QoS and Mobility Aware Optimal Resource Allocation for Dynamic Application Offloading in Mobile Cloud Computing

Submitted By

Mahinur AKter
ID: 2012-2-60-022
&
Fatema Tuz Zohra
ID: 2012-1-60-020

Supervised By

Amit Kumar Das
Lecturer
Department of CSE, EWU

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

December, 2016

Declaration

We hereby declare that, this thesis work was done under CSE497 and has not been submitted elsewhere for requirement of any degree or diploma or for any purpose except for publication.

____

Mahinur Akter

ID: 2012-2-60-022

Department of Computer Science and Engineering

East West University

____

Fatema Tuz Zohra

ID: 2012-1-60-020

Department of Computer Science and Engineering

East West University

Letter of Acceptance

This Thesis Project entitled "**QoS and Mobility Aware Optimal Resource Allocation for Dynamic Application Offloading in Mobile Cloud Computing**" submitted by Mahinur Akter (ID: 2012-2-60-022) and Fatema Tuz Zohra (ID: 2012-1-60-020), to the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh is accepted by the department in partial fulfillment of requirements for the Award of the Degree of Bachelor of Science and Engineering on December, 2016.

Supervisor

____

Amit Kumar Das

Lecturer

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

Chairperson

____

Dr. Mozammel Huq Azad Khan

Chairperson and Professor

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

Article in Review

From the outcome of this project a refereed conference paper has been submitted. This paper is in review processing which is in below for reference.

Conference Paper

M. Akter, F. T. Zohra, A. K. Das, "**QoS and Mobility Aware Optimal Resource Allocation for Dynamic Application Offloading in Mobile Cloud Computing**", International Conference on Electrical, Computer & Communication Engineering (ECCE), Cox's Bazaar, Bangladesh, 2017.

Acknowledgements

Alhamdulillah, all praises to Allah for the strengths and His blessing in our entire student life and finally completing this thesis work.

We would like to express our sincere gratitude towards our supervisor Amit Kumar Das for his continuous support in our thesis, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research and writing of this thesis book. The door to his assistance was always open whenever we had a trouble or had a question about our research or writing. We could not have imagined having a better supervisor and mentor.

Our deep gratitude to our all faculty members who have supported us in East West University. Their devotion towards our improvement helped us to increase our knowledge of the individual subject and enabled us to complete our studies in time.

Finally, we must express our very profound gratitude to our parents for providing us with unfailing support and continuous encouragement throughout our years of study and through the process of researching and writing this thesis book. This accomplishment would not have been possible without them.

Abstract

Smartphone applications are getting more multifarious and demanding of increased energy and computing resources. Mobile Cloud Computing (*MCC*) made a novel platform which allows personal Smartphone to execute heavy computing tasks with the assistance of powerful cloudlet servers attached to numerous wireless access points (*APs*). Furthermore, due to users' mobility in anywhere, ensuring the continuous connectivity of mobile devices in given wireless network access point is quite difficult, because the signal strength becomes sporadic at that time. In this paper, we have proposed *QoS* and mobility aware optimal resource allocation architecture for remote code execution in the cloud that offers higher efficiency in timeliness and reliability domains. By carrying continuous track of user, our proposed architecture performs the offloading process. Our test-bed implementation results show that our assumed system model outperforms the state-of-the-art methods in terms of success percentage, execution time and workload distribution.

Table of Contents

List of Figures

List of Algorithms

Chapter 1: Introduction

1.1 Overview

Due to recent pioneering achievement in mobile and wireless communication technologies, mobile devices such as Smartphone, tablets, laptops already have taken a large market through worldwide. Many prominent research and statistical studies are predicting 5.5 billion mobile users by 2020, which will represent 70 percent of the global population [*1*]. This assumption and current statistics of a large number of users are encouraging mobile application developers to develop different types of motivational, complex and strategy based applications. Mostly, these types of applications occupy a large amount of computational power and energy in the device which causes the shortage of battery lifetime. It is also very unhealthy for a human being. Moreover, for the increasing number of the user, handling mobile traffic with users' free movement is also a big challenge for the network provider.

Mobile cloud computing (MCC) is a Masonic pattern where data storage and CPU intensive tasks are performed on cloud and mobile devices are mainly used as a sleazy client to interact with the application and rendering the results processed from the cloud. This architecture is very much helpful to gain the battery lifetime and decrease the energy consumption for the large size of applications. To the contrary, all computations and observations are guided by the cloud is basically complicated. As a result, cloudlet (CL) concept has risen to reduce the computational pressure for the cloud and represents the middle tier of a 3-tier hierarchy: *mobile device - cloudlet - cloud*. It supports resource-intensive and interactive mobile applications by providing powerful computing resources to mobile devices with lower latency. By gathering information about the demanding tasks and necessary environmental fact of the user, cloud distributes the offloading responsibility to the cloudlets.

In this project, we have designed a system to reduce the mobile traffic by setting more scalable, efficient and fast technique for mobile code offloading through cloud and cloudlet. According to our proposal, the cloud will distribute the tasks among cloudlets by maintaining a highly efficient way as well as will reduce the dependency between cloud and user. Moreover, in random user mobility, our system would give less chance to interrupt in providing optimal resource allocation during offloading in cloud and cloudlet.

1.2 Motivation

MCC technology is the enabling technology which promises the realization of pervasive computing era because integrated mobile devices, sensors and virtual entities are widely used nowadays. It provides minimum communication cost with the mobile platform. Moreover, researches on more effective and potential cloud-based computation are showing an advancement future research scope.

With the increase in computer and mobile user's, data storage has become a priority in all fields. Large and small scale businesses today thrive on their data and they spent a huge amount of money to maintain this data. It requires a strong IT support and a storage hub. Not all businesses can afford the high cost of in-house IT infrastructure and backup support services. For them, cloud computing is an accessible solution. In addition, cloud computing gives the freedom to use services as per the requirement and pay only for what we use. Due to cloud computing, it has become possible to run IT operations as an outsourced unit without many in-house resources.

1.3 Mobile Cloud Computing

Cloud computing is a trading clop for modern technologies which provide software, data access and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. However, *MCC* is the combination of mobile computing, cloud computing, and wireless networks to serve rich computational resources to the mobile users and cloud computing providers at anywhere anytime through internet based on the pay-as-you-use principle. It enables users to quickly and securely collect and integrate data from various sources, regardless of where it resides. It also improves reliability with information backed up and stored in the cloud. Moreover, mobile applications that run on the cloud are not constrained by device storage and processing resources. Only data intensive processes can run in the cloud.

1.3.1 Basic of *MCC* Architecture

MCC uses computational augmentation approaches by which resource-constraint mobile devices can utilize computational resources of varied cloud-based resources. There are four different cloud models that we can consent according to business needs: public cloud, private cloud, community cloud and hybrid cloud. Public cloud computing resources are owned, governed and operated by the government, an academic or business organization. Private cloud computing resources are deployed for one particular organization and mostly used for intra-business interactions. Community cloud computing resources are provided for a community and organizations. And hybrid type cloud is used for both types of interactions - *B2B* (Business to Business) and *B2C* (Business to Consumer).

1.3.2 Advantages of *MCC*

Cloud computing offers a number of advantages such as scalability, agility and economy efficiency, in comparison of traditional IT infrastructure. It virtualizes physical and software resources and provides generic services (e.g. IaaS, SaaS, etc).

So, it is regarded as a new paradigm and it is dramatically changing the landscape of information technologies. Meanwhile, contributed by the rapid deployment of broadband wireless networks and fast growth of Smartphone, more and more users are using mobile phones to access Internet services.

There are some other advantages of *MCC*:

- It enables mobile users to store and access large data on the cloud. Mobile applications are no longer constrained by the storage capacity of the device.
- The data and services in the cloud are always available even when the users are moving from one place to another place.
- Storing data and applications in the cloud reduce the potential for loss of data in the event of a hardware failure, improving reliability and availability.
- Service providers can easily add and expand their service offerings.
- Multiple services from different providers can be integrated easily through the cloud to meet today's complex user demands.
- *MCC* can be designed with a comprehensive data security model for both service providers and users by allowing protected copyrighted digital contents in the cloud. *MCC* providers have security services in place such as virus scanning, malicious code detection, and authentication for mobile users.

