A framework for offloading decision making to conserve battery life on mobile devices

by

Herman Marius Barnardt

submitted in fulfillment of the requirements for the degree

Magister Scientiae

in

Information Technology

for the

Faculty of Science at the
University of Johannesburg

Supervisor: Prof. M. Coetzee

Declaration

I, Herman Marius Barnardt, hereby declare that:

- The work in this dissertation is my own work;
- All sources used and referred to have been documented and recognised;
- This document has not previously been submitted in full or partial fulfilment of the requirements for an equivalent or higher qualification at any other recognised educational institution.

Herman Marius Barnardt

Abstract

The increased use of mobile devices has led to the creation of complex mobile applications that require more resources than are readily available on mobile devices. As resources such as processing power and storage are found on the cloud, resources of mobile devices can be increased by using cloud-based mobile augmentation. However, some resources, specifically battery life, and bandwidth cannot be augmented.

To augment mobile device resources such as battery life, offloading can be used. This research discusses offloading methods and examines the approaches used in related research. It is found that most of the energy consumed when offloading is due to network communication, as opposed to computation when executing locally. When offloading to the cloud consumes less energy than local execution, the battery life of a mobile device can be conserved. Choosing between offloading and local execution is called an offloading decision. To make offloading decisions that conserve battery life, the decision-making process is explored. A challenge identified when making offloading decisions is accurately estimating the energy consumption of tasks when offloading and when executing locally. As the energy consumption profile of each device differs according to the capabilities of the device, this aspect is explored.

The research conducted in this dissertation proposes the Switch framework. The Switch framework conserves the limited battery life on mobile devices by estimating the consumption of energy of a task and choosing the least expensive option. A software-based device-specific energy consumption profile is created for this purpose. Switch is evaluated using the Switch prototype, which has been designed according to the specifications of the framework. The prototype is evaluated by comparing the estimated energy consumption against the measured energy consumption. The evaluation of the framework suggests that Switch can successfully be used to conserve battery life on mobile devices by making intelligent offloading decisions.

Table of Contents

List of Figures

List of Tables

List of Equations

UNIVERSITY
OF
JOHANNESBURG

Chapter 1: Introduction

1.1. Introduction

In recent years there has been an increase in the use of mobile devices over traditional computers. The growth in mobile computing capabilities has led to rising user expectations with regards to the functionality that should be provided by mobile applications. Unfortunately, the development of mobile applications that are powerful enough to meet the user expectations is hindered by the resource constraints of mobile devices (Fakoor et al., 2012).

Developments in recent years have yielded advances in the capabilities of mobile devices, thereby reducing some of their available resources. Computational power, memory, storage and battery life are traded to enable the mobility and flexibility of mobile devices. Limitations on mobile device resources, in turn, prevent the development of mobile applications that are able to provide the functionality expected by the users (Bahl et al., 2012). Thus, by meeting certain user expectations, other mobile device requirements pay a price.

To address mobile device resource constraints, cloud computing has provided a viable solution. Mell and Grance (2009), define cloud computing as a model that provides pervasive access to a shared pool of resources that can rapidly become available and be released with minimal management effort or service provider interaction. Some of the resources provided by the cloud include networks, servers, storage, applications, and services (Mell and Grance, 2009).

Fortunately, the resources that are lacking in mobile devices are abundantly available on the cloud. Cloud computing can ensure that the problems faced by mobile developers can be overcome by augmenting the limited resources of mobile devices with the resources from the cloud. Cloud-based mobile augmentation (CMA) enables the development of applications that both support more complex application capabilities when executed by traditional computers, as well as the mobility enabled by mobile devices (Abolfazli et al., 2014).

Using the cloud to augment mobile devices is not a new idea, as several ways exist that can incorporate the cloud with mobile devices (Christensen, 2009; Cuervo et al., 2010; Hung et al., 2012), indicating that cloud computing is a viable option to augment mobile devices resources.

The goal of this research is to create a framework that conserves battery life on mobile devices by making intelligent offloading decisions. To achieve this, mobile devices and cloud computing, specifically mobile cloud computing, are examined to provide an understanding of the limitations of mobile devices and how the cloud can be used to augment mobile devices. This examination is used to identify the requirements of the proposed framework. Thereafter, offloading is discussed to gain an understanding of and to evaluate different offloading approaches. Next, the decision-making process is reviewed to identify the factors that influence offloading decision making. The proposed framework enables developers to conserve battery life on mobile devices by making the offloading decision based on the state and hardware of the mobile device.

In section 1.2 the problem domain is defined, section 1.3 gives a brief motivation for this research. Section 1.4 states the problem to be addressed by this research. The research methodology used throughout this document is defined in section 1.5. Section 1.6 gives a brief list of important terms used throughout the research. In section 1.7 the layout of this document is discussed. Finally, the chapter is concluded.

1.2. Description of the problem area

Mobile devices have become a ubiquitous and convenient method of communication. The increased usage of mobile devices has led to an increase in the complexity of mobile applications expected by users. The development of complex mobile applications is hindered by the limited resources of mobile devices. Resource constraints can be countered by using a cloud-based mobile augmentation. Unfortunately, resource constraints of mobile devices such as battery life cannot be addressed by using the cloud (Abolfazli et al., 2014; Dinh et al., 2013). In addition, frameworks that have been defined to support software

developers to add cloud augmentation to their mobile applications are complex and require hardware components.

1.3. Motivation

Battery life is the most important resource of a mobile device, as when the battery is completely discharged it cannot be used. Research has shown that the offloading of code from a battery powered device can conserve its battery life (Chun et al., 2011; Kumar and Lu, 2010; Rudenko et al., 1998). To the knowledge of the researcher, there does not exist an offloading framework that can be integrated with a mobile app to conserve battery life, which does not require the use of an external energy consumption monitor.

1.4. Problem statement

Battery life and bandwidth are resources that cannot be augmented by cloud computing. These resource constraints prevent the development of applications that provide similar application functionality to traditional devices such as desktop and laptop computers. Therefore, the problem addressed by this dissertation is the conservation of battery life of mobile devices.

To be able to address the research problem, the primary research objective and research questions are specified next.

1.4.1. Research objective

The primary objective of this dissertation is to propose a framework that can support offloading decisions to conserve battery life. The framework is software-based and should be easily integrated with existing mobile applications. The main focus of this dissertation is to identify the factors that influence the battery life consumption of mobile devices and to accurately estimate the energy consumption cost of a task.

To achieve the research objective, a number of research questions must be answered. The next section identifies the research questions that will guide this research toward addressing the problems identified.

1.4.2. Research questions

Based on the objectives identified in the previous section research questions answered in this dissertation are discussed next.

a. ***What resources are constrained on mobile devices, and which of them can be augmented?***

In order to address this research question, an understanding of the capabilities of mobile device resources and the factors that influence them is required. The role of cloud-based augmentation needs to be investigated to address resource constraints. To address this research question in more detail, the following secondary research questions are defined:

1. Which mobile devices resources can be augmented, and which cannot?
2. What methods can be used to augment the resources on mobile devices?
3. How are mobile devices resources augmented by using the cloud?

b. ***What are the requirements of a framework that can conserve battery life on mobile devices by using offloading?***

To be able to evaluate the success of a framework that conserves battery life by leveraging the cloud, a set of requirements need to be specified.

c. ***How can an offloading decision be designed to conserve the battery life of a mobile device?***

In order to use offloading to conserve battery life on mobile devices, an offloading decision needs to be made. The factors that play a role should be identified as well as the design of the offloading decision. To address this research question in more detail the following secondary research questions are identified:

1. What is offloading and what approaches can be used to offload from mobile devices?
2. How can energy consumption be measured?
3. Which factors should be taken into account when estimating energy consumption?

d. *Does the framework proposed by this dissertation conserve battery life on mobile devices?*

The proposed framework needs to be evaluated in order to answer this question. The success of the offloading decision is based on the accuracy of the energy consumption estimates. The secondary research questions identified to address this question is:

1. What tasks and evaluation criteria can be used to determine the effectiveness of the proposed framework?
2. To what extent does the proposed framework meet the identified evaluation criteria and which deficiencies and be identified?

The primary contribution of this dissertation is a portable software framework that can be used to make offloading decisions to conserve battery life on mobile devices. The question listed above serve to narrow the scope and guide the course of this dissertation.

1.5. Research methodology

To address the objectives of this dissertation, the questions posed in the previous section needs to be further broken down over the course of this dissertation to gain an understanding of the problem domain and provide the basis for a solution. To achieve this, a scientific and well-accepted approach must be followed to ensure the validity of this research.

Research can be defined as the activity of thoroughly and analytically investigating an area, with the goal of discovering or revising facts, theories, or applications and disseminating the knowledge discovered. Research methodologies are thus the means by which a discipline acquires and constructs knowledge. Scientific research methodologies consist of successive stages with the purpose of providing answers to questions that arise from scientific theories or observations (Olivier, 2009).

Empirical research makes use of empirical evidence to gain knowledge by means of observation or experience, whether direct or indirect. By quantifying evidence, a researcher answers empirical questions that are clearly defined and answerable with the evidence that is collected. Computer science can be seen

as the study of phenomena related to computers, and thus a more empirical approach to research can be followed. Quantitative research is defined as the methodical empirical investigation of observable phenomena via statistical, mathematical or computational techniques, with the goal of to develop and employ mathematical models, theories and/or hypotheses pertaining to the phenomena (Demeyer, 2011).

Literature reviews, surveys, experiments, models, languages, case studies, prototypes, arguments and mathematical proofs are some of the approaches that can be used to perform research (Olivier, 2009). The approach used in this dissertation is dependent on the goal to be achieved where a careful analysis of the problem will assist to identify which approach to use.

1.5.1. Defining the research strategy and methodologies used

Defining a framework that can be used to conserve battery life on mobile devices by leveraging the cloud is the primary goal of this research. The research strategy chosen to achieve this goal uses the framework as a primary research method and literature review, prototype, and arguments as secondary methods. Each of these methods is described next (Olivier, 2009).

a. *Literature review*

An understanding of all the concepts that contribute to the research problem is necessary to provide a starting point for a solution. To gain the necessary understanding, the researcher must take a methodical approach when investigating the problem domain and all concepts closely related to it (Olivier, 2009).

b. *Framework*

The understanding gained during the literature review enables formulation and proposal a framework in a concise manner (Glass et al., 2004). A framework provides a simplified overview of a proposal and can include relevant elements such as processes, structures, and definitions. This high-level overview enables the understanding, evaluation, and manipulation of the proposed solution (Olivier, 2009).

c. *Prototype*

The implementation of a prototype serves as a proof of concept for the model. The prototype demonstrates that the model is viable and can be implemented. A prototype can also provide new insights into the model, which a review of the model could not provide (Olivier, 2009).

d. *Argument*

Arguments enable the logical evaluation of alternative solutions, statements, facts, and ideas. The evaluation of alternative solutions enables the logical determination, given the relevant evidence, of compatibility of the solutions with the problem domain.

1.5.2. Research methodologies applied in this research

a. *Literature review*

This literature review from chapter 2 to 6 explores mobile devices, the resource available on mobile devices, the resources available on the cloud and how the resources available on mobile devices can be augmented with the resources available on the cloud. The dissertation further explores offloading and decision making, focussing on the factors that should be taken into account when making offloading decisions. In each case, a critical evaluation of literature leads to the identification of features of the approach followed by the researcher.

b. *Framework*

This dissertation proposes the Switch framework to address the problem statement. The framework identifies the components and the interaction between the components required to successfully address the problem statement. The understanding gained in the literature review is used as a foundation for the creation of the framework.

c. *Prototype*

A prototype of the Switch framework is implemented as a proof of concept. The identified components and the interactions between them are used in the creation of the prototype.

d. *Arguments*

The Switch prototype is used to perform a critical evaluation of the Switch framework against any identified requirements.

1.6. Important terms

To avoid misunderstanding, the terminology used in throughout the dissertation is now briefly defined.

1.6.1. Battery life

Battery life is the amount of power that is available on the mobile device. It is usually presented as a percentage (Carroll and Heiser, 2010).

1.6.2. Bandwidth

Bandwidth is the throughput of the network the mobile device is currently connected to (Kumar and Lu, 2010).

1.6.3. Cloud-based mobile augmentation

Cloud-based mobile augmentation is the process of increasing, enhancing, and optimizing computing capabilities of mobile devices by leveraging the resource available on the cloud (Abolfazli et al., 2014).

1.6.4. Energy consumption profile

An energy consumption profile characterizes the energy usage of a device, during the execution of different tasks on the device (Ahmad et al., 2015).

1.7. Layout of this document

This dissertation is structured to follow the research methodology suggested by Olivier (2009). Therefore, each section builds naturally upon the preceding sections so that the end outcome and the reasoning behind it are clear. This section presents the structure of this dissertation and the flow of concepts from each chapter to the next. The four sections of this document are illustrated in figure 1.1.

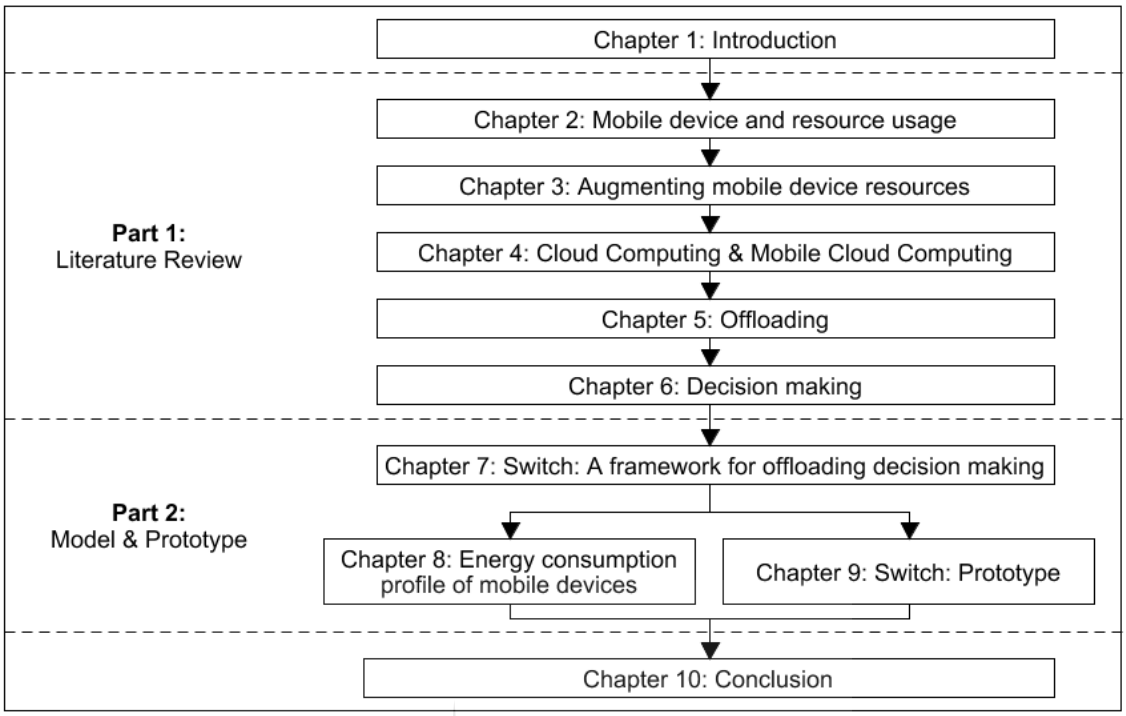

Figure 1.1 Dissertation Layout

Each of the sections is described as follows:

1.7.1. Introduction

Chapter 1 of the dissertation introduces the reader to the nature of the problems to be addressed, the objectives required to be addressed and the questions that must be satisfied to achieve the objectives. This section also serves to introduce the reader to the research methodology used and the structure of the research and the resulting document, this dissertation.

1.7.2. Part 1: Literature Review

Part 1 of this dissertation focuses on the review of relevant literature to form the basis for Part 2. This section is broken down into 5 chapters which review current, relevant literature on a topic related to the problem domain. Each chapter is briefly discussed in the sections below.

a. *Chapter 2*

Chapter 2 defines mobile devices and the resources available on mobile devices. This chapter indicates that mobile devices are being used more often

as end-users' primary device, instead of traditional computers. The increased use lead to demands of more complex applications on resource-limited mobile devices.

b. *Chapter 3*

Chapter 3 discusses the options available to increase the resources available on mobile devices and determines that the most reliable method of increasing the resources on a mobile device is augmenting the resources by offloading.

c. *Chapter 4*

Chapter 4 briefly discusses cloud computing and the potential that lies therein. The background knowledge of cloud computing is used to discuss mobile cloud computing and the different ways mobile devices can access resources on the cloud.

d. *Chapter 5*

Chapter 5 discusses offloading, focussing on offloading from mobile devices to the cloud or mobile cloud. The methods that are available to offload, the connection protocols used to connect mobile devices to the networks required for offloading, the challenges that need to be overcome when offloading and the factors that influence the offloading decision were also discussed.

e. *Chapter 6*

Chapter 6 discusses decision making in general, offloading decisions in particular and the factors that influence the offloading decision, how the factors are measured in current research and how the factors are measured in the implementations in this dissertation.

1.7.3. Part 2: Framework & Prototype

Part 2 of this research addresses the second and third actions of the research methodology suggested by Olivier (2009) by presenting the culmination of this research in the form of a framework which is evaluated by the implementation of a prototype. The sections below discuss the chapters of Part 2.

a. *Chapter 7*

Chapter 7 provides an understanding of the components of the model designed to achieve the research goal, how the components of the framework communicate and gives a broad view of the challenges of implementing a prototype based on the model.

b. *Chapter 8*

Chapter 8 discusses the experiments used to evaluate real-world energy consumption during communication and computation and the results gathered from these experiments. The data gathered from the experiments are used to inform the prototype implemented in this dissertation.

c. *Chapter 9*

Chapter 9 evaluates the framework presented in chapter 7 by documenting the prototype that is implemented in Android to make accurate offloading decisions that conserve battery life. This chapter is informed by the data gathered from the experiments done in chapter 8.

1.7.4. Conclusion

This chapter revisits the research objective and questions and evaluates the effectiveness of this research in achieving the objectives by answering the research questions. This chapter also serves to evaluate any shortcomings encountered during the course of this research.

1.8. Conclusion

This chapter introduces the problem domain of this dissertation. The research objectives are identified and the research questions are extracted from the objectives. The questions are used to limit the scope of the research and provide a means for evaluating the outcome of the research. The research methodology used in this dissertation is discussed and how the methodology shapes the structure of this research and dissertation. The next chapter is the first chapter in Part 1 of this dissertation.

Part 1:

Literature Review

Chapter 2: Mobile device and resource usage

2.1. Introduction

Today, the prolific use of mobile devices is recognised as an integral part of our lives. Statistics show that around 7.7 billion mobile devices are currently being used throughout the world. The number is predicted to rise to 12.1 billion in 2018, where 2.1 billion of these devices are smart devices (Radicati, 2014). Mobile devices are used more often than ever before, not only to make calls and send text messages but also to access the Internet, listen to music and watch videos. As these activities are all resource intensive operations it is not surprising that the biggest consumer problem with mobile devices is the lifespan of the battery (Ferreira et al., 2011; Reed, 2014).

Mobile devices generally all suffer from limited CPU, memory, storage capacity, and battery life; where battery life has been identified as the most limiting factor (Reed, 2014). Many other factors such as backlighting, wireless connections, and processor speed all have an impact on the energy consumption of the device. These limitations may cause the mobile operating system to ask the application to shut down or slow program execution. The focus of current research is to provide interventions that can optimize resource usage to ensure that an application can perform its task without any interruption (Abolfazli et al., 2012, 2014; Chun et al., 2011; Cuervo et al., 2010; Satyanarayanan et al., 2011; Wang et al., 2018).

In order to address these concerns, section 2.2 defines and discusses mobile devices in order to place them in the context of this research. As more and more complex mobile applications are executed on mobile devices, section 2.3 describes the limitations of mobile device resources that prevent the use of complex applications. The chapter also proceeds to describe possible solutions to overcome these limitations. In section 2.4, restrictions on mobile devices are re-examined to find possible methods to reduce remaining restrictions. Finally, the chapter is concluded.

2.2. Mobile devices

The rate at which mobile devices are adopted makes this technology the most popular communication medium in history (Humphreys et al., 2013). Mobile devices are an important part of modern day life as they are used by a variety of users, who make use of both simple and complex mobile applications to fulfil their personal and business needs. In order to understand the characteristics that make them resource-limited, mobile devices are discussed next. First, a definition is given, then their components and properties and finally their use are described.

2.2.1. Definition: Mobile devices

As the features of mobile devices are constantly changing, it is difficult to define the term "mobile device". For the purposes of this research, the following definition is used:

A mobile device is a handheld device hosting a mobile operating system that supports applications called apps. The device is battery powered and can access data and voice networks, via Wi-Fi (IEEE, 2016) *or cellular networks. Such devices typically have cameras, GPS* (GPS.gov, 2008) *and Bluetooth* (Bluetooth SIG, 2014) *radios, and other sensors such as accelerometers and light sensors* (Christensen, 2009; CIO Council, 2013; Souppaya and Scarfone, 2013).

Even though mobile device manufacturers are continuously creating new devices with more advanced specifications, there are underlying properties that all modern mobile devices share (Souppaya and Scarfone, 2013). A discussion of the properties of mobile devices is provided next and is divided into two sections namely hardware and software.

2.2.2. Hardware

Over the past number of years, the hardware capabilities of mobile devices have grown exponentially. In a fast competing world, mobile device manufacturers compete for the same set of consumers. Therefore, the components of mobile devices from different manufacturers may not be the

same, but they have many similar hardware components or properties (Ali et al., 2015; Souppaya and Scarfone, 2013). Next, a list of eight such properties, ranging from small form factor to storage, are described.

a. *Small form factor*

Mobile devices move around with the user of the device. Mobility is possible because devices are handheld and thus easily portable. The physical display size of the most recent devices ranges from 4 inches (Apple iPhone SE) to 7 inches (BLU Studio 7.0 II) (Apple, 2016a; Blu, 2016; Moon, 2014).

b. *Wireless connection*

Mobile devices have at least one network interface that allows access to a network for data communication. The standard connection protocols used are cellular networks (3G (ITU, 2011) or 4G (ITU, 2014)) and Wi-Fi. The specific network interface card and related connection protocol grants the mobile device access to network infrastructure, and through the infrastructure access to the Internet. The cellular network also provides voice communication, which allows the device to make and receive phone calls.

c. *Local built-in storage*

Mobile devices have built-in storage, generally ranging between 16GB and 64GB, which is used by the mobile operating system and mobile apps. Without sufficient storage, a mobile device cannot function. The storage on mobile devices is similar to the storage on traditional computers. Applications cannot be installed on the device if there is not sufficient storage.

d. *Battery*

Mobile devices are portable because they do not require a physical connection to a power source as they use batteries. The size of mobile device batteries is limited because the device needs to be small enough to be handheld. The larger the battery, the longer the mobile devices can be powered. Because of size limitations on the battery, the battery life of the device is limited (Carroll and Heiser, 2010). The technical specifications of mobile devices given by device manufacturers include how long a device can be powered by the battery under different conditions. For example:

- Stand-by time is when the device is switched on and connected to a cellular network, but not in use. Stand-by time typically lasts a couple of days.
- Talk time is when the device is continuously being used to make a phone call. Talk time is rarely longer than a day. Talk time is indicative of how long the device will stay powered when in use.

e. *Network services*

A mobile device connects to a wireless network infrastructure and the Internet using a cellular network interface. However, a mobile device typically has additional hardware components that enable communication using Bluetooth, Near Field Communication (NFC) (Minihold, 2011) and GPS. These additional network connections allow the device to connect to or create other networks such as personal area networks.

f. *Digital camera/video recording devices*

Modern mobile devices are used for more than making phone calls and sending messages. Consumers connect to the Internet to share pictures and videos on social media sites, using cameras for this purpose. The cameras on a mobile device can also be used by a barcode scanner, augmented reality and user identification apps (Dixon et al., 2013; Huang and Mow, 2013; Pan et al., 2013).

g. *Microphone*

Mobile devices can be used both as mobile phones, mobile video cameras, and mobile sound recorders. To record the sound for a video, enable the phone call or record sound, a microphone is required.

h. *Storage*

Mobile devices require built-in storage to operate. However, some devices support additional removable storage, generally ranging between 8GB and 64GB. Devices can be also be used as external storage for other devices such as traditional computers. A mobile device can be connected to a traditional computer, and files from the computer can be stored on the mobile device.

These hardware components determine the power, size, and capabilities of the mobile device. To make the device useful, the capabilities of the mobile device are further leveraged by the software on the device, discussed next.

2.2.3. Software

Manufacturers of mobile devices not only decide on the hardware components of devices but also what software is natively installed. Applications are developed by the manufacturer to support specific hardware components. For example, Samsung installs an application to count steps, and monitor the user's heart rate, using a special sensor (Samsung, 2014a). The software on the device is not limited to what the manufacturer natively provides, as the user can choose from more than two million apps to install, depending on the mobile operating system that exists (Statista, 2017). Software found on mobile devices namely the operating system and apps are discussed next, where after a comparison between mobile platforms functionality is given.

a. *Operating System*

Mobile devices are more than just phones, they are mobile computers and therefore an operating system (OS) is required to manage the applications on the device and the hardware in the device. There are a number of operating systems that have been developed for mobile devices, discussed next (Grønli et al., 2014).

i. Android

Google (Google, 2014a) released Android (Google, 2014b) in November 2007. It is the most widely used mobile operating system to date (IDC, 2014). The goal of Android is to be an open source platform for software development on mobile devices. Android is based on the Linux (Linux, 2012) kernel and uses Java (Oracle Corporation, 2014) as a programming language for the developers. Google also created Java libraries that are used in the development of applications for Android. The Android platform is more than an operating system, as it includes a development environment and a custom virtual machine. As of July 2017, there are 2 800 000 Android apps on the Google Play Store (Google, 2014c; Statista, 2017).

ii. iOS

iOS or iPhone OS, (Apple, 2014a) is the closed source and proprietary operating system for mobile Apple devices. When the iPhone was released in 2007, it revolutionized the mobile device market. iOS supports Objective-C (Apple, 2014b), an extension of the C language, Swift (Apple, 2016b), and mobile libraries to enable the development of mobile applications. Over the years, there have been many improvements in the language, mobile libraries, and the platform. As the first mobile operating system for smart devices, iOS initially had the most apps and the largest market share but has since been overtaken by Android. As of July 2017, there are 2 200 000 apps on the Apple App Store (IDC, 2014; Statista, 2017).

iii. Windows Phone

Windows Phone (Microsoft, 2014a) is the successor to Windows Mobile from Microsoft. The platform is closed source and proprietary and has the third largest installed base on smartphones following Android and iOS. Applications that run on Windows Phone are written in .NET managed code, such as C# (Microsoft, 2014b). Consequently, many developers familiar with the Microsoft's desktop development suite can easily move to Windows Phone development (Wilcox and Voskoglou, 2014).

b. *Mobile applications*

A mobile app executes above the operating system of the mobile device and is developed using specific tools defined for that platform. Each of the three mobile platforms has its own store for the applications. iOS has the Apple Store (Apple, 2014c), Android has the Play Store (Google, 2014c) and Windows Phone has the Marketplace (Microsoft, 2014c). There are countless applications for mobile devices, and there are more and more being developed (Grønli et al., 2014).

Some mobile apps are free to download and install, while others must be purchased. Mobile apps were originally created to support simple functions such as email, calendar, contacts, the stock market and weather information. Currently, public demand and the availability of developer tools has driven the

creation of more complex mobile apps such as mobile games, banking and finance, medical monitoring, GPS and location-based services, order-tracking and ticket purchases (Abolfazli et al., 2014; Dinh et al., 2013).

The number of mobile apps being developed for a platform is directly related to the number of users because apps are developed to target the largest consumer groups. A survey has shown that 71% of mobile app developers use Android, 57% use iOS and 21% use Windows Phone. Because Windows Phone uses the same development suite as Windows desktop development, there is a larger number of developers than the market share warrants (Wilcox and Voskoglou, 2014).

c. *Comparison of mobile operating systems*

A comparison between the mobile operating systems and the associated platforms has been done by Grønli et al (2014) and is summarised in Table 2.1. The table gives an overview of the software architecture, the application development and the developer support for the top three mobile operating systems.

Table 2.1 Comparison of mobile platforms

	Android	**iOS**	**Windows Phone**
Software architecture			
Development language	Java	Objective-C	.NET C#
Packaging	Android package file (APK)	Apple application package (IPA)	Windows Phone package (XAP)
Persistent storage and database support	Local SQL database support and local file access	Local SQL database support and local file access	Local SQL database support and local file access
Application development			
Debugger availability	Excellent	Very good	Excellent
Deployment speed	Relatively fast	Fast	Relatively fast
Default deployment application size	Large	Medium	Large
Developer support			
Developer community and support	Very large	Very large	Average
Market share %	84.7%	11.7%	2.5%

Integrated development environment availability	Excellent – supported by major IDE's	Very good support through Apple Xcode	Very good, but limited to Microsoft Visual Studio
Development tools cost	Free Small fee for Play Store	Free for emulator Small fee for device and App Store	Free for emulator Small fee for Marketplace

The table shows that Android and iOS compare relatively well with each other in most aspects, with Windows Phone in a distant third place. The only aspect in which there is a clear leader is market share percentage, where Android is a clear winner.

A study done by the International Data Corporation (IDC) shows that Android's market share is growing and other mobile platforms' market shares are shrinking (IDC, 2014). Android's large market share is due to the lower price of mobile devices hosting Android as an operating system, and the large variety of vendors that produce mobile devices for Android. In addition, more developers are attracted to Android because the platform is open source and the developers can gain access to system components (Page, 2014).

Due to the pervasiveness of Android, the experimentation performed in this dissertation aims to use Android as a platform, because it has the largest market share of all mobile platforms (IDC, 2014). The openness of Android is also well-suited to the collection of data relevant to the research. Given the capabilities of mobile devices, it is not surprising that the number of mobile device users is increasing, as discussed next.

2.2.4. Increased use of mobile devices

In recent years there has been an increase in the use of mobile devices over traditional computers (Fernando et al., 2013). Advances in mobile computing have further led to an increase in the complexity of mobile applications expected by users, requiring more powerful mobile devices. Unfortunately, the development of mobile applications to meet the requirements of users is hindered by the resource constraints inherent to mobile devices (Fakoor et al., 2012).

Mobile devices are becoming more ubiquitous in modern day society. These devices are always connected to the Internet and for many users, are the primary access method to the Internet. The limited resources of mobile devices prevent the creation of applications that have the functionality required by users. Typical resources and their limitations are discussed next.

2.3. Mobile device resources

The hardware components of a mobile device determine the resources of the device, and thus which resources the applications on the device can use. Hardware components such as the processor differ from device to device, and from manufacturer to manufacturer. There are a number of resources to consider, which are discussed next.

2.3.1. Computing power

The computing power of mobile devices is determined by the CPU processor of the device. There are two types of CPU processors namely single-core and multi-core. Multi-core or dual-core processors are becoming more common in current mobile devices, providing very high speed and the doubling of processing power. A more powerful CPU enables the user to run more processor intensive tasks, however, a more powerful CPU also negatively impacts the battery consumption of the device. Consequently, the battery life of a device limits the energy available to the processor. The limited energy available to the CPU processor restricts the amount of computation that can be done (Wang et al., 2014).

2.3.2. Memory

The memory on a mobile device is similar to the memory on a traditional computer. The more memory a device has, the more applications it can simultaneously run and the faster the device can respond to user requests (Kayande and Shrawankar, 2012). The memory standards for mobile devices are set by JEDEC (JEDEC, 2014a). The current standard LPDDR3 (Low Power Double Data Rate) (JEDEC, 2012) was adopted in 2013. The newer standard, published in August 2014, LPDDR4 (JEDEC, 2014b) has been met by Samsung, resulting in performance increases and lower power consumption

rates. Samsung has proposed a variant on LPDDR4, called LPDDR4x that is identical to LPDDR4 except that it uses less power (Reza, 2015). LPDDR4 and its variants are used in the flagship products of most manufacturers and will become the standard for memory for mobile devices in 2017 (Triggs, 2015).

2.3.3. Storage

The storage on mobile devices is used to store installed applications and associated media files on devices. Storage is also used to store pictures and videos taken using the camera. The capacity of storage is limited by the size of the device. Due to this limitation, most devices can support additional storage in the form of microSD cards ranging between 8GB and 128GB (Jagtap et al., 2014; SD Assocation, 2014).

2.3.4. Display

The current technology used by the screens or displays of mobile devices is Organic Light-Emitting Diode (OLED). The size of a mobile device display is limited by its small form factor (Chen et al., 2012). The display is used extensively, as it is active whenever the device is in use. Consequently, the display consumes the most battery life of all the hardware components. The brightness of the display can either be automatically set by the device, dependent on the ambient brightness, or it can be manually set by the user. The higher the brightness, the more power the display consumes (Carroll and Heiser, 2010).

2.3.5. Bandwidth

Mobile devices can access networks through various network interfaces, using different network connection protocols. The bandwidth available to the device is dependent on the network interface and connection protocol used (Dinh et al., 2013; Fernando et al., 2013; Jagtap et al., 2014). Next, the type of network interface a mobile device can connect through is discussed.

a. *Wi-Fi*

Wi-Fi (IEEE, 2016) is one of the most commonly used connection protocols, as most mobile devices have a network interface to connect to Wi-Fi networks. Wi-Fi was proposed in 1997 and since then has gone through various iterations of

improvements. The original protocol, 802.11-1997, supported speeds up to 2 Mbps. Three standard rollups have been released. 802.11-2007 supports speeds of up to 54 Mbps, 802.11-2012 supports speeds up to 150 Mbps, and 802.11-2016 supports speeds up to 866.7 Mbps. Such a connection is generally preferred to be used for resource-intensive processing, as it consumes less battery power (Dinh et al., 2013; Fernando et al., 2013; Ma et al., 2013).

b. *Cellular networks*

Most mobile devices support 3G (ITU, 2011) and more recently 4G (ITU, 2014) networks. Mobile devices can connect to mobile networks if they are in the range of the telecommunications provider's signal. The signal that is provided determines the protocol that can be used. Each type of signal supports a different bandwidth. For example, 3G has a bandwidth of at least 200Kbps up to several megabits per second and the newer 4G has a bandwidth of up to 100Mbps. Unfortunately, cellular networks are not reliable as the connection and bandwidth depend on a fluctuating signal strength, which can consume more battery life. Another factor that should be considered when using cellular networks is the monetary cost of the data (Alomari et al., 2011; Fernando et al., 2013; Parkvall and Astely, 2009).

c. *Bluetooth*

Bluetooth is used to directly connect mobile devices to each other. The range for Bluetooth is around 10m, depending on the physical objects between the devices. Bluetooth has a bandwidth of up to 800Kbps. Bluetooth connections do not consume large amounts of energy because of the limited range and bandwidth (Cherry, 2008; Fernando et al., 2013; Want et al., 2013).

2.3.6. Battery

The battery of a mobile device is one of the most important hardware components. When the battery is completely discharged, the mobile device stops functioning. The size of the battery is directly related to the charge the battery provides and the time the device can be powered. Lithium-ion batteries are widely used by many devices, including mobile devices. The latest mobile devices use Lithium-ion batteries (Carroll and Heiser, 2010; Xiaolu et al., 2012). As products need to pass rigorous safety tests, battery technology has not

advanced much in recent years. Lithium-ion is generally low-cost, easily reproducible and fairly safe. Incidents of battery fire and explosions rarely occur, considering how many lithium-ion batteries are made and sold every year (Fowler and Mozur, 2016).

2.3.7. Comparison of modern mobile devices

Table 2.2 shows a comparison of the latest devices from different manufacturers, gathered from the manufacturer's websites (Apple, 2016c; HTC, 2016; Microsoft, 2016a; Samsung, 2016a).

To clarify some of the columns in the table:

- Stand-by time: how long the phone can be on and connected to a network without being used
- Talk time: how long the phone can sustain a phone call
- Music play time: how long the device can continuously play music

The times given are the official longest times which are provided by the manufacturer and are difficult to replicate in real-world conditions. The table shows that most modern devices have very similar hardware components. The largest differences found are with the display, battery and the different battery usage times. Comparing the battery of the Samsung Galaxy S7 Edge (Samsung, 2016a) and the Microsoft Lumia 650 (Microsoft, 2016a), it is clear that the Samsung's battery is more powerful. This is due to the fact that the Samsung device has the larger screen, requiring a more powerful battery.

Throughout this dissertation, experimentation is performed using the Samsung Galaxy S7 Edge, because it is a top-of-the-range mobile device which hosts Android, the preferred mobile operating system for all experimentation performed in this dissertation.

The resources found on mobile devices are quite extensive. However, they are limited by the size of the device and batteries. When a resource such as battery life or bandwidth is limited, mobile devices cannot execute tasks as quickly or

efficiently as traditional computers. A large body of research is currently being conducted on how such resources can be extended.

Table 2.2 Comparison of modern mobile devices

Device	Operating System	CPU	RAM	Storage	Display Size	Network	Battery	Stand-by Time	Talk Time	Music Play Time
Samsung Galaxy S7 Edge	Android 6.0 (Marshmallow)	Exynos 8890 Octa (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)	4GB	32GB 64GB + microSD	5.5”	2G, 3G, 4G(LTE) Wi-Fi Bluetooth NFC	3600 mAh	Not specified	Up to 36 hours	Up to 66 hours
HTC 10	Android 6.0.1 (Marshmallow)	Qualcomm MSM8996 Snapdragon 820 Quad-core (2x2.15 GHz Kryo & 2x1.6 GHz Kryo)	4GB	32GB 64GB + microSD	5.2”	2G, 3G, 4G(LTE) Wi-Fi Bluetooth NFC	3000 mAh	Up to 456 hours	Up to 27 hours	Not specified
Apple iPhone 7	iOS 10.0.1	Apple A10 Fusion Quad-core 2.34 GHz	2GB	32GB 128GB 256GB	4.7”	2G, 3G, 4G(LTE) Wi-Fi Bluetooth NFC	1960 mAh	Up to 240 hours	Up to 14 hours	Up to 40 hours
Microsoft Lumia 650	Microsoft Windows 10	Qualcomm Snapdragon 212 Quad-core 1.3 GHz Cortex-A7	1GB	16GB + microSD	5”	2G, 3G, 4G(LTE) Wi-Fi Bluetooth NFC	2000 mAh	Up to 624 hours	Up to 16 hours	Not specified

2.4. Conclusion

This chapter defined mobile devices as battery-powered handheld devices that can constantly be connected to voice and data networks. These devices are supported by a mobile operating system, have a variety of mobile apps available and are used more and more frequently, therefore becoming a ubiquitous part of modern day life.

The increased usage of mobile devices has created a need for applications that can do everything as efficiently and quickly as a traditional computer. However, this is impossible due to the mobility of mobile devices, as they do not have the same resources as traditional computers. Resources on mobile devices such as processing power and storage are determined by their hardware components. Due to the small form factor of mobile devices, the size and power of hardware components are limited. Because mobile devices are battery powered, the battery life is one of the most important resources of mobile devices. When a mobile device has no battery life remaining, it cannot be used.

There are several ways in which resources of mobile devices can be increased or used more effectively. For example, the hardware components of a mobile device can be upgraded to give the device access to more resources, or mobile apps can be modified to use resources more effectively. Chapter 3 defines mobile resource augmentation in more detail.

Chapter 3: Augmenting mobile device resources

3.1. Introduction

The previous chapter identified that mobile devices are very sophisticated and can be used for real-world complex solutions such as government, corporate, healthcare, education, and engineering applications. Even though there is a significant improvement in mobile device capabilities, the computing requirements of e.g. corporate users are not achieved.

So far it has been pointed out that energy or battery life is the one resource that limits all the other resources of a mobile device. Currently, the energy requirements of a mobile device are supplied by lithium-ion batteries that can last only a few hours if the mobile device is used computationally. Research shows that battery capacity is only increasing at a rate of 5 to 10% a year as battery cells are excessively dense. Furthermore, the fact that mobile devices need to be lightweight and compact prevent the use of heavy long-lasting batteries. The possible loss of human life if high capacity batteries should explode further confines battery manufacturers to low capacity batteries (Ben et al., 2009).

This chapter continues the literature review by identifying that the limitations posed by the resources of mobile devices require new approaches such as augmentation (Abolfazli et al., 2014). By leveraging augmentation, more complex and intense mobile operations can become a reality. The enhancement of the computation capabilities of mobile devices is not new as approaches such as load sharing, remote execution, cyber foraging, and computation offloading have been the focus of recent research (Abolfazli et al., 2014; Bahl et al., 2012; Kumar et al., 2013).

