PERFORMANCE ENHANCEMENT FRAMEWORK FOR CLOUDLET IN MOBILE CLOUD COMPUTING

MD WHAIDUZZAMAN

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

PERFORMANCE ENHANCEMENT FRAMEWORK FOR CLOUDLET IN MOBILE CLOUD COMPUTING

MD WHAIDUZZAMAN

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND
INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Md Whaiduzzaman

Registration/Matrix No.: WHA110054

Name of Degree: Doctor of Philosophy

Performance Enhancement Framework for Cloudlet in Mobile Cloud Computing

Field of Study: Computer Science

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

ABSTRACT

A tremendous increase in the use of mobile devices, such as smart phones and tablets, has been observed in recent years. Mobile device resources, such as CPU, memory and storage resources, have also experienced dramatic increases in capacity. Simultaneously, a rich variety of mobile applications that require extensive computational resources for mobile application execution have been developed. These new applications are complemented with cloud resources through the emerging Mobile Cloud Computing (MCC) paradigm. MCC augments mobile device capabilities by leveraging the resources of distant clouds, nearby cloudlets or mobile ad-hoc clouds in the vicinity. However, practical utilization of MCC is hindered by limitations associated with network connectivity. In the case of distant clouds, jitter, bandwidth, and propagation delay pose a challenge to real-time application response. On the other hand, cloudlets and mobile ad-hoc clouds are not sufficiently resource-rich to be able to support rich mobile applications. In this research, we consider a mobile user in the vicinity of a cloudlet that is situated at a distance of one hop through a Wi-Fi communication medium. We show that a substantial number of users accessing the cloudlet for computation-intensive tasks results in delayed task completion and ultimately diminishes the benefits of using cloudlets. The problem is referred to as the cloudlet resource scarcity problem. To alleviate this problem, various researchers have offered solutions whereby mobile device resources are used for partial task completion. However, the proposed approaches do not consider the mobile device offered-service-to-load ratio. In this research, we propose the Mobile-device-based Cloudlet Resource Enhancement (MobiCoRE) framework for mobile application augmentation to employ nearby mobile devices while ensuring the following: (i) mobile devices always obtain time benefits for their tasks when submitted to the cloudlet, and (ii) the cloudlet-induced mobile device load is a fraction of its own service requirements. We map MobiCoRE on

the M/M/c/K queue and model the system using a birth-death Markov chain. We show that for cloudlets, the framework always obtains the maximum advantage for mobile devices in terms of job completion time when the cloudlet service time is set to $\frac{\bar{c}}{\lambda}$, where $\bar{c}$ is the cloudlet utilization and λ is the application arrival rate. Furthermore, the optimal service time is independent of the application's service requirement. We implemented the MobiCoRE framework using the Openstack cloud. The evaluation shows that MobiCoRE accommodates up to 50% more users when operating at the optimal service time and provides 50% time benefits for mobile users. The empirical analysis and statistical validation demonstrate that our proposed framework, i.e., MobiCoRE, significantly and positively impacts the cloudlet performance by exploiting and orchestrating nearby mobile device resources.

ABSTRAK

Peningkatan mendadak dalam penggunaan peranti mudah alih, seperti telefon pintar dan tablet, telah diperhatikan dalam tahun-tahun kebelakangan ini. Sumber peranti mudah alih, seperti CPU, memori dan storan, juga telah meningkat secara mendadak. Pada masa yang sama, aplikasi mudah alih yang berpelbagai sedang dibangunkan dan memerlukan sumber komputer yang luas untuk pelaksanaan aplikasi mudah alih ini. Aplikasi baru ini dilengkapi dengan menggunakan sumber cloud melalui paradigma Mobile Cloud Computing (MCC) yang kian memuncul. MCC menambahkan kemampuan peranti mudah alih dengan memanfaatkan sumber-sumber cloud yang jauh, cloudlet berdekatan atau cloud mudah alih ad-hoc di sekitar berhampiran. Walau bagaimanapun, penggunaan MCC yang praktikal dihalang oleh batasan yang berkaitan dengan pencapaian rangkaian. Dalam kes cloud jauh: ketar, lebar jalur, dan kelewatan perambatan menimbulkan cabaran kepada respons masa nyata aplikasi. Sebaliknya, cloudlet dan cloud mudah alih ad-hoc tidak mempunyai sumber yang mencukupi untuk menyokong pelbagai aplikasi mudah alih. Dalam kajian ini, kami mengkaji pengguna mudah alih di kawasan berhampiran cloudlet, yang terletak pada satu jarak hop melalui medium komunikasi Wi-Fi. Kita menunjukkan bahawa sebilangan besar pengguna yang mengakses cloudlet untuk tugas-tugas komputer intensif mengakibatkan kelewatan menyiapkan tugas, dan akhirnya mengurangkan manfaat menggunakan cloudlet. Masalah ini dirujuk sebagai masalah kekurangan sumber cloudlet. Untuk mengurangkan masalah ini, penyelidik mencadangkan satu penyelesaian di mana sumber peranti mudah alih akan digunakan untuk menyempurnakan tugas separa. Walau bagaimanapun, pendekatan yang dicadangkan tidak mengambil kira nisbah perkhidmatan muatan yang diberi oleh peranti mudah alih. Dalam kajian ini, kami mencadangkan satu rangka kerja Peningkatan Sumber Cloudlet yang berdasarkan rangka Cloudlet Resource Enhancement (MobiCoRE) untuk menambahkan aplikasi mudah alih

dengan menggunakan peranti mudah alih yang berhampiran di samping memastikan bahawa: (i) peranti mudah alih sentiasa mempunyai manfaat masa untuk tugas-tugas yang diserahkan kepada cloudlet, dan (ii) muatan peranti mudah alih yang disebabkan oleh cloudlet merupakan sebahagian daripada keperluan perkhidmatan cloudlet itu. Kami memetakan MobiCoRE pada barisan M / M / c / K dan model sistem itu dengan menggunakan rantaian kelahiran mati Markov. Kami menunjukkan bahawa cloudlet sentiasa mengakibatkan kelebihan maksimum bagi peranti mudah alih di segi masa tamat tugas apabila masa perkhidmatan cloudlet ditetapkan untuk c / λ di mana c adalah penggunaan cloudlet dan λ ialah kadar pencapaian aplikasi. Tambahan pula, masa perkhidmatan yang optimum adalah tidak bergantung kepada keperluan perkhidmatan aplikasi. Kami melaksanakan rangka kerja MobicoRE menggunakan Openstack cloud. Penilaian menunjukkan bahawa MobiCoRE menampung sehingga 50% pengguna tambahan semasa mengendalikan perkhidmatan optimum dan menyediakan 50% manfaat masa kepada pengguna mudah alih. Analisis empirikal dan pengesahan statistik menunjukkan bahawa rangka kerja yang dicadangkan, MobiCoRE, jelas memberi impak positif kepada prestasi cloudlet dengan mengeksploitasikan dan mengatur sumber peranti mudah alih berhampiran.

ACKNOWLEDGEMENTS

First and foremost, I want to express my gratitude and thanks to the Almighty, the creator of the universe, for enabling me with the strength, wisdom, and endless blessings to help me to go through the most challenging journey in my life.

I sincerely express my deepest and sincere gratitude to my supervisor, Professor Dr. Abdullah Gani, Dean of FCSIT, for his endless guidance, continuous support throughout my PhD study. His wisdom, motivation and encouragement have made me both a good person and a confident researcher. He helped me to see the research as well as the life in to a new perspective. His consistent support and invaluable guidance enabled my research into a good piece of work.

My hearty thanks must go to Dr. Anjum Naveed, Senior Lecturer, Faculty of Computer Science & Information Technology, UM, who has provided the vision, encouragement and technical guidance to proceed through this doctoral program and complete my thesis.

I would like to express my special appreciation and thanks to my wife, for her support, encouragement and sacrifice. Special thanks to my brothers and sisters, parent-in-law who sacrificed their joyful companion and accepted the distance and gap of my PhD pursuing time. I would also like to express my deep appreciation to my MCC lab mates who always encouraged me in my tough time.

Finally, I would like to thank the financial support and assistance of High Impact Research (HIR) grant by the Ministry of Higher Education, Malaysia. My sincere thanks go to the honorable Vice Chancellor of Jahangirnagar University and the JU administration to provide me the opportunity to complete my PhD.

DEDICATION

TO MY LATE PARENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1 : INTRODUCTION

Technological advancements in the domains of computing and communication technologies have significantly influenced all areas of life. The ubiquitous nature of computers has resulted in new applications of computing. Mobile cloud computing is a computational paradigm that has quickly garnered widespread popularity in the computing era and is expected to be an important technology in the future.

This chapter presents a holistic view of the research performed in this thesis. We present the motivation and the theoretical background for the research on cloudlets in Mobile Cloud Computing (MCC). The chapter discusses the problem statement, states the objectives and describes the methodology used for the proposed research. Moreover, the chapter specifies the objectives of this study and describes the methodology proposed to achieve said objectives. The chapter is divided into six sections.

The remainder of the chapter is organized as follows. Section 1.1 presents a brief overview of and background on mobile cloud computing and an introduction to cloudlets and their characteristics as they pertain to this research. Section 1.2 is dedicated to the motivation of this research and highlights the significance of the work by explaining the importance of the proposed work. The statement of the problem addressed in this thesis is presented in Section 1.3. This is followed by a statement of the objective and the proposed methodology for addressing the research problem in Sections 1.4 and 1.5, respectively. Section 1.6 describes the layout of the thesis.

1.1 Background

The introduction of cloud computing significantly changed the way computational paradigms and technologies work. Cloud computing aggregates the resources of smaller computational units to provide computing as a service. Instead of purchasing computers

that remain unused most of the time and that are under-utilized the remaining period of time, computation time is purchased as necessary. In other words, computing can be considered as similar to other utilities such as electricity, gas and water. This model has significant advantages. The utilization of computers can be significantly improved, resulting in lower costs to customers. Furthermore, the operation and maintenance costs of the hardware are reduced, and the end customer need not worry about these aspects. In short, the primary advantage of cloud computing is its ability to provide hassle-free and cost-effective high-end computing and storage resources.

When cloud resources are accessed using battery-powered, resource-limited and mobility-capable mobile devices such as smartphones, PDAs, tablet PCs and laptops, the computational paradigm is referred to as mobile cloud computing. Mobile cloud computing faces additional challenges because of highly dynamic wireless network conditions and low-resource mobile devices that are subject to battery power restrictions. Therefore, in addition to providing high-end computing and storage resources, mobile cloud computing must also consider the limited battery power of mobile devices. The additional advantage that mobile cloud computing must offer is the preservation of the battery power of the mobile devices. Figure 1.1 shows a typical access scenario in mobile cloud computing. With the introduction of mobile cloud computing, researchers have proposed additional scenarios for computing, some of which overcome the limitations of the original form of mobile cloud computing. The two most prominent scenarios are mobile ad hoc clouds and cloudlets.

Mobile ad hoc clouds are formed by the sharing of the resources of mobile devices amongst themselves in the absence of access to a large-scale cloud. Although the resources of individual mobile devices are limited (Fernando, Loke, & Rahayu, 2013), the aggregate of small amounts of resources becomes a quite powerful set of resources if properly managed. Such clouds/cloudlets are expected to be used in social computing in

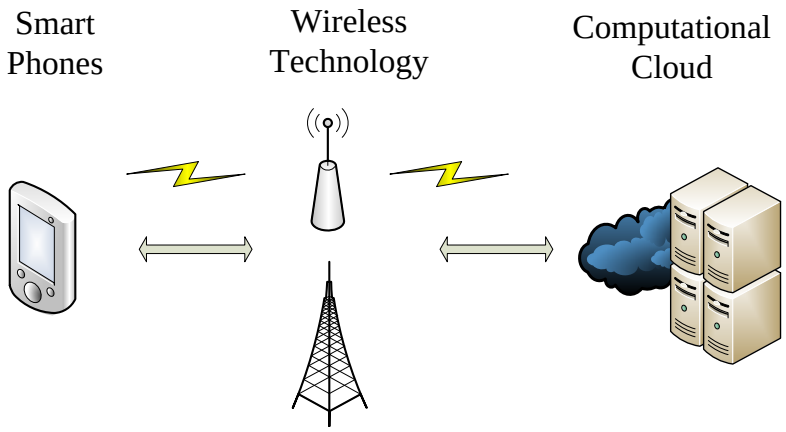

Figure 1.1: Mobile Cloud Computing Components

the future. Another form of mobile cloud computing is cloudlets. This thesis is based on cloudlets; therefore, we highlight the salients of the cloudlets in a subsequent section.

Cloudlets are small-scale clouds located in close proximity to mobile users. Typically, a mobile user can access the cloudlet through one-hop wireless access. In the future, cloudlets are expected to be deployed along with hotspots in locations such as offices, clinics and cafes. Cloudlets are installed on discoverable, localized, stateless servers running one or more virtual machines (VMs) on which mobile devices can augment resource-intensive mobile applications (Satyanarayanan, Bahl, Caceres, & Davies, 2009).

Cloudlets offer several advantages over distant clouds for mobile users. Mobile users obtain instant and direct access to the cloudlet which eliminates several drawbacks such as communication latency, jitter, and slow data transfer. In addition, mobile device can avoid resource starvation and cloudlets ensure faster application execution times compared to the mobile device results and provides a significantly enhanced user experience. Finally, because the cloudlets are in vicinity to the mobile user, which can save money by avoiding expensive data transfers.

A number of useful applications that can use cloudlet technology have been proposed

by researchers. The diversity in potential applications indicates that cloudlets will see significant deployment in the near future and be used as an enabling technology in the future.

1.2 Motivation

The recent explosive growth of mobile and distributed computing, wireless networking, and IOT (Internet Of Things) has changed ubiquitous computing for mobile users. Mobile devices such as smartphones, tablets, and Personal Digital Assistants (PDAs) have become an integrated aspect of technologically advanced lifestyles. Among them, smartphones represent a one-upmanship due to their miniature nature and interesting features, including voice and video calling, perception, multimedia, and geolocation services (Albanesius, 2011). According to Gartner, smartphones have been leading the mobile device market with a greater than 55% overall sales share in the last quarter of 2013 (Gartner, 2013). A report by Juniper Research stated that the consumer and enterprise markets for cloud-based mobile applications are expected to increase by $9.5 billion by 2014, which demonstrates the increasing mobile computing use trends.

It is evident that smartphones have become an enabling technology to serve mankind in several critical areas, particularly healthcare, education, tele-monitoring, urban management, and disaster recovery. However, smartphone capabilities are encumbered by their intrinsic limitations, particularly constraints on the battery, Central Processing Unit (CPU), and memory resources. Therefore, smartphone resource augmentation through clouds and cloudlets is as necessary a technology as smartphones themselves. The ubiquitous nature of smartphones and the mobile user expectations of being able to perform any task at any time and place require computational augmentation in close proximity to the smartphones.

A number of useful applications that can use cloudlet technology have been pro-

posed by various researchers. The most promising application proposed is the cognitive augmentation of physically disabled people by connecting their body-worn computational units to nearby cloudlets for computational offloading as they move around. Researchers have proposed context awareness, sign language translation and online dictionaries as potential applications that represent important needs in the future. Similarly, the application of cloudlets in the domain of medical emergency response, healthcare and social interaction has also been proposed. Finally, social interaction and multiplayer interactive gaming have also been proposed as potential application areas for cloudlets.

Satyanarayanan (Satyanarayanan et al., 2009) first introduced the cloudlet concept and described it as a "data center in a box". Cloudlets are simple to deploy and manage, thereby making them feasible to deploy on a business premises such as a coffee shop or a doctor's office. These same features of cloudlets make them ideal for large-scale deployment in conjunction with WiFi hotspots. Such a deployment can enable a vast range of applications, as previously discussed in this chapter. All these applications are rich applications and require intensive computations that are generally beyond the capabilities of a single mobile device.

The combination of the following facts necessitates research on addressing the issues related to cloudlets.

- Mobile devices are resource-limited. The small and light-weight design of mobile devices constrains the addition of extensive resources into mobile devices.

- Mobile device users expect their devices to be able to perform any computational task and execute any mobile application irrespective of their location and time involved. This ubiquitous nature of demand requires resource augmentation from close proximity.

- The expected mobile applications are resource hungry, therein requiring extensive

computational capacities that are not available to the mobile device itself. This necessitates mobile device resource augmentation, possibly from computational resources in close proximity that are faster and easy to access at low cost.

- The best and closest source of resource augmentation for mobile devices is cloudlets that are generally one hop away, do not experience the network issues suffered by WANs and can offer a reasonable amount of resources at minimal or no cost.

This thesis represents an effort to address one of the most critical research problems in the domain of mobile cloud computing and is directly related to cloudlets. It is expected that the proposed research will bring cloudlets one step closer to wide-scale deployment in urban areas.

1.3 Statement of Problem

Despite the significant advantages of cloudlets and potential use cases, cloudlets have certain limitations. Such limitations also represent research challenges that must be addressed before cloudlets can be widely deployed. A majority of cloudlets are expected to be deployed as free resources. In such situations, anyone can offload any piece of code onto the cloudlets to enhance the computational capacity of their mobile devices.

The cloudlet resources are shared with no significant bounds on the level of load that a mobile user can offload onto the cloudlet. This can lead to the problem of fairness and starvation. Certain users can occupy the cloudlet for extended periods of time, thereby resulting in poor performance for the remaining users. This can also lead to starvation and denial of service for the users. Furthermore, proper admission control is required to ensure the service granted for the users, which is not the default case with cloudlets.

One of the major limitations and research challenges concerns the problem of limited resources in the cloudlet. The limited and finite resources must be shared amongst multiple users. In general, the computational resources are shared using time sharing policies

with no priority. Consequently, if the number of users sharing the cloudlet resources increases, the per user time share of the computational resources decreases, and the wait time between successive computation cycles increases.

In addition, under substantial workloads, the wait time between consecutive computations can be significantly high. The effective computational capacity offered by the cloudlet to individual users can be comparable or even less than the computational capacity of the users' mobile devices. In other words, the effective execution time on the cloudlet for mobile applications can be higher than the execution time when the application is executed on the mobile device. Given that cloudlets offer free services, therefore, the limited resources of the cloudlets can significantly degrade the user experience and affect the offered service. The primary objective of improved computation time for which the cloudlets are used is not achieved when subject to substantial numbers of workloads unless an alternate arrangement is applied.

We refer to this problem as the problem of resource scarcity under increased resource utilization of the cloudlets, which affects the ability of the cloudlet to efficiently complete the user application tasks. Under such circumstances, the mobile users are unable to obtain the desired benefit of faster computations from the cloudlets. In this thesis, we analyse the problem of resource scarcity with reference to the user service experience with the cloudlet.

This issue brings us to the statement of the following research problem:

Cloudlet resource utilization, particularly CPU utilization, increases proportionally with increasing number of computation-intensive mobile application tasks augmented by the cloudlet. The time sharing mechanism of the cloudlet produces an increased wait time between computation cycles for each application. Consequently, such extensive resource utilization leads to the resource scarcity problem in the cloudlet. The limited resources of the cloudlet negatively impact the cloudlet response time, user service time, and overall cloudlet efficiency and leads to degraded cloudlet performance.

1.4 Statement of Objectives

This research is undertaken with the aim to address the problem of resource scarcity under heavy loads in cloudlets with finite resources using nearby mobile device resources. The aim is achieved through the following set of objectives:

- To study and identify recent cloudlet-based mobile application augmentation approaches and gain an insight into offloading in MCC that helps us to identify the current problems in computation-intensive mobile application augmentation in MCC.
- To investigate and analyse the impact of limited and finite resources on cloudlet performance in cloudlet-based mobile application augmentation.
- To propose a performance enhancement framework for cloudlets in MCC for cloudlet-based computation-intensive mobile application augmentation.
- To evaluate the performance of the proposed framework by considering several performance metrics such as user service time, cloudlet response time and resource utilization.

1.5 Proposed Methodology

This study has been conducted in four steps corresponding to the four research objectives. The steps are outlined as follows.

A comprehensive review and synthesis of recent mobile computation outsourcing efforts based on cloudlets in MCC is undertaken to identify the impact of the resource scarcity problem in cloudlets. The cloud-based literature on cloud-/cloudlet-based mobile augmentation performance is reviewed using scholarly digital libraries, particularly IEEE, Science Direct, Springer and Web of Science. We also examine the impacts of finite-resource cloudlets in MCC from different dimensions and identify several research issues through the literature review. We identify the most significant research problems to address in this research.

We investigate the identified problem and verify its significance through an experimental analysis in a real MCC environment using an android-based smartphone and Openstack cloud Havana version on the Linux Ubuntu 12.04 platform. Using a series of experiments on local mobile devices and a cloudlet computing test bed, we evaluate the performance when executing computation-intensive applications on the cloudlet to verify the severity of the identified research problem.

To alleviate the identified problem, we implement and design a performance enhancement framework for cloudlets in MCC to achieve efficient computation-intensive mobile application augmentation in cloudlets. The proposed framework is composed of three layers of various building blocks of workflows. To achieve the efficient execution of cloudlet applications and enhanced cloudlet performance, we design the MobiCoRE framework for orchestrating resource sharing for mobile devices on cloudlets. The implementation is used to test the proposed solution and to provide a comparison with the mathematical model.

We evaluate the performance of our proposed framework using a mathematical model as well as an empirical analysis. A computation-intensive prototype application is used for empirical Wi-Fi environment. The entire solution is mathematically modelled. The verification. The performance evaluation test bed is built using real android-based smart-phones and Open stack cloudlet-based resources in a model is verified for correctness and accuracy by comparing the results generated by the model with the empirical results. Subsequently, the model as well as the empirical results are used for data collection, optimal parameter value computation and evaluation of the proposed solution in terms of effectiveness in addressing the problem of cloudlet resource scarcity.

1.6 Thesis Layout

The thesis layout is shown in Figure 1.2. The remainder of the thesis has been divided into six chapters. The organization of the thesis is shown in Figure 1.3.

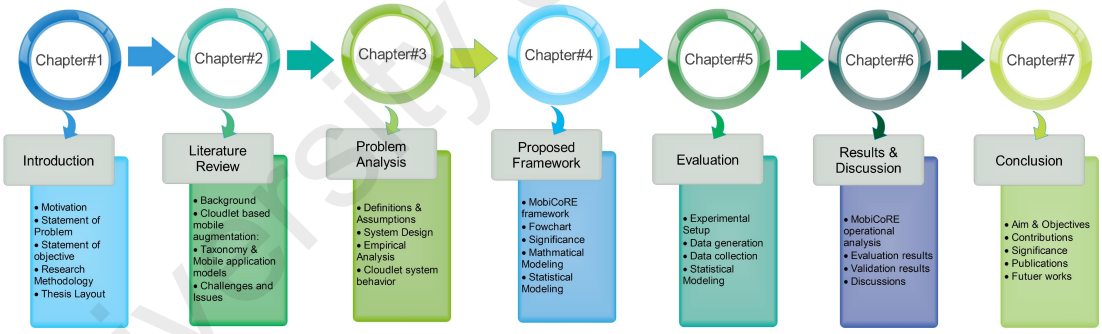

Figure 1.2: Thesis Layout Schematic Diagram

Chapter 2: Review of Literature

Chapter 2 attempts to review the research undertaken in the field of cloudlets and state-of-the-art mobile computational cloudlet-based mobile application augmentation. This chapter discusses the MCC environment, reviews mobile application computation approaches used to identify resource scarcity problems faced by cloudlets and provides various solutions. The taxonomy of cloudlet resource augmentation research is devised by considering several aspects to better understand the domain of cloudlet-based mobile

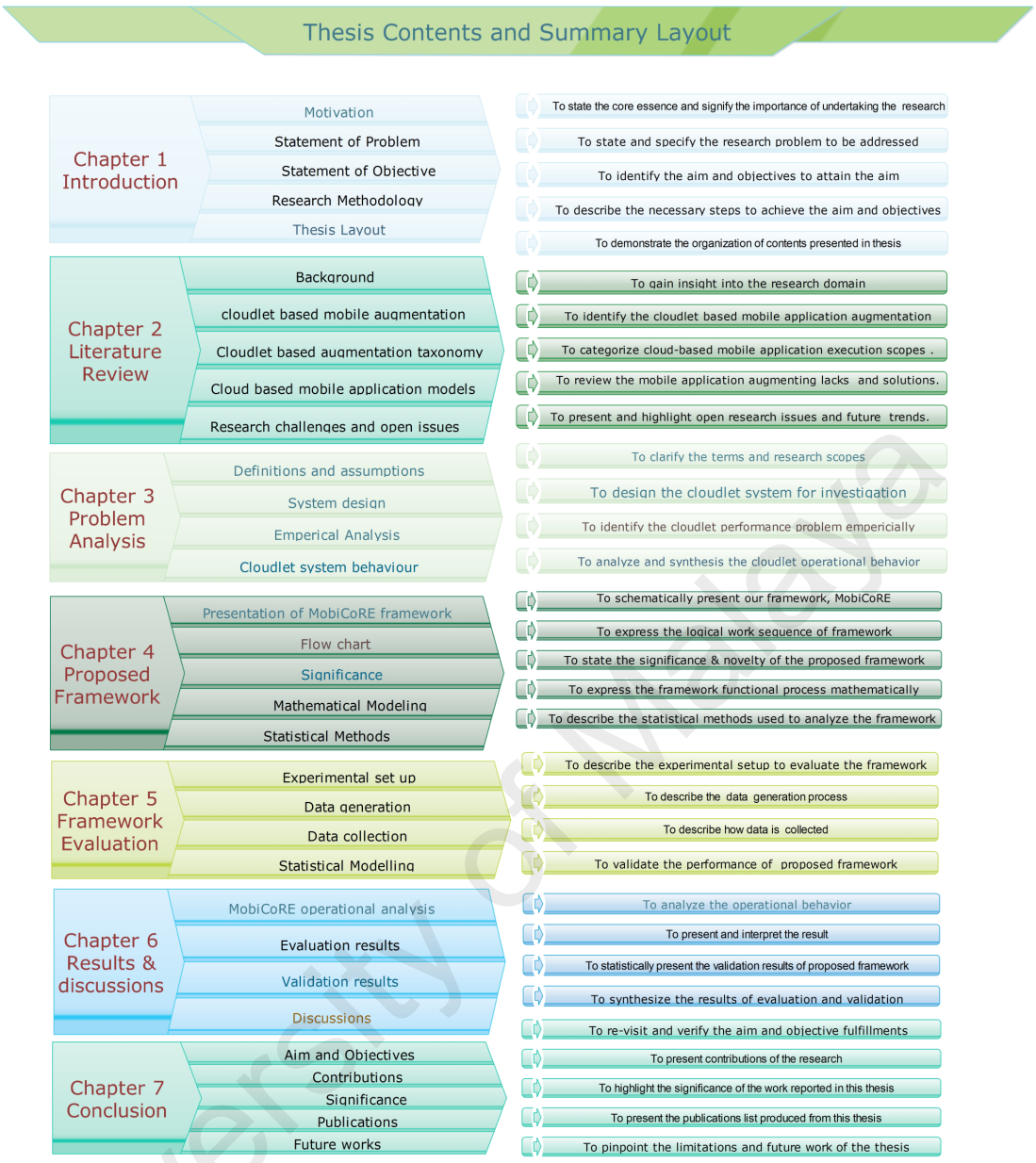

Figure 1.3: Thesis Layout Diagram

application augmentation. Finally, we investigate the impact of cloudlet resource scarcity on cloudlets and its benefits and challenges in terms of future research.