1.3.3 Challenges in *MCC*

The wide spread of Smartphone and their capabilities made them an important part of many people's life over the world. However, there are many challenges facing these devices such as low computing power and fast energy drain from their batteries. One solution is to use mobile cloud computing services to run certain tasks in the cloud and returning back the results to the mobile device saving space and processing power.

In the *MCC* landscape, a combination of mobile computing, cloud computing, and communication networks creates several complex challenges such as mobile computation offloading, seamless connectivity, long *WAN* latency, mobility management, context-processing, energy constraint, vendor/data lock-in, security and privacy,[*9*] elasticity that hinder *MCC* success and adoption.[*10*]

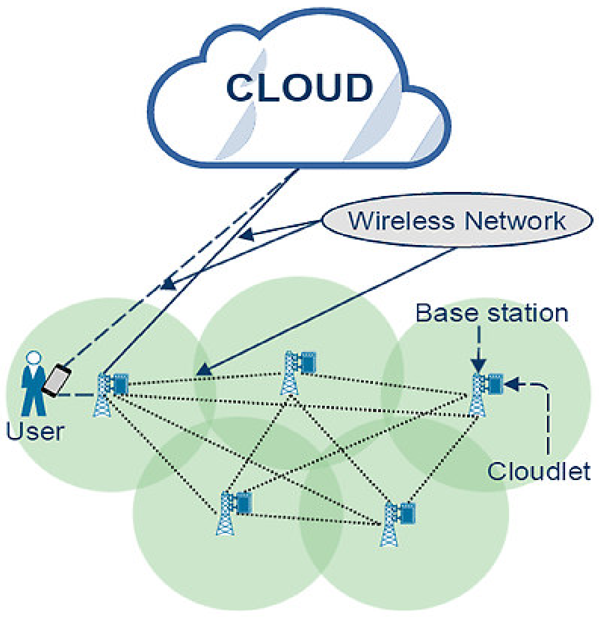

Figure 1: Three-tier MCC architecture

1.3.4 What is Cloudlet?

Gaining high capabilities (such as *CPU*, memory and etc.) in mobile devices also degrades execution in complex rich media and data analysis applications when offloading to the cloud because of the high *WAN* latencies, especially for applications with real-time constraints such as augmented reality. Therefore, small-scale data center namely cloudlet performs better cloud computing service by placing it in each base station. The goal of cloudlet is to increase the response time of applications running on mobile devices by using low latency, high-bandwidth wireless connectivity and by hosting cloud computing resources physically closer to the mobile devices accessing them. This is intended to eliminate the *WAN* latency delays that can occur in traditional cloud computing models.

1.4 Proposed Solution Model

The application code offloading mechanism from resource poor mobile device to remote server has been well studied in the literature [*6*]. The code offloading facilitates faster execution as well as resource utilization. Research techniques have already worked on many solutions to the issues of computational power and battery lifetime by offloading tasks on the cloud. Prominent among them are MAUI [*4*] and the CloneCloud [*5*] projects.

MAUI [*4*] is a system that creates the opportunity of fine-grained energy-aware offload of mobile code to the infrastructure. It does not heavily rely on programmer support to partition an application, and not even require full process (or full *VM*) migration.

CloneCloud [*5*] proposes offloading process using VM images as a powerful virtual device. Other techniques related to mobile code offloading worked on the static analyzer, dynamic profiler and optimization solver [*3*].

Cloudlets [*8*] introduce the concept of using nearby highly resourceful computers, to which Smartphone connects over wireless *LAN*. This concept helps to save higher latency and bandwidth trouble, which would have cost to be connected to the cloud constantly.

ThinkAir [*2*] focuses on the elasticity and scalability of the cloud and enhances the power of mobile cloud computing by parallelizing method execution using multiple *VM* images.

ENDA [*3*] try to work with the network consistency for dynamic application offloading, where they predict a path for the user and always care the continuous user track by storing their previous and current states. In real life scenario, this can hardly be assumed always.

All these existing offloading techniques assume that network performance will always remain consistent which is also hardly possible in the real scenario because of user mobility.

1.5 Contributions

In our thesis work, we propose the same approach of using Smartphone *VM* image inside the cloud for handling heavy computational offloading. Different from them, our system model connects the cloudlets with every base station to be under the mobile network always and also deals with a commercial cloud scenario with multiple mobile users instead of a single user. Hence, we focus not only on the offloading efficiency and convenience for developers but also on the elasticity and scalability of the cloud side for the dynamic demands of variant customers.

1.6 Organization of the Thesis Work

The rest of this book chapters are organized as – chapter 2 describes background and motivation of our proposed system; chapter 3 describes our system model and assumptions; chapter 4 describes framework activities; chapter 5 shows experimental evaluation of our work with proper explanation; chapter 6 is concluded discussing overall of our works along with the direction of our future research work. Finally, proper references of our thesis work and a list of acronyms and notations are depicted respectively.

Chapter 2: Background and Motivation

2.1 Overview of Background and Motivation

The concept of mobile cloud computing provides a great opportunity for the development of mobile applications since it allows the mobile devices to maintain a very thin layer for user applications and transfer the computation and processing overhead to the virtual environment. Famous researchers tried to give new directions to enhance the MCC field. Among the research areas in MCC, we are mostly concern on QoS and mobility aware based models. Though we studied other related thesis papers to modify or find out a new most efficient way for MCC and offloading in cloudlet to minimize real response time.

However, challenges of MCC are mainly our motivating points. Because, for increasing number of mobile users, security and privacy are a big deal to represent the best reliability of MCC to the mobile users and network service providers. Even providing best cloud service with reduced bandwidth cost, latency cost, cloud service cost, mobile network cost will make a large profitable market for MCC.
Ongoing worldwide adoption of mobile devices has created novel demand for access to e-commerce, social media, and entertainment applications anywhere, at any time. This has not only increased the amount of mobile broadband traffic transported by the carrier networks but also transformed its composition. In this sense, QoS is one of the most important and demanding fact to utilize in MCC. QoS in MCC basically defines the mechanism of controlling the performance, reliability, and usability of cloud computing service for users. Since QoS is judged by customers, that's why customers basic and mostly needed requirements get priority to think about most efficient system models.

On the other hand, modern technologies and competitive professions are making peoples busy day by day. Nowadays MCC is mostly concern on the extent the service during users' random mobility due to higher bandwidth and network service cost. There are various research works performed and performing on this issue to optimize the necessary costs relevant with MCC within the small response time.

2.2 Related Work

To enlarge and the concept of *MCC*, various thesis works propose their own view of assumed system models or frameworks or algorithms. Here are some listed below:

2.2.1MAUI

MAUI [*4*] enables fine-grained energy-aware offload of mobile code to the infrastructure. It decides at runtime which methods should be remotely executed. Each time a method is invoked and a remote server is available, MAUI uses its optimization framework to decide whether the method should be offloaded. Once an offloaded method terminates, MAUI gathers profiling information that is used to predict whether future invocations should be offloaded. If any disconnection occurs, MAUI resumes running the method on the local Smartphone. Additionally, MAUI continuously measures the network connectivity to the infrastructure, estimating its bandwidth and latency. When the Smartphone loses contact with the server while that server is executing a remote method, MAUI returns control back to the local proxy.

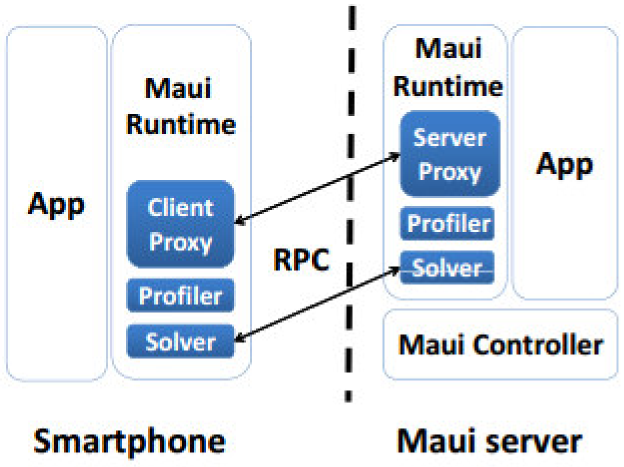

Figure 2: High-level view of MAUI's architecture

2.2.2 ThinkAir

ThinkAir [*2*] exploits the concept of Smartphone virtualization in the cloud and provides method-level computation offloading. It focuses on the elasticity and scalability of the cloud and enhances the power of mobile cloud computing by parallelizing method execution using multiple virtual machine (*VM*) images. It provides a simple library that coupled with the compiler support, makes the programmer's job very straight forward. The compiler comes in two parts: the remote able code generator and the customized native development kit (*NDK*).

