Empirical Evaluation of Mobile Applications and an Adaptive Approach to Computation Offloading

by

Kelie Zhan

B.Eng. (2010)

A Thesis submitted to the

Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Masters of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

May, 2014

The undersigned recommend to the Faculty of Graduate Studies and Research acceptance of the thesis

Empirical Evaluation of Mobile Applications and an Adaptive Approach to Computation Offloading

Submitted by

Kelie Zhan
B.Eng.

In partial fulfillment of the requirements for the degree of Masters of Applied Science

---

Thesis Supervisor, Professor Chung-Horng Lung

Chair of the Department of Systems and Computer Engineering

Carleton University

Abstract

With the widespread of rich mobile applications, the usage of mobile devices, especially smart phones and tablets, has become popular nowadays. However, mobile devices are limited in battery, CPU, memory and storage. These constraints prevent mobile devices from widely running all kinds of rich mobile applications. Computation offloading is believed to be a potential solution to the hardware limitations of mobile devices for energy saving and/or higher performance. To meet this goal, a number of experiments have been performed in this research to compare energy efficiency of mobile devices with different offloading targets, e.g., a nearby PC or a remote cloud server. Experiments have also been conducted to examine the impact of various factors and provide insights on power consumption for different mobile applications. Finally, an adaptive algorithm, based on experimental results, has been developed to automatically select the most suitable computation target which has enough capacity to execute computationally intensive applications for energy saving and achieve a desirable performance at the same time.

Acknowledgements

Firstly, I offer my sincerest gratitude to my supervisor, Professor Chung-Horng Lung, for his patience, encouragement, valuable guidance for my research. I attribute the level of my Masters degree to his encouragement and effort. Without his continuous support, I would not have been completed the thesis smoothly.

Secondly, I am very grateful for my friends and colleagues who were always there to extend support and motivation. This work was partially funded by Sanstream Technology and National Sciences and Engineering Research Council (NSERC) Canada and I would like to thank Sanstream Technology for their support.

Finally, I would like to thank my family who is always encouraging me during my studies.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADT	Android Development Tools
AMI	Amazon Machine Instance
Apk	Android application package
API	Application Programming Interface
APP	Application software
AWS	Amazon Web Services
AVI	Audio Video Interleave
CDN	Content Delivery Network
CPU	Central Name System
DNS	Domain Name System
EC2	Elastic Cloud Computing
EDR	Enhanced Data Rate
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Services
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
MCC	Mobile Cloud Computing
MFLOP	Mega Floating-point Operations
MFLOPS	Mega Floating-point Operations Per Second
OSMF	Open Source Media Framework
PaaS	Platform as a Services
PC	Personal Computer
PCAP	Packet Capture
RAM	Random Access Memory
RFCOMM	Radio Frequency Communication
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSSI	Received Signal Strength Indication
RTMP	Real Time Messaging Protocol
RTT	Round Trip Time
SaaS	Software as a Services

SDK	Software Development Kit
SPP	Serial Port Profile
SSH	Secure Shell
S3	Simple Storage Services
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USA	United states of America
USB	Universal Serial Bus
UI	User Interface
VM	Virtual Machine

Chapter 1: Introduction

This chapter introduces the problems of mobile devices and solutions for the problems with emphasis on energy consumption and computation offloading. After that, there will be a discussion on the motivation of developing an adaptive algorithm for computation offloading for energy saving for the mobile device. The contributions and the organization of the thesis will be described next.

1.1 Limited energy and computation resources of mobile devices.

With the availability of mature and advanced wireless technologies, mobile devices have gained tremendous growth. Statistics [Statistics14] show that mobile devices will take over desktop devices by 2014 and people have spent an average of 2.7 hours per day on their mobile devices in America. Consumers spent 89% time, an average of 2.7 hours per day, on media through mobile APPs, while the rest of 11% media time through mobile webs. Moreover, about 61% of mobile users play games which are energy-hungry applications and are dominating the total application usage. There are more energy hungry applications, for example, face recognition, speech and object recognition, natural language processing, computer vision and graphics, virus scan, and mobile augmented reality. Mobile devices, including smart phones, tablets and notebooks, are also constrained with small battery size, CPU, and storage. At the same

time, the current battery technology is still a bottleneck in mobile devices, which results in a growing gap between the demand of energy and the capacity of the resources of mobile devices. With rich social mobile applications available, effective power management techniques are paramount in mobile devices. Research and industrial communities have investigated the factors affecting to energy consumption of mobile devices and the techniques to measure the contribution of those factors. At the same time, researchers have also put forward many power management techniques, including offloading computationally intensive tasks from mobile devices to external resources, such as cloudlet computers or cluster of computers, nearly PCs, or even remote cloud servers, with an aim in saving power consumption in mobile devices and/or improving user experience for highly computation-intensive tasks.

1.2 Solutions for limited energy and computation resources of mobile devices.

For mobile applications requiring high processing power, computation offloading is a natural solution for mobile devices by taking advantage of their connectivity to external computation resources via wireless communication technologies, e.g., 3G, 4G, Bluetooth or Wi-Fi. Subsequently, the external computation resource performs the application and sends the execution results of the application back to the mobile device. The goal of computation offloading is to lower energy consumption and/or improve application performance of mobile devices compared the application to

be performed on the mobile device locally without taking into account the capabilities of other resource-rich computing platforms. Although, computation offloading has great advantage in energy efficiency and application performance, there are still challenges preventing us from building a more efficient offloading framework.

The most critical decisions for computation offloading are when to offload and what infrastructure to use in order to benefit mobile users for offloading as described in [Kumar13]. There are four main issues to consider in offloading algorithms. The first issue is that the algorithms should be developed successfully to save energy and/or improve performance of mobile systems. Besides the offloading algorithms, the cost of data transfer increases when using 3G or 4G wireless network technology. Next, security is also one of the critical issues in offloading, since data will be transferred between mobile systems and other computation resources. One more issue is the availability of the computing servers. In other words, the computing servers for offloading destination may not be available or accessible in a public place. These issues should be considered carefully before implementation of a computation offloading technique.

This thesis focuses on computation offloading techniques for mobile applications. To meet this goal, a Google Android device – Nexus 7 tablet – has been chosen as the test device due to the fact that Android is an open source operation system based on Linux and a number of tools with similar functionalities in Google Play are available. To support offloading, various wireless communication technologies, such as 3G, 4G, Wi-Fi and Bluetooth can be used for data transfer. However, 3G and 4G cannot be applied to the chosen test mobile device – Nexus 7 tablet – mainly for the cost reason and low

network bandwidth compared to Wi-Fi. Hence, the thesis focuses on Bluetooth and Wi-Fi technologies which can be applied to every recent mobile device. The external computation resources can be nearby (within a range of 10 meters) PCs with Bluetooth connection or remote cloud servers via Wi-Fi, which will communicate with mobile devices to execute Mobile Cloud Computing (MCC) applications.

MCC integrates mobile applications with cloud services [Dinh11]. Using MCC, data storage and data processing will be moved from local mobile devices to powerful and centralized computing platforms located in the cloud over the wireless connections. After that, requested data or processing results will be transmitted back to the mobile devices. If local mobile devices utilize available cloud computing and storage resources to deploy computationally intensive applications in the cloud environment, power consumption of mobile devices can be reduced. On the other hand, higher latency and packet losses of the wireless networks, operating cost in cloud data centers, data integrity, and user privacy are all challenges to MCC [Wang13]. Researchers have paid attention to the above challenges using MCC. Chapter 2 will address the related work and the challenges associated with MCC in details.

1.3 Motivation and thesis contribution

To our best knowledge, Most of the existing computation offloading algorithms or applications, even some mobile applications with adaptive computation offloading methods, extend computations from the mobile devices to the cloud computing resources.

Based on the drawbacks of mobile cloud computation applications [Wang13], such as expensive cloud computing cost, high network latency delay and security issues, it is not necessary to offload all computations to the cloud if nearby computation resources are available.

Existing computation offloading algorithms or applications have motivated the research to develop an algorithm that is adaptive to various computational requirements and the mobile device's proximity conditions. Specifically, the algorithm can offload a computation task to either a nearby computation platform via Bluetooth or a cloud server via Wi-Fi based on the computational and user requirements. One critical factor of choosing Bluetooth and Wi-Fi for the mobile application is the energy consumption. The energy consumption ratios for Wi-Fi over Bluetooth are 1.32 when downloading files and 2.22 when uploading files in Android phones [Kalic12]. In other words, 32% more energy for downloading and 122% more for uploading are needed when using Wi-Fi.

Meanwhile, there are a number of existing energy and performance measurement tools and widgets in Google Play. It is not trivial to select the best tools among the existing tools to examine energy consumption of mobile devices. Hence, a great deal of efforts has been spent on evaluating a number of tools for energy consumption and performance measurements. To better understand mobile cloud applications, some cloud applications have also been selected and tested to investigate affecting factors of energy consumption and performance of mobile devices.

In the thesis, the research consists of two main stages: the first stage was to select tools from Google Play and run some mobile cloud computing applications to investigate

factors that affect energy consumption and performance. The next stage is to develop an adaptive algorithm based on experimental results for various scenarios of mobile applications.

The main contributions of the thesis are listed as follows.

- Investigation and evaluation of a number of tools, including Android Tuner [Tuner13], PowerTutor [Power13], Battery Monitor Widget [Monitor13], and Speedtest [Speed13], for power or energy consumption and performance, e.g., CPU time and network throughput, for various mobile applications.
- A series of experiments on measuring power consumption and performance in mobile devices have been conducted and some factors that are related to power consumption, CPU time, file downloading elapsed time have been investigated based on the experiments for various Mobile Cloud Computing (MCC) applications, e.g., Amazon S3, Amazon CloudFront and Video streaming.
- Development of an adaptive algorithm that can decide where to execute an Android application among three alternatives: running the application locally on a mobile device, offload the computation to a nearby PC based on computational requirements and constraints via Bluetooth, or offload the computation to a remote cloud server based on some requirements or constraints via Wi-Fi. The algorithm has been evaluated for energy efficiency and application performance with different data sizes and computation complexities.

Part of this thesis appeared in the following publication:

- Kelie Zhan, Chung-Horng Lung, and Pradeep Srivastava, "A Green Analysis of Mobile Cloud Computing Applications", *Proc. of the 29th ACM Symposium On Applied Computing*, March. 2014.
- Kelie Zhan, Chung-Horng Lung, and Pradeep Srivastava, "A Comparative Evaluation of Computation Offloading for Mobile Applications", submitted to *IEEE GreenCom*, Sept. 2014.

1.4 Thesis organization

The remainder of the thesis is organized as follows. Chapter 2 describes background knowledge of offloading decision making of offloading and related work of computation offloading. Chapter 3 presents measurement results of some MCC applications and discusses power and performance measurement tools that have been investigated. Chapter 4 describes the experiments of computation offloading and an adaptive algorithm based on the experiments. Chapter 5 concludes the thesis and outlines some future directions.

Chapter 2: Background and Related Works

The chapter will firstly present background knowledge of decision making of computation offloading for mobile systems and computation resources. Next, previous related work on computation offloading to external computation resources, its applications and adaptive algorithm will be reviewed.

2.1 **Offloading decision making**

As mentioned before, the growing gap between the requirement of battery and computational capability of mobile devices and limited resources of the mobile systems was triggering academic and industry concern on solutions for computation offloading to external rich computation resources. The computation resources could be neighboring mobile devices, nearby PCs and remote cloud servers. At the same time, some critical issues of offloading have been raised, such as whether and when to decide offloading. So the decision of offloading will be discussed to meet the goals of offloading, saving energy consumption of mobile devices or improving application performance.

Energy consumption of mobile devices is the most concern for mobile systems since existing battery technology limits widespread use of increasing complex computation applications in mobile devices, e.g., video encoding and decoding, image processing, language processing. Kumar, et al. [Kumar10] and Karthik, et al. [Kumar12] formulate energy saving for computation offloading to the cloud as follows:

$$P_c \times \frac{C}{M} - P_i \times \frac{C}{S} - P_{tr} \times \frac{D}{B} \quad (2.1)$$

where

- P_c – mobile power consumption when using computing;
- P_i – mobile power consumption when idle;
- P_{tr} – mobile power consumption when sending and receiving data;
- C – total number of instructions in required computation;
- D – total bytes of data;
- S – instruction processing speed in cloud server or other computation resources;
- M – instruction processing speed in a mobile device;
- B – network bandwidth;

$P_c \times \frac{C}{M}$ is the energy consumption of mobile systems when computation is executed in the mobile devices. $P_i \times \frac{C}{S} + P_{tr} \times \frac{D}{B}$ is the total energy consumption of mobile systems, including energy consumed by the mobile devices when it is in idle state waiting for the computation result and energy consumed by a mobile device for data transfer between the device and the offloading destination, when computation is executed in other computation resources. This indicates offloading computation to the powerful computation resources can save energy for local mobile systems if equation (2.1) results in a positive number. Otherwise, offloading cannot save energy for mobile systems. From the equation, heavy computation (large C) can result in a positive value of the equation

(2.1). On the other hand, small network bandwidth or large amount of data transferred will result in a negative number of the equation (2.1). For instance, 3G network has large RTT, low download and upload throughput [Kumar12], which are challenges to computation offloading to remote cloud.

It is clearly shown that not all computation offloading to the powerful computation resources can save energy consumption of mobile systems. So it is essentially to make decision when offloading should happen. However, some parameters, such as C, M and S, are not easily to determine exact values and the equation (2.1) can be converted to an new equivalent equation (2.2) as follows.

$$E_m - (E_i + E_{tr}) \quad (2.2)$$

where

- E_m - energy consumed by a mobile device when computation happen in the device;
- E_i - energy consumed by a mobile device in idle state to wait for computation result when computation happen in external computation resources;
- E_{tr} - energy consumed by a mobile device when sending and receiving data between the device and external computation resources;

All of these three parameters, E_m, E_i and E_{tr}, can be obtained by energy consumption measurement from experiments. So if equation (2.2) results in positive number, offloading can save energy for mobile systems. Otherwise, it means that mobile devices cannot benefit from offloading and is no worth offloading computation from a

mobile device to external computation resources. The decision of the offloading algorithm is only concerned about energy consumption of a mobile device. The other reason for making offloading decision is to improve application performance of mobile devices, such as response time of applications

The most common of improving application performance of mobile devices for computation offloading is reducing response time of the application. The difference of response time between computation happen in mobile devices and external computation resources as stated in Kumar, et al. [Kumar12] is expressed in the following equation:

$$\frac{W}{S_m} - \frac{W}{S_s} - \frac{D_i}{B} \tag{2.3}$$

Where

- W - the amount of the computation;
- S_m - computation speed of the mobile systems;
- S_s - computation speed of the external computation resources;
- D_i - the amount of data will be transferred;
- B - network bandwidth;

In the equation (2.3), $\frac{W}{S_m}$ is the computation time of mobile systems when computation is executed in the mobile devices. $\frac{W}{S_s} + \frac{D_i}{B}$ is the response time including waiting time for computation result and communication time for data transfer when offloading to external computation resources of mobile systems. It is apparently that

offloading will reduce response time of mobile systems if the result of the equation (2.3) is greater than 0. Otherwise, it is no worth offloading computation to other computation resources outside of the mobile systems. From the equation, the fast computation speed of external computation resources, which is the offloading destination, can result in a positive value of the equation (2.3). Inversely, heavy data exchange may result in a negative result of the equation.

2.2 **Computation resources**

As discussed before, mobile devices are resource-constrained devices with limited energy, storage, memory and computation power. Fortunately, the cloud computing technology can offer unlimited resources for computation, storage and service provision to mobile devices. Mobile Cloud Computing (MCC) can incorporate cloud computing features and mobile applications. Except cloud resource, Cloudlet computers or cluster of computers, nearby PCs are other available resources for mobile devices. Mahadev, at el. [Mahadev09] and Jararweh, at el. [Jararweh13] claim a cloudlet can be seen as a "data center in a box" to aim to bring the cloud closer and has low latency and high bandwidth wireless network. Apparently, a cloudlet can benefit nearly mobile users. However, it is not easily to create the Cloudlet infrastructure. Therefore, Cloudlet computers and nearby mobile devices with slow processors won't be considered as computational offloading destination in the thesis. In the future, Cloudlet infrastructure will be created and computation offloading experiment will be implemented since it will bring cloud more closer to end users and thus benefit them. Nearby PCs can be traditional

servers and will be used to carry out computation offloading experiments later. The most popular computation destination, also an interested topic in the thesis, is remote cloud severs, which have been raised concern in academic and industrial communities and also have many papers related to Mobile Cloud Computing (MCC), including architecture, applications and approaches. The thesis describes experiments of some MCC applications. To start related work of MCC, it is important to give introduction of cloud architecture and an overview of MCC. The first thing needed to be clarified is cloud architecture.

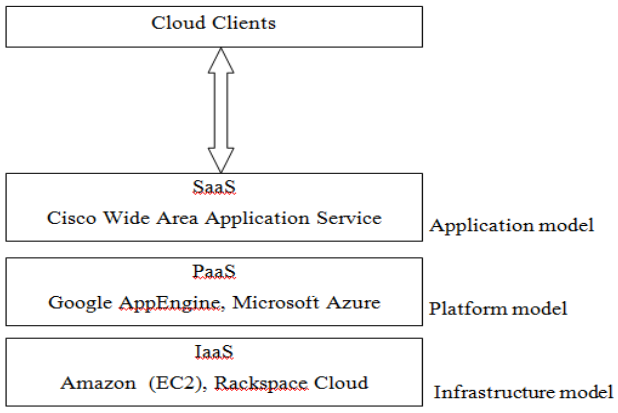

Figure 2.1 Service-Oriented Cloud Computing Architecture [Compu14]

Cloud computing providers can be classified in Figure 2.1:

- ❖ Infrastructure as a Service (IaaS)
    - Virtualized resources: computation, storage and communication
    - Example: Amazon Elastic Cloud Computing (EC2), Rackspace Cloud
- ❖ Platform as a Service (PaaS)

- Environment for developers to develop and deploy application with hardware information
- Example: Google AppEngine, Microsoft Azure

❖ Software as a Service (SaaS)

- Application software
- Example: Cisco Wide Area Application Service (WAAS)

For experiments conducted in this research, the remote cloud servers are all created in Amazon EC2 and run on Amazon's proven computing environment since EC2 can provide cloud users with complete control of computing resources, e.g., resizable computing capacity in the cloud [Amazon14].

As described in Chapter 1, MCC integrates mobile applications with cloud services. Dinh, et al. [Dinh11] list several advantages of MCC and clarify how the cloud to be used to overcome obstacles of mobile devices. The main advantages of MCC are:

- Extended battery lifetime. The lifetime of mobile devices' battery can be extended through computation offloading technique in MCC.
- Improved data storage capacity and processing power. MCC allows mobile users to store and access large data stored on the cloud. Computation-intensive applications in mobile devices can be transferred to the cloud and executed on the more powerful cloud servers.
- Improved reliability. The data and applications transferred from mobile devices to the cloud will be stored and back up on a number of servers.

