# Mobility Aware Optimal Resource Allocation in Heterogeneous Mobile Cloud Computing

**Asma Enayet**

**Roll: Curzon Hall, No. : 567**

**Reg. No. : H-1488**

**Session 2012-13**

**A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering**


**Department of Computer Science and Engineering**

**University of Dhaka**

**Dhaka, Bangladesh**

**May, 2015**

# Declaration of Authorship

We declare that this thesis titled, "Mobility Aware Optimal Resource Allocation in Heterogeneous Mobile Cloud Computing" and the work presented in it are our own. We confirm that:

- The full part of the work was done while in candidature for an MS research degree in this University.
- Any part of this thesis has not previously been submitted for a degree or any other qualification in this University or any other institution.
- We have consulted the published works of others with appropriate references.
- This thesis work is done entirely by us and our contributions and enhancements from other works are clearly stated.

Signed:

Candidate

Countersigned:

Supervisor

# Abstract

Due to the significant advancement of Smartphone technology, the applications targeted for these devices are getting more and more complex and demanding of increased energy and computing resources. Mobile cloud computing (MCC) allows the Smartphones to perform these heavy computing tasks with the help of powerful any computing resource, namely, cloudlet, acts as cloud server attached to any access points (APs). However, the connectivity of mobile devices with a given AP is not continuous, rather sporadic with varying signal strengths. Furthermore, the heterogeneity of the cloudlet resources and the application requests put additional challenges in taking optimal code execution decision. Additionally, due to free space mobility patterns of mobile device user puts significant challenge on allocation of optimal computing resource for any incoming application code execution request.

In this thesis, we develop a mobility aware optimal resource allocation mechanism for remote code execution in heterogeneous mobile cloud computing environment namely Mobi-Het. The Mobi-Het minimizes the response time and failure percentages of the remote code execution requests while it increases the load balancing among the cloudlets. Therefore, Mobi-Het can be formulated as a mixed integer linear programming problem and its performance greatly depends on accurate prediction of sojourn time of a mobile device under each AP, estimated code execution time and workload distribution among the cloudlets. We have developed a time duration prediction scheme under each cloudlet by using smooth random mobility model, execution time measurement method-

ology based on the received signal strength values from cloudlets and workload balancing through current virtual machine usage in heterogeneous environment.

We develop a prototype of our proposed Mobi-Het resource allocation system in Google cloud using access points as cloudlets and android devices. We evaluate the performances of the proposed resource allocation mechanism and the existing state-of-the-art mechanism. The results show that it outperforms the state-of-the-art methods in terms of success percentage, quality of experience (QoE), workload distribution.

# Acknowledgment

As a matter of first importance, I am thankful to almighty Allah for giving me the capability, ability, chance, collaborating hands and stamina to stroll on this path. I also want to express my sincere gratitude to my research supervisor, Md. Abdur Razzaque. Without his support and committed inclusion in every venture all through the methodology, this thesis would have never been fulfilled. His ability to indulge me in my work in order to clarify my own inquiry effectively is something significant for me which I owed to him profoundly. I would want to thank him kindly for his aspiring guidance, invaluably constructive criticism and friendly advice during the thesis work.

The individuals from the Green Networking and Research(GNR) group have contributed tremendously to my scholarly life at University of Dhaka. I would like to thank the members of GNR group with whom I had discussed the early versions of the thesis. They raised many precious points in our discussion which helps me to improve my work remarkably. I should likewise be thankful to my teachers who helped me in various ways by giving different resources and moral support. At long last, my novel thanks must go to my parents for teaching me to strive for the challenging choices and my siblings for their consistent support.

Asma Enayet

# Table of Contents

**Abstract** **i**

**Acknowledgment** **iii**

**Table of Contents** **iv**

**List of Figures** **vi**

**List of Tables** **viii**

**Chapter 1 Introduction** **1**

1.1 Introduction . . . . . 1

1.2 Remote Code Execution Technologies . . . . . 3

1.3 Mobile Cloud Computing . . . . . 3

1.3.1 Mobile Cloud Computing Architecture . . . . . 4

1.3.2 Basics of Resource Allocation in MCC . . . . . 5

1.3.2.1 VM Migration . . . . . 5

1.3.2.2 Code Partitioning . . . . . 6

1.4 Cloudlet based Mobile Cloud Computing . . . . . 6

1.4.1 Advantages of Cloudlet based MCC . . . . . 6

1.5 Motivation . . . . . 7

1.5.1 Scope of this work . . . . . 8

1.5.2 Research challenges . . . . . 9
1.5.3 Mobility Prediction . . . . . 10
1.5.4 Load Balancing . . . . . 10
1.5.5 Network Connectivity . . . . . 10
1.5.6 Heterogeneity of requests and resources . . . . . 10
1.5.7 Thesis Contribution . . . . . 10
1.6 Organization of The Thesis . . . . . 11

**Chapter 2 Background and Motivation 12**
2.1 Introduction . . . . . 12
2.2 Remote Code Executing . . . . . 13
2.2.1 Code execution in Cloud . . . . . 13
2.2.1.1 MAUI . . . . . 14
2.2.1.2 CloneCloud . . . . . 14
2.2.1.3 ThinkAir . . . . . 15
2.2.1.4 CADA . . . . . 15
2.2.2 Code execution in Cloudlets . . . . . 16
2.2.2.1 MuSIC . . . . . 16
2.2.2.2 ENDA . . . . . 17
2.3 Discussion . . . . . 17
2.4 Conclusion . . . . . 18

**Chapter 3 Mobi-Het System 19**
3.1 Introduction . . . . . 19
3.2 System Overview . . . . . 19
3.2.1 Mobile Device . . . . . 19
3.2.2 Master Cloud . . . . . 20
3.2.3 Cloudlet . . . . . 21

3.2.4 Notations . . . 22
3.3 Mobi-Het Architecture . . . 22
3.3.1 Optimal Resource Allocation . . . 23
3.3.2 Determination of degree of associativity ($\Psi$) . . . 25
3.3.3 Computation of $\sigma$ . . . 29
3.3.4 Calculation of $t_{c,a}$ . . . 30
3.4 Discussion . . . 32
3.5 Conclusion . . . 32

**Chapter 4 Performance Evaluation** **33**
4.1 Introduction . . . 33
4.2 Environment Setup . . . 33
4.3 Performance Metrics . . . 34
4.4 Experiment Results . . . 36
4.4.1 Impacts of varying mobility speeds . . . 36
4.4.2 Impacts of varying mobile devices . . . 40
4.4.3 Impacts of input data size . . . 45
4.5 Conclusion . . . 46

**Chapter 5 Conclusions** **49**
5.1 Summary of Our Work . . . 49
5.2 Discussion . . . 49
5.3 Future Work . . . 49

**Bibliography** **50**
**Appendix A List of Notations** **57**
**Appendix B List of Publications** **58**

# List of Figures


1.1 Mobile cloud computing architecture . . . . . 4
1.2 Mobile application execution environment using cloudlets . . . . . 8

3.1 Mobi-Het Design Architecture . . . . . 20
3.2 Determination of degree of associativity . . . . . 28

4.1 Percentage of application successfully completed . . . . . 37
4.2 Average execution time . . . . . 38
4.3 Standard deviation of cloudlet computation loads . . . . . 39
4.4 Operation overhead . . . . . 40
4.5 Percentage of application successfully completed . . . . . 41
4.6 Average execution time . . . . . 42
4.7 Standard deviation of cloudlet computation loads . . . . . 43
4.8 Operation overhead . . . . . 44
4.9 Percentage of application successfully completed . . . . . 45
4.10 Average execution time . . . . . 46
4.11 Percentage of application successfully completed . . . . . 47
4.12 Average execution time . . . . . 48

# List of Tables

3.1 List of Notations . . . . . 22
3.2 Predefined Velocities . . . . . 25
3.3 RSSI vs Data Rate . . . . . 31

4.1 Device specification . . . . . 35

# Chapter 1

# Introduction

## 1.1 Introduction

The use of Internet enabled mobile devices(smartphone,tablet,etc.) and the applications running on them are increasing exponentially now-a-days [1]. Pew Research Center conducted a survey in 2013, which revealed that 56% of American adults own a smartphone of some kind [2]. According to another research conducted by New Relic, a web application performance company, showed that 1.3 million Android devices are activated every day [3]. These devices are equipped with various functionalities such as GPS, WiFi, cameras, sensors, good level of storage and processing speeds that lead them installed with wide range of applications [4]. More interestingly, the applications targeted for these devices are getting more and more complex with the number of users and the increasing capabilities of the devices [5]. And in recent years, these applications have started becoming abundant in various categories such as image processing, gaming, participatory sensing, real-time monitoring, social networking, travel and news, etc [6]. These sophisticated and resource-hungry applications are increasingly posing requirements especially for more energy and computation power. Since, mobile devices are inherently constrained by processing power and energy [7], the topic of saving mobile power and minimizing CPU consumption of mobile devices has got attraction of researchers in this field. The resource scarcity of these devices has compelled them to exploit computation resources of others in different ways, one popular way is mobile cloud computing (MCC)

[8, 9, 10].

Mobile cloud computing (MCC) [11, 12, 13] enables a resource constrained mobile device to perform complex application in one of many remote compute resource with the power of cloud computing. The task that needs to perform complex computation and is not feasible to be executed on the mobile device could be migrated to any of the available cloud server and the resource allocation and execution of the task is performed on the powerful machine of the cloud rather than on the mobile devices. This process of task migration is known as code/task offloading in the literature. But offloading application codes to a distant cloud often leads much overhead such as high latency and low bandwidth as the actual cloud provider could be situated far away from the client. Therefore, there arises a computing resource named cloudlet which is attached to any access point (AP) for high speed offloading purposes [8, 10]. Now, in hotspot areas like educational institute, commercial area, bus/railway station etc. APs are densely deployed with attached cloudlets [14] as shown in Figure 1.2. In such environment, allocation of a cloudlet for application request, out of many available co-located cloudlet APs, is a great research challenge due to the mobility of the mobile devices, resource heterogeneity of the cloudlets, heterogeneity of application requests and time varying dynamic network quality. In this thesis, we address the problem of optimal resource allocation for remote job execution, namely Mobi-Het, that selects an optimal computing resource by exploiting user mobility, application QoS requirements, AP signal quality and workload distribution of the corresponding cloudlets. We develop a prototype of Mobi-Het system which shows that this system performs better under dynamic condition of the task execution environment.

## 1.2 Remote Code Execution Technologies

Now-a-days, Internet enabled mobile device usage percentage have reached at highest peak due to the dynamic behavior and diverse usage of heterogeneous applications running in the device [15]. These devices are of limited resource, such as, battery power, low computation speed, small storage capability etc which delays or fails the proper execution of those intensive tasks. Due to these resource scarcities of mobile devices, other methodologies has come to the scenario where any other resource can compute those tasks for those limited resources mobile devices. Participatory computing [16, 17], opportunistic computing [18], cloud computing [19, 20, 21, 22, 23] and mobile cloud computing (MCC) [8, 9, 10] are the most popular methodologies for remote code execution.

## 1.3 Mobile Cloud Computing

The concept of (MCC) has been recently come into market which enables these mobile devices to execute computationally intensive parts of an application on a remote cloud server [24, 25, 26]. Mobile cloud computing at its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from the mobile devices and into powerful and centralized computing platforms located in clouds, which are then accessed over the wireless connection based on a thin native client.In other words, mobile cloud computing is the combination of cloud computing and mobile networks to bring benefits for mobile users, network operators, as well as cloud computing providers. The ultimate goal of MCC is to enable execution of rich mobile applications on a plethora of mobile devices, with a rich user experience [27]. /

Mobile devices are inherently constrained by limited battery life and faces many resource challenges (limited storage, inconsistent network bandwidth etc.). On the other hand, cloud computing offers advantages to users by allowing them to use infrastructure,

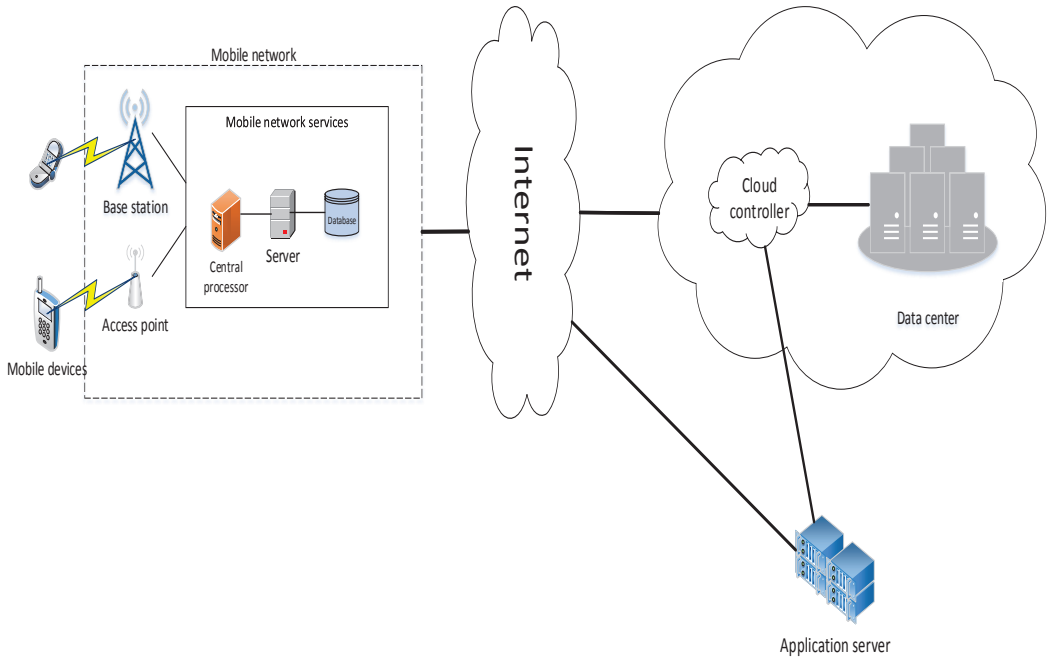


Figure 1.1: Mobile cloud computing architecture

platforms and software by cloud providers at low cost and elastically in an on-demand fashion. Mobile cloud computing provides mobile users with data storage and processing services in clouds, obviating the need to have a powerful device configuration (e.g. CPU speed, memory capacity etc), as all resource-intensive computing can be performed in the cloud [28, 29].