In section 3.2, mobile resource augmentation is discussed. Resources that cannot be augmented on mobile devices are discussed in section 3.3. In section 3.4, battery life, a resource that cannot be augmented is discussed by focussing

on elements that can increase battery life usage. This discussion aims to identify requirements for conserving battery life when offloading. Section 3.5 proposes such a set of requirements. Finally, the chapter is concluded.

3.2. Mobile resource augmentation

Developments in recent years have yielded advances in the capabilities of mobile devices, and have reduced some of the resource constraints of mobile devices (Bahl et al., 2012). However, by meeting one requirement, there is always another that pays the price. Computational power, memory, storage and battery life have been traded to enable the mobility and flexibility of mobile devices, leading to limited resources on mobile devices (Bahl et al., 2012).

There are three approaches that can be used to reduce the use of resources on mobile devices or to increase the resources available on the device (Kumar et al., 2013; Kumar and Lu, 2010; Smailagic and Ettus, 2002). They are classified according to hardware, software and offloading, discussed next.

3.2.1. Hardware

The hardware of a mobile device determines the resources that are available on the device. The companies that design the devices develop hardware components that either increase the resources available on the device or enhance how the resources are used (Smailagic and Ettus, 2002).

Advances in the augmentation of resources provided by hardware can best be illustrated by comparing the Samsung Galaxy S7 Edge (Samsung, 2016a) and its predecessor the Samsung Galaxy S6 Edge (Samsung, 2016b). The comparison between the devices released a year apart, is shown in Table 3.1.

Table 3.1 Comparison of Samsung Galaxy S7 Edge & Samsung Galaxy S6 Edge

Device	Operating System	CPU	RAM	Storage	Display Size	Network	Battery
Samsung Galaxy S7 Edge	Android 6.0 (Marshmallow)	Exynos 8890 Octa (4x2.3 GHz Mongoose & 4x1.6 GHz Cortex-A53)	4GB	32GB 64GB + microSD	5.5”	2G, 3G, 4G(LTE) Wi-Fi Bluetooth NFC	3600 mAh
Samsung Galaxy S6 Edge	Android 5.0.2 (Lollipop)	Exynos 7420 Octa (4x2.1 GHz Cortex-A57 & 4x1.5 GHz Cortex-A53)	3GB	32GB 64GB 128GB	5.1”	2G, 3G, Wi-Fi Bluetooth NFC	2600 mAh

When comparing the devices, it is clear that the Samsung Galaxy S7 is more advanced. The Samsung Galaxy S7 Edge uses a more recent version of the Android mobile operating system, has a better CPU, a larger display, and a larger battery. The microSD card slot has been added after it was removed from the S6 series. Both devices can connect to all available networks. A review of these mobile devices found that battery life is one of the Galaxy S7 Edge's strongest features as the battery of the Galaxy S6 Edge drained quickly during average to heavy use, and could not make it through a day. During a performance test, the Galaxy S6 Edge lasted an only 12 hours, but the Galaxy S7 Edge lasted almost fifteen hours with full screen playing standard definition video (GSMArena, 2016).

The hardware approach is widely used and has led to the recent increase in the capabilities of mobile devices. However, hardware resources are not under the control of the users or developers of mobile devices.

3.2.2. Software

The software deployed on the mobile device determines how hardware components and thus resources are used. Software developers can program

mobile apps to use as little resources as possible, but no resources can be added to the device without including access to a resource pool such as the cloud (Kumar and Lu, 2010).

An example of such a software approach has been implemented on the Samsung Galaxy S5 (Samsung, 2014b). The Samsung Galaxy S5 has a built-in ultra-power savings mode that reduces the battery consumption to 16.8mW. The power saving is achieved by limiting access to most applications, turning the display to black and white and turning off two of the four CPU cores (Whitwarm, 2014).

The software approach is used by both the designers of the mobile device and the developers of the applications for these mobile devices. The effectiveness of the methods that can be used to conserve the resources on mobile devices varies, but can be used to create more powerful applications.

3.2.3. Offloading

Offloading is the process of moving a computational task from a resource-poor client to a resource-rich server (Kumar et al., 2013). A computational task is thus migrated from the mobile device to another computational resource in order to perform the computation there. Results are then sent back when the computation is complete. The definition of computational tasks to offload can be done either prior to execution or dynamically during runtime.

The most important question to address is at what point computation can be offloaded. Such a decision can either be based on the statistics of the application or the surrounding environment (Imai, 2012). The offloading of a computational task from a mobile device to another computational resource results in the decrease of the use of resources on the mobile device. A study conducted in 1998 with laptop computers connected to a wireless network proved that battery life can be conserved in this manner. Rudenko and others (1998) thus illustrated that the effectiveness of offloading is not only applicable to modern mobile devices.

Although offloading is implemented in software, it is discussed separately from the current discussion on software, as offloading does not use resources on the

device, but rather resources in the cloud or on a server. The discussion of offloading is divided into two sections according to the origin of resources. Offloading computational tasks from mobile devices can be done using two types of servers namely cyber-foraged servers and cloud servers (Chun et al., 2011; Cuervo et al., 2010; Kemp et al., 2012; Liang et al., 2018; Parkkila and Porras, 2011; Satyanarayanan et al., 2011; Yousafzai et al., 2016).

a. *Cyber-foraging*

Cyber-foraging uses resources from nearby devices that are either mobile or stationary. Cyber-foraging augments the computing resources of a wireless mobile device by exploiting nearby servers. Such infrastructure may be discovered and used opportunistically at different locations in the course of a user's movements.

Nearby devices create a network to which the client device can connect to. Resources available via cyber-foraging include processing power, memory, and storage. Depending on the setup and location of all devices, some resources should not be used in certain situations. For example, storage should not be used in a public space (Parkkila and Porras, 2011).

Parkkila and Porras (2011) implemented a mobile offloading application using cyber-foraging. Their approach used the Scavenger (Kristensen, 2010) system, designed for cyber-foraging. Scavenger, developed at the Aarhus University in Denmark, consists of two software components namely a daemon running on all devices that act as servers, and a client library for creating client applications (Kristensen, 2010; Parkkila and Porras, 2011). Offloading happens when the client executes a task and a nearby server is found, regardless of whether resources are available on the client. The system searches for nearby servers, and if a server is found, the task is executed by the server and the result is sent back to the client (Parkkila and Porras, 2011).

b. *Cloud*

By offloading processes onto the cloud, mobile devices are augmented with the all resources made available by the cloud. Cloud computing is a model for enabling pervasive, convenient, on-demand access to a pool of various

resources, such as computing power, storage, and memory (Kumar and Lu, 2010; Mell and Grance, 2009). Offloading to the cloud from a mobile device is a very active field of research, with several research groups focusing on solutions that offload processes from mobile devices to the cloud. Three are discussed and compared in Table 3.2. (Chun et al., 2011; Cuervo et al., 2010; Satyanarayanan et al., 2011).

i. MAUI

MAUI (Mobile Assistance Using Infrastructure) (Cuervo et al., 2010), is an offloading approach that focuses on the optimization of energy consumption and execution times of mobile applications. The focus of MAUI is to address the limitations posed by battery life for mobile devices. MAUI enables the fine-grained energy-aware offload of either mobile processes or mobile elements without much programmer effort, as code annotations indicate which methods can be executed remotely. Thus applications need to be modified so that they can be offloaded. This approach is limited to the Microsoft .NET platform (Microsoft, 2016b) which limits the type of applications that can be offloaded. First, the application code is replicated and placed on the cloud server. Thereafter the MAUI system evaluates the code and serializes and profiles all methods to determine the offloading cost.

The offloading cost is determined before the execution of a task that can be offloaded. Part of the MAUI solver is the profiler that determines the cost and decides whether to run the method locally or remotely. The profiler used by the MAUI solver takes three factors into consideration when determining the cost of offloading, namely:

- The device's energy consumption characteristics
- The program characteristics, such as the running time and resource needs of individual methods
- The network characteristics of the wireless environment, such as the bandwidth, latency, and packet loss.

The decision made by the MAUI solver ensures that the execution takes the least amount of time and consumes the least amount of energy. The offloading occurs over Wi-Fi or mobile networks.

ii. Cloudlets

Cloudlets (Satyanarayanan et al., 2011) take a local approach to cloud offloading. Using the cloud to augment mobile device resources is not without its costs, namely bandwidth and time consumption. Cloudlets leverage the local network all mobile devices can connect to, instead of the wide area network, the Internet. Cloudlets are smaller instances of the cloud, defined by specialised hardware that is connected to the local network. Mobile devices offload to a Cloudlet, instead of the cloud, via Wi-Fi. The Cloudlet is connected to the distant cloud and can, in turn, offload the task should more resources be required.

This approach reduces bandwidth and time consumed. Less bandwidth is used due to the fact that a mobile device and Cloudlet are connected to the same local network, and less time is used as the processing of the offloaded task does not occur on a distant cloud. What is being offloaded is determined dynamically. A virtual instance of the mobile device is uploaded to the Cloudlet, and when a task is offloaded to the Cloudlet, the virtual device executes the task exactly as the mobile device would, and processing can return to the mobile device at any point. Cloudlets are only used over Wi-Fi and are focussed on reducing bandwidth and time consumption.

iii. CloneCloud

CloneCloud (Chun et al., 2011) is an offloading approach that removes the need to duplicate a mobile application's code. CloneCloud creates a virtual machine of the device and runs the virtual machine on the cloud. Whenever a process is executed, the application moves the state of the device to the virtual machine and the execution continues from where the device stopped executing. A static analyser evaluates the methods of the application offline and the appropriate methods or parts of the code are marked so that they can be offloaded.

If one of the marked methods is executed, a profiler decides whether or not the process should be offloaded. The profiler gathers the data regarding the execution of the method and determines whether or not the method should be

executed locally or be offloaded to the cloud. The profiler used in CloneCloud uses execution time and energy consumed by the mobile device to inform the decision. This approach has shown some energy conservation. The application state is offloaded over both mobile networks and Wi-Fi. The developer marks methods that can be offloaded during development, this means the code partitioning is static.

iv. Comparison of offloading solutions

The approaches to offloading from mobile devices can become very technical. Table 3.2 gives a simplified comparison of the different approaches discussed.
To clarify some of the columns used in the table:

- Communication protocol: what networks are used for communication between the client and server?
- Optimization factor: what aspects are being optimized?
- Code partitioning: when are the methods replicated to the cloud?
    - Static partitioning is done during development.
    - Dynamic partitioning occurs at runtime.
- Use of a profiler for decisions

Approach	Communication Protocol	Optimization Factor	Code Partitioning	Operating System	Use of a profiler
MAUI	Wi-Fi, 3G	Energy, Execution Time	Dynamic	Windows Phone	Yes
Cloudlets	Wi-Fi	Latency, Bandwidth	Dynamic	N/A	No
CloneCloud	Wi-Fi, 3G	Energy	Static	Android	Yes

Table 3.2 Comparison of offloading approaches

A large number of a research project that aims to optimise or reduce the amount of battery life a mobile app consumes shows the importance of the

effect of battery life on the performance of mobile devices. The solutions presented demonstrates that:

- The offloading of processes should not be limited to one network
- Cannot be used in all cases as mobile devices are not always connected to a Wi-Fi network and are not always connected to a stable mobile network.

Other factors that influence the creation of a mobile app that can leverage the cloud is:

- When the offloading decision is made
- The amount of data that is required to be sent to the cloud.

The increased use of mobile devices has forced manufacturers to extend the capabilities of these devices, but due to the limited size of mobile devices, there is only so much that can be done. Improvements in hardware capacity increases resources available on mobile devices. Even though these resources are used as effectively as possible by software developers, they still do not meet the needs of users, who use complex mobile apps. It is thus clear that the cloud can be used to increase the resources of a mobile device without adding new hardware.

Both MAUI and CloneCloud use profilers to inform offloading decisions that collect data and creates a model. The profiler is defined as an app running on a mobile device. The profiler not only allows the measurement of, for example, energy consumption, but the data collected by the profiler can also be used to estimate the energy consumption under different circumstances. In order to provide a clear understanding of the resources that can be augmented by the cloud, a discussion on mobile device resources that cannot be augmented using the cloud is given next.

3.3. Constrained resources of mobile devices

Mobile devices that make use of cloud resources are called cloud-based mobile augmented devices. By using the cloud in such a way, most of the resource constraints on mobile devices such as storage, memory and computing power

can be removed (Al-mousa and Alzoubi, 2017; Bahl et al., 2012). However, some resources on mobile devices cannot be augmented and therefore can become constrained, namely bandwidth and battery life (Kumar and Lu, 2010). The limitations on bandwidth are briefly discussed next, thereafter battery life usage is discussed in more detail.

3.3.1. Bandwidth

Bandwidth specifies the data transfer rate of a network, measured in millions of bits per second or Mbps. Bandwidth is dependent on the availability of network connections, which are provided by the cellular provider or are static, such as Wi-Fi (IEEE, 2016). The hardware components of a mobile device determine the connection protocols that can be used such as Wi-Fi, 3G, 4G, and Bluetooth. Bandwidth is thus not controlled by the user or by developers (Toma et al., 2018), therefore battery life, the last and most important constraint to be considered is discussed next.

3.4. Battery life

Battery life is one of the resources that users complain about the most (Kumar and Lu, 2010). Unfortunately, the size of a mobile device, and thus the size of its batteries are limited. The size of a battery translates directly into the amount of power it can store and the time it can power the device. Trends in battery technology show that these limitations remain a real fixture, and energy inhibits mobile device development and use (Ben et al., 2009; Carroll and Heiser, 2010; Cuervo et al., 2010). The effect of both bandwidth and mobile device usage on battery life depletion is now further described in order to determine which hardware components and factors need to be considered to reduce battery life usage.

3.4.1. Effect of bandwidth on battery life

The variety of connection protocols available to mobile devices ensures that devices are almost always connected to the Internet in some way or another. Studies have shown that most users are almost always under the cover of some network with Wi-Fi networks being the most prolific at 50% of the time (Barbera

et al., 2013). The network to which the mobile device is connected to determines the bandwidth available on the mobile device.

The available bandwidth has a large impact on the power consumption of a mobile device. For example, a study was done to measure power consumption when streaming videos under different network connections. Results showed that depending on bandwidth, energy consumption can be doubled in the presence of packet loss and increased propagation delay. Figure 3.1 shows the effect on power consumption under varying bandwidths. The mobile device is connected to Wi-Fi, but the bandwidth was lowered using third party software (Ma et al., 2013; Satyanarayanan et al., 2011).

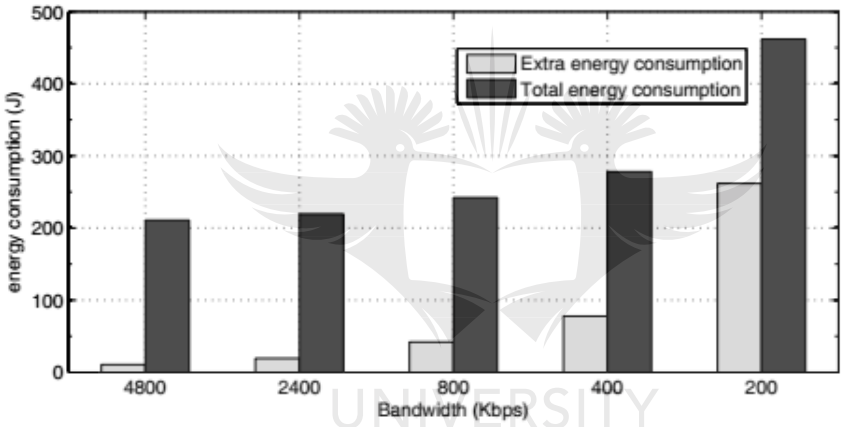

Figure 3.1 Power consumption under different available bandwidth

As shown in Figure 3.1, available bandwidths ranging between 4800Kbps and 200 Kbps dramatically impacts the power consumption of a mobile device. To the left, a bandwidth of 4800Kbps is highly effective, requiring very little additional energy. The lowest bandwidth of 200Kpbs to the right needs almost 500J to complete the task, thus using 300J unnecessarily. Therefore, the lower the bandwidth, the less reliable the network connection becomes. A poor network connection loses more packets than a good network connection, requiring communication to continue until all the packets are received. The more packets are lost, the longer the network connection is required to be maintained, and the more power is consumed. Poor network connections also take longer to send and receive packets, which further requires the network communication to be maintained (Ma et al., 2013).

Although bandwidth is one of the resources that restrains mobile application development, the availability of stable networks cannot be influenced by the developers. Developers can only control how and when the network is accessed in order to conserve the remaining battery life. The inability of developers to influence bandwidth thus excludes it from the scope of this dissertation. Next, the effect of mobile device usage on energy consumption is discussed.

3.4.2. Effect of mobile device usage on battery life

The hardware components of mobile devices use battery power to different extents. A study on the power consumption of smartphones shows that the device display consumes the most power, followed by network modems if network communication and computation is required (Carroll and Heiser, 2010).

Figures 3.2, 3.3, 3.4, and 3.5 below shows a breakdown of the power consumption by different hardware components for different tasks. Figure 3.2 shows the power consumption when making a phone call, Figure 3.3 shows the power consumption when sending an SMS, Figure 3.4 the power consumption when using an email application over both Wi-Fi and cellular networks. Lastly, Figure 3.5 shows the power consumption when using a mobile device to browse the Internet.

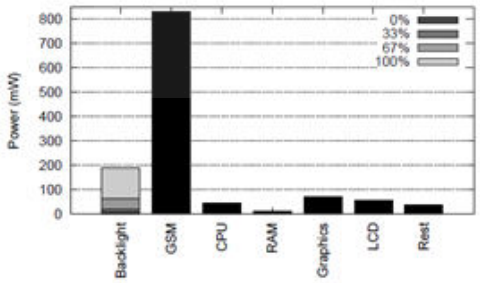

Figure 3.2 Power consumption during a the GSM phone call

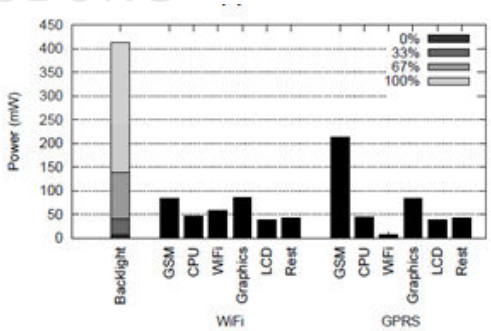

Figure 3.3 Power consumption during average use of an email application

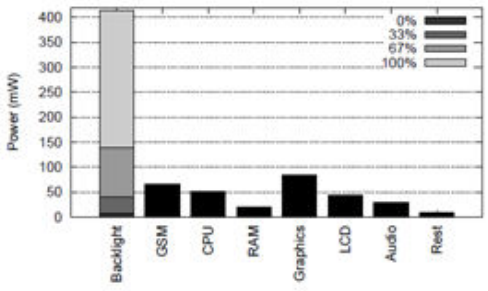

Figure 3.4 Average power consumption when when sending an SMS

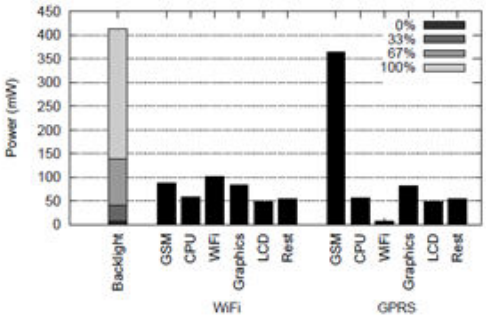

Figure 3.5 Average power consumption web browsing over Wi-Fi and GRPS

It is important to note that hardware components such as brightness (backlight), graphics and LCD are used to display images on a device's screen, each generally using a substantial amount of power. The network communication is either GSM or Wi-Fi, and computation is the usage of the device's CPU.

Figure 3.2 shows that when a phone call is made using a GSM connection, shown 2nd from the left, the most power is consumed. In this case, the power required by the display is very low, as the screen of the device is turned off when making a phone call.

Figure 3.3 shows the power consumption for sending an SMS. Here, the display consumes the most power, depending on the brightness of the screen. The GSM radio and CPU further contribute to power consumption.

Figure 3.4 shows the power consumption when using an email application over a cellular network and over Wi-Fi. Again, the display uses the most power. The communication, be it the GSM radio or the Wi-Fi radio, further contribute to power consumption, with the CPU in third place.

Figure 3.5 shows the power consumption when browsing the web. Again, the display consumes the most power, with the communication in second place and the CPU in third.

From this evaluation, it is clear that besides the display, communication and computation use the most power. When considering the effect of offloading on

battery life, mobile device usage is important to consider. Previous studies have shown that offloading can reduce the battery usage of mobile devices (Cuervo et al., 2010; Rudenko et al., 1998; Xiao et al., 2011). This is not true in all cases, as the available bandwidth and complexity of the task play a large role in the amount of energy consumed (Kumar and Lu, 2010).

By augmenting mobile device resources with cloud-based resources such as storage or CPU, mobile device resources are placed under more strain. For example, when a process is offloaded, data needs to be sent to the cloud server, requiring the use of more bandwidth and battery life. As more processes are offloaded, more bandwidth and battery life are used. If conditions are right, battery life can be conserved by offloading, depending on the bandwidth, complexity of the task and the size of the data to be transferred (Kumar and Lu, 2010). When offloading, the amount of power required for communication is determined by the size of the data to be communicated and the available bandwidth. The amount of power required to execute a task locally is determined by the complexity of the task. It is important to note that offloading can conserve battery life when the power consumption of the computation is greater than the power consumption of the communication (Barbera et al., 2013).

The central problem facing mobile devices and mobile device application development are the limited resources on mobile devices. Fortunately, the majority of resources on mobile devices such as storage or computational power can be augmented using the cloud. Even though bandwidth and total battery life cannot be influenced by the user or developer, developers and users can influence how much battery life is consumed.

Conserving battery life is the focus of this dissertation. The battery life of mobile devices should be used in such a way that the least amount of energy is consumed to complete a task. Applications that consume the least amount of battery life when executing a task have to meet some requirements. The requirements that the solution proposed in this dissertation are discussed in the next section.

3.5. Requirements for conserving battery life when offloading

Battery life is a resource that cannot be augmented by cloud computing, and cannot be influenced by users or developers. This limitation can prevent the development of mobile applications that provide similar functionality as traditional devices such as desktop and laptop computers.

This research aims to propose a software profiler to inform offloading decisions to conserve battery life in an attempt to address the limited battery life problem. The goal is achieved by developing a profiler, which estimates the amount of battery life needed when executing a task, and a decision-making component that uses the estimates the profiler provides to decide whether or not a task should be offloaded. Before a process is executed either locally or in the cloud, a comparison is done between the expected energy consumption when executing the process locally and the expected energy consumption when offloading the process. The process is executed where the least amount of battery life is consumed.

This dissertation proposes that an intelligent offloading decision-making component is needed that should meet certain requirements:

- Make intelligent offloading decisions: When offloading decisions are made, the decision-making component must consider all factors that can influence the energy consumption of the mobile device, and choose the option that consumes less energy.
- Both Wi-Fi and cellular networks need to be supported: Both Wi-Fi and cellular networks should be enabled as a Wi-Fi network may not always be available. Wi-Fi would be the preferred network to use because it is stable, has the relatively high bandwidth, and consumes less power.
- Be lightweight: The solution should consume as little energy as possible and execute as quickly as possible, as lightweight processes will not have a detrimental impact on user experience or battery life.
- Portable: The solution should be deployable on any device and integrated into any app with minimal effort.

Figure 3.6 shows the basic architecture of how the decision-making component is used. The decision-making component is installed as part of a mobile application. Whenever certain methods are called, the decision-making component gathers all relevant data from the mobile device and application and determines whether or not the method should be offloaded or not. If the method should be offloaded, the application communicates with the mobile application's cloud component, otherwise, the method is executed on the device.

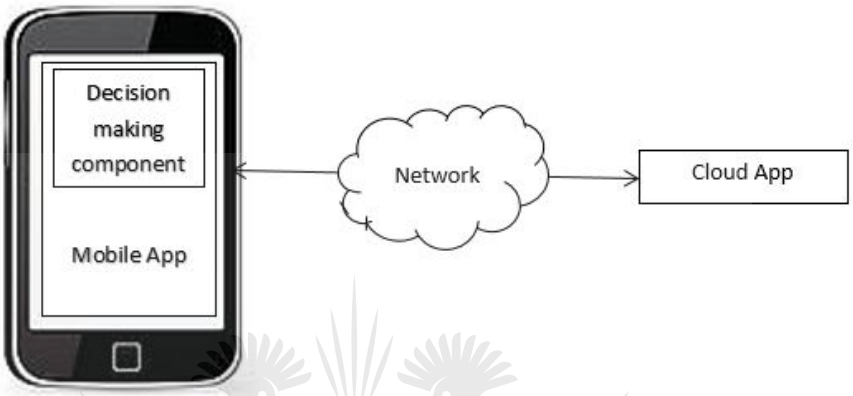

Figure 3.6 Basic architecture of the decision-making component integration with the mobile app

To define a decision-making component that meets the stated requirements, cloud computing and the use of cloud computing, different methods of offloading, decision-making methods and optimization need to be investigated.

3.6. Conclusion

Previous research shows that the offloading of processes from a mobile device to the cloud can alleviate most of the resource restrictions of mobile devices such as computing power and storage. The resources that remain constrained are bandwidth and battery life. Battery life is one of the last major bottlenecks when it comes to the development of the next generation of applications. It is thus necessary to optimize the use of the battery life to make it last as long as possible.

Unfortunately, the bandwidth available on the mobile device is determined by device connection protocols and infrastructure of networks. The process of offloading thus increases the amount of work to be done by the remaining

resources such as bandwidth and battery life, however sometimes the act of offloading can conserve the battery life of the mobile device.

This research proposes a profiler to address the problem of limited battery life on mobile devices. The profiler gathers data regarding the current state of the device and uses the state to estimate energy consumption when offloading and when executing locally, the decision-making component compares the two estimates and decides whether or not the task should be offloaded. To be able to address the requirements discussed above, cloud computing, mobile cloud computing, offloading, decision-making algorithms and optimization strategies need to be investigated.

The chapter identifies main topics to be discussed and directs the next research as follows, chapter 4 expands on cloud computing and mobile cloud computing and cloud-based mobile augmentation, chapter 5 describes offloading, and finally, chapter 6 investigates all factors that influence decision making.

Chapter 4: Cloud-based Mobile Augmentation

4.1. Introduction

The previous chapter identified that mobile device resources such as storage and processing power can be augmented by using the cloud. Due to its inherent nature, battery life is the one resource of a mobile device that cannot be augmented. In this regard, a large body of research is being conducted into techniques and frameworks to conserve battery life by offloading computation to the cloud (Chun et al., 2011; Cuervo et al., 2010; Kemp et al., 2012; Parkkila and Porras, 2011; Satyanarayanan et al., 2011; Yousafzai et al., 2016). This chapter extends the literature review further by giving a background on cloud computing and related developments to show the potential that lies therein.

The augmentation of mobile device resources is made possible by the availability of computing on demand via the cloud. Users are provided with access to computational power and storage, at a price, without the need to purchase expensive hardware components (Armbrust et al., 2010).

As offloading to the cloud is a very general concept, this chapter discusses cloud computing, mobile cloud computing (MCC) and cloud-based mobile augmentation (CMA). A review of cloud computing determines what type of cloud, private, public, community or hybrid (Mell and Grance, 2009) should be used and where the resource should be located (Abolfazli et al., 2014).

The chapter commences with a discussion on cloud computing in section 4.2. The definition, services that can be deployed in the cloud, models used to deploy the cloud and the characteristics of cloud computing are discussed. The background gained from the first section of this chapter is used as a foundation for concepts used in mobile cloud computing (MCC), discussed in section 4.3. The advantages of using cloud computing to augment mobile devices, the manner in which the augmentation can be achieved, as well as some of the challenges associated with developing mobile cloud computing applications are discussed. Finally, the chapter is concluded.

4.2. Cloud computing

During the past ten years, Internet technologies have grown at a fast rate. These new developments, the increasing costs of hardware and the need to analyse data has created a niche for cloud computing (S. Zhang et al., 2010). This section discusses the definition, architecture, services, deployments, and characteristics of cloud computing.

4.2.1. Definition: Cloud computing

Cloud computing can be used to refer to many different aspects of the cloud computing model. For the purposes of this research, the following definition is used:

Cloud computing is a model for enabling pervasive, convenient, on-demand access to a pool of various resources such as computing power, storage and memory, that can be made available quickly with minimal interaction and effort from cloud service providers and their management (Abolfazli et al., 2014; Buyya et al., 2009; Mell and Grance, 2009; Wang et al., 2010).

In general, a cloud consists of hardware and software that is exposed over the Internet, which can be accessed by consumers, sometimes at a price. Consumers using the cloud gain access to computing as a utility, similar to water and electricity. Consumers thus pay to use something that someone else provides (Buyya et al., 2009). The definition given can be expanded by including the characteristics of the cloud, discussed next.

4.2.2. Characteristics of the cloud

Cloud computing provides various services that can be made available to consumers in different ways. The cloud, irrespective of the services deployed on it or the consumers, have certain essential characteristics that are part of the definition of cloud computing (Mell and Grance, 2009). The characteristics, on-demand self-service, broad network access, rapid elasticity and measured service are discussed next.

a. *On-demand self-service*

Seen from the perspective of consumers, cloud resources such as storage and processing power are limitless. The illusion of limitless resources is achieved by the delivery of more resources than what is needed without human interaction. When a consumer requires more resources, the cloud provides those resources (Armbrust et al., 2010).

b. *Broad network access*

Cloud services are exposed using standard protocols and methods to allow consumers to access these services from any client. The client can be thin or thick, thus including the range of devices from mobile devices to standalone desktops (Mell and Grance, 2009).

c. *Resource pooling*

Limitless resource pooling is achieved by pooling resources of various physical machines and dispensing it as necessary to consumers. Physical machines can be in different locations but can be still be used by any consumers without any need for human interaction (Mell and Grance, 2009).

d. *Rapid elasticity*

Capabilities of the cloud, such as access to resources and services, can rapidly be expanded to meet the needs of consumers without human interaction. The elasticity of capabilities expands or shrinks according to the demand of consumers or their clients (Armbrust et al., 2010).

e. *Measured service*

Resource usage can be monitored, managed, and logged to provide transparency for both the provider and consumers of the utilized service. The measurement of utilized services allows the provider to be able to meet the resource demands of consumers based on the usage in the past (Mell and Grance, 2009).

The characteristics of the cloud allow for greater flexibility in the provisioning of services. As this is supported without any human interaction, cloud services can

be highly available, without the need for humans to monitor the usage of these services and permit requests for increases in resources.

Cloud services are accessed through the Internet. The communication between the consumer and the cloud provider, and the architecture of the cloud, are discussed next.

4.2.3. Cloud architecture

The cloud is supported by physical hardware and provides access to its clients via the Internet or a network (Mell and Grance, 2009). The basic model for cloud computing is shown in Figure 4.1.

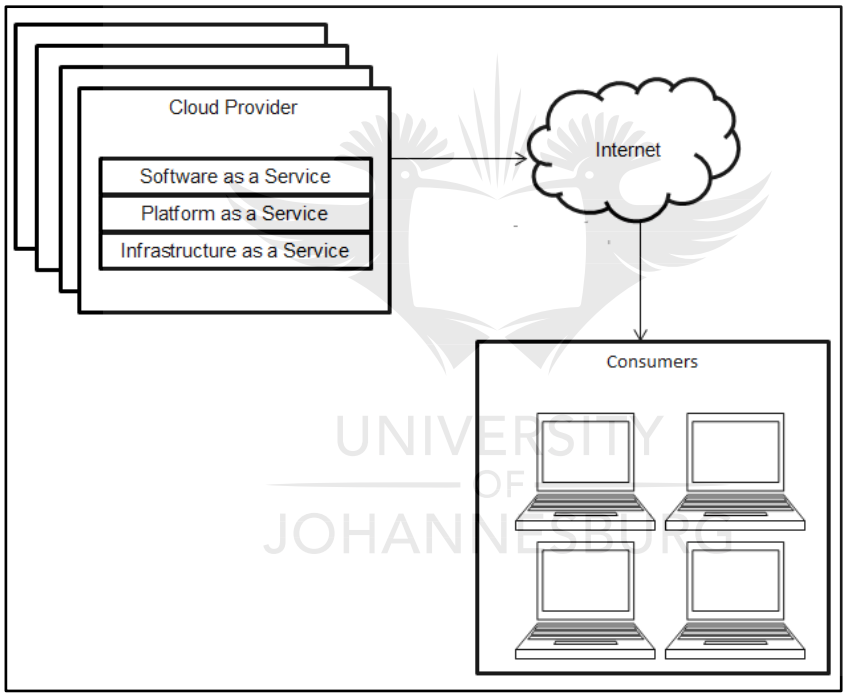

Figure 4.1 The basic layout of cloud computing

As shown in Figure 4.1, the cloud has a number of services that are provided to consumers via the Internet. The layers of services that are provided by the cloud are all stacked on top of a physical hardware layer. For most large cloud providers, the hardware layer is a data center (Dinh et al., 2013).

The given model can be expanded and changed to suit the needs of the provider or client. For example, services exposed by a cloud provider to a consumer can, in turn, be provided by another party. This means that a cloud

provider can both be a provider and consumer, leading to more layers between the consumer and the provider to create service brokers (Tsai et al., 2010). The services provided by the cloud are discussed next.

4.2.4. Service models

There are three basic service models that are associated with cloud computing namely, Infrastructure as a Service, Platform as a Service and Software as a Service (Mell and Grance, 2009). These service models are expanding with the continual growth of the cloud computing field (Zhou et al., 2010), and can be contained in the "Everything as a Service" service model (Schaffer, 2009). In this section, the service models of cloud computing are discussed. Figure 4.2, (Dinh et al., 2013), shows the layers of these models with related examples.

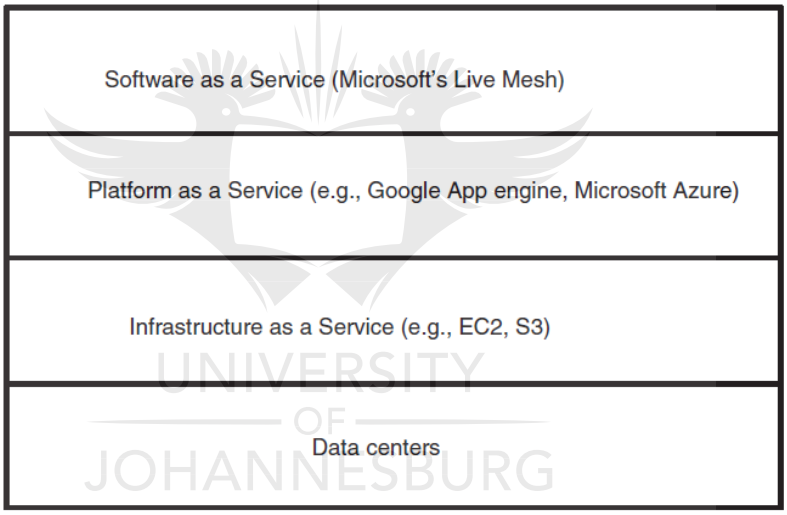

Figure 4.2 Service-oriented cloud computing architecture

Each of the service models in Figure 4.2 is discussed next.

a. *Infrastructure as a Service*

Infrastructure as a Service (IaaS) provides access to hardware via operating systems on an on-demand basis, by providing the consumer with virtual machines. The provider handles the requests from consumers if there are resources, i.e. physical machines available (Longo et al., 2011). Examples of IaaS are Amazon Elastic Cloud Computing (Amazon, 2014a) and Simple Storage Service (S3) (Amazon, 2014b).

b. *Platform as a Service*

Platform as a Service (PaaS) service model is a step above the infrastructure. The users of PaaS have direct control over applications that they deploy onto the cloud platform but have no say in the underlying infrastructure. The consumer is allowed to use the cloud provider's hardware to develop and deploy software (S. Zhang et al., 2010). Examples of PaaS are Microsoft Azure (Microsoft, 2014d) and the Google App Engine (Google, 2014d).

c. *Software as a Service*

Software as a Service (SaaS) provides users with access to software that has been deployed to the cloud. This service can be consumed by web browsers or a specific program interface. There is no limit to what software that can be exposed as over the cloud as a service (Wang et al., 2010). An example of SaaS is Microsoft's OneDrive (Microsoft, 2014e) that allows users to share files across multiple devices simultaneously.

d. *X as a Service*

Everything (X) as a Service (XaaS), as the name suggests, represents any and everything that can be represented as service. The term is clearly required when one looks at a part of the list of services that are provided by the cloud: Hardware as a Service (HaaS), Communication as a Service (CaaS), Databases as a Service (DBaaS), Security as a Service (SaaS), Identity Management as a Service (IMaaS), and Desktop as a Service (DaaS) (Schaffer, 2009).

The basis formed by the three basic service models has allowed the expansion of the services provided by the cloud to encompass more than has been mentioned. The ability of the cloud to provide anything as a service makes it an invaluable tool. Accordingly, the provision of hardware can be used to augment mobile devices that are resource constrained. The software provided as a service can be incorporated into the new software, or it can be used to provide the functionality of the software to many users. The cloud can be used to provide anything as a service to the consumers as may be required. The services on the cloud are provided to a consumer group, where the size of this group is determined by the deployment model used, discussed next.

4.2.5. Deployment models

Services that are provided by the cloud can be deployed using different models, namely, Private, Community, Public and Hybrid clouds (Mell and Grance, 2009). These models and the intended consumers are discussed in this section.

a. *Private cloud*

Private clouds are deployed for private use and are usually consumed by employees and customers of a specific organization. Private clouds can be provided and maintained by an external company or it can be done in-house.

b. *Community cloud*

Community clouds are used by a group of people or organisations that have a shared goal or concerns or are required to meet a specific set of regulations.

c. *Public cloud*

Public clouds provide services to the general public. Consumers gain access to services, be they computational power or a word processor, on a pay as you use basis or according to a contract.

d. *Hybrid cloud*

Hybrid clouds combine any of the above models. The private and public models can be combined to give employees full access to the services and allow the public to use the services that they have purchased.

The deployment model that is used limits the intended consumer pool. Service providers have access to a flexible environment to deliver their services because of the combination of deployment models that is available with the hybrid cloud deployment model.

With an understanding of cloud computing and the services provided by the cloud, one can see that cloud computing and mobile computing naturally complement each other. The combination of these two technologies is called mobile cloud computing, discussed in the following section.

4.3. Mobile Cloud Computing

Mobile Cloud Computing (MCC) was introduced not long after the introduction of cloud computing. Entrepreneurs see mobile cloud computing as a profitable alternative that can be used to provide new experiences to users through mobile applications (Dinh et al., 2013). This section discusses the definition of mobile cloud computing, and mobile cloud computing architecture.

4.3.1. Definition: Mobile cloud computing

Mobile cloud computing is more an extension of cloud computing rather than a new field. The mobile cloud is formed by using cloud computing technologies and infrastructure with mobile devices to either consume or produce services that are made available on the cloud. For example, the mobile device can be thin clients using cloud resources or mobile devices themselves can be resource providers of the cloud.

Mobile cloud computing is defined as an integration of cloud computing with mobile environments to bring new capabilities to mobile devices. Mobile cloud applications move computing power and data storage away from mobile phones into the cloud or onto a server. Mobile cloud computing refers to an infrastructure where data storage and data processing happens on a server. (Dinh et al., 2013).

Mobile cloud computing supports three types of interactions with mobile devices namely consumption, providing and offloading, which is discussed next (Alizadeh and Hassan, 2013; Fernando et al., 2013).

a. *Consumption*

Consumption is based on the server-client architecture model. The client, in this case, a mobile device, executes an application that runs on a resource-rich server. The device is seen as an extremely thin client (Fernando et al., 2013).

b. *Providing*

Providing sees other mobile devices as resource providers in the cloud to create a mobile peer-to-peer network. The consuming application uses the resources of a nearby device, either mobile or stationary (Marinelli, 2009).

c. *Offloading*

Offloading enables applications on mobile devices to execute some procedure or function on the cloud. Offloading uses resources which are abundant on the cloud while allowing the device to conserve its limited resources, and can in some cases save time, thus improving the user experience (Abolfazli et al., 2014).

The definition of mobile cloud computing indicates that the cloud can provide resources to mobile devices, especially resources that are lacking on the device. The devices can also be seen as part of the resource pool that the cloud makes available. The following section discusses the layout and architecture of mobile cloud computing to show how the mobile cloud is defined.

4.3.2. Mobile cloud computing architecture

Knowledge about mobile cloud computing does not lead to an understanding of how it works. This section aims to further expand this concept by investigating the architecture of mobile cloud computing and how mobile devices gain access to the cloud.