However, some challenges of MCC cannot be neglected: high Round Trip Time (RTT), low network throughput of wireless communication technology and security are constraints for mobile services in MCC. After providing an overview and architecture of MCC as shown in the Figure1, the related work of MCC and computation offloading will be presented.

Mobile cloud computing, as shown in Figure 2.2, is the combination of mobile computing and cloud computing, which can bring benefits for mobile user, network operators and cloud computing operators. The aim of MCC is to enable execution of rich mobile applications in a more powerful computation resource - cloud, thus improve mobile users experiences. [MCC14] lists four types of cloud-based resources, distant immobile clouds, proximate immobile computing entities, proximate mobile computing entities and hybrid, which is combination of the other three types. Amazon EC2 is distant immobile clouds. Cloudlet and surrogates are member of proximate immobile computing entities. Smart phones, tablets, and wearable computing devices can group proximate mobile computing entities. Amazon EC2 is used in the execution of the experiments of the thesis and Amazon global infrastructure and available services will be simply introduced.

Available services provided by Amazon Web Services (AWS) can be classified different categories: Compute & Networking, Storage & Content Delivery, Database, Deployment & Management, Analytics and App Services. EC2 is the popular service in Compute & Networking. Customers of AWS can specify different pre-created instances,

Amazon Machine Instance (AMI), or create a instance with resizable computation resource. Instance is a virtual server in the cloud and AMI is special type of virtual

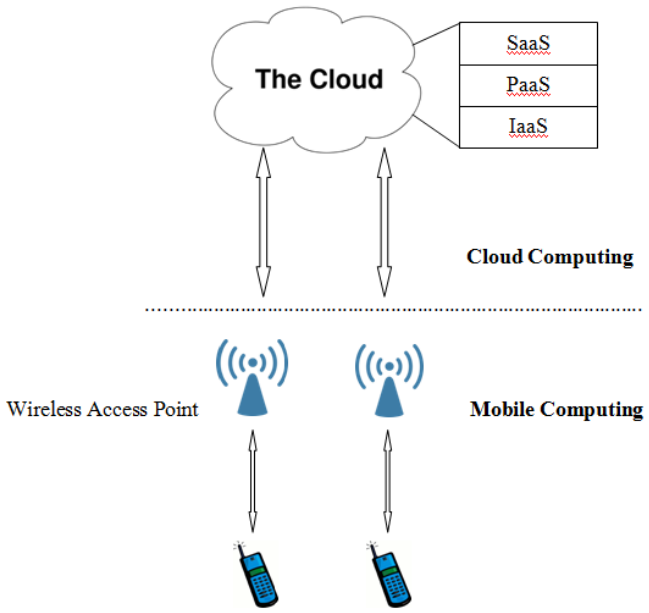

Figure 2.2 Mobile Cloud Computing Architecture [MCC14]

appliance that is used to instantiate a virtual machine within the Amazon EC2. Storage & Content Delivery includes Amazon S3 and CloudFront. In the thesis, experiments using S3 and CloudFront services in Amazon EC2 are carried out and video streaming application with an AMI in the cloud is launched.

Moreover, the location of servers will determine RTT. It should be carefully to select suitable location of the instance to be created and registered to lower network latency. The available locations for AWS are Northern Virginia, Oregon, Northern California, Ireland, Singapore, Tokyo, Sydney, Sao Paulo, GovCloud, Beijing. The best

selection of server location is an instance, which is created closer to the end user's geographical location. For example, instances in Northern Virginia are mostly created since the data center in Northern Virginia is closer to Ottawa than others data center. More locations of data centers can be used to compare network latency to check application performance, such as file download elapsed time. No all locations of data center can provide all kind of services. After discussing basic knowledge of computation resources and offloading decision making, related work of MCC will be presented.

2.3 Related work on Mobile Cloud Computing (MCC) applications and Computation Offloading

Cloud based computation offers support in multiple domain, e.g., storage, healthcare, video decoding and other intensive computation applications, to overcome limited resources of mobile devices. For storage service in the cloud, Palanker, et al. [Palankar08] present evaluation of Amazon S3 in usage cost, data security and data access performance compared with desktop computers. The authors of the paper realize that caching files at the edge of the system can significantly reduce network latency before Amazon CloudFront was launched on November 18, 2008.

Lai, et al. [Lai13] design a cloud-assisted real-time translating mechanism to dynamically adapt to unstable network bandwidth conditions and transmit different bitrate videos to end users, so that the end users can achieve good user experience. Wang, et al. [Wang13] also develop a network adaptation approach based on MCC technology,

which moves video encoding or graphic rendering to cloud servers closer to end users to improve user experience. In industry, Wowza Media Systems have launched content delivery network for high-speed media streaming through Amazon CloudFront on September 11, 2013 to offer customers good user experience of high quality video streaming [Wowza13].

The authors in [Dinh11] study mobile cloud architectures, offloading decision, the critical analysis of mobile cloud applications. The authors also detail analyze various offloading models, such as CloneCloud, MAUI, ThinkAir, Cuckoo and so on.

Fernando, at el. [Fernando13] present an overview of mobile cloud computing research and highlight the motivation for mobile cloud computing. Various definitions of mobile cloud computing in the literature will help understand MCC more clearly. Authors of the paper also discuss offloading methods in three main directions: Client - Server communication methods, Virtualization and mobile agents. Table 2.1 shows an overview of approaches in offloading methods.

Kunar, et al. [Kunar13] classify the research of computation offloading from 1996 to 2010. It not only clearly addresses offloading decisions mechanism, but also describes offloading techniques for improving performance and saving energy. Apparently, the paper provides a overview of the background of motivation of computation offloading.

Jiao, et al. [Jiao13] present a detailed comparison from cloud - based computation offloading, CloneCloud and MAUI, for mobile devices. At the same time, they also

Table 2.1 An Overview of Approaches in Offloading Methods

	Client - Server communication	**Virtualization**	**Mobile agent**
Method descriptions	Offloading process is done across the mobile device and surrogate devices via protocols	The offloaded tasks will be executed on the pre-created virtual machine in surrogate devices.	The offloaded tasks will be executed in a mobile agent in surrogate devices.
Examples	• RPC • RMI • Sockets	• CloneCloud • MAUI	• Scavenger
Pros	Stable and well support APIs	No code rewriting is needed; VM boundaries insulates servers.	Dynamic deployment
Cons	Need prior installation	VM synthesis take time; Compatibility issues between VM overlays	Security constraints; Issues with agent management.

provide some suggestion on facilitating the offloading implementation tasks with existing technologies. Eventually, authors believe MCC will greatly benefit mobile users in user experiences. The study only focus on MAUI and CloneCloud offloading mechanism in MCC.

Barbera, et al. [Barbera13] analytically study the cost of data and computation offloading in terms of bandwidth and energy consumption in MCC. The evaluation

shows that Wi-Fi technology incurs lower overhead with respect to 3G. The paper provides us a clear view how to lower overhead of bandwidth and energy consumption in MCC in real life scenarios.

Wu, et al. [Wu13] describe a mobile healthcare system with capacity of cloud computation offloading and the system can receive an energy efficient transmission between multiple cloud servers to exchange data info. The system claimed by authors is on an analytical and simulation stage and gets benefit on energy efficient and user experience from computation offloading in MCC. Unfortunately, the system wasn't verified in real life scenarios.

Chun, et al. [Chun11] design and implement CloneCloud system. The system will automatically transform mobile applications to cloud environment. Finally, the prototype of the system can speed up execution time of transfer mobile applications and decrease energy consumption on the mobile devices.

Cuervo, et al. [Cuervo10] present design and implementation MAUI system. The system focus on energy saving and offload a single application code to remote available infrastructure, such as public cloud. The evaluation shows MAUI can save energy and increase application performance.

Mtibaa, et al. [Mtibaa13] have built an emulation test bed and evaluated various offloading algorithms based on selection of different computation task executors, nearby mobile devices and Cloudlet computers. In this paper, authors claim that the developed

offloading algorithms can greatly save energy and improve user experience of mobile devices while the computation and data will be offloaded to nearby mobile devices.

Unfortunately, [Wang13][Palankar08][Lai13] focus on providing good network and application performance to end users and neglect power consumption on mobile devices, although the authors of [Wang13] claim that a gap between capacity of battery and abundant mobile applications is widening gradually. Obviously, it is important to study power consumption for mobile devices for different applications. To this end, it is necessary to look for suitable tools for power and measurements of other performance metrics.

In summary, all of above papers demonstrate some offloading potential in improving performance or saving energy. [Kumar13] and [Jiao13] present a general overview of computation offloading of mobile applications. Besides [Sharafeddine13], other papers discuss offloading research in cloud environment. As mentioned before, mobile devices can also benefit from computation offloading via Bluetooth. [Sharafeddine13] and [Mtibaa13] have addressed methods on offloading to a nearby PC via Bluetooth.

However, [Sharafeddine13] and [Mtibaa13] mostly focus on offloading implementation between mobile devices using Bluetooth communication technology without considering cloud. This thesis extends the scope by considering three different types of resource when it comes to performance and energy consumption. They are mobile device, a nearby PC, and a remote cloud server. Nearby PCs mostly have more powerful resources than mobile devices, including storage and computation.

Therefore, a nearby PC is the preferred offloading destination if certain conditions and constraints exist. On the other hand, computation may be more beneficial if it can be offloaded to the cloud for other scenarios. This thesis conducts thorough an evaluation of energy consumption of mobile devices and application performance using either Bluetooth or Wi-Fi technology, and develop an adaptive algorithm based on the experimental results with an aim to save energy and improve performance using the adaptive offloading approach.

Chapter 3: Performance Measurements for MCC Applications

This chapter investigates the power consumption and performance of MCC for various Android applications using existing tools available in Google Play. The objective of the investigation is to better understand the power consumption and performance for MCC applications before development of any mechanism to manage power consumption and facilitate trade-off analysis between power consumption and performance of MCC applications.

This chapter is organized as follows: Section 3.1 describes the MCC applications that have been investigated and the measurement metrics for each application. Section 3.2 presents the main existing tools for power and performance measurement. Finally, Section 3.3 demonstrates the experiments and results.

3. 1 MCC applications

Table 3.1 summaries the MCC applications that have been evaluated in the thesis. Although mobile Apps are believed to dominate mobile web browser usage, web browsing is also widely used in mobile devices. The new trend for mobile web browsing is that the web browser will change and adapt to the configuration of mobile devices, for instance looking and behaving more like APPs [Mobileweb14]. Google Chrome is one of the most popular web browsers. SkyFire provides social and convenient browsing and

Table 3.1 The summary of MCC applications

Applications	Application description	Measurement metrics
Amazon S3 (simple storage services)	Downloading files with test mobile devices	• Download elapsed time • power consumption for different file sizes and formats
Amazon CloudFront	Distributing files to edge locations close to end users on the internet through content delivery services	• CPU time • Power consumption
Web browsing	Comparing two web browsers – Chrome and Skyfire	• CPU time • Power consumption
Video streaming	Deploying Adobe media server in Amazon cloud platform for streaming video to test mobile devices	• CPU time • Power consumption • Memory usage

video with mobile cloud solutions. Web browsers, therefore, are included in the list of MCC applications.

Video streaming is one of energy hungry applications for mobile devices. At the same time, video streaming is very popular for mobile users, for example online and live video. Video streaming applications need to meet end users' requirements for good user

experience. Therefore, all of above applications were selected for the investigation in terms of energy consumption and performance.

3.2 Power and Performance Measurement Tools

Since the target test devices are Android mobile devices as Android devices have the largest market share, and the operating system is open source and can be modified for further design to be applied to special hardware devices, measurement tools only for Android devices are investigated and discussed in this thesis. There are a number of existing energy and performance measurement tools and widgets in Google play. However, not all of them can meet all the needs in power and performance measurements for local mobile devices. It is also not easy to select the best tools among the existing tools regarding power consumption of mobile devices. Based on the investigation, not one tool can measure all the desired power consumption and performance metrics, such as CPU time and memory usage, delay, packet loss percentage and jitter, at the same time. A number of tools have been investigated and some tools are used in the measurement of the experiments. The details of some of tools used are discussed as follows.

1) Android Tuner (including System Tuner Pro): Android Tuner can be used to monitor CPU, network and memory usage. Android Tuner can also be used to manage files, such as viewing files, editing files and accessing any folders and reading data from rooted devices. Further, Android Tuner can be used to

access some folders, clean cache and view files for unrooted devices based on experiments.

2) Mobiperf: Measurement types for Mobiperf include traceroute, HTTP, TCP Speed test, ping task, percentage of packet loss, RTT and DNS lookup time. It is possible to modify open source code and use for measurement of upload and download speed for more servers since it only support one fixed server currently.

3) tPacketCapture: This tool can be used to capture IPv4 TCP/UDP packets and save in PCAP format without root or administrator access. Captured packets can be used for more detailed analysis with special software, such as Wireshark. It is very useful to capture network packets for further analysis. The analysis can be used to analyze network problems, detect network intrusion attempts, monitor network usage, gather and report network statistics and so on. To capture wireless packets of mobile devices, the effective way is to root mobile devices and use particular network capturing software. However, manufacture guarantee of a mobile devices is invalid as soon as the device is rooted. tPacketCapture can capture packets for unrooted mobile devices. The disadvantage of the tool is it cannot be used for real time applications.

4) SpeedTest: As a trusted and accurate professional tool, SpeedTest can be used for measuring ping, upload/download speed for more than 10 servers and retrieving back past test detailed report for reference.

5) Traceping: The tool can measure packet loss, round trip time (RTT) and jitter with further configured ping interval and packet size of data to be sent. Traceping can continuously send periodical echo requests and get more accurate test results based on large amount repeated tests. However, an unknown destination of server is shown in the tracking path in some of our experiments, since Amazon does not guarantee that a fixed destination server in virtual machines will be provided to Amazon AWS users.

6) Traceroute: Traceroute can be used to measure packet loss and RTT. The advantage is it can support ICMP and UDP mode. The maximum number of hops can be configured. The drawback is corresponding to a large amount of traffic with sending and receiving packets to and from each route along the path to the remote host.

7) PowerTutor: PowerTutor can display total and individual components' power or energy consumption. Gu, et al. [Gu12] has used PowerTutor to evaluate power consumption during running time. PowerTutor [Zhang10] uses linear regression to derive the coefficients of each individual component and combines major system components such as CPU, network interface, display, GPS receiver and audio system of these components consumed power or energy. Linear regression has been widely used for modeling power consumption of hardware and has been proved that it outperforms nonlinear regression methods [Mccullough11]. The power model used for PowerTutor was specifically built on HTC G1, HTC G2 and Nexus one. As a result, power or energy consumption for other mobile devices is based on estimation. To be

accurate for the selected mobile device, e.g., Nexus 7, used in the experiments, the power consumption model should be further investigated based on experiments.

8) Battery Monitor Widget Pro: This tool can automatically record battery metrics, such as voltage, temperature and remaining percentage of battery for Android devices. It also shows Wi-Fi statistics: Received Signal Strength Index (RSSI), link speed and estimates applications run time and battery aging. Obviously, it is good for battery monitoring, battery history recording at different update rate. Chen, et al. [Chen12] used the widget to measure power consumption of mobile devices.

Based on the investigation of the functionalities of the abovementioned tools, a summary of the strength of each tool is presented as follows.

- Android Tuner can be used in Android mobile devices for file exploration, CPU time and memory usage.
- Mobiperf can be used for packet lost and RTT.
- Traceping is used to measure Jitter.
- Speedtest is adopted to measure network upload and/or download speed .
- tPacketCapture can be used for non-real time mobile applications to capture packets.
- PowerTutor is adopted for power or energy consumption measurement.
- Battery Monitor Widget Pro is good for battery monitoring.

As mentioned before, no existing tool can support measurements for all or most of the performance metrics that often are needed. The following section presents experiments that have been performed using three tools – **SpeedTest**, **Android Tuner** and **PowerTutor** – to measure the key performance metrics: network download speed, CPU time and memory usage, power consumption for mobile devices. Since the experiments focus on energy consumption and performance of mobile applications, other selected tools will not be used in the experiments. The next section presents experimental results that have been performed using these three tools.

3. 3 Experiments of MCC applications

A number of experiments have been carried out on Nexus 7 with Jelly Bean Android 4.2 operating system. Since some selected tools can not run properly on Android phones, Nexus 7 is the only mobile device evaluated in the experiments. This section demonstrates experimental results for MCC applications:

- storage services (Amazon Simple Storage Service (Amazon S3)),
- Amazon CloudFront
- Web browsing, and
- Video streaming.

3.3.1 Storage Services Using Amazon S3

The experiments with Wi-Fi only are all run on the Carleton campus. The download speed is about 19.7 Mbps for strong signal and 13.4 Mbps for fair signal.

Amazon S3 [Sthree13] provides a simple storage web services to end users and is supported by large number servers across multiple data centers (including three data centers in US). End users can select a data center where is close to their geographical location to have lower network latency. For example, the closer data center to Ottawa lies in Northern Virginia, Eastern USA. Most Amazon S3 experiments for file storage are conducted for the thesis, unless the location is specified with the data center in Northern Virginia. Then QPython, a script language for Android devices, is used to automatically download files and calculate files download elapsed time. Due to the unpredictable fluctuating network download speed for Wi-Fi network interface, each experiment was repeated 10 times to reduce random errors and the average of 10 runs is used as the final result.

Figure 3.1 and Figure 3.2 exhibit the download elapsed time versus the image file size. As shown in these two figures, the download elapsed time increases with the size of the image files. Complete downloading a 0.828MB file needs 2.8695s under fair Wi-Fi signal strength. If the file size is a 3.921MB, download elapsed time increases to 10.7175s. Comparing Figure 3.1 with Figure 3.2, download elapsed time under strong signal strength is less than that under fair signal strength for the same file. This can be seen from these two figures that the elapsed time for downloading the 0.828MB file is 1.3823s with high network bandwidth compared to 2.869 s under the fair signal strength. That is to say, high download speed reduces file download elapsed time. For the same file size with different format, download elapsed time is the same.