However, the ThinkAir application server manages the cloud side of offloaded code and is deliberately kept lightweight so that it can be easily replicated. Actually ThinkAir assumes a trustworthy cloud server execution environment: when a method is offloaded to the cloud, the code and state data are not maliciously modified or stolen. It also assumes that the remote server faithfully loads and executes any code received from clients.

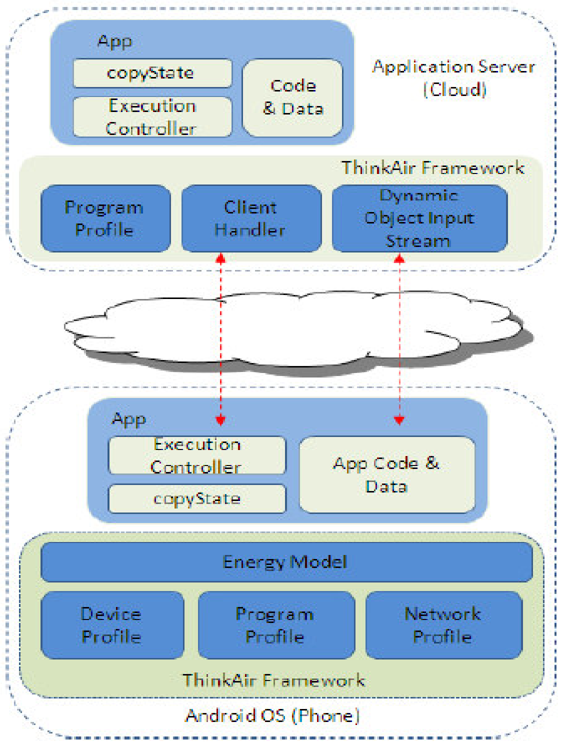

Figure 3: Overview of the ThinkAir framework

2.2.3 ENDA

ENDA [*3*] consider 3−tier architecture: Smartphone, cloudlet and cloud to select the most energy efficient network for application offloading based on user track prediction and different environmental profiling factors. According to its system architecture, Smartphone is required to communicate with clouds for application information and geographical locations. ENDA shifts the profiling section from Smartphone to cloudlets to save energy of Smartphone.

On the other hand, ENDA uses database servers on clouds to collect user traces and implements aggregation algorithms to compress them to useful routes. When Smartphone initiates an offloading request, ENDA attempts to find a route that matches most the Smartphone's reported location.

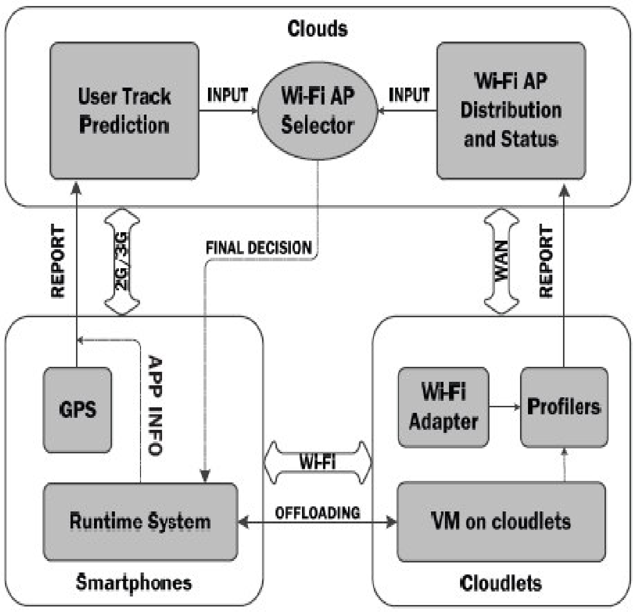

Figure 4: ENDA System Architecture

2.2.4 MuSIC

MuSIC [*7*] proposes a framework to model mobile applications as a Location-Time Workflow (*LTW*) and to develop efficient techniques for dynamic mapping of resources in the presence of mobility using tiered cloud architecture. It optimally partition the execution of the LTW in the two tier architecture based on a utility metric that combines service price, power consumption and delay of the mobile applications. Basically MuSIC algorithm is a greedy heuristic that generates a near optimal solution to the tiered cloud resource allocation problem using a simulated annealing based approach, which typically starts out with an initial solution in the potential solution space and iteratively refines this to generate increasingly improved solutions. It uses a randomized approach to increase the diversity of service selection.

2.3 Discussion on the uniqueness of the Thesis

During our research study, we noticed some limitations in some literature, which made us more careful to think about our ideas. In ThinkAIR [*2*], latency by starting, resuming and synchronizing among the virtual machines (*VMs*) is not well introduced. In ENDA [*3*], cloud service has considered with the constant movement of the user with a predicted path, which is not fully mobility aware mechanism. In MAUI [*4*], their design is not enough to handle multithreaded applications. Its applicability is also doubtful since it has a high dependency on fast Wi-Fi network. CloneCloud [*5*] system does not evaluate the negative effects of an overloaded server. Slow response time would have a huge impact on the effectiveness of this type of system. CloneCloud [*5*] uses a combination of static analysis and dynamic profiling to optimally and automatically partition an application; so that it can migrate, executes in the cloud, and reintegrates computation in a fine-grained manner that makes efficient use of resources.

Comparing with these issues, our assumed system model has outperformed during our test bed experiment. Our system model considers lower bandwidth and lowers latency during the offloading process because our system model is not only cloud dependent model but also cloudlets, where cloudlets are considered with every base station. Since the distance between the user and middle tier architecture cloudlet is not so far as like as distance between cloud and user, our system model provides little response time and fast reliable computation. It also ensures the best mechanism not to lose the data during offloading if any inconsistency shows.

2.4 Conclusion

MCC is a large communication based and technological field to handle large databases and dynamic applications offloading mechanism at anywhere anytime. Besides that, uses of less bandwidth and latency will improve to extend the servicing power of network providers. Our system model will ensure less bandwidth and less latency cost requirement during offloading in cloud and cloudlet. Since our model is considered for any kind and sizes of applications, it will handle the offloading process for each individual application with more scalable way. We are hopeful that our assumed system model will create a unique footprint in *MCC* techniques.

Chapter 3: System Model and Assumptions

3.1 Introduction

In the design of our assumed system model, we have considered cloudlets with every base station to ensure a better service for the users. It will reduce bandwidth cost, latency cost and other necessary network servicing cost. Moreover, this consideration will be very helpful if any network inconsistency (server down, adverse weather, under maintenance and so on) shows without unnerving users to continue their natural movement or ride on any vehicle.

3.2 Objectives of Our System Model

We have reflected our assumption through four key design objectives.

3.2.1 Rapidly Dynamic Adaptation in Changing Environment

Since we have given main priority on user mobility during offloading in cloud and cloudlet, so due to user random mobility, chances are very high to changing network *AP* rapidly. Besides that, in inconsistent network or in emergency case, our system in cloud will face the whole computing process to meet with user satisfaction without making interruption in their tasks offloading. That’s why, our proposed system model have to adapt quickly and efficiently with rapidly changed environment by avoiding interference during offloading in the cloud and/or the cloudlet.

3.2.2 Fast Execution Controlling Power

Some applications do not need to offload in the cloud and/or cloudlet due to their small sizes and little computational power requirement. These types of applications will be managed by the runtime system in Smartphone. And, which tasks are actually needed to offload in cloud or cloudlet that will decide our framework quickly by checking network *AP* with some mandatory parameters. It will save time as well as will earn the user satisfaction.