### 1.3.1 Mobile Cloud Computing Architecture

Figure 1.1 shows the basic architecture of mobile cloud computing environment. The basic operations of the MCC architecture is described bellow:

- Mobile devices are connected to the mobile networks via base stations that establish and control the connections and functional interfaces between the networks and

mobile devices.

- Mobile users' requests and information are transmitted to the central processors that are connected to servers providing mobile network services.
- The subscribers' requests are delivered to a cloud through the Internet.
- In the cloud, cloud controllers process the requests to provide mobile users with the corresponding cloud services.

### 1.3.2 Basics of Resource Allocation in MCC

Mobile cloud computing enables resource constrained mobile devices to execute more demanding application by executing computationally intensive parts of an application on a remote cloud server. This technique of allocating and executing the more resource demanding part of an application on the cloud server instead of the local device is known as code offloading. There are two major types of offloading process found in the literature namely *VM migration*[30] and *code partitioning*[31].

#### 1.3.2.1 VM Migration

In this technique the state of the total virtual machine from the mobile device is transferred to the cloud server. This process basically recreates the mobile execution environment on the cloud server with more computation power. Then the application is executed to that powerful virtual machine. Executing mobile application on such powerful virtual machines reduces the time of execution and the output of the execution is produced more quickly. But major problem of this technique is that it needs a huge amount of data transfer for the offloading process. Because transferring the whole virtual machine state to the cloud server requires a huge number of information to be exchanged between the cloud server and mobile device. For this reason this technique of offloading code to the cloud server is no considered to be a feasible approach in mobile cloud computing.

#### 1.3.2.2 Code Partitioning

In this technique the application code running on the mobile device is partitioned into two categories. One category of the partitioned code gets offloaded to the cloud server and other category is executed in the local device. The offloaded portion of the code basically requires more computation time and demands more energy from the mobile device. The advantage of this technique is that it gets rid of the problem of huge data transfer as it was needed by the VM migration technique. Moreover, in this technique a device or application clone is generally kept on the cloud server. So offloading in this technique often only requires exchanging the application and task id and the data that are essential for executing the task and are not already present in the cloud server.

## 1.4 Cloudlet based Mobile Cloud Computing

Offloading application codes to a distant cloud often leads much overhead such as high latency and low bandwidth because the actual cloud provider could be situated far away from the client. Therefore, there exists a computing resource named cloudlet which is attached to any access point (AP) for high speed execution purpose [8, 10, 32].The cloudlet will contact to the cloud server periodically to get the cloud statistics, load information, network delay, throughput and many more to keep updated with the dynamic cloud behavior. Mobile devices offload applications to the local cloudlet to do the required processing and return the final result back, this reduces the transmission delay, also scale downs the power consumption of the mobile device.

### 1.4.1 Advantages of Cloudlet based MCC

- **Extending battery lifetime:** Computation offloading migrates large computations and complex processing from resource-limited devices (i.e., mobile devices) to resourceful machines (i.e., servers in clouds). Executing these heavy computational

task on the remote servers can save energy significantly. Many mobile applications take advantages from task migration and remote processing.

- **Improving data storage capacity and processing power:** MCC enables mobile users to store/access large data on the cloud. Since the data now are stored on the cloud, mobile applications are no longer constrained by storage capacity.
- **Improving reliability and availability:** Keeping data and application in the clouds reduces the chance of data loss from the mobile devices.
- **Scalability:** Since cloud computing provides the users with on demand resource allocation the mobile applications can be performed and scaled to meet the unpredictable user demands.

## 1.5 Motivation

In hotspot areas like educational institute, commercial area, bus station, railway station etc. APs are densely deployed with attached cloudlets. In such environment, selection of a cloudlet for application offloading out of many available co-located cloudlet APs is a great research challenge due to the mobility of the mobile devices, resource heterogeneity of the cloudlets, heterogeneity of application requests and time varying dynamic network quality.

A mobile device user moves toward any place with any direction and velocity which is referred as free movement nature of a user. Each application running on a mobile device have different Quality of Service (Qos) typically defined by maximum allowable time to execute and resource requirements [33]. The cloudlet resources are varied from one to another in terms of their computation and storage capabilities. Therefore, due to application and resource heterogeneity its not feasible to execute every application at every cloudlet. Furthermore, the Wi-Fi signal strengths from different APs greatly

vary over time, which puts great impacts on the application that requires large data to offload. Therefore, the optimal selection of a cloudlet for code offloading is a multi-criteria objective problem and its more challenging to obtain the solution due to the dynamicity of the variables.

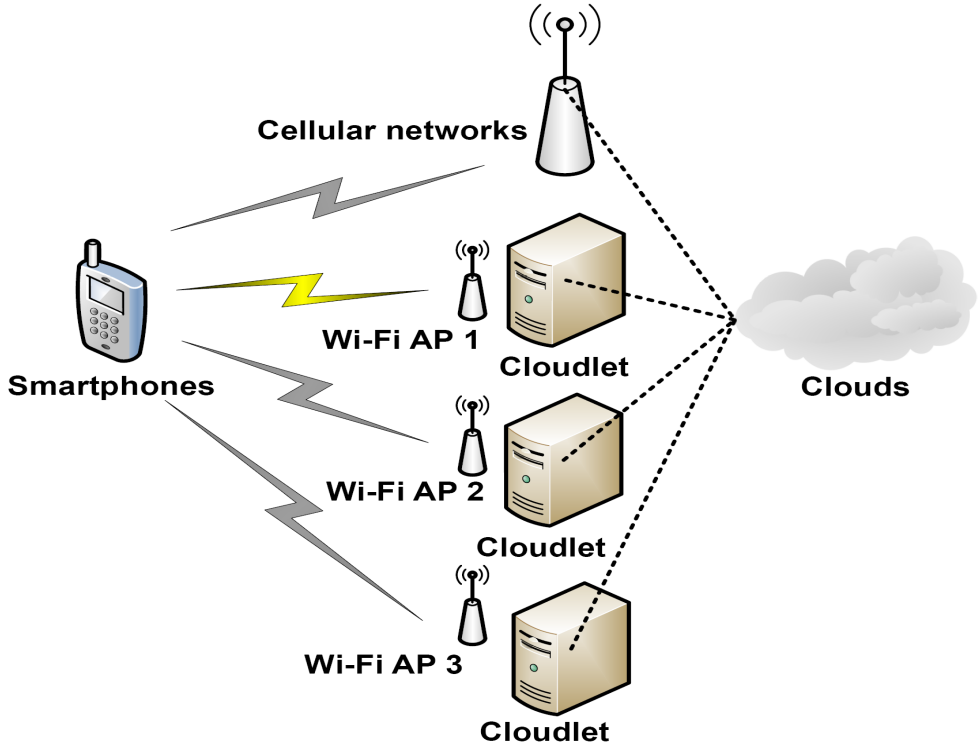


Figure 1.2: Mobile application execution environment using cloudlets

### 1.5.1 Scope of this work

The procedure of offloading tasks from a mobile device to a distant master cloud has been studied in several works in the literature [20, 19, 34, 35, 36]. Zhang et al[35] proposed an energy efficient task scheduling algorithm on mobile device under Markovian stochastic channel. They modeled minimum energy task scheduling problem as a constrained

stochastic shortest path problem on directed acyclic graph. They have also adopted LARAC (Lagrangian Relaxation Based Aggregated Cost) algorithm to obtain the approximate solution of the constrained optimization problem. But no efficient decision making policy was introduced for which task to offload. There is no proper procedure for the estimation of task executing time. At most onetime migration from mobile device to cloud is considered, which is not a real like scenario. Another paper eTime[36] aggressively and adaptively seizes good wireless connectivity of 3G or WiFi which prefetch frequently used data and defer delay tolerant data for energy saving. eTime applies Lyapunov optimization techniques to make online transmission decisions, which only requires the information of current network bandwidth and data queue backlogs to balance the trade-off between energy consumption of mobile devices and delays of heterogeneous applications. But no scheduling among tasks from different mobile devices which can be offloaded to the same cloud and also on demand resource allocation is not considered.

Several works have considered cloudlets as middle-ware compute resources in the literature. The authors of MuSIC [37] used multi-tiered cloud architecture aiming to develop efficient methods for dynamic mapping of resources considering mobility patterns. They expressed mobile application execution as location-time workflow (LTW) in order to model the mobile applications and capture mobility patterns, considering multiple QoS factors like power, price, and delay. ENDA[14] attempted to figure out the problem of making most effective offloading decision among several collocated cloudlets by predicting user track record. However, in both the works homogeneous application QoS requirements and network connectivity are considered. In real scenario, cloudlets are having heterogeneous compute, storage and network resources.

### 1.5.2 Research challenges

Now-a-days, hotspot areas like educational institute, commercial area, bus/railway station etc. APs are densely deployed with attached cloudlets [14] as shown in Figure 1.2. In

such environment, allocation of a cloudlet for application request, out of many available co-located cloudlet APs, is a great research challenge due to the mobility of the mobile devices, resource heterogeneity of the cloudlets, heterogeneity of application requests and time varying dynamic network quality.

### 1.5.3 Mobility Prediction

### 1.5.4 Load Balancing

### 1.5.5 Network Connectivity

### 1.5.6 Heterogeneity of requests and resources

### 1.5.7 Thesis Contribution

In this thesis, we propose an optimal resource allocation methodology for remote job execution, namely Mobi-Het, that selects an optimal computing resource by exploiting user mobility, application QoS requirements, AP signal quality and workload distribution of the corresponding cloudlets. This selection is performed by the master cloud with the help of some information from mobile devices. The main contributions of this paper are as follows:

- The problem of allocating the most suitable resource for executing the application request from mobile devices at time $t$ is formulated as a multi objective mixed integer linear programming (MILP) problem.
- The Mobi-Het translates the user mobility patterns into degree of assosiativity of a user with different cloudlets.
- The Mobi-Het allows free movement of the mobile devices within the network area and more accurately predict the mobility speed and direction by exploiting smooth random mobility model [38], opposing to limited mobility on predefined routes in ENDA[14] and MuSIC[37].

- We develop a prototype of the system using Google cloud as the master cloud, some APs as cloudlets Android devices. We implement two real-world rich mobile applications ($N$ Queen Application and face detection application) for offloading purpose and one *Monitoring Application*.
- Our experimental results, from the test-bed implementation of Mobi-Het system, show that it achieves as high as 52% improvement in successful execution of applications in a heterogeneous MCC environment compared to a number of state-of-the-art techniques.

## 1.6 Organization of The Thesis

The remainder of the thesis is organized as follows. Chapter 2 contains a study on the related works in this field of research. Chapter 3 contains system model and assumption and gives explicit insight into our proposed Mobi-Het architecture along with determining execution time, workload balancing and time duration prediction under each cloudlet. Chapter 4 presents the performance evaluation of the algorithm. Finally, we conclude the paper in Chapter 5.

# Chapter 2

# Background and Motivation

*In the previous chapter, we have introduced the idea of mobile cloud computing and the process of code offloading process. Offloading the computation intensive part of an application the cloud server enables the resource constrained devices to execute more complex application in an energy and computation efficient manner. The importance of estimating the execution time of a task on making the offloading decision is also discussed there. In this chapter, we give an overview on the motivation of code offloading process in mobile cloud computing. We also discuss some offloading techniques and architecture that have been introduced in earlier works.*

## 2.1 Introduction

The advancement in the smartphone technology enabled the application developer to build more complex applications for the smartphone users. But mobile devices are constrained by many resource challenges (limited battery power, limited memory, network etc.). As a result the power of cloud computing is exploited to let these resource constrained mobile devices to run complex application in order to obtain improvement on the application performance and energy consumption. Significant number of methodologies and frameworks are designed to enable the resource constrained mobile devices to use the immense power of cloud computing.

Recently several task migration techniques for mobile cloud computing have been

designed but most of them do not consider the computation and energy aspect together when making the offloading decision. On the other hand deciding optimal place for a particular request has not been studied yet. In this chapter, the subsequent sections are organized as follows. In section 2.2 we discuss a number of remote code execution technique in MCCs. After that in section 2.2.2 we discuss the technique of task execution using cloudlets.