The greatest difference between the architecture and layout of mobile cloud computing and cloud computing is the client, and how the client connects to the Internet. Figure 4.3 by Dinh et al. (2013) shows how the mobile devices fit into the layout of mobile cloud computing.

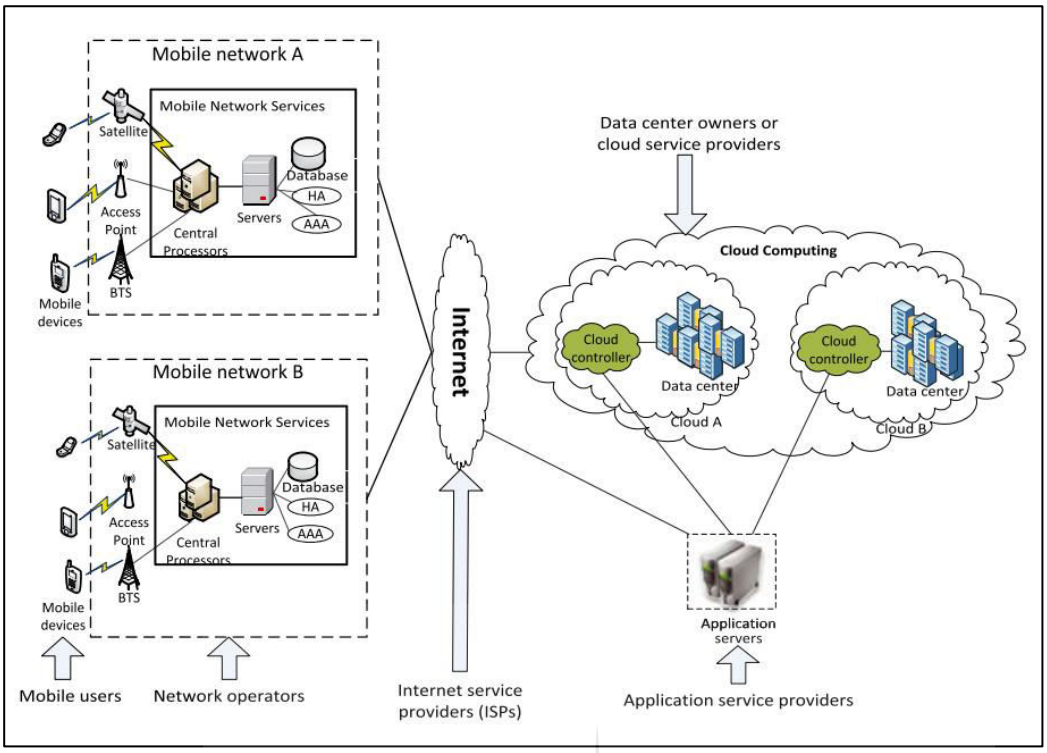

Figure 4.3 The architecture of mobile cloud computing

Mobile devices shown on the left, connect to the Internet using one of the types of connections available. Connections are facilitated by network operators via satellite, access points or cell phone towers. Network operators are connected to the Internet through their Internet service providers, shown in the middle. To the right, cloud providers are shown who host their services on the Internet. Mobile devices can access cloud services once the devices have Internet access.

When connected to the Internet, mobile devices can access the resources on the cloud in a similar manner to traditional computers. There are some challenges associated with using mobile cloud computing, discussed later in this chapter.

Having access to the cloud allows mobile devices to offload processes, or to store information on the cloud. By offloading processes or data, the device is augmented with cloud resources. This is called cloud-based mobile augmentation and is discussed in the next section.

4.3.3. Cloud-based mobile augmentation

Cloud-based Mobile Augmentation (CMA) is defined as the state-of-the-art mobile augmentation model that makes use of cloud computing technologies and principles to increase, improve, and optimize the computing capabilities of mobile devices by executing resource-intensive mobile application components in the resource-rich cloud-based resources. (Abolfazli et al., 2014). Various CMA models are discussed next.

4.3.4. Cloud-based mobile augmentation models

CMA models are determined by the manner in which the client can access resources, which devices are used to create the resource pool cloud and where these devices are (Mell and Grance, 2009). There are four CMA models namely distant fixed, proximate fixed, proximate mobile and hybrid. The models were defined according to the location of the resources that form the cloud (Abolfazli et al., 2014). For each of these models, an example is given.

a. *Distant fixed*

Distant fixed, one of the most frequently used models, uses both public and private clouds. Here, resources are provided by distant servers accessed via the Internet. By using this approach, the developer does not have to concern him-/herself with details that are already provided by the cloud provider (Huang et al., 2010). The location of the servers can be anywhere but are seen as being far away and immovable.

One of the approaches that use this method is *CloneCloud* (Chun et al., 2011). CloneCloud moves processes from the mobile device to the cloud. A virtualized clone of a device is created and run on the server. By using static and dynamic analysis, the threads of the application are distributed between the device and the cloud. The evaluation of this system has shown a dramatic decrease in execution time.

b. *Proximate fixed*

Proximate fixed uses nearby traditional computers that are inactive or are not using all of their available resources. The computers that are near the mobile

devices are configured to provide their resources or services to nearby mobile devices. This technique uses a local server instead of the distant cloud to augment the resources on the device (Abolfazli et al., 2014; Satyanarayanan et al., 2011). The resources used by proximate fixed are on local computers, thus the resources are close but immovable.

Cloudlets (Satyanarayanan et al., 2011), is one of the approaches that are based on the proximate fixed technique. The researchers that developed this approach used a nearby computer or cluster of computers as a resource pool. If the process that is offloaded to the cloudlet requires more resources than what the cloudlet can provide, the task is offloaded to the cloud. This approach was developed to counteract the delay incurred when communicating directly with the cloud from the mobile device.

c. *Proximate mobile*

Proximate fixed, also known as cyber-foraging, uses other nearby mobile devices for offloading. Here, a complex or computationally intensive task can be accomplished when the mobile device does not have access to the cloud (Marinelli, 2009). Resources can be found on nearby mobile devices, making such resources close and mobile.

Market-Oriented Mobile Cloud Computing (MOMMC) (Abolfazli et al., 2012) is an approach that uses the proximate mobile method. MOMCC is based on Service Oriented Architecture and uses services developed by programmers. The services are added to a Universal Description Discovery and Integration (UDDI) server. The UDDI provides developers with the services to create an application that can use other nearby devices. End users of applications that use the UDDI can register their devices as possible servers. The applications developed by using services on the UDDI server automatically searches for nearby devices that have been registered as possible servers. If a server is found, services are offloaded and the result is sent back. The results from this approach have not been conclusive.

d. *Hybrid*

Hybrid addresses techniques that use more than one of the above-mentioned categories to augment mobile devices. These techniques aim to reduce the disadvantages of the different approaches (Bahl et al., 2012).

MOCHA (Soyata et al., 2012) uses the hybrid approach to support a facial recognition application that uses the hardware of a mobile device. The mobile device communicates with a nearby cloudlet, and if the cloudlet requires more resources, it communicates with the distant cloud. When implemented, this approach showed significant energy consumption and time improvements, over using only either a mobile device or a standard cloud configuration. However, the development and setup of the environment are time-consuming.

The different models described here show that there are many ways to augment mobile devices with cloud computing. Alternatives to the cloud include the use of local computers as resources pools, the pooling of mobile device resources to complete a common task or a hybrid approach that uses one or more of these approaches. It can be noted that these techniques are highly dependent on the goal of the application. For instance, if an application is developed to be used in a foreign country, the proximate mobile method should be used because a mobile user would not want to spend large amounts of money on data to complete a task.

This research chooses to focus on the distant fixed model, which is not dependant on nearby devices for resources. There is no need to set up local servers and resources are not limited. The augmentation of mobile devices using the distant fixed approach supports the creation of more complex applications for mobile devices. Augmentation gives mobile devices the ability to execute tasks that require more resources than is available or it reduces the time required to execute a task. Examples of the distant fixed model are now described in more detail

4.3.5. Distant fixed cloud-based mobile augmentation

Research in distant fixed models aims to reduce the complexity and overhead of utilising the cloud. There are a number of approaches that make use of this model namely CloneCloud (Chun et al., 2011), Elastic Application (Sokol and

Hogan, 2013), VEE (Manjunatha et al., 2010), Virtualised screen (Kumar and Lu, 2010), µCloud (Chun and Maniatis, 2009), WhereStore (Kamara et al., 2010) and Wukong (Singh et al., 2012).

CloneCloud and MAIU, mentioned before, is now described in more detail to highlight potential challenges with regards to cloud augmentation and battery life. Both approaches have been found to prolong battery life the most (Abolfazli et al., 2014). Table 4.1 gives a comparison of MAUI and CloneCloud.

a. *CloneCloud*

CloneCloud achieves CMA by creating virtual instances of a mobile device on the cloud. The virtual device on the cloud allows the transfer of methods and parts of code directly to the cloud without the developer of the application to mark code as being executable on the cloud. However, for the virtual instance of the device to execute the code from the physical device, the state of the physical device has to be transferred to the cloud. The amount data that specifies the state of the device, or the application, may be large and lengthen the time the method takes to execute or how much battery life is used while executing the method (Chun et al., 2011).

b. *MAUI*

MAUI requires developers to mark methods as offloaded and to duplicate the method on the cloud. By using this approach to CMA, MAUI does not require the creation of virtual devices on the cloud and allows the use of traditional cloud infrastructure. By not using virtual devices MAUI does not need to transfer as much data to the cloud before executing. The MAUI framework can only be used once the developer has created the necessary endpoints on the cloud and has indicated that the methods can be offloaded (Cuervo et al., 2010).

Table 4.1 Comparison of offloading approaches

Approach	Comm. Protocol	Optimization Factor	Code Partitioning	Operating System	CMA approach	Initial data transfer
MAUI	Wi-Fi, 3G	Energy, Execution Time	Dynamic	Windows Phone	Distant fixed	Low

CloneCloud	Wi-Fi, 3G	Energy	Static	Android	Distant fixed	High

The distant fixed cloud-based augmentation approaches discussed above, show some of the challenges that will be faced by this research by using such an approach. The challenges identified are limiting the amount of data transferred and reducing the amount of work that has to be done by the developer to use the framework proposed in this research. The advantages of using cloud-based mobile augmentation are discussed next.

4.3.6. Advantages of cloud-based mobile augmentation

Mobile cloud computing is used to alleviate the hardware restrictions on mobile devices. It stands to reason that using mobile cloud computing provides developers with some advantages, discussed next.

a. *Extendable battery lifetime*

The battery life of a mobile device is one of its greatest limitations (Kumar and Lu, 2010). The ability to save or extend this limited resource is one of the greatest advantages of using cloud-based mobile augmentation. This conservation can be achieved by offloading tasks from the mobile device to the cloud to either reduce the time it will take to execute the task or to reduce the power needed by the device by not requiring it to execute complex tasks (Smailagic and Ettus, 2002).

b. *Resource augmentation*

The advantages of using mobile devices come at a price, which is limited processing power, storage space and memory and battery life. The cloud, on the other hand, provides most of these resources. Mobile devices can use the cloud to augment these limited resources to allow the creation of mobile applications that have access to resources equalling traditional computers (Guan et al., 2011).

c. *Improved reliability*

Mobile applications that use cloud computing access the cloud to store duplicates of user-application information in such a way that the user can

continue using the application from a different device, or in some cases platform, without requiring the transfer of data. Backups of stored user information improve the reliability of the applications (X. Zhang et al., 2010).

The access to greater resource pools and the ability to extend the battery life of mobile devices can allow the creation of more powerful applications and increase the flexibility of mobile devices. The storage of user information and data on the cloud improves the reliability of an application or device. Using cloud-based mobile augmentation does not just provide advantages, there are challenges that have to be overcome. The challenges are discussed next.

4.3.7. Challenges of mobile cloud-based augmentation

The largest difference between mobile cloud computing and cloud computing is the type of device that accesses the resources on the cloud. With cloud computing, the device is a traditional computer, and with mobile cloud computing the device is a mobile device. There are several challenges associated with using mobile devices and the cloud together, which is discussed next (Abolfazli et al., 2014).

a. *Dependency on high-performance networking infrastructure*

To access the cloud, a connection to a network is required. Mobile devices do not have access to consistently wired connections that traditional computers have. Mobile devices require a reliable, high bandwidth, high performance and robust connection to leverage the power of the cloud effectively. The availability or lack of availability is a challenge when using and developing mobile cloud computing applications.

b. *Excessive communication overhead and traffic*

The increased usage of mobile devices and mobile cloud computing has led to mobile network congestion, which is making it more expensive to communicate using mobile devices. Resource intensive actions are typically selected to be offloaded. These tasks are expensive to offload because of the amount of data that is sent to the cloud and the amount of data being received. Overcoming and managing the network communication is one of the challenges of using mobile cloud computing.

c. *Unauthorized access to offloaded data*

Once data is offloaded from a device, be it mobile or stationary, the data is no longer under the control of the owner, but rather under the control of the cloud provider. The cloud provider could gain access to the information or rearrange data in such a way that the data is exposed on a public cloud. Controlling access to data that has been offloaded is a challenge of using mobile cloud computing. The data and the access methods to the data need to be secured with the use of suitable security protocols.

d. *Application development complexity*

The development of mobile cloud computing applications requires both the development of a mobile app for the mobile device and a companion application to process resources on the cloud. A challenge is thus the dual development of these applications, each conforming to different standards, to achieve the same task.

e. *Paid infrastructures*

Cloud providers provide a utility that allows consumers to access services that they do not normally have access to, for a price. Utilities are provided to either the developer or consumer, who each have to pay the cloud provider to access the utility. The additional expense of using cloud computing is a challenge of using mobile cloud computing (Buyya et al., 2009).

The challenges outlined in this section show that the development of applications that use mobile cloud computing requires additional work. However, the advancements in the different fields, networking, security and mobile cloud computing are working on removing these challenges or at least reducing them (Alomari et al., 2011). The challenges are exactly that, challenges, they are not insurmountable.

4.4. Conclusion

This chapter set out to give a background on cloud computing, mobile cloud computing, and cloud-based mobile augmentation.

The first section discussed the definition of cloud computing, the services that can be provided by the cloud, how the consumers can get access to these services and what characteristics the cloud has. Cloud computing provides services, be it hardware or software, to consumers on an on-demand basis without excessive user interaction. Cloud computing can also be seen as a utility as it provides computational power to consumers that do not have it. The characteristics of the cloud focus on meeting consumer demands without any human interaction, to make it possible to provide services on the cloud to as many people as possible. The services provided by the cloud can be exposed due to the architecture of the cloud. The services can be combined in countless ways and can be exposed as service of their own. This can be summarized in the phrase “X as a Service” or everything can be a service. The cloud can be deployed using different models that allow different groups of consumers to access the cloud. The deployment model allows the provider to determine how large or small the consumer base will be and who can have access to it. The groups can also be combined using a hybrid deployment model.

The second section of the chapter discussed the definition of mobile cloud computing, cloud-based mobile augmentation, the techniques used to augment mobile devices and the advantages of using the cloud with mobile devices. Mobile cloud computing is the use of cloud computing technologies with mobile devices. The mobile cloud is used by mobile devices to produce or consume services or to offload tasks to the cloud. The architecture and layout of mobile cloud computing do not differ that much from the architecture and layout of traditional cloud computing. The main difference is the client and how the client communicates with the cloud servers. Cloud-based mobile augmentation or augmentation, in brief, is the process of increasing, enhancing, and optimizing computing capabilities of mobile devices by leveraging the cloud and the resources made available by the cloud. The techniques used to augment mobile devices using the cloud are categorized into four models based on the type of cloud being used and the location of the resources. These categories use the cloud, nearby computers, nearby mobile devices and a hybrid of these approaches. Each of these is viable and has different applications. This research uses the distant fixed approach of cloud-based mobile augmentation.

This chapter discussed 3 approaches that use the distant fixed approach. CloneCloud uses virtual instances of the physical mobile devices on the cloud to offload. MAUI requires the creation of more traditional cloud resources to offload functionality from the mobile device. The approaches used require either more data to be transferred to the cloud or more work from the developers to use the frameworks.

The advantages of using mobile cloud computing counteract the disadvantages of using mobile devices. The storage of users' data and information on the cloud increases the reliability of the applications. The greatest advantage of using mobile cloud computing is the ability to offload from the mobile device to the cloud to allow the conservation of one of the scarcest resources on mobile devices i.e. battery life. The advantages gained by using mobile cloud computing comes at the price of challenges that have to be faced when creating or using applications that use mobile cloud computing.

The advantages gained by using cloud-based mobile augmentation come at the price of certain challenges, this chapter discussed those challenges. By using the cloud, a reliance is created on the availability of Internet access and thus to various networks that connect the mobile device to the Internet. The networks used are unreliable and create overhead when communication occurs between the device and the cloud. To allow applications to leverage the availability of resources on the cloud requires the developers to create applications that are capable of doing so, this can lead to increased complexity. The last challenge that should be overcome when implementing a CMA framework is the monetary cost of using a cloud provider. Offloading is discussed in more detail the next chapter.

Chapter 5: Offloading

5.1. Introduction

Current research in mobile cloud computing focuses on techniques that can be used to support resource-demanding mobile applications. In chapter 3, it was identified that mobile resource augmentation can be performed using hardware, software and offloading. Increasing the resources available on mobile devices by upgrading or replacing the hardware can be expensive, and not all devices are upgradeable. Mobile augmentation achieved by offloading can be used by any mobile device that is connected to the Internet and provides access to the limitless resources available on the cloud. Offloading is a technique that is used to overcome the limitations of mobile devices in terms of computation, memory and battery life. The mobile cloud can provide resources to create more powerful applications even though the battery life of a mobile device and the bandwidth of the connection cannot be augmented.

Offloading, especially offloading from a mobile device to the cloud, is a challenging task to accomplish. Developing a framework that uses offloading to conserve battery life on mobile devices requires an understanding of the different approaches that can be used. The level at which offloading should occur, the requirements of the cloud server, the identification of offload-able components and what data is required on the cloud are some of the factors that should be considered when creating a mobile cloud computing offloading framework.

The aim of this chapter is to give an overview of offloading and how it can be used to alleviate the limited resources on mobile devices. To achieve this aim, offloading is defined in section 5.2. With offloading defined, the various methods of offloading are discussed and compared in section 5.3. Section 5.4 discusses the connection protocols that can be used to connect to the Internet, and using the connection to the Internet, the offloading of tasks to the cloud. Section 5.5 compares offloading frameworks used in mobile cloud computing. Section 5.6 covers the challenges when offloading from mobile devices to the mobile cloud. Finally, the chapter is concluded.

5.2. Definition: Offloading

Offloading is not a new idea, as offloading capabilities available to traditional computers have now become available to mobile devices due to the advancements in mobile device technologies (Dinh et al., 2013; Souppaya and Scarfone, 2013). The ability to offload processes to the cloud, and to augment the resources on mobile devices, is of key importance to the continual development of mobile devices and applications. Some resources cannot be physically transferred to the mobile device, thus the tasks that the mobile device needs to execute is moved to the cloud where the cloud resources can be used.

Offloading is defined as the process of moving a task from a resource-poor client to a resource-rich server. In this case, the server is the cloud and the client a mobile device. (Kumar et al., 2013).

Offloading differs from a traditional cloud-server-client architecture, where a thin client always needs to offload the processing to the server. Offloading also differs from grid or multi-processor computing where tasks are migrated in order to balance the load on different processors. Offloading moves processing tasks to a server that does not need to be in the vicinity of the client (Kumar et al., 2013). Thus mobile devices are not required to sacrifice mobility when using the cloud, because the cloud is remote.

The use of offloading with mobile devices has been enabled by the recent advances in mobile device technology as most modern mobile devices have near constant access to the Internet (Dinh et al., 2013). Consequently, mobile devices have access to the cloud because of the increase in the use of the cloud and the developments made in the provisioning of services by the cloud, made possible by virtualization (Kumar et al., 2013).

The methods that can be used to offload tasks from mobile devices are discussed next (Abolfazli et al., 2014).

5.3. Methods of offloading

The offloading of tasks from mobile devices has extensively been researched. Research has shown there are three main directions that these methods take namely, client-server communication, virtualization, and mobile agents (Fernando et al., 2013). The methods and a comparison between these methods are discussed next.

5.3.1. Client-server communication

Client-server communication takes place between the mobile device and the cloud provider by using protocols such as Remote Procedure Call (RPC) (Flinn et al., 2002; Marinelli, 2009), Remote Method Invocation (RMI) (Flinn et al., 2002; Marinelli, 2009) or by using sockets (Balan et al., 2003; Fernando et al., 2013).

These communication methods are well established and are considered to be stable by developers. A drawback of using these communication methods is that procedures that are remotely called, or the methods that are remotely invoked, need to be installed on the server prior to execution, which hinders the mobile and flexible nature of mobile devices. Because the application needs to communicate with a specific server that has the procedures or methods installed, network congestion can be created (Fernando et al., 2013).

Figure 5.1 (Fernando et al., 2013) illustrates the communication between a server and a client when using client-server communication to offload. The client sends the call to execute a procedure or method and if necessary the server returns the result of the execution.

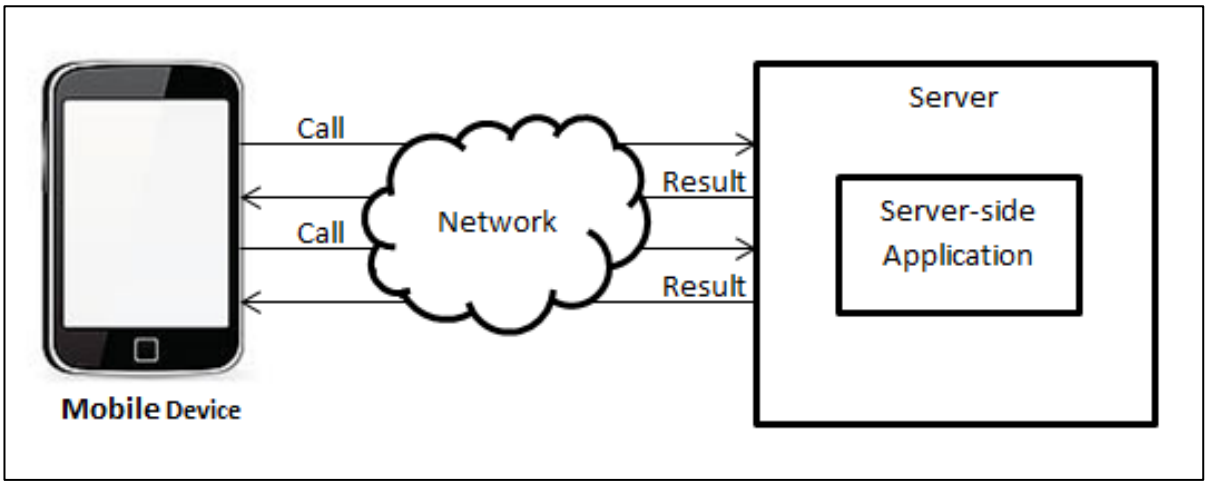

Figure 5.1 Client-server communication.

The communication between the mobile device (the client) and the server (the cloud) is straightforward. The server exposes a number of procedures that can be called remotely. The client accesses the server's procedures, provides the necessary data as parameters, and calls the procedure. The server returns a result if necessary.

The client-server method is well established and has been proved to be compatible with mobile devices. The difficulty of implementing this approach comes when deciding whether or not to offload a task as the state of the device, and the connected network's information needs to be considered when deciding whether or not the process should be offloaded.

5.3.2. Virtualization

Virtual machine migration is the process of transferring a duplicate of the memory instance of a virtual machine from the source to a destination without stopping the execution of processes running on the virtual machine (Clark et al., 2005). The memory pages of the original machine have to be transferred to the destination machine before the execution can be transferred, to provide an illusionary seamless migration (Cuervo et al., 2010; Fernando et al., 2013; Satyanarayanan et al., 2011).

Virtualization does not require the alteration or duplication of code. Because the execution is moved to a duplicate virtual machine, the physical server is insulated. However, the duplication and creation of the virtual machine are time-

consuming and there may exist compatibility issues between virtual machines. The lack of resources on mobile devices can also be a hindrance for the resource intensive task of virtual machine migration (Fernando et al., 2013).

Figure 5.2 (Fernando et al., 2013) shows the communication between the original and destination virtual machine and the layers of the virtual machine. The mobile device communicates with the virtualized hardware layer to transfer the memory, and with the operating system and application to execute the process.

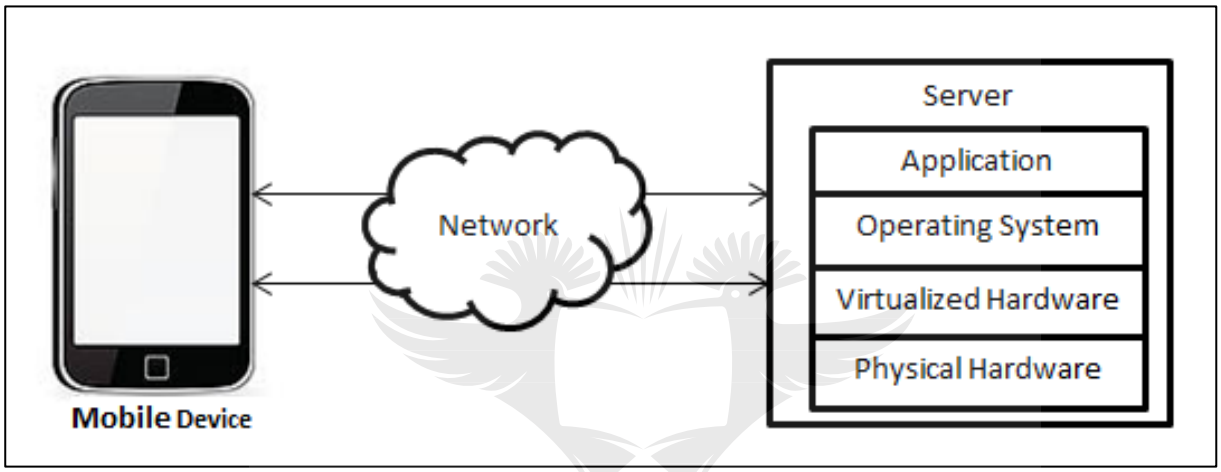

Figure 5.2 Virtual machine migration.

The communication between the mobile device and server occur via a network, which is usually the Internet for cloud-based servers. Virtual machine migration requires large amounts of data to be transferred to the cloud before processes can be offloaded. The state of the mobile device is sent to the cloud where the virtual mobile device is hosted. Once the state has been restored to the virtual device the computationally intensive task is executed, and the state after the process has been executed is sent back to the physical mobile device.

Virtualization of mobile devices and the applications that run on them allows continuous execution when offloading. Because the server is executing the process as if the mobile device was executing the process, there is no need to create code for the server and code for the client. The resource usage, however, limits the size of the applications that can be developed, because the

mobile device does not have enough resources to continue executing and to transfer the application execution.

5.3.3. Mobile agents

Unlike the previous approaches, mobile agents offload tasks not to the cloud, but to nearby devices. The client can, depending on the number of servers and resources available on them, partition tasks so that different partitions of a task is executed on different servers (Kristensen, 2010).

The use of mobile agents supports dynamic deployment because all the code that needs to be executed is already on nearby devices. The execution can occur on such devices at the cost of time and battery life. However, because the mobile agent's approach uses different devices as servers, security is questionable. Because nearby devices are being used, they need to be managed (Fernando et al., 2013).

Figure 5.3, (Fernando et al., 2013), shows the communication between the client and the server. The server contains the mobile agent that communicates with the service. The server responds to the client the result of the task or part of the task that was sent for execution.

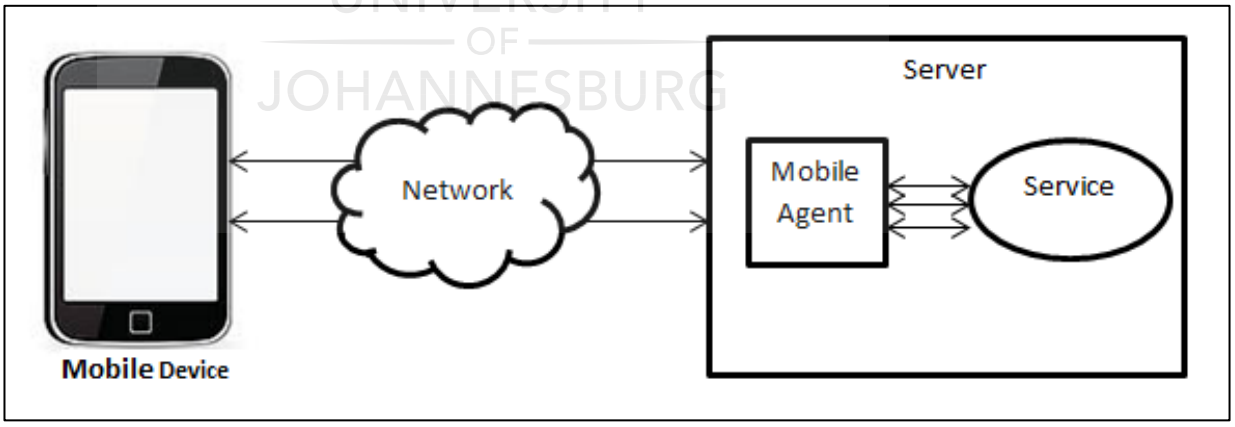

Figure 5.3 Mobile agents.

The mobile device communicates with other nearby mobile devices and moves a task to the device to be executed. The server device has a mobile agent that listens for requests from client devices. When a request is made, the server executes the required services and returns the result, if there is one, to the client. Unlike with other methods, the network that the mobile device connects

to does not have to be the Internet. The server and client devices are near each other, meaning the communication can take place via a local Wi-Fi network or the devices can create a connection using Bluetooth.

Using mobile agents for offloading is very dynamic, has offline capabilities, but the lack of security and the limited resources does not allow the creation of powerful applications. The mobile agent approach is used more to conserve the resources on the client and not used to augment the resources of the client.

5.3.4. Comparison of offloading methods

Each of the offloading methods has different advantages and disadvantages. This section provides a comparison between the methods, shown in Table 5.1. The results of the comparison provide a motivation for the choice of the method for this research (Christensen, 2009; Clark et al., 2005; Cuervo et al., 2010; Fernando et al., 2013; Satyanarayanan et al., 2011).

Table 5.1 compares not only the methods but also the server type and location for the client-server and virtualization approach. For each of the methods, the amount and location of the resources are compared, as well as the size of the communication required to use the given approach and the complexity of the code to use the approach.

Table 5.1 Comparison of offloading methods

	Resources	**Resource location**	**Communication size**	**Coding complexity**
Client-server (Using the cloud)	Nearly limitless	Distant	Small	High
Client-server (Using a local server)	Limited	Nearby	Small	High
Virtualization (Using the cloud)	Nearly limitless	Distant	Large	None
Virtualization (Using a local server)	Limited	Nearby	Large	None
Mobile Agents	Limited	Nearby	Small	Some

From an evaluation of Table 5.1, the trade-offs between the different offloading methods are clearly shown. At the expense of communication size, offloading can be achieved without the need to increase code complexity, using virtualization. The client-server method increases code complexity but does not require any unnecessary communication to the server. Mobile agents do not increase code complexity to the same extent than the client-server approach does and does not require more data to be transferred, however, the resources are not guaranteed and is not a reliable option.

The method chosen for the implementation in this dissertation is *client-server communication using the cloud*. Client-server is selected because it does not require large amounts of data to be transferred before operations can be offloaded. The server on the cloud is chosen because it provides the most resources and is constantly available. The drawback of using the client-server communication method is the coding complexity. However, because the implementation in this dissertation is done in Android, the same code can be used in a Java server-side application.

This discussion shows that processes can be offloaded from a mobile device to a server, whether the server is the cloud, a local computer or nearby mobile device. All these methods require communication with the server and communication requires connection protocols, which are discussed below.

5.4. Connection protocols

To be able to offload computation, the ability to connect to the Internet or a network is required. There are various connection protocols available on mobile devices that enable them to connect to the Internet. Protocols such as Wi-Fi, Bluetooth, 3G, and 4G are all used in offloading solutions. Each of these connection protocols is now discussed their implementation and advantages. Finally, a comparison is made between the discussed connection protocols to determine which protocols can reliably be used for offloading.

5.4.1. Wi-Fi

Most modern mobile devices have a built-in Wi-Fi radio providing the device with access to Wi-Fi networks, and through the networks, access to the Internet. Wi-Fi has a 100m range and can transfer data at 20Mbps. Research has shown that a mobile device can download a 6GB file and upload a 5.6GB file using Wi-Fi before being completely discharged (Dinh et al., 2013; Fernando et al., 2013; Kalic et al., 2012).

MAUI, an offloading solution discussed previously in chapter 3, uses Wi-Fi as the connection protocol to offload fine-grained energy-aware mobile code to the cloud. MAUI utilises managed code to reduce the burden on programmers with regards partitioning the program while maximizing the energy savings. A smartphone application's code is duplicated and the copy is executed from the cloud infrastructure. The methods in the application are profiled and serialized to determine the cost of offloading. The offloading is done via RPC and although MAUI works via both 3G and Wi-Fi, the focus was on Wi-Fi. The results from the creators, Cuervo et al, shows that MAUI can reduce energy consumption (Cuervo et al., 2010).

5.4.2. Bluetooth

Like Wi-Fi, most modern devices have Bluetooth radios. These radios allow devices to connect to each other or to a Bluetooth network. The typical range of a Bluetooth radio is 10m depending on the strength and type of the device, and the physical objects between devices (Fernando et al., 2013). Using Bluetooth, a mobile device can download a 4GB and upload a 5.6GB file before completely losing charge (Kalic et al., 2012). The most recent Bluetooth protocol, Bluetooth 4.0, has been extended to include the Bluetooth Low Energy (Bluetooth LE) protocol. The entire Bluetooth 4.0 protocol is backward compatible, and the Bluetooth LE protocol is used to create a network between long-lasting sensors (Want et al., 2013).

As offloading is being used more often, more congested mobile networks would be created. A solution to the congestion problem is suggested by (Han et al., 2012). They suggest using opportunistic communication to offload data to the

network or server when the device is connected to a device that has the capability. If a specific device has the content of an application, that device can offload the content to other devices that use a shared application, using opportunistic communication. The solution suggests that if Bluetooth is used as the communication protocol, opportunistic communication can be done without incurring a monetary cost, while at the same time ensuring that the mobile networks are not congested. Using Bluetooth to find and initialize communication between devices has been shown to be the most cost-effective for battery life. Bluetooth communication, in general, uses less battery life than the other communication protocols, but it does not support large bandwidths.

5.4.3. 3G

3G, or Third Generation, is the third generation of mobile telecommunications. Protocols that meet the 3G standard include, Edge, HSDPA and HSDPA+ (3GPP, 2017) The standard defines that 3G has at least a 200kbps transfer rate. 3G is used by most devices for wireless voice telephony, mobile Internet access, fixed wireless Internet access, video calls and mobile TV technologies. The infrastructure that cellular networks use is owned and controlled by a service provider and access to the network is sold to consumers. 3G enables a mobile device to download 3GB and upload 1.4GB of data before being completely discharged (Alomari et al., 2011; Fernando et al., 2013; Kalic et al., 2012).

Mobile commerce (m-commerce) is a business model that leverages mobile devices for commercial purposes, such as, payments, messaging and ticketing. M-commerce applications use the mobility of mobile devices to fulfil the previously mentioned task. (Yang et al., 2010), suggest an m-commerce application based on cloud computing that uses 3G. The application on the mobile device communicates with the server's web service via 3G. The web service grants users access to their m-commerce information from virtually anywhere. M-commerce uses the 3G communication protocol and cloud computing to increase the speed and security of m-commerce. The main advantage of 3G over Wi-Fi is the nearly ubiquitous Internet access (Cuervo et al., 2010).

5.4.4. 4G

4G, or Fourth Generation, is the fourth generation of mobile telecommunications, and the successor of 3G. The 4G standard supports the same services as 3G. The increased speed of connections using 4G enables more services and is ideal for mobile cloud computing. There are currently two 4G implementations namely Long Term Evolution Advanced and Mobile WiMAX (Parkvall and Astely, 2009).

4G is the successor of 3G with all the advantages of 3G but also increased bandwidth. Because 4G is the successor to 3G, most of the applications that use 3G can also use 4G. Subramanian et al (2014) have developed an application that uses the increased bandwidth of 4G. The application proposed is an m-health application that uses a mobile device and certain accessories to monitor a patient and to offload the information to a server that also has the patient's medical records. The information stored on the cloud can be accessed by the patient's physician and can be used to increase the speed at which decisions are made. Because the application deals with the lives of patients, physicians need access to the latest medical records as soon as possible, which is made possible by using 4G.

The variety of connection protocols allow developers to create applications that require constant network access. Because mobile devices have the hardware to connect to more than one of these networks, developers can use different protocols at different times.

5.4.5. Comparison of connection protocols

Mobile devices can connect to all of the networks discussed, where a specific network is better suited for certain tasks. Table 5.2 compares the connection protocols. The table compares the range, speed, and energy consumption after two hours of use (Alomari et al., 2011; Cuervo et al., 2010; Fernando et al., 2013; Kalic et al., 2012; Parkvall and Astely, 2009; Want et al., 2013).

Table 5.2 Comparison of connection protocols

	Maximum range	Maximum throughput	Energy consumption (Download)	Energy consumption (Upload)
Wi-Fi	100m	866.7 Mbps	38%	38%
Bluetooth	10m	2 Mbps	19%	19%
3G (HSDPA)	N/A	7.2 Mbps	38%	34%
4G (LTE)	N/A	100 Mbps	N/A	N/A

From the comparison, it is clear that maximum throughput affects the energy consumed. Bluetooth, which uses the least amount of data, has the lowest energy consumption. Higher throughputs consume significantly more energy. However, throughput is not the only factor that plays a role. Comparing Wi-Fi and 3G, the energy consumption is the same for downloading which is a slight difference between the results from uploading. However, the throughput of 3G is almost a third of W-Fi. These factors will be taken to into account in the experimentation in this dissertation.

For this research, the communication protocol to be used between proposed components is both Wi-Fi and cellular networks, i.e. 3G and 4G. Wi-Fi is selected because the connection provided is stable and it has a relatively high bandwidth. However, as Wi-Fi is not mobile, cellular networks are also selected because they can be used when the device is not connected to a Wi-Fi network. Bluetooth is disregarded because of the low bandwidth.

The connection protocols have different advantages, where multiple advantages can be gained by using more than one connection protocol based on what is required at that point in time. The process of offloading, no matter the connection protocol, provides mobile devices with access to the resources on a server. Unfortunately, no advantage comes without some disadvantages or challenges. The next section discusses and evaluates various offloading approaches.

5.5. Offloading approaches

Offloading from a mobile device follows the same basic steps, however how these steps are completed differ based on the framework used. This section discusses the steps taken when offloading and compares different frameworks (Akherfi et al., 2016).

5.5.1. Offloading steps

The steps that are taken when offloading are application partitioning, preparation and making the offloading decision (Akherfi et al., 2016) discussed in this section.

a. ***Application partitioning***

Partitioning the application divides the application into two component categories. The first contains application components that can be offloaded and the second contains the components that cannot be offloaded. Dividing the application into the categories can happen in different ways depending on the framework chosen. Methods can be marked by the developer as being offload-able, or the methods or components can be identified using source code analysis with performance prediction or the application profiling. If the partitioning happens during development the accuracy of the partitioning is limited because it does not take the actual execution into account (Akherfi et al., 2016).

b. ***Preparation***

The next step, preparation performs all actions required for the components marked as offload-able to enable their use in mobile applications. This includes the selection of a remote server, the transfer, and installation of the code, as the start of proxy processes that receive and execute tasks on behalf of the mobile device. Setting up the cloud server is the first part of the preparation, there is also a seconding optional step, the transfer of data from the mobile device to the server. The data transferred in the second step can include the data required to execute the cloud on the server, and the data that describes the state of the mobile device (Akherfi et al., 2016).

c. *Offloading decision*

The final step before offloading is the offloading decision. Whether or not the offload-able component is offloaded depends typically on the execution context. If the decision is taken at runtime, more precise information is available, for example, the state of the device's network connection and the estimated energy consumption for transferring data at that time. Making the decision at runtime enables the re-evaluation of the decision whenever the state of the mobile device changes. However, runtime decision making does include some overhead costs that are not incurred when making the decision during development (Akherfi et al., 2016; Al-mousa and Alzoubi, 2017).

The steps that are required to offload an application or a component of an application does not vary, however, there are differences in how the steps are achieved depending on the framework chosen. The offloading frameworks are discussed next.