3.2.3 Low Cost of Bandwidth and Latency

In our consideration since all cloudlets are situated with every base stations, there are high chances to keep user under consistent network. Moreover, when any user will get near most cloudlets services, it will minimize bandwidth and latency cost for offloading the tasks. As a result, network providers also can serve more users easily.

3.2.4 Reliability and Availability

According to business view, maintaining reliability and availability in any service for user attracts more users to dependent on that service. When user will feel free and comfort using *MCC* techniques in their daily lives, it will open the door of many innovative ideas to enlarge the *MCC* field.

3.3 System Model Architecture

Here is our high level view of assumed system model architecture where clouds, cloudlets and users mobile devices will carry different collaborative functionalities to offload the tasks. We are not bothering about what types of wireless network exist among them, because our framework will smartly handle all kind of offloading process through any types of wireless network. However, the efficiency and speed of our system will be more on the basis of strong wireless network.

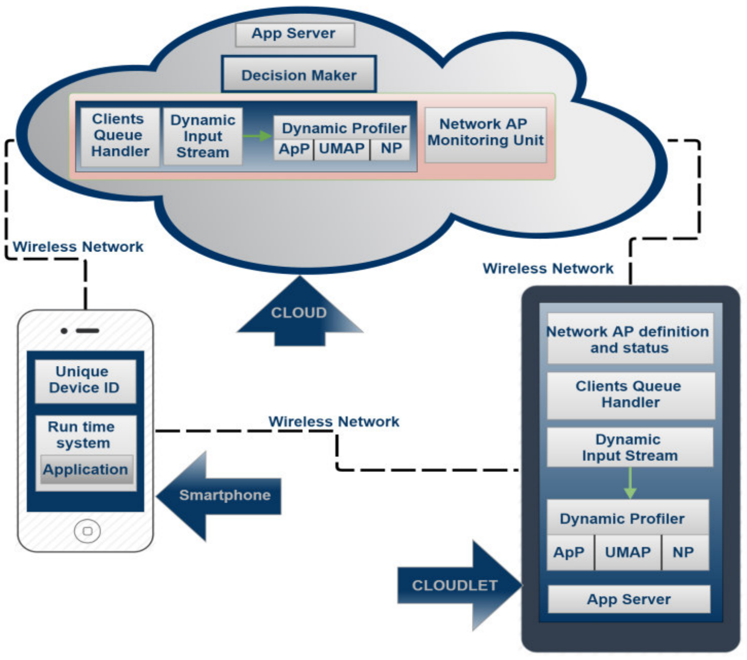

Figure 5: Overview of Our Assumed System Model Architecture

3.3.1 Runtime system in mobile device

This system is considered on every Smartphone. It will make a connection with clouds and cloudlets to pass its user request for any application offloading with the user mobile device id (DI). It's another important function is, filtering the applications which have no need to offload in clouds or cloudlets. Because all Smartphone applications do not take high computational power and energy in the mobile devices.

3.3.2 Decision Maker(DM)

It will maintain clients' requests; dynamic input stream (IS) coming from clients' mobile devices, dynamic environmental profiling and keep monitoring the network AP between clients and cloudlets. It controls and executes the entire offloading process with three basic parts:

3.3.2.1 Clients Queue Handler

When offloading requests come from clients to the cloud, it will maintain the tasks requests in a queue according to first-come-first-serve (*FCFS*) method. Same clients queue handler with the same approach will also work in every cloudlet. It will help clouds and cloudlets to identify each task individually.

3.3.2.2 Profiling with Dynamic Input Stream

Dynamic *IS* coming from clients every requested task will be stored in dynamic profiling section in a cloud database. Depending on the profilers' information, offloading process will perform. We have considered three profilers for our system which will hold all of the related information of user request.

Application Profiler (*ApP*)

It will select and keep application information. It will also mark which parts within application codes can be offloaded.

Users Mobility Analyzing Profiler (*UMAP*)

Starring from taking a request from the client till sending back the offloaded tasks to them, they may move frequently to here and there under or outside of the cloudlets range. So, wherever their position is, from the first state to the last state, it will keep track of users.

Network Profiler (*NP*)

This will check availability and strength of signal power during the offloading process in both stable position and mobility of user. During user mobility, new network *AP* can appear which information will also store.

3.3.2.3 Network *AP* Monitoring Unit

To distribute the tasks of requests in any cloudlet, the preferable cloudlet will be ensured by this unit. It will pass information to cloudlet with necessary computational instructions. If network inconsistency shows, without losing the offloaded and incomplete portions of tasks, this unit will collaborate *DM* to manage rest of the computational operation.

3.3.3 Network *AP* Definition and Status

The selected cloudlet is either capable to offload or not, will be confirmed to the cloud by this unit. Even when the selected cloudlet is not capable of offloading, it will pass user request with their DI to the next *MECL* among the list of cloudlets given by cloud. Moreover, area coverage of base station network may not same for every cloudlet. So, all basic information of base station ranges, cloudlets storage size and others will be put in this unit.

3.4 Conclusion

Our system model architecture is designed with all necessary functionalities. All of them will cooperate with each other to perform the offloading process efficiently. They will give less chance to lose any application data during offloading. So, users have no worries about their data privacy. On the other hand, cloud providers can use our system to implement practically without any hesitation. Because our system will require low cost of network service.

Chapter 4: Framework Activities

4.1 Introduction

In previous chapter we have shown overview of our system architecture. In this chapter we will describe how our system architecture will work in entire offloading process in clouds and cloudlets. We enlisted in this chapter two algorithms which will show the mechanism of selecting most efficient cloudlet/s for user and the main controlling system of cloudlet.

4.2 Computational Flow

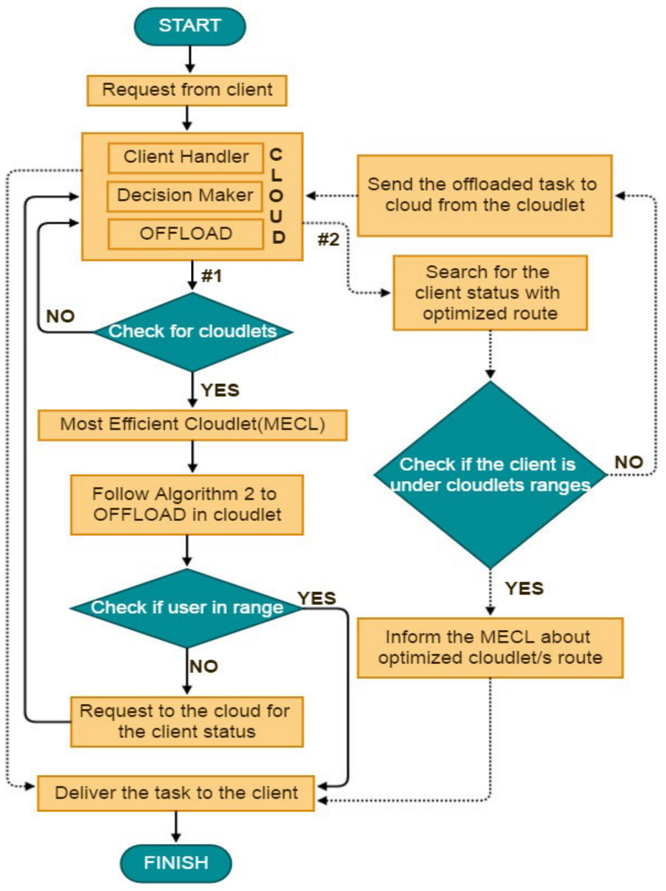
START
Request from client
Client Handler
Decision Maker
OFFLOAD
CLOUD
#1
#2
Send the offloaded task to cloud from the cloudlet
Search for the client status with optimized route
NO
Check for cloudlets
YES
Most Efficient Cloudlet(MECL)
Follow Algorithm 2 to OFFLOAD in cloudlet
Check if the client is under cloudlets ranges
NO
YES
Check if user in range
YES
NO
Inform the MECL about optimized cloudlet/s route
Request to the cloud for the client status
Deliver the task to the client
FINISH

Figure 6: Flowchart of Network AP Selection and Offloading

Step-1: According to Figure 6, initially cloud server will receive a user request (*UR*) which include users' mobile device id (*DI*) and location status. Through *DI* and location status, cloud server will know the users' current approximate location, which will help cloud server to choose the nearest as well as *MECL* for offloading.