## 2.2 Remote Code Executing

There are a good number of studies in the literature regarding the method of offloading application code from a resource poor mobile device to a rich server is known as surrogates. Satyanarayana [39] proposes cyber foraging in which, offloading application code to resource rich non mobile architecture to conserve sufficient amount of energy. Spectra [40] and Scavenger [41] are of similar approach which offloads application to another mobile device. However, offloading code to surrogates introduces safety and reliability challenges because of the absence of the supervisor that monitors the reliability and performance factors [42]. On the other hand, a distributed computing paradigm, cloud computing [43] as has recently obtained tremendous response in the field of offloading code from mobile devices. Offloading resource hungry application code (entirely or partially) to cloud, can overcome the problem of energy consumption in mobile device.

### 2.2.1 Code execution in Cloud

The code offloading works studied in the liteature can be catagorized broadly in two groups. In the 1st group the mobile application code is offloaded to remote master cloud and the significant works in this catagory are MAUI[19], ThinkAir [34], CloneCloud [20], Cuckoo [44] etc.

#### 2.2.1.1 MAUI

A number of optimization technique have been studied in the literature for offloading code in MCCs. The author of MAUI[19] first introduced a global optimization technique on the method call graph to offload methods to the cloud server for optimizing the energy consumption of a mobile device. They argued that considering each of the method individually cannot optimally solve the offloading problem since it cannot capture the whole picture of method execution. They showed that offloading some method may in effect execute more methods on the cloud server if that method invokes other methods. So offloading that method will save the cumulative cost of those methods. MAUI makes the offloading decision by analyzing the trade off between the energy consumed by the remote execution and local execution. Although in many cases offloading in this techniques often leads to performance improvement but ignoring the computational aspect of the task could lead to situations where the performance of the application might suffer.

#### 2.2.1.2 CloneCloud

The analysis and partitioning of the application binary is addressed in this paper[20]. The authors of CloneCloud introduced a technique where the application binary is first marked to define various portions of the code that could be offloaded to the cloud server. Then some random set of inputs is generated prior to the execution. Then based on each randomly generated input a linear optimization solver was run to partition the application binary into two sets. One of the sets is to be executed on the cloud server and the other is to be executed on the local device. The partition information for these randomly generated inputs are saved in a database and the application is run the input to the application is matched with this database that the partitioning information obtained from the database is used for code offloading. This technique depends heavily on the randomly generated input sets for making the offloading decision. The randomly generated input sets prior to the execution of the application cannot cover all the dynamic conditions

that could arise on the course of the execution of the application. Generally the mobile application user behaves dynamically and if the user gives input to the application that cannot be matched to any of the condition saved in the database this technique of offloading would drastically fail.

#### 2.2.1.3 ThinkAir

The problem of previous works in MAUI and CloneCloud was first attacked by the authors of [34]. They improved the technique of code offloading by adopting an online method level offloading. The code architecture of ThinkAir requires the application developer to annotate the methods that could be offloaded to the cloud server. These methods must satisfy some constraints as mentioned in the paper. They also designed their custom compiler to convert a task for remote execution. This paper makes the offloading decision based on a set of previous execution history. The previous remote and local execution information is saved in the database. When making the offloading decision the average of the previous local and remote execution time is taken estimate the cost of remote and local execution. Comparing these execution cost this technique makes the code offloading decision. The problem with this approach is that by averaging the previous execution history cannot make accurate prediction on the task execution cost if the behavior of the task is not consistent. Since the offloading decision mainly depends on this estimation of the execution cost inaccurate estimation can lead to poor choices.

#### 2.2.1.4 CADA

The context-aware approach for offloading code the cloud server from the mobile device was introduce in the paper [45]. The author of CADA[45] argued that the mobile device user moves within a certain set of location. For example a student is expected to be in his school campus on class times or an employee is expected to be at his office on

office time. They also argued that the move of a mobile user is similar on a given time of the day. For example an employee will use the same road on his way from home to office. They assumed that the network condition remains the same at given time and location. The saved the past history of a task execution on a given time and location in a database. Whenever a task is considered for offloading the local and remote execution cost of the task is estimated using the past history of that task on that particular time and location. Comparing this estimated cost the offloading decision is made. While in some of the cases this technique can estimate the execution cost of a task effectively, the work ignores dynamic behavior of the application user. If the application user does not behave in consistency with the time and location of the mobile device this technique cannot accurately estimate the task execution cost and fails to make correction decision on code offloading.

### 2.2.2 Code execution in Cloudlets

Above methods only include mobile device and the remote clouds into the scenario. But offloading application code to distant cloud often leads too much overhead such as high latency and low bandwidth because the actual cloud provider could be situated far away from the Smartphone. So, some of the previous works assumed that there is an intermediate server called cloudlet attached to any Wi-Fi access point for offloading purpose.

In the second catagory the cloudlet based multi-tier offloading is studied and the significant works in this catagory are ENDA [14], MuSIC [37] etc.

#### 2.2.2.1 MuSIC

The authors of MuSIC[37] considered multiple users with a three-tier architecture for MCC with limited poor local cloud. They also take into consideration user mobility patterns to develop dynamic resource mapping methodologies. They expressed mobile

application execution as location-time workflow(LTW) and formulated it as an optimization problem and propose a greedy heuristic approach for solution.

#### 2.2.2.2 ENDA

In ENDA[14] applications are offloaded to cloudlets and most optimal cloudlet is chosen based on historic user traces. Route prediction algorithm is developed to list down the all possible route for a mobile device. Based on the listed route and user trace, a route is predicted to connect the smartphone to cloudlets. If no cloudlets are available, application is offloaded to master cloud through 3G. Still, if there is any link failures during offloading, the device waits for a re-connection. However, in both the papers homogeneous cloudlets are considered. In real scenario cloudlets are of heterogeneous speed, storage and computation power. They also did not considered user's velocities and moving direction which is crucial for track prediction.

## 2.3 Discussion

In this Thesis, we develop a mobility prediction based solution to the problem of optimal resource allocation with improved efficiency and reliability in mobile cloud computing. We consider free movement of mobile device with different velocities and direction and predict speed and direction for next time step using smooth random mobility model. We address the heterogeneity of cloudlets and application requests for a balanced workload distribution in the cloudlets and determine the remote execution time. To the best of our knowledge, mobility and heterogeneity aware compute resource selecting with workload balancing for optimal allocation in MCC has not been introduced in any of the existing works.

Our solution framework has a good level of similarity with ENDA[14], however, the key working principle and the methodologies are different in following way. First, we

address the limitation of smartphone's movement on some designated route and consider real scenario of free movement patterns of any smartphone and predicted next movement pattern based on historical data. Second, application and compute resource heterogeneity is also considered in this paper in order to increase the quality of service because in real world there are different types of cloudlets having dissimilarities among their computation and storage capabilities. Third, different type of application requests arriving for offloading with various requirements have been addressed in this paper. Finally, in wireless network minimum distance is not sufficient for determining the minimum execution time. Here we consider network signal strength along with the distance and time duration of connectivity for selecting the optimal compute resource.

## 2.4 Conclusion

# Chapter 3

# Mobi-Het System

*In this chapter, we present our proposed Mobi-Het optimal resource allocation system in detail. We first discuss the preliminaries and assumptions that have been considered in our proposed Mobi-Het system. Then we go through the system architecture and its operations.*

## 3.1 Introduction

## 3.2 System Overview

Figure 3.1 depicts the major components of Mobi-Het which work combindly targeting to allocate optimal computing resource for a received application execution request. This model consists of three major components mobile devices, cloudlets and cloud.

### 3.2.1 Mobile Device

We assume multiple mobile devices in the environment and there can be one request for remote execution for each mobile device at a time. Each application request is associated with the mobile ID from each that particular request arrives. Here, mobile devices send heterogeneous application requests to the master cloud for allocating optimal computing resource for remote application execution. As different applications have diverse resource requirements, each application request consists of application size, type, instruction count and maximum allowable time for completing the execution. The decision maker compo-

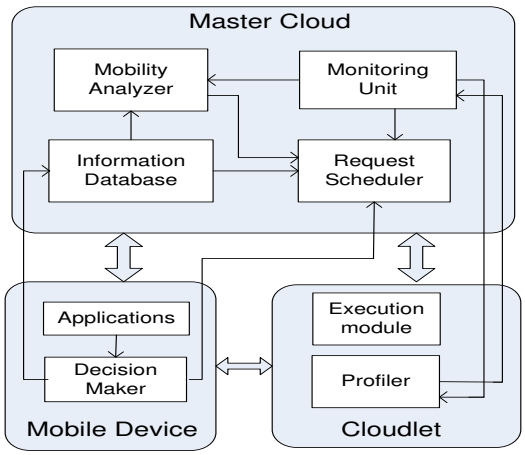


Figure 3.1: Mobi-Het Design Architecture

nent takes the decision of which task to offload from mobile device. Some of them are data intensive, i.e. N-Queen application and some are data intensive, i.e image processing applications. This decision maker module takes the decision of which job request to send for optimal allocation for execution depending on execution time, network connectivity and amount of residual energy [36, 19, 34]. If the requirements of these parameters are fulfilled then this module will initiate remote execution request. Then the mobile device sends its geographical location $(x_t, y_t)$, speed/velocity $(v(t))$ and direction $(\phi(t))$ at time $t$ for an application request $a$ from corresponding mobile device.

### 3.2.2 Master Cloud

We assume that the master cloud have large infrastructure resources (virtual machine instances) and high computation power. The resource allocation algorithm is running in the master cloud after each scheduling interval $\delta$. The heterogeneity of computing resource, i.e cloudlet adds additional challenge on the optimal resource allocation for each of the application request. The master cloud has four main units *Information database, Mobility analyzer, Monitoring unit and Request scheduler.* The information database

maintains a set of information for each mobile device which includes its previously reported information at every time step. Previous $n$ number of historical information is stored in the database for every application request $a$ from corresponding mobile device. Also the geographic location $(x_c, y_c)$ and range(radius $R_c$) of each computing resource namely cloudlet is known to the master cloud on priory. Then the mobility analyzer analysis the mobility patterns of a mobile device to estimate the velocity and direction at next time step based on the historical information saved in the information database. Monitoring unit monitors current cloudlet situation including geographical location of each cloudlet, virtual machine instances running on each cloudlet and workload of each cloudlet at time $t$. Then the request scheduler schedules the incoming requests after a certain time interval $\delta$ and selects an optimal computing for each application request arrived at any scheduling interval $\delta$ for execution.

### 3.2.3 Cloudlet

In multi tiered mobile cloud computing environment, middle-ware computing resources, namely *cloudlets* are assumed each of which is attached to one Wi-Fi access point. We assume each cloudlet is a source powered mini cloud server with limited infrastructure resources and computation power than the master cloud but situated near the mobile devices having heterogeneous computing and computation capabilities. Mobile devices and cloudlets are connected using Wi-Fi network. Each cloudlet is capable of running some virtual machine instances so application from mobile device can execute on the cloudlet. There can be several type of VMs in a cloudlet $c$ and $\zeta$ is the size of one $VM$ instance of a certain type of a cloudlet. The value of $MIPS$ (millions of instruction execution per second) of each VM instances at each type of VM also differs from one another. We assumed that the capacity of each VM instances in one VM type is equal. The workload $W_c$ of a cloudlet $c$ is determined by the ratio of allocated $VM$s in cloudlet $c$ for usage $W'_c$ the maximum allowable workload $W_c^{max}$.

### 3.2.4 Notations

The notations used throughout the paper are listed in Table 3.1.

Table 3.1: List of Notations

| | |
|---|---|
| $\mathcal{M}$ | Set of all mobile devices |
| $\mathcal{A}$ | Set of all application requests arrived in a scheduling interval $\delta$ |
| $\mathbb{C}$ | Set of all cloudlets in the network |
| $\mathcal{C}$ | Set of all cloudlets within the range of each mobile |
| $\mathbb{V}_c$ | Set of all virtual machine instances available at cloudlet $c \in \mathcal{C}$ |
| $t_{c,a}$ | Execution time of an application $a$ if computed in cloudlet $c \in \mathbb{C}$ |
| $T_a$ | Maximum allowable time for executing an application $a \in \mathcal{A}$ |
| $W_c$ | Workload of cloudlet $c \in \mathcal{C}$ |
| $\delta$ | Scheduling interval in the master cloud. |

## 3.3 Mobi-Het Architecture

The proposed Mobi-Het is a multi layer architecture consisting of mobile devices, local cloudlets and the master cloud that interact with each other, as shown in Fig. 3.1. The key idea of Mobi-Het is to schedule the remote code execution requests to the available VMs on cloudlets in such a way that the total execution time is minimized and execution reliability is increased, while minimizing the deviation of load distribution among

the cloudlets. The master cloud is responsible for the most complex computation that finds the optimal mapping of assigning remote code execution requests to solve the optimization problem. The Mobi-Het exploits the mobility patterns of the devices, current workloads of the cloudlets and the network signal strengths a mobile is receiving from different access points to determine the most optimal execution points for the requested tasks. What follows next, we describe the detail operations of Mobi-Het design components.

### 3.3.1 Optimal Resource Allocation

The problem of selecting the most optimal computing resource, i.e., the cloudlets for a group of arrived remote code execution requests from mobile devices is a difficult one due to the presence of multiple uncertain parameters associated with this. The application QoS requirement imposes a strict requirement of completing the execution within a maximum allowable time, $T_a$. Furthermore, the heterogeneous cloudlet resources take different amount of time, $t_{c,a}$ for executing an application $a \in \mathcal{A}$ onto a cloudlet $c \in \mathcal{A}$. For user satisfaction, the scheduling algorithm should try to keep $t_{c,a}$ as minimum as possible. Due to the mobility of a smartphone, it might not receive sufficient signal strength from the computationally feasible cloudlet. In addition to that, the workload of a cloudlet puts great impact onto the response time of the execution of the application, which can be minimized by keeping the standard deviation of the cloudlet workloads as low as possible. Therefore, we need to make a trade-off among the performance parameters.