Chapter 3: Resource Scarcity in Cloudlets: Problem Analysis

Chapter 3 presents an investigation and analysis of the impacts of finite resources on the performance of cloudlet applications. Using a series of experiments with an android-based mobile device and an Open Stack cloudlet on a Linux Ubuntu platform, we identify the impact of cloudlet efficiency, user service time, and CPU utilization on the perfor-

mance of the cloudlet. We verify and establish the research problem and demonstrate its significance.

Chapter 4: MOBICORE: Mobile Device based Cloudlet Resource Enhancement Framework

In Chapter 4, we propose a performance enhancement framework for cloudlets in MCC to enhance the performance of cloudlets. A presentation of the framework is given, and the functional and non-functional properties of the main system components are explained. The significance of the proposed framework is highlighted, and the performance evaluation setup is described.

Chapter 5:Implementation and Evaluation

Chapter 5 presents the techniques employed for the experiment and collection of data and the evaluation methodology. We describe the evaluation methods and statistical modelling that have been used to evaluate and validate the performance of the proposed framework. The benchmarking model is explained, and the methodology used to build the statistical model is described. The method used to validate the statistical model is also described.

Chapter 6: Results and Discussion

In Chapter 6, we present the results of our performance evaluation and discuss the findings from different perspectives of applications scenarios. We compare and contrast the benchmarking results with the results of the statistical modelling (which is validated) to validate the performance of the proposed framework.

Chapter 7: Conclusion

Chapter 7 concludes the thesis by describing how the aim and objectives of the research are achieved. The main contributions are summarized, and the research significance and the proposed framework's usability are highlighted. We list publications, including conference and journal articles, generated based on this research work. Future

works and our limitations conclude the chapter.

CHAPTER 2 : LITERATURE REVIEW

This chapter reviews the cloud, cloudlet and mobile cloud computing aspects and traditional solutions for application outsourcing and provides a thematic taxonomy for current applications in cloudlet-based resource augmentation. We analyze the implications and critical aspects of the application augmenting techniques and frameworks, highlight the challenges and identify issues in the cloudlet development and management of application offloading in Mobile Cloud Computing (MCC). Further, we investigate commonalities and deviations that play vital roles concerning critical cloudlet-performance-related parameters. In this chapter, we highlight the state of the art with reference to challenges relating both technologies. We summarize the chapter by highlighting the limitations of cloudlets, therein presenting the challenges that form the basis of our research.

The chapter is organised as follows. Section 2.1 highlights the use of clouds and cloudlets in mobile cloud computing and introduces cloud-based mobile applications. Section 2.2 describes the application execution fundamentals in MCC. Section 2.3 discusses the literature on cloudlet augmentation. Cloudlet resource optimization and management are highlighted in Section 2.4. Cloudlet applications are detailed in Section 2.5. Section 2.6 highlights the limitations of the state of the art with reference to the resource scarcity problem. Research challenges and open issues are outlined in Section 2.7. Section 2.8 concludes the chapter.

2.1 Background

This section describes the concepts of cloud computing and mobile cloud computing along with the fundamentals of cloudlets and mobile application augmentation in the computational cloud.

2.1.1 Cloud and Mobile Cloud Computing

The introduction of cloud computing significantly changed computational paradigms and technologies (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009). Cloud computing aggregates the resources of smaller computational units to provide computing as a service (Whaiduzzaman, Gani, Anuar, et al., 2014). Instead of purchasing computers that remain unused most of the time and that are under-utilized the remaining period of time, the computation time is purchased as needed (Whaiduzzaman, Haque, Rejaul Karim Chowdhury, & Gani, 2014). In other words, computing can be considered as similar to other utilities such as electricity, gas and water (Armbrust et al., 2010). In short, the primary advantage of cloud computing is the ability to enable the hassle-free and cost-effective availability of high-end computation and storage resources (Qi, Shiraz, Gani, Whaiduzzaman, & Khan, 2014).

Mobile cloud computing is becoming a well-established computational paradigm. Mobile device resources are drastically improving; however, simultaneously, resource-intensive mobile applications are being developed (Shiraz, Gani, Khokhar, & Buyya, 2013). Mobile cloud computing helps mobile devices augment their computational resources by offloading computation-intensive tasks to clouds or nearby cloudlets (Abolfazli, Sanaei, Ahmed, Gani, & Buyya, 2014). Although clouds have sufficient resources for user tasks, the propagation delay involved in accessing the clouds from the mobile devices is significantly higher and often unacceptable for many applications (Whaiduzzaman, Sookhak, Gani, & Buyya, 2014). Similarly, the bandwidth is limited for high-demand in-

teractions between the application front end of the mobile device and the back-end computation in the cloud (Ahmed, Akhunzada, et al., 2015). On the other hand, cloudlets, which are typically one hop from mobile devices, do not suffer from these problems. However, the resources available in cloudlets are often limited.

Mobile cloud computing is a computation paradigm whereby mobile devices augment their computational and storage resources using clouds and nearby cloudlets. The use of clouds for mobile device resource augmentation has seen the greatest development in the domain of mobile cloud computing. A number of researchers have analyzed various aspects of mobile resources and clouds in terms of the feasibility of mobile cloud computing using clouds. For example, Kumar and Lu (Kumar & Lu, 2010) have shown that offloading tasks from mobile devices to clouds can reduce the energy consumption of mobile devices. Miettinen and Nurminen (Miettinen & Nurminen, 2010) have analyzed the energy consumption of mobile devices and have shown that not all applications can benefit from the offloading of tasks to the cloud. Canepa and Lee (Huerta-Canepa & Lee, 2010) have proposed a virtual cloud computing platform wherein mobile devices can use resources from neighbouring devices in addition to available cloud resources.

2.1.2 Cloudlets and Mobile Cloud Computing

Cloudlets are small-scale clouds located in close proximity to mobile users. Typically, a mobile user can access the cloudlet through one-hop wireless access. In the future, cloudlets are expected to be deployed along with the hotspots in locations such as offices, clinics and cafes. Cloudlets are installed on discoverable, localized, stateless servers running one or more Virtual Machines (VMs) on which mobile devices can augment resource-intensive mobile applications (Satyanarayanan et al., 2009). Such cloudlets are formed by a set of powerful, compared to the mobile devices, computers; are well-connected to the Internet; and are available for use by nearby mobile devices (Bahtovski

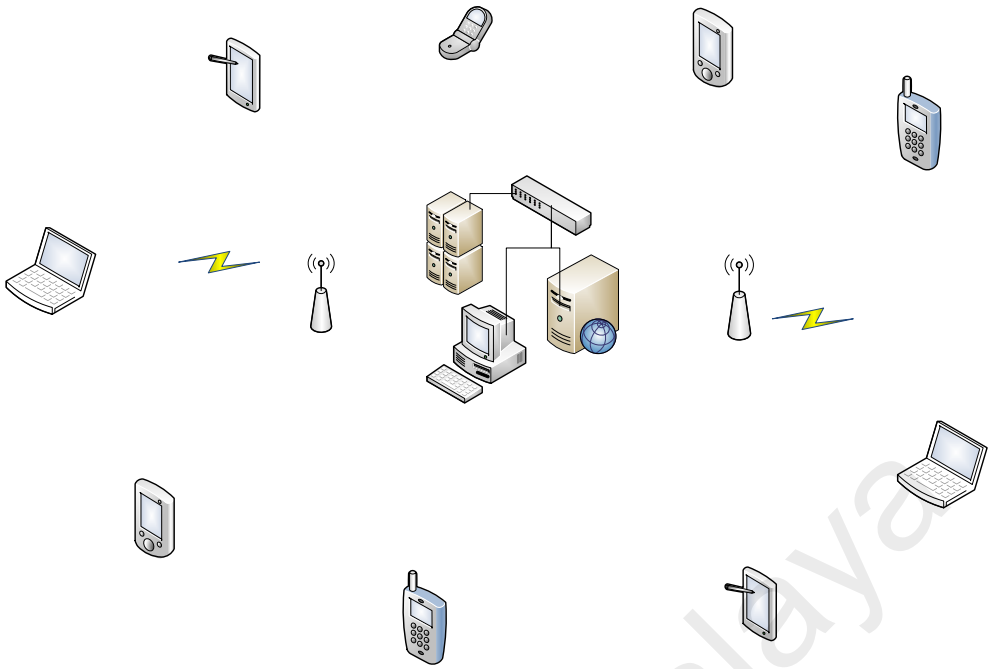

Figure 2.1: Cloudlet and Surrounding Mobile Devices Connected via Wi-Fi

& Gusev, 2014a). A typical cloudlet access scenario is shown in Figure 2.1, where mobile devices as well as thin clients can access the cloudlet over WiFi using single-hop access.

Satyanarayanan et al. (Satyanarayanan et al., 2009) first introduced the cloudlet concept and described it as a "data center in a box". A cloudlet is self-managing, requires minimal power, and requires Internet connectivity and access control for setup. The simplicity of management makes it feasible to deploy cloudlets on business premises such as coffee shops or a doctor's office, quite similar to hotspots. Researchers have highlighted multiple advantages of cloudlets over the traditional cloud-based augmentation of mobile devices. These advantages include speed of service, support of mobility, enhanced application performance and battery life and reduced roaming data costs (Satyanarayanan et al., 2009; Rahimi, 2012; Bohez, Turck, Verbelen, Simoens, & Dhoedt, 2013). Cloudlets have an important role in mobile cloud computing because of these advantages.

Cloudlets can be trusted or non-trusted computing resources. A trusted cloudlet is generally available for mobile devices within small office environments, where employees

can access the cloudlet using their security credentials. Such cloudlets are expected to be smaller in number. The majority of cloudlets are to be deployed in a non-trusted fashion, quite similar to hotspots.

A cloudlet can act independently when it is the only source of computation available for mobile device resource augmentation. Alternatively, a cloudlet can act as a proxy, thereby providing access to the clouds for delay-insensitive application components and performing computations for delay-sensitive components locally. The task division between cloudlets and clouds can also be based on the workload, whereby light-resource tasks can be executed in the cloudlet and resource-intensive tasks can be delegated to the clouds for computation. Cloudlets act as a middle tier in this type of three-tier hierarchy of mobile device, cloudlet, and cloud (Rahimi, 2012; Satyanarayanan et al., 2013).

Cloudlets offer several advantages over distant clouds for mobile users. First, mobile users obtain instant and direct access to the cloudlet due to the close proximity of the user and the cloudlet. This eliminates several drawbacks introduced by the communication latency, jitter, and slow data transfer of cellular and WAN networks that are experienced when distant clouds are accessed. Second, the conventional benefit of offloading computational-resource-intensive tasks onto a cloud is preserved in cloudlets, namely, the mobile device is not subject to resource scarcity. Third, cloudlets provide smaller application execution times compared to the mobile device, thereby reducing the waiting time for the offloaded application results and providing a significantly enhanced user experience. Finally, because the cloudlet is in close proximity to the mobile user, expensive data transfers can be avoided, thereby lowering costs to the user. The cloud based computing resources taxonomy (Abolfazli, Sanaei, Ahmed, et al., 2014) is shown in Figure 2.2

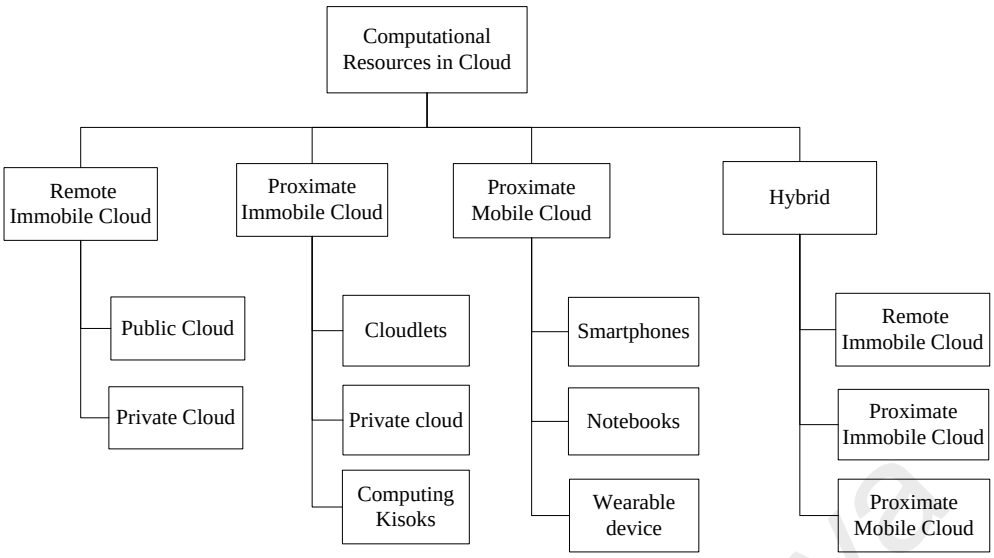

Figure 2.2: Cloud based Source of Computing Resources

2.1.3 Cloud based Mobile Applications

With the ever increasing number of smart phones, cloud-based mobile applications have become ubiquitous (Shiraz, Whaiduzzaman, & Gani, 2013). We only touch the tip of this iceberg by mentioning some of the most commonly used applications and application engines (Shiraz, Ahmed, Gani, & Han, 2014). The Mobile App Store from Apple (Apple, 2015), the Play Store from Google, and Microsoft Lumia for Nokia applications (Lumia, 2015) are among the popular application stores and engines, which host millions of cloud-based mobile applications (Abolfazli, Sanaei, Gani, Xia, & Yang, 2014). Among the most popular cloud-based mobile applications, well-known applications include Google Mail, Google Maps, Dropbox, Microsoft Outlook, and iCloud-based apps.

Christensen (Christensen, 2009) has proposed the use of RESTful web services to apply cloud computing to next-generation mobile applications. Nkosi and Mekuria (Nkosi & Mekuria, 2010) have studied the use of clouds in terms of enhancing mobile-based health applications. The entire body of research work is supported by a huge set of commercial activities for supporting mobile-based application development using clouds as the back end.

Given that smartphones will become the major technology of the future, it is safe to state that cloud-based mobile applications are the future and that mobile cloud computing is the computational paradigm that will shape trends in the coming years.

2.1.4 Limitations of Clouds in MCC

Despite their significant advantages and substantial development as well as the millions of cloud-based mobile applications, the use of clouds for mobile cloud computing has its own limitations (Whaiduzzaman & Gani, 2014). Issues such as WAN latency, jitter, congestion and slow data transfer hinder the development of resource-intensive applications because of increased power consumption (Rahimi, 2012). Currently, a majority of cloud-based mobile applications are developed for entertainment and social interaction (Ahmed, Gani, Khan, Buyya, & Khan, 2015). Applications for real-time audio and video interaction as well as mission-critical applications, such as healthcare, are affected by these factors and have not yet found practical implementation at large scales (Ahmed, Gani, Sookhak, Ab Hamid, & Xia, 2015). To alleviate these problems, clouds are being taken closer to the user through the cloudlet concept (Bohez et al., 2013). Table 2.1 summarizes the comparison of the cloudlets with the distant clouds.

2.2 Application Execution in MCC

All cloudlets as well as cloud-based applications rely on application execution frameworks for managing various aspects of application execution. In the next section, we briefly discuss the related aspects of application execution in MCC (Liu et al., 2015). A key component in all cases of cloudlet utilization is code offloading from the mobile device to the cloudlet (Abolfazli, Sanaei, Alizadeh, Gani, & Xia, 2014). Cloud and cloudlet-based mobile applications can be executed in one of the three manners: Client-server execution, VM-based execution and code offloading. The cloud/cloudlet application execution taxonomy is shown in Figure 2.3. In the following section, we briefly

Table 2.1: Comparison of Cloudlet and Distant Cloud

Properties	**Cloudlet**	**Distant Cloud**
State	Only Soft State	Hard and soft state
Management	Self-managed and less professional attention needed	All time administered professionally
Environment	Works at business premises as small Data centre	Huge servers with cooling system and devices with hardware infrastructure
Ownership	By the local business owner and decentralized.	Ownership centralized such as Amazon, Rackspace
Network	LAN latency/bandwidth	Internet latency/bandwidth
Distance	Near to the mobile user	Remote cloud
Sharing	Few users at a time	100s-1000s of users at a time
Connection type	Wi-Fi	3G/LTE
Context awareness	Neutral	Neutral
Location granularity	Fine grain location granularity	Lack of fine grained location granularity
Programming Abstraction	Neutral	Neutral
Latency	Favorable	Unfavorable
Privacy	Favorable	Unfavorable
Scalability	Low scalability	Highly Scalable

touch on these three execution types.

2.2.1 Client Server based Mobile Application Execution

A majority of cloud-based mobile applications use the client-server model, wherein most of the computation is performed in the cloud and only a thin client user interface is available on the mobile device. In addition to performing the computations, the data are stored on the cloud. Researchers (Clinch, Harkes, Friday, Davies, & Satyanarayanan, 2012; Satyanarayanan et al., 2013) have studied applications wherein the computation is performed on the cloud or cloudlet and the display is rendered on the mobile device, therein treating the mobile device as a thin client. Qing et al. (Qing, Zheng, Ming, & Haifeng, 2013) have proposed , Cloudlet Aided Cooperative Terminals Service (CACTSE), as a content distribution system for mobile terminals formed by exploiting nearby cloudlet systems.

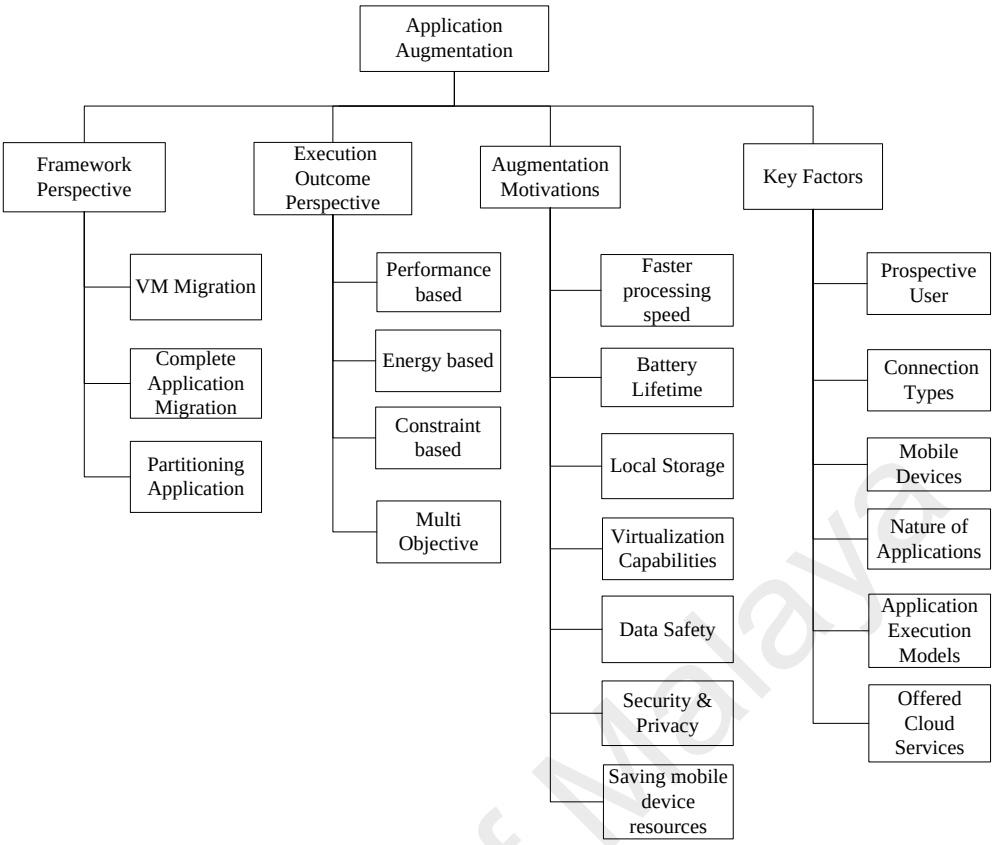

Figure 2.3: Mobile Applications Augmentation Taxonomy

This execution model is affected by intermittent network connectivity. The mobile application ceases to function as soon as the user becomes disconnected from the Internet. Several studies have been conducted to show that network connectivity is intermittent (Silva, Silva, & Boavida, 2014; Treurniet, 2014). Similarly, many researchers have attempted to predict and mitigate the impact of intermittent network connectivity on mobile devices (Modares, Moravejosharieh, Lloret, & Salleh, 2014; Pirozmand, Wu, Jedari, & Xia, 2014; Vu, Nguyen, Nahrstedt, & Richerzhagen, 2014). However, this problem has not been solved to a level at which cloud-based mobile applications can enjoy uninterrupted network connectivity.

2.2.2 VM Based Mobile Application Execution

Satyanarayanan et al. (Satyanarayanan et al., 2009) have introduced a dynamic VM synthesis mechanism that enables mobile devices to deliver a small VM overlay to a

cloudlet infrastructure that already possesses the base VM from which this overlay was derived. The infrastructure applies the overlay to the base to derive the launch VM, which initiates the suspended execution at the exact time of suspension. Ha et al. (Ha, Pillai, Richter, Abe, & Satyanarayanan, 2013) have improved the performance of the dynamic synthesis process by reducing the overlay size using a series of optimizations. A deduplication approach has been used to eliminate duplicate data that exist at various sources, such as Input/Output caches and virtual memory, to decrease the size of the VM overlay. Similarly, Harney et al. (Harney, Goasguen, Martin, Murphy, & Westall, 2007) have proposed live VM migration using Xen and IPv6 support. These two components ensure that a virtual network is not required between migrating elements, thus reducing the migration overhead.

Similar to client-server techniques, these techniques are highly dependent on network connectivity, and the entire set of computations will likely be lost when the user moves from one cloudlet to the next cloudlet because of the delayed discovery of potential network disconnections. We now discuss the application offloading mechanism for cloud-based mobile application execution. The proposed research in this thesis assumes that such mechanisms are in place, although the research can be applied to other mechanisms of application execution. Therefore, we discuss offloading mechanisms in additional detail.

2.2.3 Application Offloading for Cloud/Cloudlet Execution

Code offloading is a well-established and frequently used technique in the domain of distributed computing. In distributed computing, the computational elements that are offloaded are relatively less delay sensitive and are transmitted using high-speed, limited-error networks. This is not the case in mobile cloud computing, where the computational elements are transported over wireless channels that are prone to errors and that provide

limited data rates. Furthermore, the computations are time sensitive in most cases. Another significant difference is the pre-partitioning of the code to be offloaded in the case of distributed computing, which is not the case with mobile cloud computing.

Researchers in the domain of MCC have attempted to address the additional challenges. MAUI (Cuervo et al., 2010) enables the fine-grained, energy-aware offloading of mobile code to the cloud. MAUI uses a managed code environment for run-time code offloading to maximise energy savings. Automated and optimized mechanisms require minimal effort by the programmer. μ-Cloud (March et al., 2011) is another effort whereby the developers demonstrated that rich mobile applications can be achieved by modelling applications using distributed graphs with components distributed on mobile devices and clouds. In addition, CloneCloud (Chun, Ihm, Maniatis, Naik, & Patti, 2011) has been demonstrated as an automated system that transforms mobile applications to benefit from the cloud. The system is flexible in terms of application partitioning and runtime execution, therein enabling unmodified mobile applications running in an application-level virtual machine. CloneCloud uses static analysis and dynamic profiling for automatic application partitioning with a fine granularity by optimizing execution time and energy.

The Cuckoo framework (Kemp, Palmer, Kielmann, & Bal, 2012) has been proposed to offload computations from smartphones to remote cloud resources to enhance the performance of smart phone applications while reducing energy usage. This framework simplifies the development of smart phone applications to obtain various benefits from computation offloading and provides a dynamic runtime system to select between the remote or local execution of application components.

The elastic application execution model (Zhang, Kunjithapatham, Jeong, & Gibbs, 2011) has been proposed to enhance the seamless and transparent use of cloud resources to augment the capabilities of resource-constrained mobile devices. Soyata et al. (Soyata, Muraleedharan, Funai, Kwon, & Heinzelman, 2012) have designed and implemented

facial recognition applications using a Mobile-Cloudlet-Cloud Architecture(MOCHA) named MOCHA. The authors suggested that the software components on the mobile device can be redeployed at run-time to other nodes in the cloudlet according to optimization criteria such as the execution time (Kristensen, 2010; Verbelen, Simoens, De Turck, & Dhoedt, 2012a) energy consumption and throughput (Cuervo et al., 2010).

Verbelen et al. (Verbelen et al., 2012a; Verbelen, Simoens, De Turck, & Dhoedt, 2012b) have introduced AIOLOS, a mobile middleware framework for cyber foraging on the Android platform. AIOLOS can migrate application components at run-time to a discovered server using an efficient history-based server selection model. Flores et al. (Flores, Srirama, & Paniagua, 2012) have proposed a middleware that can be used to offload applications onto multiple clouds with heterogeneous platforms. Kosta et al. (Kosta, Aucinas, Hui, Mortier, & Zhang, 2012) have proposed the ThinkAir framework to migrate tasks from mobile devices to the cloud by exploiting the concept of smart phone virtualization in the cloud.

Verbelen et al. (Verbelen, Simoens, De Turck, & Dhoedt, 2013), in developing component-based collaborative immersive application frameworks, have proposed an adaptive component-based approach whereby software components and dependencies are configured and defined with parameters. These distributed components are configured at run-time to optimize the user experience. The proposed component-based platform deploys components, application-specific metrics, and developer-specified constraints that optimize application quality and adapt to changing network conditions.

Bohez et al.(Bohez et al., 2013) have proposed offloading middle-ware for collaborative mobile applications to ensure smooth collaboration. Li and Wang (Li & Wang, 2013) have studied the cloudlet accessibility issues resulting from user mobility and estimated the success rate and execution time of offloaded application. Ravi and Peddoju (Ravi & Peddoju, 2013) have presented mobility- and energy-aware mobile device collaboration

with cloudlets using a fuzzy offloading decision approach.

