -64, 9]$	$[42, 100]$
Env'_7	τ'_6	74	$[-70, 9, -64, 9]$	$[42, 100]$
Env'_9	τ'_8	74	$[-70, 9, -64, 9]$	$[42, 100]$
Env'_{11}	τ'_{10}	74	$[-70, 9, -64, 9]$	$[42, 100]$
Env'_6	τ'_5	75	$[-71, 6, -64, 9]$	$[40, 100]$
Env'_7	τ'_6	75	$[-71, 6, -64, 9]$	$[40, 100]$
Env'_9	τ'_8	75	$[-71, 6, -64, 9]$	$[40, 100]$
Env'_{11}	τ'_{10}	75	$[-71, 6, -64, 9]$	$[40, 100]$
Env'_6	τ'_5	76	$[-72, 3, -64, 9]$	$[38, 100]$
Env'_7	τ'_6	76	$[-72, 3, -64, 9]$	$[38, 100]$
Env'_9	τ'_8	76	$[-72, 3, -64, 9]$	$[38, 100]$
Env'_{11}	τ'_{10}	76	$[-72, 3, -64, 9]$	$[38, 100]$
Env'_6	τ'_5	77	$[-72, 9, -64, 9]$	$[36, 100]$
Env'_7	τ'_6	77	$[-72, 9, -64, 9]$	$[36, 100]$
Env'_9	τ'_8	77	$[-72, 9, -64, 9]$	$[36, 100]$
Env'_{11}	τ'_{10}	77	$[-72, 9, -64, 9]$	$[36, 100]$
Env'_6	τ'_5	78	$[-73, 2, -64, 9]$	$[34, 100]$
Env'_7	τ'_6	78	$[-73, 2, -64, 9]$	$[34, 100]$
Env'_9	τ'_8	78	$[-73, 2, -64, 9]$	$[34, 100]$

Env'_{11}	τ'_{10}	78	$[-73, 2, -64, 9]$	$[34, 100]$
Env'_6	τ'_5	79	$[-72, 8, -64, 9]$	$[32, 100]$
Env'_7	τ'_6	79	$[-72, 8, -64, 9]$	$[32, 100]$
Env'_9	τ'_8	79	$[-72, 8, -64, 9]$	$[32, 100]$
Env'_{11}	τ'_{10}	79	$[-72, 8, -64, 9]$	$[32, 100]$
Env'_6	τ'_5	80	$[-73, 1, -64, 9]$	$[30, 100]$
Env'_7	τ'_6	80	$[-73, 1, -64, 9]$	$[30, 100]$
Env'_9	τ'_8	80	$[-73, 1, -64, 9]$	$[30, 100]$
Env'_{11}	τ'_{10}	80	$[-73, 1, -64, 9]$	$[30, 100]$
Env'_6	τ'_5	81	$[-73, 7, -64, 9]$	$[28, 100]$
Env'_7	τ'_6	81	$[-73, 7, -64, 9]$	$[28, 100]$
Env'_9	τ'_8	81	$[-73, 7, -64, 9]$	$[28, 100]$
Env'_{11}	τ'_{10}	81	$[-73, 7, -64, 9]$	$[28, 100]$
Env'_6	τ'_5	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_7	τ'_6	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_9	τ'_8	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_{11}	τ'_{10}	82	$[-74, 4, -64, 9]$	$[26, 100]$
Env'_6	τ'_5	83	$[-75, -75]$	$[26, 100]$
Env'_7	τ'_{13}	83	$[-75, -75]$	$[26, 100]$

BIBLIOGRAPHY

[1] Y.C. Hu A. Pathak and M. Zhang. Bootstrapping energy debugging on smartphones: A first look at energy bugs in mobile devices. *Proc. Hotnets'11*, 2011.

[2] Alyaa Syaza Azini, Muhammad Ramlee Kamarudin, and Muzammil Jusoh. Rssi and throughput performances.

[3] G. Birkhoff. *Lattice Theory*. American Mathematical Society: Colloquium publications. American Mathematical Society, 1940.

[4] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[5] Ranveer Chandra, Ratul Mahajan, Thomas Moscibroda, Ramya Raghavendra, and Paramvir Bahl. A case for adapting channel width in wireless networks. *SIGCOMM Comput. Commun. Rev.*, 38(4):135–146, August 2008.

[6] Agostino Cortesi and Francesco Logozzo. *Abstract Interpretation-Based Verification of Non-functional Requirements*, pages 49–62. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[7] Agostino Cortesi and Matteo Zanioli. Widening and narrowing operators for abstract interpretation. *Comput. Lang. Syst. Struct.*, 37(1):24–42, April 2011.

[8] Patrick Cousot. Abstract interpretation based formal methods and future challenges. In *Informatics - 10 Years Back. 10 Years Ahead.*, pages 138–156, London, UK, UK, 2001. Springer-Verlag.

[9] Patrick Cousot and Radhia Cousot. *Comparing the Galois connection and widening/narrowing approaches to abstract interpretation*, pages 269–295. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[10] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge mathematical text books. Cambridge University Press, 2002.

[11] Ning Ding, Daniel Wagner, Xiaomeng Chen, Abhinav Pathak, Y Charlie Hu, and Andrew Rice. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 29–40. ACM, 2013.

[12] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[13] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 225–238, New York, NY, USA, 2012. ACM.

[14] K. H. Kim, A. W. Min, D. Gupta, P. Mohapatra, and J. Pal Singh. Improving energy efficiency of wi-fi sensing on smartphones. In *INFOCOM, 2011 Proceedings IEEE*, pages 2930–2938, April 2011.

[15] Justin Manweiler and Romit Roy Choudhury. Avoiding the rush hours: Wifi energy management via traffic isolation. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 253–266, New York, NY, USA, 2011. ACM.

[16] Zhonghong Ou, Shichao Dong, Jiang Dong, Jukka K. Nurminen, Antti Ylä-Jääski, and Ren Wang. Characterize energy impact of concurrent network-intensive applications on mobile platforms. In *Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '13, pages 23–28, New York, NY, USA, 2013. ACM.

[17] Radhia Cousot: Patrick Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. (USA). pages 238–252, 1977.

[18] source.android.com. Measuring power values, 2015. [Online 2015-3-15].

[19] Y. Xiao, Y. Cui, P. Savolainen, M. Siekkinen, A. Wang, L. Yang, A. Ylä-Jääski, and S. Tarkoma. Modeling energy consumption of data transmission over wi-fi. *IEEE Transactions on Mobile Computing*, 13(8):1760–1773, Aug 2014.