In this work, we measure the degree of associativity ($\Psi_{c,a}$) of an application request $a \in \mathcal{A}$ from a particular mobile device with a cloudlet $c \in \mathcal{C}$ that captures how long the mobile device would be connected with that cloudlet. The application request includes the ID of that mobile device from which the request is sent. The decision maker would try to maximize $\Psi_c$ for assigning a code execution to a certain cloudlet. Given that, the

set of application requests arrived during time interval $\delta$, the master cloud invokes the following objective function to determine the optimal scheduling of code execution.

$$\min U(c,a) = \sum_{c\in\mathcal{C}} \sum_{a\in\mathcal{A}} (\frac{t_{c,a}}{\Psi_{c,a}} + \sigma_c) \tag{3.1}$$

$$s.t. : t_{c,a} \leq T_a, \quad \forall c \in \mathcal{C}, \forall a \in \mathcal{A} \tag{3.2}$$

$$\mu_c - \sigma_c \leq W_c \leq \mu_c + \sigma_c, \quad \forall c \in \mathcal{C} \tag{3.3}$$

$$W_c \leq W_c^{max}, \quad \forall c \in \mathcal{C} \tag{3.4}$$

$$RSSI_c \geq \Gamma_{RSSI}, \quad \forall c \in \mathcal{C} \tag{3.5}$$

Here, Eq. 3.1 is the objective function which is formulated as a multi objective mixed integer linear programming (MILP) problem to be solved by the master cloud. It is quite reasonable to assume that the master cloud has sufficient computing resources to solve the MILP problem in Eq. 3.1 within the stipulated time. In this case, the number of cloudlets and the application requests is also not very large, making the problem less complex. The Eq. 3.2 to 3.5 are the constraints of this objective function. The Eq. 3.2 is the *QoS constraint*, that states the time for remote execution of an application must be less than the maximum allowable time. The Eq. 3.3 represents the *workload distribution constraint*, after assignment, the workload $W_c$ of a cloudlet $c \in \mathcal{C}$ should not deviate by more than $\sigma_c$ amount from the system-wide mean workload $\mu_c$. Achieving the balanced workload distribution in a mobile cloud computing (MCC) system is quite important since it facilitates each of the cloudlets to decrease waiting time for the jobs and to have some extra room to accommodate newly arrived user requests in such a high mobility environment; otherwise, most of the users getting good connectivity signals from a certain cloudlet may occupy it's total computation capability and might make it exhaustive to accept a critical request. Thus, the constraint in Eq. 3.3 helps our Mobi-Het to acquire greater percentage of successfully executing application requests. The Eq. 3.4 denotes the *capacity constraint*, that states the current workload of a cloudlet must

not exceed its maximum computation capacity, $W_c^{max}$. The Eq. 3.5 means the received signal strength indication of each cloudlet within the range of a mobile device must be greater than a predefined threshold $\Gamma_{RSSI}$; otherwise, it won't be possible for the mobile device to transfer and receive codes and results, respectively, within the stipulated time.

### 3.3.2 Determination of degree of associativity ($\Psi$)

Degree of associativity ($\Psi_c$) is the predicted time duration of each mobile device within the range of a cloudlet $c \in \mathcal{C}$. For the measurement of $\Psi_c$, we need to consider the mobility patterns of each mobile device. The $Mobi-Het$ architecture allows a mobile device user to move at different velocities in random directions. Therefore, the key challenge of precisely determining the degree the associativity is the accurate estimation of the user moving speed and direction.

Table 3.2: Predefined Velocities

| | |
|---|---|
| $\mathcal{V}_0$ | static |
| $\mathcal{V}_1$ | slow walk |
| $\mathcal{V}_2$ | normal walk |
| $\mathcal{V}_3$ | fast walk |
| $\mathcal{V}_4$ | cycle speed |
| $\mathcal{V}_5$ | slow vehicle speed |
| $\mathcal{V}_6$ | normal vehicle speed |
| $\mathcal{V}_7$ | fast vehicle speed |
| $\vdots$ | $\vdots$ |
| $\mathcal{V}_{max}$ | maximum speed |

In this work, we consider a set of predefined velocities of a user as stated in Table 3.2. We exploit smooth random mobility model [38], in which the current velocity of a mobile device at different time slots are 'correlated', i.e., the current velocity of a mobile

device depends on its previous $n$ number of velocities. It is observed that mobile nodes in real life tend to move at certain preferred speeds, rather than at speeds that uniformly distributed in the range $[0,\mathcal{V}_{max}]$. In smooth random mobility model, the preferred speed values have high probabilities, while a uniform distribution is assumed on the remaining part of entire interval $[0,\mathcal{V}_{max}]$. For example, if a mobile node has the set of preferred velocities $\mathcal{V} = \{0, \mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_{max}\}$ then the probability distribution is,

$$P_{\mathcal{V}}(v) = \begin{cases} P(v=0)\delta(v) & v = 0 \\ P(v=\mathcal{V}_1)\delta(v-\mathcal{V}_1) & v = \mathcal{V}_1 \\ P(v=\mathcal{V}_2)\delta(v-\mathcal{V}_2) & v = \mathcal{V}_2 \\ \vdots & \\ P(v=\mathcal{V}_{max})\delta(v-\mathcal{V}_{max}) & v = \mathcal{V}_{max} \\ \frac{1-P(v=0)-P(v=\mathcal{V}_1)-\cdots-P(v=\mathcal{V}_{max})}{\mathcal{V}_{max}} & 0 < v < \mathcal{V}_{max} \\ 0 & \text{otherwise,} \end{cases} \tag{3.6}$$

where, $P(v=0)+P(v=\mathcal{V}_1)+\cdots+P(v=\mathcal{V}_{max}) < 1$. This model considers acceleration and de-acceleration speed $\alpha(t)$, which is uniformly distributed over $[0, \alpha_{max}]$ and $[\alpha_{min}, 0]$ for acceleration and de-acceleration, respectively, determined as follows,

$$P_{\alpha}(\alpha(t)) = \begin{cases} \frac{1}{\alpha_{max}} & \text{for acceleration, } 0 < \alpha(t) < \alpha_{max}, \\ \frac{1}{\alpha_{min}} & \text{for de-acceleration, } \alpha_{min} < \alpha(t) < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

For each time slot $t$, the new speed after a small time interval $\Delta t$ is calculated as,

$$v(t) = v(t-\Delta t) + \alpha(t)\Delta t \tag{3.8}$$

Now, the expected velocity at next time slot ($\hat{E}(v)$) for a mobile device can be estimated from the previous $n$ number of velocities and probability distribution function in

Eq. 3.6 as follows,

$$\hat{E}(v) = \sum_{i=1}^{n} v_i \times P_{\mathcal{V}}(v_i) \tag{3.9}$$

In smooth random mobility model, the direction of the mobile device is assumed to be uniformly distributed over a range $[0, 2\pi]$ with the probability distribution,

$$P_{\phi}(\phi) = \frac{1}{2\pi} \quad 0 \leq \phi < 2\pi. \tag{3.10}$$

Similarly, the expected direction at next time slot can be $(\hat{E}(\phi))$ for a particular mobile device can be determined from the previous $n$ number of directions and the probability distribution function of Eq. 3.10 as follows,

$$\hat{E}(\phi) = \sum_{i=1}^{n} \phi_i \times P(\phi_i). \tag{3.11}$$

From the expected angle calculated in Eq. 3.11 and the geographic location of a mobile device at current time, we can draw a line in that direction, as follows,

$$y - y_t = \chi \times (x - x_t), \tag{3.12}$$

where, $(x_t, y_t)$ is the geographic location of the mobile device $m \in \mathcal{M}$ at current time $(t)$, $\chi = \tan\phi$ is the slope of the line; and (x,y) is any co-linear point on the line. As stated in Section 3.2, each AP has a circular range of signal coverage and the location and range of each AP are known to the master cloud. So, we can assume of a circle centered at $(x_c, y_c)$ and radius $R_c$ corresponding to the geographic location and coverage range, respectively, of an AP attached with a cloudlet $c \in \mathcal{C}$. Thus, the circle covered by an AP with which a cloudlet $c \in \mathcal{C}$ is attached can be represented by Eq. 3.13.

$$(x - x_c)^2 + (y - y_c)^2 = R_c^2, \qquad \forall c \in \mathcal{C} \tag{3.13}$$

By solving Eq. 3.12 and Eq. 3.13, we can have the intersection point $(x'_c, y'_c)$, after which the mobile device would leave from the coverage area of the cloudlet $c \in \mathcal{C}$. As a

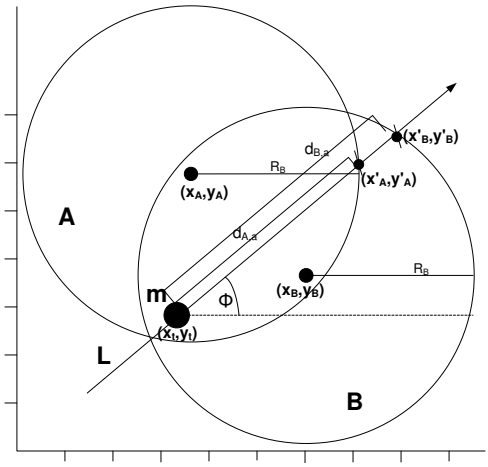


Figure 3.2: Determination of degree of associativity

result, the distance $d_{c,a}$ between the current position $(x_t, y_t)$ of the mobile device from which application request $a \in \mathcal{A}$ has come and the point of intersection $(x'_c, y'_c)$, as of Eq. 3.14, might determine the time duration for which the mobile device is connected with the cloudlet $c \in \mathcal{C}$.

$$d_{c,a} = \sqrt{(x'_c - x_t)^2 - (y'_c - y_t)^2}, \qquad \forall c \in \mathcal{C} \tag{3.14}$$

Now, the predicted time duration that a mobile device gets to execute its application $a \in \mathcal{A}$ under a cloudlet $c \in \mathcal{C}$ is termed as the degree of associativity, $\Psi_{c,a}$, determined by Eq. 3.15.

$$\Psi_{c,a} = \frac{d_c}{\hat{E}(v)} \tag{3.15}$$

In Figure 3.2 an example of determining $\Psi_c$ is illustrated. Here $A$ and $B$ are two co-located cloudlets with radius $R_A = R_B = 14$ and geographic location $(x_A, y_A) = (14, 23)$ and $(x_B, y_B) = (24, 15)$ respectively. A mobile device $m$ with location $(x_t, y_t) = (13, 11)$ is located within the range of both cloudlets and its expected direction,$\phi = 45°$. Now

if we put the values of $(x_A, y_A), R_A, (x_t, y_t)$ and $\phi$ in Eq. 3.12 and Eq. 3.13 then the intersection point of cloudlet $A$ and the line at expected direction is $(x'_A, y'_A) = (25, 28)$ approximately. Again if we put the values of $(x_B, y_B), R_B, (x_t, y_t)$ and $\phi$ in Eq. 3.12 and Eq. 3.13 then the intersection point of cloudlet $B$ and the line at expected direction is $(x'_B, y'_B) = (32, 26)$. Now from Eq. 3.14, the distance between $(x_t, y_t)$ and $(x'_A, y'_A)$ is $d_A = 20$ and the distance between $(x_t, y_t)$ and $(x'_B, y'_B)$ is $d_B = 25$. If the expected velocity is $2m/s$, we can calculate $\Psi_A = 20/2 = 10$ and $\Psi_B = 25/2 = 22.5$.

### 3.3.3 Computation of $\sigma$

Mobi-Het is a heterogeneity aware architecture. As stated in Section 3.2, like the application execution requests with diverse size and deadline requirements, the computing resources available at different cloudlets are also varied greatly from each other. Therefore, they have heterogeneous total computation capacities. Each cloudlet $c \in \mathcal{C}$ supports a set of VM types, denoted as $\mathbb{V}_c$, and each type of VM has different number of instances, represented by a corresponding set of instances $\mathbb{I}_c$. Suppose, $\zeta_i$ is the size of $i$-th VM type, then the total computation capacity ($W_c^{max}$) of a cloudlet $c \in \mathcal{C}$ can be calculated as follows,

$$W_c^{max} = \sum_{\forall i \in \mathbb{V}_c} \zeta_i \times \mathbb{I}_c(i), \tag{3.16}$$

where, $\mathbb{I}_c(i)$ represents the $i$-th element of the set $\mathbb{I}_c$. However, at a certain scheduling interval, only a portion of the cloudlet resources would be found busy for computing application requests and that determines the current working load of the incumbent cloudlet. Given that the set $\mathbb{I}'_c$ represents the current number of instances of the VM types in use for a cloudlet $c \in \mathcal{C}$, then the amount of resources used by it, $W'_c$, can be computed as follows,

$$W'_c = \sum_{\forall i \in \mathbb{V}_c} \zeta_i \times \mathbb{I}'_c(i). \tag{3.17}$$

Now, the ratio of $W'_c$ to the total computation capacity $W_c^{max}$ gives the current workload, $W_c$ of a cloudlet $c \in \mathcal{C}$ as of Eq. 3.18.

$$W_c = \frac{W'_c}{W_c^{max}}, \quad \forall c \in \mathcal{C} \tag{3.18}$$

Thus, the standard deviation ($\sigma$) for the workloads of all cloudlets is calculated as in Eq. 3.19 that quantifies the amount of variation in the workload values.

$$\sigma = \sqrt{\frac{1}{|\mathcal{C}|}\sum_{c=1}^{|\mathcal{C}|}(W_c - \mu)^2} \tag{3.19}$$

where, $\mu$ is the mean workloads of all cloudlets.