A major effort has been directed towards the partitioning of mobile code for hybrid execution on the cloud and mobile device simultaneously. Furthermore, code annotation and re-writing have become major challenges for making code workable on the cloud when it has been originally written for mobile device platforms. These are issues that hinder the wide-spread use of code offloading in the domain of mobile cloud computing. The present research addresses both issues, as explained in subsequent chapters.

2.3 Cloudlet Resource Augmentation

The problem of cloudlet resource scarcity has been identified by multiple researchers. Consequently, significant research efforts have been directed at this issue. In this section, we present the body of work in this domain and highlight issues faced by current research. Subsequently, we summarize the shortcomings of the techniques, therein forming the basis for the proposed research. The cloudlet resource augmentation mechanisms can be grouped into four categories: (i) cloudlet-based augmentation of cloudlet resources, (ii) cloud-based augmentation of cloudlet resources and (iii) mobile-device-based augmentation of cloudlet resources, (iv) Adhoc-Mobile-devices based Cloudlet Augmentation. Figure 2.4 shows the cloudlet resource augmentation models and different researchers contributions. The remainder of this section provides a description of the mechanisms and highlights their shortcomings.

2.3.1 Cloudlet based Cloudlet Augmentation

Multiple researchers have proposed the sharing of resources amongst cloudlets to address the problem of resource scarcity.

Cloudlet-to-cloudlet resource augmentation has been proposed by Bohez et al. (Bohez, Verbelen, Simoens, & Dhoedt, 2014) . The authors have proposed the runtime optimization of collaborative cloudlet middleware. The runtime collaboration amongst cloudlets

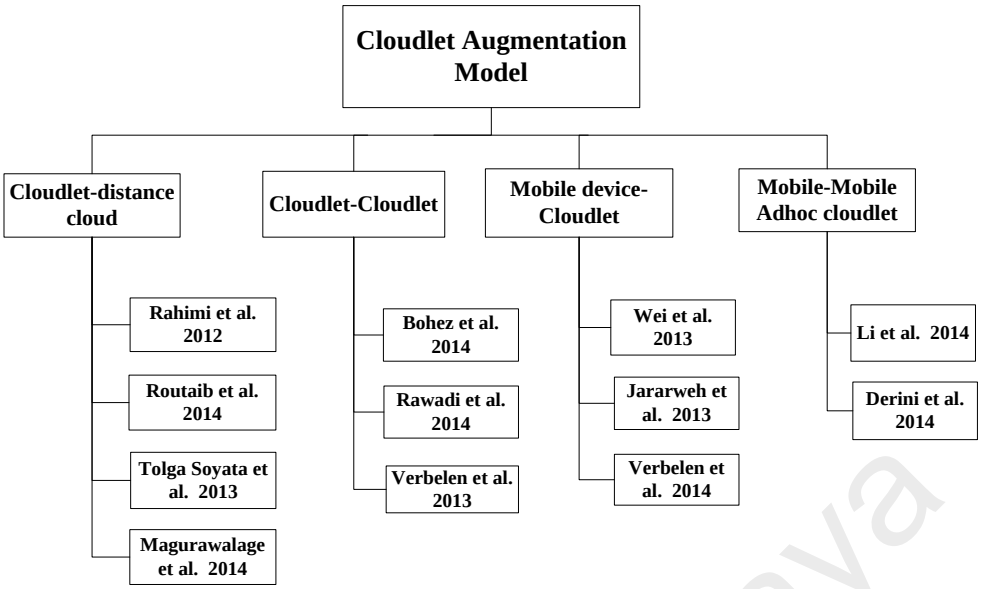

Figure 2.4: Cloudlet based Mobile Application Augmentation Models and Researchers

has been formulated as an optimization problem that describes the cloudlet infrastructure, application structure and application behavior. Two heuristic allocation algorithms based on Simulated Annealing(SA) and Steepest Descent(SD) have been proposed; SD and SA were found to yield very similar performances in terms of average CPU usage and constraint violations. Compared to SD, SA requires over five-times more actions (e.g., component migrations) to be performed in total. To improve the stability of the allocation, a hysteresis factor is introduced to enforce a minimum gain before a reallocation is accepted. However, multiple applications simultaneously run in the cloudlet, and network bandwidth is considered as constant. The theoretical model is used for run-time optimization.

Along similar lines, Rawadi et al. (Rawadi, Artail, & Safa, 2014) have proposed a collaborative cloudlet system that can communicate to delegate tasks to cloudlets. The cloudlets should be linked to a central cloudlet server that is discoverable by users. Mobile users submit requests to the discoverable server, which distributes the tasks to the cloudlets. However, when one cloudlet is overloaded, the probability of finding an underutilized cloudlet in close proximity is very small. Although the model performs well

in the absence of background applications, in practice, during the load hours when one cloudlet is loaded by multiple applications, the probability that an alternate cloudlet in close proximity is free is negligible.

Ceselli et al. (Ceselli, Premoli, & Secci, 2015) have proposed the integration of cloudlets into large-scale mobile access networks. A cluster of cloudlets is deployed to provide uniform access to the mobile users. In this model, virtual machines are associated with mobile users and are allocated to cloudlets. The design of the cloudlet service network involves determining the sites for cloudlet deployment and associating access points with the sites to provide mobile user connectivity, therein considering virtual machine migration and user mobility. The design attempts to ensure the satisfaction of service-level agreements. The authors claimed that, considering user mobility, the proposed design requires 40% fewer virtual machine migrations and 40% fewer cloudlet resources. For the migration of virtual machines, two modules have been considered. The authors found that a strong preference should be given to bulk migrations for delay-stringent services, such as augmented reality support, whereas for applications with less stringent delay requirements, live migration is largely preferable.

The above-mentioned schemes rely on the fact that computations from a cloudlet that is heavily loaded can be shifted to a nearby cloudlet. However, the extensive use of one cloudlet typically indicates that a significant number of users are in the area, indicating high activity because of an event or peak usage hours. Under such conditions, the situation whereby one cloudlet is heavily loaded while a nearby cloudlet is free does not occur. We can safely conclude that although such schemes are useful for coverage enhancement and help in reducing communication overhead as well as costs, this mechanism is less effective for cloudlet resource augmentation.

2.3.2 Cloud based Cloudlet Augmentation

A number of researchers have proposed frameworks wherein a three-layer approach to mobile cloud computing is utilized. Under such approaches, the cloudlets function as a middle layer, therein providing services to mobile users while simultaneously connecting with the clouds to obtain services for themselves or for the mobile devices. In this case, cloudlet resources can be augmented using clouds, where less time-sensitive computations can be shifted to clouds and time-critical execution can be performed on the cloudlet.

Rahimi et al. (Rahimi, Venkatasubramanian, Mehrotra, & Vasilakos, 2012) have proposed a two-tier architecture and shown that rich mobile applications, such as augmented reality and image processing, can use this architecture by modelling such applications as a workflow of tasks. Workflows are used to decide upon execution on local or distant clouds based on a simulated annealing approach. The authors have introduced MAP-Cloud, a hybrid, tiered cloud architecture consisting of local and public clouds that can be leveraged to increase both the performance and scalability of mobile applications. The authors have modelled the mobile application as a work-flow of tasks and attempt to optimally decompose the set of tasks for subsequent execution on the mobile client and 2-tier cloud architecture considering multiple QoS factors such as power, price, and delay. The authors have proposed the CRAM framework, which is able to achieve an approximately 84% optimal solution for large numbers of users.

Mobile Cloud Hybrid Architecture (MOCHA) has been proposed by Soyata et al. (Soyata, Muraleedharan, Funai, et al., 2012). MOCHA is a mobile-cloudlet-cloud framework. The authors have developed algorithms that minimize the overall response time for face recognition based on the estimated communication latencies and processing power of the cloud. The framework provides face recognition applications while collaborat-

ing between mobile devices, cloudlets, and distant clouds. The critical aspect of the framework is to address the issue of how to perform task partitioning from mobile devices to clouds and distribute computational loads amongst cloud servers and cloudlets to minimize the response time given diverse communication latencies and server computation resources. The authors have designed, implemented, and validated the basic functionalities of MOCHA as a proof of concept and developed algorithms that minimize the overall response time for face recognition. The experimental results show that high-power cloudlets are technically feasible and indeed help reduce overall processing time when face recognition applications run on mobile devices using the cloud as the back end servers.

Routaib et al. (Routaib, Badidi, Elmachkour, Sabir, & Elkoutbi, 2014) have proposed a centralized architecture for the use of cloudlets and clouds. The authors have modelled the architecture using Continuous-Time Markov Chains (CTMCs) to estimate the delay incurred from different nodes for simulated search engine queries. The authors have also proposed a cloudlet architecture for managing data caches. A new routing algorithm for mobile search engines for finding mobile content on websites has also been proposed. Initial simulation results show that the usage of a cloudlet-based architecture provides increased performance in terms of latency, delay and synchronization.

Magurawalage et al. (Magurawalage, Yang, Hu, & Zhang, 2014) have proposed a system architecture consisting of a new middle layer called a cloudlet layer. This layer operates between mobile devices and their cloud infrastructure or clone and is composed of cloudlets; thus, it is called a cloudlet layer. Cloudlets are deployed next to IEEE 802.11 access points and serve as a localized service point in close proximity to mobile devices to improve the performance of mobile cloud services. The decision making considers the energy consumption for task execution and the network status while satisfying certain task response time constraints. A data caching mechanism at the cloudlets is also

considered to further improve the overall MCC performance. An offloading algorithm determines whether and where to offload, with the objective of saving battery life for mobile devices while satisfying the response time constraints of applications by increasing efficiency compared to conventional offloading architectures. However, the algorithm lacks experimental verification.

The primary issue with using distant clouds for cloudlet resource enhancement is the transfer of computations from distant clouds to a user when the user moves out of the vicinity of the cloudlet. In this case, only the cloudlet knows which task belongs to which mobile device, and the information is only available to the cloudlet; however, the mobile device, being out of range, is unable to obtain the results. Mobile device mobility management and interactive and collaborative cloudlets can address this issue. Another issue is the pricing model for accessing such services from clouds. In general, cloudlets are free resources and are not expected to be augmented by a paid cloud service. On the other hand, if we assume that the mobile device has an associated cloud service, it is highly unlikely that the cloud service can allow the cloudlet to use its services on behalf of the mobile device. Who should pay for the cloud server and what is the incentive for such payment remain as unanswered questions. Such administrative issues make this option less attractive.

2.3.3 Mobile Device based Cloudlet Augmentation

Cloudlet resource augmentation using mobile devices has been studied by multiple researchers. Wei et al. (Wei, Fan, Lu, & Ding, 2013) have proposed the Hybrid Local Mobile Cloud Model (HLMCM) as an extended cloudlet using mobile devices. The authors have proposed Hybrid Ant Colony algorithm-based Application Scheduling (HACAS) for efficiently scheduling applications given the limited resources of the cloudlets. The algorithm only considers the available resources and does not consider overhead when

calculating the advantage ratio of mobile devices for joining the cloudlet. Consequently, a mobile device can be significantly overloaded after joining the cloudlet, although its service requirements from the cloudlet might be limited. Similarly, the task might be offloaded to a very weakly resourced device. Such devices can create bottlenecks and lead to poor overall performance. Furthermore, the departure of devices from the cloudlet can result in a partially completed task leaving with the device.

Jaraweh et al. (Jararweh, Tawalbeh, Ababneh, & Dosari, 2013) have proposed merging cloudlets with mobile devices to obtain reduced power consumption and communication latency. Verbelen et al. (Verbelen, Simoens, De Turck, & Dhoedt, 2014) have proposed a platform to enhance cloudlet resources by dynamically including devices within wireless networks and using application-component-level migration. The problem with this approach is the lack of consideration of the load added to the mobile devices compared to the advantages provided by using the cloudlet.

Ali et al. (Ali, Ahmad, & Amin, 2014) have proposed a scalable and lightweight intelligent distributed surveillance system using a framework that integrates the Internet-of-Things (IoT) and cloudlets. To address the resource scarcity of the IoT and the dependency on massive data transmissions to distant clouds, this system provides real-time on-site object detection, distributed retrieval and processing of critical and sensitive data. Issues concerning data migration over limited-bandwidth and high-latency communication networks and the heterogeneous nature of data obtained from the surveillance system are successfully addressed. Stored data in cloudlets should be maintained and processed with high security.

2.3.4 Adhoc-Mobile-devices based Cloudlet Augmentation

Li and Wang (Li & Wang, 2014) have proposed the fundamental mobile cloudlet properties that unfold whether and when a mobile cloudlet can provide mobile applica-

tion service. Specifically, they investigate the cloudlet size, cloudlet node's lifetime and reachable time. They also investigate the required conditions which are needed to form mobile cloudlet in order to provide mobile application services. Between the one hop and multi hop mobile cloudlets, one hop is adopted for proximity and simplicity. However, still lacks of design and implementation of mobile applications on a mobile cloudlet system to feasibility study of mobile cloudlet computing. Similarly, Mohamed Nazih El-Derini et al. (El-Derini, Aly, El-Barbary, & El-Sayed, 2014) have proposed Droid-Cloudlet, a cloudlet architecture in any available mobile device with abundant processing or power resources, can participate as a server. Offloading is carried out dynamically at runtime according to specific policies that target reducing execution time and saving battery. The mobile device suffering lack of resources can offload partitions of its running application to those servers. However, not adequate prototype implementation has been done yet.

2.4 Cloudlet Resource Optimization and Management

Researchers have also proposed mechanisms for achieving efficient cloudlet resource management. Hoang et al. (Hoang, Niyato, & Wang, 2012) have proposed an SMDP-based optimization model for MCC by integrating cloud computing hotspots with cloudlets. A resource allocation model for the mobile cloud computing hotspot with a cloudlet has been proposed by optimizing the dynamic resource sharing of mobile users in Mobile Cloud Computing (MCC). The authors have used a Semi-Markov Decision Process (SMDP) that transforms into a linear Programming (LP) model to determine an optimal solution. This model maximises the reward (e.g., the revenue of the service provider) of the resource usage for MCC hotspots while meeting the QoS requirements of mobile users.

Xia et al. (Xia, Liang, & Xu, 2013) have proposed an admission cost model for

resource consumption in cloudlets. The online request admission issue for cloudlets with the objective of maximizing the system throughput and a novel admission cost model have been proposed for addressing resource consumption using efficient control algorithms. The maximization of the system throughput of the cloudlets is such that each request can be represented by a demand resource vector. The proposed algorithm outperforms other heuristics in terms of system throughput. Li and Wang (Li & Wang, 2014) have analyzed the hybrid mobile cloudlet performance. The authors have shown that frequent access to mobile devices can increase the available computational resources and that mobility does not significantly affect this resource pool in the long term. The authors have also derived upper and lower bounds on the computing capacity and computing speed of a mobile cloudlet. Although these mechanisms can optimize the use of existing resources, the additional resources offered are generally incremental and not sufficient to address the resource scarcity problem faced by cloudlets.

Several researchers have proposed mechanisms to enable efficient cloudlet resource management. Zhang et al. (Yang, Niyato, & Wang, 2015) have developed an optimal offloading algorithm for mobile users in an intermittently connected cloudlet system, therein considering the user local load and the availability of cloudlets based on user mobility patterns and cloudlet admission control. The authors have derived the probability of successfully performing offloading actions. They formulated and solved a Markov Decision Process (MDP) model to obtain an optimal policy for the mobile user with the objective of minimizing the computation and offloading costs by considering a stochastic mobility model to describe an intermittent connection between mobile users and cloudlets. Mobile application is divided into multiple phases, and the user has the flexibility to make the decision to run each phase locally or to offload the computations to nearby cloudlets.

Hoang et al. (Hoang, Niyato, & Le, 2014) have proposed Simulation-Based Optimization for Admission Control of Mobile Cloudlets, which is a simulation-based algo-

rithm that is applied to obtain the optimal policy for the MDP. The algorithm can estimate the performance measure to update the policy gradient in an online manner. The complexity issues are addressed to obtain the optimal admission control policy based on parameterization and approximate gradient improvement to maximize revenue. The authors have shown that the algorithms are efficient and converge.

Kommineni (Kommineni, De, Alladi, & Chilukuri, 2014) has proposed Choosing the right cloudlet, a systematic guideline for selecting the optimal nearby cloudlet. This guideline ensures better performance in selecting the optimal cloudlet for a given application. This choice is based on a weighted function of the processing power, available memory, bandwidth on the fixed network and the WAN latency of the cloudlet. Jin et al. (Jin, Song, Wang, Niyato, & Ju, 2015) have proposed a feasible and efficient double auction mechanism to stimulate cloudlets for serving nearby mobile devices to satisfy computational efficiency, individual rationality, budget balance, and truthfulness constraints. The authors have considered an appealing MCC application paradigm for mobile devices to acquire the resources of nearby cloudlets. Numerical results confirm the analysis and demonstrate the high system efficiency.

Resource management systems addressing cloud providers must provide a uniform interface for various services. Munteanu et al. (Munteanu, Şandru, & Petcu, 2014) have discussed the solution adopted by a recently developed open-source cloud for multi-cloud application deployment systems for automated cloud resource management. They provided a flexible approach to encompassing new cloud service offers and resources that enables resource abstraction and automated management. Yamato et al. (Yamato, Muroi, Tanaka, & Uchimura, 2014) have described the development of a template management technology for building virtual resource environments on OpenStack. The mechanism tracks transaction management functions, such as rollback and roll forward, in the case of abnormal failures during stack operations, which allows users to easily replicate and

build virtual environments to achieve efficient resource utilization.

To effectively manage cloud/cloudlet resources, Chen et al. (Chen, Wang, & Pedram, 2013) have presented an architecture-based approach. Cloud resources are abstracted as runtime models to propagate any observable runtime changes in target resources to corresponding architecture models. A customized model is constructed according to the personalized management requirement and according to the synchronization between the customized model and the cloud resources. The runtime models are obtained through model transformation.

Kara et al. (Kara, Soualhia, Belqasmi, Azar, & Glitho, 2014) have proposed virtualization task scheduling and computational resource sharing strategies based on genetic algorithms. They analysed and simulated strategic algorithms used to measure the cloud resource performance. Rak et al. (Rak, Venticinque, Máhr, Echevarria, & Esnal, 2011) have proposed the mOSAIC framework, which attempts to offer a solution for the development of interoperable, portable and cloud-provider-independent cloud applications by introducing the mOSAIC monitoring components to custom monitoring systems for cloud applications.

A cloudlet-based multi-lateral resource exchange framework (Wu & Ying, 2015) that does not rely on central entities has been proposed for mobile users. A novel virtual-currency-tailored framework has been proposed for efficient resource exchange markets, wherein flexible pricing strategies are adopted by individual users. A prototype design for enabling seamless trading amongst mobile users on Internet bandwidth has been introduced as a proof of concept with minimal user intervention. Both simulations and experiments have demonstrated the practicality and efficiency of the system. Bohez et al. (Bohez, Verbelen, Simoens, & Dhoedt, 2015) have proposed cloudlet middleware that more optimally distributes loads across the cloudlet nodes. User mobility and fluctuations in wireless bandwidth are considered as factors in load distribution. Two heuristic alloca-

tion algorithms based on Steepest Descent (SD) and Simulated Annealing (SA) have been used. The authors have found that SD produces 4-times fewer reallocations compared to SA and that it is more stable.

2.5 Applications of Cloudlets

Given the clear advantages of cloudlets in the domain of mobile cloud computing, researchers have proposed multiple applications that use cloudlets as the underlying technology. The first and most prominent application has been proposed by Satyanarayanan et al. in the form of augmenting cognitive reality for physically disabled people (Satyanarayanan et al., 2009). The authors have proposed that people with special needs can utilise a body-mounted computational unit that can connect to nearby cloudlets for computation augmentation as the user moves around. The cloudlets can also share the precomputed information about the surroundings with the body-mounted computing device. This can greatly facilitate the day-to-day navigation of the user.

Multiple researchers, including Achanta et al. (Achanta, Sureshbabu, Thomas, Sahitya, & Rao, 2012) and Bahtovski and Gusev (Bahtovski & Gusev, 2014b), have proposed cloudlet-based multilingual dictionaries. Achanta et al. (Achanta et al., 2012) have proposed a cloudlet-based multilingual dictionary that uses a system wherein nearby computers are used as a "cloudlet" to relieve the mobile device from most of its computational load. This system does not require continuous Internet connectivity and is not affected by WAN latency. This solution can be used to provide language translation in scenarios with limited bandwidth and in specialized and smaller settings wherein the complexity and expense of a standard cloud deployment are not feasible.

Bahtovski and Gusev (Bahtovski & Gusev, 2014b) have proposed a cloudlet-based dictionary application specifically developed to be processed on a nearby cloudlet as well as on a distant cloud for translating words using a VM deployment on the cloud/cloudlet.

The translation is subsequently displayed on the mobile device. This approach is faster and provides a better performance. Interactive Mobile Cloud Applications (IMCAs) have been proposed by Fesehaye et al. (Fesehaye, Gao, Nahrstedt, & Wang, 2012). These applications include editing, video streaming and chatting, with mobile nodes moving from one cloudlet coverage area to another cloudlet coverage area. The authors have shown that the introduction of cloudlets for such applications reduces data content transfer delay and increases content delivery throughput. Other researchers have also proposed the use of cloudlets for real-time collaborative applications (Bohez et al., 2015, 2014).

A hierarchical cloudlet-based storage architecture for mobile clouds (Duro, Blas, Higuero, Perez, & Carretero, 2015) has been proposed by Duro et al. The authors have implemented a hierarchical cloud storage system for mobile devices based on multiple I/O caching layers. The system offers flexibility and easy deployment, and generic features enable implementation in heterogeneous environments in terms of hardware and networking. The system dynamically permits the system's capacity to increase with increasing number of users. Park et al. (Park, Parwani, Satyanarayanan, & Pantanowitz, 2012) have proposed the use of cloudlets in pathological data recording. Chang et al. (Chang, Fan, Lo, Hung, & Yuan, 2015) have developed mobile-cloudlet-based applications for depression detection. Soyota et al. (Soyata, Muraleedharan, Langdon, Funai, & Ames, 2012) have evaluated cloudlets for defence applications. Similarly, Wang et al. (Qing et al., 2013) have proposed the use of cloudlets for terminal content delivery. Jindal and Dave (Jindal & Dave, 2014) have proposed security protocols for distributed cloudlets.

Quwaider and Jararweh (Quwaider & Jararweh, 2013) have proposed cloudlet-based Big Data processing for big data analysis networks . The authors extended their work to Wireless Body Area Network (WBANs) in a subsequent publication (Quwaider & Jararweh, 2015). Khan et al. (Khan, Wang, Grecos, Luo, & Wang, 2013) have proposed an integrated cloudlet and wireless mesh network framework in the form of MeshCloud

for real-time applications. In another study, the MC-Skynet (Bayat & Lutfiyya, 2014) framework has been introduced as an fine-grained offloading approach and to support runtime and dynamic partitioning of an application by employing cloudlet mesh networks. Jindal and Dave (Jindal & Dave, 2014) have proposed a distributed cloud architecture for mobile multimedia services.

Ali et al. (Ali et al., 2014) have proposed distributed surveillance systems based on cloudlets. Along similar lines, Shi et al. (Shi, Abhilash, & Hwang, 2015) have proposed a new cloudlet mesh architecture for security enforcement to provide trusted mobile cloud computing. Chi et al. (Chi, Wang, Cai, & Leung, 2014) have proposed the use of cloudlets for sharing downloaded game data. Cai et al. (Cai, Leung, & Hu, 2014) have proposed the concept of Gaming as a Service (GaaS) for mobile devices. In short, a large variety of applications of cloudlets in the domain of mobile cloud computing have already been proposed, and the near future will see many more applications.

2.6 Limitations of Cloudlets

Cloudlets have several significant beneficial features. However, cloudlets also suffer from several limitations. We list several drawbacks here.

1. Cloudlets are normally resource constrained due to their limited resources compared to remote commercial clouds such as Amazon and Rackspace. Among the various resources, the essential resources that directly impact the mobile augmentation process are CPU and memory (Rawadi et al., 2014; Xia et al., 2013).

2. Unlike distant clouds where user soft states as well as user hard states can be stored for extended periods of time, in the case of cloudlets, only soft states can be saved because of limited permanent storage. A soft state is also only stored for a limited time period because a subsequent visit by the same user is not expected to occur in the near future.

3. Resources of cloudlets are less or not at all scalable due to the simplicity of the cloudlet deployment and management (Rahimi, 2012).

4. In case of middleware designs for cloudlets, development, deployment and administration are considered to be tedious tasks (Bohez et al., 2015) because any such system must be fully automated and must not require user intervention.

5. Cloudlets should be discoverable by the mobile user. In the case of multiple cloudlets in the vicinity, users must be able to decide which cloudlet to use. User-specific criteria must be maintained to select the most feasible cloudlet for user task augmentation for the mobile device to cloudlet (Kommineni et al., 2014) .

6. User mobility has a substantial impact on cloudlet users. Therefore, the probability of mobility should be considered to ensure a good user experience for rich applications. Researcher have worked in this direction, and examples include the work of Li and Wang (Li & Wang, 2014) and Jararweh et al. (Jararweh et al., 2013).

7. If use of the cloudlet requires payment, then identifying a pricing scheme/model for every application and use of resources and implementing an instant payment system become difficult tasks (Li & Wang, 2013).

8. Unknown cloudlets from the user perspective always present a privacy and security issue for mobile users (Bahtovski & Gusev, 2014a) .

2.7 Research Challenges and Open Issues

Despite several applications of cloudlets and their significance in the domain of mobile cloud computing, a number of issues concerning cloudlets need to be addressed. This section presents several concurrent issues in mobile application augmentation and research challenges of mobile application augmentation. We focus on cloudlet-based mobile application augmentation. The proper management of cloudlets for addressing the

Table 2.2: Cloud/cloudlet based Augmentation Approaches and Issues

Work	Author and Year	Approach
Hybrid Local Mobile Cloud Model (HLMCM)	Xianglin Wei et al , 2013	Integrated cloudlet with mobile device
CACTSE: Cloudlet Aided Cooperative Terminals Service	WANG Qing et al, 2013	Content distribution exploiting cloudlet
MOCHA (Mobile Cloud Hybrid Architecture)	Tolga Soyata et al , 2012	Minimizes response time latencies
Interactive mobile cloud applications (IMCA)	Debessay Fesehaye , 2012	Cloudlets to cloudlet interactive applications
Component based Collaborative Immersive Applications framework	Tim Verbelen,, 2013	Component-based approach for run time optimization.
Cloudlet based centralized architecture	Hayat Routaib et al	Continuous Time Markov-Chain (CTMC) model
MDP based optimization model for MCC	Dinh Thai Hoang et al, 2012	MCC hotspot resource sharing by SMDP.
Co-operative inter-cloudlet	JeanMarc Rawadi et al, 2014	On demand services by a cloudlet root server.
Impact of User mobility in cloudlet performance	Yujin Li and Wenye Wang, 2013	Access probability, execution speed for mobility pattern.
AIOLOS	Tim Verbelen et al, 2012	Middleware built on the OSGi model.
Wi-Fi access enables adhoc mobile devices.	Tim Verbelen et al, 2013	Component level, optimization on available resources.
collaborative cloudlet middleware	Steven Bohez et al, 2014	Heuristic and algorithms SD and SA.
2 tired Cloud Architecture CRAM	M. Reza Rahimi, 2012	Space-Time workflow modelling framework CRAM

adverse impact of these issues is very important in achieving the wide-scale adoption of cloudlets. In this section, we summarize and highlight several important open issues and research challenges identified through the literature review. Table 2.2 summarizes these works and related issues.