Figure 3.3 and Figure 3.4 illustrate the power consumption of the test mobile device. As shown in these two figures, the power consumption increases as the size of image files increases. Moreover, high download speed can save the mobile device's

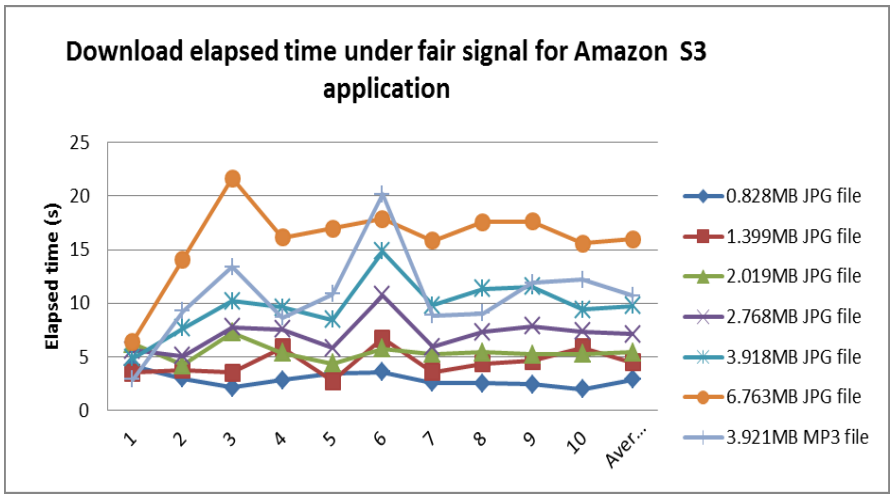

Figure 3.1 Download Elapsed Time for S3 (fair signal)

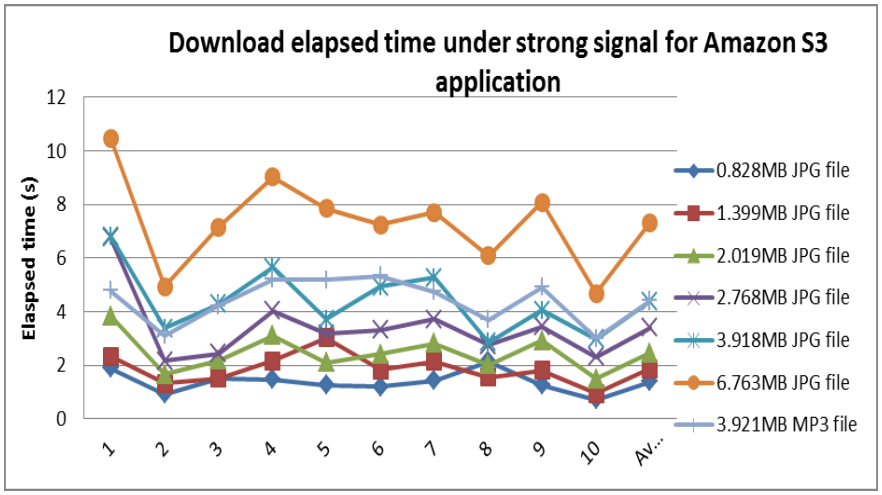

Figure 3.2 Download Elapsed Time for s3 (strong signal)

power consumption for file downloading. Downloading the 0.828MB file costs 372.3mw under the fair signal strength, while only 328.3mw under the strong signal strength.

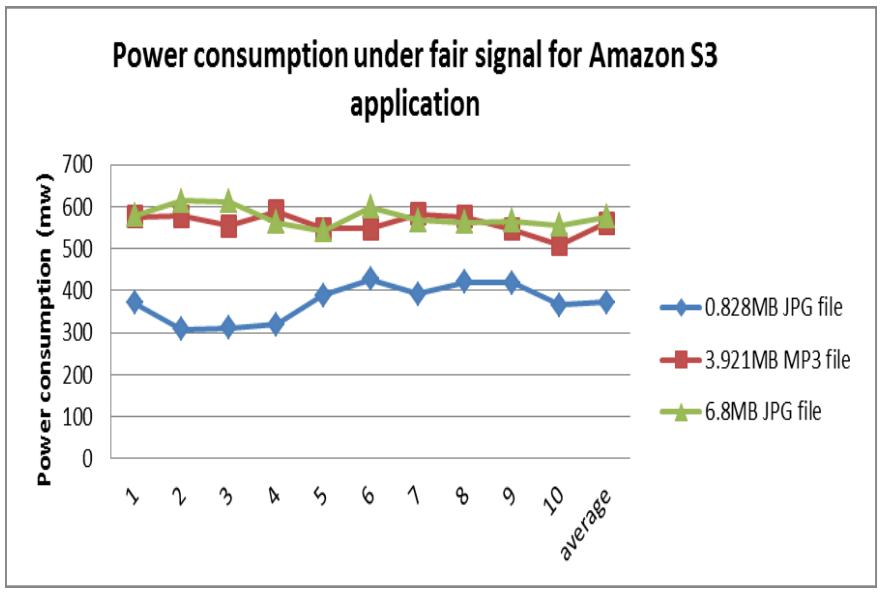

Figure 3.3 Power Consumption for S3 (fair signal)

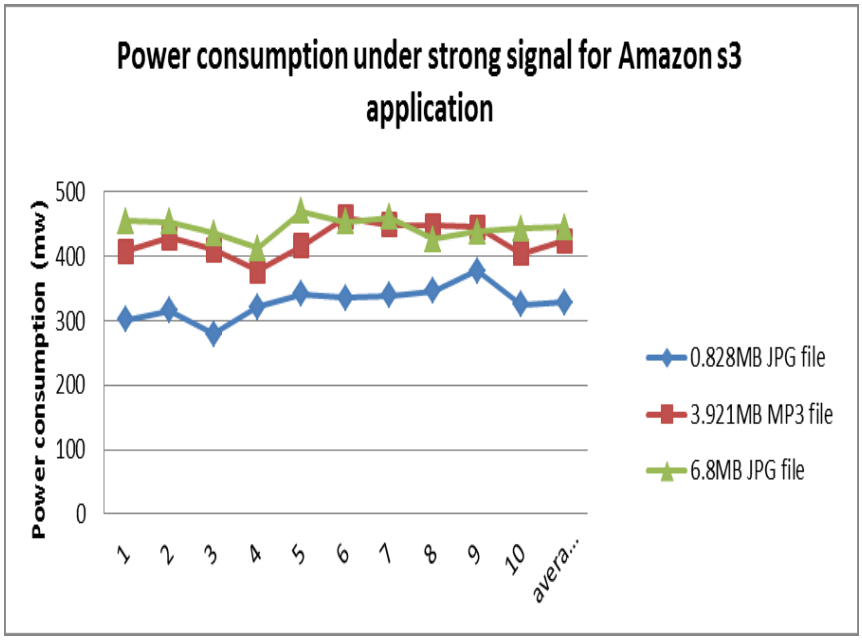

Figure 3.4 Power Consumption for S3 (strong signal)

As expected, Figure 3.3 and Figure 3.4 also depict that mobile devices need to consume more power to download larger files. Power consumption of the tablet is

372.3mw for the 0.828MB file and 575.7mw for a file of 6.8MB. However, power consumption in mobile devices is not proportional to files size. For instance, the power consumption values for downloading the files with 0.828MB and 3.921MB are 372.3mw and 560.4mw, respectively. For the file size doubles from 3.921MB to 6.8MB, the power consumption only increases by 15.3mw. Balasubramanian, et al. [Balasubramanian09] explain that mobile devices have high power consumption to establish connection with servers and the power needed to download data only lightly increases, even flat after the connection is set up.

To compare file download elapsed time with respect to different data centers, the tested files are stored in different data centers – Northern Virginia, Northern California and Tokyo. The distances between these three cities and Ottawa are 1,369Km, 4,512Km and 10,312Km, respectively. Figure 3.5 clearly shows that the download elapsed time increases as the distance increases, as the propagation delay increases. For a 1.399MB file, the download elapsed time is 1.7139s from North Virginia, 1.867s from North California, and 5.1623s form Tokyo, respectively. In addition to the file size, the network download speed will also affect files download elapsed time.

As a larger file size needs more transmission delay to push all of the bits of each packet into the transmission channel, results in longer download elapsed time. High network speed spends less propagation delay to travel a bit from a sender to a receiver and reduces file download elapsed time. High network speed can process bits faster but it also needs more power. The longer the distance between a sender and a receiver results in

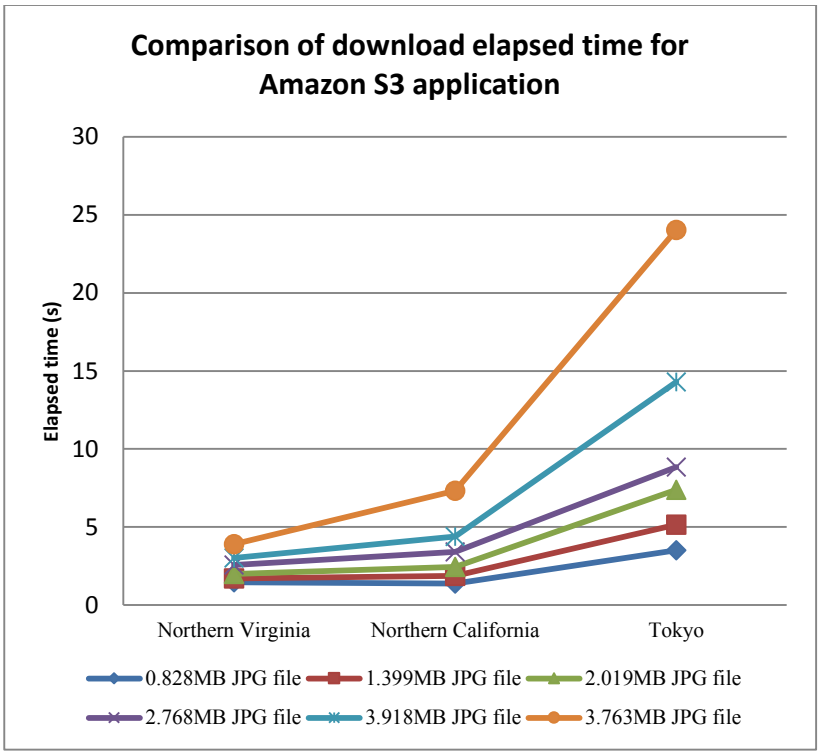

Figure 3.5 Download Elapsed Time for Amazon S3

the higher propagation delay and spends more time to download files from the data center to the mobile device.

3.3.2 Amazon CloudFront

Amazon CloudFront [Cloudfront14] distributes files saved in Amazon S3 to edge locations closer to end users on the Internet through content delivery services and thus lower network latency when downloading files. Amazon CloudFront is based on the concept of Content Delivery Networks (CDNs) using a distributed approach and file

downloading requests will be automatically routed to the nearest edge location. The difference between S3 and CloudFront is that Amazon S3 is centralized, while Amazon CloudFront is geographically distributed. Figure 3.6 shows that CloudFront can save download elapsed time. For example, the download elapsed time using CloudFront drops from 1.3823s (using Amazon S3) to 1.0202s for the same size of 0.828MB in JPG format, which is a significant improvement in terms of percentage.

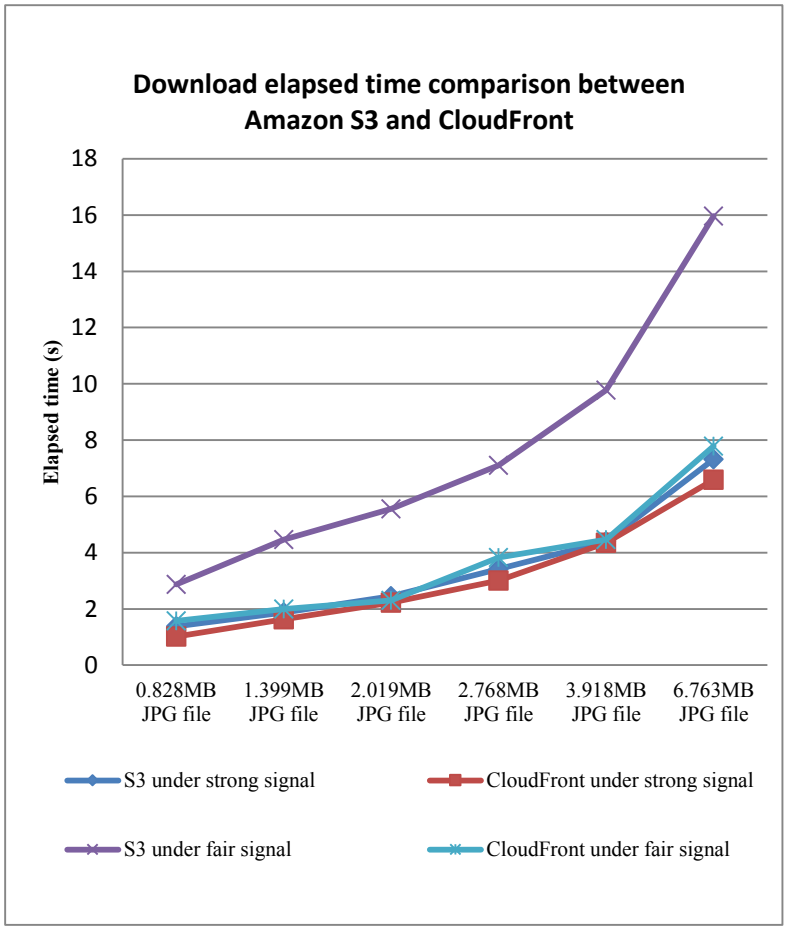

Figure 3.6 Download Elapsed Time between Amazon S3 and CloudFront

3.3.3 Web Browsing

Web browsing is one of the most popular web services. Thiagarajan, at el. [Thiagarajan12] present a detailed analysis on various factors affecting the web browsing performance. In this experiments, two web browsers – Google Chrome and Skyfire have been compared in terms of performance. Skyfire can offload partial image rendering to the front-end proxy servers in the cloud and resize images to fit the device screen. After that, the end device just simply displays the processing result.

In the experiment, power consumption and CPU time between Chrome and SkyFire have been compared. These two web browsers are used to browse a web site and play a 2:30-minute video clip on the test device, Nexus 7. The results show that Google Chrome has a slightly lower amount of power consumption using the test tablet, as shown in Figure 3.7. Figure 3.8 shows that SkyFire has higher CPU time to download and play the video, which contributes more power consumption for the test tablet, compared to Google Chrome. In Figure 3.8, the x-axis is for 5 different experiments and the average CPU time of 5 runs.

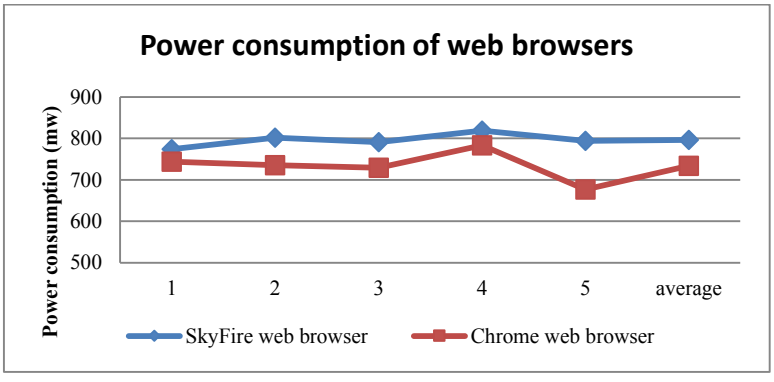

Figure 3.7 Power Consumption for Web Browsing

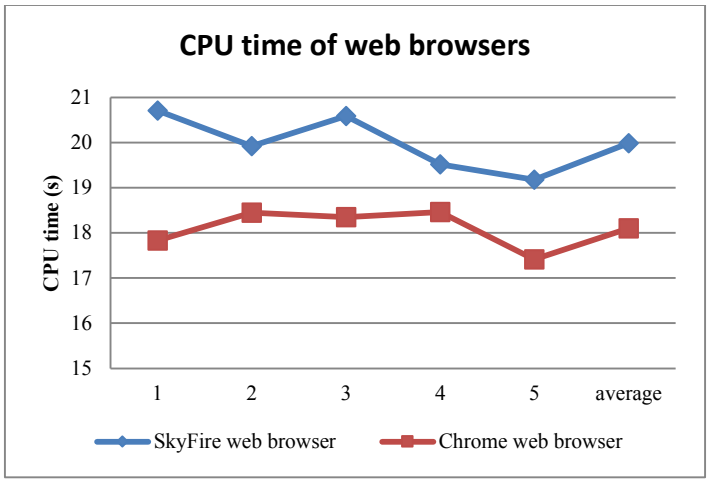

Figure 3.8 CPU time for Web Browsing

The aim of SkyFire is to save mobile device's power consumption through offloading computation intensive tasks, e.g., image rendering, to a remote cloud server which sends the computation result back to the mobile device for display. However, the SkyFire web browser does not save the power consumption of the tablet in the experiments. The experiment result shows that the computation offloading method used in SkyFire does not function properly in the tablet.

3.3.4 Video streaming

in this experiment, video streaming is deployed by an Adobe media server [Adobe13], a pre-configured instance (ami-08cd7261) in Amazon Elastic Computer Cloud (EC2), which can stream live and on–demand media to video players. The media can be selected with various bitrates when converting from AVI to mp4, since mobile devices does not play AVI. The video players built with the Open Source Media

Framework (OSMF), an online tool to publish video on website, will be running on Adobe Flash Players. Adobe Media Servers on Amazon Web Services can be easily designed to deploy and manage multi-protocol (RTMP – Real Time Messaging Protocol and HTTP – HyperText Transfer Protocol) media streaming. In the experiment, an Adobe media server instance with the m1.large Amazon EC2 configuration (in the US East region) and PuTTY are used to connect to the instance as instructed in [Adobe14]. PuTTY is a free tool of Telnet and SSH for Windows and Unix platforms,

In the video streaming application, Figure 3.9 shows that the power consumption of the mobile device has not much difference for video using the same bitrate. Power consumption of the tablet is 616.9mw for 3.4MB file, 623.9mw for 8.27MB, 624 mw for 15.8MB, 633.5mw for 27.3MB, respectively. However, the bitrate of video files has an effect on the mobile device's power consumption when streaming video as shown in Figure 3.12. The power consumption of the test tablet is 605.8mw for 0.5Mbps bitrate, 622.2mw for 0.8Mbps bitrate, 633.2mw for 3Mbps bitrate, 636.8mw for 6Mbps bitrate and 662.6mw for 10Mbps bitrate, respectively.

At the same time, Figure 3.9 and 3.10 depict that higher bitrate and larger size of streams take significantly more CPU resources. Bitrate is the number of bits processed per unit of time. Although the bitrate of the video files is the same, the CPU time increases to process more bits when the file size increases. If the bitrate of files increases, CPU needs more time to process the packet. Hence, the CPU time increases with larger file size and higher bitrate. In Figure 3.11, the CPU time increases from

36.893s (3.4MB file) to 151.48s (27.3MB file). The CPU time is 36.718s for 0.5Mbps bitrate file and 53.848s 10Mbps bitrate file, respectively, as shown in Figure 3.10.