Step-2: Secondly, according to *Algorithm 1*, the cloud will check whether any cloudlets exist or not nearby the user.

If there is no cloudlet exists nearby the user, the cloud will take the offloading request as self-responsibility. After completing offload, the cloud will directly send the task to the user.

But if cloudlet/s exists, *Algorithm 1* will choose *MECL* among them and will sort a list of next most efficient cloudlets (*Li[n]*) for the user. We have considered three parameters to choose *MECL* and the list *Li[n]*, which are: bandwidth (*BW*), latency (*LY*) and the distance between user and cloudlet (*DUC*). This two information (*MECL* and *Li[n]*) with *UR* and their *DI* will be sent by *Algorithm1* to the first cloudlet *CL(i=1)*, where i=1,2,3,...,n.

Step-3: Now, *Algorithm 2* will check the chosen cloudlet status [*CL(i=1)*], either it is busy or free to offload.

If *CL(i=1)* is free, it will start to offload. Then it will check the **step-4**.

But if *CL(i=1)* is busy, after how much time *(WT)*, *CL(i=1)* will be free to take the requested task to offload, will be calculated. Then *WT* will be compared with $X\lambda$, where $X\lambda$ is a constant time given by the cloud.

If *WT* is less or equal to $X\lambda$, until *WT* is not equal to zero (the time, when the cloudlet will be free to accept the requested task in its clients queue), the task will wait for that cloudlet and will start offloading when *WT* will be zero and then follow **step-4**.

If *WT* is greater than *Xλ*, it means the cloudlet is not free enough to operate the new task. So, according to the next *MECL* list, *Li[i++]* will take place, which will be compared with *Li[n]*.

If the list has minimum a number of cloudlet in next stage, **step-3** will repeat from the first step by considering *CL[i]* as *CL[i++]*.

But if the list is already empty, *Algorithm 2* will pass the *UR* with *DI* to the cloud and then the cloud will perform the offload.

Step-4: In cloudlet, after completing offload, it will check whether the client is under that same cloudlet range or not.

If the client is still in the range of that specific cloudlet, it will deliver the task to the client and then send a confirmation message to the cloud that the cloudlet successfully sent the task/s to the client.

If the cloudlet can't find out the client in its range, it will notify the cloud and will ask to track the client. Here the tracking result has two possibilities, one is- the client is under different cloudlet range and another is- the client is not under any cloudlet range.

If the client is under any different cloudlet range, the cloud will notify the offloading completed cloudlet an optimized route to send the offloaded task to the client. And then the cloudlet delivers the task/s via the optimized route of cloudlet/s by transferring the offloaded task/s with the client's DI from one cloudlet to another cloudlet sequentially.

If the client is not under any cloudlet range, the offloading completed cloudlet will transfer the offloaded task/s with the client's *DI* to the cloud. Then the cloud will deliver the task/s to the client.

4.3 Algorithms

4.3.1 *Algorithm 1: Selecting mechanism of MECL*

INPUT: *BW: Bandwidth, LY: Latency,*

DUC: Distance between user and cloudlet, DI: Users' unique mobile device id

μ1: Value of bandwidth given by cloud provider

μ2: Value of latency given by cloud provider

μ3: Value of DUC given by cloud provider

OUTPUT: *Confirmation about existence of cloudlet/s with respect to user location, MECL,*

Next most efficient cloudlets list (Li[n])

```
 1. Search $CL$ with respect to location of $DI$
 2. if $CL$ Exist then
 3.   Find $n$ (number of Cloudlets)
 4.   for i=0 to i<$n$ do
 5.     $BW$ = Found value *  1
 6.     $LY$ = Found value *  2
 7.     $DUC$ = Found value *  3
 8.     $Z[i] = BW + LY + DUC$
 9.   end for
10.   for i=0 to i<$n-1$ do
```

```
11.     for j=0 to j <n-i-1 do
12.       if Z[i] >Z[i+1] then
13.         TEMP = Z[i]
14.         Z[i] = Z[i+1]
15.         Z[i+1] = TEMP
16.       end if
17.     end for
18.   end for
19.     else
20.   OFFLOAD in Cloud
21.     end if
```

4.3.2Algorithm 2: Main Control System in Cloudlet

INPUT: *UR: User Request, LY: Latency,*
DI: Users' unique mobile device id, Li[n]: Next most efficient cloudlets list
OUTPUT: *Completing Task*

```
1. if CL[i] free then
2.     OFFLOAD
3.     Follow step-4 (Description of computational flow)
4.     BREAK
5.   else
6.     Measure WT
7.     if WT <= Xλ then
8.       Wait till WT == 0
```

```
 9.        OFFLOAD
10.        Follow step-4 (Description of computational flow)
11.        BREAK
12.     else
13.        Li[i++]
14.        if Li[i++] <= Li[n] then
15.           CL[i] == Li[i++]
16.           Repeat from step-1
17.         else
18.           Send UR with DI to Cloud
19.           Break
20.         end if
21.     end if
22.   end if
```

4.4 Conclusion

From **step-1** to **step-4** in section 4.2, we have considered the whole process with the consistent network. But, if any network inconsistency occurs, the cloud will take the full responsibility to perform the offloading process. Moreover, for network inconsistency during offloading, completed and incomplete part of offloading will be handled by cloud instantly, which will ensure:

1. All activities in cloudlets are continuously observed by the cloud.
2. Our proposal will ensure relativity and less dependency between cloud and cloudlets.
3. Requirement of less bandwidth and less latency cost during offloading.
4. Fast real response time in full offloading processing mechanism.

Chapter 5: Experimental Evaluation

5.1 Introduction

To evaluate our assumed system model, we arranged a performance comparisons scenario with respect to ThinkAIR [*2*], ENDA [*3*] and MuSIC [*7*] systems, for optimizing resource allocation in mobile cloud computing (*MCC*) environment. A test-bed environment is implemented to find out the performance difference. We denoted our system name shortly as *Q-MAC* in the graphs.

5.2 Environmental setup

We have setup cloudlet based environment as follows:

1. 12 laptops of different models are used as cloudlets to the corresponding *APs* for the mobile devices.
2. The IEEE 802.11n *WLAN* interfaces of the laptop are configured as hotspots.
3. 6 Android Jelly Bean Samsung Galaxy S2 mobile phones and s Android KitKat based Samsung Galaxy Grand2 as clients.
4. Google App Engine served as the cloud.

The detailed device specifications are listed in following table:

	Mobile Type 1	**Mobile Type 2**	**Cloudlet Type 1**	**Cloudlet Type 2**	**Cloud**
	Samsung Galaxy SII	Samsung Galaxy Grand2	Samsung Laptop	Dell Inspiron series Laptop	Google App Engine
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	

To observe the optimal resource allocation technique and measure offloading performance according to our framework, we have chosen few random sizes of applications with the random mobility of users.

5.3 Performance metrics

We have compared the performances on the following performance metrics:

5.3.1 Percentage of Requests Executed

Within the maximum allowable time, the ratio of the number of results received and the number of tasks offloaded for remote execution is considered here.

5.3.2 Average Time for Request Execution

The difference between the instant at which an individual task start its execution and the time instant at which the result is obtained is considered as execution time. The execution time of an individual task is averaged to evaluate the total number of executions of different tasks during the experiment.

5.3.3 Standard Deviation of Cloudlet Computation Loads

The values of standard deviation define how well the load is distributed among the cloudlets.

5.4 Simulation Results

We have carried out the experiments for increasing mobility speeds of users and observed the impacts on it, ranging from 0m/sec to 10m/sec. Here, the number of APs is fixed at 6 and 5 mobile devices are used for experiments. Each mobile device has sent different type and size of applications.