2.7.1 Mobile Application Modification

The majority of cloudlet-based mobile application execution frameworks require that the mobile application be adapted to the cloud/cloudlet architecture. This is challenging given that there are millions of applications that have been developed for mobile platforms. A better approach is to provide a virtual environment on the cloudlets that matches that of the mobile device. This solution remains less practical given the hundreds of available mobile platforms.

2.7.2 Mobile Application Partitioning

The execution of mobile applications is always based on partitioning the application and executing part of the application on the mobile device while shifting the remaining part to the cloud/cloudlet. This requires compiler support, code annotation, and expert programmers who can decide on the load that each component will introduce and its interaction with the system. Generally, this is not a straight-forward process, and a lightweight dynamic partitioning mechanism has yet to be developed.

2.7.3 Heterogeneity in Application Augmentation

For mobile application augmentation, a homogeneous platform that provides the greatest benefit in application processing must be obtained. In MCC, the different architectures and operating systems used by mobile devices are a challenging aspect of mobile application augmentation. Therefore, optimal and efficient application augmentation remains a challenge.

2.7.4 Mobile-based Augmentation and Software Installs

When cloudlets are augmented using mobile devices, the frameworks require the installation of cloudlet components onto the mobile device. In general, this is not a practical solution for two reasons. First, the mobile device only joins for a limited period of

time and may not have sufficient time to install cloudlet components and then perform valuable resource augmentation. Second, the cloudlets are free and untrusted resources. Therefore, mobile devices may be unwilling to have additional software installed from an untrusted source.

2.7.5 Resource Availability and Scalability

Resource availability and cloudlet scalability are two important considerations for mobile application augmentation. The unavailability of cloudlet resources or inadequate performance when subject to an increasing number of users can affect the service of the cloudlets. Resource unavailability can be minimized via optimal resource allocation and management policies and by identifying peak loads. However, scalability typically remains an issue for cloudlets because of inadequate funds and their self-managed nature.

2.7.6 Mobile-based Augmentation and Load Balancing

When mobile devices are used for cloudlet-based resource augmentation, current schemes do not keep track of the load shifted to the mobile devices and the service requirements of the mobile devices. In general, the mobile devices become part of the cloudlet and are treated as equal to the cloudlet computational units. Such an integration increases book keeping overhead, results in sub-optimal performance when a slower mobile device joins and often overloads resource-poor as well as high-power mobile devices.

2.7.7 Consistent and Seamless Connectivity

For MCC, mobility is an important attribute for mobile users. Mobile users enjoy the freedom of computing and communication on the move. However, a number of obstacles hinder the goals of seamless connectivity and consistency in mobile application augmentation. For example, handoffs, high-speed travelling, geographical location diversity and various environmental factors can affect such augmentation. Therefore, seamless

connectivity and uninterrupted access to centralized cloud data centres is vital to application augmentation research in MCC. As a result, ensuring the transparency of application augmentation is important. Specifically, the seamless and transparent deployment of distributed application platforms for computation-intensive applications is challenging in MCC. Clearly, the complexity should be reduced, and users should be provided with rich application execution experiences by abstracting the technical complexity of application augmentation.

2.7.8 Privacy and Integrity

Privacy and the integrity of the data as well as computations is an important issue, specifically in the case of free cloudlets and mobile-device-based resource augmentation. If a cloudlet decides to forward data of a particular user to another device in the network for computation, the privacy of the data can be compromised by the serving device. Similarly, there is no way of verifying the integrity of the computations performed by the mobile device when the code is offloaded by the cloudlet. A malicious device can easily become part of the cloudlet by offering its resources and performing malicious computations or compromising the integrity of the data. In addition, such sharing mechanisms can also result in the migration of malicious code from the networked devices to the intended victims.

2.8 Conclusion

In this chapter we presented a literature review of the state of the art of cloudlet-based mobile application augmentation in mobile cloud computing. We developed the taxonomy for cloudlet-based mobile application execution and resource augmentation for MCC. We analyzed current resource augmentation frameworks by providing a thematic taxonomy and highlighted the commonalties and differences of such frameworks in terms of various significant parameters. We categorized and highlighted a detailed analysis of

code offloading mechanisms. We presented applications of cloudlet-based augmentation and the weaknesses of the cloudlets. We discussed cloudlet resource constraints in this chapter and identified key factors and influential parameters through an extensive literature review. We considered the various approaches, frameworks, techniques and tools that are frequently used by researchers. Our research objective is to identify and establish problems that have a significant impact on MCC. Further, we will develop a new framework and justify the applicability and validity of the framework in later chapters. Finally, a number of issues were identified for future research directions. This chapter highlighted fundamental aspects of the application and use of cloudlet-based mobile applications for MCC. We highlighted challenges to the current deployment of cloudlet applications and issues concerning the optimal and easy use of cloudlet-based augmentation frameworks for MCC.

In a subsequent chapter, we further investigate the highlighted issues, especially the resource scarcity problem, which addresses the impact of finite resources on cloudlet performance. Finally, with reference to the resource augmentation of cloudlets using mobile devices, we will present a solution that addresses the highlighted issues.

CHAPTER 3 : RESOURCE SCARCITY IN CLOUDLETS: PROBLEM ANALYSIS

In this chapter, we attempt to highlight the resource scarcity problem and establish the impact of resource scarcity on cloudlet performance via empirical analysis. Moreover, we extend our analysis by modeling cloudlet user arrival and serving scenarios and establish the research problem by analyzing the observed output. To this end, we conduct a series of experiments, and based on the results of the experiments, we show that the problem of resource scarcity is real and non-trivial. Subsequently, we develop a preliminary mathematical model for the problem and derive the basic relationship between the number of resources available in the system and the number of users that can be served simultaneously.

Section 3.1 presents several definitions and the initial assumptions. Sections 3.2 and 3.3 describe the empirical investigations using details of the experimental setup and model of the experiment. Section 3.4 presents the results of the empirical analysis. The performed cloudlet operational system behaviour synthesis is explained in Section 3.5. Based on these findings, we develop and discuss a preliminary mathematical model of the problem in Section 3.5. The chapter is concluded in Section 3.6.

3.1 System Description

In this section, we present the details of the operation of cloudlet systems by defining several system definitions and assumptions. This will help us better understand the factors that affect the response of the cloudlet to user applications (Whaiduzzaman, Gani, & Naveed, 2015)

3.1.1 System Parameter Definitions

We first define several relevant terms that are used frequently in this chapter and in the remainder of the thesis. The definitions for the analysis are as follows:

Mobile Device: A mobile device is a smart phone, tablet, notebook or any other portable device that can connect to a nearby network or the Internet through Wi-Fi or cellular network connections and that is capable of requesting and receiving required services from the computational cloud.

Mobile Service: A mobile service is initially started using a mobile user request through the device for a specific purpose by maintaining a work flow service. Accordingly, for every mobile service request, the tasks are executed by processing the input and finally providing feedback with an output.

Cloudlet Service: A cloudlet having finite resources offers services to mobile users in close proximity to serve computational user service requests and other task execution requests using its greater resources compared to the mobile device.

Resource: A resource represents an available unit that is required for executing a task. We can denote r as a resource, and R can be denoted as a set of available resources.

CPU Utilization or CPU Use Time: CPU usage, or CPU time, is the amount of time for which a central processing unit (CPU) is used for processing the instructions of a computer program.

Task: A task can be a logical unit of work that is executed by a resource.

Cloudlet Execution Time: The execution time is the total time needed to complete a task by the available resources.

Cloudlet Response Time The cloudlet response time is the total time taken for the execution of the task by the cloudlet, including the waiting time in the system.

User service time: The user service time indicates the total time taken by the

cloudlet to deliver the computation service to the mobile user and other related transfer times and network delays. Therefore, the user service time includes cloudlet response, code transfer, and network delay.

User Time Benefit: The user time benefit is the ratio of the mobile device execution time to the total cloudlet execution time. The user time benefit depends on mobile device resources, particularly the CPU speed, the cloudlet CPU speed, the code transfer time and other overhead. This benefit can be expressed as follows: user time benefit = Time taken by mobile device/Total user service time for cloudlet X100

Throughput: Throughput is the number of tasks completed by the cloudlet per unit of time. This metric depends on several factors that can affect the execution of a task.

Efficiency: The cloud system efficiency indicates the effective utilization of cloudlet resources and services. Therefore, a higher value of the efficiency indicates that the overhead will be smaller.

3.1.2 CPU Resources, Queue and User Arrival Scenario

A cloudlet consists of a set of CPU resources associated with one or more queues. When the user application is submitted to the computational system, the application is initialized and added to the queue. In general, the queue does not use priorities, and the user applications are served in a round robin manner. Every application is served for a fixed period of time or shorter when it is assigned to a CPU resource. The component of the cloudlet that assigns the applications to the CPU cores is known as a process manager, and it is part of the kernel of the operating system.

The applications join the queue in a process based on three events. First, when the application is initialized, it is added to the queue. Second, after the amount of execution time on the designated CPU is used, the application is added back to the queue. Finally, if the application was in sleep mode or waiting for a specific event, upon completion of

the event, the application is triggered and rejoins the queue. The applications leave the queue only when the process control decides to assign the application to a CPU core for execution (Shiraz et al., 2014).

At the CPU level, The process manager is the process that periodically executes computations on each CPU. The process manager decides on the next process to be executed. It loads the process state into memory and sets the CPU in such a way that the process can execute a given number of instructions at most. Subsequently, control is handed over to the process, which executes its own instructions. After the execution of the given number of instructions, the control is handed back to the process control, which repeats the entire setup for the next process in the queue. Consequently, there is a constant overhead resulting from process control execution.

For the purpose of this analysis, we assume that the applications are computationally intensive and do not wait on hardware resources (Whaiduzzaman, Gani, & Naveed, 2014). Therefore, the two items of interest inside the cloudlet system are (i) the wait time until a CPU becomes available when another process or process controller is being executed and (ii) the actual instruction execution time. Throughout the remainder of this thesis, we refer to the CPU wait time as the wait time and the instruction execution time as the cloudlet service time. The sum of the two times is referred to as the cloudlet response time.

3.1.3 Assumptions

In our experiments, we consider a cloudlet that has various resources such as CPU, memory, and disk space. We mainly focus on computation-intensive applications in this research. Therefore, we mainly focus on computational resources, especially cloudlet CPU resources, thereby primarily addressing computation-intensive applications.

For our investigation and experiment, we consider the following assumptions to de-

fine our research scope and simplify the problem to analyze the cloudlet system.

1. We consider computation-intensive applications that do not require access to hardware resources. Therefore, the wait time of the applications for resources is zero. This is a simplifying assumption. In practice, the wait time can simply be added to the application time.

2. Cloudlet resources are all single VMs. Users are allocated resources within the VM.

3. The cloudlet resources are all of the same type, i.e., they are homogeneous. For example, all the CPU cores have the same configuration and are of the type and make.

4. The user application size is the same, and we repeatedly use the same application to ensure that the same effects are produced.

5. For the mobile device execution time, no other applications are running on the mobile device.

6. All the mobile devices are homogeneous; specifically, they are all of the same make and model and have the same specifications.

7. A mobile user is allowed to submit only one task to the cloudlet.

8. The processing speed of the cloudlet is higher than the mobile device.

3.2 Experimental Setup

In our experiments, we consider a cloudlet that has resources such as CPU, memory, and disk space. For the cloudlet, we considered that any user can join, but storing data in the cloudlet and returning to use the stored data is not possible. This is because the cloudlets can only maintain soft states. Maintaining hard states requires substantial

storage resource, and most of the stored data will never be used again. This is because users do not repeatedly join the same cloudlet. If they do, the stored state is quite old and no longer valuable to the user. Therefore, we do not consider the disk space of the cloudlet as a resource for our experiments, and we mainly focus on computational resources, with the primary focus on cloudlet CPUs. Thus, our research primarily address computation-intensive applications.

3.2.1 Experimental Model

In this analysis, the two test bed modes are named as the local mobile execution mode and the remote execution mode specifically in the cloudlet execution mode. We test our computation-intensive prototype application using these two modes. In local execution mode, all the application components are executed locally on the mobile device, whereas in the cloudlet, the computation-intensive prototype mobile application is executed. In our experiments, we deployed a small-scale Openstack-based cloud as the cloudlet. Our experimental model and its components are described below.

3.2.1.1 Cloudlet Resources

We create a 4-core virtual machine for the mobile device. Note that in general, a mobile device is offered less than 4 cores because of the limited resources available in the cloudlet. The mobile device is allocated 8 GB of RAM and 80 GB of hard disk space. The processor is a 2.4 GHz Xeon processor, which has a processing speed of 32000 MIPS. The virtual machine is running Ubuntu linux version 12.0.4 LTS as the operating system. The cloudlet specifications are summarized in Table 3.1

3.2.1.2 Mobile Device

The mobile device used in this experiment is a Samsung Galaxy S2 GT-19100G, featuring a dual-core 1.2 GHz Cortex-A9 processor; 512 MB of RAM; 802.11 a/b/g/n

Table 3.1: Technical Specifications of Cloudlet Resources Used in Empirical Analysis

Cloudlet Platform	**Open Stack**
Version	Havana
Original OS	Linux UBUNTU 12.04
Processor	2.4 GHz Xeon
MIPS	32000MIPS
VCP	4 core
RAM	8GB
Storage	80GB
OS	Cirros3.03

Wi-Fi; Exynos 4210 chipset; Mali-400 GPU; accelerometer, gyro, and proximity sensors; a compass; GPS; and Android v2.3.4. The processing capacity of the mobile device is 7500 MIPS. This results in a greater than 4 times speedup factor when the mobile application is shifted from the mobile device to the cloudlet. Details of the mobile device are summarized in Table 3.2.

Table 3.2: Technical Specifications of Mobile Device Used in Experiment

Mobile device	**Samsung Galaxy S2**
Series	GT I-9100
OS	Android
OS Version	V2.3.4
Processor	Dual core 1.2 GHz Cortex-A9
RAM	512 MB
Chipset	Exynos-4210
Storage	16GB
Wi-Fi	802.11 a/b/g/n

3.2.1.3 Connectivity

The connectivity between the mobile device and the cloudlet is established by attaching a wireless router to the cloudlet network. We use a Cisco Linksys WRT45GL wireless router with the 4.30.16 build 6 version of the firmware to enable wireless connectivity. The wireless link speed is 54 Mbps, and indoor communications occur via the 2.4 GHZ band. The router is connected to a high-speed LAN network to communicate with the Internet.

3.2.1.4 Prototype Application

The prototype application is a computation-intensive math application that receives values and performs several addition and multiplication operations. The prototype application is developed in the C programming language. For local execution on the mobile device, we first rooted the smartphone to obtain full control as an administrator or super user. The program was compiled on a laptop using the ARM linux GCC compiler version 4.8.3. The workload of the program instruction size is $120x10^9$ instructions, with the program taking approximately 210 seconds to execute on the mobile device. The program has been compiled statically, and the size of the executable is 8.47 KB. To embed the library file to run and execute the application, we compiled the application as a static code by differentiating the architecture as an arm processor. The same application has been

Table 3.3: Performance Metrics and Units Used in Empirical Analysis

Performance Metrics	**Units**
CPU Utilization	% of use
User service time	Seconds
Throughput	
Efficiency	% of use
Response time	Seconds
Waiting time	Seconds

used for all mobile devices to ensure that the impact of the variable workload is mitigated. In addition, we performed an alternate set of experiments with variable application sizes. The results are as expected irrespective of the application size and diversity in the size of the applications.

3.2.1.5 Performance Metrics

We use several performance metrics, namely user service time, throughput, CPU utilization and efficiency, for our experiment, as previously described. The metrics are summarized in Table 3.3.

Linux console commands have been used to measure the performance metrics that were presented in the previous section. We used the *Top* command to extract the values of the CPU load and the number of users on the system. The time command was used to measure the time taken by the applications *Awk* and *Sed* scripts were used to extract the values from the raw output files.

3.3 Empirical Analysis of Cloudlet Finite Resources Impact

In this section, we present our empirical findings with reference to the impact of cloudlet resources on the performance of cloudlet in delivering the service to the mobile users.We first present the CPU utilization. This is followed by the cloudlet efficiency, cloudlet throughput and finally the cloudlet response time.

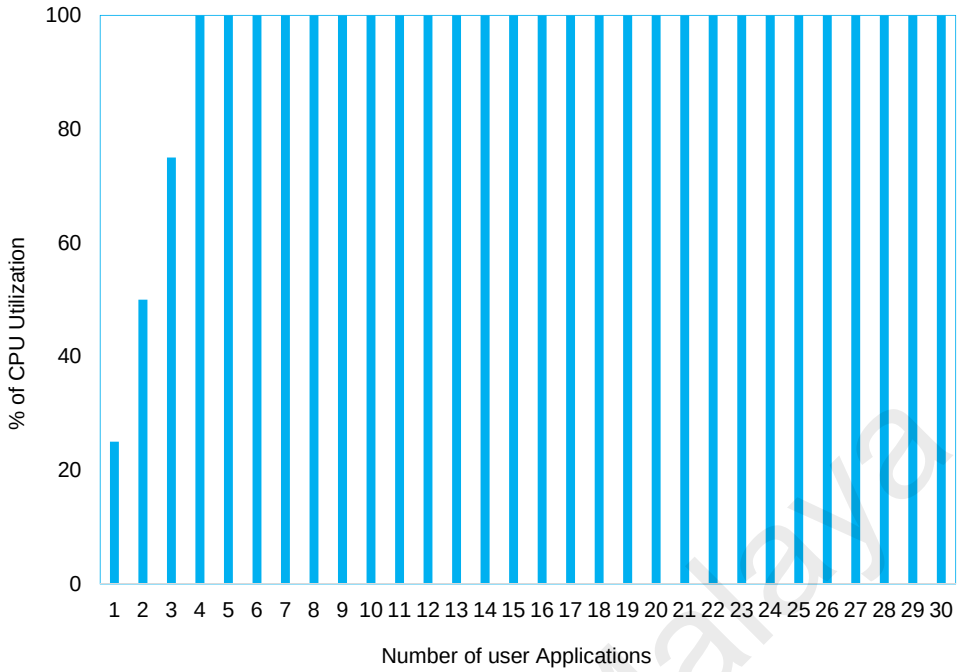

Figure 3.1: CPU Utilization

3.3.1 CPU Utilization

The CPU Utilization, or CPU usage time, is the amount of time for which a central processing unit (CPU) was used for processing instructions of a computer program. In general, over an extended period of time, the average CPU utilization is low because of I/O operations. However, the instantaneous utilization can reach 100%. In this experiment, we increase the number of mobile applications being executed on the cloudlet and observe the impact on the cloudlet CPU utilization as the instructions are being executed. The results are presented in Figure 3.1

Note that the cloudlet CPU utilization increases as long as the number of user applications is less than the number of CPUs available on the cloudlet. This is because a newly submitted application can be executed on an idle CPU. When the idle CPU is brought into use, the overall utilisation of the CPUs on the cloudlet increases. However, when the number of applications on the cloudlet increases beyond the number of installed CPU

cores, the cloudlet CPU utilization reaches 100% and remains there.

A utilization of 100% for only four computation-intensive mobile applications is alarming. However, the exact impact of increasing the number of applications beyond the number of CPU cores is not obvious from this graph. To this end, we consider cloudlet efficiency as our next metric.

3.3.2 Efficiency

For the cloudlet, efficiency is defined as the ratio of the service time of the task to the total cloudlet response time. This is given in Equation 3.1, where T_s, T_o and T_w represent the cloudlet service time, operating system overhead and the cloudlet wait time, respectively.

$$Efficiency = \frac{T_s}{(T_s + T_o + T_w)} * 100 \tag{3.1}$$

Equation 3.1 shows that as the overhead increases, the system efficiency decreases. The results for different numbers of user applications being executed on the cloudlet are shown in Table 3.4. The results for efficiency are shown in Figure 3.2. We can see that the system efficiency is nearly 100% when the number of users is less than the number of CPU cores. However, as the number of mobile applications being executed by the cloudlet increases further, the efficiency starts decreasing. The initial uniform value is because each CPU is handling only one task. This leads to the negligible overhead of the operating system only. On the other hand, as the number of users increases, the processes start sharing the CPUs, thus resulting in an increased wait time with increasing number of applications. Therefore, the efficiency decreases exponentially with increasing number of applications.

The decrease in efficiency indicates that the performance of the cloudlet decreases with increasing number of users and that the cloudlet needs additional resources to cater to the higher number of users. However, such resources are not available, which in turn

Table 3.4: Cloudlet Systems Parameters and Efficiency

No.	T	$T_{o\text{-}vm}$	$T_{o\text{-}os}$	T_o	$T\text{-}T_o$	T_w	Eff
1	54.65	0.02	0.98	1.00	53.65	0.00	98.17
2	55.71	0.11	0.99	1.11	54.61	0.96	96.30
3	56.52	0.30	0.95	1.25	55.27	1.62	94.92
4	56.75	0.34	0.93	1.27	55.49	1.84	94.54
5	70.72	0.38	1.08	1.46	69.26	15.61	75.86
6	84.35	0.42	1.40	1.82	82.53	28.88	63.60
7	98.25	0.58	1.54	2.12	96.13	42.48	54.61
8	113.62	0.32	2.18	2.50	111.12	57.47	47.22
9	127.55	1.02	2.07	3.09	124.46	70.81	42.06
10	141.79	0.70	2.30	2.99	138.80	85.15	37.84
11	155.20	1.32	2.51	3.83	151.37	97.72	34.57
12	170.15	1.06	2.76	3.81	166.34	112.69	31.53
13	184.57	1.35	3.30	4.65	179.92	126.27	29.07
14	198.11	1.19	3.19	4.38	193.73	140.08	27.08
15	212.95	1.62	3.37	4.98	207.97	154.32	25.19
16	226.15	1.74	3.55	5.29	220.86	167.21	23.72
17	240.26	1.81	3.78	5.59	234.66	181.02	22.32
18	255.19	1.89	3.87	5.76	249.43	195.78	21.02
19	269.46	1.92	3.95	5.87	263.59	209.94	19.91
20	284.54	1.99	4.12	6.11	278.43	224.78	18.85
21	299.28	2.12	4.27	6.39	292.89	239.24	17.92
22	314.61	2.46	4.39	6.85	307.76	254.11	17.05
23	329.67	2.49	4.45	6.94	322.73	269.09	16.27
24	345.75	2.89	4.59	7.48	338.27	284.62	15.51
25	361.38	2.99	4.75	7.74	353.63	299.98	14.85
26	377.73	3.17	4.89	8.069	369.66	316.01	14.20
27	394.39	3.28	4.95	8.23	386.15	332.51	13.60
28	411.33	3.49	5.35	8.84	402.49	348.84	13.04
29	428.31	3.79	5.79	9.58	418.73	365.08	12.52
30	444.34	4.56	6.99	11.55	432.79	379.14	12.07

affects the service that the users receive in terms of service time. This is further confirmed by examining the cloudlet throughput and service time.

3.3.3 Throughput

The cloudlet system throughput is a ratio defined as the number of tasks completed over a given period of time. The can be represented by Equation 3.2, where N is the total number of tasks completed during the experiment time.

$$Throughput = \frac{N}{T_{exp}} \tag{3.2}$$

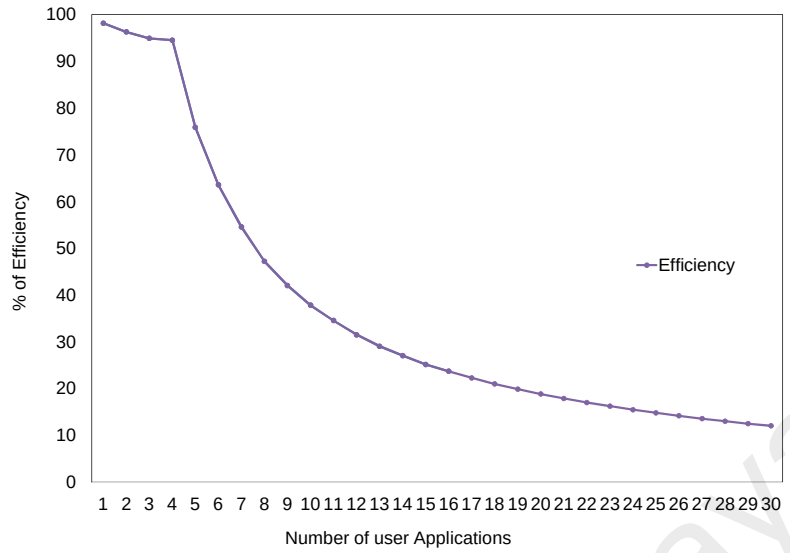

Figure 3.2: Cloudlet Efficiency

Figure 3.3 shows that the throughput initially increases as the number of tasks increases up to a maximum value when the number of user applications being executed on the cloudlet is equal to the number of CPU cores. Subsequently, the throughput remains constant and continues to constant as the number of tasks in the system increase.