A low standard deviation indicates that the workload distribution tends to be very close to the mean, while a high $\sigma$ value indicates that the workload distribution is spread out over a wider range of values. Now, as heterogeneous cloudlets are considered, different cloudlets will be loaded with different values when application requests are allocated to the cloudlets. Workload unawre scheduling approach might increase the standard deviation significantly, depriving many requests unserved. We aim to minimize the standard deviation of cloudlet workloads as much as possible as stated in the objective function (Eq. 3.1). The optimization function trade-offs among the job execution time, associativity and the load distribution.

### 3.3.4 Calculation of $t_{c,a}$

The execution time of an application in a remote computing resource, i.e. a VM in a cloudlet, involves transmission and reception times along with the actual computation time. More explicitly, some data intensive applications may need to transfer reasonably large size of data to the remote cloudlet and the results need to be received back by the mobile device; some other compute intensive applications may need to transfer few tens of bytes only. As a result, the total response time might greatly vary depending on application data size as well as the quality of connectivity in between the mobile device

Table 3.3: RSSI vs Data Rate

| RSSI (dBm) | Data Rate ($D_{c,a}$) (Mbps) |
|---|---|
| -81 | 1 |
| -79 | 2 |
| -77 | 6 |
| -75 | 11 |
| -73 | 24 |
| -69 | 36 |
| -65 | 48 |
| -64 | 54 |

and the AP (with which a cloudlet is attached). The total time for remote execution ($t_{c,a}$) includes the time of sending the initial data ($t^s_{c,a}$), the execution time ($t^e_{c,a}$) and time to receive the result ($t^r_{c,a}$), expressed as follows,

$$t_{c,a} = t^s_{c,a} + t^e_{c,a} + t^r_{c,a} \qquad c \in \mathcal{C}, a \in \mathcal{A}. \tag{3.20}$$

The values of $t^s_{c,a}$ and $t^r_{c,a}$ highly depend on the $RSSI$ value received by the incumbent mobile device from an AP attached to the cloudlet $c \in \mathcal{C}$. When the $RSSI$ value from an AP is low, the achievable data rate of the connection would be poor and thus the time to send the application's initial data would be high and vise versa. A mapping table, as shown in Table 3.3 [46], for the $RSSI$ values and corresponding data rates can be looked up for measuring the data transfer time.

Denoting $S_a$ and $S'_a$ as the amount of data bits to be sent and received by a mobile device for executing an application $a \in \mathcal{A}$ having instruction count $\mathcal{I}_a$; and, $D_{c,a}$ be the achievable data rate of the mobile device for an AP attached to cloudlet $c \in \mathcal{C}$, then the

time components are determined as follows,

$$t^s_{c,a} = \frac{S_a}{D_{c,a}}, \qquad \forall c \in \mathcal{C}, \forall a \in \mathcal{A} \tag{3.21}$$

$$t^r_{c,a} = \frac{S'_a}{D_{c,a}}, \qquad \forall c \in \mathcal{C}, \forall a \in \mathcal{A} \tag{3.22}$$

$$t^e_{c,a} = \frac{\mathcal{I}_a}{MIPS_i}, \quad \forall c \in \mathcal{C}, \forall a \in \mathcal{A}, \forall i \in \mathbb{V}_c. \tag{3.23}$$

As cloudlets in the vicinity consists of heterogeneous signal strength, computing and storage resource , the execution time of an application highly differs when executing in different cloudlets. We aim to minimize the execution time of each remotely executed application code as much as possible as stated in the objective function (Eq. 3.1). The optimization function trade-offs among the job execution time, associativity and the load distribution.

## 3.4 Discussion

## 3.5 Conclusion

# Chapter 4

# Performance Evaluation

*In the previous chapter, we have discussed on the preliminaries and assumptions while formulating Mobi-Het mechanism that finds the most optimal resource allocation strategy for the requesting application. In this chapter, we present the detail performance evaluation results of our proposed HT-CAM and analyze its effectiveness by comparing HT-CAM with state-of-the-arts channel allocation mechanisms.*

## 4.1 Introduction

In this chapter, we study the comparative performances of the proposed Mobi-Het, ENDA [14] and MuSIC[37] systems for remote computation resource selection in mobile cloud computing environment. A test-bed environment is implemented and experiments are carried out in the University of Dhaka campus.

## 4.2 Environment Setup

We have setup a cloudlet based remote code execution environment as follows: 6 laptops of different models are used as cloudlets attached to the corresponding APs for the mobile devices. The IEEE 802.11n WLAN interfaces of the laptop are configured as hotspots. Three Android Jelly Bean Samsung Galaxy S2 mobile phones and three Android kitkat based Samsung Galaxy Grand2 are served as clients. The detailed device specifications is listed in Table 4.1. The optimal resource scheduling algorithms have been implemented

in Google Cloud, which is periodically updated about the resources that are available on the cloudlets and receives application code execution requests from the mobile devices. The scheduling interval of the cloudlet selection algorithms has been kept at $\delta$=100ms. To closely model the real world application behavior, we develop two android applications on client side: *Monitor Application* and *Code Offloading Application*. The *Monitor Application* of a mobile device periodically (with interval of 10ms) sends its location, speed, direction and RSSI values received from different APs to Google cloud. The *Code Offloading Application* allocate the application code to the selected cloudlets. The *Code Offloading Application* is invoked when the *decision maker* makes an offloading decision. For remote code execution, we have taken two example application scenarios, an $N$-Queen problem which is much computation intensive but requires less data transfer and a face detection application which requires larger amount of data transfer [47, 48, 49]. The size of each VM instance and MIPS of each VM is emulated and the clones of these applications are also running in the cloudlets for remote execution. Code is finally offloaded to a cloudlet from the mobile device after the master cloud server makes an optimal scheduling decision.

## 4.3 Performance Metrics

We compare the performances of the studied systems on the following performance matrices.

- *Percentage of requests executed:* The percentage of successful execution of requests is measured as the ratio of the number of results received with in the maximum allowable time and the number of tasks offloaded for remote execution.
- *Average time for request execution:* Execution time of an individual task is measured as the time difference between the instant at which the task starts its execution and the time instant at which the result of the task is obtained. Execution

Table 4.1: Device specification

| | **Mobile Type 1** | **Mobile Type 2** | **Cloudlet Type 1** | **Cloudlet Type 2** |
|---|---|---|---|---|
| | Samsung Galaxy SII | Samsung Galaxy Grand2 | Samsung Laptop | Dell Inspiron series Laptop |
| **CPU** | Dual-core | Quad-core | Core-i5 | Core-i3 |
| **Clock Speed** | 1.2 GHz | 1.2 GHz | 2.1 GHz | 1.8 GHz |
| **Memory** | 1 GB | 1.5 GB | 8 GB | 4 GB |
| **Operating System** | Android OS v4.1 (Jelly Bean) | Android OS v4.4.2 (KitKat) | Windows 7 64 bit | Windows 7 32 bit |

times of individual tasks are averaged over the total number of executions of different tasks during the experiments.

- *Standard deviation of cloudlet computation loads:* The standard deviation of cloudlet computation loads defines the deviation of a cloudlet's workload from the average workload distribution on all cloudlets and is measured by Eq. (3.19). The value of the standard deviation $\sigma$ indicates how well the load is distributed among the cloudlets. The smaller is the value, the better is the capability of balancing the workloads.

- *Average scheduling delay per request:* The time for determining optimal allocation of cloudlets for submitted requests at each interval is measured as the time difference between the instant at which a request is submitted to the master cloud and the instant at which the mobile device gets the selection result and then average is taken for all requests. The lower value is corresponding to smart scheduling system.

## 4.4 Experiment Results

To evaluate the robustness of our proposed Mobi-Het resource allocation algorithm in real environment, we study the performances for varying number of mobile devices, mobility speeds and size of the application requests.

### 4.4.1 Impacts of varying mobility speeds

In this section, we discuss the impacts of increasing mobility speeds of the users, ranging from 0m/s to 10m/s, on to the performances of the studied systems. In this case, the number of APs is fixed at 6 and 4 mobile devices are used for experiments. Each mobile device sends request for $N$-Queen application with random queen size, ranging from $3 \sim 16$. Each data point of the graph is corresponding to the average of results from 10 different experiments.

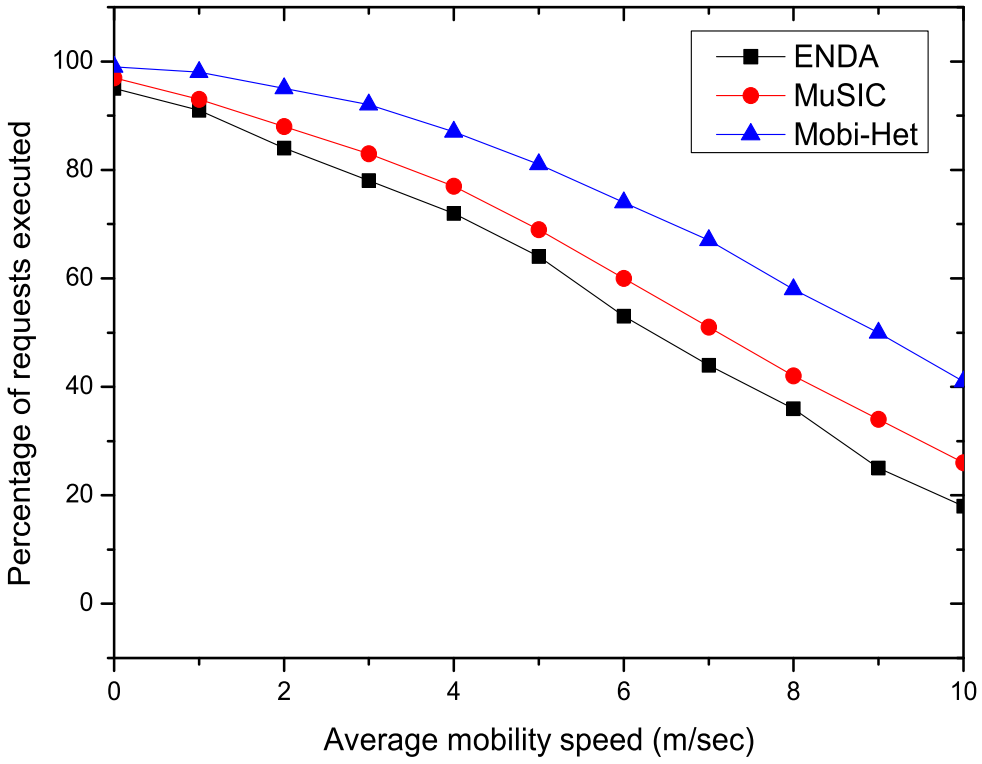


Figure 4.1: Percentage of application successfully completed

The graphs of Fig. 4.1 state that the percentage of requests executed successfully gradually decreases for increasing user mobility speeds in all the studied remote code execution systems. Our proposed Mobi-Het algorithm provides with more stable performance compared to ENDA and MuSIC. This happens because ENDA and MuSIC do not exploit the mobility patterns of users to predict future associativity with the APs. For high speed movements, the time duration of a mobile user under any cloudlet becomes less, resulting in high failure rate of the submitted and assigned job executions for ENDA and MuSIC. The comparative performance of Mobi-Het is better due to its smart task execution mapping through prediction of the degree of associativity of a mobile device with the different APs.