5.5.2. Comparison of mobile cloud computing offloading frameworks

There exist many mechanisms to achieve offloading, however, there are two framework categories that are prevalent when offloading tasks from mobile devices, namely frameworks based on virtual machine cloning, such as CloneCloud and frameworks based on code offloading, such as MAUI (Akherfi et al., 2016). MAUI and CloneCloud and the respective framework mechanisms are evaluated and compared in this section. The offloading steps of MAUI and CloneCloud are now examined.

a. *CloneCloud*

The offloading steps taken in CloneCoud, shown in figure 5.4. are discussed to further evaluate the solution presented by Chun et al. (Chun et al., 2011).

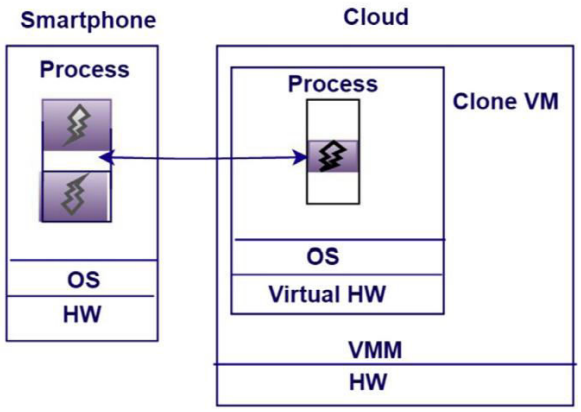

Figure 5.4 CloneCloud execution model

i. Partitioning

CloneCloud achieves partitioning by combining static program analysis with program profiling to produce a set of offload-able components while meeting some constraints, such as methods that require resources available only on the mobile device should be pinned to the device. CloneCloud uses thread level granularity for partitioning of applications. Static analysis is used to discover constraints on components, whereas profiling is used to build a cost model for offloading and execution. Partitioning and integration of application are performed at the application level (Chun et al., 2011).

ii. Preparation

The preparation step is achieved by creating a virtual instance of the mobile device on the cloud and transferring the necessary data to the clone before offloading. Initially, a duplicate of the smartphone's software is created in the cloud. The state of the smartphone and the clone is synchronized periodically or on-demand. As the partitioning happens at thread level all the threads are suspended and the states of the threads are transferred to the cloud. In figure 5.4 the Cloud VM is created in this step (Chun et al., 2011).

iii. Offloading decision

The offloading decision is made at runtime, once the decision has been made to offload the second step in the preparation occurs, namely the thread states are transferred. CloneCloud is based on virtual machine instance migration to the cloud server. Figure. 5.4 shows the CloneCloud execution model. After the

execution of offloaded components, results from the execution on the clone are reintegrated back into the smartphone state (Chun et al., 2011).

b. *MAUI*

The offloading steps taken in MAUI, shown in figure 5.5, are now discussed to evaluate the solution presented by Cuervo et al. (Cuervo et al., 2010).

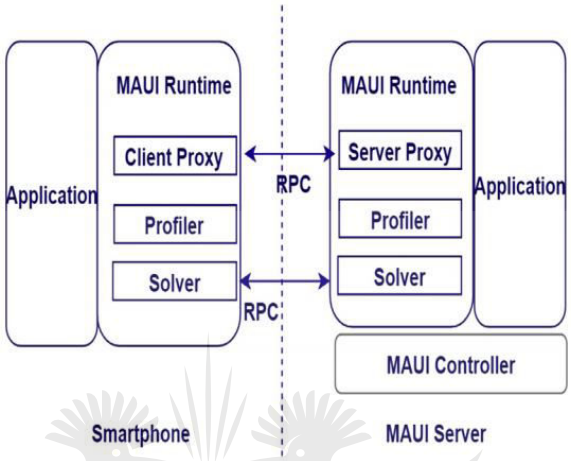

Figure 5.5 MAUI execution model

i. Partitioning

Partitioning is achieved in MAUI by developers who mark methods as offloadable during development. MAUI offloads code so the developer is required to create a cloud counterpart to the methods that are marked offloadable (Cuervo et al., 2010).

ii. Preparation

MAUI prepares for offloading by ensuring that the application is available on both the mobile device and the cloud server and also that proxies, solvers, and profilers are installed on the mobile device and the cloud server (Cuervo et al., 2010).

iii. Offloading decision

The MAUI profiler is used to create an initial profile of the mobile device characteristics, and continually monitors changes to the device and the network to update the profile if the profile is not kept current wrong decisions can be made. The offloading decision is taken at runtime. The framework chooses

which components should be remotely executed according to the decision of the MAUI solver. The decision is based upon the input of the MAUI profiler. Figure 5.5 shows the MAUI execution model.

c. *Comparison of mobile cloud computing offloading frameworks*

The comparison of MAUI and CloneCloud offloading is given in table 5.3. Both MAUI and CloneCloud achieve offloading, but by using different mechanisms. Both frameworks make use of dynamic decision making. MAUI uses information that is gathered by the MAUI profiler and then fed to the MAUI solver to determine whether or not a method should be offloaded. CloneCloud uses a built-in profiler to migrate threads to the clone on the cloud. Virtual machine cloning is expensive in terms of computation and storage on the cloud, but such costs can be covered by the use of container technology (Bernstein, 2014). As stated before, code offloading moves the burden of identifying offloadable components to the developer. Thread level granularity introduces complexity, as there is more information required to be transferred and the migration of the result back into the local instance can negatively influence the user experience.

Table 5.3 A comparative review of MAUI and CloneCloud

Framework	Comm. Protocol	Optimization Factor	OS	CMA approach	Initial data transfer	Partitioning	Preparation	Decision	Offloading mechanism	Contribution	Level of granularity
MAUI	Wi-Fi, 3G	Energy, Execution Time	Windows Phone	Distant fixed	Low	Individual method annotation	Two versions of a mobile application are created	Dynamic	Code	Energy-aware code offload	Method
CloneCloud	Wi-Fi, 3G	Energy	Android	Distant fixed	High	Static program analysis and program profiling	A duplicate of mobile device's software stored on the cloud server	Dynamic	Virtual machine cloning	Elastic execution between mobile devices and the cloud	Thread

The solution proposed by this dissertation uses code offloading on a method level of granularity to reduce complexity. Decisions making is discussed in the next chapter in more detail. The challenges associated with offloading and enabling offloading are discussed next.

5.6. Challenges of offloading

Offloading provides many advantages when applied to traditional computers. When offloading is done from a mobile device, the number of advantages greatly increases. To gain the advantages of offloading, certain challenges need to be overcome, as discussed next.

Offloading requires at least two types of devices to be implemented namely a server and client. For this research, the two devices are the cloud and the mobile device. The challenges facing offloading is two-sided; from the viewpoint of mobile communication, and from the viewpoint of computing (Dinh et al., 2013). The challenges discussed here are focussed from the viewpoint of the mobile device. The challenges related to the server-side components and the challenges of mobile cloud augmentation were discussed in the previous chapter.

The challenges discussed below are related to mobile device communication which hinders offloading. The challenges discussed are low bandwidth, availability, heterogeneity, and security.

5.6.1. Low bandwidth

The ability of mobile devices to connect to the Internet and other networks has been well established. The first challenge that needs to be overcome when offloading is the bandwidth available to the mobile device. Cellular networks are nearly everywhere but do have an associated cost and do not have the reliable bandwidths. In contrast, Wi-Fi can provide network access at greater bandwidth but does not cover such a large area as the mobile networks (Cuervo et al., 2010; Dinh et al., 2013).

5.6.2. Availability

The availability of network connections is the second challenge that needs to be overcome. Because mobile devices are resource constrained, the importance of network availability increases as devices may not be capable to complete complex tasks without offloading. As previously stated, mobile data networks provide nearly complete coverage, but areas without network access can still exist. Wi-Fi access points provide network access when the user is in the limited range, forcing a mobile user to use Wi-Fi, which severely limits the mobility of the user (Dinh et al., 2013; Huerta-Canepa and Lee, 2010).

5.6.3. Heterogeneity

The heterogeneity of the different types of networks that can be used to offload is the third challenge that needs to be overcome. The different networks and communication protocols need to be considered when the offloading server is created. Different servers have different requirements and meeting these requirements can have influenced the capabilities of the application. (Dinh et al., 2013)

5.6.4. Security

The security of the data being stored, sent or processed is the fourth challenge. The security on a mobile device can be augmented by third-party applications but there is still a risk involved in having sensitive data on a small mobile device. Not only is the device vulnerable to cyber-attacks but also to physical threats, the device can destroy or stolen quite easily (Dinh et al., 2013; Oberheide et al., 2007).

The challenges discussed here are daunting, but the widespread use of offloading shows that the challenges can be, and have been overcome as advances in the fields relating to these challenges are reducing the impact and the size of the challenges.

The last component of the proposed solution, namely the decision to be taken when offloading should occur, is of utmost importance. The factors that influence the offloading decision making are discussed in the next chapter.

5.7. Conclusion

This chapter discussed offloading, focussing on the offloading from mobile devices to the cloud or mobile cloud. The methods that are available to offload, the connection protocols used to connect mobile devices to the networks required for offloading, the challenges that need to be overcome when offloading and the factors that influence the offloading decision were also discussed.

The use of offloading with mobile devices is an extension of the offloading used by traditional computers. By using offloading to augment mobile devices with the resources available on the cloud, more complex applications can be developed for mobile devices.

The methods available to offload processes from mobile devices to the cloud can be categorized into three overarching methods. These are Client-Server, Virtualization, and Mobile Agents. These methods each have advantages and disadvantages, and each uses a different computer or device as a server. The advantages and disadvantages make the approaches more useful in some situations and less so in others. In the case of the solution proposed by this dissertation, the client-server method is best suited.

The connection protocols available are discussed as they are integral to offloading. The different protocols are all used by mobile devices and have been used to create applications that offload processes from mobile devices. The variety of connection protocols can be used in concert to keep a device connected to a network continuously. The examples of applications that use the different communication protocols made the different advantages clear.

Offloading frameworks typically take one of two approaches when offloading. The first is virtual machine cloning, which relies on creating a virtual instance of the mobile device and migrating threads and data regarding the state from the physical device to the cloud for execution, once execution is completed the thread is migrated back to the physical device and merged into the application.

The second approach is code offloading, which relies on the creation of a function on a cloud server that executes instead of the local method. CloneCloud uses the first approach, whereas MAUI uses the second approach.

Offloading is used to increase the resources of the client but the increase of resources comes with some challenges. These challenges, focussing on the mobile device, are discussed. The low bandwidth and availability of network connections on the devices cannot be influenced by users but should be considered by app developers. The difference of the networks that can be used on mobile devices should be also be considered because the requirements of the connection protocols vary. Lastly, the security of the data being offloaded should be considered because once the data leaves the device the owner is no longer control of what happens to the data.

The goal of the offloading of processes is to augment mobile devices with cloud resources, where offloading becomes necessary because of the limited resources on mobile devices. As limited resources are not always conserved when processes are offloaded, the decision of whether or not a process should be offloaded is discussed in chapter 6.

Chapter 6: Decision Making

6.1. Introduction

Studies (Barbera et al., 2013; Kumar and Lu, 2010) have shown that offloading can be used to conserve the battery life of mobile devices in the right circumstances. As discussed in the previous chapter, offloading is the process of moving the execution of a process from a resource-poor client to a resource-rich server. Before a process is executed, a decision has to be made to either offload the process or execute the process on the mobile device. The decision maker evaluates the circumstances and chooses whether or not to offload. The decision made can conserve the battery life of the mobile device and extend the length of time the mobile device can be used.

The previous chapter discussed offloading and the fact that the right offloading decision can conserve battery life. This chapter completes the literature review by defining decision making and identifying the factors that can influence the offloading decision. To achieve these aims, section 6.2 defines decision making and section 6.3 discusses the process of decision making. In section 6.4 the factors that influence the offloading decision are discussed. Section 6.5 discusses and evaluates the decision-making process followed in related research. Finally, the chapter is concluded.

6.2. Definition: Decision making

The study of decision making influences many intellectual disciplines such as mathematics, statistics, economics, political science, sociology, psychology and computer science (Kahneman and Tversky, 2000).

Decision making is the process of identifying and choosing an option between several alternative options based, on certain factors and the goal of the decision maker. The decision making process results in a final choice that determines the actions of the decision maker (Kahneman and Tversky, 2000; Zsambok and Klein, 2014)

The main goal of this research is to develop a component that can make offloading decisions to conserve battery life, which can be integrated with an existing application. An offloading decision can be considered as good if the offloading of a task conserves more energy than when a local computation is performed.

The definition indicates that there are three components that are required to make decisions namely goals, options, and factors. These components are discussed next.

6.3. Decision making

A decision-making component chooses between options based on certain factors to achieve a goal, identified next.

6.3.1. Goal

Decisions are made to reach a goal (Kahneman and Tversky, 2000). In this dissertation, the goal is to consume as little energy as possible by choosing the less expensive option in terms of energy consumed.

6.3.2. Options

The options or choices for this research are between deciding to offload or not to offload. The decision is made based on factors that influence the energy consumption cost of each option.

6.3.3. Factors

Factors that influence the energy cost of offloading and local execution include the size of the data (being sent and received), the bandwidth available, the communication protocol that is used, the complexity of the code and the duration of executing the processes locally (Kumar and Lu, 2010).

The factors ultimately determine whether or not the decision maker achieves the goal. The factors and how they are measured are discussed next.

6.4. Factors influencing the offloading decision

The offloading of a mobile task is a trade-off between the energy consumed when executing locally, and the energy needed for offloading the task, as well as uploading and downloading the relevant data (Kumar and Lu, 2010). It is expressed as the equation in equation 6.1.

Equation 6.1 Offloading decision-making equation

$$E_{local} - E_{offloading} = E_{saving}$$

$$E_{saving} > 0$$

Where:

- E_{local} represents the cost of local execution.
- $E_{offloading}$ represents the energy used by offloading the task.
- $E_{saving} > 0$ determines that energy will be saved if the task is offloaded.

The factors can be divided into two categories. The first is *communication*, which consists of the size of the data, the bandwidth and the communication protocol used. The second is a *computation*, which consists of code complexity and the time needed to execute (Kumar and Lu, 2010). The factors and how they are measured are discussed next. After each of the categories are discussed, the equations that represent the energy costs are presented.

6.4.1. Communication

Communication represents the cost of offloading a process. This factor determines how much battery life is used if all the necessary information is sent to the cloud and the result is returned. The cost of the communication mainly depends on the length of time of the communication. The length of the communication is determined by two factors namely the size of the data and the available bandwidth. A third factor is the communication protocol used where the communication protocol determines how much energy is used to communicate (Cuervo et al., 2010; Kalic et al., 2012; Ma et al., 2013).

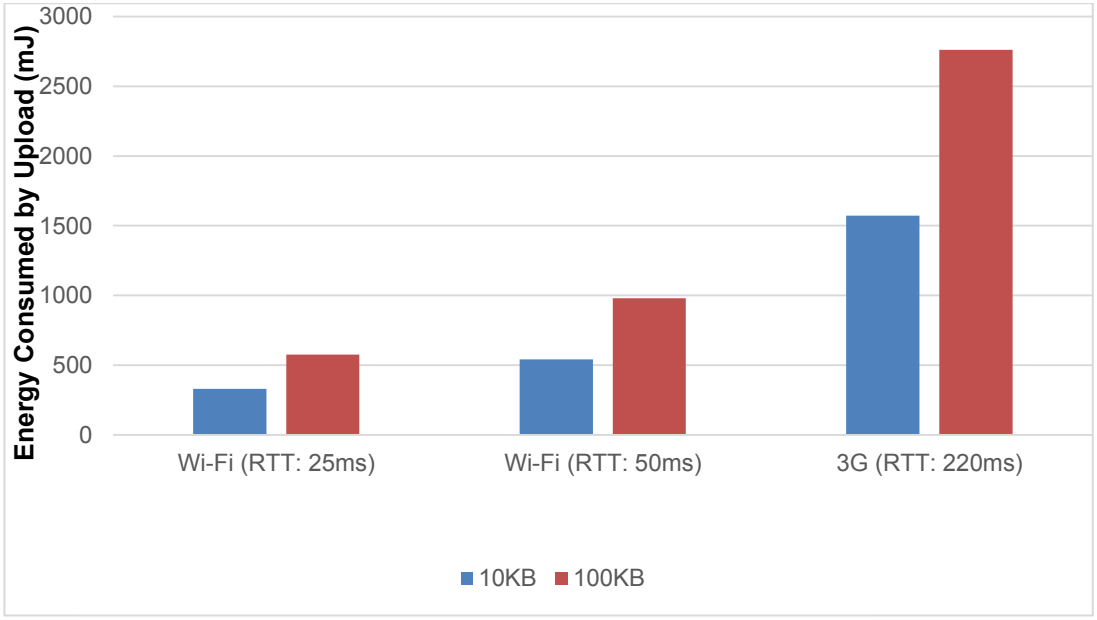

Figure 6.1 Energy Consumption: Wi-Fi vs. 3G

Figure 6.1 shows the power consumption when transferring data via Wi-Fi and 3G. The blue bars, to the left of each grouping, represent the energy consumed when transferring 10KB files and the red bars represent transferring 100KB files. The Round-Trip-Time (RTT) given is the time a message takes to be sent to the server and the time for the server's response to return. The longer the RTT is, the longer the connection is required to stay open. The comparison of the two files in each case shows that the larger the file, the more power is consumed. The first two data sets show the power consumption when transferring data via Wi-Fi, whereas the third data set shows the consumption when transferring data via 3G. It is clear that a 3G connection consumes more energy than a Wi-Fi connection.

a. *Size of data*

Both the size of the data and available bandwidth determines the length of the communication (Kumar et al., 2013). The longer the communication, the more power is used. The effect on energy consumption of communication is shown in Figure 6.1. The size of the data that is transferred determines the time the connection is required to be open. The larger the size of the data transferred, the longer the connection is kept open. The data transferred consists of the objects sent and received, and the size of the headers associated with the communication protocol (Cuervo et al., 2010).

The objects which are sent can either be primitive or composite data types. Primitive data types have set sizes and are used to store a single value. Composite data types are more complex and can store many different related values. At runtime, the values of the composite data type are set, which allows the measurement of the size of the composite data type (Cleveland, 1993; Sale, 1977).

The size of the data to be communicated can be measured using different methods. The first method examines the attributes of an object and totals the size of the primitive data types. If the objects are instances of composite data types, the calculation is done recursively for all of the objects that are sent and all of the attributes. The size of the objects sent and received does not include the size of the transportation headers. This method can be used to estimate the size of the data to be communicated before the communication takes place (JavaWorld, 2003).

The second method monitors the connection and counts the number of bytes that are transferred. This measurement takes into account transportation headers and does not have to calculate the size of the serialized objects being transferred. However, the measurement is made after the communication takes place (StackOverflow, 2014).

This research chooses the second approach for the experiments performed because the loss of packets is taken into account by measuring the size of the communication, as opposed to calculating the size of the communication.

b. *Communication protocol*

The communication protocol used directly affects the bandwidth available. Cuervo et al (2010) have done studies on the effects on power usage between mobile data networks. The results of these studies are shown in Figure 6.1.

Mobile devices connect to different types of networks using the appropriate communication protocol, which determines the maximum bandwidth. However, the energy cost for each communication protocol differs. Figure 6.2 below

shows the energy consumption of a mobile device when communicating using different communication protocols to upload and download (Kalic et al., 2012).

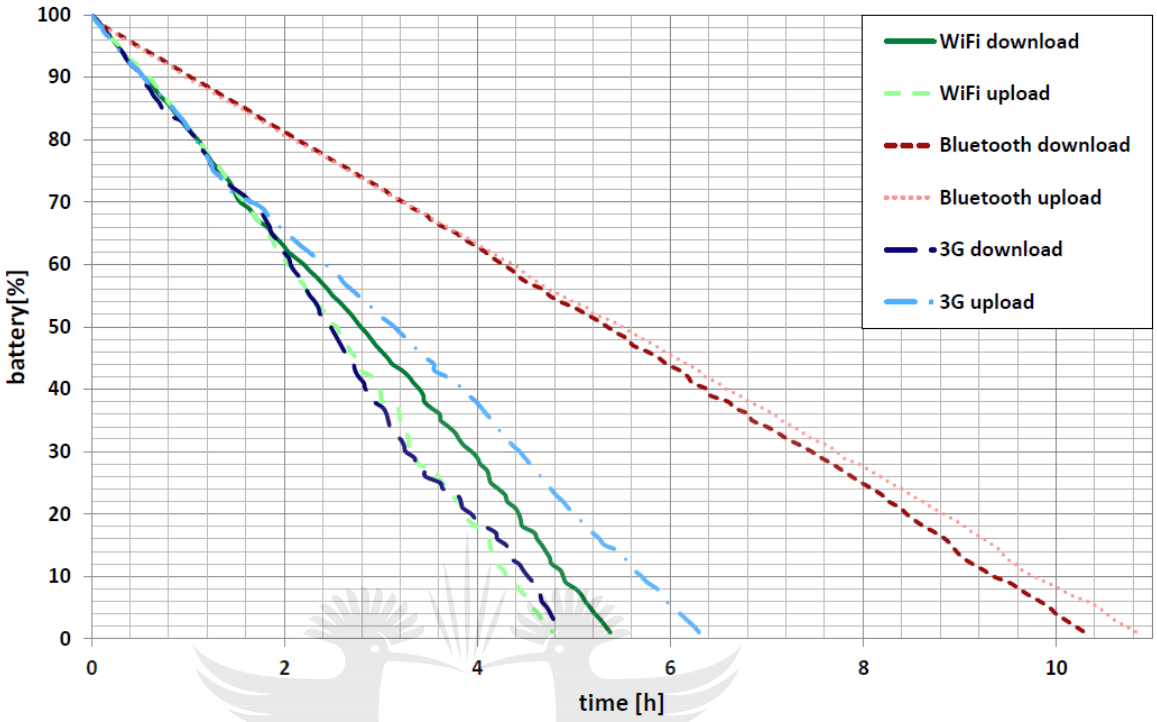

Figure 6.2 Energy consumption compared to the elapsed time

Figure 6.2 shows the energy usage for different connection protocols. A mobile device can continuously communicate via Bluetooth, on the far right, for more than 10 hours. However, the same mobile device can only download via Wi-Fi, in the middle (third from the left), for a little over 5 hours, and upload, first on the left, for even less time. Figure 6.2 illustrates the point that the energy consumption differs from communication protocol to communication protocol.

c. *Bandwidth*

As previously stated, the greater the bandwidth, the less time the communication needs. Figure 6.1 shows the power consumption under different available bandwidth (Ma et al., 2013). The bandwidth available determines the time required to keep open the connection, which determines how much battery life is used.

The available bandwidth is determined by the connection protocol used and the signal strength. The higher the bandwidth, the shorter the time needed by the

communication. For instance, sending a 1 megabyte package over a connection with 1 Mbps bandwidth takes 8 seconds, whereas sending the same package over a 1 Kbps bandwidth connection takes longer than 2 hours (Kalic et al., 2012).

Bandwidth determines the maximum speed at which packets can be transferred. The bandwidth available on a network is calculated by sending packets through the network and measuring the time the packets take to reach their destinations.(Johnsson et al., 2006; Johnsson and Bjorkman, 2008).

The underlying network influences the speed by which packets are sent. The Internet is a TCP/IP network (Transmission Control Protocol/Internet Protocol) (Wright and Stevens, 1995) that uses a slow start algorithm to reduce congestion (Zhang et al., 2012). Once a connection is opened to the server a congestion window is opened, for each acknowledgment received the congestion windows is enlarged. The maximum speed at which the communication can take place is determined by the congestion window up to a certain point, and then the bandwidth. The use of the slow start algorithm limits the speed of transferring smaller files because the communication is over before the congestion window's size meets the bandwidth maximum (Zhang et al., 2012).

Available bandwidth is to be measured in this dissertation by downloading and uploading files to the server, and measuring the time the communication takes. The slow start algorithm can limit the available bandwidth calculation. If a small file is used to measure the bandwidth, the communication is done before the congestion window's size reaches the actual bandwidth limit.

Size of data, bandwidth and communication protocol influences how long the communication takes place. The longer the connection is open, the more battery life is used. The energy cost of communication can be measured by multiplying the length, in time, of the communication with energy usage per second.

The communication cost is expressed in equation 6.2.

Equation 6.2 Communication cost equation

$$communicationTime = \frac{size}{bandwitdh}$$

$$E_{communication} = communicationTime \times protocolConsumption/second$$

Where:

- $size$ represents the size of the data that is uploaded and downloaded.
- $bandwitdh$ represents the current available bandwidth.
- $communicationTime$ is the length of time the transfer of data takes at the current bandwidth.
- $protocolConsumption/second$ represents the energy consumption per second for different connection protocols.

$protocolConsumption/second$ is a constant per device, per connection protocol. The cost of the communication needs to be compared to the cost of executing a process locally. The comparison determines whether or not the process should be offloaded. The computation of a mobile task, which is the cost of not offloading, is discussed next.

6.4.2. Computation

Computation determines how much battery life is used to execute a mobile task locally (Kumar and Lu, 2010). The cost of computation is determined by the complexity of the procedure to be executed and the length of time the execution takes.

a. *Code complexity*

Complexity encompasses many properties, such as the number of paths through the code, the number of operations and the number of variables, of a piece of code. Each of the properties affects the interactions between components. There is a difference between the complex and complicated code. Complicated code is difficult to understand, whereas complex code has many interactions between different components (De Silva et al., 2012; Henry and Kafura, 1981; Yu and Zhou, 2010).

Several methods exist to measure the complexity of a program or part thereof. McCabe's cyclomatic complexity (De Silva et al., 2012), Halstead complexity measures (Halstead, 1977) and Big-O notation (Stephens, 2013), are discussed.

i. McCabe's cyclomatic complexity

McCabe introduced the cyclomatic complexity measure. Cyclomatic complexity determines the number of linear paths through the code. Each if statement in the code creates a different path. A program's code is more complex the more paths there are through the code. Cyclomatic complexity is also an indicator of testability and maintainability (De Silva et al., 2012).

ii. Halstead's complexity measures

Halstead's complexity measures are based on the number of operands and operators in a module. Operands are objects in the module on which operators execute operations. The metrics gathered by Halstead's measure can be used to determine the volume, difficulty, effort, time to program and the number of delivered bugs (Halstead, 1977).

iii. Big-O notation

Big-O notation measures the scalability of code, how long execution takes as the number of input values increase. Big-O notation is not a direct measure of complexity but does give an indication of complexity and how long execution takes. Common results from big-O investigations include O(1), O(n), O(n^2), O(log(n)), and O(nlog(n)).The order of the function determines how the number of input values (n), influence the execution time. For instance order of 1 (O(1)) means that the execution takes the same amount of time regardless of the number of input values and order of N (O(n)) means that the execution time is directly related to the number of input values (Stephens, 2013).

The more complex code takes longer to execute. Complexity influences an application's execution time, but cannot accurately estimate the execution time. The code complexity gives an indication of execution time in terms of the

number of inputs, however, the hardware on which the code is executed and the number of inputs vary and cannot be predicted.

b. *Length of time to execute*

Execution time is influenced by the hardware resources available. When executing locally on a mobile device resources are limited and the task will take more time to complete. However, when the task is offloaded to the cloud, the abundance of resources reduces the length of execution time.

Time to execute can easily be measured. The current time is stored before the application or code is executed and again after the execution. The time to execute is calculated by subtracting the time in the beginning from the time at the end (StackOverflow, 2010).

Java and Android have the ability to provide the current time of the system in milliseconds. By using the time in milliseconds, the time to execute can accurately be measured. Time to execute is measured, in the implementations in this dissertation, by using the current time in milliseconds for accuracy. Time to execute is multiplied with an energy cost of the processor per second to result in the energy cost of not offloading.

The computation cost is expressed in equation 6.3.

Equation 6.3 Computation cost equation

$$E_{computation} = computationTime \times computationConsumption/second$$

Where:

- $computationTime$ is the length of time the local execution takes.
- $computationConsumption/second$ represents the energy consumption per second for local processing.

Given the expression of $E_{computation}$ and $E_{communication}$ the trade of between local execution and offloading can be expressed as:

Equation 6.4 Offloading decision making equation wdecision-making values

$$E_{computation} - E_{communication} = E_{saving}$$

$$E_{saving} > 0$$

It is important to note that $computationConsumption/second$ is a constant that is device specific. Next, a review of related research on offloading decisions is given to provide a foundation for this research.

6.5. Comparison of decision-making for offloading approaches

A large body of research focuses on implementations that can make offloading decisions. Of these, the researcher identified that the most representatives is MAUI and CloneCloud (Chun et al., 2011; Cuervo et al., 2010). Their goals, options, and factors are discussed and evaluated in this section to provide a foundation for the proposed solution.

6.5.1. CloneCloud

CloneCloud (Chun et al., 2011) creates a virtual instance of a mobile device and offloads the state of the device to the virtual instance when a method needs to be executed on the cloud. The goal of CloneCloud is to conserve battery life. The options are between deciding to offload or not to offload. The factors that influence the offloading decisions are execution time and energy usage on the mobile device. The profiler and optimization solver used by CloneCloud are discussed below (Chun et al., 2011).

a. *Profiler*

CloneCloud creates profile trees, as shown in Figure 6.3 (b), through random executions of the application, while running the profiler. The two trees, one for local execution and one for execution on the clone on the cloud, are compared and if the execution on the cloud is less expensive in terms of battery life, the method is offloaded to the cloud. Each node of a tree represents a method invocation in the execution; it is rooted at the starting method invocation of the application. Specific method calls in the execution are represented as edges from the node of the caller method invocation to the nodes of the callees. Each

node is annotated with the cost of its particular invocation in the cost metric, such as execution time in the case of CloneCloud. In addition to its called-method children, every non-leaf node also has a leaf child called its residual node. The residual node represents the cost of running the body of code excluding the costs of the methods called by it. Each edge is annotated with the amount of data that will be transferred to and from the cloud if the edge were to be a migration point. (Chun et al., 2011).

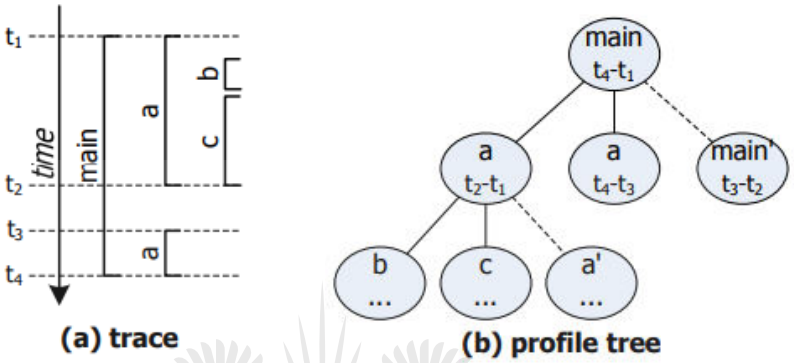

Figure 6.3 An example of a CloneCloud trace (a) and profile tree (b)

Figure 6.3 (a) shows the method trace of the execution of the method *main*. In *main* the method, *a* is called twice. The first invocation of *a* invokes methods *b* and *c*. The second invocation of the method *a* invokes no other methods. Figure 6.3 (b) shows the profile generated from the execution of the *main.* The tree nodes in figure 6.3 (b) contain the execution time of the corresponding method in the trace (the length of the square bracket on the left of figure 6.3 (a)). Node *main'* and node *a'* are residual nodes, which hold the difference between the value of their parent node and the sum of their sibling nodes.

The profile tree is filled by temporarily creating an application-method entry and exit points during each profile run on both platforms. The execution-time cost metric is collected from timings of method entry and exit points. Migration costs and edge weights are calculated by simulating migration at each profiled method and calculating the time the migration takes. The execution-time cost metric represents the length of time local execution takes whereas the migration cost represents the length of time migration takes. These metrics are multiplied

by the power estimation function to give an estimate of energy consumption (Chun et al., 2011).

CloneCloud calculates the power consumption by using a model that can be expressed as:

Equation 6.5 CloneCloud power consumption estimation

$$powerConsumptionEstimate = P(CPU, Screen, Network)$$

Where

- CPU represents whether or not the CPU on the mobile device is active
- $Screen$ represents whether or not the screen is on
- $Network$ represents whether or not the mobile device is transmitting data of the network

The computation cost of each node is calculated by taking the average time of execution with the estimated power consumption per second. The estimated power consumption per second is calculated by considering:

- CPU activity.
- Display state.
- Network state.

The function used to estimate power consumption is compared to the measurements taken from an external power meter.

The migration cost is calculated by considering:

- The size of the state before migration takes place.
- The time required to suspend and resume the thread being migrated.
- The size of the state when migrating from the cloud takes place.

A pre-calculated per-byte cost is used with the size of the state to calculate the migration cost (Chun et al., 2011).

The local computation cost is calculated by using the formula:

Equation 6.6 CloneCloud local execution power estimation

$$E_{local} = P(true, true, false) \times localExecutionTime$$

The offloading cost is calculated a by using the formula:

Equation 6.7 CloneCloud offloading power estimation

$$E_{offloading} = P(false, true, false) \times cloudExecutionTime + P(true, true, true) \times migrationTime$$

Where:

- $localExecutionTime$ represents the length of time of local execution.
- $P(true, true, false)$ represents the power consumption when the CPU is active, the screen is on and the device's network is idle.
- $cloudExecutionTime$ represents the length of time execution takes on the cloud.
- $P(false, true, false)$ represents the power consumption when the device is idle and the screen is on.
- $migrationTime$ represents the length of time it takes for a thread to migrated to and from the cloud.
- $P(true, true, true)$ represents the power consumption during migration, the CPU is active, the screen is on, and the device is sending/receiving data.

The formulas given above are used to populate the profile trees used by the optimization solver, discussed next.

b. *Optimization Solver*

The purpose of the optimizer is to decide which application methods to offload, so as to minimize the expected cost of the partitioned application. Given a particular execution *E* and its two profile trees *T* on the mobile device and *T'* on the clone, the task can be pictured as optimally replacing annotations in *T* with those in *T'*, to minimize the total node and weight cost of the hybrid profile tree. The decision evaluates E_{local} and $E_{offloading}$ to determine whether or not a method should be offloaded.

The output of the optimizer is a value assigned to binary decision variables $R(m)$, where $R(m) = \{0,1\}$ and m is every method in the application. If the optimizer chooses $R(m) = 1$ the partitioner places a migration point at the entry

into the method, and a re-integration point at the exit from the method. If the optimizer chooses $R(m) = 0$, method m is unmodified in the application binary. Every invocation of a method is either offloaded or not offloaded, for simplicity's sake (Chun et al., 2011).

6.5.2. MAUI

MAUI (Cuervo et al., 2010) developed for Windows Phone is divided into two components, the Profiler and Solver. The Profiler is used to gather data about the state of the mobile devices and all factors, where the Solver uses the data gathered to make the offloading decision. The goal of using MAUI is to reduce execution time and energy usage. The options are between deciding to offload or not to offload. The factors that influence the decision are the device's energy usage characteristics, the application's characteristics (running time and resource requirements of individual methods) and the characteristics of the wireless network the device is connected to (bandwidth, latency, and packet loss). The two components are discussed in the next sections (Cuervo et al., 2010).

a. *Profiler*

The Profiler gathers information regarding the device, the application, and the network to inform the MAUI Solver. The device profile, program profile and network profile are used to estimate power consumption on the mobile device. The profiles and how they are created are discussed next.

i. <u>Device profile</u>

The device profile is defined by information that is collected when tasks consume energy as they execute on the mobile device. The measurements that are used to create the profile are obtained by attaching an external power meter to the mobile device during the execution of various tasks. The CPU usage with the measured power consumption is used to create a linear model. The linear model is validated by comparing the prediction of energy consumption with the actual measurements produced by a hardware power monitor. The external power meter is also used to create models for the consumption when transferring data over different networks (Cuervo et al., 2010).

ii. Program profile

The application profile consists of:

- The size of data that is required to be sent to the server.
- The size of the data that is returned after the execution.
- The number of CPU cycles required to execute the method locally.

The number of CPU cycles and length of time of local execution is used with the device profile to estimate the cost of local execution. The size of data that is sent and received from the cloud is used with the device and network profiles to calculate the cost of offloading.

The program profile continuously monitors the application as it executes and updates the program profile with every new execution. During the execution of the application, the length of time between offloads influences the amount of data that is required to be transferred to the cloud. For example, if a method was just offloaded, the cloud already has the state of the device. If the next method to be executed is also offloaded, there is no need to transfer the entire state data again (Cuervo et al., 2010).

iii. Network profile

The network profile consists of the average throughput from the application to the server. The average throughput is the only factor stored in the network profile because the round trip time, bandwidth and packet loss all influence the throughput. The throughput is calculated by sending a 10KB file to the server and measuring the length of time it takes to complete. Whenever a method is offloaded the network profile is updated with the latest throughput calculated. If a minute passes without a method being offloaded a 10KB file is uploaded again to update the network profile (Cuervo et al., 2010).

The local computation cost is calculated by using the formula:

Equation 6.8 MAUI local execution power estimation

$$E_{local} = numberOfCPUcycles \times energyCostPerCPUcycle$$

The offloading cost is calculated a by using the formula:

Equation 6.9 MAUI MAUI offloading power estimation

$$E_{offloading} = \frac{sizeOfStateData}{availableThroughput} \times networkEnergyCost$$

Where:

- $numberOfCPUcycles$ represents the number of local CPU cycles to execute a task, as measured by the program profile
- $energyCostPerCPUcycle$ represents the energy consumption cost for each local CPU cycle, as measured by the device profile
- $sizeOfStateData$ represents the size of the state data, as measured by the program profile
- $availableThroughput$ represents the average throughput available on the device, as measured by the network profile
- $networkEnergyCost$ represents the energy cost per second to transfer data to the cloud, as measured by the device profile

b. *Solver*

The decision made by the MAUI solver is globally defined across the application and not on a per-method basis, the data gathered by the Profiler is used to calculate the cost of offloading the state at a certain point before a method is called and that is compared to the cost of executing the method locally. Because a global approach is used, the solver can determine when the same data will need to be transferred multiple times if a 'per method' approach is used. By partitioning the application, the state of the entire process at a certain point in time can be offloaded without needing to return to the mobile device. As the application runs, the solver is re-run periodically for two reasons: to adapt to changing environmental conditions, and also to learn from the historical behaviour of the program. The MAUI solver is invoked asynchronously from the mobile device to avoid affecting the interactive performance of the application. Figure 6.3 shows the simplified approach that MAUI takes (Cuervo et al., 2010).

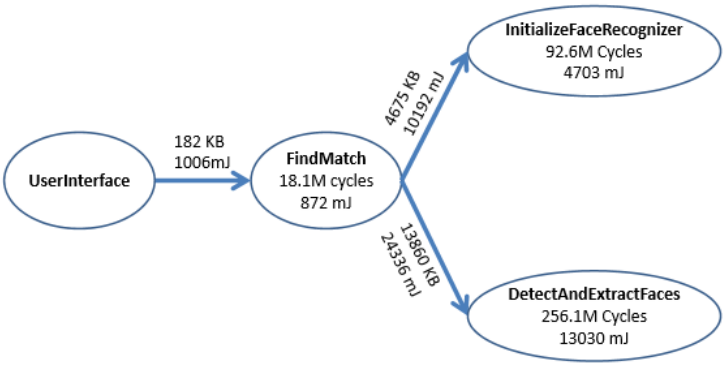

Figure 6.4 MAUI offloading decision-making process

In figure 6.3, each vertex represents a method and its computational and energy costs, and each edge represents the size of the method's state and the energy consumed to transfer the state to the cloud. In the example given in figure 6.3, executing the FindMatch method will consume 18.1 million CPU cycles, using 872 mJ of power. For the same method, 182KB of data is required to be transferred to the cloud, which consumes 1006mJ (Cuervo et al., 2010). MAUI and CloneCloud are compared in the next section.

6.5.3. Comparison of decision-making for offloading approaches

MAUI and CloneCloud both aim to reduce power consumption on mobile devices by offloading tasks to the cloud. The reduction in power consumption is achieved by offloading tasks that will consume more power when executed locally compared to the power consumption of transferring the necessary data to the cloud for remote execution. The methods that should be offloaded are decided by the solvers in each case. To make the decision, the local cost and the offloading cost are calculated and compared.

When estimating the cost of local execution, MAUI considers the length of time of local execution and the number of CPU cycles, where CloneCloud only considers the length of time execution. Considering the number of CPU cycles is not necessary as it has a direct relation with the duration of execution.

When estimating the cost of offloading, both MAUI and CloneCloud consider the size of the data that has to be transferred to and from the cloud. The size of the

data is converted to the length of time the communication will take in both cases. MAUI uses the throughput of the network available to calculate the duration of the communication. CloneCloud measures the cost of transmitting data to create a per byte cost that is used to populate a profile tree. Table 6.1 shows a comparison of the approaches used by MAUI and CloneCloud.