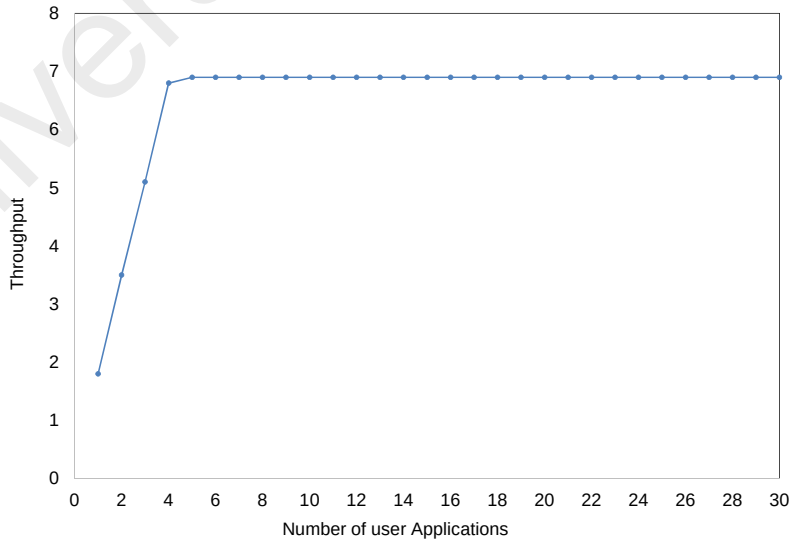

Figure 3.3: Cloudlet Throughput with Increasing User Task Loads

This suggests that the cloudlet task will not vacate earlier that means wait time will

Table 3.5: Cloudlet Task Time with Increasing Number of Users in Cloudlet

No.	T	SD	E	CI
UserAps1	54.65	0.1678	0.07906	54.65(+/-)0.07906
UserAps2	55.71	0.1926	0.09074	55.71(+/-)0.09074
UserAps3	56.52	0.1520	0.07162	56.52(+/-)0.07162
UserAps4	56.75	0.1511	0.07118	56.75(+/-)0.07118
UserAps5	70.72	0.1942	0.09149	70.72(+/-)0.09149
UserAps6	84.35	0.1366	0.06434	84.35(+/-)0.06434
UserAps7	98.25	0.1380	0.06502	98.25(+/-)0.06502
UserAps8	113.62	0.1776	0.08366	113.62(+/-)0.08366
UserAps9	127.59	0.1476	0.06955	127.59(+/-)0.06955
UserAps10	141.79	0.1733	0.08161	141.79(+/-)0.08161
UserAps11	155.21	0.1788	0.08423	155.21(+/-)0.08423
UserAps12	170.15	0.2542	0.11972	170.15(+/-)0.11972
UserAps13	184.57	0.2049	0.09650	184.57(+/-)0.09650
UserAps14	198.11	0.2514	0.11841	198.11(+/-)0.11841
UserAps15	212.96	0.2483	0.11696	212.96(+/-)0.11696
UserAps16	226.15	0.2202	0.10375	226.15(+/-)0.10375
UserAps17	240.26	0.2218	0.10449	240.26(+/-)0.10449
UserAps18	255.19	0.4265	0.20091	255.19(+/-)0.20091
UserAps19	269.46	0.1900	0.08950	269.46(+/-)0.08950
UserAps20	284.54	0.3060	0.14412	284.54(+/-)0.14412
UserAps21	299.28	0.3300	0.15543	299.28(+/-)0.15543
UserAps22	314.61	0.1854	0.08733	314.61(+/-)0.08733
UserAps23	329.67	0.2602	0.12256	329.67(+/-)0.12256
UserAps24	345.75	0.2629	0.12385	345.75(+/-)0.12385
UserAps25	361.38	0.1831	0.08625	361.38(+/-)0.08625
UserAps26	377.73	0.2377	0.11198	377.73(+/-)0.11198
UserAps27	394.39	0.2217	0.10441	394.39(+/-)0.10441
UserAps28	411.33	0.2738	0.12896	411.33(+/-)0.12896
UserAps29	428.31	0.2552	0.12022	428.31(+/-)0.12022
UserAps30	444.34	0.2812	0.13245	444.34(+/-)0.13245

increase, ultimately increase the cloudlet response time with the number of users or tasks entered into the cloudlet system.

3.3.4 Task Completion Time

We finally consider the task completion time to observe the impact of limited cloudlet resources and increasing number of user applications on the service provided by the cloudlet. Table 3.5 presents the data concerning the total task time for 20 different executions with the results averaged.

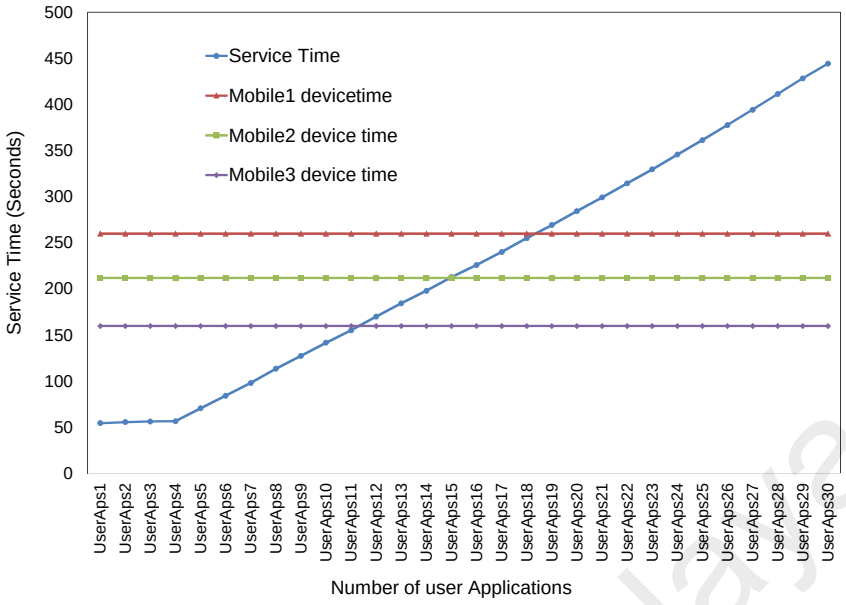

Figure 3.4: Application Execution Time in Cloudlet and Different Mobile Devices

Figure 3.4 graphically represents the service time with increasing number of users. For comparison, the time taken by the mobile device to execute the same application locally has also been plotted as a straight line.

Note that the task completion time linearly increases, and for 17 users, the task completion time from the cloudlet increases to being greater than the time taken by the mobile device itself. The increase in task completion time is mainly caused by the increased waiting time, which is directly proportional to the number of users on the cloudlet.

3.3.5 User Time Benefit

The scarcity of cloudlet resources can be best captured by considering the user time benefit for offloading the mobile device's computational task to the cloudlet. Table 3.6 shows the data concerning the user benefit with increasing number of applications being executed by the cloudlet. The data are presented graphically in Figure 3.5. The benefit to the user decreases with increasing number of users and diminishes when the number of applications reaches 16.

Table 3.6: User Time Benefit with Different Tasks Load.

No	**T**	**T-T_o**	**T_{Mob}**	**benefit**
1	54.65	53.65	212	3.88
2	55.71	54.60	212	3.80
3	56.52	55.27	212	3.75
4	56.75	55.49	212	3.74
5	70.72	69.26	212	2.99
6	84.35	82.53	212	2.51
7	98.25	96.13	212	2.16
8	113.62	111.12	212	1.87
9	127.55	124.46	212	1.66
10	141.79	138.80	212	1.49
11	155.2	151.37	212	1.37
12	170.15	166.34	212	1.25
13	184.57	179.92	212	1.15
14	198.11	193.73	212	1.07
15	212.95	207.97	212	0.99
16	226.15	220.86	212	0.94
17	240.26	234.67	212	0.88
18	255.19	249.42	212	0.83
19	269.46	263.59	212	0.79
20	284.54	278.43	212	0.75
21	299.28	292.89	212	0.71
22	314.61	307.75	212	0.67
23	329.67	322.73	212	0.64
24	345.75	338.27	212	0.61
25	361.39	353.64	212	0.59
26	377.73	369.67	212	0.56
27	394.39	386.15	212	0.54
28	411.33	402.49	212	0.52
29	428.31	418.72	212	0.49
30	444.34	432.79	212	0.48

The results show that the cloudlet resources are not sufficient to handle a substantial number of work loads. For the cloudlet that we created for testing, barely 16 user applications being executed by the cloudlet system can result in no time benefit for the mobile device, which is a major factor when deciding to use the cloudlet. In other words, the service of the cloudlet becomes unacceptable as the number of users increases. This is mainly because of the limited resources of the cloudlets.

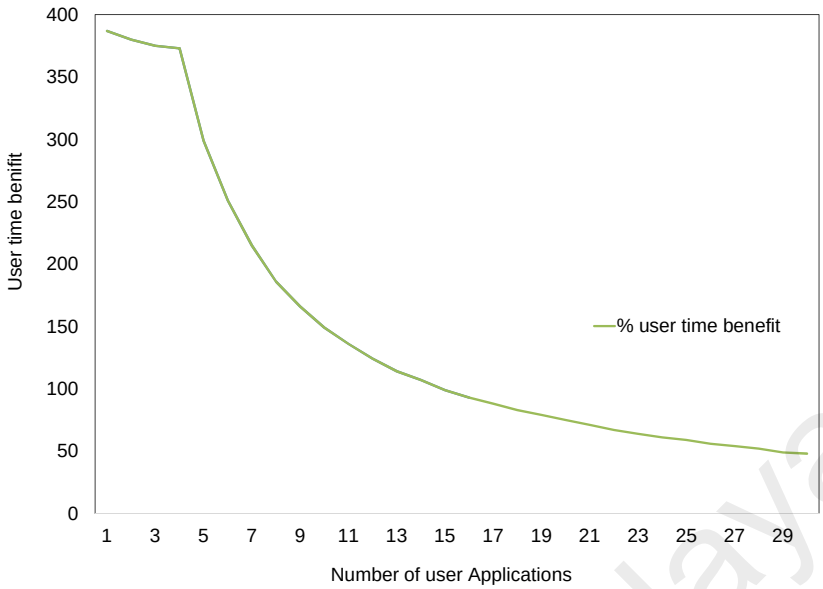

Figure 3.5: Cloudlet Task Loads and User Time Benefit

3.4 Cloudlet System Behavior Synthesis

The cloudlet system behaviour scenarios are studied by varying influential factors such as resources, access rate, and service time. This allows us to closely observe, thoroughly study and analyze the cloudlet system performance from various aspects and from a technical perspective.

In this section, we closely analyze the cloudlet system behavior. We discuss several aspects of the cloudlet system behavior, including user arrival rate (Lambda, λ), user service time, cloudlet response time, queue length and waiting stage of the cloudlet system as a function of the number of cloudlet parameters from a broader perspective and expand the analysis by increasing the amount of resources. We report and analyse from different angels and perspectives and critically analyse the overall system behaviors in two main categories: First, we observe and report on the resource enhancement, specially the CPU core effect. Second, we observe and report on the overall effects if we allow more users with fixed resource impacts on the overall performance of the cloudlet as follows:

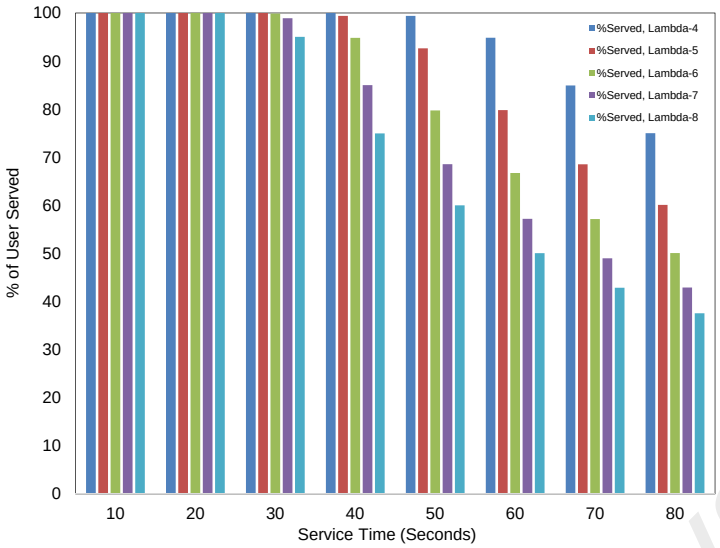

Figure 3.6: Number of User Served in Cloudlet at CPU4

3.4.1 Resource Enhancement Effects:

In this section, we investigate and analyse the impact on cloudlet behaviour with varying resources.

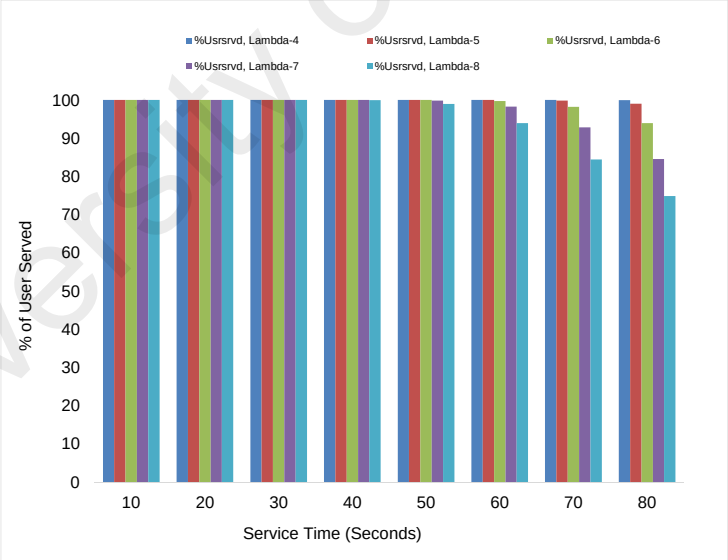

Figure 3.7: Number of User Served in Cloudlet at CPU8

3.4.1.1 Number of User Served:

In this experiment, we observed the number of users served by the cloudlet. Figure 3.6 shows that the number of users served is initially 100% and remains at this level up to a service time of 20. From a service time of 30, an arrival rate, lambda, of 8 results in

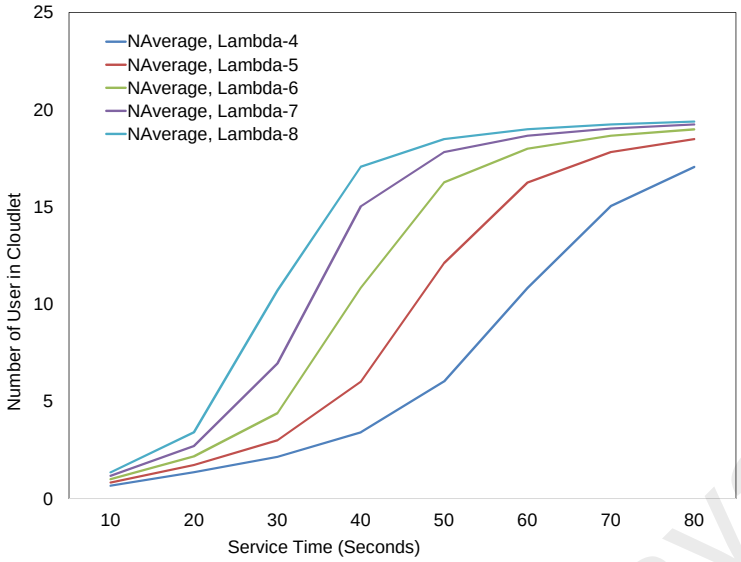

Figure 3.8: Number of User in Cloudlet at CPU4

users being dropped. From a service time of 60, the cloudlet starts dropping users for any arrival pattern and continues providing services. Figure 3.7 shows the improvement to the number of users served up to a service time of 60. Therefore, increasing the resources strongly affects the number of users served by the cloudlet. However, for the cloudlet, we do not consider the ability to add any extra resources to the cloudlet during operation.

3.4.1.2 Number of Users in the System

Figure 3.8 shows the number of users on the cloudlet system against the service time. The number of users reaches a maximum of 20 very quickly when the user arrival rate is higher, and this occurs less quickly when a low arrival rate is used.

In Figure 3.9, we observe the same effect as in Figure 3.8 by increasing the CPU resources from 4 to 8 cores. The graph indicates that increasing resources can allow users to be served even with linearly increasing arrival rates. The trends show that a fully linear relationship will result if further resources are provided to the cloudlet.

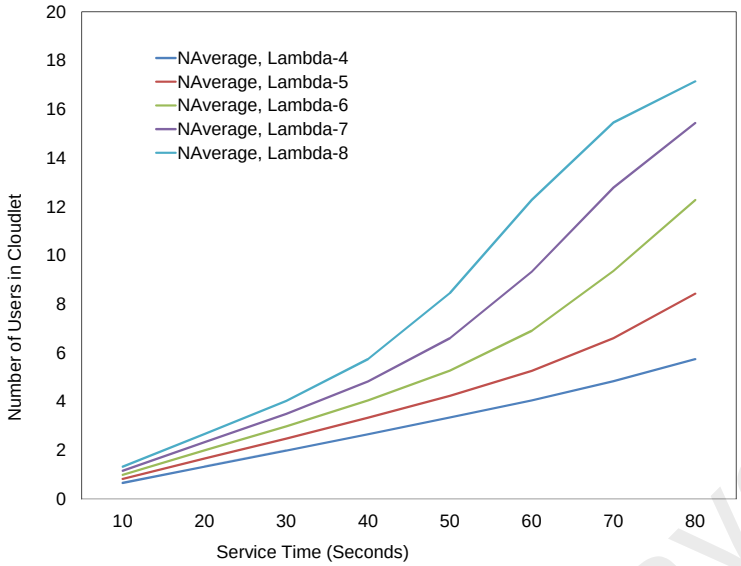

Figure 3.9: Number of User in Cloudlet at CPU8

3.4.1.3 Number of Users Waiting in the System

In Figures 3.10 and 3.11, we can observe the number of waiting users in the system when varying the number of CPU cores from 4 to 8. Figure 3.10 shows no waiting from a service time of 10 to a service time of 20. However, Figure 3.11 shows that the added CPU resources impact the cloudlet by resulting in a minimal waiting time up to a service time of 40.

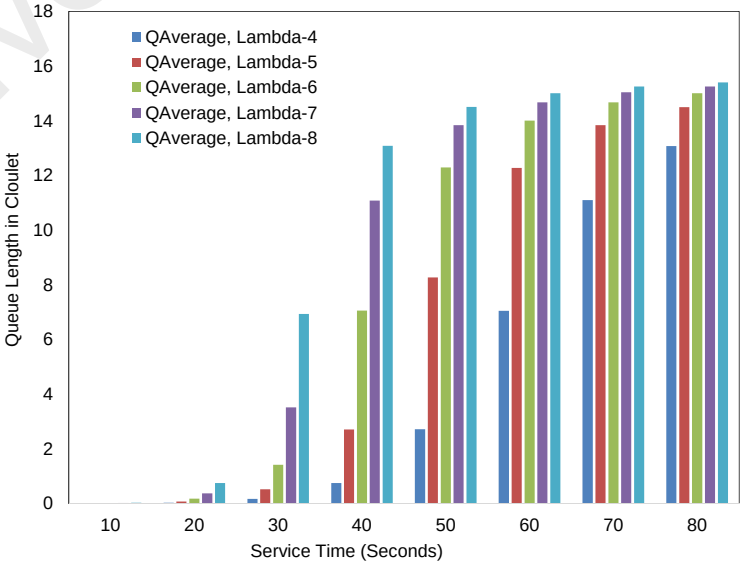

Figure 3.10: Number of Waiting User in Cloudlet Queue at CPU4

3.4.1.4 *Uses of CPU Resource*

Cloudlet CPU utilization is an important indicator of the resource scarcity of cloudlets. Figures 3.12 and 3.13 show the CPU utilization for 4 cores and 8 cores, respectively.

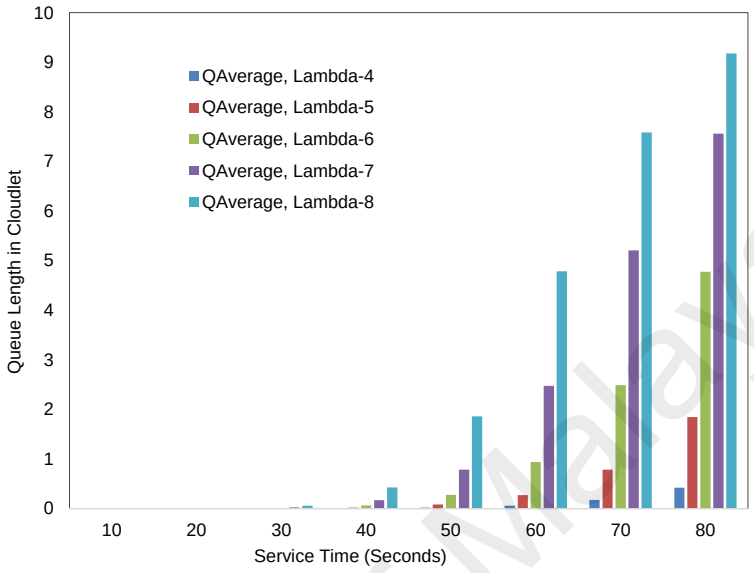

Figure 3.11: Number of Waiting User in Cloudlet Queue at CPU8

With fewer CPU cores, 100% utilization is quickly reached; on the other hand, Figure 3.13 shows that 100% utilisation is achieved later due to the increased resources. In addition, the higher arrival rate obviously reaches its peak earlier.

3.4.1.5 *Response Time of Cloudlet*

Figures 3.14 and 3.15 show the cloudlet response time with 4 and 8 CPU cores. Initially, the response time is the same irrespective of the arrival rate. However, the response time changes with increasing service time.

3.4.1.6 *Waiting Time in the Cloudlet*

Figures 3.16 and 3.17 show the waiting effect of the cloudlet with increasing number of CPU cores. For 4 cores, the waiting time increases almost linearly irrespective of the arrival rate. However, for 8 CPU cores, no waiting time is experienced up to a service time of 40; then, the waiting time starts increasing with the service time. When the number of

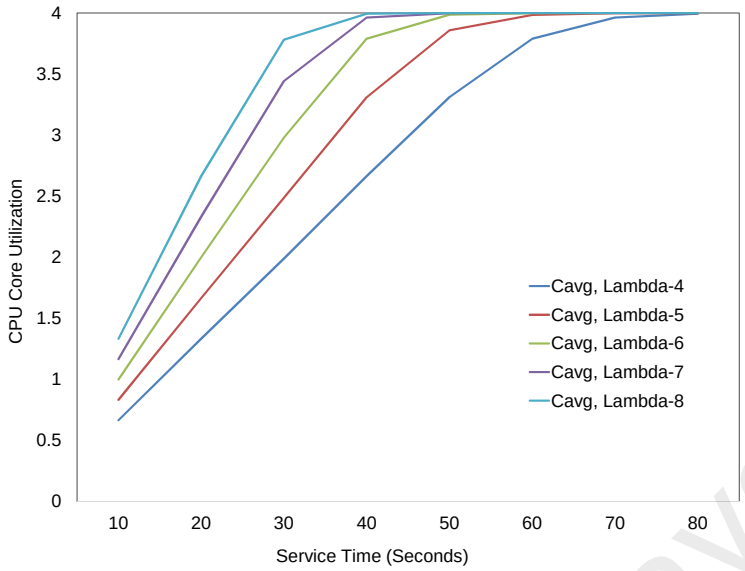

Figure 3.12: Uses of CPU Resources in Cloudlet at CPU4

CPU cores increases and the overall response time decreases, the cloudlet can work faster as a result of the increased resources.

For all the aforementioned cases, we observe and discuss the effects on the cloudlet when increasing and decreasing the number of CPU cores. In the following section, we will closely observe the cloudlet system behavior based on user interaction by fixing the resources, but we will allow more users to be served by the cloudlet at any given time.

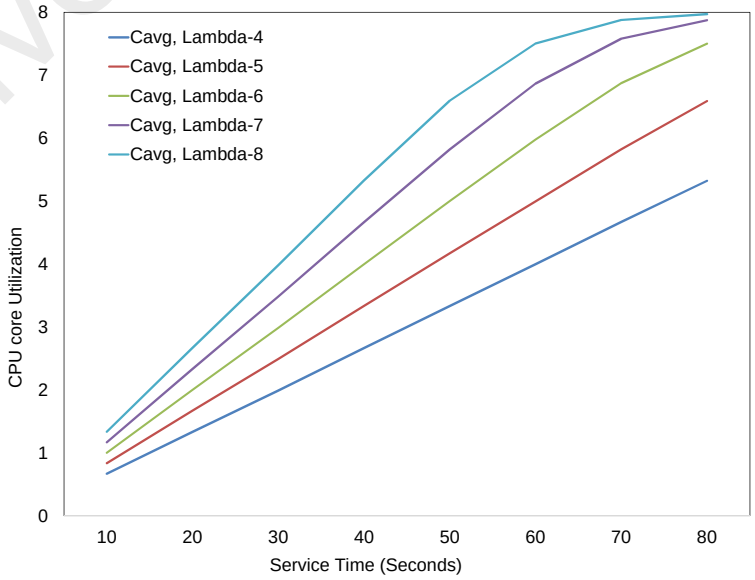

Figure 3.13: Uses of CPU Resources in cloudlet at CPU8

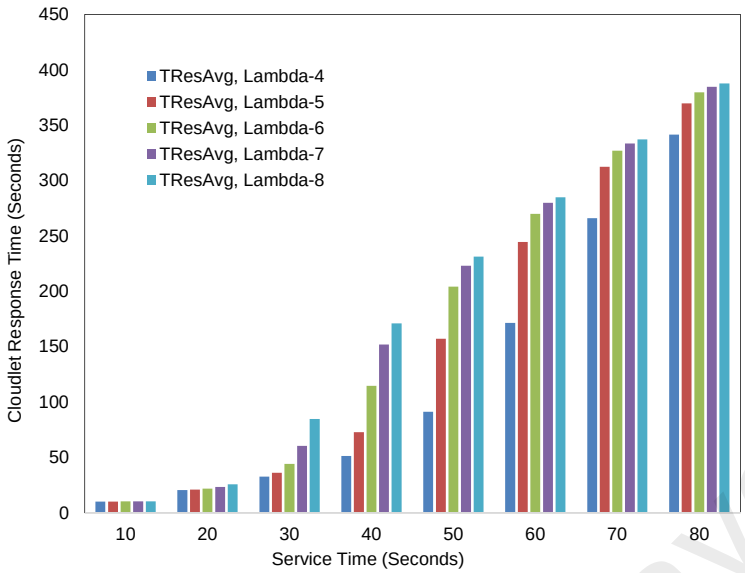

Figure 3.14: Cloudlet Response Time at CPU4

3.4.2 Maximum Number of User Enhancement: K-effect

We observed the results of varying the maximum number of allowed users K and the corresponding effects on the cloudlet. For this experiment, we set CPU=4, User application requirement=60 seconds, and service time=60 seconds. We varied the maximum number of allowed users on the cloudlet in the order of 20, 30 and 40 to observe the relevant effects. We obtained the quite interesting findings described in the relevant graphs.