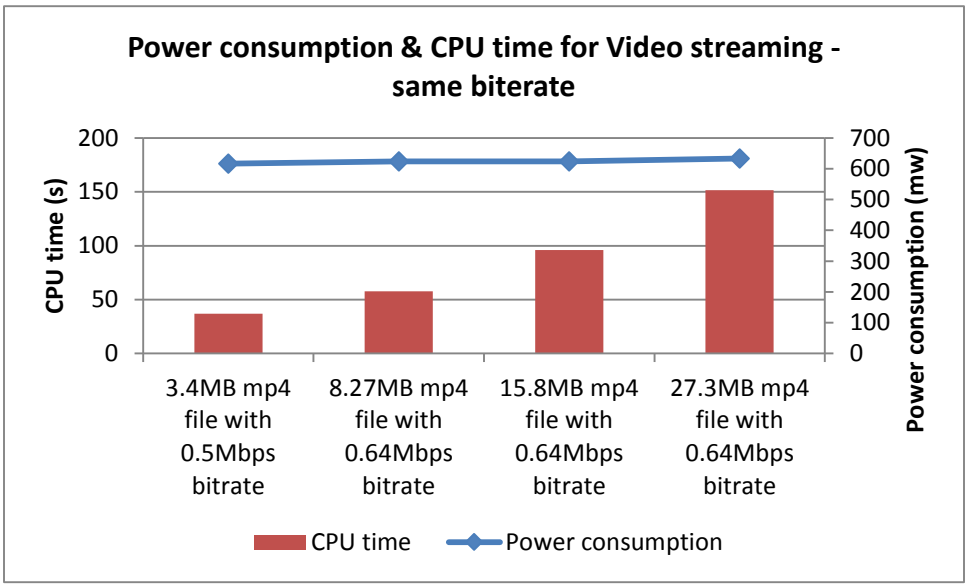

Figure 3.9 Power Consumption & CPU time for Video Streaming (same bitrate)

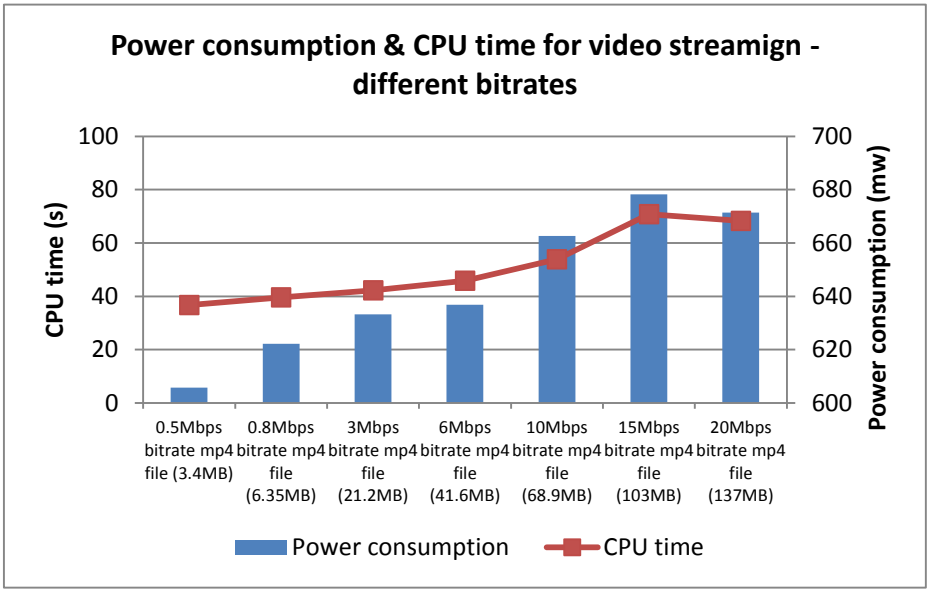

Figure 3.10 Power Consumption & CPU time for Video Streaming (different bitrate)

Figure 3.11 shows that the memory usage is the same (within the allowable error) for video streaming applications. The memory usage is 54.228MB (0.5Mbps bitrate with 3.4MB file size), 54.812MB (0.64Mbps bitrate with 8.27MB file size), 56.91MB (0.64Mbps bitrate with 15.8MB file size), and 55.416MB (0.64 Mbps bitrate with 27.3MB file size), respectively. However, more memory usage is needed for higher bitrate of video streaming as shown in Figure 3.12. Memory usage is 53.918MB (0.8Mbps with 6.35MB file size) and 75.814MB (15Mbps with 103MB file size). Comparing Figures 3.12 with 3.13, the memory usage is almost the same for the same files even under different network speed. It shows that the signal strength have no relationship with the memory usage. For example, memory usage is 53.918MB under the strong signal strength and 54.228MB under the fair signal strength for the same file 0.5Mbps with 3.4MB file size.

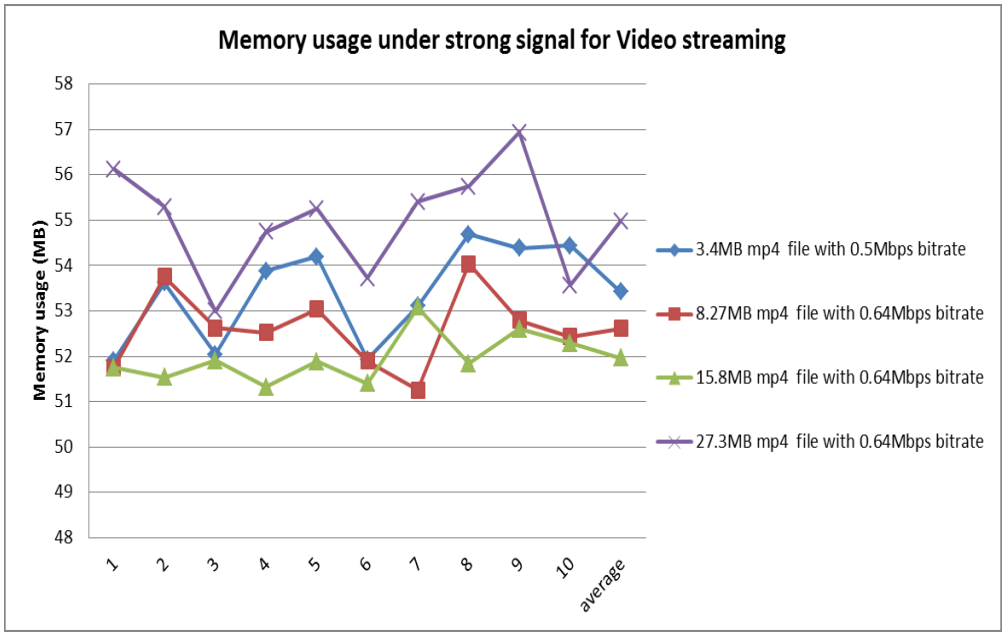

Figure 3.11 Memory Usage for Video Streaming (same bitrate)

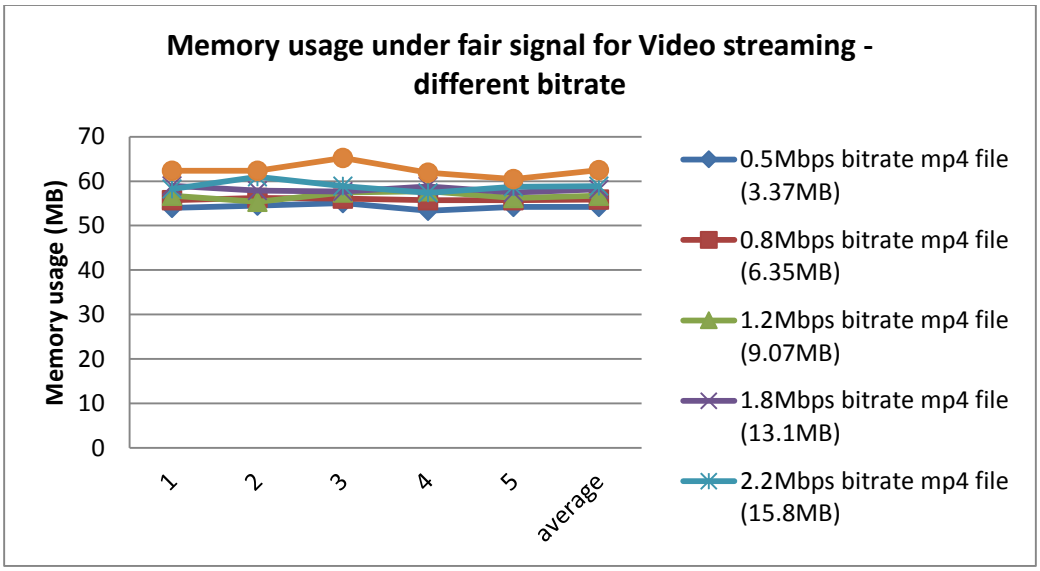

Figure 3.12 Memory Usage for Video Streaming (different bitrate)

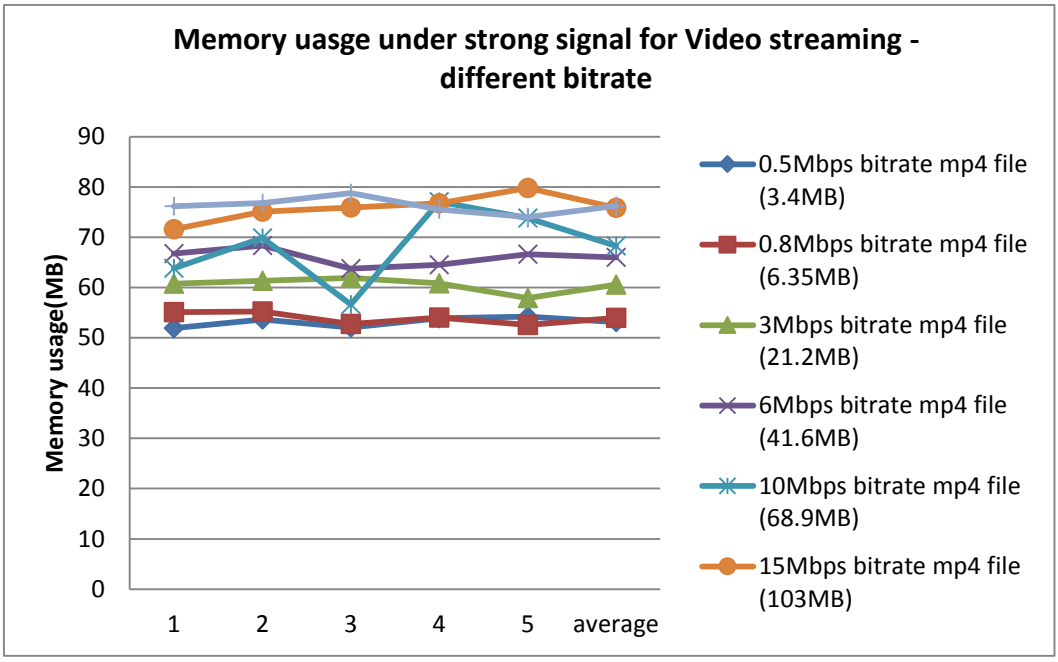

Figure 3.13 Memory Usage for Video Streaming (different bitrate under strong signal)

After examining some MCC applications and having a better understanding of energy consumption and performance for various applications with appropriate tools, the next step is to develop a mechanism to effectively manage energy consumption of mobile applications while maintaining desirable performance. Computation offloading, an effective energy saving technique, is needed for computation intensive applications. Computation offloading is discussed in Chapter 4.

Chapter 4: Computation Offloading Experiments and an Adaptive Algorithm for Computation Offloading

A number of experiments have been performed for MCC, as presented in Chapter 3, to investigate the trade-off between energy consumption and performance for a tablet. When the computation cost is high, one approach that has been advocated by researchers is to offload the computation operations to the cloud to save energy. But offloading to the cloud comes with higher delay, especially if the cloud center is geographically far from the mobile device. Section 3.3.1 presented experimental results based on the idea of offloading to the cloud. The same idea has also been reported in the literature, e.g., [Kumar10][Jiao13] [Mtibaa13].

However, the main caveats of offloading to the cloud are potential high communication delay and the energy consumption for communications using Wi-Fi, especially if the file size is large and bandwidth of Wi-Fi is low. For example, Figure 3.3 shows that energy consumption of the mobile device is high, since high power consumption for 6.8MB file size and high network latency under fair signal. To reduce the delay and energy consumption due to communications to the cloud, Bluetooth can be considered to connect to a nearby PC as an alternative as Bluetooth consumes lower energy than that of Wi-Fi [Kalic12]. Some studies also have shown performance and energy consumption results for mobile device using different wireless technologies, including Bluetooth [Mtibaa13][Sharafeddine13]. This chapter presents an idea that it is not always necessary to offload all computations to a remote cloud server from a mobile

device mainly due to the high communication delay that may be generated. Instead, the offloading target can also be a nearby PC via a lower communication cost mechanism, such as Bluetooth, depending on the computation demands. The chapter also presents a method that is adaptive to various computational requirements and the mobile device's proximity network conditions.

This chapter is organized as follows. Section 4.1 describes a method of computation offloading, which is used in the experiments. Bluetooth and Wi-Fi communication programming using the method is also introduced in this section. Section 4.2 presents three Android applications to carry out computation offloading experiments using an Android mobile device. Section 4.3 demonstrates detailed analyses of measurement results for computation offloading. Section 4.4 presents an adaptive algorithm, which is developed based on the experiments, and an Android application using the adaptive algorithm.

4.1 The model of computation offloading

As discussed in Section 2.3, there are three approaches, client–server communication, virtualization and mobile agent, in offloading method used in MCC systems. The main advantage of client–server communication is well support APIs. Based on the advantage, the model, as shown in Figure 4.1, is chosen to conduct computation offloading in the experiments.

In the client-server model, process communication is established through a socket based program between one server and one or more clients. In the experiments, the client

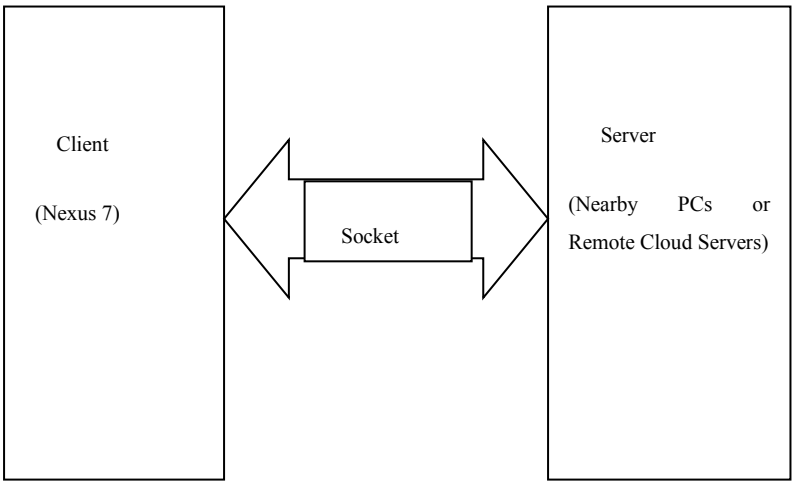

Figure 4.1. Client - Server Connection

is the test mobile device–Nexus 7 and the server can be a nearby PC, e.g., a laptop or a desktop, or a remote cloud server, e.g., an Amazon cloud server. The socket associates the server with a specific port where the client can communicate with the server. Another socket associates the client with the same port as the server's port number. The server provides resources to the client and always listens for requests from clients. When the client sends a connection request to the server, the server accepts the request. Upon acceptance, a connection of the communication is established and the socket manages the connection and provides communication services to the server and the client. The socket programming establishes bidirectional communications between the server and one client. Therefore, the client and the server are ready for communications. To communicate, the client and the server each reads from and writes to the sockets bound to the connection.

This method, which transforms computation process from the resource constrained mobile devices to an external resource-rich device, is an approach to mitigate

the constraints of mobile devices. The transformation can be done through different wireless communication technologies, 3G, 4G, Bluetooth and Wi-Fi. Since no other mobile devices are considered as external computation devices, only Bluetooth and Wi-Fi are used and computation is offloaded to more powerful computation resources in the experiments.

If the communication uses the Wi-Fi technology, a TCP socket is used. TCP provides reliable and in-order transfer of bytes between clients and servers. This is why TCP sockets, rather than UDP sockets, are chosen in the experiments. Server process must be running firstly and a server creates a listening socket with a specific port number that waiting for client's connection request in the process. In the client side, a client socket specifying an IP address and a port number of the server, is created and a connection request is sent. The server socket accepts the request and the connection between the client and the server is established. So the communication service via Wi-Fi is ready for the client and the server.

If the communication uses the Bluetooth technology, a Bluetooth socket [Btsocket14] is used. The most common type of BluetoothSocket is Radio Frequency Communication (RFCOMM) protocol, which is the type supported by Android APIs. RFCOMM is a connection-oriented, streaming transport over Bluetooth and is used for providing a simple reliable data stream to end users. The Bluetooth Serial Port Profile (SPP) is based on this protocol. SPP [Spp14] defines how to set up virtual serial ports and connect two Bluetooth enabled devices. After a server process is running, a listening server socket is created on the server side. On the client side, a single BluetoothSocket is used to initiate an outgoing connection and to manage the connection. When a connecting

from the client is accepted by the server, a communication connection is established. The connection is ready for data transfer between the server and the client, such as providing *InputStream* and *OutputStream* Objects.

Once the method of computation offloading is selected, the next step is to develop Android applications for computation offloading experiments.

4.2 Android applications for offloading experiments

In this section, approaches using offloading and no offloading for Android applications are discussed. Three computation resources in the experiments are listed in the section. Firstly, the necessary tools to develop offloading for Android applications are introduced.

Android applications are usually developed in the Java programming language using the Android Software Development Kit (SDK). The Android SDK provides software developer the API libraries and tools necessary to build, test and debug apps for Android [Sdk14]. The officially supported Android SDK in Integrated Development Environment (IDE) is Eclipse using the Android Development Tools (ADT) bundle. The bundle includes the necessary Android SDK components and different version of the Eclipse IDE. After coding and compiling, Android applications are packaged in the .apk (Android package file) format. Developers install the .apk application in mobile devices via Universal Serial Bus (USB) cable and the application is ready for use.

4.2.1 Computation resources

In comparison of energy consumption for mobile applications, the experiments are performed with three different computation resources:

- Computations are performed locally on a mobile device.
- Computations are performed in a nearby PC with data transferred between a mobile device and the PC via Bluetooth.
- Computations are performed in a remote cloud server with data transferred between a mobile device and the server via Wi-Fi.

If computations are performed locally on a mobile device, offloading will not happen. For the other two computation resources, e.g., a nearby PC and a remote cloud server, the computation will be offloaded to the resources. As hardware capacities of computation resources can greatly affect energy efficiency and application performance. It is necessary to present the system specifications of these resources that have been used for the experiments, which is illustrated in Table 4.1.