Figure 4.2 shows that the average execution time of job requests is increased with

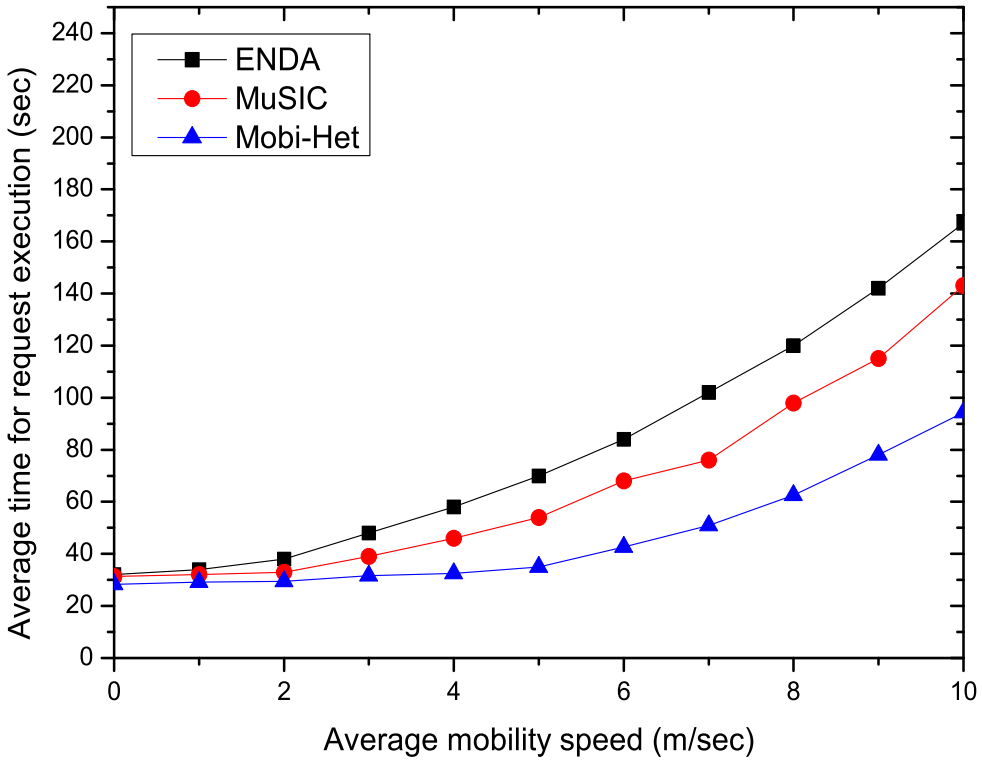


Figure 4.2: Average execution time

the mobility speed. This execution time includes the data transmission, computation, waiting and scheduling delays. As a user is moving faster, the level of network strength changes so frequently that data transfer time increases sharply. In addition to that, when the movement is faster, the associativity of a mobile device with an AP rapidly changes, resulting in increased retransmissions and waiting time. As a result, the average execution time increases. As stated above, the mobility-aware task execution assignment of Mobi-Het helps it to experience reduced number of failures and retransmissions as well as reduced waiting time and thus it achieves better delay performance compared to ENDA and MuSIC systems.

Figure 4.3 illustrates that the standard deviation of cloudlet computation loads increases significantly with increasing speed levels. As high mobility speed causes rapid

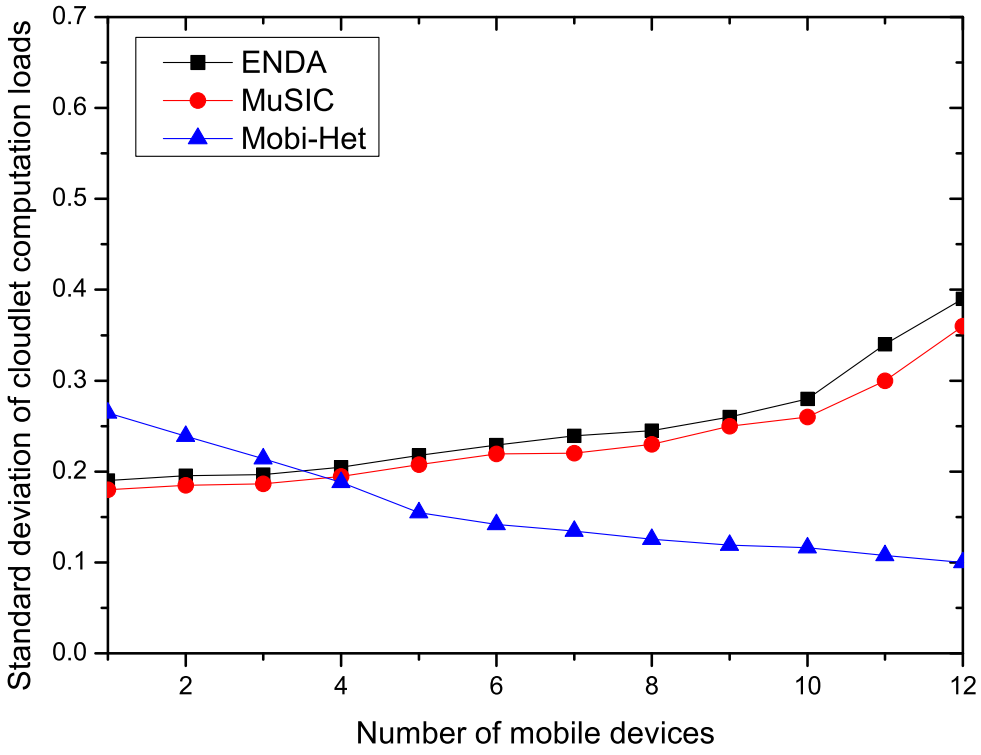


Figure 4.3: Standard deviation of cloudlet computation loads

movement from the range of one cloudlet to another, system is bound to allocate requests to the cloudlets in the range even when those are loaded enough. Our proposed Mobi-Het algorithm gives better performance than ENDA and MuSIC because Mobi-Het predicts the next possible direction and location of any user so that it can rationally allocate requests to those cloudlets toward which the user has much possibility of moving forward.

Each request arrived from the mobile devices has to wait for few amount of time before allocation due to scheduling purpose. Figure 4.4 depicts the scheduling delay per request, in which delay increases with the speed. As the network connectivity is not uniformly distributed over the area and also each cloudlet have heterogeneous capacity and power, so the resource scheduling systems give better result when the speed is low.

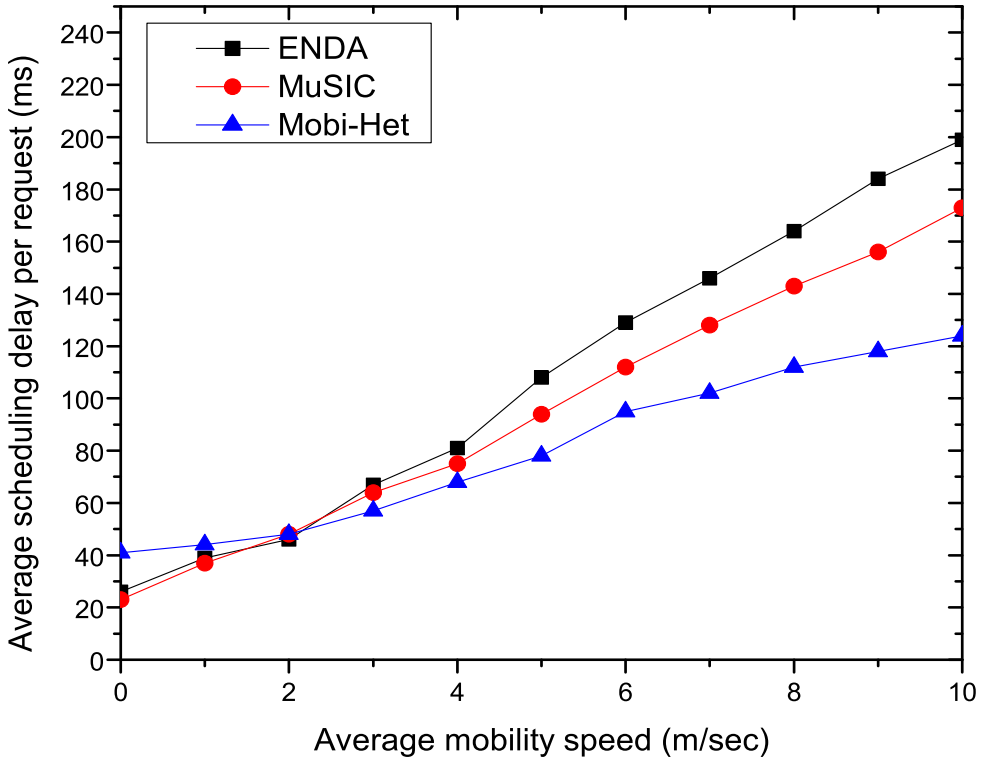


Figure 4.4: Operation overhead

At high speeds, users move out from the range of current cloudlet at which it's request is assigned to executem, resulting in prolonged waiting time.

### 4.4.2 Impacts of varying mobile devices

In this section, we discuss the impacts of number of mobile devices, ranging from 1 to 6, on to the performances of the studied systems. In this experiment, the number of APs is fixed at 6, the mobility speed is kept at 1-3 m/s and the size of submitted $N$-Queen application request is randomly chosen from the range $3 \sim 16$.

Figure 4.5 shows that the percentage of requests executed successfully at remote cloudlets gradually decreases with the increasing number of mobile devices in all the

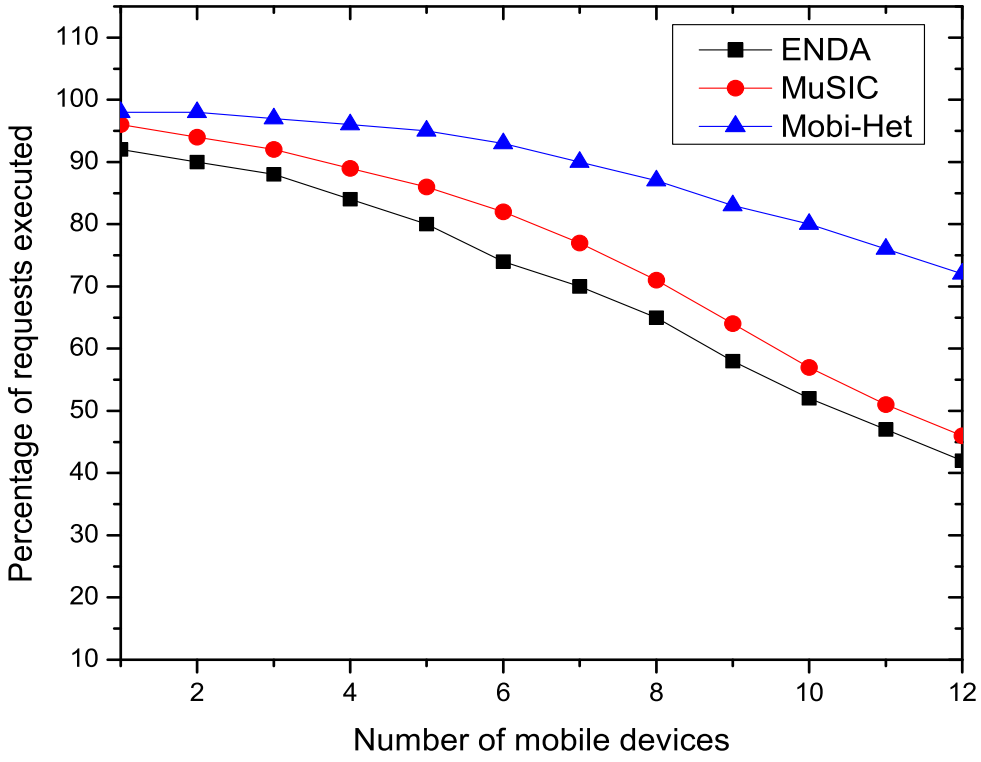


Figure 4.5: Percentage of application successfully completed

studied scheduling algorithms. However, the performance gap between our proposed Mobi-Het algorithm and the other systems increases for larger number of mobile devices. This is caused by the fact that both ENDA [14] and MuSIC [37] focus on individual request allocation to the immediately available resources rather than scheduling the multiple requests together considering heterogeneities in application and hardware resources. Our in-depth look into the scheduling behavior of ENDA and MuSIC reveals that often lightweight requests are assigned to comparatively high power computing resources, leaving no room for forthcoming high priority and heavyweight requests. As a result, their successful execution rate is decreased fastly compared to the Mobi-Het system for increasing computation loads. In addition to that, the locality-driven resource allocation in Mobi-Het, by exploiting mobility prediction and signal strengths, helps it to reduce job

execution failures significantly due to user relocations and poor connectivity problems.

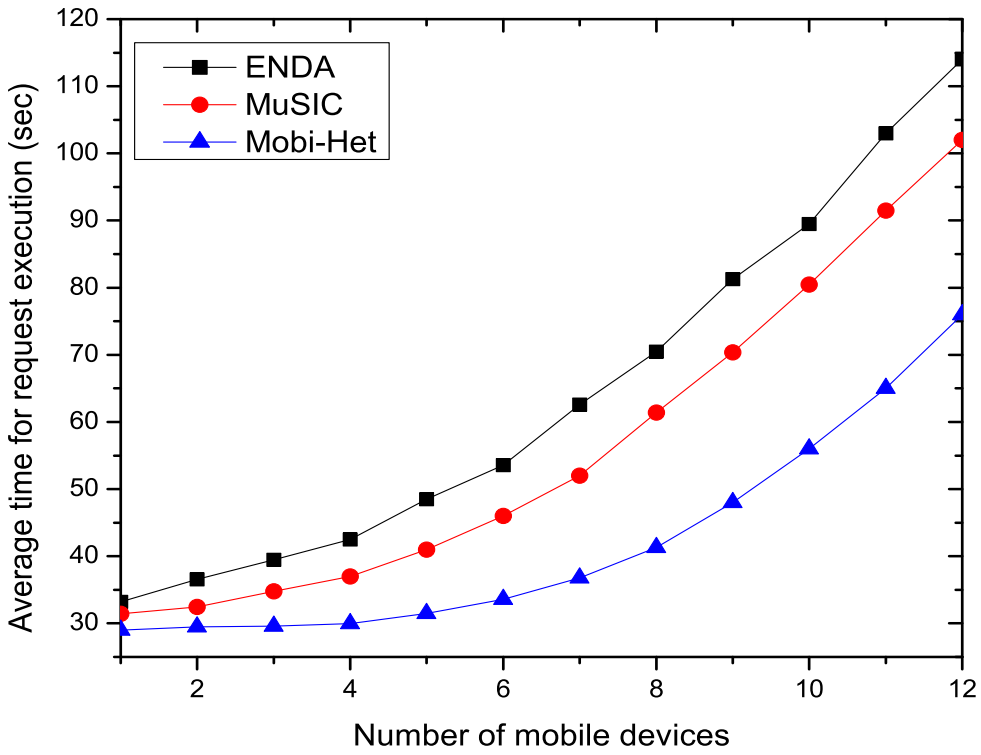


Figure 4.6: Average execution time

Figure 4.6 shows that the average time for request execution for increasing number of mobile devices for all the studied algorithms. The performance gap between our proposed Mobi-Het algorithm and state-of-the-art works (ENDA and Mobi-Het) increaes with the number of mobile devices. For lower number of requests, all the studied algorithms take the same amount of time. The estimation of velocity and its direction for every mobile user and allocating requests accordingly results in reduced number of interferences during execution in our Mobi-Het system; whereas, other approaches lack consideration of degree of associativity with the APs and resource heterogeneity and thus they elapse much tiime for transmission, re-submission and re-execution of requests.