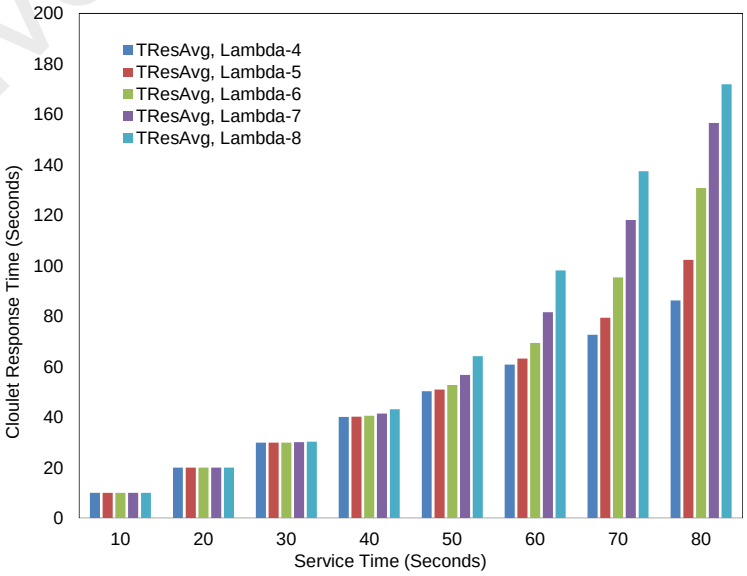

Figure 3.15: Cloudlet Response Time at CPU8

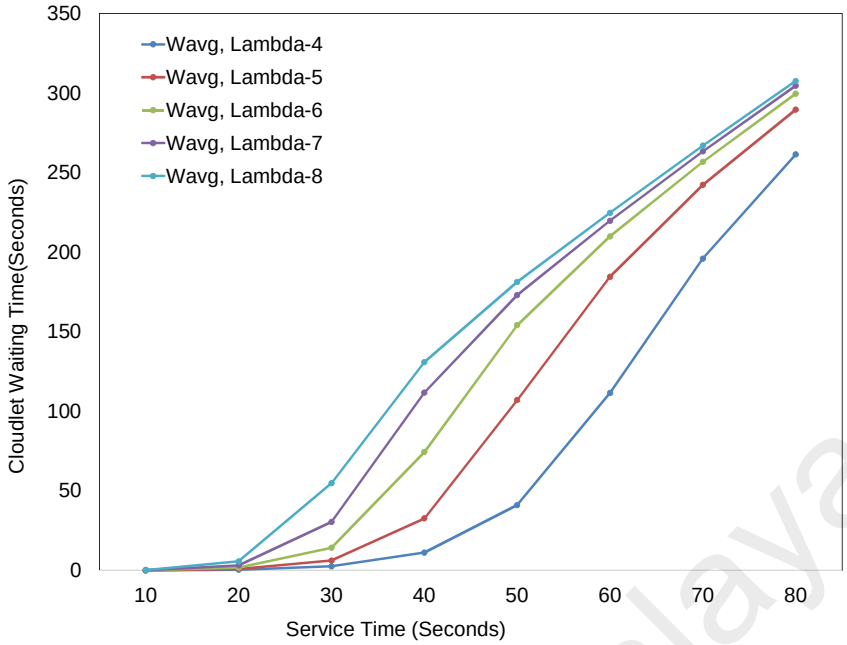

Figure 3.16: Cloudlet Waiting Time at CPU4

3.4.2.1 Number of Users in the Cloudlet with Varying Lambda and K

In this observation, we concentrate on the number of users in the cloudlet with varying the user arrival rate and the maximum number allowed user.

From Figure 3.18, shows the arrival rate and the maximum allowed number of users

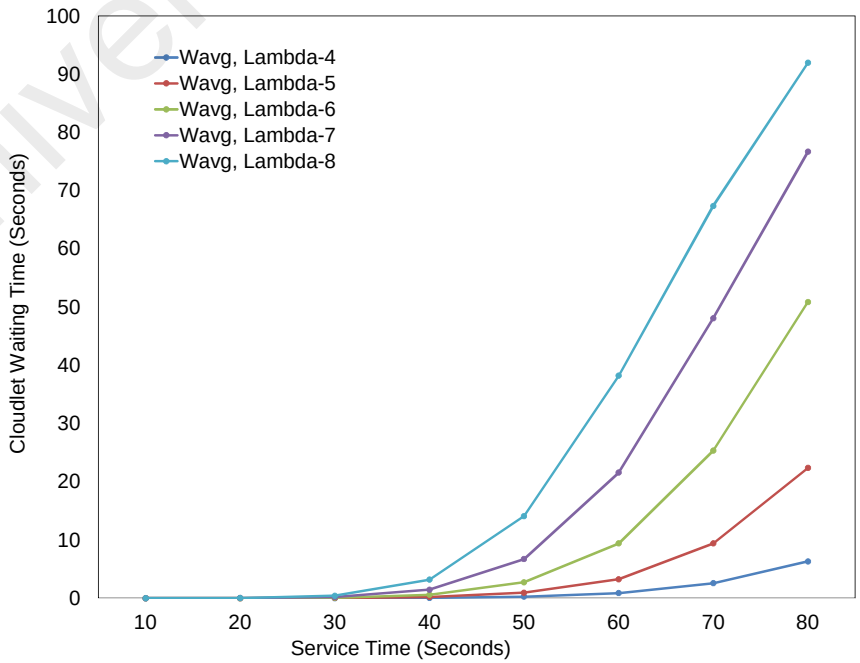

Figure 3.17: Cloudlet Waiting Time at CPU8

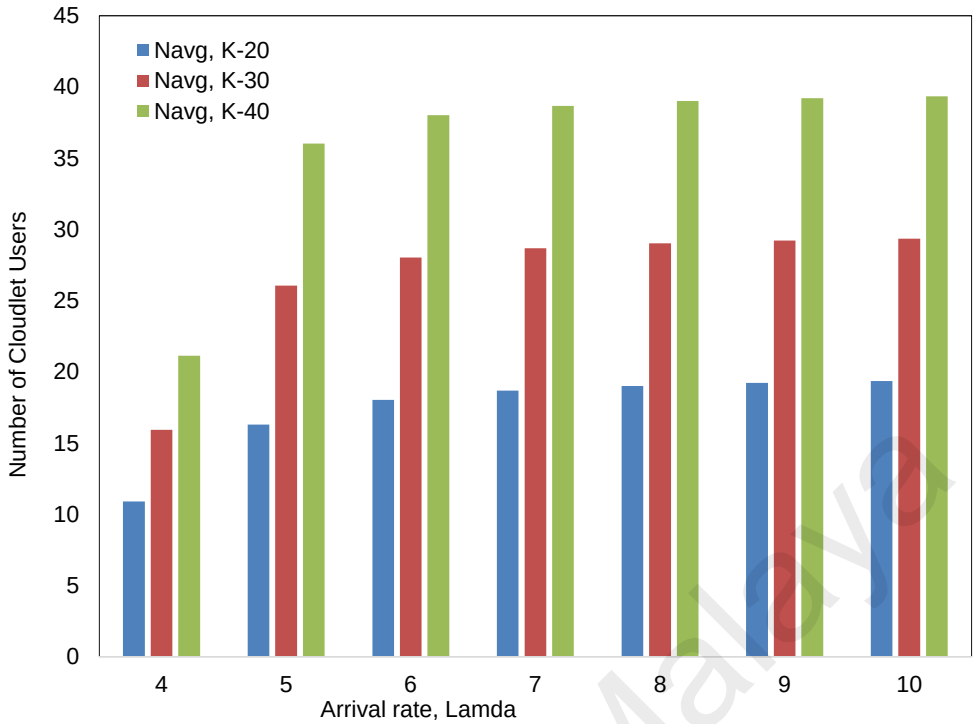

Figure 3.18: Cloudlet Users Effect Varying the Maximum Allowed User in the Cloudlet

in the cloudlet system. we observe that for lower arrival rates, the maximum number of users is slowly reached. However, for increased arrival rates, the maximum number of users in the system for higher K is quickly reached. The case of K=40 saturated the cloudlet earlier than did the cases of K=30 and K=20.

3.4.2.2 Resource CPU Utilization Uses by Varying Lambda and K

In Figure 3.19, shows the resource CPU utilization uses by varying lambda and the maximum allowed user. We observe the effect of varying K with the number of CPUs fixed at 4. When varying the maximum number of allowed users on the system, we observe that the CPUs are quickly used in an optimal manner when a higher arrival rate is used. For an arrival rate of 4, K=40 results in a higher CPU utilization compared to K30 and K20.

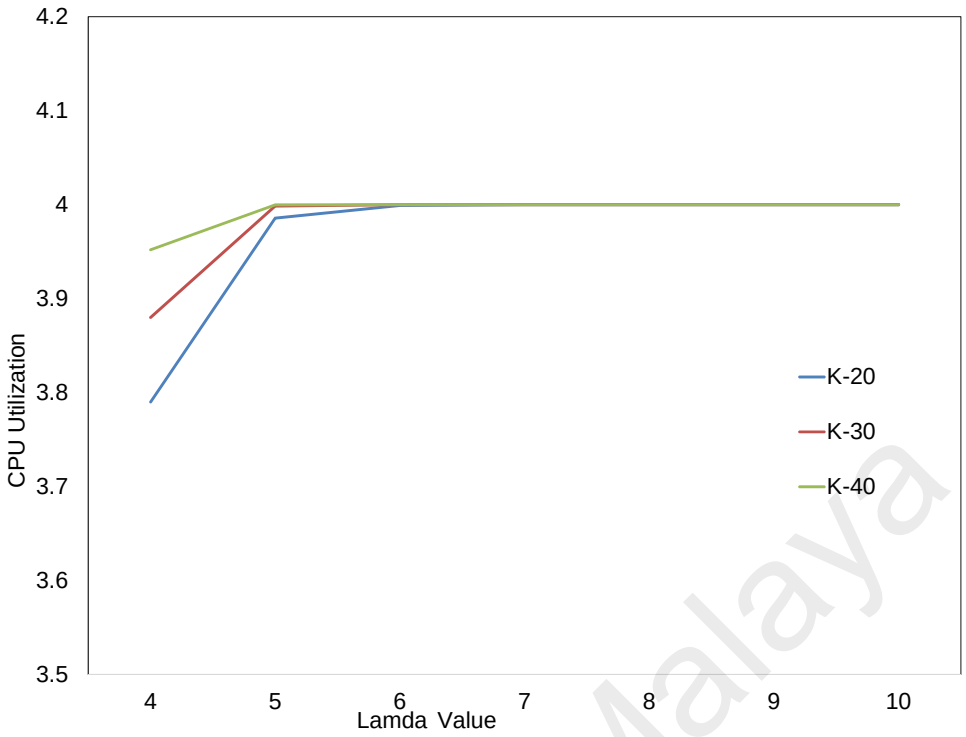

Figure 3.19: Cloudlet CPU Resource Uses Pattern Varying the Maximum Allowed User

However, when increasing the arrival rate, all the cases obtain the maximum CPU utilization. Therefore, the maximum number of users and the user arrival rate affect the CPU utilization. The greater the arrival rate and maximum number of allowed users in the system, the stronger the CPU utilization will impact the cloudlet system.

3.4.2.3 Response Time with Varying Maximum Number of Allowed User in Cloudlet

In this case, we observe the cloudlet response time, *Tres*, in Figure 3.20, which shows the total cloudlet execution time of an application and the effect. Here, we simply use 4 CPU cores and vary the maximum number of allowed users in the cloudlet to observe the overall effects.

Figure 3.20 shows that the maximum number of allowed users on the cloudlet system has a significant effect. The greater the maximum number of allowed users, the longer the response time, namely, slower user response and increased overall cloudlet execution time. Interestingly, we observe that increasing the maximum number of allowed users

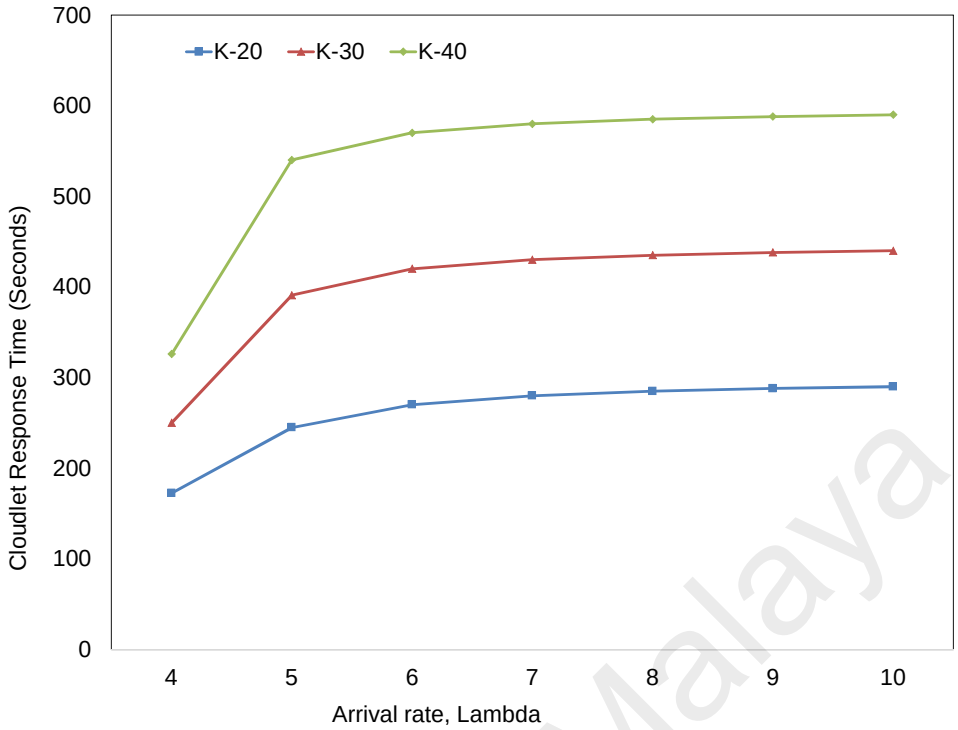

Figure 3.20: Cloudlet Response Time Observation Varying the Maximum Allowed User

produces a pattern of increased response time when shifting the line upward.

3.5 Preliminary Model

In this section, we present the preliminary model that we have developed with the objective of performing a problem analysis. The model defines a relationship between the cloudlet resources and the number of cloudlet users that leads to the unacceptable performance of the cloudlet. This model will be used to show that the problem established through empirical analysis is not limited to one instance but can be generalized to all possible cloudlet instances. We first present the notations that will be used in the model. This is followed by the model presentation.

3.5.1 Notations

The following notations are used in the model:

1. S_m: Mobile execution speed in MIPS.

2. S_c: Cloudlet virtual machine execution speed in MIPS.

3. T_{mob}: Application execution time on the mobile device.

4. T_s: Application execution time on the cloudlet when no other applications are being executed.

5. T: Application execution time on the cloudlet when other applications are being executed.

6. T_{mc}: Application transfer time from the mobile device to the cloudlet.

7. T_{cm}: Application transfer time from the cloudlet to the mobile device.

8. T_{pd}: Network propagation delay.

3.5.2 Model

We assume that the cloudlet execution speed will always be greater than the mobile device execution speed; otherwise, the mobile device will not join the cloudlet. Specifically,

$$S_c > S_m$$

We also assume that the two speeds are linked by a speedup factor, where

$$S_c = F * S_m$$

Because the execution time of the instructions is dependent on the processing speed of the computational unit, it naturally follows that the mobile device execution time will be higher than the cloudlet execution time in the absence of external overhead.

$$T_{mob} > T_s$$

The total time taken for the remote execution of an application is given by the following equation:

$$T_{tot} = 2 * T_{pd} + T_{mc} + T_{cm} + T$$

We assume that the transfer time can be ignored for simplicity because it is not a factor that directly affects the cloudlet performance and is beyond control of the cloudlet parameters. We obtain

$$T_{tot} = T = T_s + T_o + T_w$$

To ensure that the mobile device obtains the benefit from offloading its computations to the cloudlet, the following condition must be satisfied.

$$T_{mob} > T \tag{3.3}$$

Let the number of cloudlet CPU cores be c; then, T can be given as Equation 3.4, where n is the number of users on the cloudlet.

$$T = max(0,(n-c)) * \frac{T_s}{c} + T_s \tag{3.4}$$

Substituting the value of T from Equation 3.4 into Equation 3.3 and solving, we obtain

$$\begin{aligned}
T_{mob} &> max(0,(n-c)) * \frac{T_s}{c} + T_s \\
T_{mob} &> max(0,(n-c)) * \frac{T_{mob}}{Fc} + \frac{T_{mob}}{F} \\
1 &> max(0,(n-c)) * \frac{1}{Fc} + \frac{1}{F} \\
1 &> \frac{(max(0,(n-c)) + c)}{Fc} \\
Fc &> (max(0,(n-c)) + c)
\end{aligned} \tag{3.5}$$

Equation 3.5 is a simplified equation that is based on multiple assumptions. Specifically, it assumes that all applications submitted to the cloudlet are equally sized and that the speed-up factor for all mobile devices is the same. Nevertheless, the equation gives valuable insight into cloudlet performance and the problem of resource scarcity. The equation states that the product of the speed-up factor of the mobile devices and the number of cores in the cloudlet must be greater than the sum of the number of users in the cloudlet and the number of CPU cores in the cloudlets for the mobile devices to be able to achieve any benefit from the cloudlets. Using this expression, we can compute the maximum number of permitted users on the cloudlet and show that the resource scarcity problem of the cloudlet arises for fewer numbers of user applications.

3.5.3 Example

Consider a 16-core cloudlet that is connected to relatively powerful and newer mobile devices such that the speed-up factor is 2. We have F=2, and c=16. We want to determine the maximum number of users that the cloudlet can accommodate before it should reject further users or else result in no benefit for the mobile applications in terms of execution time. We have

$$Fc > (max(0,(n-c)) + c)$$

$$2 * 16 > n$$

$$32 > n$$

This means that a maximum of 31 users can simultaneously coexist on the cloudlet and that a further increase will result in no time benefit. Note that a steady user arrival rate of 8 users per minute with a service requirement of slightly over 4 minutes can easily result in 32 users after only 4 minutes of cloudlet operation. The proposed preliminary model

links the CPU resources available in the cloudlet with the number of users that the system can accommodate. The relationship shows that resource scarcity is experienced fairly early. Hence, it can be concluded that the resource scarcity problem is real and needs to be addressed.

3.6 Conclusion

In this chapter, we presented the empirical analysis to demonstrate that the scarcity resource problem experienced by cloudlets is a non-trivial problem. We investigated the impact on finite-resource cloudlet performance by allowing computation-intensive mobile application augmentation to be obtained using a computational cloudlet. We experimentally and analytically demonstrated that a large number of user tasks proportionally exacerbates the resource scarcity problem faced by cloudlets.

Based on the aforementioned experiments and results, we conclude that if the resources are overutilized or are scarce, then for all the cases, the cloudlet system performance will be negatively affected through reduced user service time, throughput, efficiency, user time benefit and overall cloudlet performance. We also link resource scarcity in cloudlets with user time benefit. We developed a preliminary mathematical model to generalize the empirical findings. Finally, we derived an expression under simplifying assumptions and found that the resource scarcity problem can arise even when substantial cloudlet resources are available.

In this research, our aim is to enhance the performance of cloudlets without increasing the cloudlet resources or requiring manual intervention during normal cloudlet operation. In the next chapter, we will propose a framework to enhance the optimal cloudlet performance by exploiting and orchestrating nearby mobile devices. We expect that the MobiCoRE framework will significantly enhance the performance of finite-resource cloudlets and ensure mobile user benefit.

CHAPTER 4 : MOBICORE: MOBILE DEVICE BASED CLOUDLET RESOURCE ENHANCEMENT FRAMEWORK

This chapter presents our proposed framework, Mobile-device-based Cloudlet Resource Enhancement (MobiCoRE), for cloudlet performance enhancement. Based on the literature review and the empirical results presented in Chapter 3, it is evident that the performance of cloudlets can become sub-optimal, specifically under heavy loads. Therefore, a number of researchers have addressed the issue of cloudlet resource scarcity by augmenting the cloudlet resources using distant clouds, cloudlets in close proximity and mobile devices. The objective is to explain and address the issues identified concerning resource scarcity problems discussed in the previous chapter. This chapter explains the framework and operating procedure and presents the working principle of the proposed framework. We present the main building blocks, component diagram, and overall schematic diagrams of the proposed framework and describe their functionality. In addition, the interaction among major components of the framework is illustrated in detail using sequential diagrams and flowcharts. In this chapter, we address the limitations of mobile-device-based augmentation approaches. We propose mobile-device-based cloudlet resource enhancement (MobiCoRE), which effectively addresses the limitations of existing approaches. We modelled the MobiCoRE framework mathematically. MobiCoRE effectively improves the user experience for mobile devices as well as the cloudlet response time.

The chapter starts with an overview of MobiCoRE in Section 4.1. This is followed by a detailed description of MobiCoRE in Section 4.2, where we explain the operation of various components of MobiCoRE. We highlight how MobiCoRE has addressed the limitations of the state of the art in Section 4.3. In Section 4.4, we present the mathematical

model of MobiCoRE. The chapter is concluded in Section 4.5.

4.1 Overview of MobiCoRE

The MobiCoRE framework essentially consists of two major building blocks. One is the user side; we assume it to be a mobile device, and other part is Cloudlet. In this study, we assume that all the devices are of the same type from the operation perspective. For the cloudlet, a small-scale cloud is built and operated using the open-source cloud software Openstack. We explain how the frameworks interact with each other in the following sections, and we demonstrate the performance enhancement obtained by our proposed framework based on the the substantial performance gains and enhancement of the overall cloudlet performance of mobile application augmentation.

MobiCoRE performs four functions: (i) parameter monitoring, (ii) dynamic service time control, (iii) partially executed task state wrapping and (iv) admission control. The block diagram of MobiCoRE is shown in Figure 4.1. MobiCoRE monitors a number of cloudlet parameters, including user arrival rate, average number of users on the system and average time served. The complete set of monitored and derived parameters is listed in table 4.1. Based on the monitored parameters, a desired average service time is periodically set to ensure an optimal cloudlet response time for a given arrival rate and cloudlet resources.

The key feature of MobiCoRE is the determination of (i) when to use mobile device resources and (ii) which tasks should use the mobile device resources. MobiCoRE uses mobile device resources when the average number of user tasks in the cloudlet increases beyond the expected number of tasks for an optimal response. At this stage, the desired service time is reduced, and based on the average time served for the active tasks, a set of tasks is chosen to use the mobile device resources and vacate the cloudlet. Tasks that have a time served that is greater than the desired average service time are selected to use

Table 4.1: MobiCoRE Parameters

Parameter	Symbol
Cloudlet CPU cores	c
Cloudlet max. active tasks	K
User task arrival rate	λ
Task rejection rate	P_K
Task departure rate	μ
Average number of tasks in Cloudlet	$\bar{N}$
Average expected service time	$\bar{T_s} = \frac{1}{\mu}$
Average desired service time	$\bar{T_{ds}}$
Average time served	$\bar{T_{ts}}$

the mobile device resources to complete the remaining task. The execution state of these tasks is generated, wrapped into process resumption code and sent to their corresponding mobile device.

4.2 System Description

A functional diagram of MobiCoRE is shown in Figure 4.2. The cloudlet is represented as a group of CPU cores and a ready queue. Although a process can be in states, such as waiting for resources or sleep, such states do not have a significant impact on the cloudlet performance and are omitted in this diagram. Similarly, the cloudlet contains other resources, including memory and disk space; however, the combined effect of all resources is a certain processing speed (in units of MIPS) for user applications. Therefore, such details have also been omitted. Three components, namely, Admission Controller, Service Controller and Reply Wrapper, provide the key functionality of MobiCoRE. The three components are explained in the following sections in detail.

4.2.1 Admission Controller (AC)

The mobile devices submit their tasks to the Admission Controller component. This component admits the mobile device task to the cloudlet as long as the number of tasks in the system is less than the maximum user threshold K. K is set by considering the task arrival rate λ. This ensures that λ in the cloudlet does not exceed a limit such

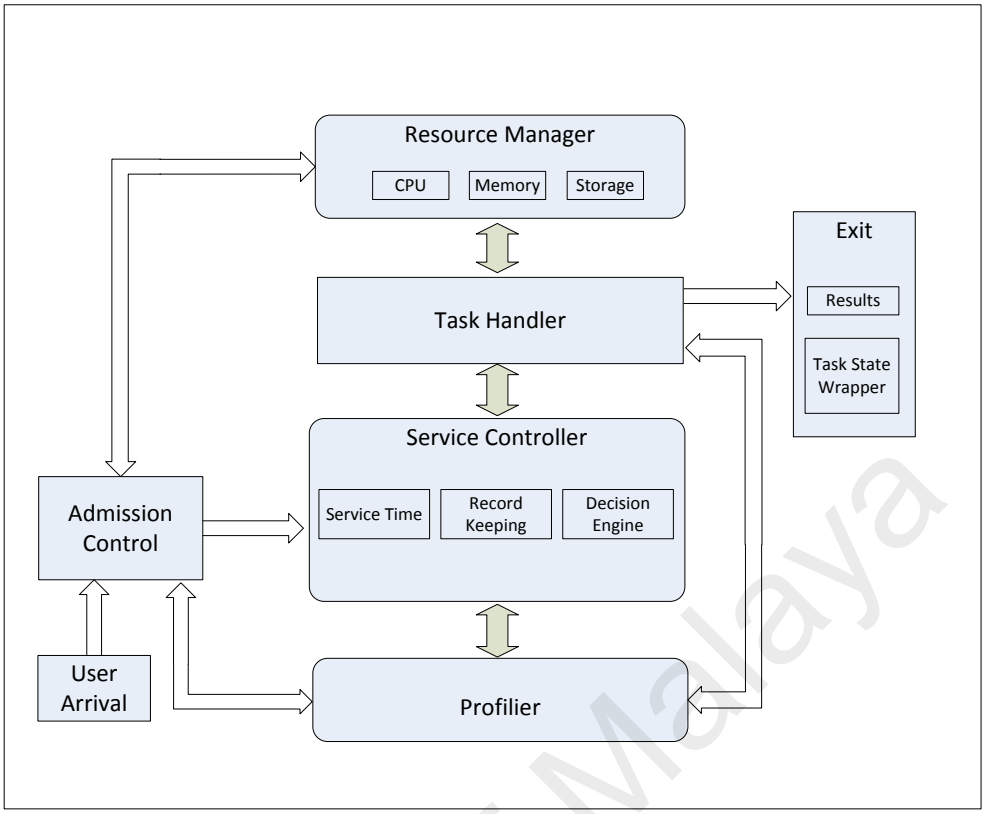

Figure 4.1: Schematic Block Diagram of MobiCoRE Framework

that the cloudlet cannot efficiently process the submitted tasks. To achieve the above-mentioned functionality, the Admission Controller must know the number of application tasks joining the cloudlet system as well as the number of tasks departing the cloudlet system.

The tasks joining the system are directly counted by the Admission Control, and the Service Controller informs the Admission Controller of the departing tasks. Using the two values, the Admission Controller computes the number of tasks in the system and uses the computed value to provide the admission control functionality. Using the instantaneous values, the Admission Controller also computes the truncated running average of the number of users N_{Avg} in the system. This parameter is also communicated to the Service Controller component.