As shown in Table 4.1, the nearby PC has the most powerful computation capacities. The remote cloud server, created as medium Amazon EC2 instance type, has more powerful capacities than the mobile device. Amazon EC2 provides customers a wide selection of instance types. The processor selected, a medium instance type, for the experiments, is cost effective and can provide a reasonable performance. Based on the three computation configurations, three applications named *CompuInAndroid, CompuViaB*T and *CompuViaWiFi*, are implemented on the Nexus 7. *CompuInAndroid* is used to run an application on the Andriod device, *CompuViaB*T is used to offload the

Table 4.1. System Specifications of Computation Resources Used for Experiments

	The Mobile Device	**The Nearby PC**	**The Remote Cloud Server**
Device name	Google Nexus 7	Lenovo Edge E420	Amazon EC2 m1.medium
CPU	Quad-Core Tegra 3 processor @1.3GHz	Intel i3-2350 @2.3GHz	Intel Xeon E5-2650 @2.00 GHz
RAM	1 GB	6GB	3.75 GB
Operating System	Jelly Bean Android 4.2	64 bit Windows 7	32 bit Windows Server 2008
Wireless	• Wi-Fi 802.11 b/g/n • Bluetooth	• Wi-Fi 802.11 b/g/n • Bluetooth V2.0 + EDR	N/A

the computation to a nearby PC via Bluetooth, and *CompuViaWiFi* is used to offload the computation to a cloud server using Wi-Fi, respectively.

4.2.2 Computation offloading applications

All of aforementioned three Android applications have been tested in experiments and they are described further as follows:

- *CompuInAndroid*: A file, saved in the tablet, is retrieved from the device. Then an application is launched and preformed on the tablet. The computation result of the application is displayed on the tablet.
- *CompuViaBT*: A file is transferred from the tablet to a nearby PC via Bluetooth. Then the PC performs the application and the tablet displays the computation result that is sent back from the PC via Bluetooth.
- *CompuViaWiFi*: A file is transferred from the tablet to a remote cloud server via Wi-Fi. Then the server runs an application and the tablet displays the computation result that is sent back from the server via Wi-Fi.

It can be seen that *CompuInAndroid* does not perform offloading operation. *CompuViaBT* and *CompuViaWiFi* offload computations to an external device via Bluetooth or Wi-Fi. These applications (*CompuInAndroid, CompuViaB*T and *CompuViaWiFi)* allow a user to create a task which requires different computation loads in Million Floating Point Operations (MFLOP) and select the amount of data (in bytes) to evaluate various applications tasks and the performance results. In the experiments, different offloading techniques with two parameters, computation load and the amount of data to be transferred, are evaluated.

To quantity the computational operations of an application in the experiments, the Java benchmark of computation workload instead of real applications is used. The benchmark [Docjar14] is:

```
float alpha = 2.123456789f;
// By default 2.123456789 is type double which is represented using f at the end//
float  a = 1.1f, b = 2.2f, c = 3.3f,  d = 4.4f, e = 5.5f;
```

```
float = 6.6f, g = 7.7f, h = 8.8f, i = 9.9f; j = 10.10f;
a = a * alpha;
b = b * alpha;
c = c * alpha;
d = d * alpha;
e = e * alpha;
 f = f * alpha;
g = g * alpha;
h = h * alpha;
i = i * alpha;
j = j * alpha;
```

The above computation is a block of ten instructions. In the block, there are 20 read operations (ten fetch operations for alpha and one fetch operation for each of a, b, c, d, e, f, g, h, i, and j) from memory and 10 memory write operations (for saving a, b, c, d, e, f, g, h, i, and j). The block includes 10 float point operations. If the loop number is iterated N times, then the total number of MFLOP is 10*N/1,000,000 mega float point operations.

The reason to use MFLOP, as adopted in the literature, is to mimic different numbers of floating point operations and facilitate comparative studies for various applications. For example, 20 MFLOP is the approximate complexity of a face recognition application and 60 MFLOP is the approximate complexity of a virus scan application against a library of 1000 virus with 1MB memory size as presented in [Mtibaa13]. Actually, more complex computations, e.g., 200, 500, 800 MFLOP, have also been conducted to evaluate possibly more complex computational needs in the future. The amount of data, considered as offloaded data, is also another factor that affects energy consumption of mobile devices and application performance, since the

data need to be transferred between the mobile device and external computation devices for offloading. For the *CompuInAndroid* application, the files needed already reside in the device and hence data transfer is not necessary.

4.3 Experimental results for computation offloading

In this section, experiment scenarios will be described first. Then, the detailed analysis of the experiment results will be presented.

4.3.1 Experiment scenarios

Table 4.2 summarizes the experiment scenarios, including computation load, the size of files to be transferred, general description of the experiment and to be measured metrics. These metrics will be used to evaluate energy efficiency and application performance. It also provides users practical insights to facilitate decision making of adaptive offloading algorithm based on concrete experimental results. The following three figures show the User Interface (UI) of the applications (*CompuInAndroid, CompuViaB*T and *CompuViaWiFi)*. The UI shows a clear view of running the applications.

In scenarios 1, *CompuInAndroid* is performed as shown in Figure 4.2. After the application finishes, computation load and computation elapsed time, will be displayed on the screen of the tablet.

Table 4.2. Experiment scenarios of three computation resources

Scenarios	Description	Metrics measured
1. Computation is executed on the Android device (Nexus 7).	An android application with 20, 60, 200, 500 and 800 MFLOP computation load was performed.	• Energy consumption of the tablet; • Computation elapsed time; • CPU time;
2. Computation is offloaded to a nearby PC from the Android device via Bluetooth.	A file (97KB, 291KB, 484KB, 677KB or 967KB) was transferred from the test tablet to a nearby PC via Bluetooth. As soon as the PC received the file, an application with 20, 60, 200, 500 or 800 MFLOP computation load was performed and then the execution result, such as the value of the computation load and computation elapsed time were sent back to the tablet.	• Energy consumption of the tablet; • Response time; • Computation elapsed time;
3. Computation is offloaded to a remote cloud server from the Android device via Wi-Fi.	A file (97KB, 291KB, 484KB, 677KB or 967KB) was transferred from the test tablet to a remote cloud server via Wi-Fi. As soon as the server received the file, an application with 20, 60, 200, 500 or 800 MFOLP computation load was performed and then the execution result, such as the value of the computation load and computation elapsed time were sent back to the tablet.	• Energy consumption of the tablet; • Response time; • Computation elapsed time; • CPU time;

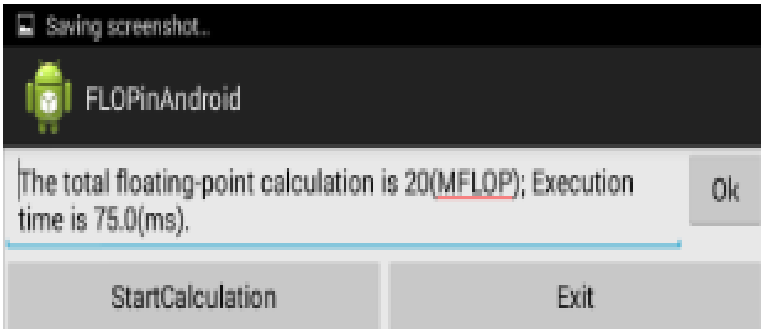

Figure 4.2 Computation result of CompuInAndroid

In scenarios 2, *CompuViaBT* is performed on a nearby PC, as shown in Figure 4.3. The figure shows a paired device of the tablet, which is a computational device available via Bluetooth. Computation load, response time and computation elapsed times, will be shown on the screen of the tablet after the computation is completed.

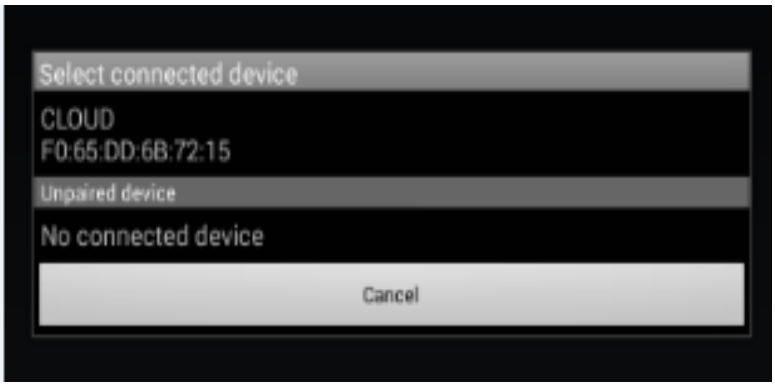

Figure 4.3 UI for CompuViaBT

In scenarios 3, *CompuViaWiFi* is performed in the remote cloud, as shown in Figure 4.4. The figure shows the UI for computation offloading application via Wi-Fi. When a user clicks "Connect and Execute", the client–server connection will be established and an offloading application is performed.

Figure 4.4 UI for CompuViaWiFi

Performance metrics that are measured for these three scenarios are energy consumption of the tablet, response time and CPU time. The measured results showed in this chapter is the average of 5 runs. In the experiments, PowerTutor [Power14] has been used to measure the energy consumption of mobile devices. The measurement results of energy consumption of mobile devices are approximate values as the power model used for PowerTutor in the experiments is not built specifically based on the test mobile device - Nexus 7. As mentioned in Chapter 3, SpeedTest and Android Tuner have been used to measure network throughput and CPU time in computation offloading experiments. For offloading scenarios, response time, including communication time and computation time, is also measured. These metrics will be used to evaluate the energy efficiency of the mobile device and the application performance. The results are then used in determining the thresholds that will be used for offloading decision making of the adaptive offloading algorithm.

4.3.2 Experimental results and analysis

4.3.2.1 Energy consumption

Figure 4.5 – Figure 4.9 plots the results of energy consumption of the mobile device versus the amount of data to be transferred with different computation complexities. Taking Figure 4.5 as an example, legend "Computation in Android device" stands for the computation to be performed on the mobile device locally without file transfer. The other two options mean that the computation is performed on a nearby PC or a remote cloud server after offloading computation from the mobile device.

In Figure 4.5, the blue line, showing energy consumption of the Android device when computations are preformed on the Android device, is always lower than the red and green lines which show the energy consumption of the mobile device when computations are performed on the nearby PC and the cloud server, respectively. The results illustrate that the Android device consumed the least amount of energy without offloading compared to offloading for light computation load, e.g., <= 20 MFLOP. If the computation load and the amount of data are the same for all of these three applications (*CompuInAndroid, CompuViaB*T and *CompuViaWiFi)*, the mobile device transferring data with Bluetooth consumed less energy compared to transferring data with Wi-Fi as Bluetooth has higher energy efficiency than Wi-Fi [Kalic12]. This can be seen that the offloading via Bluetooth line is lower than the offloading via Wi-Fi line in Figure 4.5. In other words, computation offloading to an external computation device is not efficient in term of energy saving for the mobile device if computation load is less than or equal to 20 MFLOP for the devices that are used for the experiments. The results obtained from the experiments help us choose 20 MFLOP as a threshold in the adaptive method which

will be presented in Section 4.2.2. This is because that 20 MFLOP is the approximate complexity of a face recognition application.

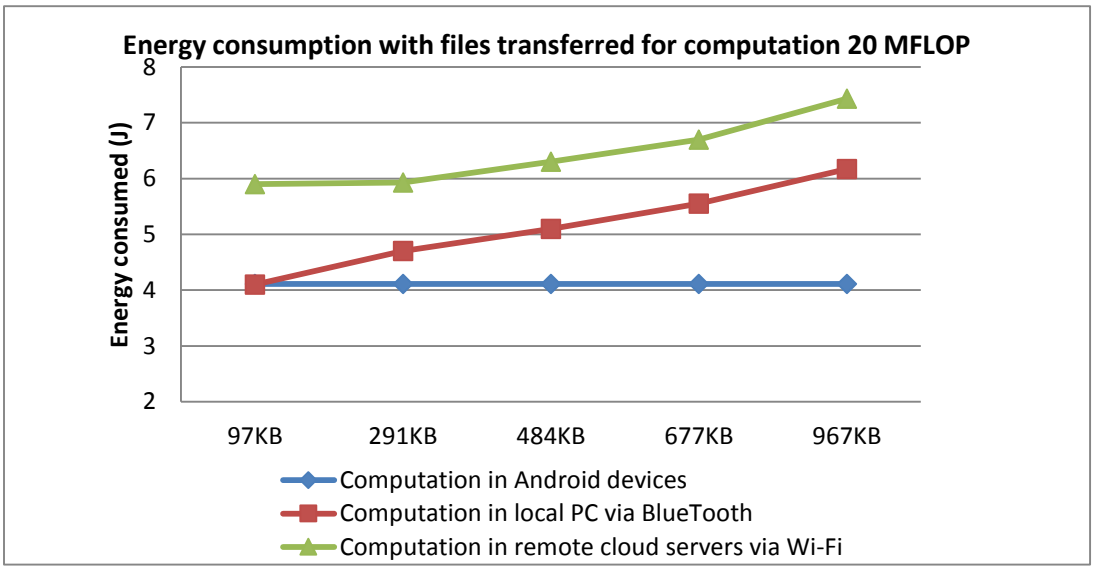

Figure 4.5 Energy Consumption for 20 MFLOP

For the computation performed on the Android device, the device consumed almost the same amount of energy (as shown in Figure 4.5 Computation in Android devices line) with different file sizes. This is because that the file has already stored in the device and no file delivery is needed. It can be said that the energy consumption of the mobile device is only relative to the computation load for Scenario 1. This is can be seen from Figure 4.5 — Figure 4.9. Figures 4.5 - 4.9 also show that the energy consumption of the mobile device increases as the size of files increases, as file transfer is needed. The reason is that a larger file transferred needs more energy consumption of the mobile device and the experiments in Section 3.3.1 have demonstrated the phenomenon already. As presented in Figures 4.5 – 4.9, the energy consumption of the mobile device using Bluetooth offloading to a nearby PC is less than that of using Wi-Fi offloading to a

remote cloud server. Kalic, et al. [Kalic12] also present that Bluetooth has higher energy efficiency than Wi-Fi.

In Figure 4.6, offloading to a nearby PC via Bluetooth (Scenario 2) can save the mobile device's energy when the transferred file size is smaller than 500 KB. If the file size is larger than 500 KB, offloading does not reduce the energy of the mobile device, since energy cost for file transfer is higher than that of the computation energy savings from offloading. The energy consumption of the mobile device when the file size is 291 KB is lower than that of transferring 97 KB file. The fluctuation of Wi-Fi network throughput also affects the results. For offloading computation to a remote cloud server (Scenario 3), energy consumption of the mobile device is always higher than that of no offloading. In other words, it is not worth offloading computation operations to a cloud server via Wi-Fi when computation load is less than 60 MFLOP. Hence, computation offloading to the nearby PC can save energy consumption of the mobile device only when the file size is less than 500 K and computation load is 60 MFLOP.

Figure 4.7 shows that computation offloading to a remote cloud server using medium EC2 instance (Scenario 3) can reduce the energy cost of the mobile device when file size is less than 750 KB. The reasons to choose the medium EC2 instance are lower cost and it also has similar configuration to the PC used in the experiments. For a nearby PC connecting with the mobile device via Bluetooth, offloading can save energy cost of the mobile device, only when the file size is less than 900 KB. Figure 4.7 shows that

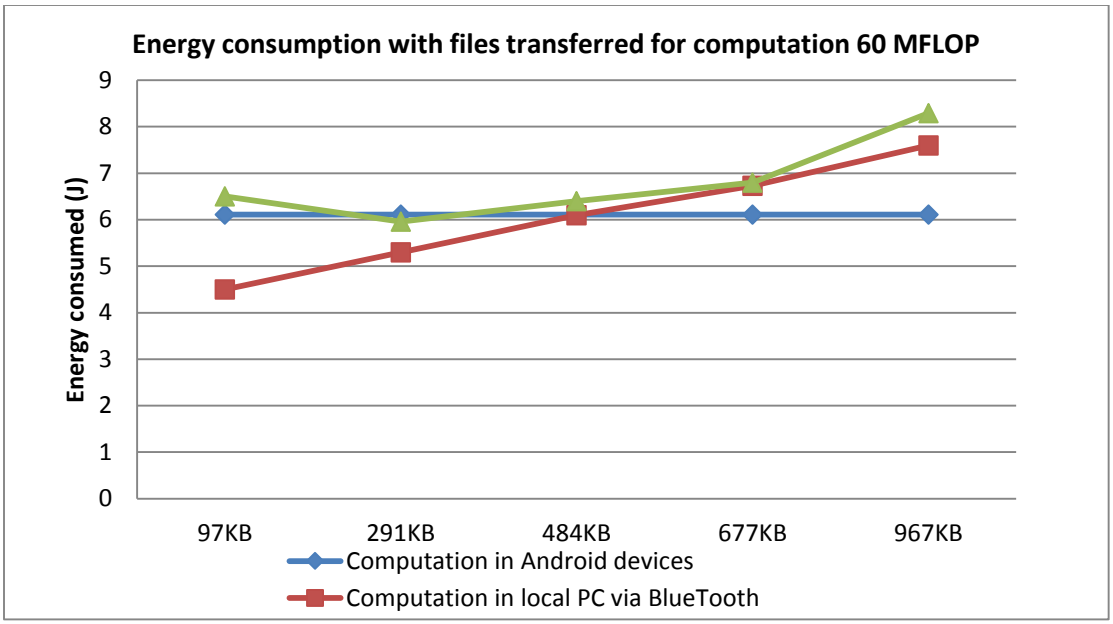

Figure 4.6 Energy Consumption for 60 MFLOP

fluctuating Wi-Fi network bandwidth which results in different energy consumption of the mobile device. Therefore, computation offloading to a nearby PC when the size of files is less than 900 KB, and to a remote cloud server when the size of files is less than 750 KB can save energy of the mobile device, if computation load is 200 MFLOP.