Table 6.1 Comparison of MAUI and CloneCloud

	Local computation factors	**Offloading factors**	**Energy consumption profile creation method**	**Energy consumption model**
MAUI	Duration of local execution Number of CPU cycles	Size of data Available throughput	Hardware-based	Linear model
CloneCloud	Duration of local execution	Size of data	Hardware-based	Method

The duration of local execution and offloading alone is not indicative of energy consumption. The durations are used with the device profile's linear model, in the case of MAUI, and the energy consumption model, in the case of CloneCloud. Both models are created by evaluating the results from measurements taken from an external power monitor. The MAUI linear model uses the duration of CPU usage or the number of CPU cycles to estimate energy consumption, where the method used by CloneCloud takes more factors, such as whether the screen is on, if the CPU is active and whether the device's network radio is in use, into account as well. The durations calculated by the profilers from each approach are substituted into the energy consumption models to estimate the cost of either offloading or local execution. The energy consumption model is crucial to the offloading decision making.

Measuring the duration of local execution and measuring the size of data being transferred are trivial exercises given the tools available to developers. The accuracy of energy consumption estimation is thus based on the accuracy of

the energy consumption profile used. The methods of profiling the energy consumption of a mobile device are discussed next.

6.5.4. Energy consumption profiling

Profiling the energy consumption of a mobile device characterizes the energy consumption when the mobile device is in use. To create a profile of a device's energy consumption two approaches can be used, namely software based and hardware based energy consumption profiling. The two approaches are discussed in this section (Ahmad et al., 2015).

a. *Software-based energy consumption profiling*

Software-based energy consumption profiling uses a software module to collect a component's power usage statistics to construct power models to estimate energy consumption. The energy profiler estimates the energy consumption either at the process, thread, path, or source-code line level. Software-based profiling can use measurements that can come from a mobile app, the operating system of the mobile device or in some cases the battery used by the mobile device (Ahmad et al., 2015).

b. *Hardware-based energy consumption profiling*

Hardware-based energy consumption profiling utilizes external hardware equipment to obtain voltage and current readings to estimate the power consumed by a mobile phone against system activities. Measurements taken by the external hardware can be taken at various granularities, namely application, path/line, process, or thread. The measurements taken can be used to create models in a deterministic or statistical manner. The deterministic approach approximates mobile power consumption based on the power state machine. The power state machine modelling method also estimates mobile power consumption by employing the per-state energy model and hardware component transition states. Statistical power modelling uses pre-built statistical models to estimate software/mobile power consumption (Ahmad et al., 2015).

c. *Proposed energy consumption profile*

Both MAUI and CloneCloud use hardware-based approaches when creating energy consumption profiles. Hardware-based approaches are expensive,

labour intensive and are not scalable when compared to software-based approaches (Ahmad et al., 2015).

This research now identifies the following additional requirements for the proposed solution to address the identified energy consumption profile:

- Software-based: To enable developers to create and use an energy consumption profile on any device, a software-based energy consumption profiling method should be used.
- Continuously monitor factors: The factors that influence energy consumption should be continuously monitored to ensure the decisions made are correctly at any point in time.
- Lightweight: The model representing the proposed energy consumption profile should be lightweight, to prevent the excessive consumption of additional energy and do not negatively impact the user experience.

MAUI and CloneCloud are large frameworks that enable the conservation of battery life on mobile devices by supporting offloading. The inclusion of large frameworks into the application environment of mobile apps can be time-consuming and increase complexity. To address these concerns, the solution proposed by this dissertation is developed to be portable so that it can be included into any mobile app to support the making of offloading decisions, without constraining the mobile device.

6.6. Conclusion

This chapter discussed general decision making, the factors that influence offloading decisions, and the factors that are measured by current research and how these factors are measured by the implementation in this dissertation.

Decision making is the process of choosing between options based on factors and the goal of the decision maker. The decision maker in the case of this dissertation is the decision-making component and the choice is between offloading and not offloading, to conserve battery life. When making the offloading decision, the cost of offloading is compared with the cost of local execution. The factors that influence the costs, namely the size of data,

bandwidth, communication protocol, code complexity and time to execute locally, are identified and discussed.

The cost of local execution is influenced by the code complexity and the time to execute. The code complexity cannot easily be used to determine the power consumption of a method, whereas the length of time, which is influenced by the code complexity, can directly be used in the calculations of power consumption. The cost of offloading is influenced by the size of the data, the available bandwidth and the communication protocol used. The size of the data and the available bandwidth can be used to determine the duration of the communication, and based on the connection protocol in use can determine the power consumption.

The cost of offloading is measured against the local processing cost to make the decision. The cost of offloading is calculated by multiplying the time of communication with the energy usage per second by the communication protocol. The cost of local processing is calculated by multiplying the time to execute with the energy usage per second of computation.

Previous chapters have shown that, besides display, communication and computation are the most expensive hardware components to use. The offloading decision is the choice between communicating the required data to and from the cloud and using the data locally to execute.

Related research has shown that offloading can be used to reduce the energy consumption of mobile devices. MAUI uses a profiler and solver to collect data and make informed offloading decisions. The profile continuously gathers data regarding the current state of the device, such as the available bandwidth. The information gathered by the profiler is used with the energy consumption profile created by using linear regression on measurements taken from an external power meter, to estimate the cost of local execution and the cost of offloading. The MAUI solver uses these costs to determine whether or not a method should be offloaded.

CloneCloud also uses a profiler to collect data regarding the state of the mobile device. The size of the state that is required to be transferred and the duration of local execution is measured by the profiler and used in conjunction with an energy consumption estimation function. The energy consumption estimation function is created by creating a model from data collected from an external power meter. The data gathered by the profiler is used by an optimizer to make the offloading decision.

The research identifies that an energy consumption profile characterizes the energy consumption when the mobile device is in use. The two approaches that can be used to create an energy consumption profile is hardware based and software based approaches. Hardware-based approaches result in profiles that are highly accurate. However, the creation of a hardware-based profile is labor-intensive and is not scalable. Software-based approaches result in less accurate profiles. However, software-based energy consumption profiles enable the monitoring of the application on many different granularities and it does not require expensive external power meters.

The evaluation of related approaches highlights the importance of energy consumption profiles. This chapter concludes the background information required throughout the dissertation. The next chapter introduces the Switch framework proposed by this dissertation that defines a portable software-based energy consumption profile.

Part 2:

Model & Prototype

Chapter 7: Switch: A framework for offloading decision making

7.1. Introduction

Part 2 of the dissertation now commences by introducing the Switch framework as a solution to the research problem posed by in this dissertation. Switch can assist developers to determine whether or not a task should be offloaded to the cloud in order to reduce energy consumption so that the battery life of a mobile device is conserved.

This research to this point identified that due to their mobility, mobile devices do not have the same resources as traditional computers. In this regard, battery life is seen as one of the most important resources of mobile devices that need to be conserved. Developers and end users cannot increase the battery life on mobile devices, however, they can influence how the battery life is used. In order to support developers with a software-based solution, this research identifies that a software framework could be used to reduce the battery life consumption of a mobile app. A set of four requirements were identified to be met by this research.

To be able to define a software framework, four techniques were identified that are used to augment mobile devices namely using the cloud, nearby computers, nearby mobile devices and a hybrid of these approaches. This research then identified that the distant fixed approach would be more suitable. To provide a solid foundation for this research, MAUI, and CloneCloud, examples of the distant fixed approach were discussed in more detail. The distant fixed approach offloads tasks to the cloud using either client-server communication or virtualization. After consideration of related work, the researcher chose the client-server communication approach.

Next, it was identified that offloading frameworks could either make use of virtual machine cloning or code offloading. This research focuses on code offloading where a function is executed on a cloud server instead of the local method, similar to what MAUI does. Important considerations when offloading

are low bandwidth, availability of network connections and different network types. To determine if a method should be executed on a cloud server, the cost of offloading is compared with the cost of local execution. This decision is influenced by the size of data, bandwidth, communication protocol, code complexity and time to execute locally. As such factors are difficult to constantly measure, the concept of a profiler was investigated. A device-specific energy consumption profile can characterize the energy consumption of a mobile device. As a hardware-based is labor-intensive and not scalable, this research decided to make use of a software-based approach that does not require expensive external power meters and enables the monitoring of the application on various levels of granularity.

The result of the literature review leads to the identification of a software framework that can support accurate battery consumption costs by making use of a device-specific energy consumption profile. Next, section 7.2 discusses and expands on the requirements of the Switch framework identified in chapter 3. In section 7.3 the energy consumption profile and the role it plays in the framework is discussed. Section 7.4 discusses the offloading decision process used by Switch. The architecture of the Switch framework and the interactions between the different components are discussed in section 7.5. Section 7.6 discusses the challenges of implementing the proposed. Finally, the chapter is concluded.

7.2. Requirements for conserving battery life when offloading

In order to achieve the aim of the research, the requirements identified in chapter 3 are now revisited and further discussed.

7.2.1. Intelligent offloading decision making

The making of offloading decisions is a relatively simple action. However, only by making precise energy consumption estimates, the goal of battery life conservation can successfully be achieved. To be able to support precise offloading decisions, all of the identified factors that influence the energy consumption of a mobile device should be taken into account. As every mobile device is uniquely defined with respect to hardware and software resources and

capabilities, these factors in their turn influence the energy consumption profile of any specific mobile device. This research proposes to make use of a mobile device specific energy consumption profile per network type to support intelligent decisions.

The energy consumption profile needs to address the following requirements:

- To enable developers to create and use an energy consumption profile on any device, a software-based energy consumption profiling method should be used.
- The factors that influence energy consumption should be continuously monitored to ensure the decisions made are correctly at any point in time.
- To be lightweight, the energy consumption profile should provide a simple model to estimate energy consumption.

7.2.2. Multiple network support

As mobile devices can make use of multiple types of networks, they all need to be supported when offloading decisions are made. Multiple network support can be achieved by defining an energy consumption profile for each network type.

7.2.3. Lightweight

When integrated with a mobile app, the Switch framework should consume as little battery life as possible and execute as efficiently as possible. Thus, the Switch framework should be lightweight with respect to communication overhead and CPU usage. Lightweight communication can be achieved by not requiring Switch to communicate with external sources whereas lightweight CPU usage can be achieved by simplifying the estimation of energy consumption.

7.2.4. Portable

The Switch framework should be able to integrate into any mobile app that needs to support offloading. This integration will enable the creation of an energy consumption profile of a specific device.

An important factor to be used by this research is that of the energy consumption profile, defined next.

7.3. Switch energy consumption profile

For this purposes of this research, an energy consumption profile is defined that can support accurate estimations of battery life usage per mobile device.

An energy consumption profile is created from measurements taken during the execution of different tasks on a specific mobile device, such as executing a computationally intensive process for a set amount of time or downloading a file over a specific network. The collected data is simplified into four linear models that represent battery life usage. An energy consumption profile per network can enable an accurate estimation of battery life usage when communicating over a specific network.

To achieve the desired accuracy, the energy consumption profile used by Switch is described by four linear models as follows:

The energy consumption when the CPU of the mobile device is in use is displayed in equation 7.1.

Equation 7.1 CPU energy consumption equation

$$EC_{CPU} = localRate(time) + localConstant$$

Where:

- $localRate$ and $localConstant$ are device specific constants
- $time$ is the length of time the CPU will be active
-

The energy consumption when communicating with the 3G cellular network is displayed in equation 7.2.

Equation 7.2 3G communication energy consumption equation

$$EC_{3G} = threeGRate(time) + threeGConstant$$

Where:

- $threeGRate$ and $threeGConstant$ are device specific constants

- $time$ is the length of time the 3G network radio will be active

The energy consumption when communicating with the 4G cellular network is displayed in equation 7.3.

Equation 7.3 4G communication energy consumption equation

$$EC_{4G} = fourGRate(time) + fourGConstant$$

Where:

- $fourGRate$ and $fourGConstant$ are device specific constants
- $time$ is the length of time the 4G network radio will be active

The energy consumption when communicating with Wi-Fi is displayed in equation 7.4

Equation 7.4 Wi-Fi communication energy consumption equation

$$EC_{Wi-Fi} = wifiRate(time) + wifiConstant$$

Where:

- $wifiRate$ and $wifiConstant$ are device specific constants
- $time$ is the length of time the Wi-Fi radio will be active

The models defined above are used to estimate the battery life usage on a method level of granularity, however, the models are created by monitoring the relevant hardware components of the mobile device. Models created by measuring the energy consumption of hardware components are more flexible as they are not bound to a specific app. An application that monitors the operating system is used to sample the energy consumption of the specific devices.

Currently, the energy consumption profile is created manually, however with further investigation and experimentation the process can be automated. Due to the energy consumption required to analyse the automatically gathered data the

task of generating the consumption model should be done remotely on the cloud.

The creation of the energy consumption profile used in this dissertation is discussed in the next chapter.

7.4. Switch offloading decision

The Switch offloading decision-making process is presented in this section. The decision, and the methods used to estimate the costs are discussed.

The offloading decision is simply the comparison of the estimated energy consumption when executing locally and the estimated energy consumption when offloading. The offloading decision made by Switch is determined as shown in equation 7.5

Equation 7.5 Switch offloading decision

$$shouldOffload = E_{local}(time) > E_{offloading}(bandwidth, size)$$

Where:

- $shouldOffload$ is a true or false value that determines whether or not offloading should occur
- E_{local} represent the calculation of the cost of local execution
- $E_{offloading}$ represents the calculation of the cost of offloading

The cost of local execution is determined as shown in equation 7.6.

Equation 7.6 Local cost estimation

$$localCost = localRate(time) + localConstant$$

Where:

- $localRate$ and $localConstant$ are the constants provided by the CPU energy consumption model
- $time$ is the duration of local execution

The cost of offloading is determined as shown in equation 7.7.

Equation 7.7 Offloading cost estimation

$$offloadingCost = networkRate\left(\frac{size}{bandwidth}\right) + networkConstant$$

Where:

- $networkRate$ and $networkConstant$ are constants provided by the energy consumption models. The network model used is determined by the network the device is currently connected to.
- $size$ represents the size of the data that is transferred from and to the cloud
- $bandwidth$ is the current available bandwidth.

It is important to note that this research follows an approach to offload code to the cloud if energy can be saved, even if it may not result in a major saving. Another approach would be to enable the use of offloading if certain battery thresholds per device are passed. The offloading decision-making process is not overly complex but is essential in achieving the goal of this research. Given the understanding of the decision-making process and the energy consumption profile and the models, the architecture of the Switch framework is now discussed.

7.5. Switch architecture

Switch is an Android-based framework that can be integrated into any Android mobile application. The contribution of Switch is to support developers, to enable them to to make offloading decisions without requiring the integration of a large framework in both the app and the cloud. In order to achieve this, the energy consumption of the mobile device needs to be sourced, the offloadable tasks should be identified, and the duration of local execution and the size of the data communicated when offloading should be measured.

The architecture of Switch is shown in Figure 7.1 consisting of a number of components. A mobile app is loaded onto the Android OS. The Switch component is called from within the mobile app and consists of the Decision-Making component, the Profiler, and the Energy Consumption Profile component.

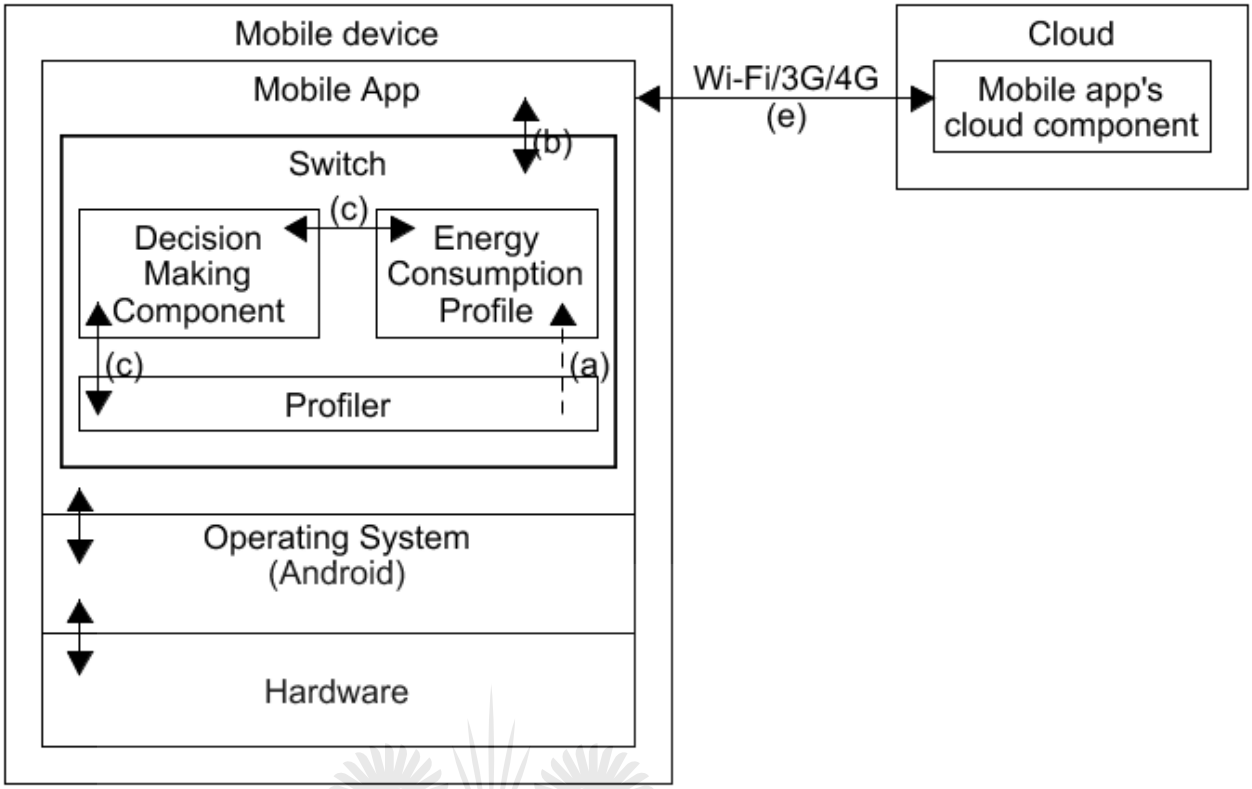

Figure 7.1 Architecture of Switch

First, before offloading decisions can be made, the energy consumption profile model needs to be determined for the specific mobile device in an offline manner. A number of tests are run for each network type the device is connected to. Similar sets of tests are executed for local execution. Historical averages of the local and cloud execution costs are stored. The results are examined and the model is created. If offloading decisions become unreliable after some time passes, this phase can be executed again. The creation of the energy consumption profile is shown by the communication between the Profiler and the Energy Consumption Profile in figure 7.1 (a).

To make use of Switch, a mobile app developer defines a mobile app with any number of methods. Offloadable methods are identified and their server-side counterparts are created in the cloud. Traditionally, the developer would decide whether or not the method is to be offloaded. However, by integrating Switch, such decisions are made on a per execution basis. The communication between the mobile app and Switch is shown in figure 7.1 (b).

Within Switch, the Decision-Making component makes the offloading decision by utilising the Profiler component and the Energy Consumption Profile, shown in figure 7.1 (c) and (d). The Profiler component continuously monitors the state of the mobile device and the app. The Energy Consumption Profile component contain models that are used to estimate the energy consumption when provided with a set of variables. The estimated costs determine the outcome of the decision made by the decision-making component. If the decision is made to offload the app communicates with the cloud, shown in figure 7.1 (e).

7.5.1. Switch operation

There are two phases to be considered namely the *initial* and *operational* phase. In the initial phase discussed above, the energy consumption profile is created. This step must be performed before Switch can be used to make offloading decisions.

Figure 7.2 represents the *operational* phase of Switch where communication between the mobile app and Switch takes place. The sequence of method calls during app startup and when executing an offloadable method is given.

The initial network check is shown in Figure 7.2 (a). The network check determines the current network the mobile device is connected to and calculates the available bandwidth by timing the download and upload of a file. The network check is performed when the *checkNetwork* method is called by the mobile app.

In figure 7.2 (b) the mobile app calls methods in Switch to determine whether or not the task should be offloaded. Once the *shouldOffload* method has been called, the decision-making component retrieves the network state from the profiler by calling the *getNetworkState* method. The decision-making component then sends the stored averages of the method to the energy consumption profile to calculate estimated energy consumption, calling the *estimateOffloadingCost* and *estimateLocalCost* methods. The estimates provided are used to make the offloading decision, the result of the decision is returned to the mobile app.

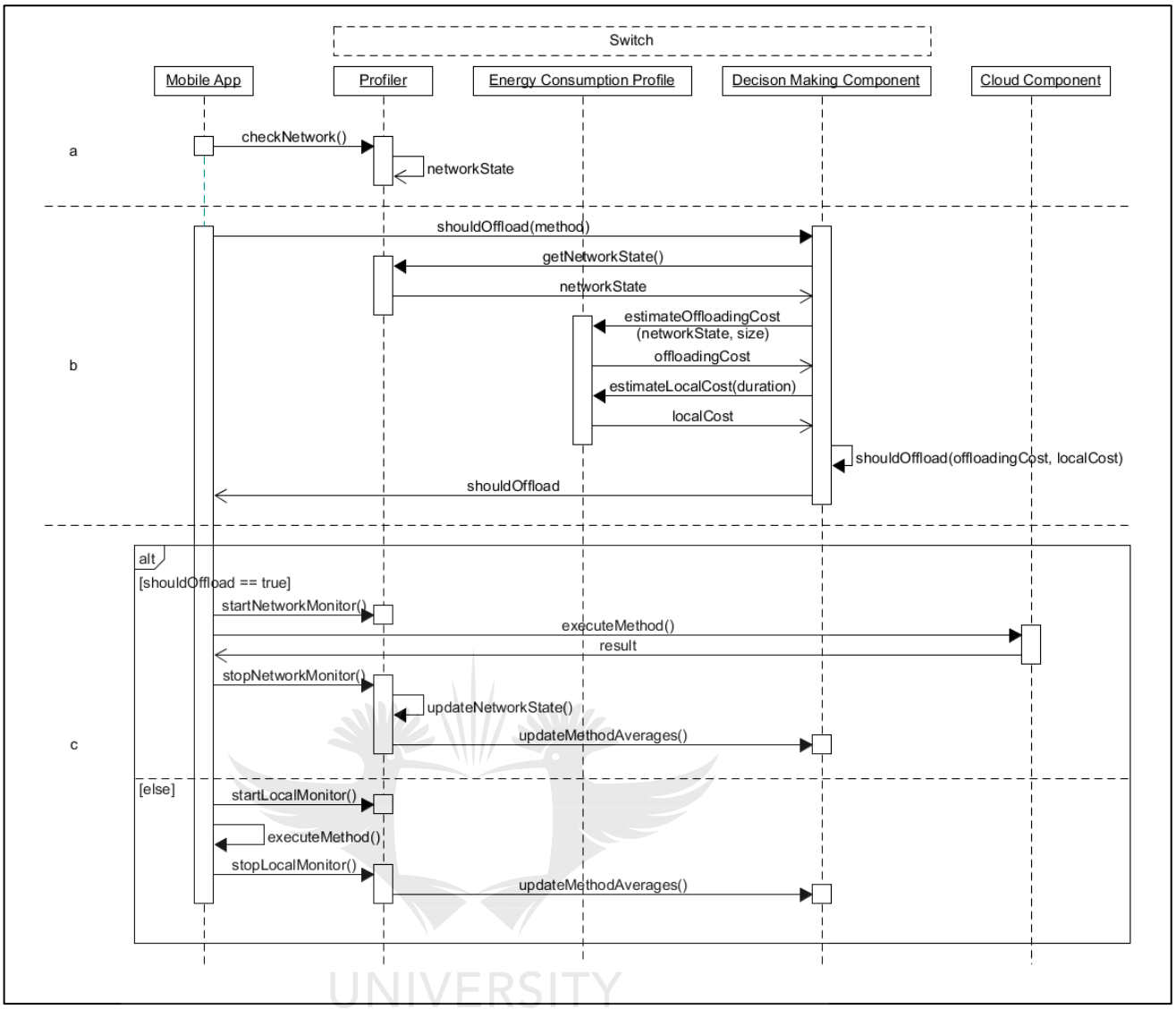

Figure 7.2 Interaction between Switch and a mobile app

The sequence of method calls diverges in figure 7.2 (c) based on the result of the *shouldOffload* method. If the method should be offloaded the mobile app communicates with its cloud component, and if the method should not be offloaded the local method is executed. The *startNetworkMonitoring*, *stopNetworkMonitoring*, *startLocalMonitoring*, and the *stopLocalMonitoring* methods are exposed by Switch, to update the historical averages of a method.

Figure 7.2 is referred to in the next discussion of the Switch decision-making component, Switch profiler, energy consumption profile, mobile app, and the cloud component.

7.5.2. Switch profiler

The profiler gathers data regarding the state of the mobile device and monitors the mobile app during runtime. Information is gathered both from the operating system and by monitoring the mobile app. During app startup as shown in figure 7.2 (a), a call is made to the Switch profiler to collect data regarding the current network state. When an offloadable method is executed, as shown in figure 7.2 (c), the profiler starts monitoring the resource usage of the mobile device. After the execution completes the monitoring is stopped. The data is used to update network state and the average size of communication or duration of local execution.

7.5.3. Switch decision-making component

When an offloadable method is executed as shown in figure 7.2 (b), the decision-making component is called to determine whether the method should be offloaded. The offloading decision-making process is initiated, the necessary data is collected and used to estimate the offloading and local execution costs, the estimated costs are used to make the offloading decision. Each execution of an offloadable method is used to update the averages of that method, as shown in figure 7.2 (c).

7.5.4. Energy consumption profile

The energy consumption profile contains the energy consumption models that are used to estimate battery life consumption. The estimates are calculated with parameters provided by the decision making the profile as shown in figure 7.2 (b).

7.5.5. Mobile application and a cloud component

The mobile app and the cloud component makes use of Switch. However, Switch does not influence the development of these components or the communication between them. The mobile application can, for example, use a local cloud or use cyber-foraging while still using Switch. However, Switch is designed to only make offloading decisions when offloading occurs over Wi-Fi or cellular networks.

The architecture discussed ensures that Switch is lightweight, and also ensures that the communication between the mobile application and the cloud component of the application use Wi-Fi, 3G or 4G, the networks that the solution are required to support.

7.6. Conclusion

This chapter discusses the requirements of Switch, the architecture of Switch and the challenges of implementing it. The basic requirements of Switch are intelligent offloading decisions, support for both Wi-Fi and cellular networks, and being a lightweight solution that is portable. Intelligent offloading decisions are made by taking all factors into account and using an accurate energy consumption profile. For Switch to support multiple networks the energy consumption profile should cater for the different networks. Switch should be lightweight as to not increase the drain on battery life and to not interfere with the user experience. The portability of Switch allows the integration of the framework into any mobile app on any device.

Switch is used to make the offloading decision for mobile apps that use offloading. The Switch framework is integrated into existing applications without interfering with their execution or communication. The profiler gathers data regarding the mobile device and makes the offloading decision.

This chapter provides an understanding of the components of Switch, how they communicate, and interact with a mobile app. The next chapter presents the Switch energy consumption profile.

Chapter 8: Energy consumption profile of mobile devices

8.1. Introduction

This chapter demonstrates the initial phase of the Switch framework, namely the creation of an energy consumption profile for a specific mobile device. The Samsung Galaxy S7 Edge is chosen as the device to test the Switch framework prototype.

To be able to create an energy consumption profile, a number of experiments are performed to collect energy consumption data for different network usage and CPU intensive tasks. The manner in which data is collected and the analysis of the collected data is discussed in this chapter.

Section 8.2 discusses models used by the energy consumption profile. The environment of the evaluation is discussed in section 8.3. The experiments used to sample energy consumption are discussed in section 8.4. Section 8.5 discusses the results of the experiments, and finally, the chapter is concluded.

8.2. Energy consumption profile models

An energy consumption profile describes the energy consumption of a mobile device that is formalised using a linear model. The energy consumption profile created in this research consists of four distinct linear models, where each defines a different type of execution.

The first model describes energy consumption when performing tasks locally and is created by sampling the energy consumption of the CPU of the mobile device. The second, third and fourth models describe the energy consumption when communicating via a specific network. For each of the network models, the model is created by sampling the energy consumption of the network radio of the mobile device when it is connected to a specific network. The four models are used in conjunction with historical execution data of methods and the current state of the device to estimate energy consumption.

The data needed to create a model requires a means to measure the energy that is consumed by a task. The energy consumption sampling process used in this dissertation is discussed next.

8.3. The conditions of the evaluation

This research determines energy consumption by using third-party software that can be installed on the mobile device. The environment and the set of tests to be executed in the environment are discussed in this section.

8.3.1. Environment

The study considers the mobile device being tested, its network connections and the software used to take measurements on the device.

a. *Device under test*

The device under test is the Samsung Galaxy S7 Edge, SM-G900F, mobile device. The device runs Android 6.0.1. The hardware of the device is shown in Table 8.1 (Samsung, 2014b).

Table 8.1 Samsung Galaxy S7 Edge Hardware specifications

Feature	Specifications
Operating System	Android 6 (Marshmallow)
CPU	Qualcomm MSM8996 Snapdragon 820 - Quad-core (2x2.15 GHz Kryo & 2x1.6 GHz Kryo)
RAM	4 GB
Storage	32GB + microSD
Display Size	5.5”
Network	2G, 3G, 4G(LTE) Wi-Fi Bluetooth NFC
Battery	3600 mAh

The Samsung Galaxy S7 Edge is chosen because its hardware is representative of most modern mobile devices, considering the number of networks that can be connected to and the power of the processor. The Samsung Galaxy S7 Edge can connect to 4G networks and has a powerful processor. Power consumption is determined by the length of computation time and communication. The assumption is that the stronger the processor or higher the bandwidth of the network that the device connects to, the shorter the length of time required for communication or computation.

b. *Third-party measuring software*

Software is used to measure the power usage, length of time of computation and communication, the network signal strength and size of communication data, both for upload and download conditions. Measurements are made using software in order to create a power consumption profile on different devices, without dismantling the device and attaching measurement hardware.

The measurements of signal strength and battery life are made using two apps. The Network Signal Info Pro app, developed by KAIBITS Software, (KAIBITS, 2015), is used to get accurate readings of the networks the device is connected to. The GSam Battery Monitor app, developed by GSam Labs, (GSam Labs, 2013) is used to monitor the battery life usage and the length of computation time. These third-party apps have been chosen out of a large number of contenders, due to their accuracy and usefulness, after a careful review by the researcher.

i. Network Signal Info Pro

Network Signal Info Pro is an app used to evaluate energy consumption that measures the current signal strength of the device. The app is installed on the device and measures the strength of the connection of the current network the device is connected to. When in use, the app automatically connects to different signal strengths, as the device is moved to different locations with stronger or weaker signal strength. The signal strength is given as a percentage and in decibel milliwatts (dBm).

ii. GSam Battery Monitor

GSam Battery Monitor is an app used to evaluate energy consumption that measures how much battery life or energy is used. The app gives the percentage of battery life consumed by each app on the device, between two points in time. Whenever the energy expenditure of the device drops a complete percentage point, the first point in time is selected. The experiment is continued and after another percentage point drop is recorded, the second point in time is selected. Thus, the app gives the percentage of battery life consumed during the experiment.

8.3.2. Experiments

The power consumption of the device being tested is evaluated by executing several tasks and measuring the power consumption and the length of time required for either communication or computation. Computation and communication are tested separately, both tests are discussed in this section.

a. *Communication*

The power consumption of the device when it is communicating is tested on different networks that can be used to access the Internet and the cloud. For each network, files of differing sizes are downloaded and uploaded at various signal strengths. The signal strength, network type, length of communication, size of communication and power consumption is recorded.

- Signal strength: How strong is the connection to the network? Signal strength is measured by Network Signal Info Pro and the results are recorded in dBm.
- Network type: Which network is the device connected to? Network type can either be HPSA, 4G or Wi-Fi.
- Length of communication: How long did it take to execute the experiment? The length of time of communication is measured in seconds.
- Size of communication: How much data is uploaded or downloaded during the experiment? The size of the communication is measured in bytes. The differences between the file size and the size of communication are caused by packet loss during the experiment.
- Power consumption: How much power did the experiment take? GSam Battery Monitor is used to measure how much battery life is consumed. The result is given as a percentage of total device battery life consumed.

The files are downloaded and uploaded to the SAP Hana Cloud Platform (SAP, 2017a, 2017b), using a trial account. The mobile connections, 3G and 4G are provided by MTN (MTN, 2017), using Afrihost (Afrihost, 2017a, 2017b) data.

The Wi-Fi connections are made to an ADSL line over the Telkom (Telkom, 2017) infrastructure connected to an Afrihost 2MB line.

b. *Computation*

The power consumption of the device when it is performing computational tasks is measured after specifying an amount of time and allowing the application to calculate prime numbers for the specified time duration. Calculating primes is selected as the computational test because it is a very complex task that uses a large amount CPU power. The amount of time selected corresponds with the time the CPU is used, and this is recorded as well as the percentage of battery life consumed.

- Length of computation: How long did it take to execute the experiment? The length of time of computation is measured in seconds.
- Power consumption: How much power did the experiment take? GSam Battery Monitor is used to measure how much battery life is consumed and the result is given as a percentage of total device battery life consumed.

8.4. Results

The results of the study are divided into two sections. Firstly, communication is discussed by considering the results recorded when downloading and uploading files at different signal strengths and on different networks. Secondly, computation is discussed by reviewing the results when tasks are executed on the device.

8.4.1. Communication

The results of the experiments when measuring the power consumption of the device when it is communicating are discussed in this section. The results are divided into sections according to the different types of networks, namely, 3G, 4G, and Wi-Fi. Each subsection is divided into two parts, namely the upload and download of data. The results for both downloaded and uploaded data is further separated into sections corresponding to the size of data used in the experiments. Under each section, the percentage battery life consumed is

compared to the length of time the communication lasted and signal strength is compared to the length of time of the communication.

The subsections are structured as follows:

- Network type (3G, 4G, Wi-Fi)
    - Download
        - 100 KB
        - 1 MB
        - 10 MB
    - Upload
        - 100 KB
        - 1 MB
        - 10 MB
    - Evaluation

The results recorded reflect the average of executing the same the test five times. The complete data set is attached in Appendix A.

a. *3G*

The results of downloading and uploading files over the 3G network are discussed in this section.

i. Download

To measure the energy consumption of downloading data from the cloud, files of size 100 KB, 1 MB, and 10 MB are downloaded and the results are measured. In Table A.2, Table A.3 and Table A.4 in Appendix A show the averages of downloading a 100 KB, 1 MB and 10 MB file five times at each of the listed signal strengths. These results are summarized in the following figures.

100 KB

Figure 8.1 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.028% to 0.046% as the length of time increases from 1.2 seconds to 4.5 seconds. Very little battery life

is consumed because of the of the small file size. Figure 8.2 shows the length of time of the communication against signal strength, as the signal strength increase from -109 dBm to -83 dBm the length of time of communication decreases from 4.5 seconds to 1.2 seconds.

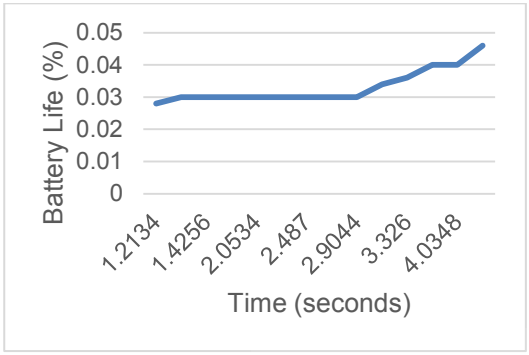

Figure 8.1 100 KB file downloaded over 3G

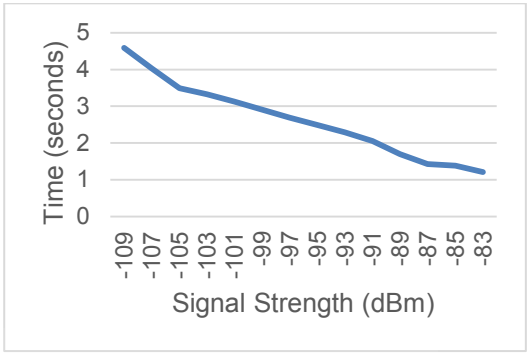

Figure 8.2 100 KB file downloaded over 3G

1 MB

Figure 8.3 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.04% to 0.074% as the length of time of communication increases from 4.3 seconds to 7.6 seconds. Figure 8.4 shows the length of time of the communication against signal strength, as the signal strength increases from -109 dBm to -83 dBm the length of time of communication decreases from 8.6 seconds to 4.3 seconds.

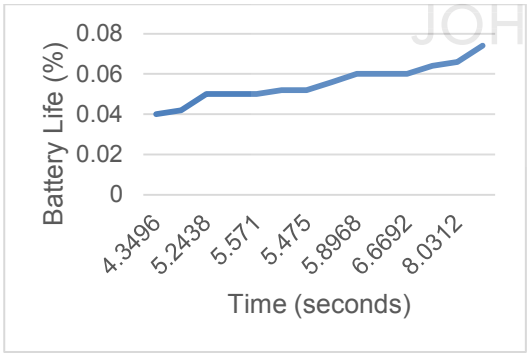

Figure 8.3 1 MB file downloaded over 3G

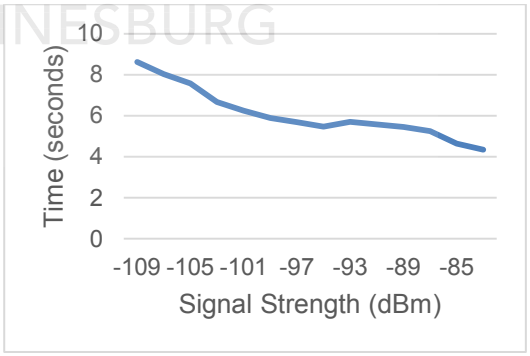

Figure 8.4 1 MB file downloaded over 3G

10 MB

Figure 8.5 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.242% to 0.344% as the length of time of communication increases from 44.4 seconds to 86.4 seconds.

Figure 8.6 shows the length of time of the communication against signal strength.

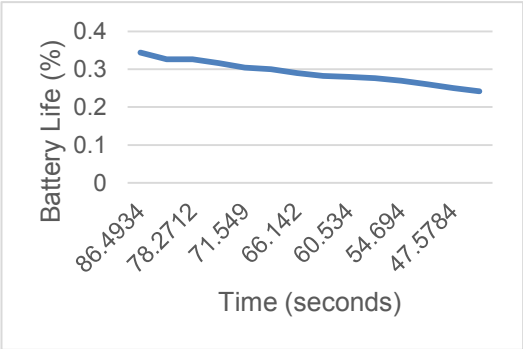

Figure 8.5 10 MB file downloaded over 3G

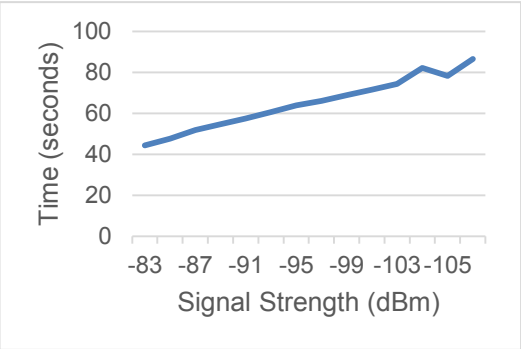

Figure 8.6 10 MB file downloaded over 3G

ii. Upload

To measure the energy consumption of uploading data to the cloud, files of size 100 KB, 1 MB, and 10 MB are uploaded and the results are measured. The results of the uploading of the files are displayed in Table A.5, Table A.6 and Table A.7 in Appendix A.

100 KB

Figure 8.7 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.02% to 0.05% while the length of time of communication increases from 2.1 seconds to 5.9 seconds. Figure 8.8 shows the length of time of the communication against signal strength, the length of time of communication decreases from 5.9 seconds to 2.1 seconds while the signal strength increases from -109 dBm to -83 dBm.