Figure 4.7 describes how the standard deviation of cloudlet computation loads is

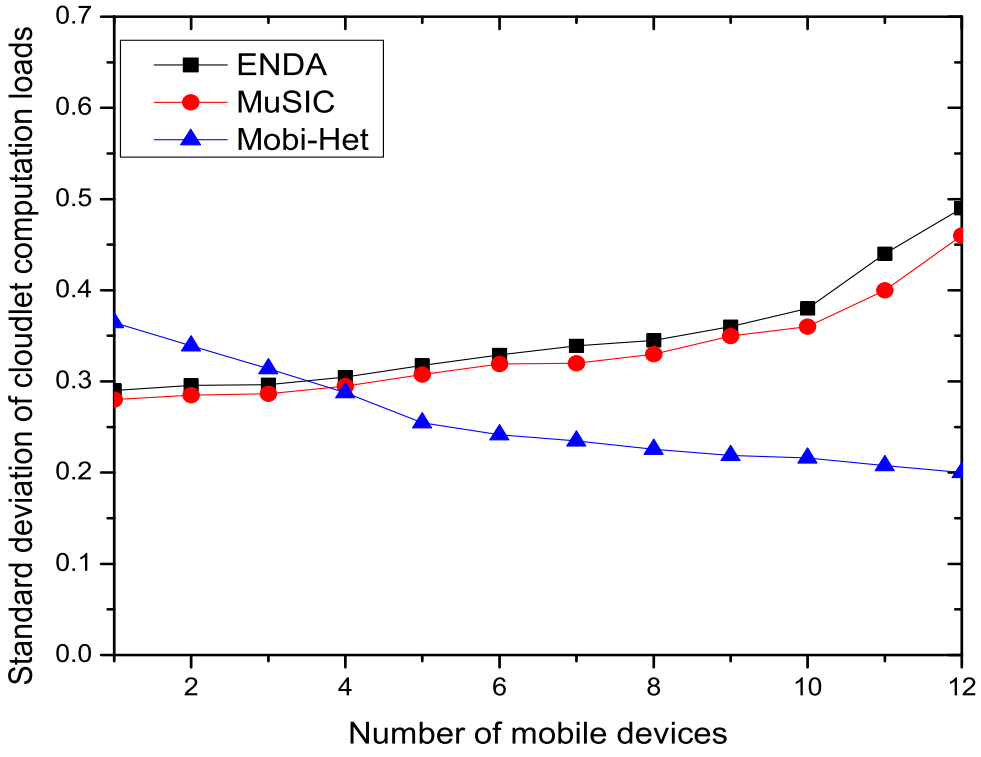


Figure 4.7: Standard deviation of cloudlet computation loads

affected by varying number of mobile devices. In our proposed Mobi-Het system, it has been observed that initially when the workload of all the cloudlets is zero and few requests arrive within an interval then Mobi-Het allocates optimal cloudlets to those requests and other cloudlets poses no workload at all. As a result, initially, the amount of deviation from the mean value is significantly higher in Mobi-Het than the other two algorithms. ENDA and MuSIC do not consider workload distribution while allocating resources to the cloudlets. So, in initial situation, the standard deviation is almost same for each of them. However, when number of requests increases, the workload distribution of Mobi-Het becomes more balanced, thus the standard deviation of computation loads gradually decreases due to considering workload heterogeneity while for ENDA and MuSIC the standard deviation gradually increases with increasing number of mobile devices,

as expected theoretically.

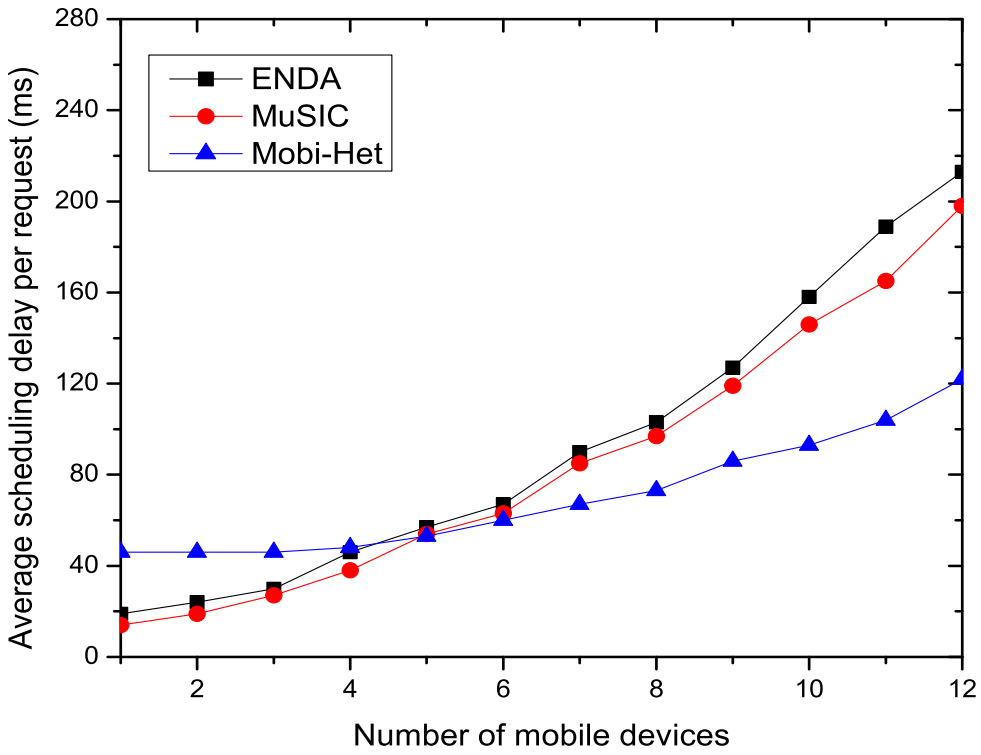


Figure 4.8: Operation overhead

Figure 4.8 depicts that scheduling delay per request for all the studied remote execution methods for random data size. Our proposed Mobi-Het system schedules all the requests after each time interval rather than allocating each request individually, as were done by ENDA and MuSIC systems. As a result, compared to other algorithms, Mobi-Het takes much time for scheduling when the number of requests is low. This scheduling delay is almost stable for some duration in which each request can be scheduled to a dedicated cloudlet without any further waiting. However, when number of random sized requests is increased, the waiting queue for ENDA and MuSIC sharply increases due to the fact that resources are engaged in executing other requests for prolonged time span caused by poor resource allocation. The Mobi-Het experiences reduced scheduling delay

when the system is laoded with sufficient computation demands.

### 4.4.3 Impacts of input data size

In this section, we discuss the impacts of input data size of both an $N$-Queen application and a face detection application for varying number of queens and varying image sizes. Here, we have fixed the number of mobile devices to 4, the average speed at 2m/sec, and the number of APs at 6.

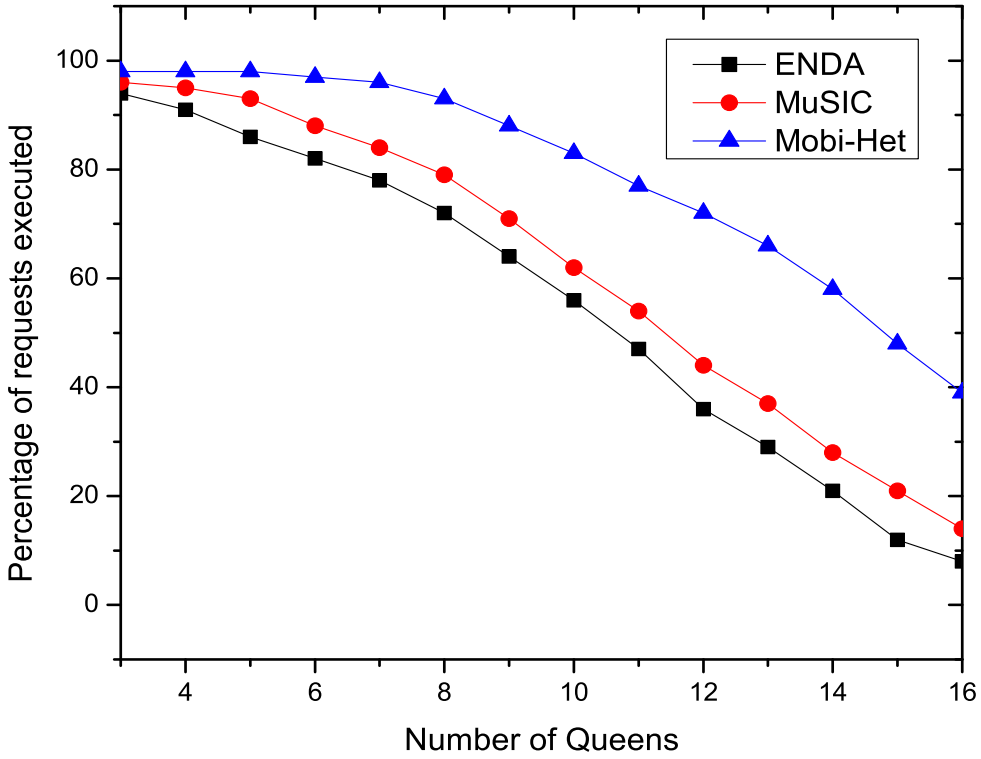


Figure 4.9: Percentage of application successfully completed

The graphs of Fig. 4.9 shows that the percentage of application requests executed successfully is decreased slowly with increasing number of queens; however, the average execution time increases exponentially for all the studied remote execution methods,

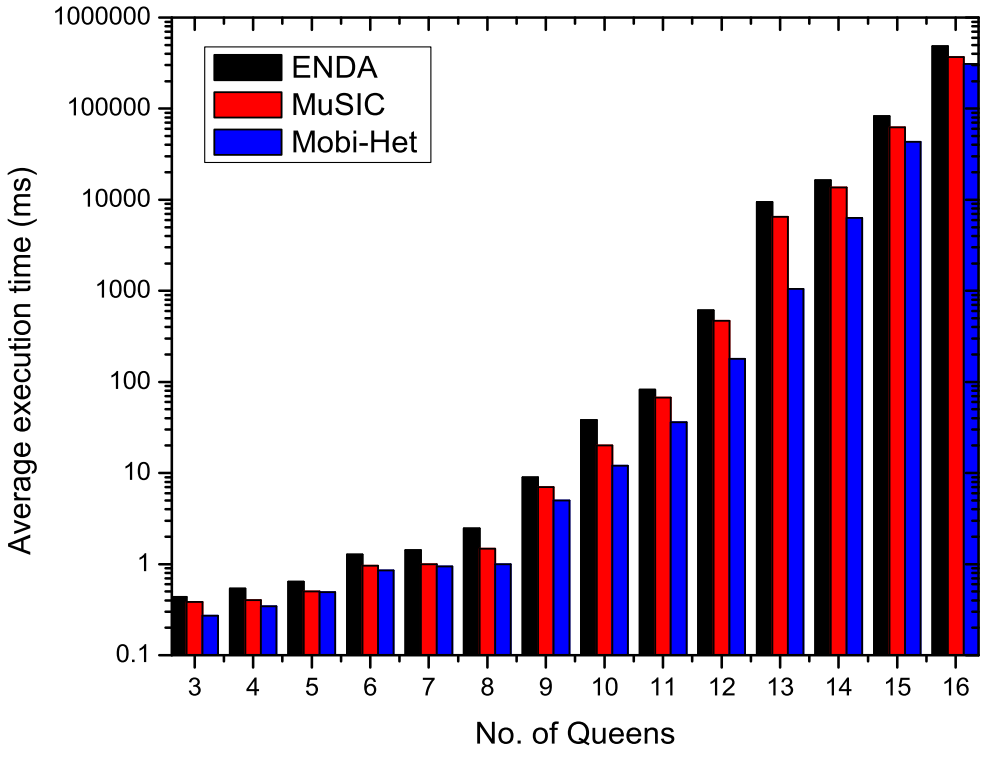


Figure 4.10: Average execution time

as shown in Fig. 4.10. The reason behind these results is that the resource allocation of Mobi-Het is both mobility and resource-heterogeneity aware, whereas the others are not. Similarly, the percentage of application requests executed successfully and average execution time required by Mobi-Het is much better compared to other approaches, as shown in Fig. 4.11 and Fig. 4.12, respectively.

## 4.5 Conclusion

As the image size increases, the Mobi-Het performs much better compared to ENDA and MuSIC since the former is able to reduce transmission time significantly by considering signal strengths as one of the criteria for choosing a cloudlet resources.