It can be seen from Figures 4.8 and 4.9 that the blue line is always above the red and green line. This means offloading can save energy of the mobile device when computation load is higher than 500 MFLOP. It can be concluded that computation offloading can reduce energy consumption of the mobile device for highly computation-intensive applications. This is because the computation energy saving from offloading is greater than energy cost for files delivery from the mobile device to the nearby PC or the

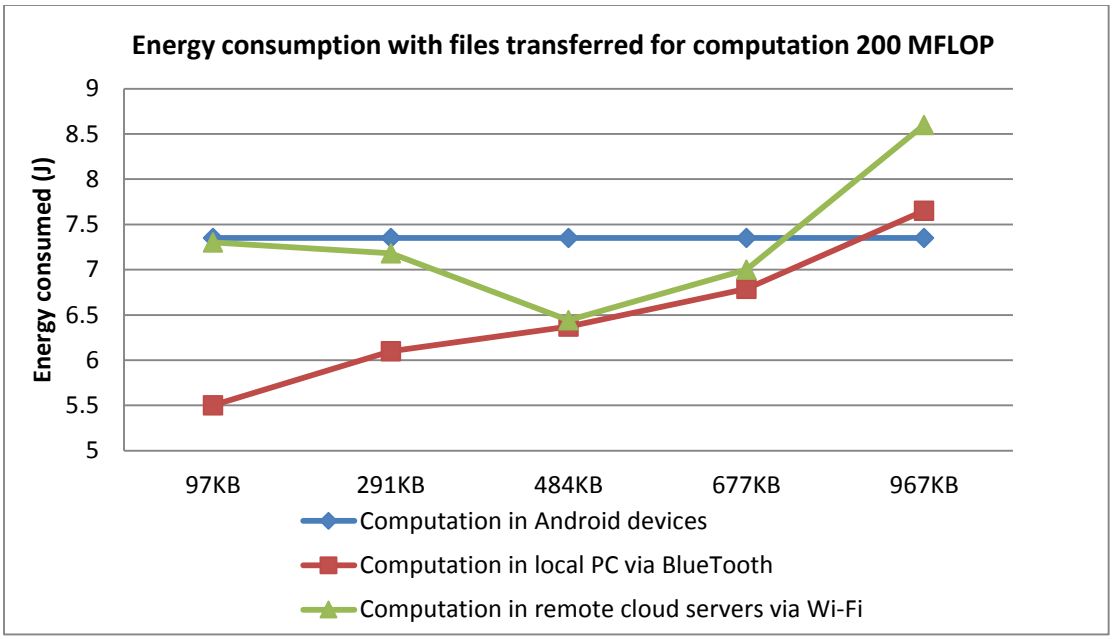

Figure 4.7 Energy Consumption for 200 MFLOP

remote cloud server.

Figures 4.8 and 4.9 show that the difference between offloading and no offloading for 800 MFLOP is larger than that of 500 MFLOP. For example, energy saving for the mobile device, when offloading computation to the cloud server for 800 MFLOP, is 4.52J for 97 KB file size and 1.82J for 967 KB file size, respectively. The energy saving for the mobile device, when offloading computation to the cloud server for 500 MFLOP, is 3.5J for 97KB file size and 0.74J for 967 KB file size, respectively. This means that offloading can save more energy consumption of mobile devices for higher computation load.

Based on results shown in Figures 4.5 - 4.9, Figure 4.10 and Figure 4.11 are

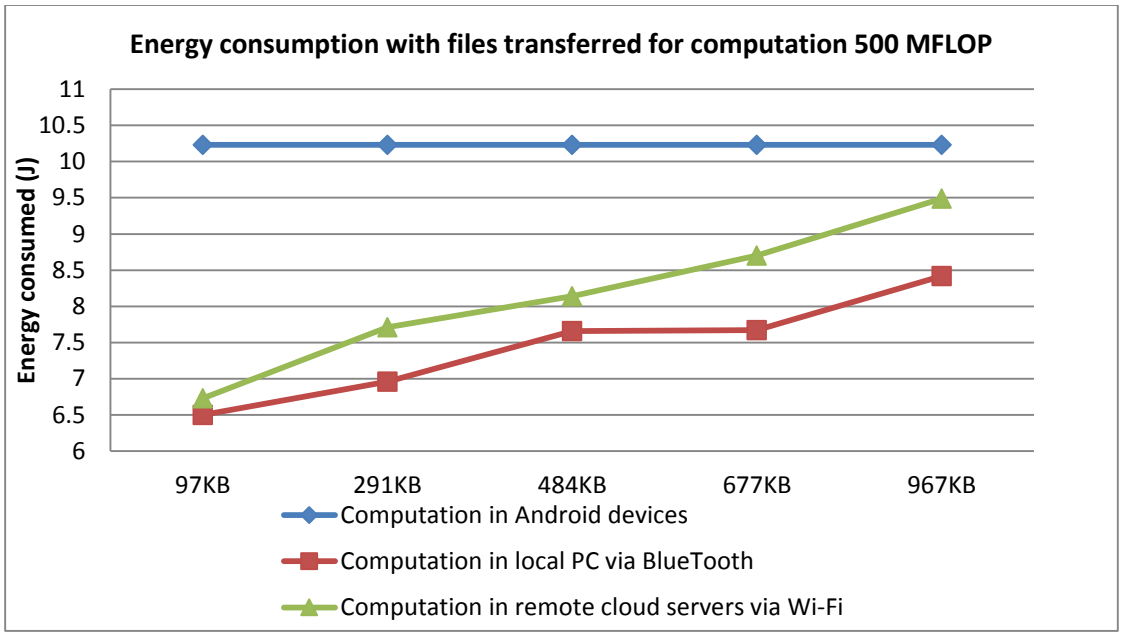

Figure 4.8 Energy Consumption for 500 MFLOP

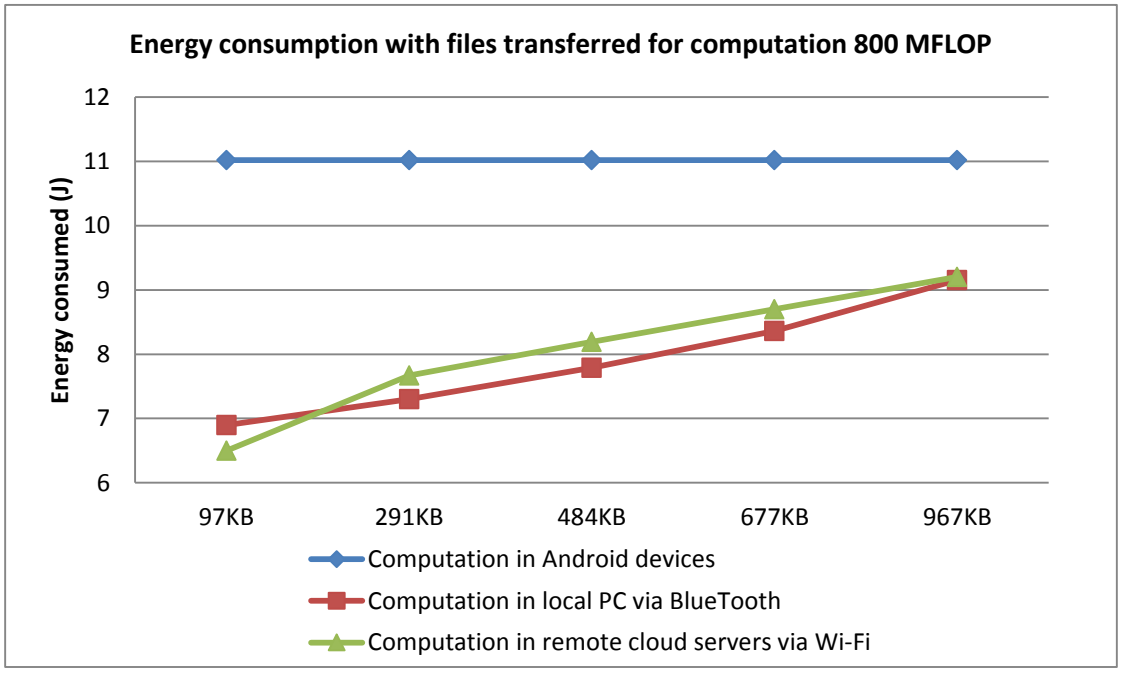

Figure 4.9 Energy Consumption for 800 MFLOP

created to facilitate making the offloading decision. Legends "20_2", "60_2", "200_2", "500_2" stand for computation to be performed on the nearby PC when computation loads are 20 MFLOP, 60 MFLOP, 200 MFLOP and 500 MFLOP. The reason to use different load values (20_2, 60_2, 200_2, and 500_2) is to distinguish curves with the same computation loads for scenario 1 and scenario 2. From Figure 4.10, it can be found that computation offloading via Bluetooth can benefit the mobile device when the size of files is less than 500 KB in 60 MFLOP, or less than 900 KB in 200 MFLOP, or computation load is greater than or equal to 500 MFLOP.

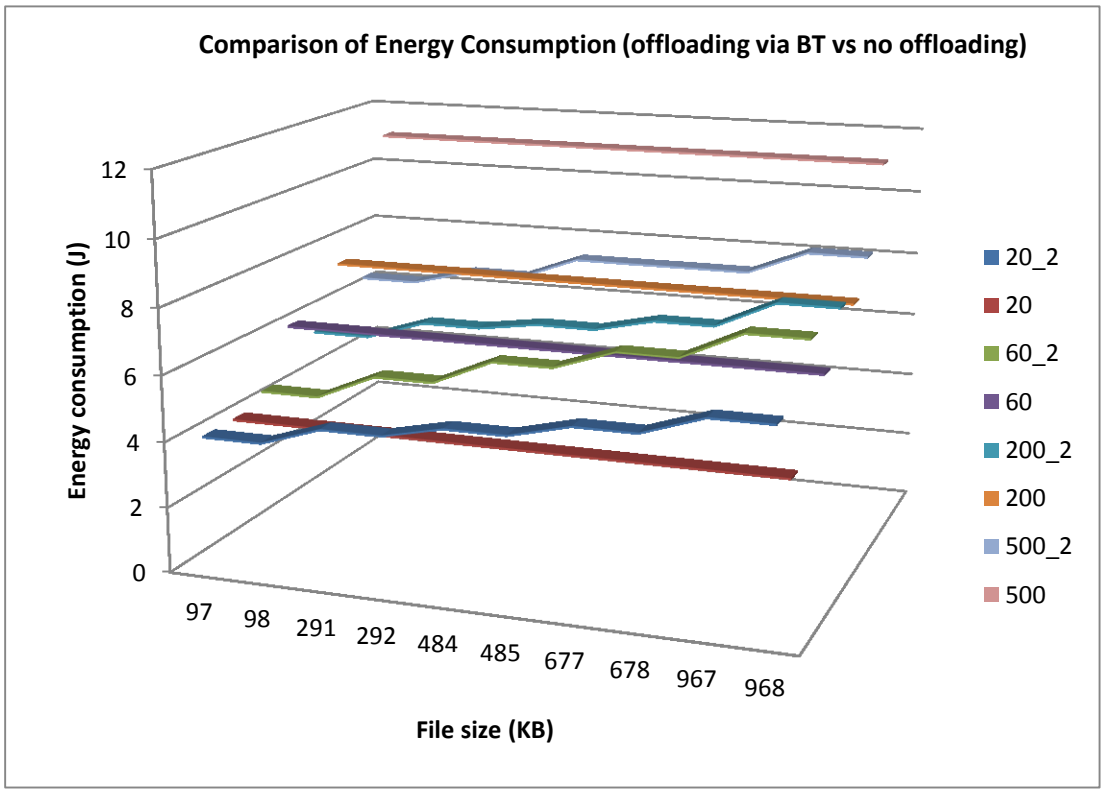

Figure 4.10 Energy Consumption Comparison - 1

The reason using different load values (20_3, 60_3, 200_3, and 500_3) is to distinguish curves with the same computation load for scenario 1 and scenario 3. From Figure 4.11, it can be found that computation offloading via Wi-Fi can benefit the mobile device when the size of files is less than 750 KB in 200 MFLOP, or computation load is greater than or equal to 500 MFLOP. All the conclusions from Figure 4.10 and Figure 4.11 have been used to develop the adaptive algorithm for energy saving of the mobile device.

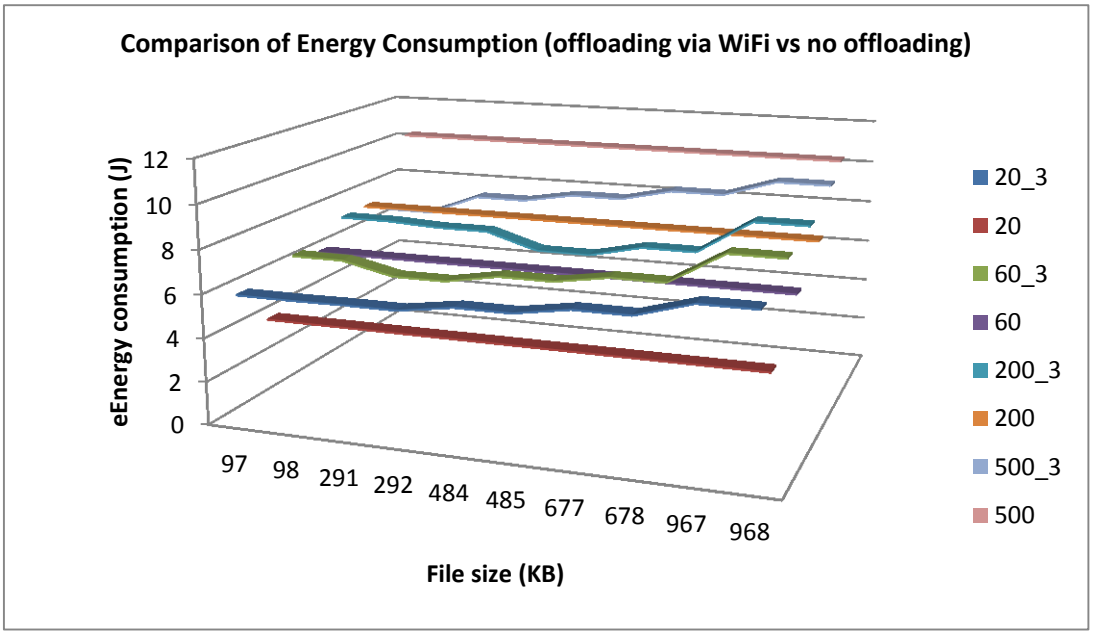

Figure 4.11 Energy Consumption Comparison -2

4.3.2.2 Response time of applications

Though Bluetooth is more energy efficient than Wi-Fi, the response time using Bluetooth is higher than Wi-Fi, since the bandwidth of Bluetooth is lower than that of

Wi-Fi [Diffen14]. Note that the hardware capacities for the EC2 medium instance are a little bit smaller than that of the PC used in the experiments. If higher capacities of EC2 instances are used, the response time is likely to be even smaller. In Figure 4.12, the response time using Bluetooth is 2110 ms, while only 1320 ms using Wi-Fi, for the same file size of 97 KB. More results of response time can be seen from Figures 4.12 - 4.16. The fluctuating Wi-Fi network causes that the response time of offloading using Wi-Fi is higher than that of offloading using Bluetooth for some cases, i.e., transferring 97 KB data in 60 MFLOP and 200 MFLOP. Five runs were repeated at different times of the day to eliminate some network fluctuating effect on experimental results. Figures 4.12–4.16 show that the response time for no offloading is constant since the response time is only the computation elapsed time. This is because the same computation load consumes the same computation elapsed time for the mobile devices.

However, the remote cloud server is created in Northern Virginia data center, which is the closest to Ottawa among all Amazon data centers. Section 3.3.1 shows that geographical location of data center will greatly affect communication time of files transfer. If another farther data center is chosen, the response time for Wi-Fi technology will be closer to that of using Bluetooth, or even larger than that of using Bluetooth.

4.3.2.3 Computation time and the Number of Floating-Point Operations Per Second Performance

Floating-Point Operations Per Second (FLOPS) is a measure of computer performance in [Flops14] and can be obtained by $\frac{FLOP}{Time}$. FLOP is computation load that is

performed. The time is the elapsed time to perform the computation load. In Figure 4.17,