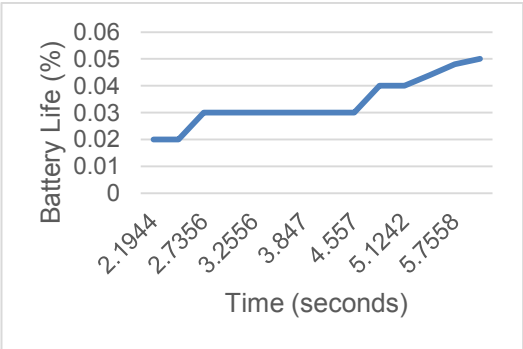

Figure 8.7 100 KB file uploaded over 3G

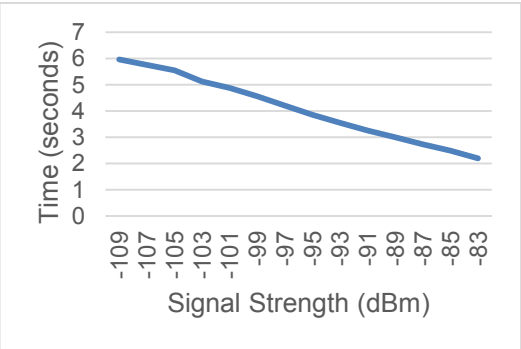

Figure 8.8 100 KB file uploaded over 3G

1 MB

Figure 8.9 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.06% to 0.09% while the length of time of communication increases from 6.8 seconds to 13.2 seconds. Figure 8.10 shows the length of time of the communication against signal strength, the length of time of communication decreases from 6.8 seconds to 13.2 seconds while the signal strength increases from -109 dBm to -83 dBm.

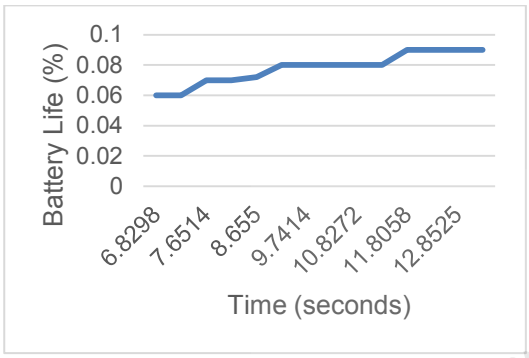

Figure 8.9 1 MB file uploaded over 3G

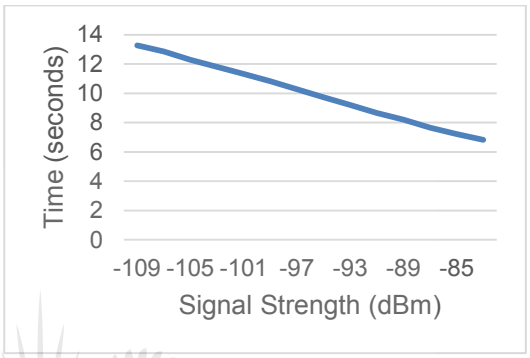

Figure 8.10 1 MB file uploaded over 3G

10 MB

Figure 8.11 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.242% to 0.342% while the length of time of communication increases from 43.9 seconds to 87.3 seconds. Figure 8.12 shows the length of time of the communication against signal strength, the length of time of communication decreases from 43.9 seconds to 87.3 seconds while the signal strength increases from -109 dBm to -83 dBm.

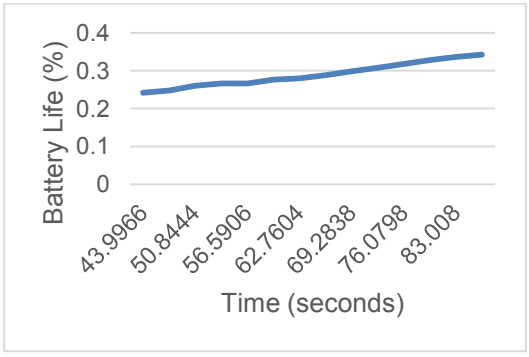

Figure 8.11 1 MB file uploaded over 3G

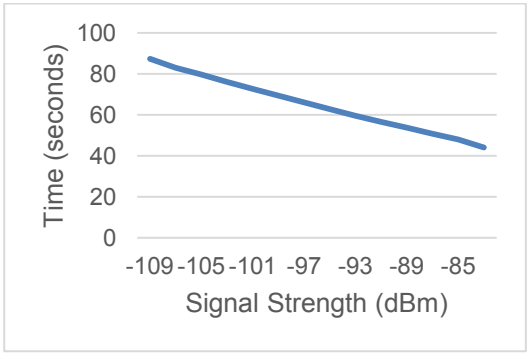

Figure 8.12 10 MB file uploaded over 3G

iii. Evaluation

The results given in the above sections show that there is a direct relationship between the length of time of communication and the battery life consumed, and there exists an indirect relationship between the signal strength and the length of time the communication last.

Comparing download and upload for the different file sizes shows that they follow the same pattern, however, uploading is a slightly more expensive operation, in terms of battery life consumed.

b. *4G*

The results of downloading and uploading files over 4G are discussed in this section.

i. Download

To measure the energy consumption of downloading data from the cloud, files of size 100 KB, 1 MB, and 10 MB are downloaded and the results are measured. In Table A.8, Table A.9 and Table A.10 in Appendix A show the averages of downloading a 100 KB, 1 MB and 10 MB file five times at each of the listed signal strengths. These results are summarized in the following figures.

100 KB

Figure 8.13 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.02% to 0.03% while the length of time of communication increases from 0.4 seconds to 0.7 seconds. Figure 8.14 shows the length of time of the communication against signal strength, the length of time of communication decreases from 0.7 seconds to 0.4 seconds while the signal strength increases from -117 dBm to -87 dBm.

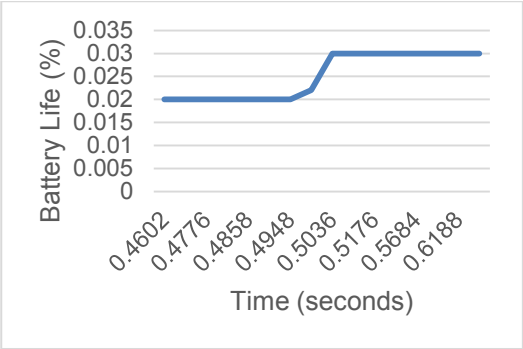

Figure 8.13 100 KB file downloaded over 4G

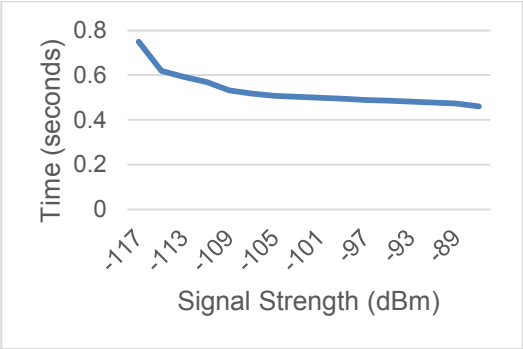

Figure 8.14 100 KB file downloaded over 4G

1 MB

Figure 8.15 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.04% to 0.066% while the length of time of communication increases from 3.9 seconds to 6.6 seconds. Figure 8.16 shows the length of time of the communication against signal strength, the length of time of communication decreases from 6.6 seconds to 3.9 seconds while the signal strength increases from -117 dBm to -87 dBm.

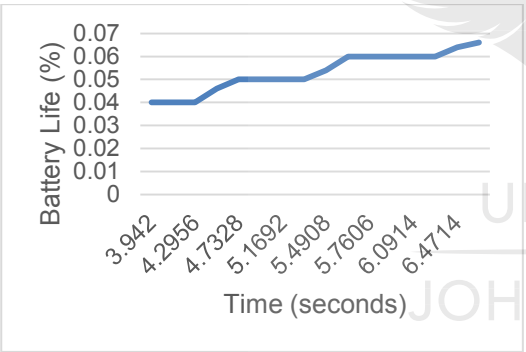

Figure 8.15 1 MB file downloaded over 4G

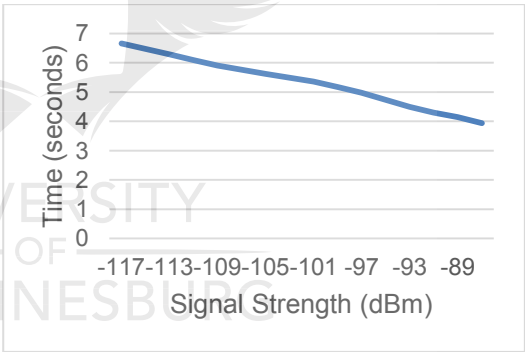

Figure 8.16 1 MB file downloaded over 4G

10 MB

Figure 8.17 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.248% to 0.34% while the length of time of communication increases from 37.4 seconds to 54.6 seconds. Figure 8.18 shows the length of time of the communication against signal strength, the length of time of communication decreases from 54.6 seconds to 37.4 seconds while the signal strength increases from -117 dBm to -87 dBm.

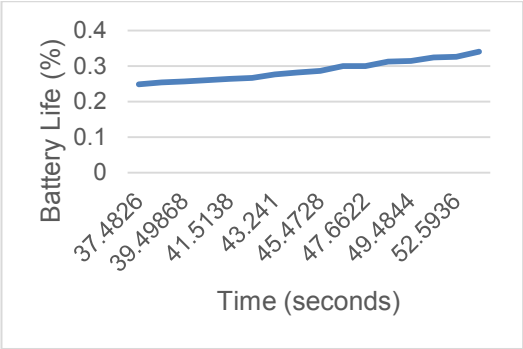

Figure 8.17 10 MB file downloaded over 4G

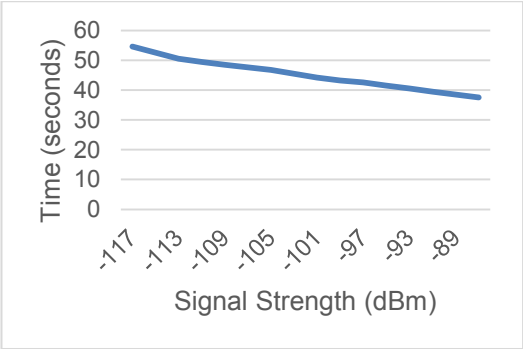

Figure 8.18 10 MB file downloaded over 4G

ii. Upload

To measure the energy consumption of uploading data to the cloud, files of size 100 KB, 1 MB, and 10 MB are uploaded and the results are measured. The results of the uploading of the files are displayed in Table A.11, Table A.12 and Table A.13 in Appendix A.

100 KB

Figure 8.19 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.03% to 0.04% while the length of time of communication increases from 2.4 seconds to 4.3 seconds. Figure 8.20 shows the length of time of the communication against signal strength, the length of time of communication decreases from 4.3 seconds to 2.4 seconds while the signal strength increases from -117 dBm to -87 dBm.

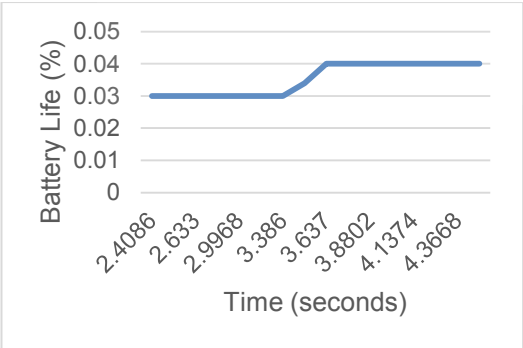

Figure 8.19 100 KB file uploaded over 4G

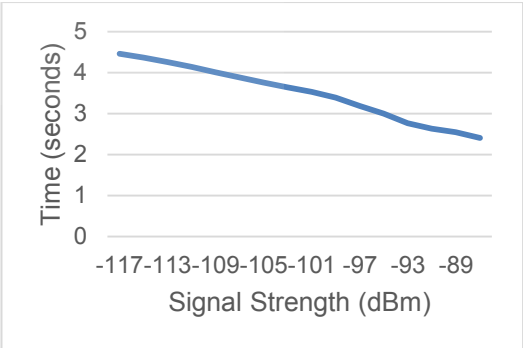

Figure 8.20 100 KB file uploaded over 4G

1 MB

Figure 8.21 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.06% to 0.08% while the length of time of communication increases from 5.7 seconds to 8.8 seconds.

Figure 8.22 shows the length of time of the communication against signal strength, the length of time of communication decreases from 8.8 seconds to 5.7 seconds while the signal strength increases from -117 dBm to -87 dBm.

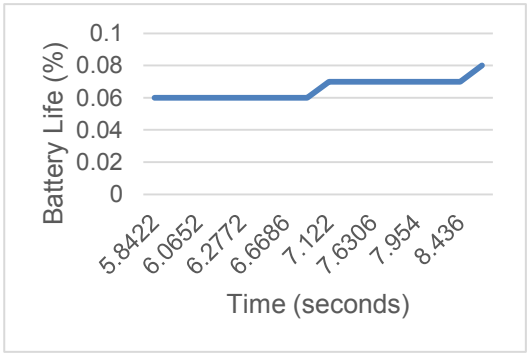

Figure 8.21 1 MB file uploaded over 4G

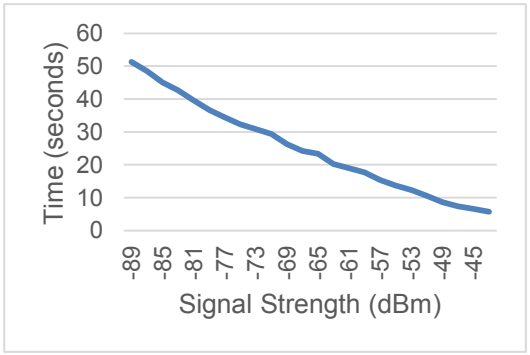

Figure 8.22 1 MB file uploaded over 4G

10 MB

Figure 8.23 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.256% to 0.35% while the length of time of communication increases from 39.5 seconds to 59 seconds. Figure 8.24 shows the length of time of the communication against signal strength, the length of time of communication decreases from 59 seconds to 39.5 seconds while the signal strength increases from -117 dBm to -87 dBm.

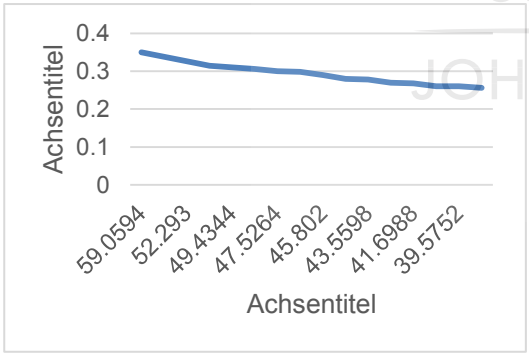

Figure 8.23 10 MB file uploaded over 4G

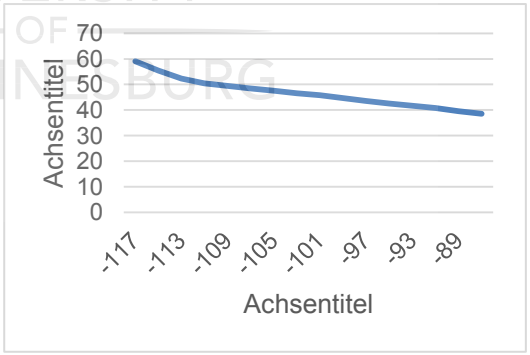

Figure 8.24 10 MB file uploaded over 4G

iii. Evaluation

The results given in the above sections show that there is a direct relationship between the length of time of communication and the battery life consumed, and there exists an indirect relationship between the signal strength and the length of time the communication last.

Comparing download and upload for the different file sizes shows that they follow the same pattern, however, uploading is a slightly more expensive operation, in terms of battery life consumed.

c. *Wi-Fi*

The results of downloading and uploading files over Wi-Fi are discussed in this section.

i. Download

To measure the energy consumption of downloading data from the cloud, files of size 100 KB, 1 MB, and 10 MB are downloaded and the results are measured. In Table A.14, Table A.15 and Table A.16 in Appendix A show the averages of downloading a 100 KB, 1 MB and 10 MB file five times at each of the listed signal strengths. These results are summarized in the following figures.

100 KB

Figure 8.25 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.01% to 0.05% while the length of time of communication increases from 1.2 seconds to 9.8 seconds. Figure 8.26 shows the length of time of the communication against signal strength, the length of time of communication decreases from 9.8 seconds to 1.2 seconds while the signal strength increases from -89 dBm to -45 dBm.

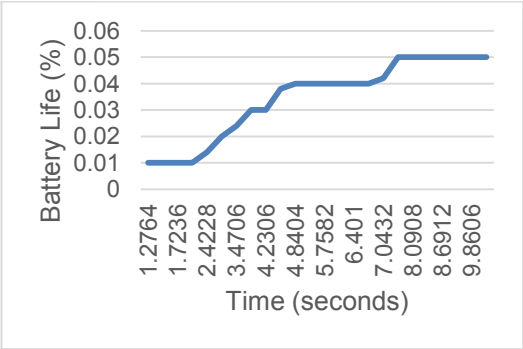

Figure 8.25 100 KB file downloaded over Wi-Fi

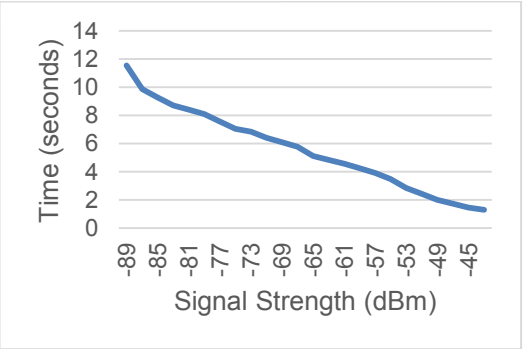

Figure 8.26 100 KB file downloaded over Wi-Fi

1 MB

Figure 8.27 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.04% to 0.132% while the length of time of communication increases from 5.7 seconds to 48.4 seconds. Figure 8.28 shows the length of time of the communication against signal strength, the length of time of communication decreases from 48.4 seconds to 5.7 seconds while the signal strength increases from -89 dBm to -45 dBm.

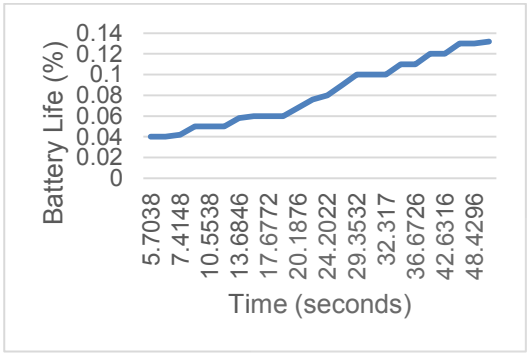

Figure 8.27 1 MB file downloaded over Wi-Fi

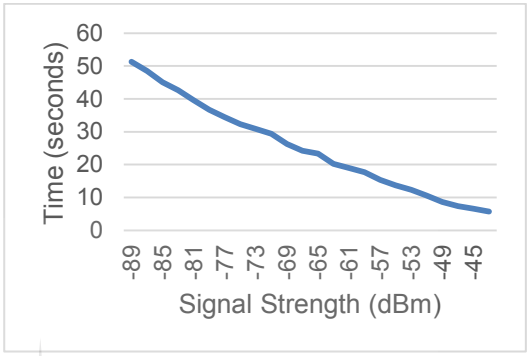

Figure 8.28 1 MB file downloaded over Wi-Fi

10 MB

Figure 8.29 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.136% to 0.23% while the length of time of communication increases from 51.8 seconds to 82.6 seconds. Figure 8.30 shows the length of time of the communication against signal strength, the length of time of communication decreases from 82.6 seconds to 51.8 seconds while the signal strength increases from -89 dBm to -45 dBm.

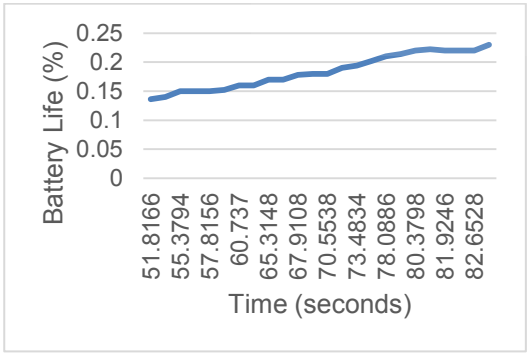

Figure 8.29 10 MB file downloaded over Wi-Fi

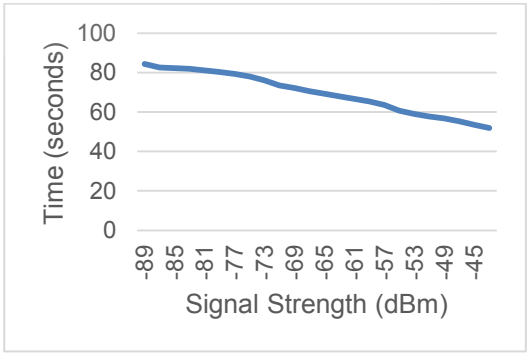

Figure 8.30 10 MB file downloaded over Wi-Fi

ii. Upload

To measure the energy consumption of uploading data to the cloud, files of size 100 KB, 1 MB, and 10 MB are uploaded and the results are measured. The results of the uploading of the files are displayed in Table A.17, Table A.18 and Table A.19 in Appendix A.

100 KB

Figure 8.31 shows the length of time of the communication over the battery life consumed, the battery life consumed increases from 0.01% to 0.02% while the length of time of communication increases from 5.4 seconds to 9.5 seconds. Figure 8.32 shows the length of time of the communication against signal strength, the length of time of communication decreases from 9.5 seconds to 5.4 seconds while the signal strength increases from -89 dBm to -45 dBm.

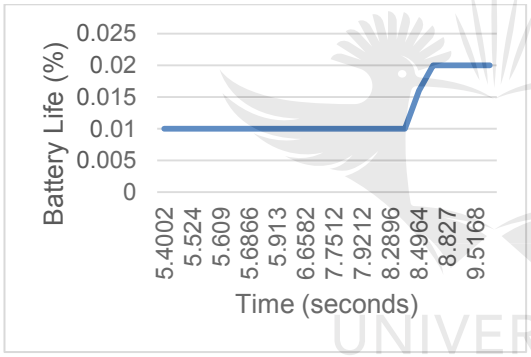

Figure 8.31 100 KB file uploaded over Wi-Fi

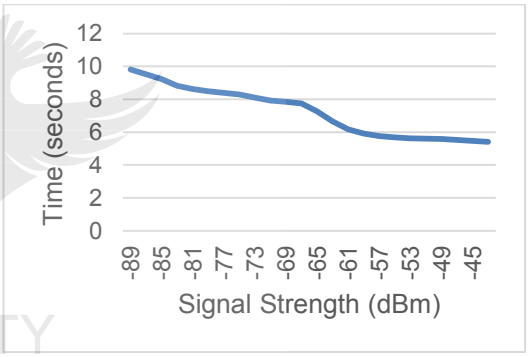

Figure 8.32 100 KB file uploaded over Wi-Fi

The experiments are done using natural network connectivity. Uploading 1 MB and 10 MB files over the Wi-Fi network does not complete.

iii. Evaluation

The results given in the above sections show that there is a direct relationship between the length of time of communication and the battery life consumed, and there exists an indirect relationship between the signal strength and the length of time the communication last.

Comparing download and upload for the different file sizes shows that they follow the same pattern, however, uploading is a slightly more expensive operation, in terms of battery life consumed.

d. *Comparison*

The results from the experiments show that the longer the communication lasts, the more battery life is consumed. For each of the networks the stronger the signal strength the less time the communication lasts.

The different networks all conform to the same pattern, however, the battery consumed by the different networks differs when executing the same task. Figure 8.33 shows the energy consumption for the different networks when downloading a 1 MB file.

In Figure 8.33 all the networks have a similar best point in terms of percentage battery life consumed and length of time of communication. Wi-Fi increases at a slower pace than the mobile networks, however, it does use the most battery life and take the longest time at the worst signal strength. The mobile networks increase at almost the pace, however, 4G does not have the same peak as 3G.

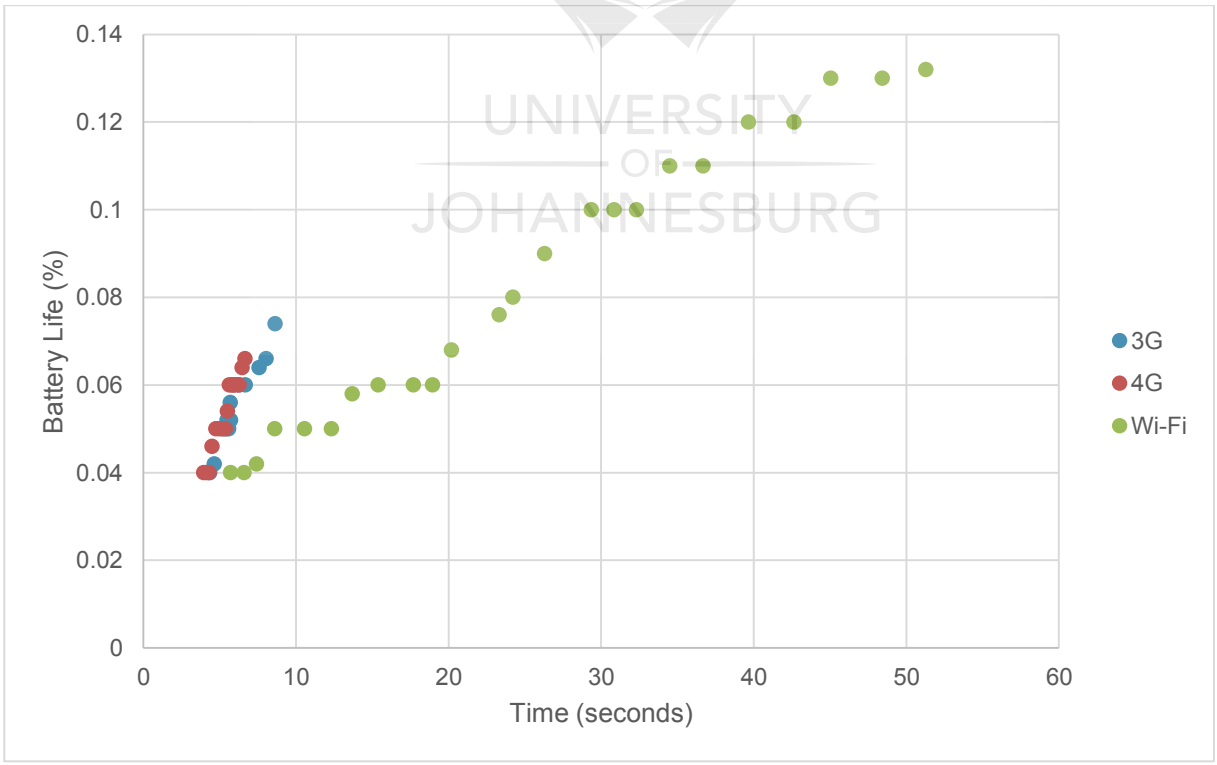

Figure 8.33 Downloading 1 MB over 3G, 4G and Wi-Fi

The comparison shows that in most cases with strong signal strength, Wi-Fi is the less expensive network to use in terms of energy consumption. The cellular networks, however, consume battery life at almost the same rate. The available bandwidth on the networks determine the length of time of the communication because Wi-Fi is generally more stable and has a higher bandwidth it will consume less battery life than the cellular networks, in the same manner, 4G that has a higher available bandwidth than 3G will consume less power because the communication will not need to last as long.

The results of the communication results show that with the network the mobile device is connected to, the signal strength of the network and the size of the communication an estimated energy consumption can be calculated.

8.4.2. Computation

The results from the experiments when measuring the power consumption of the device when it is performing computations is discussed in this section. The percentage of battery life consumed is compared to the length of time the computation lasted. The results shown are the averages for executing the experiment 3 times.

Table A.17 shows the results of the computational experiments. Figure 8.34 shows the percentage battery life consumed over the length of time the computation lasted.

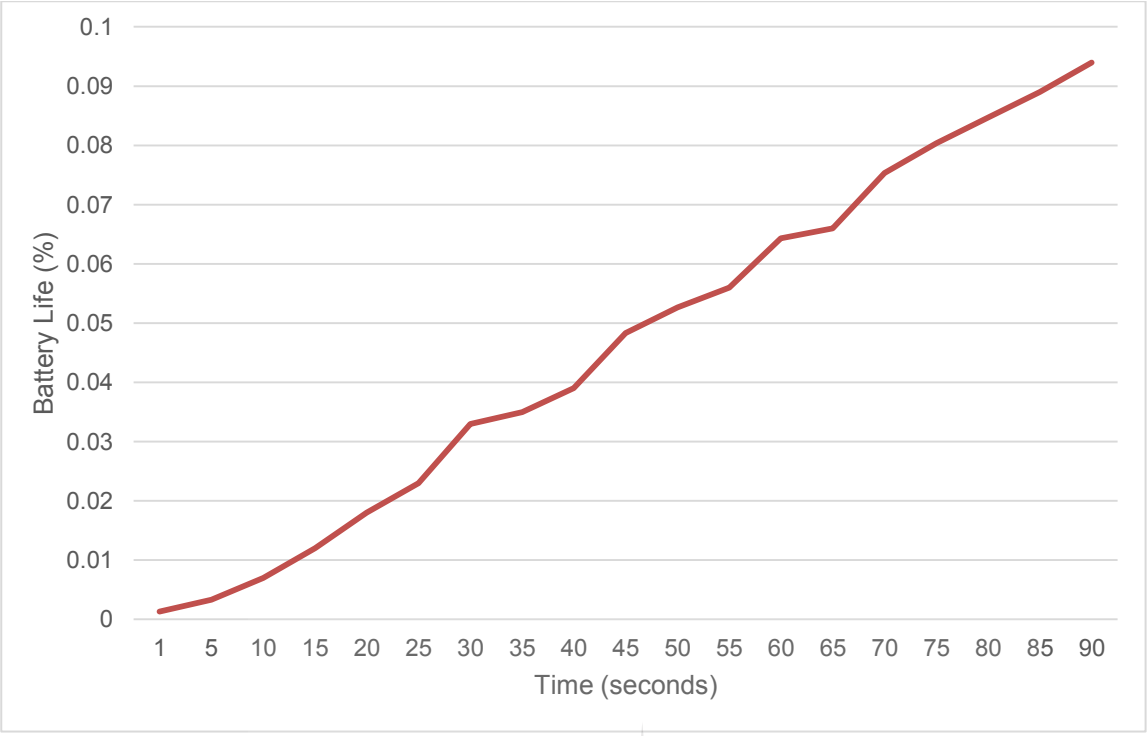

Figure 8.34 Battery life consumed over the length of local computation

In Figure 8.34 the battery life consumed increases as the length of time of time of computation increases. The battery life consumed increases from 0.001% when the CPU is active for 1 second to 0.094% when the CPU is used for 90 seconds.

The results of the computational experiments show that the longer the computation lasts, the more battery life is consumed. The results show that with the length of time of the computation the battery life consumed during computation can be calculated.

The data gathered is now used to create the Switch energy consumption profile.

8.5. Energy consumption profile

The data gathered for the Samsung Galaxy S7 Edge is used to create the linear models that are representative of the energy consumption when communicating over the different networks and when computing locally.

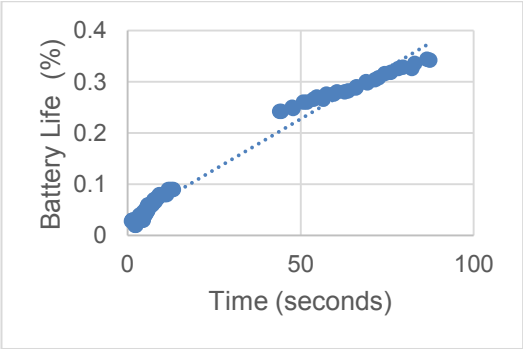

Figure 8.35 3G rate of consumption

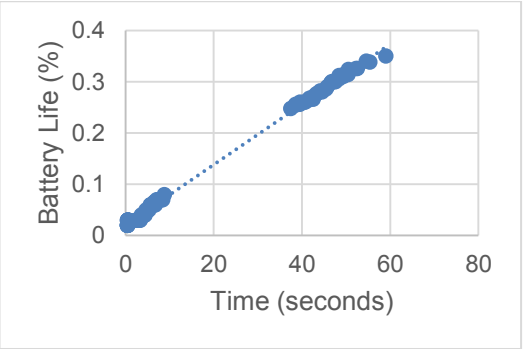

Figure 8.36 4G rate of consumption

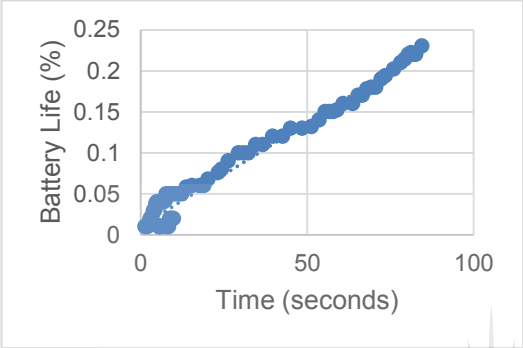

Figure 8.37 Wi-Fi rate of consumption

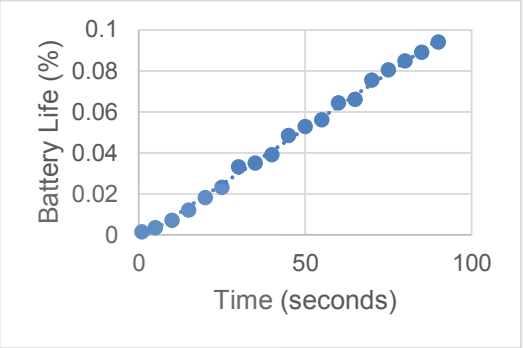

Figure 8.38 Computation rate of consumption

Figure 8.35 shows all the points of the length of time of communication and battery life consumed for 3G, both uploaded and downloaded. A linear line is fitted to the data points to get the average rate of consumption over time for the network. The equation of the fitted line for 3G is: $y = 0{,}004x + 0{,}0293$. The equation provided can now be substituted into the EC_{3G} energy consumption model in the energy consumption profile. The model used in the prototype is shown in equation 8.1.

Equation 8.1 3G energy consumption model

$$EC_{3G} = 0{,}004(time) + 0{,}0293$$

Figure 8.36 shows all the points of the length of time of communication and battery life consumed for 4G, both uploaded and downloaded. A linear line is fitted to the data points to get the average rate of consumption over time for the network. The equation of the fitted line for 4G is: $y = 0{,}0059x + 0{,}0207$. The equation provided can now be substituted into the EC_{4G} energy consumption model in the energy consumption profile. The model used in the prototype is shown in equation 8.2:

Equation 8.2 4G energy consumption model

$$EC_{4G} = 0{,}0059(time) + 0{,}0207$$

Figure 8.37 shows all the points of the length of time of communication and battery life consumed for Wi-Fi, both uploaded and downloaded. A linear line is fitted to the data points to get the average rate of consumption over time for the network. The equation of the fitted line for Wi-Fi is: $y = 0{,}0025x + 0{,}0105$. The equation provided can now be substituted into the EC_{wifi} energy consumption model in the energy consumption profile. The model used in the prototype is shown in equation 8.3.

Equation 8.3 Wi-Fi energy consumption model

$$EC_{wifi} = 0{,}0025(time) + 0{,}0105$$

Figure 8.38 shows all the points of the length of time of computation and battery life consumed for local computation. A linear line is fitted to the data points to get the average rate of consumption over time for the network. The equation of the fitted line for computation is: $y = 0{,}0011x - 0{,}0024$. The equation provided can now be substituted into the EC_{CPU} energy consumption model in the energy consumption profile. The model used in the prototype is shown in equation 8.4.

Equation 8.4 Local execution energy consumption model

$$EC_{CPU} = 0{,}0011(time) - 0{,}0024$$

The rates of consumption are used to estimate the energy consumption on a Samsung Galaxy S7 Edge, SM-G900F, which increases the accuracy of the estimation of battery life consumed. The energy consumption models presented above are used in the energy consumption profile in the Switch prototype.

8.6. Conclusion

This chapter discusses the experiments used to evaluate real-world energy consumption during communication and computation and the results gathered from these experiments.

The experiments for communication are executed by downloading and uploading files of varying sizes to and from the cloud. The size of uploaded and downloaded data, length of time of the communication, signal strength and battery life consumed are recorded. The results of the communication experiments show that there exists a direct relation between the length of the communication and the battery life consumed and there exists an indirect relation between the signal strength and the length of time of communication.

The experiments for computation are completed by calculating the number of primes for a set amount of time. The length of time of the computation and the battery life consumed is recorded. The results of the experiments show that there exists a direct relation between the length of time of computation and battery life consumed.

The manual creation of the energy consumption profile has laid the groundwork for the automatic creation of energy consumption profiles for various devices, however, this is not covered in the scope of this research.

The gathered results are used to create the energy consumption models. For each of the networks and for local computation a line is fitted to the collected data. The equation for the fitted line represents the model.

Chapter 9: Switch: Prototype

9.1. Introduction

In this chapter, the Switch framework is evaluated by creating a prototype of Switch that can be integrated with a mobile app that supports offloading. The purpose is to determine if the battery life usage can be reduced by intelligently offloading tasks to the cloud. A comparison of the actual energy consumption to the estimated energy consumption can determine the accuracy of the offloading decision.

The implementation of the device-specific energy consumption profile, profiler, offloading decision and mobile app is presented. The integration of Switch and subsequent executions of the tasks of the mobile app are used to collect data in order to evaluate the prototype.

In Section 9.2 the components of Switch are discussed and presented programmatically. The mobile apps that the Switch prototype are integrated with, and the results of the integration and execution, are discussed in section 9.3. Finally, the chapter is concluded.

9.2. Switch components

The components of the Switch framework, namely the profiler, energy consumption profile, and decision-making component, are discussed and presented in this section. Figure 9.1 shows the UML class diagram of the components of the Switch framework.