5.4.1 Impacts of Average Mobility Speed of Users

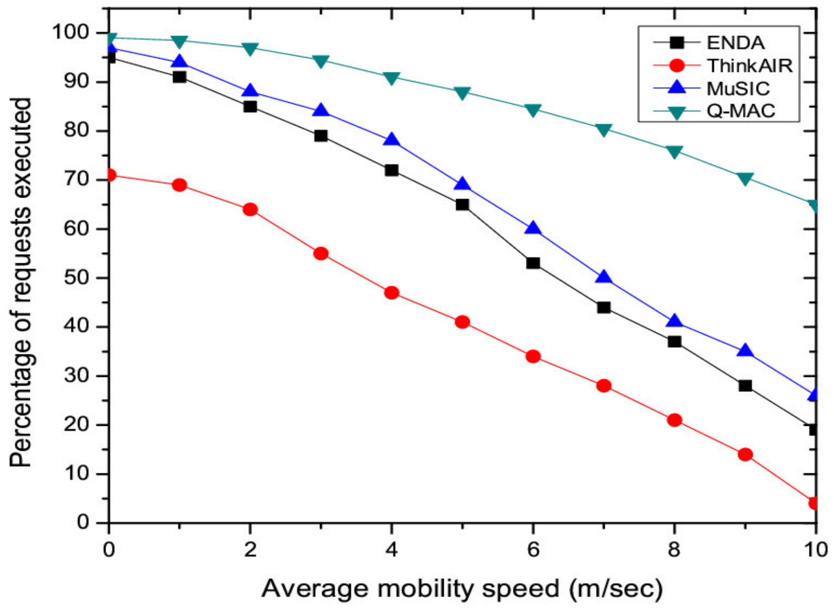

Figure 7.1(a): Impacts of average speed of mobile users (No. of mobile device=5)

Figure *7.1(a)* states that the percentage of requests executed successfully and gradually decreases for increasing mobility speeds. Our proposed *Q-MAC* system provides more stable performance compared to ThinkAIR, ENDA, and MuSIC. ThinkAIR, ENDA, and MuSIC do not exploit the mobility patterns of users. For random high-speed movements, it results in high failure rate of the submitted and assigned tasks to offload under any cloudlet for all three systems. To the contrary, *Q-MAC* uses cloudlets as intermediate servers to assure optimal resource allocation technique with its smart task execution mechanism by tracking only users' unique mobile device ID. Moreover, the cloudlets can transfer the offloaded task to each other by following optimized route, when it is necessary due to users higher mobility speed.

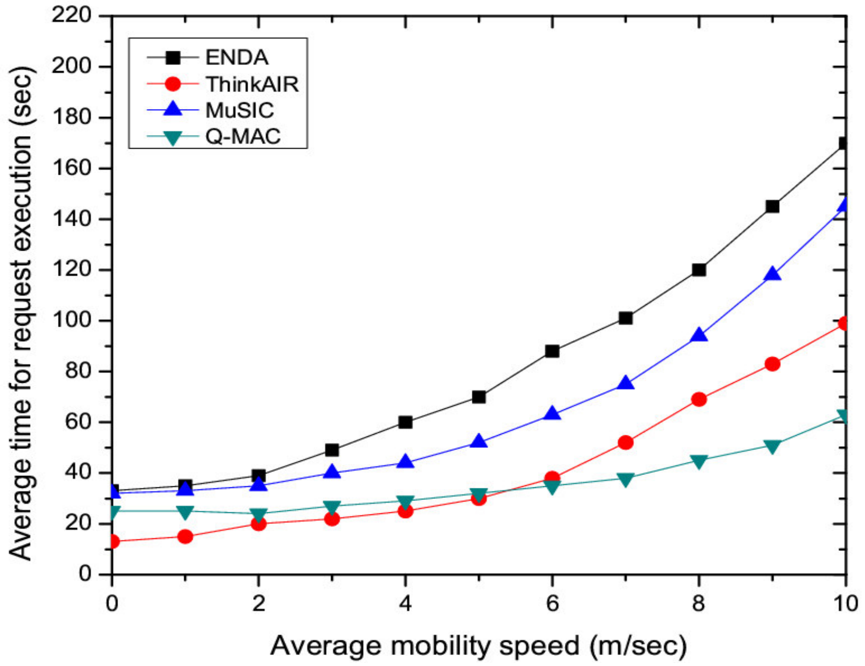

Figure 7.1(b): Impacts of average speed of mobile users (No. of mobile device=5)

Figure *7.1(b)* shows that the average execution time of offloading is increased with the mobility speed. This execution time includes the data transmission, computation, waiting delays and maintaining user requests queue. When user moves fast, the network strength changes so rapidly which increases that data transfer time increases sharply. Our mobility aware task execution system *Q-MAC* has showed reduced number of failures and retransmissions as well as reduced waiting time and thus it has achieved better delay performance compared to ThinkAIR, ENDA, and MuSIC.

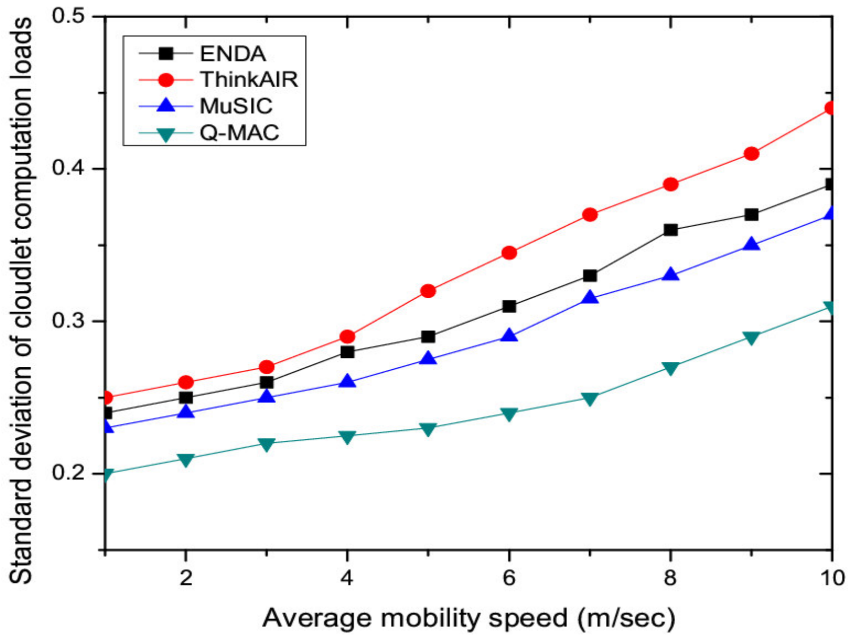

Figure 7.1(c): Impacts of average speed of mobile users (No. of mobile device=5)

Finally, Figure *7.1(c)* illustrates that with increasing mobility speed, the standard deviation of cloudlet computation loads increases significantly. Our proposed *Q-MAC* algorithm provides better performance than ThinkAIR, ENDA, and MuSIC because the workloads are well distributed among the cloudlets; and cloud always keep better care the whole execution process as well as during users random movement if user do not stable under the initial cloudlet, cloud manages the computation, execution and delivery the tasks to the user with its smart mechanism.