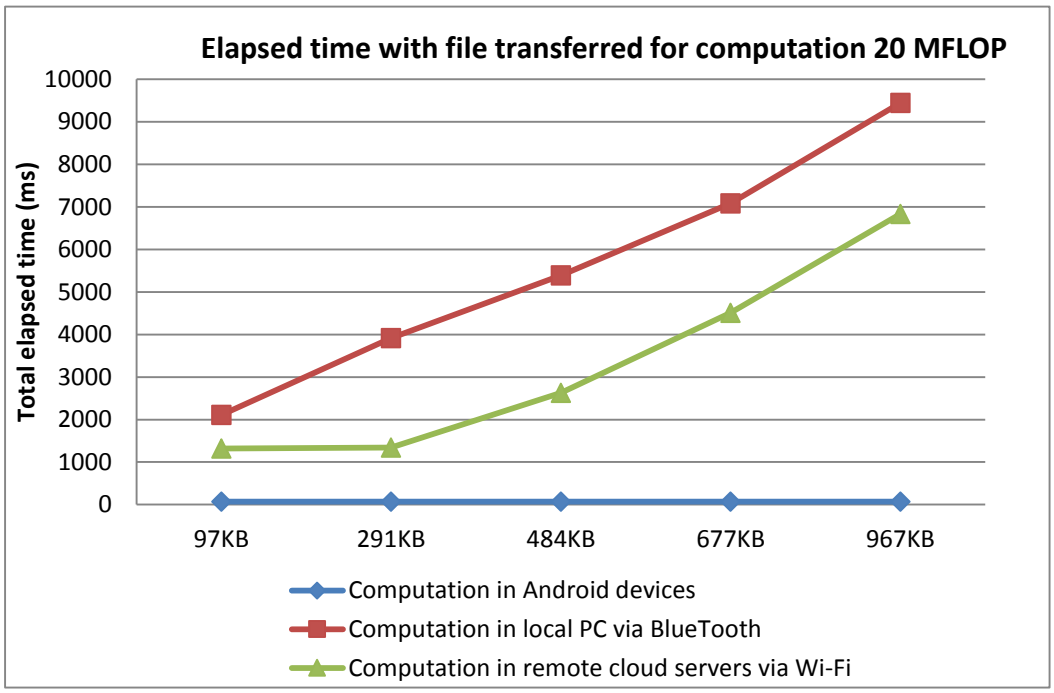

Figure 4.12 Response time for 20 MFLOP

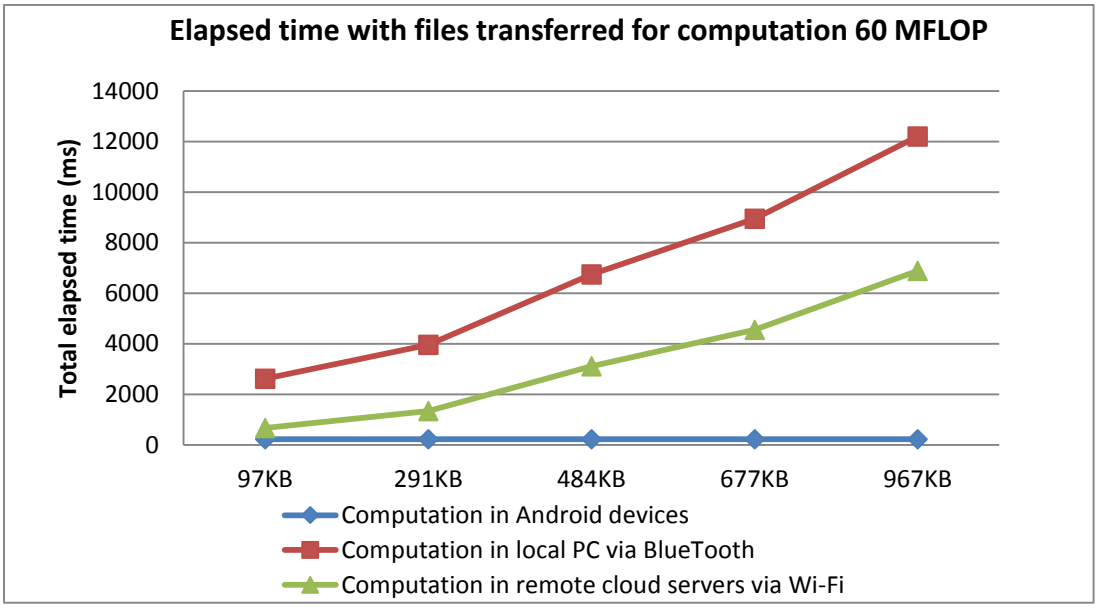

Figure 4.13 Response time for 60 MFLOP

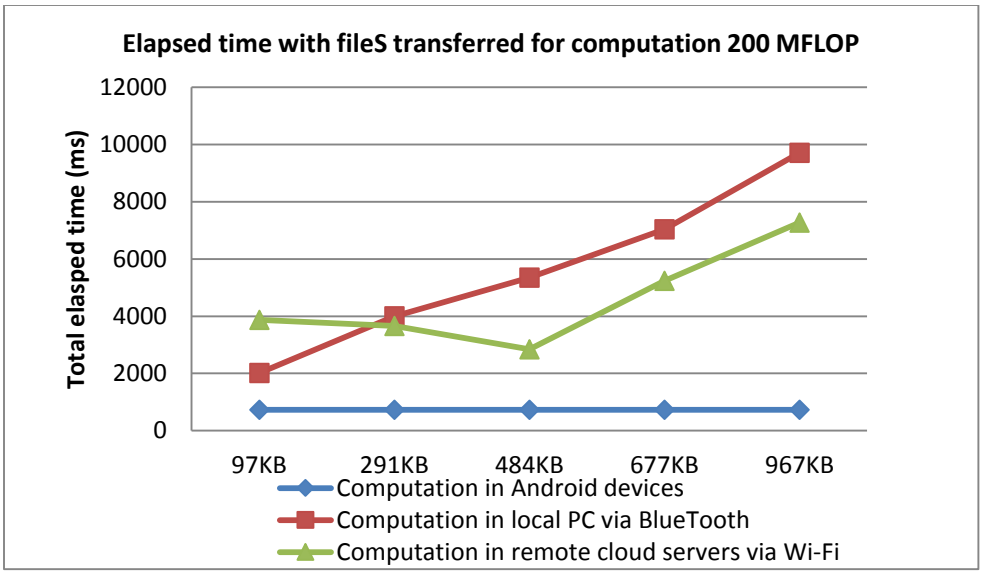

Figure 4.14 Response time for 200 MFLOP

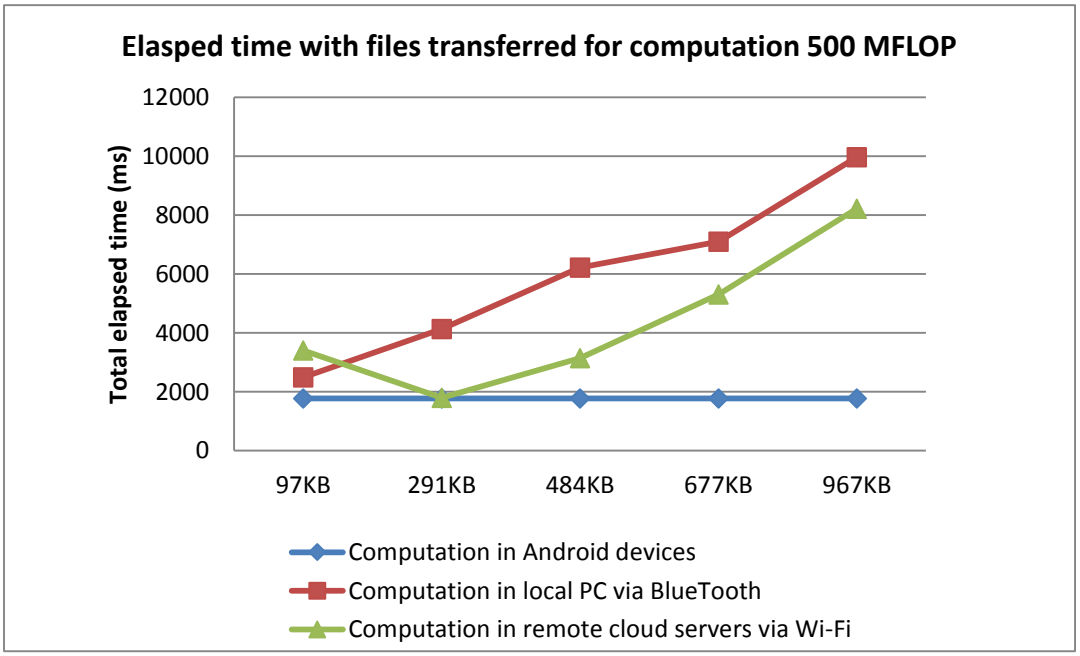

Figure 4.15 Response time for 500 MFLOP

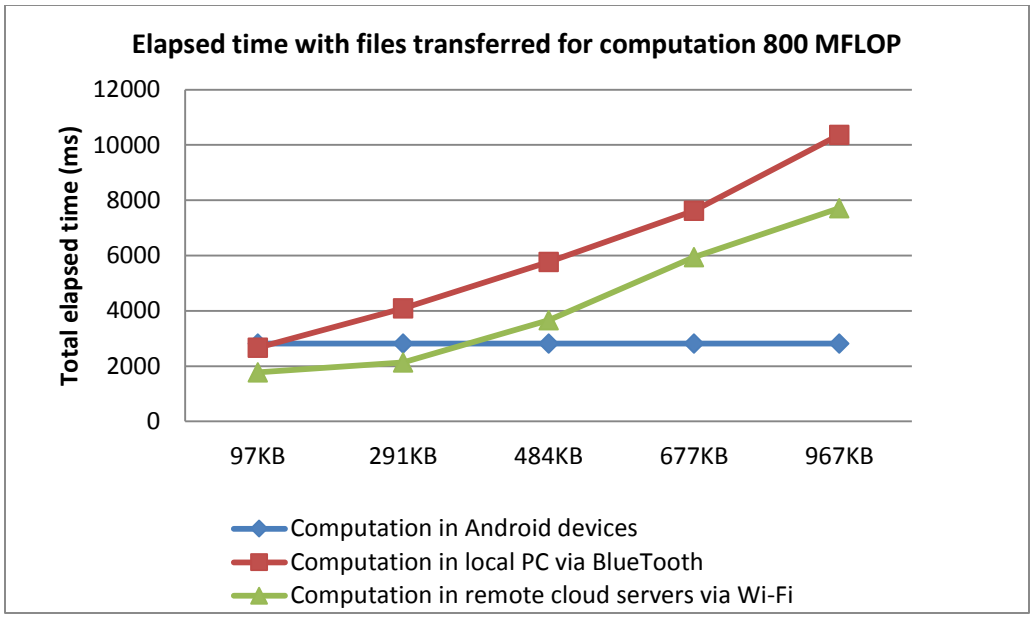

Figure 4.16 Response time for 800 MFLOP

MFLOP is the value of X-Axis and *Time* is the value of Y-Axis in Cartesian Coordinate system. It can be seen that the *no offloading* line (Computation on a mobile device) has the smallest FLOPS and the *offloading via Bluetooth* line (Computation on a nearby PC) has the largest FLOPS. Therefore, the mobile device has the lowest computation performance and the nearby PC has the largest computation performance among these three computation resources. The computation performance of the cloud server is in the middle of these computation resources.

Decision making is one of the key considerations for offloading [Kumar12] and is needed to be carefully consideration. The experimental results have been used to identify thresholds for decision making in the adaptive algorithm which is presented in the next section.

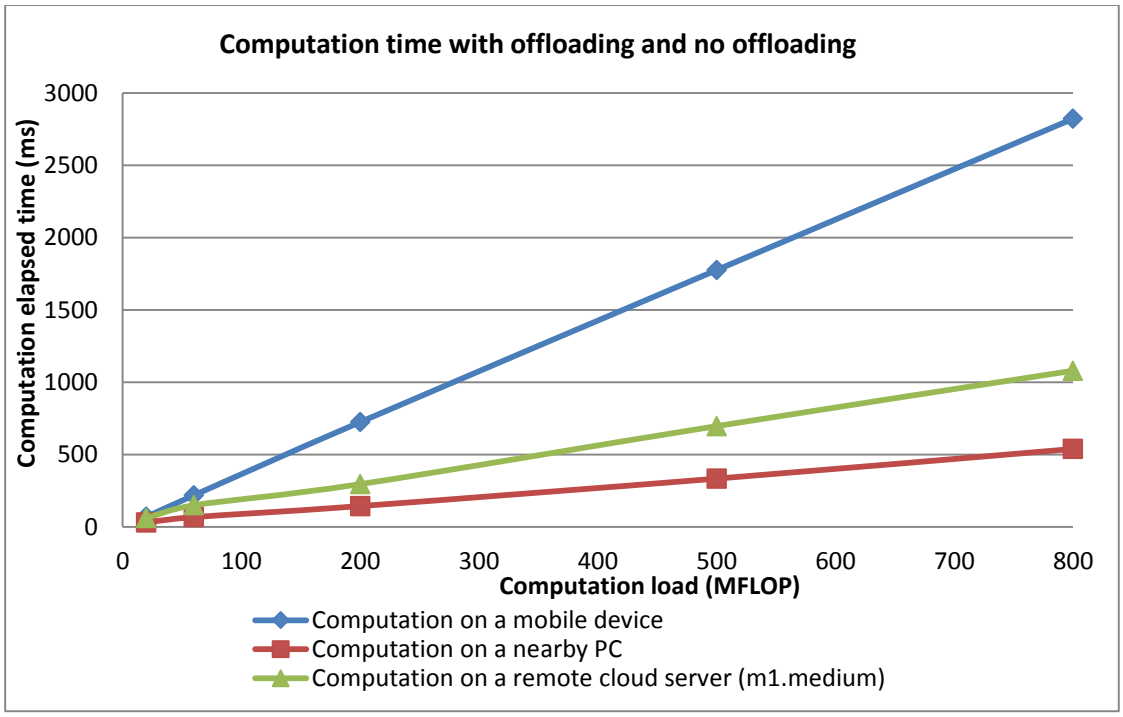

Figure 4.17 Computation Performance of resources

4.4 Adaptive algorithm and its application

As mentioned before, the growing gap between the requirement of energy and computation capacities of mobile devices, and the limited resources of the mobile systems is triggering academic and industry concern on solutions for computation offloading to external computation-rich resources. Energy consumption of mobile devices is the most concern for mobile systems, since existing battery technology limits the widespread use of increasing complex computation applications in mobile devices, e.g., video encoding and decoding, image processing, language processing, etc. The computation resources could be nearby PCs and remote cloud servers. It is critical to

decide whether or when offloading computation operations to a more suitable external resource. Otherwise, mobile devices may not benefit from offloading and hence it is not worth offloading computation from mobile devices to other computational resources. Hence, the offloading decision is discussed to meet the goals of offloading, saving energy consumption of mobile devices and/or improving application performance.

In this section, an algorithm which is adaptive to computation requirements and a mobile device's proximity network condition, is presented based on the experimental results in the previous section. At the same time, an Android application using the adaptive algorithm is developed and used to verify the feasibility and correctness of the algorithm.

4.4.1 The process of computation offloading decision making

The main aim of computation offloading is to save energy of mobile devices, which have limited in constrained resource. As discussed above, no all offloading can save energy of mobile devices and the external computation resource selected generally is more efficient than mobile devices. This thesis investigates energy consumption of the mobile device to facilitate decision making needed in the offloading algorithm. Figure 4.18 describes the process of computation offloading decision making. To judge whether saving a mobile device's energy cost or not, the equation 2.2 in chapter 2 should be greater than 0, i.e.,:

$$\mathbf{E_m - (E_i + E_{tr})}$$

where

- E_m - energy consumed by a mobile device when computation happen in the device;
- E_i - energy consumed by a mobile device in idle state to wait for computation result when computation happen in external computation resources;
- E_{tr} - energy consumed by a mobile device when sending and receiving data between the device and external computation resources;

If the result of equation 2.2 is greater than 0, offloading can save a mobile device's energy. Otherwise, it is not worth offloading. The next step is to develop an adaptive algorithm based on the experiment results. Figure 4.18 illustrates the process of decision making among the three computation devices.

In the figure, the steps of offloading decision making are described as follows.

Step 1. Check if Bluetooth is available. If no, the process goes to step 3. If yes, the process checks if energy saving of a mobile device can be obtained using equation 2.2. If offloading can benefit the mobile device, the process goes to step 2 - offloading computation operation to a nearby PC via Bluetooth. Otherwise, the process decides no offloading operation and goes to step 3.

Step 2. The application of offloading to a nearby PC is launched until the application finishes.

Step 3. Check if Wi-Fi is available. If no, the process goes to step 5. If yes, the

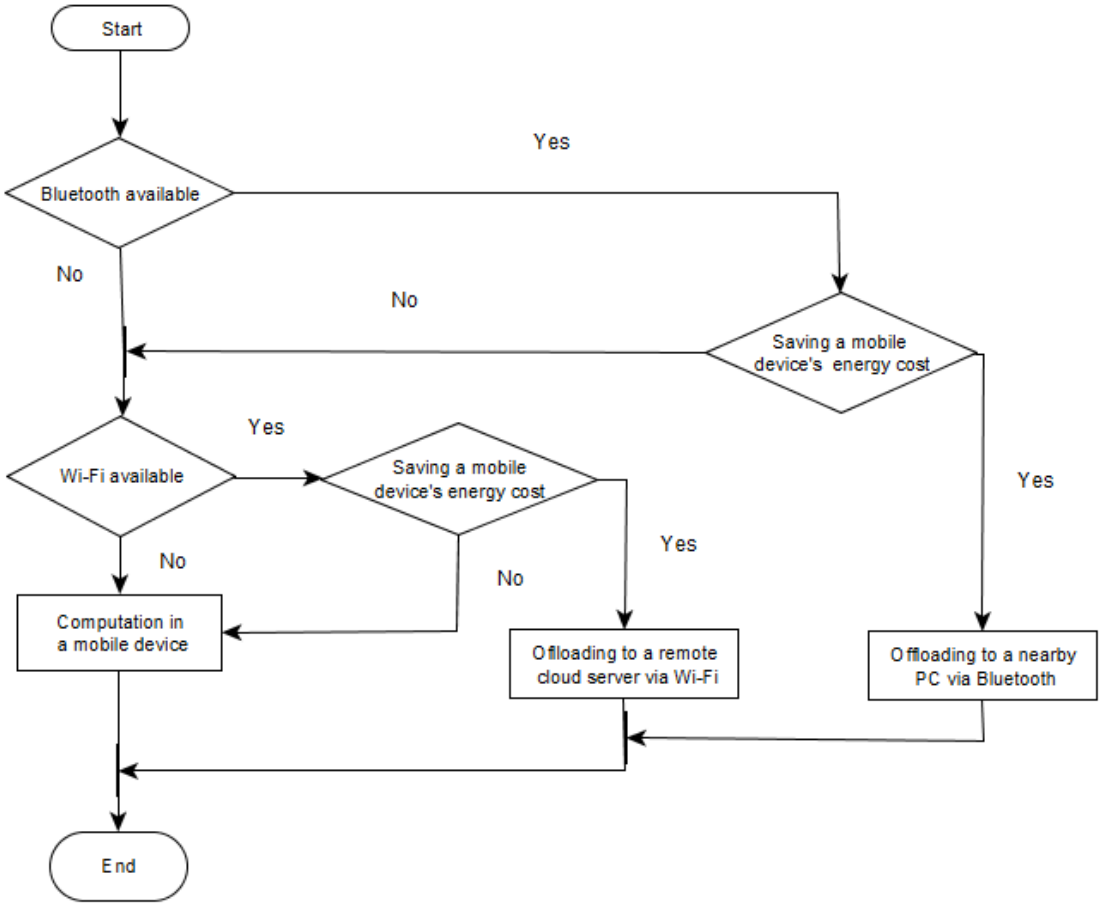

Figure 4.18 Process of offloading decision making

process checks if energy saving of a mobile device can be obtained using the equation 2.2. If offloading can benefit the mobile devices, the process goes to step 4 - offloading computation operation to a remote cloud server via Wi-Fi. Otherwise, the process decides no offloading operation and goes to step 5.

Step 4. The application of offloading to a remote cloud server is launched and until the application finishes.

Step 5. No offloading - computation will be performed in the mobile device.

4.4.2 An adaptive algorithm

The adaptive algorithm is developed following the aforementioned description. The energy consumption of mobile devices in the algorithm has higher priority. If performance of mobile applications is considered in the algorithm, equation 2.3 needs to be combined in the offloading decision making. The algorithm is presented as follows:

****

Algorithm: The Adaptive Offloading Algorithm

****

Assumptions: other computation resources are available: a nearby PC is connected via Bluetooth and a remote cloud server is connected through Wi-Fi.

Objective: select the computation resource and offload computation to the resource in order to save mobile devices' energy consumption.

Constants: FLOP1, FLOP2, FLOP3, SIZE1, SIZE2, SIZE3

// FLOP1, FLOP2, FLOP3 are increasing values of computation load; SIZE1, SIZE2, SIZE3 are increasing values of file size. Those values are threshold values obtained from experiments, which will be used for decision making. //

- *FLOP1 is used to determine offloading to the nearby PC for small file size (500 KB to 900 KB).*
- *FLOP2 is used to determine offloading to the cloud server for medium file size (750 KB to 2 MB)*
- *FLOP3 is used to determine offloading to the nearby PC for large file size (900 KB to 1.8 MB)*
- *SIZE1 is the upper bound of small file size (500 KB to 900 KB)*
- *SIZE2 is the upper bound of large file size (900 KB to 1.8 MB)*

- *SIZE3 is the upper bound medium file size (750 KB to 2 MB)*

INPUT: Computation load (MFLOP) and file size (KB)

OUTPUT: Energy consumption of mobile devices, response time and computation elapsed time

1. **IF Bluetooth is available**

   //If Bluetooth connection is available and energy consumed when computation in mobile devices is greater than the sum of energy consumption of energy consumed when mobile devices is in idle state and energy consumed when a file transferred between a mobile device and a nearby PC. //

2. **IF $E_{com\_in\_mob} > E_{idle} + E_{de\_via\_BT}$**

3. **Offloading to a nearby PC and computation to be performed in the PC**

4. **ENDIF**

5. **ELSEIF Wi-Fi available**

   //If Wi-Fi is available and energy consumed when computation in mobile devices is greater than the sum of energy consumption of energy consumed when mobile devices is in idle state and energy consumed when a file transferred between a mobile device and a remote cloud servers.//

6. **IF $E_{com\_in\_mob} > E_{idle} + E_{de\_via\_WiFi}$**

7. **Offloading to a remote cloud server and computation to be performed in the server**

8. **ENDIF**

9. **ELSE**

10. **No offloading and computation to be performed in the mobile device locally**

11. **ENDIF**

where

- $E_{com\_in\_mob}$: Energy consumption of a mobile device when computation to be performed in the device locally.