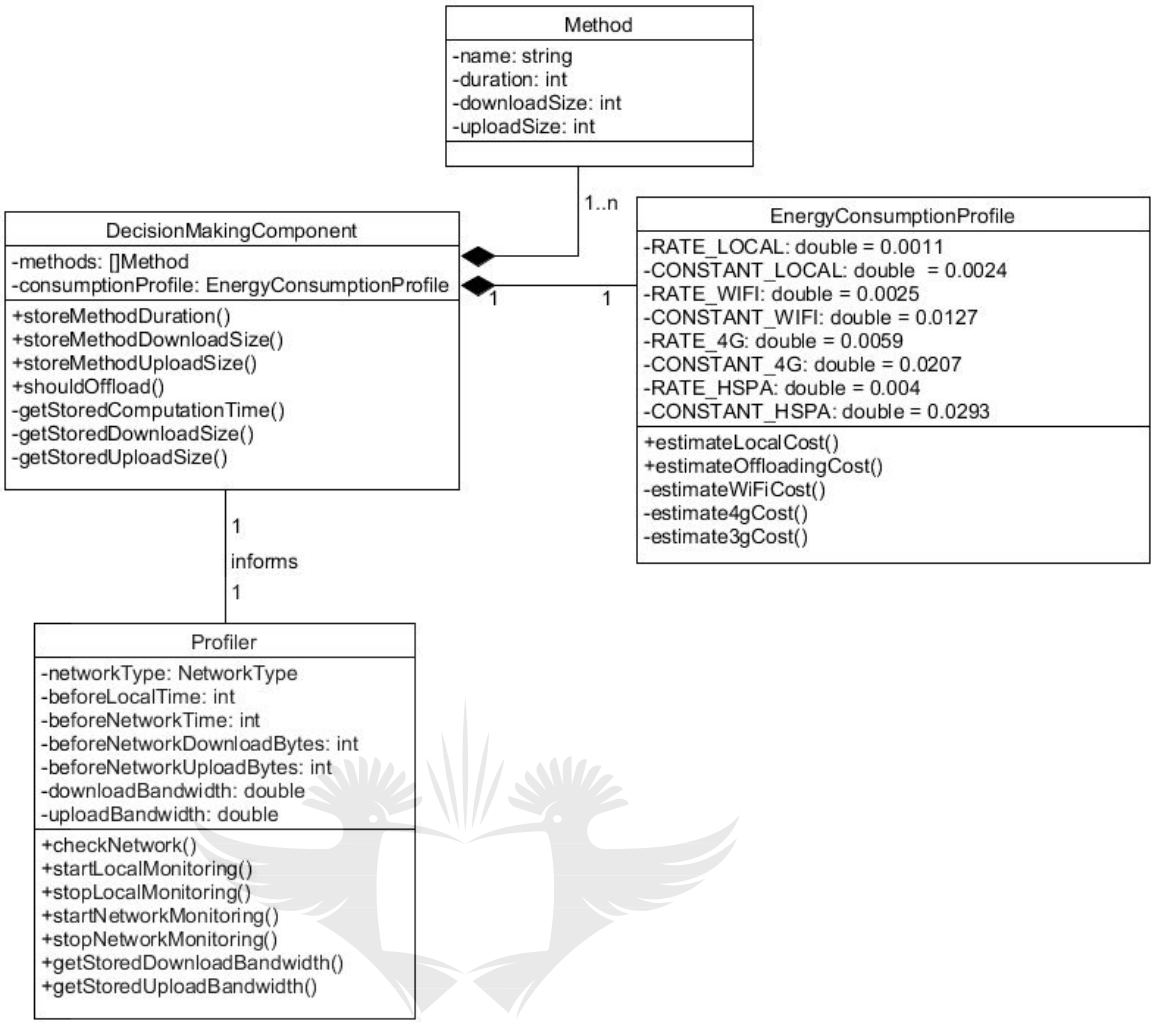

Figure 9.1 Class diagram of the Switch framework

9.2.1. Switch Profiler

The profiler collects data regarding the execution of the application and the current state of the mobile device. The methods made available by the profiler are, *checkNetwork*, *startLocalMonitoring*, *stopLocalMonitoring,* *startNetworkMonitoring*, and *stopNetworkMonitoring*, are now discussed.

a. *checkNetwork*

checkNetwork consists of two parts. The first queries the mobile operating system to determine which network the device is currently connected to. The second determines the available bandwidth on the network, this calculation is done by downloading and uploading files to and from a remote server and timing the duration of the communication. This method is shown in figure 9.2.

```
function checkNetwork() {
    networkType = getNetworkType();
    storeNetworkType(networkType);

    startNetworkMonitoring();
    downloadFile();
    uploadFile();
    stopNetworkMonitoring();
}
```

Figure 9.2 checkNetwork *method*

b. *startLocalMonitoring & stopLocalMonitoring*

startLocalMonitoring, shown in figure 9.3, is used to collect and store data before local execution. The stored data is used in *stopLocalMonitoring*, shown in figure 9.3, to calculate the duration of local execution.

```
function startLocalMonitoring() {
    beforeTime = getCurrentTime();
    storeBeforeLocalTime(beforeTime);
}

function stopLocalMonitoring(methodName) {
    afterTime = getCurrentTime();
    beforeTime = getStoredBeforeLocalTime();

    duration = (afterTime - beforeTime) / 1000;

    decisionMakingComponent.storeMethodDuration(methodName, duration);
}
```


Figure 9.3 startLocalMonitoring *and* stopLocalMonitoring *methods*

c. *startNetworkMonitoring & stopNetworkMonitoring*

startNetworkMonitoring, presented in figure 9.4, is used to collect and store data before network communication. The stored data is used in *stopNetworkMonitoring*, shown in figure 9.4, to calculate the available bandwidth and the size of the data communicated by the method.

```
function startNetworkMonitoring() {
    beforeTime = getCurrentTime();
    storeBeforeNetworkTime(beforeTime);

    beforeBytesDownload = getBytesDownloaded(applicationId);
    storeBeforeNetworkDownloadBytes(beforeBytesDownload);

    beforeBytesUpload = getBytesUploaded(applicationId);
    storeBeforeNetworkUploadBytes(beforeBytesUpload);
}

function stopNetworkMonitoring(methodName) {
    afterTime = getCurrentTime();
    beforeTime = getStoredBeforeNetworkTime();

    durationInSeconds = (afterTimeDownload - beforeTimeDownload) / 1000;

    afterBytesDownload = getBytesDownloaded(applicationId);
    beforeBytesDownload = getStoredBeforeNetworkDownloadBytes();

    downloadSizeInBits = (afterBytesDownload - beforeBytesDownload) * 8;

    decisionMakingComponent.storeMethodDownloadSize(methodName,
    downloadSizeInBits);

    downloadBandwidth = downloadSizeInBits / durationInSeconds;
    downloadBandwidthKbps = downloadBandwidth / 1000;
    downloadBandwidthMbps = downloadBandwidthKbps / 1000;

    storeDownloadBandwidth(downloadBandwidthMbps);

    afterBytesUpload = getBytesUploaded(applicationId);
    beforeBytesUpload = getStoredBeforeNetworkUploadBytes();

    uploadSizeInBits = (afterBytesUpload - beforeBytesUpload) * 8;

    decisionMakingComponent.storeMethodUploadSize(methodName,
    uploadSizeInBits);

    uploadBandwidth = uploadSizeInBits / durationInSeconds;
    uploadBandwidthKbps = uploadBandwidth / 1000;
    uploadBandwidthMbps = uploadBandwidthKbps / 1000;

    storeUploadBandwidth(uploadBandwidthMbps);
}
```

Figure 9.4 startNetworkMonitoring *and* stopNetworkMonitoring *methods*

9.2.2. Energy consumption profile

Using the energy consumption profile and the length of time of communication and computation the prototype can estimate the energy consumption. The *estimateLocalCost* and *estimateOffloadingCost* methods are used by Switch to estimate energy consumption when offloading over the different networks and when executing locally, the methods are shown in figure 9.5. The *estimateLocalCost* represent the equation 8.4 energy consumption model,

RATE_LOCAL and *CONSTANT_LOCAL* are values used to draw the line fitted to the collected data.

```
RATE_LOCAL = 0.0011
CONSTANT_LOCAL = 0.0024

function estimateLocalCost(time) : double {
    return RATE_LOCAL * time + CONSTANT_LOCAL
}

function estimateOffloadingCost(networkType, time) : double {
    switch (networkType)
        case WIFI: return estimateWiFiConsumption(time)
        case FOURG: return estimate4gConsumption(time)
        case TREEG: return estimate3gConsumption(time)
}
```

Figure 9.5 estimateLocalCost *and* estimateOffloadingCost *methods*

The methods used by *estimateOffloadingCost*, namely *estimateWiFiConsumption*, *esimate4gConsumption* and *estimate3gConsumption* are shown in figure 9.6. The methods each represent a linear model, *estimate3gConsumption* is a representation of equation 8.1, *estimate4gConsumption* represents equation 8.2 and *estimateWiFiConsumption* represents equation 8.3. The constants and rates used in these methods are determined by the line fitted to the data collected for each network.

```
RATE_WIFI = 0.0025
CONSTANT_WIFI = 0.0127

RATE_4G = 0.0059
CONSTANT_4G = 0.0207

RATE_3G = 0.004
CONSTANT_3G = 0.0293

function estimateWiFiConsumption(time) : double {
    return RATE_WIFI * time + CONSTANT_WIFI
}

function estimate4gConsumption(time) : double {
    return RATE_4G * time + CONSTANT_4G
}

function estimate3gConsumption(time) : double {
    return RATE_3G * time + CONSTANT_3G
}
```

Figure 9.6 estimateWiFiConsumption, esimate4gConsumption *and* estimate3gConsumption *methods*

9.2.3. Decision-making component

The decision-making component makes the offloading decision. The offloading decision is made by the *shouldOffload* method, as shown in figure 9.7.

```
function shouldOffload(methodName) : boolean {
    computationTime = getStoredComputationTime(method);
    localCost = consumptionProfile.estimateLocalCost(computationTime);

    downloadSize = getStoredDownloadSize(methodName);
    downloadBandwidth = profiler.getStoredDownloadBandwidth();
    downloadTime = downloadSize / downloadBandwidth;

    uploadSize = getStoredUploadSize(methodName);
    uploadBandwidth = profiler.getStoredUploadBandwidth();
    uploadTime = uploadSize / uploadBandwidth;

    communicationTime = downloadTime + uploadTime;

    network = profiler.getNetworkType();
    offloadingCost =
consumptionProfile.estimateOffloadingCost(network,communicationTime);

    return localCost > offloadingCost;
}
```

Figure 9.7 shouldOffload *method*

The next section discusses the application the Switch prototype is integrated with and the results of using the prototype.

9.3. Implementation and evaluation

The goal of evaluating Switch is to compare the calculated estimated energy consumption and the measured energy consumption. This section discusses and evaluates the experiments and the results of the experiments executed with the prototype.

9.3.1. Results

The functions discussed earlier are used to determine whether or not a task should be offloaded. The solution is implemented in both a steganography mobile application and a prime number calculation mobile application that each illustrates different constraints with respect to data communication that Switch needs to consider.

- *Steganography* is used because it is a computationally intensive task and when offloaded it requires the transfer of large amounts of data.
- *Prime number generation* is used because it is a computationally intensive task that requires very little data to be communicated.

The mobile apps that are used to evaluate the prototype offload to a Heroku container (Heroku, 2017a).The Heroku container used for the evaluation is on the free tier, it has 512MB of memory, and has 1x CPU share and between 1x and 4x compute share (Heroku, 2017b). Although the resources available in the container used for offloading is not limitless it does surpass the available resources on the mobile device.

As mentioned, Switch is evaluated for both local computations and offloading over different types of networks where the estimated energy consumption is compared to the actual energy consumption. When estimating the energy consumption for offloading, the available bandwidth is measured and used to estimate the length of time of communication. The length of time of communication is used to estimate the energy consumption. The available bandwidth is used instead of the signal strength because the signal strength is not a reliable indicator of how long communication lasts.

To determine the estimation of energy consumption for the execution of a local task, the length of time of computation is measured. The actual energy consumption is measured by using GSam Battery Monitor, the same software that was used to create the energy consumption profile. The same methodology is used to measure the energy consumption for each of the tasks used to evaluate the prototype.

Steganography mobile application

a. *Task 1 – Steganographic file encoding*

The first task used is the steganographic encoding a 1.4 megabyte file into a 5 megabyte image. Offloading this task require the image and the file to be uploaded and the encoded image to be downloaded. The task is executed locally, over Wi-Fi, 4G, and 3G.

i. Local

Executing the file encoding task locally takes the device 7.391 seconds, using the power consumption estimation functions, the task should consume 0.057% battery life. Measuring the energy consumption for this task, return the actual energy consumption of 0.058%.

ii. 3G

Executing the file encoding task over HPSA completes in 51 seconds. The bandwidth available over HPSA for downloading at the time of the experiment is 2.6179 Mbps and 2.3104 Mbps for uploading. The prototype estimates the energy consumption is 0.1768%. The actual energy consumption is 0.177%.

iii. 4G

Executing the file encoding task over 4G completes in 34 seconds. The bandwidth available over 4G for downloading at the time of the experiment is 8.0353 Mbps and 5.8963 Mbps for uploading. The prototype estimates the energy consumption is 0.0983%. The actual energy consumption is 0.098%.

iv. Wi-Fi

Executing the file encoding task over Wi-Fi completes in 104 seconds. The bandwidth available over Wi-Fi for downloading at the time of the experiment is 1.416 Mbps and 0.6462 Mbps for uploading. The prototype estimates the energy consumption is 0.2578%. The actual energy consumption is 0.258%.

v. Comparison

The data collected from the executions of this task is shown in table 9.1. The estimated cost of task execution is compared to the actual cost that was measured.

Table 9.1 Steganographic file encoding results

	Time	**Estimated cost**	**Measured cost**
Local	7.291 seconds	0.057%	0.058%
HPSA	51 seconds	0.1768%	0.177%
4G	34 seconds	0.0983%	0.098%
Wi-Fi	104 seconds	0.2578%	0.258%

A large amount of data required to be transferred skews this task to be executed locally. It only takes 7.291 seconds to execute the task and consumes 0.058% of the battery life. As can be expected, it takes much longer to offload a task using a Wi-Fi connection and this consumes more battery life. Even though a task executed using a 4G connection takes 34 seconds to complete, it is still much slower than a local execution and consumes more battery life.

From the results, it is clear that the estimated energy consumption is relatively accurate when compared to the actual battery life as measured by software.

b. *Task 2 – Steganographic file decoding*

The second task is to decode a 5 megabyte image. The encoded file contains a 1.4 megabyte file. Offloading this task requires the image to be uploaded and the encoded file to be downloaded. The task is executed locally, over Wi-Fi, 4G, and 3G.

i. Local

Executing the file decoding task locally takes the device 2.304 seconds, using the power consumption estimation functions, the task should consume 0.0013% battery life. Measuring the energy consumption for this task, return the actual energy consumption of 0.001%.

ii. 3G

Executing the file decoding task over 3G completes in 30 seconds. The bandwidth available over 3G for downloading at the time of the experiment is 1.3434 Mbps, and 1.312 Mbps for uploading. The prototype estimates the energy consumption is 0.1825%. The actual energy consumption is 0.183%.

iii. 4G

Executing the file decoding task over 4G completes in 10 seconds. The bandwidth available over 4G for downloading at the time of the experiment is 9.0265 Mbps, and 6.0004 Mbps for uploading. The prototype estimates the energy consumption is 0.075%. The actual energy consumption is 0.076%.

iv. Wi-Fi

Executing the file decoding task over Wi-Fi completes in 114 seconds. The bandwidth available over Wi-Fi for downloading at the time of the experiment is 4.0074 Mbps, and 0.7109 Mbps for uploading. The prototype estimates the energy consumption is 0.1846%. The actual energy consumption is 0.185%.

v. Comparison

The data collected from the executions of this task is shown in table 9.2.

Table 9.2 Steganographic file decoding results

	Time	**Estimated cost**	**Measured cost**
Local	2.304 seconds	0.0011%	0.001%
HPSA	30 seconds	0.1825%	0.183%
4G	10 seconds	0.075%	0.076%
Wi-Fi	114 seconds	0.1846%	0.185%

Comparing the results of this tasks with the results from the first task shows that in each instance this task consumes less battery life. The decrease in battery

life consumption can be attributed to the decrease in the amount of data that is required to be transferred from 11.4 megabytes to 6.4 megabytes. As with the first task, the amount of data that is required to be transferred skews the task to be executed locally. Local execution takes 2.304 seconds and consumes 0.001% of the battery life. The fastest alternative when offloading, 4G, takes 10 seconds and consumes 0.076% battery life. Due to the lower bandwidth available on Wi-Fi during testing the execution is even slower, 114 seconds, and consumes much more battery life, 0.185%.

Prime number calculation mobile application

c. *Task 3 – Prime number counting*

The third task used to evaluate Switch is calculating the number of primes there are between zero and ten million (10 000 000). Offloading this tasks requires the upload of an integer (4 bytes) and the download of an integer.

i. Local

Executing the prime number counting task locally takes the device 31 seconds, using the power consumption estimation functions, the task should consume 0.0317% battery life. Measuring the energy consumption for this task, return the actual energy consumption of 0.032%.

ii. 3G

Executing the prime number counting task over 3G completes in 16 seconds. The bandwidth available over 3G for downloading at the time of the experiment is 3.8563 Mbps, and 2.0361 Mbps for uploading. The prototype estimates the energy consumption is 0.0293%. The actual energy consumption is 0.03%.

iii. 4G

Executing the prime number counting task over 4G completes in 12 seconds. The bandwidth available over 4G for downloading at the time of the experiment is 9.1174 Mbps, and 7.3823 Mbps for uploading. The prototype estimates the energy consumption is 0.0207%. The actual energy consumption is 0.02%.

iv. Wi-Fi

Executing the prime number counting task over Wi-Fi completes in 10 seconds. The bandwidth available over Wi-Fi for downloading at the time of the experiment is 3.5499 Mbps, and 1.1372 Mbps for uploading. The prototype estimates the energy consumption is 0.0127%. The actual energy consumption is 0.013%.

v. Comparison

The data collected from the executions of this task is shown in table 9.3.

Table 9.3 Prime number counting results

	Time	Estimated cost	Measured cost
Local	31 seconds	0.0317%	0.032%
HPSA	16 seconds	0.0293%	0.03%
4G	12 seconds	0.0207%	0.02%
Wi-Fi	10 seconds	0.0127%	0.013%

The amount data to be transferred for this task is extremely small and the task is computationally complex, this skews the task toward offloading. Due to the complexity of the task execution on the mobile device takes 31 seconds and consumes 0.032% battery life. The battery life consumed when offloading over 4G is 0.02% and only takes 12 seconds. Over Wi-Fi, the same operation takes 10 seconds and consumes 0.013% battery life.

9.3.2. Evaluation

The three tasks executed to evaluate the Switch prototype were selected because they are computationally intensive and used different quantities of data to be communicated to and from the cloud.

When comparing the estimated cost for local execution with the cost when offloading, for each of the networks, for the first and second tasks, the prototype suggests that local execution conserves battery life. Manual inspection of the measured energy consumption shows that it is the least expensive option.

For the third task, the prototype suggests offloading in all cases when comparing the energy consumption estimates. Again, manual inspection of the

measured energy consumption values shows that local execution is the least expensive option.

The estimated energy consumption and the measured energy consumption values are nearly identical, as shown in tables 9.1, 9.2 and 9.3. The differences between the values are attributed to the difficulty in accurately measuring energy consumption on Android devices.

In order to evaluate the prototype, one needs to evaluate the accuracy of energy consumption estimates. Therefore, the accuracy of the energy consumption models is now evaluated.

a. *Switch experimentation percentage error*

For each of the models, the average percentage error is calculated. The percentage error is used to show the difference between estimated values and measured values (Helmenstine, 2017).

The formula used to calculate the percentage error is shown in equation 9.1.

Equation 9.1 Formula for percentage error

$$percent\ error = \frac{|measured - estimated|}{measured} \times 100$$

For each of the energy consumption models, for each of the tasks, the percentage error is calculated and averaged to give the average percentage error for the models. The inverse of the percentage error shows the accuracy of the model. The accuracy percentage is calculated using the formula shown in equation 9.2.

Equation 9.2 Formula for percentage accuracy

$$accuracy\ percent = 100 - percent\ error$$

The calculation of the percentage of errors for the CPU energy consumption model is shown in table 9.4.

Table 9.4 Percentage error calculation for the CPU model

	Estimated cost	Measured cost	Percentage error
Task 1	0.057%	0.058%	1.724137931%
Task 2	0.0011%	0.001%	10%
Task 3	0.0317%	0.032%	0.9375%

Averaging the percentage errors results in an average percentage error of 4.22%. The average percentage error shows that the CPU energy consumption model has an accuracy percentage of 95.78%.

The calculation of the percentage error for the 3G energy consumption model is shown in table 9.5 using the results from the tasks used to evaluate Switch.

Table 9.5 Error rate calculation for the 3G model

	Estimated cost	Measured cost	Percentage error
Task 1	0.1768%	0.177%	0.11%
Task 2	0.1825%	0.183%	0.27%
Task 3	0.0293%	0.03%	2.33%

The average percentage error for the model is 0.91%. The accuracy percentage calculated for the 3G energy consumption model is 99.09%.

The calculation of the percentage error for the results of the 4G energy consumption model is shown in table 9.6.

Table 9.6 Error rate calculation for the 4G model

	Estimated cost	Measured cost	Percentage error
Task 1	0.0983%	0.098%	0.31%
Task 2	0.075%	0.076%	0.79%
Task 3	0.0207%	0.02%	3.5%

Averaging the percentage errors results in an overall percentage error for the model of 1.71%. Using this percentage error to calculate the accuracy percentage shows that the 4G energy consumption model is 98.29% accurate.

The calculation of the error rate for the Wi-Fi energy consumption model is shown in table 9.7.

Table 9.7 Error rate calculation for the Wi-Fi model

	Estimated cost	Measured cost	Percentage error
Task 1	0.2578%	0.258%	0.08%
Task 2	0.1846%	0.185%	0.22%
Task 3	0.0127%	0.013%	2.31%

The average percentage error for the Wi-Fi energy consumption model is 0.87%. The accuracy percentage for the model is thus 99.13%

The high accuracy percentages of the models show that the costs estimated by Switch are very close to the measured values. The accuracy of energy consumption profile allows Switch to make accurate offloading decisions to conserve battery life.

9.4. Conclusion

This chapter discusses and evaluates the prototype of Switch. The components of the prototype, the implementation and the evaluation of the prototype are discussed. The discussion of the components briefly presents the components programmatically.

The prototype is evaluated by executing 3 tasks and evaluating the accuracy of the estimated energy consumption with the measured energy consumption. The first task used to evaluate the prototype is the steganographic encoding a file into an image. The second task is decoding an encoded image and retrieving the file encoded in it. The third task is counting the number of primes between zero and ten million. Each task is executed locally and offloaded over 3G, 4G and Wi-Fi.

The results gathered from the first task show that a large amount of data that is required to be transferred skews the task to local execution. Local execution of the steganographic encoding is the fastest option and the option that consumes the least amount of battery life. The decision made by the Switch prototype is to execute the task locally in all cases.

The second task also requires a large amount of data to be transferred, however, it is almost half the amount of data of the first task. The amount of

data again makes local execution the better choice. Local execution is the fastest and least expensive option for steganographic decoding. Again the decision made by the prototype is to execute locally in all cases.

The third task requires very little data to be transferred and is computationally complex, this skews the task toward offloading. The prototype suggested in each case that the task is offloaded, as the local execution was the most expensive option and took the longest time.

The results gathered from the execution of the three tasks are used to evaluate the accuracy of the energy consumption models. The CPU energy consumption model has the lowest accuracy of 95.78%, an accuracy percentage that is still high. The other models have accuracy percentages of 99.09%, for the 3G model, 98.47%, for the 4G model, and 99.13% for the Wi-Fi model.

The overall accuracy of the energy consumption profile enabled the Switch prototype to make offloading decisions that conserve the limited battery life on resource-constrained mobile devices.

Chapter 10: Conclusion

10.1. Introduction

This dissertation aims to prove that the battery life on mobile devices can be conserved by leveraging the cloud. It was identified that battery life, a resource that is integral to the operation of mobile devices, cannot be augmented by directly using resources from the cloud. Considering this limitation, this topic presented itself as an appropriate topic for research.

This chapter is the culmination of research covered throughout the course of this dissertation. The chapter sets out to address the research objective, by answering the research questions in section 10.2. Section 10.3 describes the limitations of the Switch framework. Section 10.4 gives the research contribution and future work that can be done to improve the Switch framework. Finally, the chapter and dissertation draw to a close in section 10.5.

10.2. Revisiting the research objective and questions

The primary goal of this dissertation is to investigate the energy consumption of mobile devices to determine whether or not the battery life of mobile devices can be conserved by making offloading decisions based on accurate energy consumption estimates. The development of the Switch framework answers the research questions posed in chapter 1. Each of the research questions is now revisited.

10.2.1. What resources are constrained on mobile devices, and which of them can be augmented?

Due to the nature of mobile devices, all of the resources on a mobile device is constrained, as discussed in chapter 2. However, the majority of resources such as computing power, memory, and storage, can be augmented with resources from the cloud.

a. ***Which mobile devices resources can be augmented, and which cannot?***

In chapter 2 it is identified that mobile devices have the same resources as traditional computers, namely: computing power, memory, and storage and all

can display information to the end user. The examination of cloud computing and mobile cloud computing in chapters 3 and 4 show that the resources that are found in both the cloud and mobile devices can be augmented. Battery life and bandwidth cannot be augmented by offloading.

b. ***What methods can be used to augment the resources on mobile devices?***

In chapter 3 the methods that can be used to augment the resources on mobile devices are identified as hardware, software and offloading. Upgrading the hardware components of a device to increase the resources available. Software can be used to efficiently use the limited resources. When the software on the mobile device uses offloading, the mobile device has access to the greater resource pool of the server.

c. ***How are mobile devices resources augmented by using the cloud?***

The discussion in chapter 4 shows that the computing capabilities of mobile devices can be augmented by executing resource-intensive mobile application components in the resource-rich cloud-based resources.

10.2.2. What are the requirements of a framework that can conserve battery life on mobile devices by using offloading?

The requirements identified in chapter 3 of this dissertation are:

- Intelligent offloading decisions
- Multiple network support
- Lightweight
- Portable

The requirements identified are based on the research objectives and are used to evaluate the framework proposed in the dissertation.

10.2.3. How can an offloading decision be designed to conserve the battery life of a mobile device?

The decision-making process is defined in chapter 6 as the process of identifying and choosing an option between several alternative options based on factors and the goal of the decision maker. An offloading decision is a

process of choosing between executing a task locally and executing the task remotely. In order to conserve battery life, the energy cost of executing the task locally or executing the same task remotely should be known. Practically, the energy used in a specific task cannot be known before execution, therefore a method of estimating the energy usage of a specific task is required. Thus, in order to conserve battery life, an energy estimation technique is designed.

a. *What is offloading and what approaches can be used to offload from mobile devices?*

Offloading is defined in chapter 5 as the process of moving a task from a mobile device to the cloud. Two offloading approaches are discussed namely virtual machine cloning and code-based offloading. Virtual machine cloning creates a virtual instance of the mobile device on the cloud where the virtual instance has access to the resources on the cloud. Code-based offloading is chosen for the Switch framework as it is simpler and more efficient as it relies on the creation of methods on the server that perform the same task as the methods on the mobile device.

b. *How can energy consumption be measured?*

The energy consumption of a mobile device or an app can be measured using either a hardware or software-based approach, as discussed in chapter 6. Hardware-based approaches result in profiles that are highly accurate but the creation of these profiles are labor-intensive and not scalable. Software-based approaches result in less accurate profiles. However, software-based energy consumption profiles enable the monitoring of the application on many different granularities and it does not require expensive external power meters.

c. *Which factors should be taken into account when estimating energy consumption?*

Chapter 5 discusses the offloading decisions made by frameworks proposed in related research. The understanding gained from the evaluation of related research is expanded on in chapter 6 to identify the factors that influence the estimation of energy consumption. The factors identified are grouped into two categories, namely communication, and computation. The factors in the communication category are size of data, communication protocol and

bandwidth. The computation category factors are code complexity and the duration of execution.

10.2.4. Does the framework proposed by this dissertation conserve battery life on mobile devices?

The Switch framework is proposed in chapter 7 and the prototype implemented in chapter 9 is evaluated against the requirements identified in chapter 3.

a. ***What tasks and evaluation criteria can be used to determine the effectiveness of the proposed framework?***

Three tasks were identified in chapter 9 to evaluate the Switch framework. The first task involves the steganographic encoding of a file into an image, this task when offloaded requires the upload of two files, one image and one data file, and the download of the encoded image file. The second task involves the decoding of an image that has been steganographically encoded when offloaded this requires the image to be uploaded and the decoded file to be downloaded. The third task involves the counting of prime numbers, this computationally intensive task does not require large amounts of data to be transferred when offloading.

The proposed framework, Switch, is evaluated by comparing the estimated energy consumption and the measured energy consumption to determine the accuracy of the offloading decisions made by the framework.

b. ***To what extent does the proposed framework meet the identified evaluation criteria and which deficiencies and be identified?***

The Switch prototype is evaluated against the identified requirements to answer this question.

i. *Intelligent offloading decision making*

As discussed in chapter 7, in order for an intelligent offloading decision to occur, an accurate energy estimation is required to inform the decision-making process. In chapter 9, the analysis showed that the prototype provides an accurate energy estimation of real-world tasks. Therefore, the accurate energy

estimations can be used to intelligently inform the offloading decisions made by the Switch prototype.

ii. Multiple network support

Mobile devices are by their nature mobile, thus they are continuously connecting to different networks. Because mobile devices are used when connected to different networks, the prototype is capable of estimating energy consumption and make offloading decisions regardless of the network connection. The energy consumption profile generated for the prototype takes into account Wi-Fi, 4G and 3G networks, which allows offloading decisions to be made when communicating via one of the networks.

iii. Lightweight

By adding the prototype to an existing application, the energy consumption on the mobile device is minimally affected and no detrimental effect on the user experience is noted. The prototype collects environmental data in the background of the application which does not affect the user, and the offloading decisions are made as simple as possible so that it requires as little computational power and battery life as possible.

iv. Portable

The Switch prototype is developed as an external package that can be included by any developer into an app to enable the conservation of battery life. The package has a simple interface and is easy to integrate into an application. Because no device-specific functions are used, the package is not device specific and can be included on any Android device, for any app. Thus, the prototype is portable.

The next section addresses the limitations of this research.

10.3. Limitation of this research

In order to determine whether battery life can be conserved on mobile devices, it was necessary to implement an application that is capable of executing tasks locally and offloading them to the cloud. The implementation of the prototype in

this application allowed the evaluation of the framework in a practical environment.

The prototype is not perfect. There are several areas in which it can be improved. The first of which is the automatic creation of an energy consumption profile. The limitations on the Android operating system requires the use of a third-party app to measure energy consumption. The differences in the hardware available on mobile devices require an energy consumption profile for each device. The profile used was not tested against other physical devices with the same hardware specifications.

Another area in which the prototype can be improved is the integration with other applications. In the current state, the prototype has to be integrated by the developer and end-users cannot control it. The prototype requires developers to identify which tasks can be offloaded and create endpoints through which the task can be offloaded.

It is important to note that it would be important to secure all communications between the mobile device and the cloud, as well as the data and tasks on the cloud. This aspect should be addressed very carefully but is beyond the scope of this research.

The prototype helped answer the primary research goal of this dissertation and can be improved upon. The next section discusses how the research can be improved upon in future work.

10.4. Research contribution and future work

The research done in this dissertation shows that a framework can be created that can be integrated into any mobile app to conserve the battery life of the mobile device by making offloading decisions informed by accurate energy consumption estimations. The concept of conserving battery life by offloading has been shown to be effective in the existing problem domain.

Previous prototypes that were develop were focussed on specific fields, required specialized hardware to function or to measure energy consumption, or required in-depth analysis of the methods executed by the app that the prototypes were integrated with. The framework proposed in this dissertation took a different approach in that it aimed to be integrated with any mobile app that uses offloading and software to measure energy consumption. The framework in this dissertation is focussed on a device-specific energy consumption profile and not on the energy consumption of the mobile app. With the energy consumption profile created for a device, minimal measurements are required to be made by the developer. As verified in the prototype, the battery life of a mobile device can be conserved by integrating the prototype into an app.

The ongoing improvements in the fields of mobile devices and cloud computing leads to improvements that can be made in the prototype and framework. The following is a list of considerations:

- The manual creation of the energy consumption profile of a mobile device can be automated by creating a test suite that measures the energy consumption under different circumstances.
- The information gathered by the profiler during execution can be used to update the existing energy consumption profile.
- The offloading decision can be expanded to include monetary cost instead of just battery life based on user preference.

The list of items above is not an exhaustive list of changes that can be made. This list serves to illustrate some of the limitations of the framework on a conceptual level that could be improved through future work as they are beyond the scope of the objective of this dissertation. The following section concludes the dissertation.

10.5. Conclusion

The chapter has provided a brief overview of the content of this dissertation. This research focussed on the creation of the Switch framework that can be integrated into any application that uses offloading to conserve battery life by

making offloading decisions informed by accurate energy consumption estimates. In order to achieve this objective, various topics related to the problem domain are explored in this dissertation.

The Switch framework and prototype was developed and tested to demonstrate that a component can be created to accurately estimate energy consumption on a mobile device. Such estimations can be used to make offloading decisions without requiring specialised hardware or in-depth analysis of methods.

To this end, this dissertation concludes that a component can be created that can be integrated into any application to conserve the battery life of a mobile device by accurately estimating energy consumption and making a decision between offloading and local execution.

References

&

Appendix

References

3GPP, 2017. GPRS & EDGE. Available: http://www.3gpp.org/technologies/keywords-acronyms/102-gprs-edge (Accessed 3 November 2017).

Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., Buyya, R., 2014. Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges. IEEE Commun. Surv. Tutor. 16, 337–368. https://doi.org/10.1109/SURV.2013.070813.00285

Abolfazli, S., Sanaei, Z., Shiraz, M., Gani, A., 2012. MOMCC: Market-oriented architecture for Mobile Cloud Computing based on Service Oriented Architecture, in: 2012 1st IEEE International Conference on Communications in China Workshops (ICCC). Presented at the 2012 1st IEEE International Conference on Communications in China Workshops (ICCC), pp. 8–13. https://doi.org/10.1109/ICCCW.2012.6316481

Afrihost, 2017a. The Afrihost Story. Afrihost Internet Serv. Available: https://www.afrihost.com/site/page/the_afrihost_story (Accessed 19 March 2017).

Afrihost, 2017b. Afrihost Mobile Data from only R29pm. Afrihost Internet Serv. Available: https://www.afrihost.com/site/product/mobile_data (Accessed 19 March 2017).

Ahmad, R.W., Gani, A., Hamid, S.H.A., Xia, F., Shiraz, M., 2015. A Review on mobile application energy profiling: Taxonomy, state-of-the-art, and open research issues. J. Netw. Comput. Appl. 58, 42–59. https://doi.org/10.1016/j.jnca.2015.09.002

Akherfi, K., Gerndt, M., Harroud, H., 2016. Mobile cloud computing for computation offloading: Issues and challenges. Appl. Comput. Inform. https://doi.org/10.1016/j.aci.2016.11.002

Ali, M., Zain, J.M., Zolkipli, M.F., Badshah, G., 2015. Battery efficiency of mobile devices through computational offloading: A review, in: 2015 IEEE Student Conference on Research and Development (SCOReD). Presented at the 2015 IEEE Student Conference on Research and Development (SCOReD), pp. 317–322. https://doi.org/10.1109/SCORED.2015.7449347

Alizadeh, M., Hassan, W.H., 2013. Challenges and opportunities of Mobile Cloud Computing, in: Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International. Presented at the Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International, pp. 660–666. https://doi.org/10.1109/IWCMC.2013.6583636

Al-mousa, A., Alzoubi, A., 2017. Intelligent offloading of reports processing in aging mobile devices, in: 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and

Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). Presented at the 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1–4. https://doi.org/10.1109/UIC-ATC.2017.8397652

Alomari, S.A., Sumari, P., Taghizadeh, A., 2011. A Comprehensive Study of Wireless Communication Technology for the Future Mobile Devices. Eur. J. Sci. Res. 60, 565–573.

Amazon, 2014a. AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. Amazon. Available: https://aws.amazon.com/ec2/ (Accessed 2 December 2014).

Amazon, 2014b. AWS | Amazon Simple Storage Service (S3) - Online Cloud Storage for Data & Files. Amazon. Available: http://aws.amazon.com/s3 (Accessed 2 December 2014).

Apple, 2016a. iPhone SE - Technical Specifications. Apple South Afr. Available: http://www.apple.com/za/iphone-se/specs/ (Accessed 20 November 2016).

Apple, 2016b. Swift. Apple. Available: http://www.apple.com/swift/ (Accessed 22 November 2016).

Apple, 2016c. iPhone 7 - Technical Specifications. Apple South Afr. Available: http://www.apple.com/za/iphone-7/specs/ (Accessed 20 November 2016).

Apple, 2014a. What is iOS. Apple. Available: https://www.apple.com/ios/what-is/ (Accessed 14 November 2014).

Apple, 2014b. Programming with Objective-C: About Objective-C. Apple. Available: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html (Accessed 25 November 2014).

Apple, 2014c. Official Apple Store US - iPhone, iPad and more. Apple. Available: http://store.apple.com/us (Accessed 25 November 2014).

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M., 2010. A View of Cloud Computing. Commun ACM 53, 50–58. https://doi.org/10.1145/1721654.1721672

Bahl, P., Han, R.Y., Li, L.E., Satyanarayanan, M., 2012. Advancing the State of Mobile Cloud Computing, in: Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services, MCS '12. ACM, New York, NY, USA, pp. 21–28. https://doi.org/10.1145/2307849.2307856

Balan, R.K., Satyanarayanan, M., Park, S.Y., Okoshi, T., 2003. Tactics-based Remote Execution for Mobile Computing, in: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, MobiSys '03. ACM, New York, NY, USA, pp. 273–286. https://doi.org/10.1145/1066116.1066125

Barbera, M.V., Kosta, S., Mei, A., STEFA, J., 2013. To offload or not to offload? The bandwidth and energy costs of mobile cloud computing, in: 2013 Proceedings IEEE INFOCOM. Presented at the 2013 Proceedings IEEE INFOCOM, pp. 1285–1293. https://doi.org/10.1109/INFCOM.2013.6566921

Ben, D., Ma, B., Liu, L., Xia, Z., Zhang, W., Liu, F., 2009. Unusual Burns With Combined Injuries Caused by Mobile Phone Explosion: Watch Out for the "Mini-Bomb"! J. Burn Care Res. Off. Publ. Am. Burn Assoc. 30, 1048. https://doi.org/10.1097/BCR.0b013e3181bfb8c0

Bernstein, D., 2014. Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Comput. 1, 81–84. https://doi.org/10.1109/MCC.2014.51

Blu, 2016. Studio 7.0 II. BluProducts.com. Available: http://bluproducts.com/studio-7-0-ii (Accessed 20 November 2016).

Bluetooth SIG, 2014. Specification. Bluetooth SIG. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications (Accessed 25 November 2014).

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I., 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25, 599–616. https://doi.org/10.1016/j.future.2008.12.001

Carroll, A., Heiser, G., 2010. An Analysis of Power Consumption in a Smartphone, in: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10. USENIX Association, Berkeley, CA, USA, pp. 21–21.

Chen, X., Liu, B., Chen, Y., Zhao, M., Xue, C.J., Guo, X., 2012. Active compensation technique for the thin-film transistor variations and OLED aging of mobile device displays, in: 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). Presented at the 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 516–522.

Cherry, S., 2008. Wi-Fi Takes On Bluetooth - IEEE Spectrum. IEEE Spectr.

Christensen, J.H., 2009. Using RESTful Web-services and Cloud Computing to Create Next Generation Mobile Applications, in: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA '09. ACM, New York, NY, USA, pp. 627–634. https://doi.org/10.1145/1639950.1639958

Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., Patti, A., 2011. CloneCloud: Elastic Execution Between Mobile Device and Cloud, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys '11. ACM, New York, NY, USA, pp. 301–314. https://doi.org/10.1145/1966445.1966473

Chun, B.-G., Maniatis, P., 2009. Augmented smartphone applications through clone cloud execution, in: Proceedings of the 12th Conference on Hot Topics in Operating Systems. USENIX Association, pp. 8–8.

CIO Council, 2013. Government Mobile and Wireless Security Baseline. CIO Council.

Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A., 2005. Live Migration of Virtual Machines, in: Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05. USENIX Association, Berkeley, CA, USA, pp. 273–286.

Cleveland, W.S., 1993. Visualizing Data. At&T Bell Laboratories.

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., Bahl, P., 2010. MAUI: Making Smartphones Last Longer with Code Offload, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10. ACM, New York, NY, USA, pp. 49–62. https://doi.org/10.1145/1814433.1814441

De Silva, D.I., Kodagoda, N., Perera, H., 2012. Applicability of three complexity metrics, in: 2012 International Conference on Advances in ICT for Emerging Regions (ICTer). Presented at the 2012 International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 82–88. https://doi.org/10.1109/ICTer.2012.6421409

Demeyer, S., 2011. Research methods in computer science, in: 2011 27th IEEE International Conference on Software Maintenance (ICSM). Presented at the 2011 27th IEEE International Conference on Software Maintenance (ICSM), pp. 600–600. https://doi.org/10.1109/ICSM.2011.6080841

Dinh, H.T., Lee, C., Niyato, D., Wang, P., 2013. A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput. 13, 1587–1611. https://doi.org/10.1002/wcm.1203

Dixon, D., Kiani, S.L., Ikram, A., 2013. Experiences with AR plots: design issues and recommendations for augmented reality based mobile games. Commun. Mob. Comput. 2, 1–6. https://doi.org/10.1186/2192-1121-2-1

Fakoor, R., Raj, M., Nazi, A., Di Francesco, M., Das, S.K., 2012. An Integrated Cloud-based Framework for Mobile Phone Sensing, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12. ACM, New York, NY, USA, pp. 47–52. https://doi.org/10.1145/2342509.2342520

Fernando, N., Loke, S.W., Rahayu, W., 2013. Mobile cloud computing: A survey. Future Gener. Comput. Syst., Including Special section: AIRCC-

NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures 29, 84–106. https://doi.org/10.1016/j.future.2012.05.023

Ferreira, D., Dey, A.K., Kostakos, V., 2011. Understanding Human-Smartphone Concerns: A Study of Battery Life, in: Pervasive Computing. Presented at the 9th International Conference, Pervasive 2011, Springer Berlin Heidelberg, San Francisco, USA, pp. 19–33. https://doi.org/10.1007/978-3-642-21726-5_2

Flinn, J., Park, S., Satyanarayanan, M., 2002. Balancing performance, energy, and quality in pervasive computing, in: 22nd International Conference on Distributed Computing Systems, 2002. Proceedings. Presented at the 22nd International Conference on Distributed Computing Systems, 2002. Proceedings, pp. 217–226. https://doi.org/10.1109/ICDCS.2002.1022259

Fowler, S., Mozur, P., 2016. Samsung's Recall: The Problem With Lithium-Ion Batteries. N. Y. Times.

Glass, R.L., Ramesh, V., Vessey, I., 2004. An Analysis of Research in Computing Disciplines. Commun ACM 47, 89–94. https://doi.org/10.1145/990680.990686

Google, 2014a. About Google. Google. Available: https://www.google.co.za/intl/en/about/ (Accessed 27 November 2014).

Google, 2014b. Android. Android. Available: http://www.android.com (Accessed 13 November 2014).

Google, 2014c. Google Play. Google Play. Available: https://play.google.com/store (Accessed 25 November 2014).