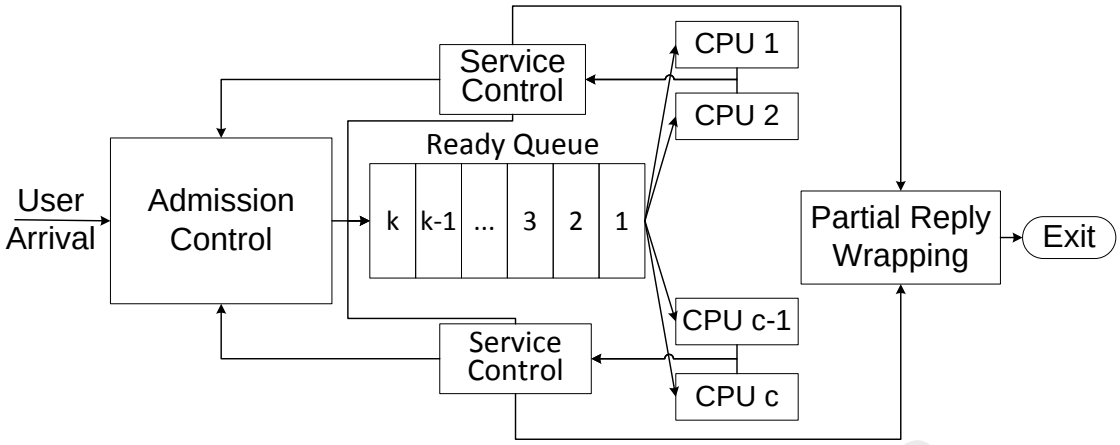

Figure 4.2: Functional Diagram of MobiCoRE

4.2.2 Task Execution

Once the mobile user task is admitted to the cloudlet, it can be in one of two states with reference to the cloudlet operations. If the task is currently being served, it will be in the execution state, with task instructions being executed by one of the available CPUs. Each CPU spends at most a specified amount of time executing the instructions of a task. After the specified number of instructions are executed, the task state is saved by the cloudlet and is subsequently swapped out of the CPU, with its process control block then added to the wait/ready queue. In MobiCoRE, the task is submitted to the Service Control component before re-entering the queue.

4.2.3 Service Controller (SC)

The Service Controller performs three operations:

- Periodic computation of desired average service time $\bar{T_{ds}}$
- Record keeping of individual time served T_{ts}^{i} for each task i and computation of average time served to active tasks $\bar{T_{ts}}$
- Decides on the vacating of mobile tasks considering $\bar{T_{ds}}$, $\bar{T_{ts}}$ and the optimal average service time $\bar{T_{s}}$

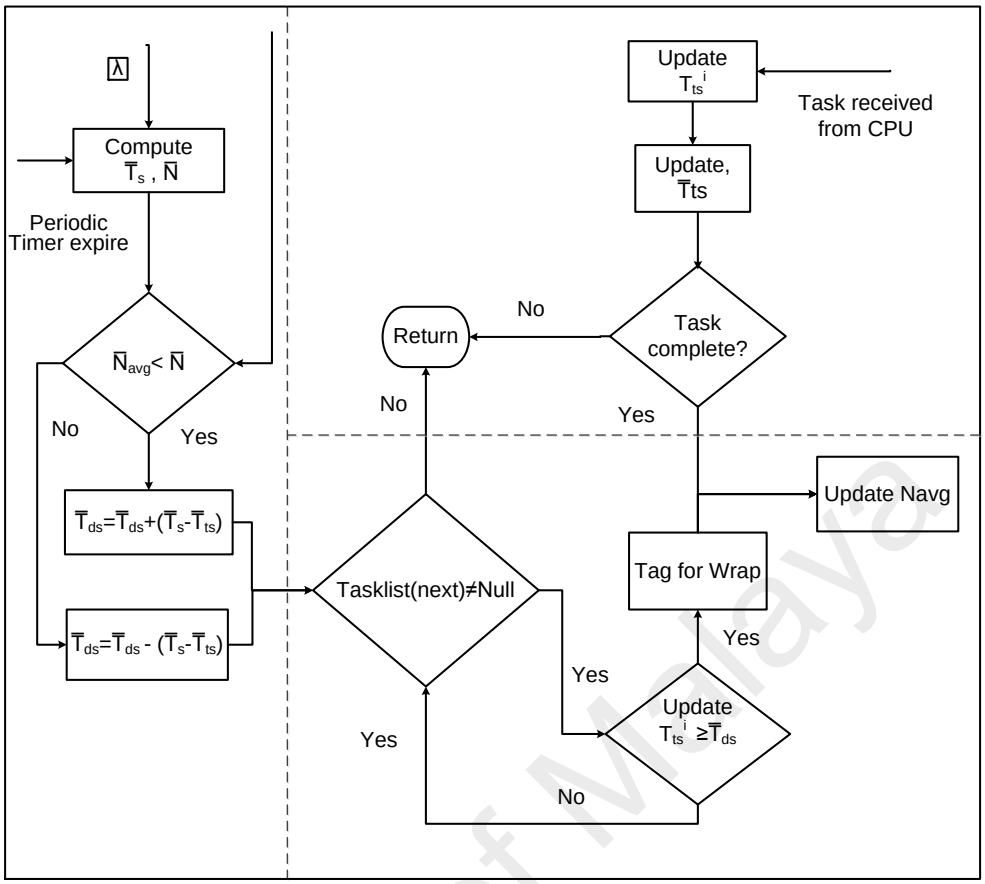

Figure 4.3: Service Controller Work Flow Diagram

Figure 4.3 shows the three operations as well as the interaction of the Service Controller with the other components. Each function is explained in the following sections.

4.2.3.1 Desired Service Time Computation

MobiCoRE attempts to operate the cloudlet with an optimal average service time such that the mobile devices can complete their tasks using the cloudlet service as quickly as possible. The mobile application arrival rate and cloudlet resources are used to compute the optimal average service time (the computation is explained in Chapter 6). At a given instant, the average time served for the active mobile tasks and the optimal average service time can differ. The difference mainly arises from a varying arrival rate, which can result in a different optimal average service time. The increase in optimal average service time requires an increase in time served for the mobile tasks before vacating them from the

cloudlet system and vice versa. For this purpose, the desired service time is used and computed as explained in the following paragraph.

There is an associated value of the average number of users $\bar{N}$ in the cloudlet corresponding to each value of the average service time. The average number of users N_{avg} in the cloudlet should be equal to the expected average number of users $\bar{N}$ to obtain the optimal operation of the cloudlet. Given the task arrival rate λ and system resources c, MobiCoRE computes the expected average number of users $\bar{N}$ and expected service time $\bar{T}_s$ for optimal operation. If $N_{avg} < \bar{N}$, $\bar{T_{ds}}$ is increased by the truncated running average of $(\bar{T}_s - \bar{T_{ts}})$, where $\bar{T_{ts}}$ is the average time served for the active tasks. This results in the desired time served $\bar{T_{ds}}$ being set higher than the expected service time $\bar{T}_s$. Similarly, if $N_{avg} > \bar{N}$, $\bar{T_{ds}}$ is decreased using the truncated running average of $(\bar{T_{ts}} - \bar{T}_s)$.

The effect of this periodic adjustment is a fluctuation of the desired average service time $\bar{T_{ds}}$ around $\bar{T}_s$. The long-term average of $\bar{T_{ds}}$ is in general equal to $\bar{T}_s$, which ensures an optimal response time for the mobile devices from the cloudlet.

4.2.3.2 Record Keeping of Time Served

The second function of the SC is to keep a record of the time served to each active mobile task in the cloudlet. For this purpose, the SC records the amount of time that was served to the tasks by the CPU during each cycle. The time served during each cycle is added to the total time served to the task T_{ts}^{i} when the task is submitted to the SC. Using the individual time served for the active tasks, the SC computes the average time served $\bar{T_{ts}}$ to currently active tasks by the cloudlet.

4.2.3.3 Decision on Task Vacation

The third function of the SC is decision making with reference to using the resources of mobile devices. The objective of the decision-making process is to maintain the parameter – the instantaneous number of users in the system – N_{inst} as close to $\bar{N}$ as possible.

To achieve this, the average time served to the active tasks is maintained as close to $\bar{T_{ds}}$ as possible. When a task is submitted to the SC, it compares the time served to the task T_{ts}^{i} by the cloudlet with $\bar{T_{ds}}$. If $T_{ts}^{i} \geq \bar{T_{ds}}$, the task is chosen for resource sharing with the mobile device. Such tasks are marked for wrapping and sent to the state wrapper, and the task is marked as departing the cloudlet. If the submitted task is complete, it is marked as departing the cloudlet without wrapping. Otherwise, the task is added to the ready queue for execution.

4.2.4 Task State Wrapper

If the task is not completely executed by the cloudlet, the mobile device must resume the task from the last process state in the cloudlet to be able to complete the task. A number of mechanisms are available that save the process state of a process and resume the process on a remote system using the saved state. We recommend the checkpointing and migration mechanism based on the wrapper methods proposed by Litzkow et al. (Litzkow, Tannenbaum, Basney, & Livny, 1997). In this mechanism, no modification or pre-installed code is required on the remote end, which in our case is the mobile device. A signal is generated at the beginning of the execution of the generated code. The signal handler consists of a jump to the process resumption code, which invokes the same application on the remote device and immediately suspends it, thereby creating the skeleton structures in the kernel space. Subsequently, the saved state is used to populate the structures. This is followed by another jump to the actual instruction pointer of the application.

4.2.5 Task Flow

The complete task flow of a mobile application in the MobiCoRE-based cloudlet is shown in Figure 4.4. The mobile device submits the task to the cloudlet using a request for augmentation. The cloudlet either denies service through the admission controller or

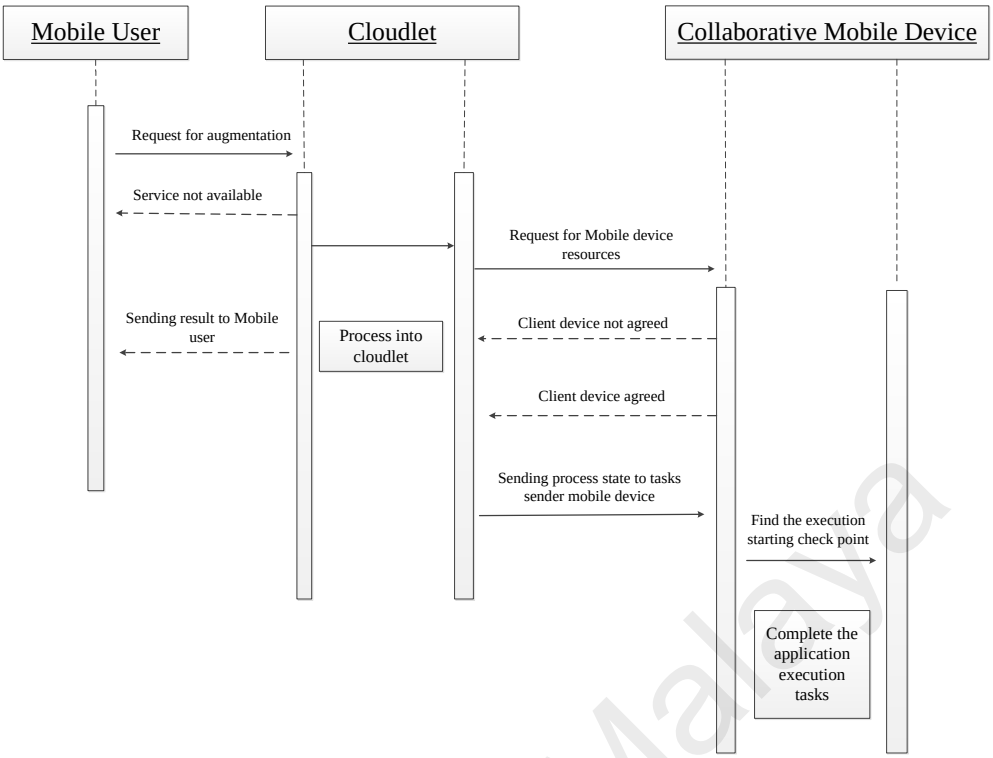

Figure 4.4: Sequential Work Flow Interaction Diagram of Mobile Devices and Cloudlet Task Completion

admits the task. In the case of a heavy load, the cloudlet performs partial tasks and informs the mobile device that its resources will be required to complete the task. The partially completed tasks are wrapped and sent to the mobile device for completion. Alternatively, the task is completed by the cloudlet in the case of light loads or limited demand by the mobile device, and completed tasks are returned to the mobile device.

The performance and effectiveness of MobiCoRE are dependent on the proper selection of the parameters, specifically the expected average number of users and desired average service time. To derive the values for the parameters, we have mathematically modelled MobiCoRE. Before presenting the mathematical model, we highlight the limitations of existing schemes that have been addressed by MobiCoRE. Some of these limitations represent important challenged for achieving the wide-spread use of cloudlets as offloading elements.

4.3 Existing Limitations and MobiCoRE Solution

The limitations of existing mobile-device-based cloudlet resource enhancement mechanisms were discussed in Chapter 2, Section 2.6. In this section, we highlight how MobiCoRE can address these limitations.

1. All schemes proposed in the literature require partitioning of the computational part of the application code before part of it can be migrated to the cloudlet. This partitioning is based on a complex decision-making process and often results in negligible advantage in terms of execution time. No such partitioning is required in the case of MobiCoRE as long as user and system interaction elements have been separated from the computation. This is because MobiCoRE wraps the partial state into an executable state that is directly executable on the mobile device.

2. No change in mobile application is required to make it compatible for cloudlet execution. Moreover, the executable for the mobile device can be used to run the mobile application on the cloudlet, provided that the mobile device emulator is available on the cloudlet.

3. No component of the system needs to be installed on the mobile device, which is one of the major concerns of users given the fact that the cloudlets are untrusted entities available for public access. The execution of applications from partial states is realised.

4. Unlike current schemes whereby mobile devices share the computational load of other devices, MobiCoRE does not use the mobile devices to execute the tasks of any other device. The resource sharing is conducted with the mobile device to facilitate the completion of the task. This ensures the privacy as well as the integrity of the computation and prevents malicious code propagation.

5. The resources of only those devices that have high service demands are used. Therefore, it is ensured that the resources of mobile devices used by the cloudlet are always lesser than the resources required to complete its own task. This is because only a fraction of the mobile device tasks are sent back.

4.4 Mathematical Model of MobiCoRE

In this section, we present the mathematical model for MobiCoRE. The execution of the mobile task using MobiCoRE is performed in two parts. The first mandatory part of the computation is performed in the cloudlet. The second part is optional and dependent on the cloudlet average service time and mobile service requirements. If the service requirement of a mobile device is higher than the average service time, there is significant probability that for such a device, the device's resources will be shared with the cloudlet to complete its submitted task. On the other hand, for devices with service requirements that are less than the cloudlet service time, no resources from the mobile device are used, resulting in the absence of the second part of computation.

4.4.1 Cloudlet Service Model

The mobile applications submitted to the cloudlet can be represented by two MobiCoRE-related attributes. Every application has a service requirement T^i_{req} from the cloudlet. This is the amount of time that the application will require if the cloudlet serves only one application. In general, this time is not known in advance and cannot be used as a parameter to tune the system. The second attribute is the ratio of the cloudlet processing speed and the mobile device processing speed, given the cloudlet serves only the specific mobile user application. This ratio is known in advance and is represented by f^i for device i. We assume that the applications arriving for service at the cloudlet follow a Poisson distribution with an arrival rate of λ. Every application will be served in the cloudlet for time T^i_{ts}, where $T^i_{ts} \leq T^i_{req}$. Because of MobiCoRE's service control, the average time served

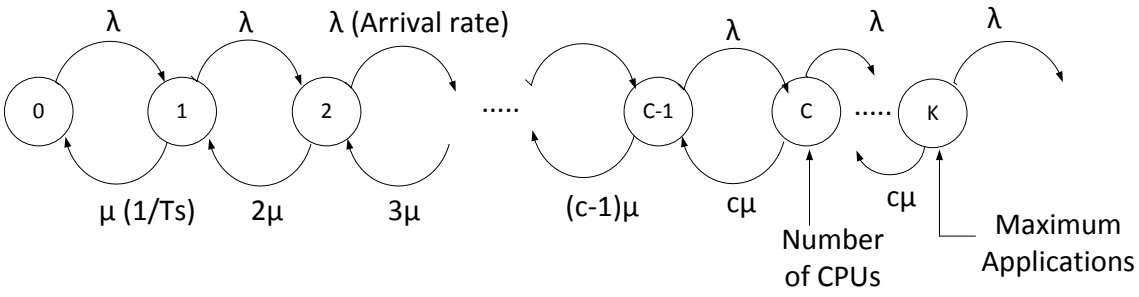

Figure 4.5: Birth Death Model of MobiCoRE

for all applications in the cloudlet will be $\bar{T_s} \approx \frac{1}{n} \sum_{i=1}^{n} T_{ts}^{i}$. Given that there are multiple user applications requiring the resources of the cloudlet, every application will wait a time T_w^i for the resources to be free. In the case of a cloudlet, the wait time is distributed between the time served because of the interleaving of the processes.

Let c be the number of CPUs available in the cloudlet, and the maximum users that can be served at any instance is K. The cloudlet service under MobiCoRE can be mapped on an M/M/c/K queue with an arrival rate λ and a departure rate $\mu = \frac{1}{\bar{T_s}}$. Such a system can be modelled using a birth-death Markov chain model, as shown in Figure 4.5. The steady-state probabilities of being in any state of the model maps to the probabilities of having n users in the cloudlet (waiting or being served). These probabilities are given by equations 4.1 and 4.2. The probability P_K is the probability that a user will be rejected by the cloudlet because the system cannot accommodate additional users. Using these probability values and the equations $\rho = \frac{\lambda}{\mu}$ and $a = \frac{\rho}{c}$, we can derive the parameters related to the cloudlet performance.

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{\lambda^n}{n!\mu^n} + \sum_{n=c}^{K} \frac{\lambda^n}{c^{n-c}c!\mu^n} \right]^{-1} \tag{4.1}$$

$$P_n = \begin{cases} \frac{\lambda^n}{n!\mu^n} P_0 & \text{for } 0 < n < c \\ \frac{\lambda^n}{c^{n-c}c!\mu^n} P_0 & \text{for } c \leq n \leq K \end{cases} \tag{4.2}$$

Given the probabilities P_0 and P_K, we can compute the cloudlet normalized utilization U using equation 4.3, and the average number of users $\bar{N}$ in the cloudlet is given by equation 4.4. $\bar{N}$ and c determine the load on the cloudlet as it serves the mobile device applications. A reduced load results in an improved performance by the cloudlet. The two parameters contribute to the average waiting time and the response time of the cloudlet, both of which indicate the usefulness of the cloudlet service to the mobile devices. The average time spent by the applications in the cloudlet $\bar{T_{rc}}$ is the sum of the average service time $\bar{T_s}$ and average waiting time $\bar{T_w}$ and is given by equation 4.5. The average waiting time is given by equation 4.6.

$$U = \frac{\lambda(1-P_K)}{c\mu} \tag{4.3}$$

$$\bar{N} = \rho(1-P_K) + \frac{P_0 \rho a}{c!(i-a)^2} * \left[1 - a^{K-c+1} - (1-a)(K-c+1)a^{(K-c)} \right] \tag{4.4}$$

$$\bar{T_{rc}} = \frac{\bar{N}}{\lambda(1-P_K)} \tag{4.5}$$

$$\bar{T_w} = \frac{\bar{N} - \rho(1-P_K)}{\lambda(1-P_K)} \tag{4.6}$$

Equations 4.3-4.6 can be used to compute the performance of the cloudlet. However, these equations can only be useful in improving the mobile device application perfor-

mance in the cloudlet if the entire mobile application is executed in the cloudlet. This is the case for applications where the time served by the cloudlet for the application is equal to the application requirement, i.e., $T^i_{req} = T^i_{ts} \leq \bar{T}_s$. On the other hand, for applications where $T^i_{req} > T^i_{ts}$, MobiCoRE uses the respective mobile devices to complete the application execution. Therefore, we also need to consider the computations being performed by the mobile devices to completely characterise the performance of MobiCoRE-based cloudlets.

4.4.2 Mobile Computation Model

The time required by the mobile device to complete the partially executed task depends on the remaining applications as well as the speed of the mobile device with reference to the cloudlet processing speed. Recall that the service requirement time T^i_{req} is the time required by the ith mobile device application to be completed by the cloudlet, provided there are no other applications in the cloudlet. We know that the ratio of the cloudlet to mobile device processing speed is given by f^i. Given these two values, the mobile device can complete the entire task (without cloudlet assistance) in time $T^i_{req} * f^i$ or greater, depending on the load on the mobile device. In the case of MobiCoRE, the mobile application is served for a time T^i_{ts} before returning its state to the mobile device. Assuming that the overhead of the application state transfer and resume can be ignored, the time required by the mobile device T^i_m to complete the task is given by equation 4.7. Consequently, the total time T^i required by the mobile application to complete the execution is the sum of the time served by the cloudlet T^i_{ts}, the wait time in the cloudlet T^i_w and the time required by the mobile device T^i_{ts}, as given in equation 4.8.

$$T_m^i = max(T_{req}^i - T_{ts}^i), 0) * f^i \tag{4.7}$$

$$T^i = T_{ts}^i + T_w^i + T_m^i \tag{4.8}$$

The introduction of the mobile computation into the cloudlet analysis significantly changes the behavior of the system compared to standard birth-death Markov models.

4.5 Conclusion

In this chapter, we proposed a resource augmentation mechanism for cloudlets that uses mobile devices to augment the resources of cloudlets. MobiCoRE adjusts the service time for the mobile applications in the cloudlet. The tuning of the service time leads to reduced service times for individual mobile applications when the cloudlet is subject to heavy loads. The reduced service time results in the early vacation of users from the system, which in turn improves the efficiency and performance of the cloudlet. This also results in service for an increased number of users, even under heavy loads. The partially vacated task states are wrapped and sent to the mobile device for further execution, resulting in time benefits at the cost of minimal overhead.

MobiCoRE requires no code modification and no code partitioning, thereby relieving programmers from having to think about code partitions at the time of programming. Simultaneously, no code component is required on the mobile device for MobiCoRE to function properly. Furthermore, the application of a mobile device is offloaded for augmentation to the same device, thus eliminating the issues of privacy and integrity. The resource sharing from the mobile devices for their own tasks guarantees that the resources used from the mobile device for cloudlet augmentation are always lesser than the application demand of the respective mobile device. Further, only those devices that

require extensive cloudlet resources are used for the augmentation of the cloudlet.

A comprehensive mathematical model has been proposed to facilitate a performance evaluation of MobiCoRE. In the next chapter, we present the details of the empirical setup and model the solution setup. The data collection details and the collected data are presented. Subsequently, the data are analyzed and subsequently used to verify the correctness of model and draw meaningful conclusions about the cloudlet's performance under MobiCoRE.

CHAPTER 5 : IMPLEMENTATION AND EVALUATION

We have investigated and identified the resource scarcity problem in cloudlets.For solution, we have proposed MobiCoRE, a performance enhancement framework for cloudlet resource augmentation using mobile devices. This chapter reports on the evaluation approach of the proposed MobiCoRE framework performance. We describe the implementation details and data collection method for the evaluation of our proposed MobiCoRE framework. We explain the tools used for testing the proposed framework, the data collection technique and the statistical method used for the processing of the data. Using a series of experiments in a real environment, we evaluate the performance of our framework. The performance evaluation results of our framework are validated using statistical modelling with the SPSS package. To build our statistical model, we leverage standard well-recognised statistical data analysis tools.

The chapter is organized as follows. Section 5.1 explains the experimental setup that has been used for the collection of data. Section 5.2 presents the data relating to model validation. Section 5.3 presents the data relating to the parametric evaluation. The performance analysis data are presented in Section 5.4. The chapter is concluded in Section 5.5.

5.1 Experimental Setup

In this section, we explain the experimental setup and data collection mechanism. We also highlight the statistical tools that we have used to verify the correctness of the data.

5.1.1 MobiCoRE Implementation

The first and foremost component of the data collection process was the implementation of the MobiCoRE framework.

We implemented MobiCoRE in a lab environment wherein the mobile device was modeled using a single computer and a small-scale OpenStack-based cloud deployment was used as the cloudlet. We used an Openstack-based cloudlet deployed on Dell Poweredge blade servers. To kept the scale of the experiment small; a virtual machine was created with 4 CPUs having a hardware configuration that included a 1.8 GHz Es-2403 processor and 16 GB of RAM. The cloudlet was deployed over the WAN and accessed through Ethernet interfacing. The data rate of the Ethernet network was reduced to match the rate of WiFi access.

A laptop was used as the mobile device, which emulated a Samsung Galaxy II smartphone with a cloudlet to mobile processing ratio of 4. This ratio was achieved by adding background processes onto the laptop until we achieved the speed in MIPS of the Samsung Galaxy II. MobiCoRE was implemented as a shell script on the cloudlet and performed the admission control activity of the cloudlet. We used Linux kernel modules to extract and restore the state of the application processes.

We used a fixed-size application that performed extensive mathematical computations to keep the CPU busy when the application was executed. Application arrival was emulated by generating a Poisson distribution in Matlab with a given average. The generated distribution was added to the script, which selected one value every minute and created a number of applications equal to the selected value. The generated processes were submitted to the cloudlet virtual machine based on the admission control. The time served to each application by cloudlet was measured using the top command and was recorded by the service control. The experiment was run for 90 minutes, and users were

admitted for the first 60 minutes. A grace time of 30 minutes was allowed for the applications to complete execution.

5.1.2 Mobile Device Data Collection

We performed the data collection for the mobile device using a real Samsung Galaxy II device. The device was rooted to allow the C-based application to be executed from the shell. The application was transferred over WiFi as well as a 3G network to the mobile device from a laptop to measure the transfer time and energy consumed during the transfer. The application was executed on the mobile device with and without the background processes to estimate the time taken by the application execution on the mobile device. This time was used for comparison with the time achieved by the cloudlet-based and hybrid executions. This time was also used to calibrate the laptop, which acted as a mobile device for the remainder of the experiments.

5.1.3 Model Data Collection Mechanism

The mathematical model was implemented in Matlab. A Poisson distribution was used for the generation of the user arrival time as well as the user service requirement. We generated data for 80 different service time values by varying the service time from 1-80 seconds. Similarly, the CPU resources and average user arrival rate were varied between 4-32 and 4-10, respectively. Multiple results were generated and subsequently averaged with a confidence interval of 99%.

The Matlab script directly generated all the desired values, and the data were directly written into Excel files by the Matlab script. The generated data were only processed to generate graphs. Otherwise, the entire data generation process was automated.

5.1.4 Evaluation Metrics

A large set of metrics were selected for the evaluation. These metrics along with a brief description of each are given here.