5.4.2 Impacts of Varying Number of Mobile Devices

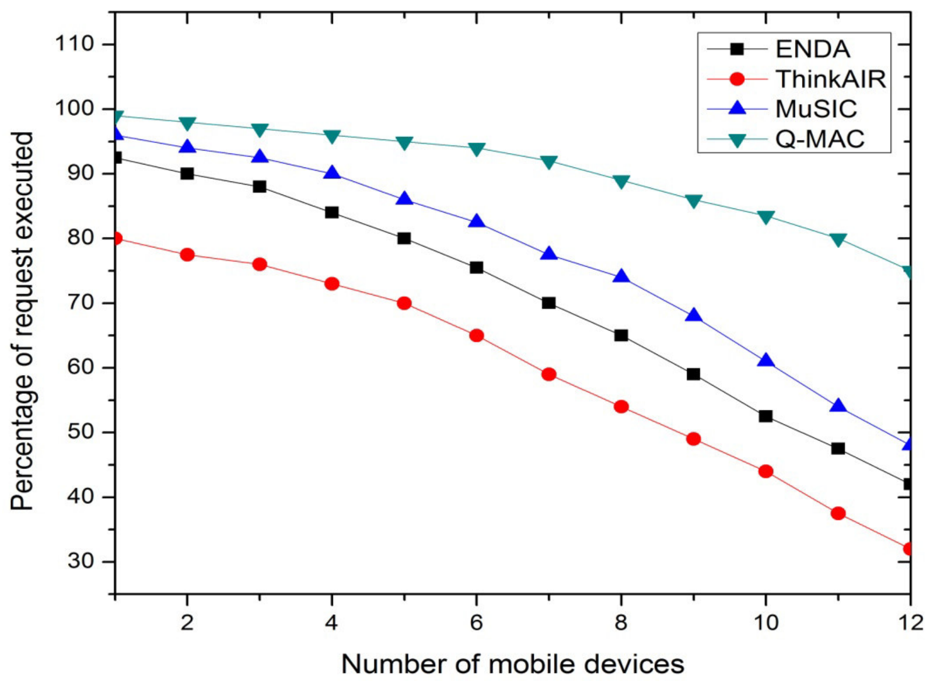

Figure 7.2(a): Impacts of varying number of mobile devices (Average mobility speed=2m/sec)

Figure *7.2(a)* shows that the percentage of request executed successfully at remote cloudlets and cloud decreases slowly with the increasing number of mobile devices. ENDA and MuSIC systems do not focus on scheduling the multiple requests together considering hardware resources and heterogeneities in application. ThinkAIR system is only cloud dependant which increases the load in cloud. In *Q-MAC*, we have placed cloudlets with each base station so that it can enhance the percentage of request execution efficiently.

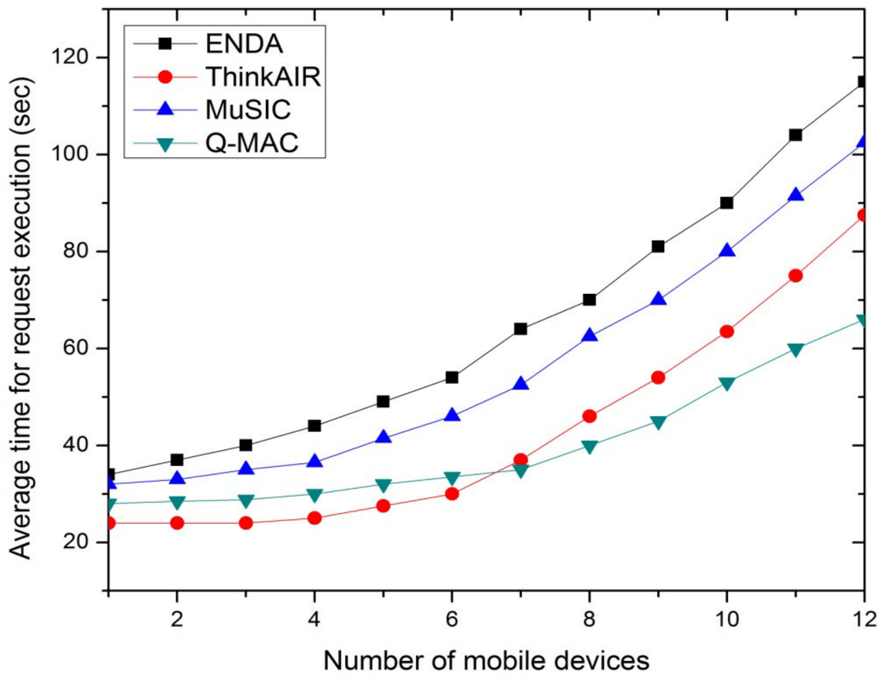

Figure 7.2(b): Impacts of varying number of mobile devices (Average mobility speed=2m/sec)

Figure *7.2(b)* shows that for less number of requests, all systems (ENDA, ThinkAIR and *Q-MAC*) take around same amount of time to perform the computation process except ThinkAIR due to its only cloud dependency, which takes much time to perform the full operation. However, in *Q-MAC* system, the estimation of user different velocity and allocating requests in different cloudlets results with less interferences.

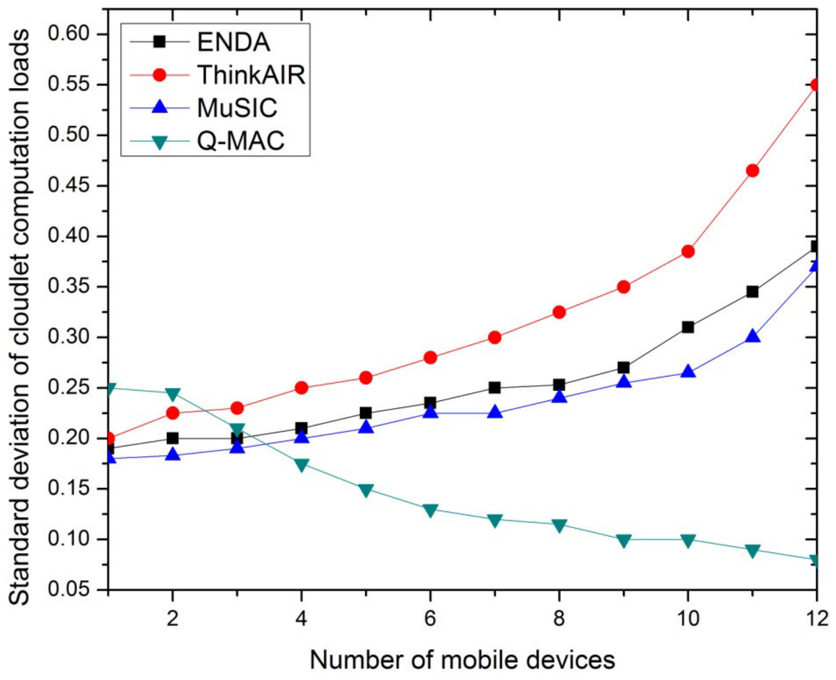

Figure 7.2(c): Impacts of varying number of mobile devices (Average mobility speed=2m/sec)

According to figure *7.2(c)*, the initial workload of *Q-MAC* is higher than other systems such as Think Air, ENDA and MuSIC. As the number of devices increase, the standard deviation of *Q-MAC* decreases due to its optimized and balanced distribution of workload. On the other hand, workload distribution is not well defined in ENDA and MuSIC systems and ThinkAIR does not include any cloudlets to distribute the workload. As a result, the standard deviation for that systems increase with the increasing number of mobile devices.

5.5 Conclusion

In our test bed experiment, comparing with ENDA, ThinkAIR, and MuSIC, *Q-MAC* has given satisfactory results. The graphs and discussions show the better efficiency of *Q-MAC* than rest of the three models. Our considered performance metrics are very good answer to compete with MCC challenges. We are very hopeful that Q-MAC will be able to retain its better efficiency in real life scenario. As a result, cloud computing providers and network providers can provide better service to the mobile users. It will increase a large reliable market for users and very profitable job sector also.

Chapter 6: Conclusion & Future Scope

6.1 Discussion

In this book, we have introduced a new framework for offloading mobile application codes to the cloud and cloudlets. In our system architecture, we have assumed cloudlet with each base station as an intermediate server under a cloud. It will adapt with rapidly changed environment due to users random mobility and velocity. Its fast execution controlling power will reduce bandwidth and latency cost. For both user and cloud providers, our system will give best reliability on users' data privacy and setting up cloudlets with every base station will increase availability of network service. The introduced algorithms will choose the best near most cloudlets for user and control fast computation in cloudlets.

However, our assumed system model gave better view comparing with famous ThinkAir, ENDA and MuSIC system models. Because, basically our system planning is simple and fast to execute many tasks altogether and most frequent to load the tasks distribution among cloudlets. After performing simulation, we are very hopeful about our thesis work that our system will also retain its' better performance in real world scenario in mobile cloud computing techniques.

6.2 Future Work

Mobile cloud computing (*MCC*) is the next big thing in the current market scenario. It will not provide benefits only to the Smartphone users but also very helpful with a broader range of mobile subscriber. With *MCC*, mobile phone user will get benefit in number of ways and help them to run their business applications without large amount of capital investment in infrastructure and services.