Google, 2014d. What Is Google App Engine? - Google App Engine — Google Cloud Platform. Google. Available: https://cloud.google.com/appengine/docs/whatisgoogleappengine (Accessed 2 December 2014).

GPS.gov, 2008. Global Position System Standard Positioning Service Performance Standard.

Grønli, T.-M., Hansen, J., Ghinea, G., Younas, M., 2014. Mobile Application Platform Heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS, in: 2014 IEEE 28th International Conference on Advanced Information Networking and Applications (AINA). Presented at the 2014 IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), pp. 635–641. https://doi.org/10.1109/AINA.2014.78

GSam Labs, 2013. GSam Labs. GSam Labs. Available: http://www.gsamlabs.com (Accessed 10 August 2015).

GSMArena, 2016. Battery life tests - GSMArena.com. GSMArena. Available: http://www.gsmarena.com/battery-test.php3 (Accessed 22 November 2016).

Guan, L., Ke, X., Song, M., Song, J., 2011. A Survey of Research on Mobile Cloud Computing, in: 2011 IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS). Presented at the 2011 IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS), pp. 387–392. https://doi.org/10.1109/ICIS.2011.67

Halstead, M.H., 1977. Elements of software science. Elsevier.

Han, B., Hui, P., Kumar, V.S.A., Marathe, M.V., Shao, J., Srinivasan, A., 2012. Mobile Data Offloading through Opportunistic Communications and Social Participation. IEEE Trans. Mob. Comput. 11, 821–834. https://doi.org/10.1109/TMC.2011.101

Helmenstine, T., 2017. Calculate Percent Error. Sci. Notes Proj.

Henry, S., Kafura, D., 1981. Software Structure Metrics Based on Information Flow. IEEE Trans. Softw. Eng. SE-7, 510–518. https://doi.org/10.1109/TSE.1981.231113

Heroku, 2017a. About Heroku | Heroku. Available: https://www.heroku.com/about (Accessed 8 December 2017).

Heroku, 2017b. Dyno Types | Heroku Dev Center. Available: https://devcenter.heroku.com/articles/dyno-types (Accessed 8 December 2017).

HTC, 2016. HTC 10 Industry Leading Features and Specs | HTC United States. HTC. Available: http://www.htc.com/us/smartphones/htc-10/ (Accessed 20 November 2016).

Huang, D., Zhang, X., Kang, M., Luo, J., 2010. MobiCloud: Building Secure Cloud Framework for Mobile Computing and Communication, in: 2010 Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE). Presented at the 2010 Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), pp. 27–34. https://doi.org/10.1109/SOSE.2010.20

Huang, W., Mow, W.H., 2013. PiCode: 2D Barcode with Embedded Picture and ViCode: 3D Barcode with Embedded Video, in: Proceedings of the 19th Annual International Conference on Mobile Computing & Networking, MobiCom '13. ACM, New York, NY, USA, pp. 139–142. https://doi.org/10.1145/2500423.2505295

Huerta-Canepa, G., Lee, D., 2010. A Virtual Cloud Computing Provider for Mobile Devices, in: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS '10. ACM, New York, NY, USA, p. 6:1–6:5. https://doi.org/10.1145/1810931.1810937

Humphreys, L., Von Pape, T., Karnowski, V., 2013. Evolving Mobile Media: Uses and Conceptualizations of the Mobile Internet. J. Comput.-Mediat. Commun. 18, 491–507. https://doi.org/10.1111/jcc4.12019

Hung, S.-H., Shih, C.-S., Shieh, J.-P., Lee, C.-P., Huang, Y.-H., 2012. Executing mobile applications on the cloud: Framework and issues. Comput. Math. Appl. 63, 573–587. https://doi.org/10.1016/j.camwa.2011.10.044

IDC, 2014. IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011. IDC. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp (Accessed 25 November 2014).

IEEE, 2016. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 80211-2016 Revis. IEEE Std 80211-2012 1–3534. https://doi.org/10.1109/IEEESTD.2016.7786995

Imai, S., 2012. Task offloading between smartphones and distributed computational resources. Rensselaer Polytechnic Institute.

ITU, 2014. Radiocommunication Sector (ITU-R) - IMT-Advanced submission and evaluation process. ITU. Available: http://www.itu.int/ITU-R/index.asp?category=study-groups&rlink=rsg5-imt-advanced&lang=en (Accessed 25 November 2014).

ITU, 2011. About mobile technology and IMT-2000. ITU. Available: http://www.itu.int/osg/spu/imt-2000/technology.html#Cellular Standards for the Third Generation (Accessed 25 November 2014).

Jagtap, V.S., Pawar, K.V., Pathak, A.R., 2014. Augmented Execution in Mobile Cloud Computing: A Survey, in: 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC). Presented at the 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC), pp. 237–244. https://doi.org/10.1109/ICESC.2014.46

JavaWorld, 2003. Sizeof for Java. JavaWorld. Available: http://www.javaworld.com/article/2077408/core-java/sizeof-for-java.html (Accessed 5 February 2015).

JEDEC, 2014a. About JEDEC | JEDEC. JEDEC. Available: http://www.jedec.org/about-jedec (Accessed 28 November 2014).

JEDEC, 2014b. JESD209-4 Low Power Double Data Rate 4 (LPDDR4) (No. JESD209-4). JEDEC.

JEDEC, 2012. JESD209-3 Low Power Memory Device Standard (No. JESD209-3). JEDEC.

Johnsson, A., Bjorkman, M., 2008. On measuring available bandwidth in wireless networks, in: 33rd IEEE Conference on Local Computer Networks, 2008. LCN 2008. Presented at the 33rd IEEE Conference on Local Computer Networks, 2008. LCN 2008, pp. 861–868. https://doi.org/10.1109/LCN.2008.4664295

Johnsson, A., Melander, B., Björkman, M., 2006. Bandwidth Measurement in Wireless Networks, in: Agha, K.A., Lassous, I.G., Pujolle, G. (Eds.), Challenges in Ad Hoc Networking, IFIP International Federation for Information Processing. Springer US, pp. 89–98.

Kahneman, D., Tversky, A., 2000. Choices, Values, and Frames. Cambridge University Press.

KAIBITS, 2015. KAIBITS Software. KAIBITS. Available: http://www.kaibits-software.com/ (Accessed 10 August 2015).

Kalic, G., Bojic, I., Kusek, M., 2012. Energy consumption in android phones when using wireless communication technologies, in: 2012 Proceedings of the 35th International Convention MIPRO. Presented at the 2012 Proceedings of the 35th International Convention MIPRO, pp. 754–759.

Kamara, S., Lauter, K.E., others, 2010. Cryptographic Cloud Storage., in: Financial Cryptography Workshops. Springer, pp. 136–149.

Kayande, D., Shrawankar, U., 2012. Performance analysis for improved RAM utilization for Android applications, in: 2012 CSI Sixth International Conference on Software Engineering (CONSEG). Presented at the 2012 CSI Sixth International Conference on Software Engineering (CONSEG), pp. 1–6. https://doi.org/10.1109/CONSEG.2012.6349500

Kemp, R., Palmer, N., Kielmann, T., Bal, H., 2012. Cuckoo: A Computation Offloading Framework for Smartphones, in: Gris, M., Yang, G. (Eds.), Mobile Computing, Applications, and Services, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, pp. 59–79.

Kristensen, M.D., 2010. Scavenger: Transparent development of efficient cyber foraging applications, in: 2010 IEEE International Conference on Pervasive Computing and Communications (PerCom). Presented at the 2010 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 217–226. https://doi.org/10.1109/PERCOM.2010.5466972

Kumar, K., Liu, J., Lu, Y.-H., Bhargava, B., 2013. A Survey of Computation Offloading for Mobile Systems. Mob. Netw. Appl. 18, 129–140. https://doi.org/10.1007/s11036-012-0368-0

Kumar, K., Lu, Y.-H., 2010. Cloud Computing for Mobile Users: Can Offloading Computation Save Energy? Computer 43, 51–56. https://doi.org/10.1109/MC.2010.98

Liang, R., Zhong, Y., Xia, Q., 2018. Energy-saved data transfer model for mobile devices in cloudlet computing environment, in: 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). Presented at the 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 271–274. https://doi.org/10.1109/ICCCBDA.2018.8386525

Linux, 2012. The source for Linux information. Linux. Available: http://www.linux.com/ (Accessed 25 November 2014).

Longo, F., Ghosh, R., Naik, V.K., Trivedi, K.S., 2011. A scalable availability model for Infrastructure-as-a-Service cloud, in: 2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN). Presented at the 2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN), pp. 335–346. https://doi.org/10.1109/DSN.2011.5958247

Ma, J., Deng, X., Liu, Y., Wu, D., 2013. Power consumption of mobile video streaming under adverse network conditions, in: 2013 IEEE/CIC International Conference on Communications in China (ICCC). Presented at the 2013 IEEE/CIC International Conference on Communications in China (ICCC), pp. 106–111. https://doi.org/10.1109/ICCChina.2013.6671098

Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. Presented at the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pp. 496–503. https://doi.org/10.1109/CloudCom.2010.78

Marinelli, E.E., 2009. Hyrax: Cloud Computing on Mobile Devices using MapReduce. Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Mell, P., Grance, T., 2009. The NIST definition of cloud computing. Natl. Inst. Stand. Technol. 53, 50.

Microsoft, 2016a. Microsoft Lumia 650 Specifications - Microsoft - Global. Microsoft. Available: https://www.microsoft.com/en/mobile/phone/lumia650/specifications/ (Accessed 20 November 2016).

Microsoft, 2016b. .NET - Powerful Open Source Cross Platform Development. Microsoft. Available: https://www.microsoft.com/net (Accessed 22 November 2016).

Microsoft, 2014a. Windows Phone. Microsoft. Available: http://www.windowsphone.com/en-us/features-8-1 (Accessed 14 November 2014).

Microsoft, 2014b. C# Programming Guide. Microsoft. Available: http://msdn.microsoft.com/en-us/library/67ef8sbd.aspx (Accessed 25 November 2014).

Microsoft, 2014c. Windows Phone For AT&T, T-Mobile, Sprint, & Verizon - Microsoft Store. Microsoft. Available: http://www.microsoftstore.com/store/msusa/en_US/cat/Windows-Phone/categoryID.62685000 (Accessed 25 November 2014).

Microsoft, 2014d. What is Azure? Azure. Available: http://azure.microsoft.com/en-us/overview/what-is-azure/ (Accessed 2 December 2014).

Microsoft, 2014e. Microsoft OneDrive. Microsoft. Available: https://onedrive.live.com/about/en-us/ (Accessed 2 December 2014).

Minihold, R., 2011. Near Field Communication (NFC) Technology and Measurements White Paper.

Moon, B., 2014. 5 Small Smartphones That Pack a Big Punch. InvestorPlace.

MTN, 2017. About Us | MTN Group | MTN. MTN. Available: https://www.mtn.co.za/About_us/Pages/Overview.aspx (Accessed 19 March 2017).

Oberheide, J., Cooke, E., Jahanian, F., 2007. Rethinking Antivirus: Executable Analysis in the Network Cloud, in: 6th USENIX Security Symposium. Presented at the HotSec '07, Boston, MA, U.S.A.

Olivier, M., 2009. Information Technology Research - A Practical Guide for Computer Science and Informatics, 2nd Edition. ed. Van Schaik.

Oracle Corporation, 2014. What is Java? Oracle Corp. Available: https://www.java.com/en/download/whatis_java.jsp (Accessed 25 November 2014).

Page, C., 2014. Android hits 83.6 percent marketshare while iOS, Windows and BlackBerry slide. The Inquirer.

Pan, S., Chen, A., Zhang, P., 2013. Securitas: User Identification Through RGB-NIR Camera Pair on Mobile Devices, in: Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, SPSM '13. ACM, New York, NY, USA, pp. 99–104. https://doi.org/10.1145/2516760.2516766

Parkkila, J., Porras, J., 2011. Improving battery life and performance of mobile devices with cyber foraging, in: 2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC). Presented at the 2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), pp. 91–95. https://doi.org/10.1109/PIMRC.2011.6140102

Parkvall, S., Astely, D., 2009. The Evolution of LTE towards IMT-Advanced. J. Commun. 4, 146–154. https://doi.org/10.4304/jcm.4.3.146-154

Radicati, S., 2014. Mobile Statistics Report, 2014-2018 (Executive Summary). The Radicati Group, Inc., Palo Alto, CA, USA.

Reed, B., 2014. Apple might want to start taking gripes about the iPhone's battery life seriously. BGR.

Reza, A., 2015. "Memory Need" Gives Birth To "New Memory."

Rudenko, A., Reiher, P., Popek, G.J., Kuenning, G.H., 1998. Saving Portable Computer Battery Power Through Remote Process Execution. SIGMOBILE Mob Comput Commun Rev 2, 19–26. https://doi.org/10.1145/584007.584008

Sale, A.H.J., 1977. Primitive data types. Aust. Compuer J. 9, 63–71.

Samsung, 2016a. Galaxy S7 Feature and Specs. Samsung Electron. Am. Available: http://www.samsung.com/us/explore/galaxy-s7-features-and-specs/ (Accessed 20 November 2016).

Samsung, 2016b. Samsung Galaxy S6 edge. Off. Samsung Galaxy Site. Available: http://www.samsung.com/global/galaxy/galaxys6/galaxy-s6-edge/ (Accessed 22 November 2016).

Samsung, 2014a. S Health. Samsung. Available: http://content.samsung.com/za/contents/aboutn/sHealthIntro.do (Accessed 28 November 2014).

Samsung, 2014b. Samsung Galaxy S5 LTE. Samsung. Available: http://www.samsung.com/za/consumer/mobile-phone/smart-phone/smart-phone/SM-G900FZBAXFV (Accessed 18 November 2014).

SAP, 2017a. Overview | SAP Cloud Platform. SAP. Available: https://cloudplatform.sap.com/index.html (Accessed 19 March 2017).

SAP, 2017b. SAP Cloud Platform Intelligently Connects People, Things and Businesses. SAP News Cent. Available: http://news.sap.com/mwc-sap-cloud-platform-intelligently-connects-people-things-businesses/ (Accessed 19 March 2017).

Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N., 2011. The Case for VM-based Cloudlets in Mobile Computing. IEEE Pervasive Comput. Early Access Online. https://doi.org/10.1109/MPRV.2009.64

Schaffer, H.E., 2009. X as a Service, Cloud Computing, and the Need for Good Judgment. IT Prof. 11, 4–5. https://doi.org/10.1109/MITP.2009.112

SD Assocation, 2014. SD Standard Overview. SD Assocation. Available: https://www.sdcard.org/developers/overview/ (Accessed 18 November 2014).

Singh, R., Kumar, S., Agrahari, S.K., 2012. Ensuring data storage security in cloud computing. IOSR J. Eng. 2, 17–21.

Smailagic, A., Ettus, M., 2002. System design and power optimization for mobile computers, in: IEEE Computer Society Annual Symposium on VLSI, 2002. Proceedings. Presented at the IEEE Computer Society Annual Symposium on VLSI, 2002. Proceedings, pp. 10–14. https://doi.org/10.1109/ISVLSI.2002.1016867

Sokol, A.W., Hogan, M.D., 2013. NIST Cloud Computing Standards Roadmap. Spec. Publ. NIST SP - 500-291r2. https://doi.org/913661

Souppaya, M., Scarfone, S., 2013. Guidelines for Managing the Security of Mobile Devices in the Enterprise (No. SP 800-124 r1). National Institute of Standards and Technology (NIST).

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., Heinzelman, W., 2012. Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture, in: 2012 IEEE Symposium on Computers and Communications (ISCC). Presented at the 2012 IEEE Symposium on Computers and Communications (ISCC), pp. 000059–000066. https://doi.org/10.1109/ISCC.2012.6249269

StackOverflow, 2014. how to measure upload/download speed and latency in android wifi connection - Stack Overflow. StackOverflow. Available: http://stackoverflow.com/questions/5193518/how-to-measure-upload-download-speed-and-latency-in-android-wifi-connection (Accessed 5 February 2015).

StackOverflow, 2010. Measure execution time for a Java method - Stack Overflow. StackOverflow. Available: http://stackoverflow.com/questions/3382954/measure-execution-time-for-a-java-method (Accessed 6 February 2015).

Statista, 2017. App stores: number of apps in leading app stores 2017. Statista. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/ (Accessed 3 November 2017).

Stephens, R., 2013. Essential Algorithms: A Practical Approach to Computer Algorithms. John Wiley & Sons.

Telkom, 2017. About Us | Telkom. Telkom. Available: http://www.telkom.co.za/about_us/index.shtml (Accessed 19 March 2017).

Toma, A., Starinow, A., Lenssen, J.E., Chen, J.J., 2018. Saving Energy for Cloud Applications in Mobile Devices Using Nearby Resources, in: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). Presented at the 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 541–545. https://doi.org/10.1109/PDP2018.2018.00091

Triggs, R., 2015. LPDDR4 – everything you need to know | AndroidAuthority. AndroidAuthority. Available: http://www.androidauthority.com/lpddr4-everything-need-know-599759/ (Accessed 23 January 2017).

Tsai, W.-T., Sun, X., Balasooriya, J., 2010. Service-Oriented Cloud Computing Architecture, in: 2010 Seventh International Conference on Information Technology: New Generations (ITNG). Presented at the 2010 Seventh International Conference on Information Technology: New Generations (ITNG), pp. 684–689. https://doi.org/10.1109/ITNG.2010.214

Wang, G., Xiong, Y., Yun, J., Cavallaro, J.R., 2014. Computer Vision Accelerators for Mobile Systems based on OpenCL GPGPU Co-

Processing. J. Signal Process. Syst. 76, 283–299. https://doi.org/10.1007/s11265-014-0878-z

Wang, H., Xie, J., Liu, X., 2018. Rethinking Mobile Devices' Energy Efficiency in WLAN Management Services, in: 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). Presented at the 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1–9. https://doi.org/10.1109/SAHCN.2018.8397137

Wang, L., Laszewski, G. von, Younge, A., He, X., Kunze, M., Tao, J., Fu, C., 2010. Cloud Computing: a Perspective Study. New Gener. Comput. 28, 137–146. https://doi.org/10.1007/s00354-008-0081-5

Want, R., Schilit, B., Laskowski, D., 2013. Bluetooth LE Finds Its Niche. IEEE Pervasive Comput. 12, 12–16. https://doi.org/10.1109/MPRV.2013.60

Whitwarm, R., 2014. How Samsung Galaxy S5′s Ultra Power Saving Mode makes 10% battery last 24 hours. ExtremeTech. Available: http://www.extremetech.com/mobile/180110-how-samsung-galaxy-s5s-ultra-power-saving-mode-makes-10-last-24-hours (Accessed 5 May 2014).

Wilcox, M., Voskoglou, C., 2014. Developer Economics Q3 2014: State of the Developer Nation. VisionMobile.

Wright, G.R., Stevens, W.R., 1995. TCP/IP Illustrated. Addison-Wesley Professional.

Xiao, Y., Hui, P., Savolainen, P., Ylä-Jääski, A., 2011. CasCap: Cloud-assisted Context-aware Power Management for Mobile Devices, in: Proceedings of the Second International Workshop on Mobile Cloud Computing and Services, MCS '11. ACM, New York, NY, USA, pp. 13–18. https://doi.org/10.1145/1999732.1999736

Xiaolu, Y., Ma, E.W.M., Pecht, M., 2012. Cell balancing technology in battery packs, in: 2012 13th International Conference on Electronic Packaging Technology and High Density Packaging (ICEPT-HDP). Presented at the 2012 13th International Conference on Electronic Packaging Technology and High Density Packaging (ICEPT-HDP), pp. 1038–1041. https://doi.org/10.1109/ICEPT-HDP.2012.6474785

Yang, X., Pan, T., Shen, J., 2010. On 3G mobile E-commerce platform based on Cloud Computing, in: 2010 3rd IEEE International Conference on Ubi-Media Computing (U-Media). Presented at the 2010 3rd IEEE International Conference on Ubi-media Computing (U-Media), pp. 198–201. https://doi.org/10.1109/UMEDIA.2010.5544470

Yousafzai, A., Gani, A., Md Noor, R., Naveed, A., Ahmad, R.W., Chang, V., 2016. Computational offloading mechanism for native and android runtime based mobile applications. J. Syst. Softw. 121, 28–39. https://doi.org/10.1016/j.jss.2016.07.043

Yu, S., Zhou, S., 2010. A survey on metric of software complexity, in: 2010 The 2nd IEEE International Conference on Information Management and Engineering (ICIME). Presented at the 2010 The 2nd IEEE International Conference on Information Management and Engineering (ICIME), pp. 352–356. https://doi.org/10.1109/ICIME.2010.5477581

Zhang, S., Zhang, Shufen, Chen, X., Huo, X., 2010. Cloud Computing Research and Development Trend, in: Second International Conference on Future Networks, 2010. ICFN '10. Presented at the Second International Conference on Future Networks, 2010. ICFN '10, pp. 93–97. https://doi.org/10.1109/ICFN.2010.58

Zhang, X., Jeong, S., Kunjithapatham, A., Gibbs, S., 2010. Towards an Elastic Application Model for Augmenting Computing Capabilities of Mobile Platforms, in: Cai, Y., Magedanz, T., Li, M., Xia, J., Giannelli, C. (Eds.), Mobile Wireless Middleware, Operating Systems, and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, pp. 161–174.

Zhang, Y., Ansari, N., Wu, M., Yu, H., 2012. AFStart: An adaptive fast TCP slow start for wide area networks, in: 2012 IEEE International Conference on Communications (ICC). Presented at the 2012 IEEE International Conference on Communications (ICC), pp. 1260–1264. https://doi.org/10.1109/ICC.2012.6363716

Zhou, M., Zhang, R., Zeng, D., Qian, W., 2010. Services in the Cloud Computing era: A survey, in: Universal Communication Symposium (IUCS), 2010 4th International. Presented at the Universal Communication Symposium (IUCS), 2010 4th International, pp. 40–46. https://doi.org/10.1109/IUCS.2010.5666772

Zsambok, C.E., Klein, G., 2014. Naturalistic Decision Making. Psychology Press.

Appendix A: Real-world energy consumption data

This appendix contains the data gathered and used in chapter 10. Each table contains the results of either uploading or download a file that is 100KB, 1MB or 10MB in size at different signal strengths over different networks. The different networks are 3G, 4G and Wi-Fi. The appendix is divided into 4 sections, namely 3G, 4G, Wi-Fi, and Computation.

A.1. 3G

Table A.1 shows the averages of downloading a 100 KB file five times over 3G at the listed signal strengths.

Table A.1 Downloading a 100 KB file at differing signal strengths over 3G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
113586	6336,2	4,5854	-109	0,046
112755,6	6422	4,0348	-107	0,04
110777,4	6141,6	3,4922	-105	0,04
110943,8	6157,2	3,326	-103	0,036
110953,2	5973,2	3,1232	-101	0,034
110735,8	2,893	2,9044	-99	0,03
110735,8	2,893	2,6878	-97	0,03
108576	3256,2	2,487	-95	0,03
110767	5866	2,2854	-93	0,03
111772,2	5408,6	2,0534	-91	0,03
110798,2	5771,6	1,7	-89	0,03
110777,4	5633,2	1,4256	-87	0,03
110839,8	5930	1,381	-85	0,03
110756,6	5759,6	1,2134	-83	0,028

Table A.2 shows the averages of downloading a 1 MB file five times over 3G at the listed signal strengths.

Table A.2 Downloading a 1 MB file at differing signal strengths over 3G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
1151687,2	24341,4	8,621	-109	0,074
1095458,2	22898,6	8,0312	-107	0,066
1095075,6	22721,4	7,5674	-105	0,064
1095109	22339,6	6,6692	-103	0,06
1094606,8	22359,6	6,2522	-101	0,06
1094409,4	22563,4	5,8968	-99	0,06
1095219,8	23494,6	5,6866	-97	0,056
1094604,2	22615,4	5,475	-95	0,052
1094085,4	21965,8	5,697	-93	0,052
1094286,4	22125,2	5,571	-91	0,05
1094273,4	22166,8	5,4596	-89	0,05
1094294,8	22321,8	5,2438	-87	0,05
1094296,4	22718,6	4,6392	-85	0,042
1094184,4	21763,4	4,3496	-83	0,04

Table A.3 shows the averages of downloading a 10 MB file five times over 3G at the listed signal strengths.

Table A.3 Downloading a 10 MB file at differing signal strengths over 3G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
11259890,4	568355,078	86,4934	-109	0,344
11251983,2	558212	82,1978	-107	0,2666
11246931,4	547895,4	78,2712	-105	0,326
11250887,6	567491,8	74,3442	-103	0,316
11253099	566594,4	71,549	-101	0,304

11254983,8	566118	68,9362	-99	0,3
11250560,2	556423	66,142	-97	0,29
11253254,8	564541,8	63,8092	-95	0,282
11250604,4	574715,8	60,534	-93	0,28
11249730	528646,8	57,5236	-91	0,276
11249249,4	571794,8	54,694	-89	0,27
11247522,8	569023,6	51,8506	-87	0,26
11246010,2	561508,4	47,5784	-85	0,25
11247039,8	574509,4	44,4142	-83	0,242

Table A.4 shows the averages of uploading a 100 KB file five times over 3G at the listed signal strengths.

Table A.4 Uploading a 100 KB file at differing signal strengths over 3G

Download size (bytes)	**Upload size (bytes)**	**Length of time (seconds)**	**Signal Strength (dBm)**	**Battery life consumed (%)**
7146,4	109469,4	5,9658	-109	0,05
7177,6	109450	5,7558	-107	0,048
7583,2	109477	5,5452	-105	0,044
7148,8	109373	5,1242	-103	0,04
7117,6	109465	4,8738	-101	0,04
7073,6	109397	4,557	-99	0,03
7167,2	109450	4,2022	-97	0,03
7084	109537,8	3,847	-95	0,03
7136	109438	3,5516	-93	0,03
7042,4	109442	3,2556	-91	0,03
7167,2	109450	2,996	-89	0,03
7042,4	109361	2,7356	-87	0,03
7032	109363,4	2,4886	-85	0,02
7312,8	109430,8	2,1944	-83	0,02

Table A.5 shows the averages of uploading a 1 MB file five times over 3G at the listed signal strengths.

Table A.5 Uploading a 1 MB file at differing signal strengths over 3G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
26329,2	1104551,4	13,2722	-109	0,09
26189	1105474,5	12,8525	-107	0,09
26281	1105535	12,2852	-105	0,09
25846,6	1105415,4	11,8058	-103	0,09
25701	1105465,2	11,3168	-101	0,08
25515,4	1105478	10,8272	-99	0,08
25701	1105709	10,2844	-97	0,08
25586,6	1106077,4	9,7414	-95	0,08
25378,6	1105425	9,1984	-93	0,08
25317,8	1105427,4	8,655	-91	0,072
25586,6	1105802,2	8,1992	-89	0,07
25781,8	1105473,8	7,6514	-87	0,07
25430,6	1105529,4	7,208	-85	0,06
25482,6	1105440	6,8298	-83	0,06

Table A.6 shows the averages of uploading a 10 MB file five times over 3G at the listed signal strengths.

Table A.6 Uploading a 10 MB file at differing signal strengths over 3G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
209599,4	10863542,8	87,36	-109	0,342
206657,8	10772445	83,008	-107	0,336
210445	10882590,2	79,6002	-105	0,328
203296	10594173,6	76,0798	-103	0,318
210189,8	10883938,6	72,5594	-101	0,308
203017	10571106	69,2838	-99	0,298
200337,8	10531352,8	66,0078	-97	0,288

209763,4	10871322	62,7604	-95	0,28
207787,4	10752166,4	59,5904	-93	0,276
203633,8	10612600	56,5906	-91	0,266
209748,2	10932941,6	53,7952	-89	0,266
200157,8	10457278,4	50,8444	-87	0,26
207313,8	10825073,8	47,8932	-85	0,248
208122,6	10838406	43,9966	-83	0,242

A.2. 4G

Table A.7 shows the averages of downloading a 100 KB file five times over 4G at the listed signal strengths.

Table A.7 Downloading a 100 KB file at differing signal strengths over 4G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
110798,2	5228,4	0,7498	-117	0,03
110881,4	5386,8	0,6188	-115	0,03
110829,4	5548,4	0,5916	-113	0,03
110943,8	5563,6	0,5684	-111	0,03
110819	5226	0,532	-109	0,03
110746,2	5327,6	0,5176	-107	0,03
110881,4	5236,4	0,5074	-105	0,03
110829,4	5199,6	0,5036	-103	0,03
110902,2	5216,4	0,4994	-101	0,022
110746,2	5130	0,4948	-99	0,02
110850,2	5542,8	0,4896	-97	0,02
110767	5054,8	0,4858	-95	0,02
110798,2	5174	0,4816	-93	0,02
110683,8	4966	0,4776	-91	0,02
111709,8	4718,6	0,473	-89	0,02
110860,6	4832,4	0,4602	-87	0,02

Table A.8 shows the averages of downloading a 1 MB file five times over 4G at the listed signal strengths.

Table A.8 Downloading a 1 MB file at differing signal strengths over 4G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
1094986,6	24437,8	6,6614	-117	0,066
1126085	48722,8	6,4714	-115	0,064
1097711,2	23280,6	6,2812	-113	0,06
1125177,4	48812	6,0914	-111	0,06
1079054,2	23920,2	5,9008	-109	0,06
1126284,2	50048	5,7606	-107	0,06
1096759,2	23364,2	5,6196	-105	0,06
1126322,6	50604,8	5,4908	-103	0,054
1094458	22163	5,3614	-101	0,05
1095091,2	22606,2	5,1692	-99	0,05
1095100,2	22077,8	4,9762	-97	0,05
1094959,2	22077,8	4,7328	-95	0,05
1094743,6	22808,2	4,489	-93	0,046
1094388,4	22225,8	4,2956	-91	0,04
1094600,4	22481	4,1418	-89	0,04
1094510	22223,4	3,942	-87	0,04

Table A.9 shows the averages of downloading a 10 MB file five times over 4G at the listed signal strengths.

Table A.9 Downloading a 10 MB file at differing signal strengths over 4G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
11246435,4	521323,2	54,6284	-117	0,34
11249471,2	502325,8	52,5936	-115	0,326
11245909	484040,2	50,526	-113	0,324

11251990,2	572539	49,4844	-111	0,314
11248859,6	502855,8	48,442	-109	0,312
11246244,4	505032,6	47,6622	-107	0,3
11245313,6	498445,2	46,808	-105	0,3
11243960,8	487614	45,4728	-103	0,286
11250818,4	516791,6	44,1776	-101	0,282
11243416	498219,4	43,241	-99	0,276
11243953,4	500960,8	42,5732	-97	0,266
11244812,2	502604,4	41,5138	-95	0,264
11244354,8	501160,4	40,5716	-93	0,26
11244090,8	498405,8	39,49868	-91	0,256
11243377,4	509142,6	38,5354	-89	0,254
11243379,4	514120,2	37,4826	-87	0,248

Table A.10 shows the averages of uploading a 100 KB file five times over 4G at the listed signal strengths.

Table A.10 Uploading a 100 KB file at differing signal strengths over 4G

Download size (bytes)	**Upload size (bytes)**	**Length of time (seconds)**	**Signal Strength (dBm)**	**Battery life consumed (%)**
7125,6	109411	4,4554	-117	0,04
7115,2	109365,8	4,3668	-115	0,04
7253,6	117616,2	4,2524	-113	0,04
7175,2	112886	4,1374	-111	0,04
7229,6	113623,4	4,0092	-109	0,04
7084	109438	3,8802	-107	0,04
7042,4	109403,4	3,7588	-105	0,04
7052,8	109363,4	3,637	-103	0,04
7042,4	109365,8	3,5302	-101	0,034
7146,4	109361	3,386	-99	0,03
7042,4	109353	3,1858	-97	0,03
7032	109365,8	2,9968	-95	0,03
7032	109361	2,7682	-93	0,03

7032	109361	2,633	-91	0,03
7032	109365,8	2,549	-89	0,03
7073,6	109416,4	2,4086	-87	0,03

Table A.11 shows the averages of uploading a 1 MB file five times over 4G at the listed signal strengths.

Table A.11 Uploading a 1 MB file at differing signal strengths over 4G

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
25285	1105470	8,8188	-117	0,08
25285	1105445	8,436	-115	0,07
25285	1105448	8,255	-113	0,07
25285	1105416	7,954	-111	0,07
25285	1105421	7,7924	-109	0,07
25300,2	1105389	7,6306	-107	0,07
25295,4	1105389	7,376	-105	0,07
25295,4	1105445	7,122	-103	0,07
25285	1105470	6,8072	-101	0,06
25295,4	1105470	6,6686	-99	0,06
25285	1105443	6,443	-97	0,06
25305,8	1105416	6,2772	-95	0,06
25285	1105416	6,1548	-93	0,06
25305,8	1105391	6,0652	-91	0,06
25295,4	1105421	5,9718	-89	0,06
25287,4	1105445	5,8422	-87	0,06

Table A.12 shows the averages of uploading a 10 MB file five times over 4G at the listed signal strengths.

Table A.12 Uploading a 10 MB file at differing signal strengths over 4G

Download size (bytes)	Upload size (bytes)	Length of time	Signal Strength	Battery life consumed

		(seconds)	(dBm)	(%)
201510,6	10471336	59,0594	-117	0,35
200762,6	10465785	55,4008	-115	0,338
199177	10364064	52,293	-113	0,326
199341	10348835	50,4806	-111	0,314
199472,2	10377540	49,4344	-109	0,31
203216,2	10564234	48,4624	-107	0,306
200951,8	10494400	47,5264	-105	0,3
204889,8	10619118	46,5952	-103	0,298
200816,2	10529033	45,802	-101	0,29
204594,6	10645834	44,6222	-99	0,28
201983,4	10461478	43,5598	-97	0,278
202041	10510461	42,5236	-95	0,27
202637,8	10520769	41,6988	-93	0,268
205294,6	10578951	40,7294	-91	0,26
202706,6	10435655	39,5752	-89	0,26
196701	10272467	38,5584	-87	0,256

A.3. Wi-Fi

Table A.13 shows the averages of downloading a 100 KB file five times over Wi-Fi at the listed signal strengths.

Table A.13 Downloading a 100 KB file at differing signal strengths over Wi-Fi

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
112088,4	6767,6	11,5288	-89	0,05
112114,2	6098,6	9,8606	-87	0,05
112378,4	6487,2	9,2762	-85	0,05
110943,2	6512,4	8,6912	-83	0,05
111027,2	5933,2	8,3962	-81	0,05
110873	4809,2	8,0908	-79	0,05
110547,4	4518	7,5672	-77	0,05

110821	5304,4	7,0432	-75	0,042
110740,4	5008,4	6,843	-73	0,04
110650,6	4666	6,401	-71	0,04
111214,6	5082,8	6,0798	-69	0,04
110860,8	5482,8	5,7582	-67	0,04
111111,6	6245,6	5,1164	-65	0,04
110645,4	5163,6	4,8404	-63	0,04
110957,4	5407,6	4,5638	-61	0,038
110926,2	5482,8	4,2306	-59	0,03
110853,4	4931,6	3,8972	-57	0,03
110525,6	4894	3,4706	-55	0,024
111305,2	5389,6	2,8232	-53	0,02
111198,6	5547,6	2,4228	-51	0,014
110600,6	4938,8	1,9942	-49	0,01
110548,6	5053,2	1,7236	-47	0,01
110642,2	5060,4	1,4448	-45	0,01
107544,8	3147,4	1,2764	-43	0,01

Table A.14 shows the averages of downloading a 1 MB file five times over Wi-Fi at the listed signal strengths.

Table A.14 Downloading a 1 MB file at differing signal strengths over Wi-Fi

Download size (bytes)	**Upload size (bytes)**	**Length of time (seconds)**	**Signal Strength (dBm)**	**Battery life consumed (%)**
1134633	48723	51,2684	-89	0,132
1134720,8	48113	48,4296	-87	0,13
1126345,2	41969,6	45,0336	-85	0,13
1128914,8	46413,6	42,6316	-83	0,12
1126351,4	41923	39,6522	-81	0,12
1130019,8	47146,8	36,6726	-79	0,11
1125910,6	42125,6	34,495	-77	0,11

1128193,2	45880,4	32,317	-75	0,1
1125894,6	43044,8	30,836	-73	0,1
1126228,2	46106,4	29,3532	-71	0,1
1131896,2	45632,2	26,2856	-69	0,09
1125596,6	43558,4	24,2022	-67	0,08
1126554,8	45391,2	23,3164	-65	0,076
1127054,6	43096	20,1876	-63	0,068
1125668,2	42893,6	18,9346	-61	0,06
1128451,4	44663,6	17,6772	-59	0,06
1125975,8	43621,6	15,3774	-57	0,06
1127161	43484,6	13,6846	-55	0,058
1125130,6	41281,6	12,313	-53	0,05
1125779	42752	10,5538	-51	0,05
1125470,8	41855,2	8,6028	-49	0,05
1125084,8	40770,4	7,4148	-47	0,042
1125341,4	41275,2	6,5798	-45	0,04
1124951,8	40327,2	5,7038	-43	0,04

Table A.15 shows the averages of downloading a 10 MB file five times over Wi-Fi at the listed signal strengths.

Table A.15 Downloading a 10 MB file at differing signal strengths over Wi-Fi

Download size (bytes)	**Upload size (bytes)**	**Length of time (seconds)**	**Signal Strength (dBm)**	**Battery life consumed (%)**
11255460,4	405210,8	84,3942	-89	0,23
11265216	410017	82,6528	-87	0,22
11264742,2	422079,8	82,2888	-85	0,22
11274463,2	408708,6	81,9246	-83	0,22
11263927,6	414553	81,1524	-81	0,222
11277097,6	410928,6	80,3798	-79	0,22
11243476,6	405368,2	79,3584	-77	0,214
11245876,4	402410	78,0886	-75	0,21
11244205,6	403266	76,0036	-73	0,202

11247193,4	399644,4	73,4834	-71	0,194
11245444,2	402324,6	72,2574	-69	0,19
11245071,2	403136,4	70,5538	-67	0,18
11247066,4	402787,6	69,2806	-65	0,18
11249186,6	401762,2	67,9108	-63	0,178
11247341,8	401909,4	66,5866	-61	0,17
11243962	398658	65,3148	-59	0,17
11243439,6	400203,4	63,6594	-57	0,16
11242077,2	398474	60,737	-55	0,16
11242146,8	395993	58,9942	-53	0,152
11241255,2	393752,2	57,8156	-51	0,15
11241640,2	393705,8	56,7202	-49	0,15
11242702,6	393246,8	55,3794	-47	0,15
11241172,6	393603,4	53,5682	-45	0,14
11246843,8	400070,8	51,8166	-43	0,136

Table A.16 shows the averages of uploading a 100 KB file five times over Wi-Fi at the listed signal strengths.

Table A.16 Uploading a 100 KB file at differing signal strengths over Wi-Fi

Download size (bytes)	Upload size (bytes)	Length of time (seconds)	Signal Strength (dBm)	Battery life consumed (%)
9188,8	138591,4	9,806	-89	0,02
8900,8	140516	9,5168	-87	0,02
8221,2	130624,4	9,2244	-85	0,02
8297,6	128299,6	8,827	-83	0,02
8309,6	130089,8	8,6144	-81	0,02
7468,8	115084	8,4964	-79	0,016
8092	126867,6	8,3922	-77	0,01
7740,8	115914,6	8,2896	-75	0,01
8603,2	133721,8	8,0934	-73	0,01
8237,6	127772,6	7,9212	-71	0,01

8888,8	139886,4	7,8446	-69	0,01
8914,4	141812,2	7,7512	-67	0,01
8652,8	134141	7,2574	-65	0,01
7073,6	109626,2	6,6582	-63	0,01
7855,2	121336,6	6,1602	-61	0,01
7973,6	123641	5,913	-59	0,01
8290,4	126973,8	5,7586	-57	0,01
7890,4	122464	5,6866	-55	0,01
7885,6	121600	5,6326	-53	0,01
7042,4	109361	5,609	-51	0,01
7817,6	119898,4	5,5792	-49	0,01
7421,6	115609	5,524	-47	0,01
7032	109361	5,4656	-45	0,01
7441,6	115041	5,4002	-43	0,01

A.4. Computation

The results from the experiments when measuring the power consumption of the device when it is performing computations is discussed in this section. The percentage battery life consumed is compared to the length of time the computation lasted. The results shown are the averages of executing the experiment 3 times.

Table A.17 Results of computational experiments

Length of time (seconds)	Battery life consumed (%)
1	0,001333
5	0,003333
10	0,007
15	0,012
20	0,018
25	0,023
30	0,033
35	0,035
40	0,039
45	0,048333
50	0,052667
55	0,056
60	0,064333
65	0,066
70	0,075333
75	0,080333
80	0,084667
85	0,089
90	0,094