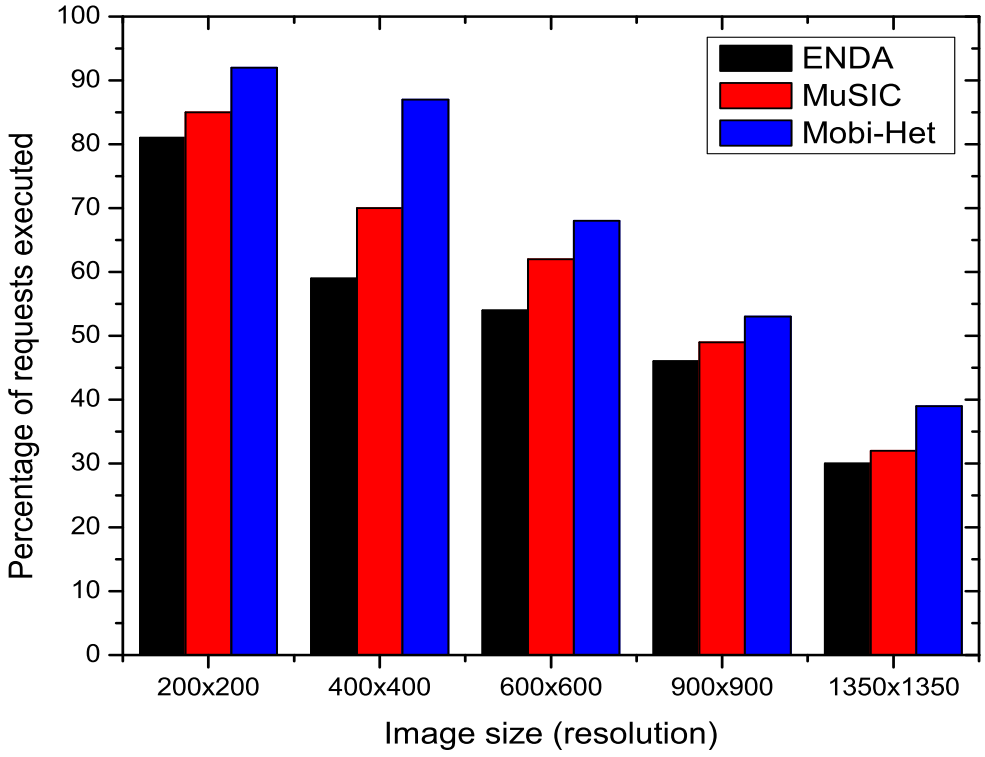


Figure 4.11: Percentage of application successfully completed

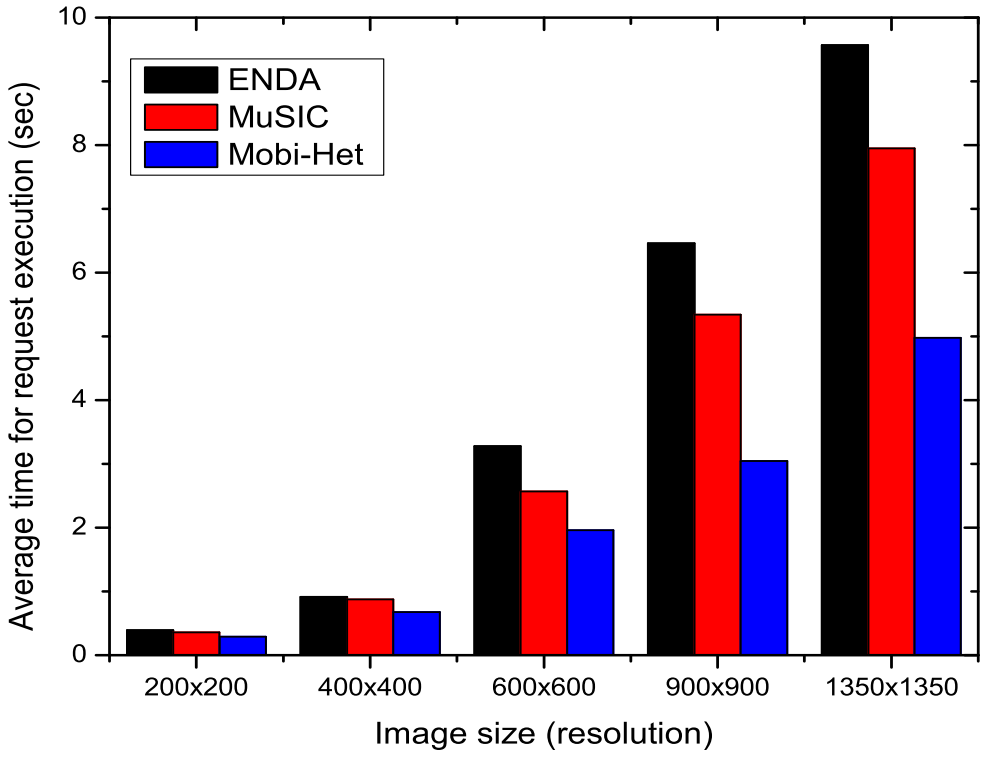


Figure 4.12: Average execution time

# Chapter 5

# Conclusions

*In this chapter, we summarize the research results presented in this thesis and state few directions for future works.*

## 5.1 Summary of Our Work

## 5.2 Discussion

In this chapter, we summarize the research results presented in this thesis. We propose Mobi-Het, a three-tier architecture that makes adaptable energy efficient offloading decisions in constantly changing environment, such as frequent user movements, varying server loads, network strength and cloudlet heterogeneity. MACS places most operations of decision making on clouds and cloudlets, and only requires smartphones to communicate very few data with clouds. We present how MACS predicts user's next time duration under each cloudlet within the range based on collected user speed and direction at each time interval and how it selects the most optimal cloudlet for each application request.

We evaluate its performance under real world test bed implementation. Our experiment results demonstrate that Mobi-Het outperforms existing approaches in terms of success percentage, quality of experience (QoE) and workload distribution.

## 5.3 Future Work

# Bibliography


[1] D. Ferreira, A. K. Dey, and V. Kostakos, "Understanding human-smartphone concerns: a study of battery life," in *Pervasive Computing*. Springer, 2011, pp. 19–33.

[2] A. Smith, "Smartphone ownership–2013 update," *Pew Research Center: Washington DC*, 2013.

[3] "Survey reveals more smartphones activated each day than babies born (accessed on april 12, 2014)," http://www.netgenie.net/blog/survey-reveals-more-smartphones-activated-each-day-than-babies-born.

[4] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 140–150, 2010.

[5] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "The smartphone and the cloud: Power to the user," in *Mobile Computing, Applications, and Services*. Springer, 2012, pp. 342–348.

[6] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[7] M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing.* ACM, 1996, pp. 1–7.

[8] H. Wang, "Accelerating mobile-cloud computing using a cloudlet," Master's thesis, University of Rochester. Department of Electrical and Computer Engineering, Rochester, NY., 2013.

[9] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, 2014.

[10] F. a. Yaser Jararweh, Lo'ai Tawalbeh and F. Dosari, "Resource efficient mobile computing using cloudlet infrastructure," in *IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks.* IEEE, 2013.

[11] D. Huang *et al.*, "Mobile cloud computing," *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, vol. 6, no. 10, pp. 27–31, 2011.

[12] M. Mehta, I. Ajmera, and R. Jondhale, "Mobile cloud computing," *International journal of Electronics*, 2013.

[13] P. A. Cox, "Mobile cloud computing," *IBM developerWorks*, pp. 1–10, 2011.

[14] J. Li, K. Bu, X. Liu, and B. Xiao, "Enda: embracing network inconsistency for dynamic application offloading in mobile cloud computing," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing.* ACM, 2013, pp. 39–44.

[15] "Gartner says worldwide mobile device sales to end users reached 1.6 billion units in 2010; smartphone sales grew 72 percent in 2010 (accessed on april 11, 2014)," http://www.gartner.com/newsroom/id/1543014.

[16] S. S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," in *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, vol. 2. IEEE, 2011, pp. 3–6.

[17] D. Helbing, S. Bishop, R. Conte, P. Lukowicz, and J. B. McCarthy, "Futurict: Participatory computing to understand and manage our complex world in a more sustainable and resilient way," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 11–39, 2012.

[18] M. Conti, S. Giordano, M. May, and A. Passarella, "From opportunistic networks to opportunistic computing," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 126–139, 2010.

[19] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[20] S. I. B. Chun, M. N. P. Maniatis, , and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *EuroSys '11 Proceedings of the sixth conference on Computer systems*. ACM New York, NY, USA: ACM, 2011, pp. 301–314.

[21] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[22] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: Computation offloading as the bridge," *IEEE NETWORK*, vol. 27, no. 5, pp. 28–33, 2013.

[23] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond.* ACM, 2010, p. 6.

[24] S. W. L. Niroshinie Fernando and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, pp. No. 29 (2013) 84–106, 2013.

[25] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges," *IEEE Communications Surveys and Tutorials 99*, 2013.

[26] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *Wireless Communications, IEEE*, vol. 20, no. 3, 2013.

[27] S. Abolfazli, Z. Sanaei, A. Gani, F. Xia, and L. T. Yang, "Rich mobile applications: Genesis, taxonomy, and open issues," *Journal of Network and Computer Applications*, 2013.

[28] Y. E. Gelogo and H.-K. Kim, "Mobile hybrid cloud computing issues and solutions," *Advanced Science and Technology Letters*, vol. 29, pp. 341–345, 2013.

[29] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient customization of mobile computing infrastructure," in *Proceedings of the First Workshop on Virtualization in Mobile Computing.* ACM, 2008, pp. 37–41.

[30] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 3, pp. 1294–1313, 2013.

[31] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.

[32] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, ser. MCS '12. New York, NY, USA: ACM, 2012, pp. 29–36.

[33] D. Chalmers and M. Sloman, "A survey of quality of service in mobile computing environments," *Communications Surveys & Tutorials, IEEE*, vol. 2, no. 2, pp. 2–10, 1999.

[34] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953.

[35] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, 2015.

[36] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, "etime: energy-efficient transmission between cloud and mobile devices," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 195–199.

[37] M. R. Rahimi, N. Venkatasubramanian, and A. V. Vasilakos, "Music: Mobility-aware optimal service allocation in mobile cloud computing," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 75–82.

[38] C. Bettstetter, "Smooth is better than sharp: a random mobility model for simulation of wireless networks," in *Proceedings of the 4th ACM international workshop*

*on Modeling, analysis and simulation of wireless and mobile systems.* ACM, 2001, pp. 19–27.

[39] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *Personal Communications, IEEE*, vol. 8, no. 4, pp. 10–17, 2001.

[40] J. Flinn, S. Park, and M. Satyanarayanan, "Balancing performance, energy, and quality in pervasive computing," in *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on.* IEEE, 2002, pp. 217–226.

[41] M. D. Kristensen, "Scavenger: Transparent development of efficient cyber foraging applications," in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on.* IEEE, 2010, pp. 217–226.

[42] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1232–1243, 2012.

[43] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[44] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services.* Springer, 2012, pp. 59–79.

[45] D. Shivarudrappa, M. Chen, and S. Bharadwaj, "Context-aware decision engine for mobile cloud offloading," in *Proc. of IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, April 2013, pp. 111–116.

[46] M. Tauber and S. N. Bhatti, "Low rssi in wlans: Impact on application-level performance," in *Computing, Networking and Communications (ICNC), 2013 International Conference on.* IEEE, 2013, pp. 123–129.

[47] D. Shivarudrappa, M. Chen, and S. Bharadwaj, "Cofa: Automatic and dynamic code offload for android," *Technical report published by University of Colorado, Boulder*, 2012.

[48] http://developer.android.com/guide/components/processes-and-threads.html, accessed: May 1,2015.

[49] S. Yang, Y. Kwon, Y. Cho, H. Yi, D. Kwon, J. Youn, and Y. Paek, "Fast dynamic execution offloading for efficient mobile cloud computing," *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, vol. 0, pp. 20–28, 2013.

# Appendix A

# List of Notations

| | |
|---|---|
| $\mathcal{M}$ | Set of all mobile devices |
| $\mathcal{A}$ | Set of all application requests arrived in a scheduling interval $\delta$ |
| $\mathbb{C}$ | Set of all cloudlets in the network |
| $\mathcal{C}$ | Set of all cloudlets within the range of each mobile |
| $\mathbb{V}_c$ | Set of all virtual machine instances available at cloudlet $c \in \mathcal{C}$ |
| $t_{c,a}$ | Execution time of an application $a$ if computed in cloudlet $c \in \mathbb{C}$ |
| $T_a$ | Maximum allowable time for executing an application $a \in \mathcal{A}$ |
| $W_c$ | Workload of cloudlet $c \in \mathcal{C}$ |
| $\delta$ | Scheduling interval in the master cloud. |

# Appendix B

# List of Publications

1. ————————, "Interference-Aware High Throughput Channel Allocation Mechanism for CR-VANETs," *Submitted to Elseviers Computer Communications*, April 2015.

2. ————————, "An Energy Aware Event-Driven Routing Protocol for Cognitive Radio Sensor Networks," *Submitted to Elseviers Wireless Networks*, March 2015.

3. ————————, "An energy-efficient common control channel selection mechanism for cognitive radio ad hoc networks," *Annals of Telecommunications*, vol. 70(1-2), pp.11-28, February, 2014 (10.1007/s12243-014-0420-0).