We know that big data is a new term of modern large and complex datasets. We are interested to work for handling big data with respect to *MCC*. Additionally, computation of offloading tasks by using multiple VMs of multiple cloudlets is very challenging, because collaborative execution among cloudlets is quite complicated. Although if this come true practically, it will reduce full computational pressure on a particular cloudlet when user will not stable under its range. Our future work will be distributing the offloading task among multiple *VMs*, in multiple cloudlets. We also aim to expand on analyzing the trade-offs between costs and benefits in real world scenario.

Bibliography

1. A survey on soil testing, “https://www.ericsson.com/news/1925907”, accessed on 02 October 2016.
2. 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”, INFOCOM, 2012 Proceedings IEEE, Orlando, FL, 2012, pp 945-953.
3. J. Li, K. Bu, X. Liu, and B. Xiao, ”ENDA: embracing networkinconsistency for dynamic application offloading in mobile cloud computing”, In Proceedings of the 8th internationalconference on Mobile systems, applications, and services (Mo-biSys ’10). ACM, New York, NY, USA, 2010, 49-62.
4. E. Cuervo, A. Balasubramanian, Dae-ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. ”MAUI: making smartphoneslast longer with code offload”, IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, 2013, pp. 75-82.
5. B. Chun, S.Ihm, P.Maniatis, M.Naik, and A. Patti, ”CloneCloud: elastic execution between themobile device and cloud”, In Proceedings of the Sixth conference on Computersystems (EuroSys ’11). ACM, New York, NY, USA, 2011, 301-314.
6. M. Sharifi, S. Kafaie and O. Kashefi, ”A Survey and Taxonomyof Cyber Foraging of Mobile Devices”, in IEEE Communications Surveys and Tutorials, vol. 14, no. 4, pp. 1232-1243, Fourth Quarter 2012.
7. M. R. Rahimi, N. Venkatasubramanian and A. V. Vasilakos, ”MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing”, IEEE Sixth

International Conference onCloud Computing, Santa Clara, CA, 2013, pp. 75-82.

8. F. Berg, F. Drr, and K. Rothermel, ”Optimal predictive code offloading”, In Proceedings of the 11th International Conferenceon Mobile and Ubiquitous Systems: Computing, Networking, and Services (MOBIQUITOUS ’14), ICST, Brussels, Belgium, 2014, 1-10.
9. Khan, Atta urRehman; Othman, Mazliza; Ali, Mazhar; Khan, Abdul Nasir; Madani, Sajjad Ahmad (2013-12-01). "Pirax: a framework for application piracy control in a mobile cloud environment". The Journal of Supercomputing. 68 (2):753–776. doi:10.1007/s11227-013-1061-1. ISSN 0920-8542.
10. Khan, A. u R.; Othman, M.; Xia, F.; Khan, A. N. (2015-05-01). "Context-Aware Mobile Cloud Computing and Its Challenges". IEEE Cloud Computing. 2 (3): 42–49. doi:10.1109/MCC.2015.62. ISSN 2325-6095
11. Khan, A. u R.; Othman, M.; Madani, S. A.; Khan, S. U. (2014-01-01). "A Survey of Mobile Cloud Computing Application Models". IEEE Communications Surveys Tutorials. 16 (1): 393–413. doi:10.1109/SURV.2013.062613.00160. ISSN 1553-877X
12. Abolfazli, Saeid; Sanaei, Zohreh; Ahmed, Ejaz; Gani, Abdullah; Buyya, Rajkumar (1 July 2013). "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges". IEEE Communications Surveys & Tutorials. 99 (pp): 1–32.doi:10.1109/SURV.2013.070813.00285
13. Fangming Liu, PengShu, Hai Jin, Linjie Ding, Jie Yu, Di Niu, Bo Li, "Gearing Resource-Poor Mobile Devices with Powerful Clouds: Architecture, Challenges and Applications";,IEEE Wireless Communications Magazine, Special Issue on Mobile Cloud Computing, vol. 20, no. 3, pp.14-22, June, 2013.
14. Sanaei, Zohreh; Abolfazli, Saeid; Gani, Abdullah; Buyya, Rajkumar (1 January 2013). "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open

Challenges" (PDF). IEEE Communications Surveys & Tutorials (99): 1–24. doi:10.1109/SURV.2013.050113.00090porn

15. M. Gordon, D. Jamshidi, S. Mahlke, Z. Mao, and X. Chen. Comet: Code offload by migrating execution transparently. In Proc. of USENIX OSDI, 2012.
16. P. Shankar, B. Nath, L. Iftode, W. Huang, and P. Castro. Crowds replace experts: Building better location-based services using mobile social network interactions. In Proc. of IEEE PerCom, 2012.
17. Marco Valerio, SokolKosta,AlessandroMei,JulindaStefa: " To Offload or Not to offload? The Bandwidth and Energy Costs for Mobile Cloud Computing", In IEEE INFOCOM 2013.
18. M. Reza. Rahimi, NaliniVenkatasubramanian, SharadMehrotra and AthanasiosVasilakos, "MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture", In IEEE/ACM UCC 2012.
19. G. H. Canepa, 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 (MCS '10), New York, 2010.
20. K. Kumar and Y. H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?," IEEE Computer, vol. 43, no. 4, pp. 51–56, 2010.
21. A. Agarwal, D. Gupta, "Security Requirements Elicitation Using View Points for Online System", First International Conference on Emerging Trends in Engineering and Technology, pp.1238-1243, July 2008.
22. M. D. de Assunc, A. di Costanzo, and R. Buyya, "A cost-benefit analysis of using Cloud computing to extend the capacity of clusters," in Cluster Computing,vol. 13, no. 3, pp. 335-347, Sep. 2010.
23. Q. Zhang, E. Gurses, R. Boutaba and J. Xiao, "Dynamic resource allocation for spot markets in Clouds," in Proc of the 2nd Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), Mar. 2011.

24. A. Andrzejak, D. Kondo, and S. Yi, “Decision model for Cloud computing under SLA constraints”, Proc. of the IEEE Int’l Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS) Aug. 2010, pp. 257-266.
25. He W, Yan G, Xu LD (2014) Developing vehicular data cloud ser- vices in the IoT environment. IEEE Trans Indus Inf 10(2):1587– 1595
26. Son J, Eun H, Oh H, Kim S, Hussain R (2012) Rethinking vehic- ular communications: merging VANET with cloud computing. In: IEEE Int’l conf. on cloud computing (CLOUDCOM), pp 606–609
27. Smith A (2013) Smartphone ownership–2013 update. Pew Research Center, Washington DC
28. Verbelen T, Simoens P, De Turck F, Dhoedt B (2012) Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the third ACM workshop on Mobile cloud computing and services, MCS ’12. ACM, New York, pp 29–36
29. Conti M, Mascitti D, Passarella A (2015) Offloading service provisioning on mobile devices in mobile cloud computing envi- ronments. In: Euro-Par 2015: parallel processing workshops, p 2015
30. Shivarudrappa D, Chen ML, Bharadwaj S (2012) Cofa: auto- matic and dynamic code offload for android. Technical report, University of Colorado, Boulder

Appendix A: List of Acronyms

Acronym	Full form
MCC	Mobile Cloud Computing
CPU	Control Processing Unit
IT	Information Technology
IaaS	Infrastructure as a Service
SaaS	Software as a Service
QoS	Quality of Service
WAN	Wide Area Network
LTW	Location-Time Workflow
NDK	Native Development Kit

Appendix B: List of Notations

Acronym	Full form
UR	User Request
DM	Decision Maker
CL	Cloudlet
Li[n]	A number of next most efficient cloudlets List
i	Used to identify individual cloudlet from the list, where i=1,2,3,………,n
WT	Waiting time to get the chance in queue for offloading
Xλ	Constant value, set up by Cloud
BW	Bandwidth
LY	Latency
DUC	Distance between user and cloudlet
AP	Access Point
IS	Input Stream
ApP	Application Profiler
NP	Network Profiler
UMAP	Users Mobility Analyzing Profiler
DI	Users' unique mobile device ID