- E_{idle} : Energy consumption of a mobile device in idle state to wait for computation result when computation to be performed in external computation resources.
- $E_{de\_via\_BT}$: Energy consumption of a mobile device when file delivery via Bluetooth
- $E_{de\_via\_WiFi}$: Energy consumption of a mobile device when file delivery via Wi-Fi

****

For an Android application, based on the experiments, the inequality

$$\mathbf{E_{com\_in\_mob} > E_{idle} + E_{de\_via\_BT}}$$

will happen when

- (flop >= FLOP3) or
- ((FLOP2 =< flop < FLOP3) and (size < ((SIZE2/615) * (flop + 115)))) or
- ((FLOP1 =< flop < FLOP2) and (size < ((SIZE1/315) * (flop + 115)))).

Another inequality

$$\mathbf{E_{com\_in\_mob} > E_{idle} + E_{de\_via\_WiFi}}$$

will happen when

- (flop >= FLOP3) or
- ((FLOP2 =< flop < FLOP3) and (size < ((SIZE3/476) * (flop - 24)))).

For example, based on the experimental results, the threshold values are FLOP1 = 60 MFLOP, FLOP2 = 200 MFLOP, FLOP3 = 500 MFLOP, SIZE1 = 900 KB, SIZE2 = 1.8 MB and SIZE3 = 2 MB. The setting simplifies the equation, equation 2.2, of energy consumption of a mobile device. In fact, it is more complex to express energy consumption of mobile devices with one equation. The last step is to verify the presented adaptive algorithm.

4.4.3 An Android application based on the adaptive algorithm

Android application has been developed using the adaptive algorithm which will automatically select an offloading destination, i.e., the smart tablet itself, a nearby PC, or a remote cloud servers. The UI of the application is depicted in Figure 4.19. Some hints are shown on the screen of the application.

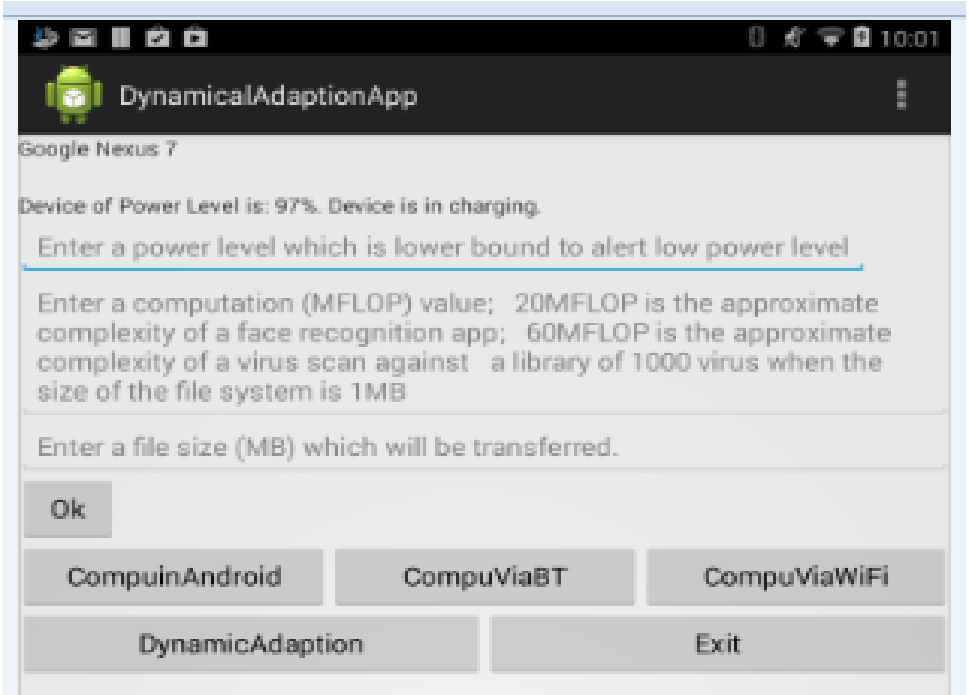

Figure 4.19 UI of an Android Application

- The text "Enter a power level which is lower bound to alert low power level" indicates that the battery needs to charge.
- The text "Enter a computation load" and "Enter a file size" indicates that a user needs to input computation load and the size of a file to be transferred, which are used to make offloading decision.
- The text "20MFLOP and 60MFLOP" just gives an example of computation complexity of applications to users.

Three applications (*CompuInAndroid, CompuViaB*T and *CompuViaWiFi)* can be launched separately. If choosing *DynamicAdaption*, the computation resource, a mobile device, or a nearby PC, or a remote cloud sever, will be automatically selected based on the above adaptive algorithm.

Finally, the Android application, *DynamicAdaption,* has been repeatedly launched and tested with different computation loads and file sizes. The evaluation outcome demonstrates that the adaptive algorithm is functioning properly. That is to say, *DynamicAdaption* can automatically select different computation resources based on computation loads, file sizes and network conditions. The results also are inline with the ones that have been presented in Section 4.4.1, as the threshold values were specifically obtained from the same configurations and application demands.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

MCC have become more popular and the trend is still growing at a fast pace. Some applications, on the other hand, become more complicated and consume more energy. Energy consumption is a key concern for mobile devices. A number of studies have been reported in the literature addressing energy consumption, performance, and offloading. However, there is still a gap in terms of the scope of applications and measurement tools and the performance metrics that those tools can reliably generate, as this area is evolving rapidly.

In the research, a large number of tools have been investigated for performance measurements for various applications. Each of those tools could be used to measure different performance metrics and some of them were limited in terms of accuracy, applicability, or consistency. Only those tools that generate reasonably accurate and consistent performance results were reported in the thesis. Moreover, a wide range of applications, a large set of performance metrics and parameters that may affect the application performance, and different distributed system architectures and wireless technologies have been investigated.

The applications that have been examined in the thesis include video applications, web content download using a centralized server (Amazon S3) or content delivery

networks (Amazon CloudFront), web browsing using Google Chrome and a cloud based browser (SkyFire), computation offloading applications, etc. The main parameters studied for performance impact include signal strength, bitrate, file size or the amount of data to be transferred, computation load, etc. The primary performance metrics covered contain E2E delay, computation elapsed time, CPU time, memory, power or energy consumption, etc. In addition, the performance comparison of Bluetooth and Wi-Fi using a nearby PC or a cloud server has also been conducted.

The main reason for performing a large number of experiments was to better understand the impact of energy consumption of a mobile device and performance results for various parameters for different applications and technologies. Based on the survey, no report in the literature has covered as many applications, tools, parameters, and different technologies as this thesis. The wide coverage provides valuable information in the field. Furthermore, the objective was to identify some basic threshold values that would be useful for decision making for computation offloading for saving energy of mobile devices. The decision making is crucial for computation offloading. An adaptive algorithm was developed to support decision making, which made use of the results obtained from the experiments as threshold values.

5.2 Future work

A few areas warrant further research, as technologies and new applications are evolving in MCC. One future research direction is to investigate computation offloading

using Wi-Fi Direct which directly connects peer to peer devices and has high network bandwidth (up to 250Mbps). Wi-Fi Direct does not need a wireless access point or the Internet, which has competitive advantages than Wi-Fi and Bluetooth. However, Wi-Fi Direct is an immature and emerging technology, and it has not been widely used in wireless communication products. Currently, Wi-Fi Direct is available in Android mobile devices, for example Android 4.0 and up, Samsung Galaxy II with Android 2.3. If Wi-Fi Direct can be applied to wireless network adapters, this will benefit mobile devices through computation offloading to more powerful resources. Hence, it is worth investigating the effect of Wi-Fi Direct on laptops or PCs for computation offloading if the technology becomes stable and available

The second research direction is to make use of virtualization method in more powerful machines, such as nearby PCs and remote cloud servers. In the experiments, the client - server communication is used to create reliable communications for clients and the server. An apparent drawback is the necessity of pre-installed server applications on the server side. The virtualization method can overcome the drawback and has been used in Cloudlet [Jararweh13] for nearby PCs and CloneCloud and MAUI [Jiao13] for the remote cloud server.

The third research direction is to extend the tool developed for the adaptive algorithm to real applications, e.g., face and object recognition, healthcare management, gaming, etc. The current tool only mimics the computation load using MFLOP. Fahim, et al. [Fahim13] present a method to calculate MFLOP of applications and the method can be used to map applications to exact the MFLOP value. The MFLOP value will be

used to help make offloading decision for the application based on the adaptive algorithm discussed in Chapter 4.

The medium EC2 instance has been adopted for the experiments primarily for the cost reason. For highly complicated applications that require a large number of MFLOP, using Amazon Compute Optimized EC2 instance with larger capacities would improve the performance, which warrants further research. The future experiments of computation offloading can include 3G/4G to compare the results using different wireless technologies.

Energy consumption is closely related to hardware devices. Hence, for different hardware configurations or other mobile devices, measurements of new performance data are needed, which can be stored in a performance knowledge base [Woodside07] for further analysis. Moreover, the overhead cost of energy consumption of mobile devices is needed for further investigation, since it is often neglected in the experiments.

References

[Adobe13] Adobe media server user manual. [online]. Available: http://help.adobe.com/en_US/adobemediaserver/amazonec2/adobemediaserver_5.0_amazonec2.pdf, last accessed on September 24, 2013.

[Amazon14] Amazon EC2. [online]. Available: http://aws.amazon.com/ec2/, last accessed on March 20, 2014.

[Android14] Android SDK. [online]. Available: http://developer.android.com/sdk/index.html?hl=sk, last accessed on April 25, 2014.

[Balasubramanian09] N. Balasubramanian, A. Balasubramanian, A. Venkataramani. "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications", *Proc. of the 9th ACM SIGCOMM Conference on Internet measurement*, November 2009, pp. 280-293.

[Barbera13] M. V. Barbera, S. Kosta, A. Mei, J. Stefa, "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing", *Proc. of 2013 IEEE INFOCOM*, April 2013, pp. 1285-1293.

[Btsocket14] Bluetooth Socket. [online]. Available: http://developer.android.com/reference/android/bluetooth/BluetoothSocket.html, last accessed on April 27, 2014.

[Chen12] W. Chen, C. C. Chiu, Y. B. Chang, “A Performance Study on Context-Aware Real-time Multimedia Transmission between Smartphone and Cloud”, *Proc. of Intelligent Signal Processing and Communications System (ISPACS)*, November 2012, pp.211-215.

[Chun11] B. G. Chun, S. H. Ihm, P. Maniatis, M. Naik, A. Patti, "CloneCloud Elastic Execution between mobile device and cloud", *Proc. of the 6th Conference on Computer Systems, April 2011*, pp. 301-314.

[Cloudfront14] Amazon CloudFront. [online]. Available: http://aws.amazon.com/cloudfront/, last accessed on February 12, 2014.

[Compu14] Cloud Computing. [online]. Available: http://en.wikipedia.org/wiki/Cloud_computing, last accessed on February 12, 2014.

[Cuervo10] E. Cuervo, A. Balasubramanian, D. K. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, "MAUI: making smartphones Last Longer with code offloading", *Proc. of the 8th International Conference on Mobile Systems, Applications and Services*, June 2010, pp. 49-62.

[Diffen14] [online]. Available: http://www.diffen.com/difference/Bluetooth_vs_Wifi, last accessed on March 16, 2014.

[Dinh11] H. Dinh, C. Lee, D. Niyato, P. Wang, “A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches”, *Wiley Online Library*, April 2011.

[Docjar14] MFLOP benchmark. [online]. Available: http://www.docjar.org/html/api/musli/aorta/Benchmark.java.html, last accessed on April 27, 2014

[Fahim13] A. Fahim, A. Mtibaa, K.Harras, "Making the case for computational offloading in mobile device clouds", Carnegie Mellon University, Tech. Rep. CMU-CS-13-119, June 2013.

[Fernando13] N. Fernando, S. W. Loke, W. Rahayu, "Mobile Cloud Computing: A Survey", *Future Generation Computer Systems*, July 2013, pp. 1278-1299.

[Flops14] FLOPS. [online]. Available: http://en.wikipedia.org/wiki/FLOPS, last accessed on April 27, 2014.

[Gu12] Y. Gu, V. March, B. S. Lee, "GMoCA: Green Mobile Cloud Applications", *Proc. of the 1st International Workshop on Green and Sustainable Software (GREENS)*, June 2012, pp.15-20.

[Huang10] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, P. Bahl, "Anatomizing Application Performance Differences on Smartphones", *Proc. of the 8th Internation Conf. on Mobile Systems, application, and services (MobiSys'10)*, 2010.

[Jararweh13] Y. Jararwej, L. Tawalbeh, F. Ababneh, F. Dosari, "Resource Efficient Mobile Computing using Cloudlet Infrastructure", *Proc. of the 9th IEEE International Conference on Mobile Ad-hoc and Sensor Network*, December 2013, pp. 373-377.

[Jiao13] L. Jiao, R, Friedman, X. M, Fu, S. Secci, Z. Smoreda, H. Tschofenig, "Cloud-based Computation offloading for mobile devices state of the Art, challenges and opportunities", *Proc. of Future Network and Mobile Summit*, July 2013, pp. 1-11.

[Kalic12] G. Kalic, I. Bojic, M. Kusek, "Energy Consumption in Android Phones when using Wireless Communication Technologies", *Proc. of the 35th MIPRO*, May 2012, pp. 754-759.

[Khan14] A. R. Khan, L. Kuala, M. Othman, S. A. Madani, S. U. Khan, "A Survey of Mobile Cloud Computing Application models", *IEEE Communications Surveys & Tutorials*, February 2014, pp. 393-413.

[Kumar10] K. Kumar., Y. H. Lu, "Cloud computing for mobile users: Can offloading Computation Save Energy?. *Computer*, vol.43, no.4, April 2010, pp.51-56.

[Kumar12] K. Kumar, J. Liu, Y. H. Lu, B. Bhargava, " A Survey of Computation Offloading for Mobile Systems", *Mobile Networks and Application*s, vol.18, Feb. 2013, pp.129-140

[Lai13] C. F. Lai, H. C. Chao, Y. X. Lai, J. F. Wan, "Cloud-assisted real-time transrating for http live streaming", *Proc. of IEEE Wireless Communications*, vol.20, no.3, June 2013.

[Mahadev09] S. Mahadev, P. Bahl, R. Caceres, N. Davies, "The Case for VM-Based cloudlets in Mobile Computing", *Proc. of IEEE Pervasive Computing*, vol.8, no.4, Oct 209, pp.14-23

[MCC14] Mobile Cloud Computing. [online]. Available: http://en.wikipedia.org/wiki/Mobile_cloud_computing, last accessed on February 16, 2014.

[Mccullough11] J. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. Snoeren, R. Gupta, "Evaluating the Effectiveness of Model-based Power Characterization", *Proc. of 2011 USENIX conference on USENIX annual technical conference*, June 2011, pp. 12-12.

[Mobileweb14] [online]. Available: http://blog.flurry.com/bid/95723/Flurry-Five-Year-Report-It-s-an-App-World-The-Just-Web-Lives-in-It, last accessed on April 1, 2014.

[Monitor13] Battery Monitor. [online]. Available: https://play.google.com/store/apps/details?id=ccc71.bmw, last accessed on September 26, 2013.

[Mtibaa13] A. Mtibaa, K. A. Harras, A. Fahim, "Towards Computational Offloading in Mobile Device Clouds", *Proc. of 2013 IEEE International Conference on Cloud Computing Technology and Science*, Dec. 2013, pp. 331-338.

[Palankar08] M.Palankar, A.Iamnitchi, M.Ripeanu, S.Garfinkel, "Amazon S3 for science grids: a viable solution?" *Prof. of the 2008 international workshop on Data-aware distributed computing*, June 2008, pp. 55-64

[Power13] PowerTutor. [online]. Available: http://ziyang.eecs.umich.edu/projects/powertutor/, last accessed on September 16, 2013.

[Sdk14] Software Development Kit. [online]. Available: http://developer.android.com/sdk/index.html, last accessed on March 5, 2014.

[Speed13] SpeedTest. [online]. Available: http://www.speedtest.net/, last accessed on September 3, 2013.

[Spp14] Serial Port Protocol. [online]. Available: https://developer.bluetooth.org/TechnologyOverview/Pages/SPP.aspx, last accessed on April 3, 2014.

[Statistics14] Mobile marking statistics 2014. [online]. Available: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/, last accessed in April 2014.

[Sthree13] Amazon S3. [online]. Available: http://aws.amazon.com/s3/, last accessed on September 18, 2013.

[Thiagarajan12] N.Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, J. P. Singh, "Who Killed My Battery: Analyzing Mobile Browser Energy Consumption", *Proc. of the 21st International Conference on World Wide Web (WWW)*, April 2012, pp. 41-50.

[Tuner13] Android Tuner [online]. Available: https://play.google.com/store/apps/details?id=ccc71.at, last accessed on September 16, 2013.

[Wang13] Wang, S. Dey, “Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications”, *IEEE Multimedia*, Jan. 16, 2013, pp. 870-883.

[Woodside07] C.M. Woodside, G. Franks, and D. C. Petriu, "The Future of Software Performance Engineering", *Proc. of the 29th International Conference On Software Engineering*; 2007, pp. 171-187.

[Wowza13] Amazon CloudFront services in Wowza. [online]. Available: http: www.wowza.com/news/wowza-announces-contentdelivery-through-amazon-web-services/, last accessed on September 18, 2013.

[Wu13] H. Wu, Q. Wang, K. Wolter, "Mobile Healthcare Systems with Multi-Cloud Offloading", *Proc. of the 14th IEEE International Conference on Mobile Data Management*, June 2013, pp. 188-193

[Zhang10] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z.Mao, L. Yang. “Accrate Online Power Estimation and Automatic Battery Behavior based Power Estimation and Automatic Battery Behavior based Power Model Generation for Smartphones”, *Proc. of the 8th IEEE international conference on Hardware/Software codesign and system synthesis*, October 2010, pp. 105-114.