- *Number of CPUs in the Cloudlet*: The total number of CPU cores in the cloudlet that are available for the mobile devices.
- *Maximum Number of Allowed Users*: The maximum number of users that can be accommodated by the cloudlet. Additional users are rejected if this number is reached.
- *Average Number of Users in the Cloudlet*: The average number of users in the cloudlet during a given execution cycle.
- *Service Time from the Cloudlet*: The CPU time given to the mobile application by the cloudlet.
- *Wait Time in the Cloudlet*: The time when the application is in the cloudlet but is waiting for the CPUs to vacate and is thus not being served.
- *Response Time from the Cloudlet*: The sum of the service time and the wait time in the cloudlet.
- *Mobile Device Time*: Time taken by the mobile device to complete the partially finished task.
- *Total Time by the Mobile Device*: Time taken by the mobile device if entire task is executed by the mobile device without the cloudlet. This time is used a benchmark to measure the benefit for the mobile device for submitting the task to the cloudlet.
- *Total Response Time of the Application*: The sum of the cloudlet time and the mobile time required for a task to be completed under MobiCoRE.

Table 5.1: Evaluation Metrics

Parameter	**Unit**
Number of CPUs in cloudlet	
Maximum number of users possible	
Average number of users in the cloudlet	
Service time from the cloudlet	Seconds
Wait time in the cloudlet	Seconds
Response time from the cloudlet	Seconds
Mobile device time	Seconds
Total time of the mobile device	Seconds
Total response time of the application	Seconds
User arrival rate	per Minute
Number of users served	
User service requirement	Seconds
Cloudlet utilization	
Queue size	

- *User Arrival Rate*: The average number of user applications that are submitted to the cloudlet per minute.
- *Number of Users Served*: The total number of users served by the cloudlet over a given period of time.
- *User Service Requirement*: The service requirement time of the mobile application from the cloudlet if the cloudlet serves only this application.
- *Cloudlet Utilization*: The ratio of the average number of utilized cloudlet CPUs to the total number of available CPUs.
- *Queue Size*: The number of users that are in the cloudlet but that are not being served.

Table 5.1 provides the list of metrics evaluated along with the units.

5.1.5 Statistical Validation of Data

According to the sample central limit theorem, approximately 99% of the sample means fall within 2.58 standard deviations of the population mean, provided that the sample size is greater than or equal to 30 (n>= 30). Hence, a computation-intensive prototype

application is evaluated on the basis of five parameters with 30 different computational intensities. The empirical data are collected for all the components of the mobile application in 30 experiments. Each experiment is conducted 30 times for the evaluation of each parameter to derive the values of the point estimator.

The measurement of the central tendency of the data sample of each experiment is calculated using the sample mean(-X) because the sample mean ascertains a better point estimate of the population mean compared to the median or mode (Confidence Intervals and Sample Size). Data sampling involves the sampling error; hence, the sample mean can differ from the population mean. Therefore, to signify the goodness of the calculated point estimate, the interval estimate of each sample is determined. The interval estimate of a parameter represents the interval or range of values used to estimate the parameters. The confidence level of an interval estimate of a parameter indicates the probability that the interval estimate contains the parameter. Let E represent the error estimate for the 99% confidence interval, which is calculated using the following equation:

$$E = 2.58 \times (\frac{\sigma}{\sqrt{n}}) \tag{5.1}$$

where σ indicates the standard deviation of the sample values and n indicates the size of the sample space. The interval estimates for each sample mean ($\bar{X}$) of the primary data are calculated with a 99% confidence interval using the following equation.

$$ConfidenceInterval = (\bar{X}) \pm E \tag{5.2}$$

The following section presents the data collected during different experiments for the evaluation of the MobiCoRE framework as applied to the real-time cloudlet environment.

Table 5.2: Number of Average User in Model Validation

Service Time	**Number of average user (Empirical)**	**Number of average user (Model)**
40	3.85	3.42
45	5.04	4.48
50	6.4	6
55	8.38	8.28
59	10.56	10.55
65	10.54	10.77

5.2 Model Validation Data

We heavily rely on the mathematical model results that we obtained for the performance evaluation of MobiCoRE. Therefore, the first step is to verify the correctness of the model. To this end, we collected the model validation data; then, we compared the data of the model and, under the same conditions, the data generated when using the MobiCoRE implementation. We validated our model by employing statistical analysis tools for the statistical validation; in this case, we use a T-test:Paired sample test and Pearson coefficient. The statistical tests validate our model.

Table 5.2 presents the data for the average number of users in the system for different values of cloudlet service time and constant values of the number of CPUs and the user arrival rate, both of which were set to 4. The values presented in the table show a near perfect match between the empirical and model results. Table 5.3 shows the results of the average task completion time for 240 applications that were generated using a Poisson arrival rate distribution under MobiCoRE and the model. Once again, the results show a perfect match between the two results, verifying the correctness of our model.

5.3 Parametric Analysis Data

In this section, we present the data that has been used to identify the optimal operational parameters.

Table 5.3: Task Completion Time for Model Validation

Service Time	Job Completion Time (Empirical)	Job Completion Time (Model)
40	158	153
45	153	147
50	156	151
55	167	166
59	188	190
65	173.6	184.8

5.3.1 Task Completion Time

The cloudlet response time along with the mobile time are the first indicators and provide us an idea about the performance of the cloudlet. Therefore, we start with the presentation of the data for the cloudlet response time. Table 5.4 presents the data for the response time. In addition, the data for the cloudlet and mobile time as well as the mobile device benefit are included. The values of other parameters are set as follows. The service requirement of the mobile device is 60 seconds, the user arrival rate is set to 4 applications per minute, the number of available cloudlet CPUs is 4, and the maximum number of users allowed in the cloudlet at any given time is set to 20.

Note that with increasing service time, the task completion time initially decreases until it reaches a minimum value for a service time of 45 seconds. The task completion time increases beyond this point to a maximum value. This behavior indicates that the cloudlet has an optimal operational point. Similar trends have been observed for all values of the arrival rate and the cloudlet resources as long as the system remains stable. This trend will be discussed further in a subsequent chapter.

5.3.2 Probability of User Drop

Based on observations of the task completion time, we study similar trends of other parameters. Table 5.5 presents the data on the probability that a user will be dropped when it arrives at the system. It can be observed that the probability of application drop increases with increasing cloudlet service time.

Table 5.4: Response time of $T_{Cld}, T_{Mob}, T, T_{Avd}$ for Varying Service Time

T_s	T_{Cld}	T_{Mob}	T	T_{Adv}
13.00	13.05	188.00	201.05	38.95
14.00	14.07	184.00	198.07	41.93
15.00	15.10	180.00	195.10	44.90
16.00	16.14	176.00	192.14	47.86
17.00	17.18	172.00	189.18	50.82
18.00	18.24	168.00	186.24	53.76
19.00	19.31	164.00	183.31	56.69
20.00	20.39	160.00	180.39	59.61
21.00	21.49	156.00	177.49	62.51
22.00	22.60	152.00	174.60	65.40
23.00	23.74	148.00	171.74	68.26
24.00	24.91	144.00	168.91	71.09
25.00	26.10	140.00	166.10	73.90
26.00	27.32	136.00	163.32	76.68
27.00	28.58	132.00	160.58	79.42
28.00	29.87	128.00	157.87	82.13
29.00	31.22	124.00	155.22	84.78
30.00	32.61	120.00	152.61	87.39
31.00	34.06	116.00	150.06	89.94
32.00	35.57	112.00	147.57	92.43
33.00	37.16	108.00	145.16	94.84
34.00	38.82	104.00	142.82	97.18
35.00	40.58	100.00	140.58	99.42
36.00	42.45	96.00	138.45	101.55
37.00	44.43	92.00	136.43	103.57
38.00	46.54	88.00	134.54	105.46
39.00	48.81	84.00	132.81	107.19
40.00	51.25	80.00	131.25	108.75
41.00	53.88	76.00	129.88	110.12
42.00	56.73	72.00	128.73	111.27
43.00	59.83	68.00	127.83	112.17
44.00	63.21	64.00	127.21	112.79
45.00	66.89	60.00	126.89	113.11
46.00	70.91	56.00	126.91	113.09
47.00	75.30	52.00	127.30	112.70
48.00	80.09	48.00	128.09	111.91
49.00	85.32	44.00	129.32	110.68
50.00	91.00	40.00	131.00	109.00
51.00	97.16	36.00	133.16	106.84
52.00	103.81	32.00	135.81	104.19
53.00	110.93	28.00	138.93	101.07
54.00	118.53	24.00	142.53	97.47
55.00	126.58	20.00	146.58	93.42
56.00	135.04	16.00	151.04	88.96
57.00	143.87	12.00	155.87	84.13
58.00	153.01	8.00	161.01	78.99
59.00	162.40	4.00	166.40	73.60
60.00	171.00	0.00	171.00	69.00

This is because a grater number of users remain in the system because of the longer service time. Furthermore, the drop ratio is higher for higher arrival rates for the cloudlet service time. We further observe that the value of the drop probability of 0.01 coincides with the value of the minimum task completion time presented in the previous section. This indicates that there is a correlation between the user drop probability and the optimal service time.

5.3.3 Average CPUs Used

Table 5.6 shows the average number of CPUs used in the cloudlet for different user application arrival rates and cloudlet service times. The number of available CPUs for the experiment was set to 4, the maximum number of users allowed in the cloudlet was set to 20, and the average user service requirement was set to 60 seconds. The actual user requirement for one hour of arrival at a given arrival rate is generated using a Poisson distribution. This means that for an arrival rate of 4 users per minute, a total of 240 values were generated. Similarly, for an arrival rate of 10 users per minute, a total of 600 values for the requirement were generated.

The data show that the CPU utilization increases with increasing service time. This is because the user application departure rate decreases with increasing service time, and more users continue to occupy the system for an extended period of time. Similarly, the higher arrival rate results in a greater CPU utilization. A critical observation is that a CPU utilization of 75% coincides with a user drop probability of 0.01. We investigate the relationship between the optimal service time of the cloudlet, user drop probability and cloudlet utilization in a subsequent chapter.

Table 5.8 presents the data on average CPU utilization as a function of service requirement. Similarly to the case for varying numbers of CPUs, for a given value of cloudlet service time, the number of CPUs utilized remains constant, irrespective of the

Table 5.5: Probability of User Drop,P_K Varying with Service Time

Servicetime	**λ=4**	**λ=5**	**λ=6**	**λ=7**	**λ=8**	**λ=9**	**λ=10**
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15	0.00	0.00	0.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	0.00	0.00	0.00	0.00	0.00	0.00	0.01
21	0.00	0.00	0.00	0.00	0.00	0.00	0.01
22	0.00	0.00	0.00	0.00	0.00	0.01	0.02
23	0.00	0.00	0.00	0.00	0.00	0.01	0.03
24	0.00	0.00	0.00	0.00	0.00	0.02	0.05
25	0.00	0.00	0.00	0.00	0.01	0.03	0.07
26	0.00	0.00	0.00	0.00	0.01	0.04	0.10
27	0.00	0.00	0.00	0.00	0.02	0.06	0.12
28	0.00	0.00	0.00	0.00	0.03	0.08	0.15
29	0.00	0.00	0.00	0.01	0.04	0.10	0.18
30	0.00	0.00	0.00	0.01	0.05	0.12	0.20
31	0.00	0.00	0.00	0.02	0.07	0.15	0.23
32	0.00	0.00	0.00	0.03	0.09	0.17	0.25
33	0.00	0.00	0.01	0.04	0.11	0.20	0.27
34	0.00	0.00	0.01	0.05	0.13	0.22	0.29
35	0.00	0.00	0.01	0.06	0.15	0.24	0.31
36	0.00	0.00	0.02	0.08	0.17	0.26	0.33
37	0.00	0.00	0.02	0.10	0.19	0.28	0.35
38	0.00	0.00	0.03	0.11	0.21	0.30	0.37
39	0.00	0.00	0.04	0.13	0.23	0.32	0.38
40	0.00	0.01	0.05	0.15	0.25	0.33	0.40
41	0.00	0.01	0.06	0.17	0.27	0.35	0.41
42	0.00	0.01	0.08	0.19	0.29	0.37	0.43
43	0.00	0.02	0.09	0.21	0.30	0.38	0.44
44	0.00	0.02	0.11	0.22	0.32	0.39	0.45
45	0.00	0.03	0.12	0.24	0.33	0.41	0.47
46	0.00	0.03	0.14	0.26	0.35	0.42	0.48
47	0.00	0.04	0.16	0.27	0.36	0.43	0.49
48	0.00	0.05	0.17	0.29	0.38	0.44	0.50
49	0.00	0.06	0.19	0.30	0.39	0.46	0.51
50	0.01	0.07	0.20	0.31	0.40	0.47	0.52
51	0.01	0.09	0.22	0.33	0.41	0.48	0.53
52	0.01	0.10	0.23	0.34	0.42	0.49	0.54
53	0.01	0.11	0.25	0.35	0.43	0.50	0.55
54	0.02	0.12	0.26	0.37	0.44	0.51	0.56
55	0.02	0.14	0.27	0.38	0.45	0.52	0.56
56	0.03	0.15	0.29	0.39	0.46	0.52	0.57
57	0.03	0.16	0.30	0.40	0.47	0.53	0.58
58	0.04	0.18	0.31	0.41	0.48	0.54	0.59
59	0.04	0.19	0.32	0.42	0.49	0.55	0.59
60	0.05	0.20	0.33	0.43	0.50	0.56	0.60

Table 5.6: Average CPUs Used for Varying Service Time

Servicetime	**λ=4**	**λ=5**	**λ=6**	**λ=7**	**λ=8**	**λ=9**	**λ=10**
13.00	0.87	1.08	1.30	1.52	1.73	1.95	2.17
14.00	0.93	1.17	1.40	1.63	1.87	2.10	2.33
15.00	1.00	1.25	1.50	1.75	2.00	2.25	2.50
16.00	1.07	1.33	1.60	1.87	2.13	2.40	2.67
17.00	1.13	1.42	1.70	1.98	2.27	2.55	2.83
18.00	1.20	1.50	1.80	2.10	2.40	2.70	3.00
19.00	1.27	1.58	1.90	2.22	2.53	2.85	3.16
20.00	1.33	1.67	2.00	2.33	2.67	3.00	3.31
21.00	1.40	1.75	2.10	2.45	2.80	3.14	3.46
22.00	1.47	1.83	2.20	2.57	2.93	3.28	3.59
23.00	1.53	1.92	2.30	2.68	3.06	3.42	3.70
24.00	1.60	2.00	2.40	2.80	3.19	3.54	3.79
25.00	1.67	2.08	2.50	2.91	3.31	3.65	3.86
26.00	1.73	2.17	2.60	3.03	3.43	3.74	3.91
27.00	1.80	2.25	2.70	3.14	3.54	3.81	3.94
28.00	1.87	2.33	2.80	3.25	3.64	3.87	3.96
29.00	1.93	2.42	2.90	3.36	3.72	3.91	3.98
30.00	2.00	2.50	3.00	3.46	3.79	3.94	3.99
31.00	2.07	2.58	3.09	3.55	3.85	3.96	3.99
32.00	2.13	2.67	3.19	3.64	3.89	3.97	3.99
33.00	2.20	2.75	3.28	3.71	3.92	3.98	4.00
34.00	2.27	2.83	3.37	3.78	3.95	3.99	4.00
35.00	2.33	2.91	3.46	3.83	3.96	3.99	4.00
36.00	2.40	3.00	3.54	3.87	3.97	4.00	4.00
37.00	2.47	3.08	3.61	3.90	3.98	4.00	4.00
38.00	2.53	3.16	3.68	3.93	3.99	4.00	4.00
39.00	2.60	3.24	3.74	3.95	3.99	4.00	4.00
40.00	2.67	3.31	3.79	3.96	3.99	4.00	4.00
41.00	2.73	3.39	3.84	3.97	4.00	4.00	4.00
42.00	2.80	3.46	3.87	3.98	4.00	4.00	4.00
43.00	2.86	3.53	3.90	3.99	4.00	4.00	4.00
44.00	2.93	3.59	3.92	3.99	4.00	4.00	4.00
45.00	3.00	3.65	3.94	3.99	4.00	4.00	4.00
46.00	3.06	3.70	3.96	3.99	4.00	4.00	4.00
47.00	3.13	3.75	3.97	4.00	4.00	4.00	4.00
48.00	3.19	3.79	3.97	4.00	4.00	4.00	4.00
49.00	3.25	3.83	3.98	4.00	4.00	4.00	4.00
50.00	3.31	3.86	3.99	4.00	4.00	4.00	4.00
51.00	3.37	3.89	3.99	4.00	4.00	4.00	4.00
52.00	3.43	3.91	3.99	4.00	4.00	4.00	4.00
53.00	3.49	3.93	3.99	4.00	4.00	4.00	4.00
54.00	3.54	3.94	4.00	4.00	4.00	4.00	4.00
55.00	3.59	3.95	4.00	4.00	4.00	4.00	4.00
56.00	3.64	3.96	4.00	4.00	4.00	4.00	4.00
57.00	3.68	3.97	4.00	4.00	4.00	4.00	4.00
58.00	3.72	3.98	4.00	4.00	4.00	4.00	4.00
59.00	3.76	3.98	4.00	4.00	4.00	4.00	4.00
60.00	3.80	3.99	4.00	4.00	4.00	4.00	4.00

Table 5.7: Lambda=4, Require Time=60, Varying CPU

Servicetime	CPU=4	CPU=5	CPU=6	CPU=7	CPU=8
1	0.07	0.07	0.07	0.07	0.07
5	0.33	0.33	0.33	0.33	0.33
10	0.67	0.67	0.67	0.67	0.67
15	1.01	1.00	1.00	1.00	1.00
20	1.36	1.34	1.33	1.33	1.33
25	1.74	1.68	1.67	1.67	1.67
30	2.17	2.04	2.01	2.00	2.00
35	2.71	2.42	2.36	2.34	2.33
40	3.42	2.85	2.72	2.68	2.67
45	4.45	3.35	3.10	3.03	3.01
50	6.03	3.98	3.52	3.39	3.35
55	8.26	4.80	3.99	3.77	3.70
60	10.57	5.92	4.56	4.18	4.06

number of available CPUs.

The presented data provide significant information for the cloudlet parametric analysis. The fixed value of the cloudlet service time for the minimum task completion time, user drop probability of 0.01 and CPU utilization of 75% indicate that there is a relationship between the three parameters. This shall be explored in a subsequent chapter.

Table 5.8: C=4, Lambda=4, Cavg, Requirement Varies

T_s	Treq=40	Treq=50	Treq=60	Treq=70	Treq=80
1	0.07	0.07	0.07	0.07	0.07
5	0.33	0.33	0.33	0.33	0.33
10	0.67	0.67	0.67	0.67	0.67
15	1.00	1.00	1.00	1.00	1.00
20	1.33	1.33	1.33	1.33	1.33
25	1.67	1.67	1.67	1.67	1.67
30	2.00	2.00	2.00	2.00	2.00
35	2.33	2.33	2.33	2.33	2.33
40	2.67	2.67	2.67	2.67	2.67
45		3.00	3.00	3.00	3.00
50		3.31	3.31	3.31	3.31
55			3.59	3.59	3.59
60			3.80	3.80	3.80
65				3.91	3.91
70				3.96	3.96
75					3.99
80					3.99

5.4 Performance Evaluation Data

In this section, we present data on the performance evaluation of MobiCoRE.

5.4.1 Task Completion Time

Table 5.9 presents the data on the task completion time under MobiCoRE for varying values of user arrival rate. The number of available CPUs for the experiment has been set to 4. Maximum number of users allowed in the cloudlet is set to 20. Initially, the user arrival rate does not affect the task completion time. This is because a majority of the computation is being performed by the mobile device, which is independent of the user arrival rate. However, as the cloudlet service time increases, increasingly more computation is being performed by the cloudlet. Consequently, the task completion time decreases for lower user arrival rates. This is because the lower arrival rate results in fewer user applications in the system. This in turn results in a smaller wait time and early vacation. Consequently, the task completion time is reduced for lower user arrival rates.

Based on the observation made using the previous table, Table5.10 shows the cloudlet component of the task completion time as a function of CPU resources. The user arrival rate for this experiment was set to 4. The maximum number of users allowed in the cloudlet was set to 20. It can be observed that increasing the number of CPU resources results in an improved cloudlet response time. This clearly indicates that additional computational resources can improve the performance of the cloudlet. In the case of MobiCoRE, these additional resources are added through mobile-device-based augmentation instead of adding additional CPU cores.

Table 5.9: C=4, Require Time=60, Tresponse Varying Lambda

Tserv	**λ=4**	**λ=5**	**λ=6**	**λ=7**	**λ=8**	**λ=9**	**λ=10**
13	201	201	201	201	202	202	203
14	198	198	198	199	199	199	200
15	195	195	195	196	196	197	198
16	192	192	193	193	194	195	196
17	189	189	190	190	191	193	195
18	186	187	187	188	189	191	195
19	183	184	184	185	187	190	195
20	180	181	182	183	186	190	196
21	177	178	179	181	184	190	199
22	175	175	177	179	184	191	203
23	172	173	174	178	183	193	207
24	169	170	172	176	184	197	213
25	166	168	170	176	186	201	219
26	163	165	169	175	188	206	224
27	161	163	167	176	191	212	229
28	158	160	166	177	196	217	234
29	155	158	165	179	201	223	239
30	153	156	165	181	206	228	242
31	150	155	165	185	212	233	246
32	148	153	165	189	217	238	249
33	145	152	167	194	223	242	251
34	143	151	169	200	228	245	254
35	141	150	172	205	233	248	256
36	138	150	175	211	237	251	258
37	136	150	179	217	241	254	260
38	135	150	184	222	245	256	262
39	133	151	189	227	248	258	263
40	131	153	194	232	251	260	265
41	130	155	200	236	254	262	267
42	129	158	206	240	256	264	268
43	128	161	212	244	258	265	269
44	127	165	217	247	261	267	271
45	127	170	222	251	263	269	272
46	127	175	227	253	265	270	273
47	127	180	232	256	266	272	275
48	128	185	236	259	268	273	276
49	129	191	240	261	270	274	277
50	131	197	244	263	271	276	278
51	133	203	248	265	273	277	280
52	136	208	251	267	274	278	281
53	139	214	254	269	276	280	282
54	143	219	257	271	277	281	283
55	147	224	259	272	279	282	284
56	151	229	262	274	280	283	286
57	156	233	264	276	281	284	287
58	161	237	266	277	282	286	288
59	166	241	268	279	284	287	289
60	171	245	270	280	285	288	290

Table 5.10: C=4, Require Time=60, Tresponse Varying CPU

Servicetime	**CPU=4**	**CPU=5**	**CPU=6**	**CPU=7**	**CPU=8**
1	1.00	1.00	1.00	1.00	1.00
5	5.00	5.00	5.00	5.00	5.00
10	10.02	10.00	10.00	10.00	10.00
15	15.10	15.01	15.00	15.00	15.00
20	20.39	20.07	20.01	20.00	20.00
25	26.10	25.23	25.04	25.01	25.00
30	32.61	30.60	30.14	30.03	30.01
35	40.58	36.35	35.34	35.08	35.02
40	51.25	42.77	40.74	40.20	40.05
45	66.89	50.30	46.49	45.42	45.12
50	91.00	59.67	52.77	50.83	50.25
55	126.58	72.07	59.92	56.54	55.48
60	171.00	89.14	68.39	62.69	60.88

Finally, Table5.11 shows the cloudlet response time as a function of service requirement. The user arrival rate was set to 4, the number of cloudlet CPU cores was set to 4, the maximum number of allowed users in the cloudlet was set to 20, and the service requirement for 240 users was generated using a Poisson distribution around the average value given as the heading of each column. It can be observed that increasing the service requirement dramatically increases the cloudlet response time. This is because users very slowly vacate the system. The table also indicates that the reverse can be highly beneficial as well. Effectively, slightly reducing the service requirement of the users can dramatically decrease the cloudlet response time. This is the basic idea behind MobiCoRE, whereby the cloudlet service time is decreased in response to an increase in the number of users. A slight decrease in the cloudlet service time can dramatically increase the response time as a result of the reduced user wait time and early vacation of the system.

Table 5.11: C=4, Lambda=4, Time Require Varies

Tserv	Treq=40	Treq=50	Treq=60	Treq=70	Treq=80
1	1.00	1.00	1.00	1.00	1.00
5	5.00	5.00	5.00	5.00	5.00
10	10.02	10.02	10.02	10.02	10.02
15	15.10	15.10	15.10	15.10	15.10
20	20.39	20.39	20.39	20.39	20.39
25	26.10	26.10	26.10	26.10	26.10
30	32.61	32.61	32.61	32.61	32.61
35	40.58	40.58	40.58	40.58	40.58
40	51.25	51.25	51.25	51.25	51.25
45		66.89	66.89	66.89	66.89
50		91.00	91.00	91.00	91.00
55			126.58	126.58	126.58
60			171.00	171.45	171.45
65				220.44	220.44
75					306.16
80					342.04

5.4.2 Number of Users Served

In addition to the cloudlet response time and the task completion time for mobile applications, an important performance parameter for the cloudlet is the number of users served by the cloudlet. In this scenario, we collected data mainly concerned with the number of users served by the cloudlet. Table 5.12 presents data on the number of users served by the cloudlet for a 4-core system with a service requirement of 60 seconds, and the maximum number of users in the cloudlet was set to 20. We find that for all values of the arrival rate, there is a point whereby user drop begins. The user drop starts earlier for higher values of the arrival rate. This indicates that if the cloudlet does not use the resource augmentation option by exploiting mobile device resources, the number of users served will be significantly reduced, as indicated by the higher arrival rates, which is equivalent to longer service times.

Table 5.13 presents the number of users served as the cloudlet resources are increased. The user arrival rate was set to 4 users per minute, with a service requirement of 60 seconds, and the maximum number of users in the cloudlet was set to 20. It can be observed that higher resources lead to fewer dropped users.

Table 5.12: C=4, Require Time=60, User Served Varying Service Time

servicetime	**λ=4**	**λ=5**	**λ=6**	**λ=7**	**λ=8**	**λ=9**	**λ=10**
13	480	600	720	840	960	1080	1200
14	480	600	720	840	960	1080	1200
15	480	600	720	840	960	1080	1200
16	480	600	720	840	960	1080	1200
17	480	600	720	840	960	1080	1199
18	480	600	720	840	960	1080	1198
19	480	600	720	840	960	1079	1197
20	480	600	720	840	960	1079	1193
21	480	600	720	840	960	1077	1186
22	480	600	720	840	959	1074	1175
23	480	600	720	840	958	1069	1159
24	480	600	720	840	957	1062	1138
25	480	600	720	839	954	1051	1112
26	480	600	720	839	950	1036	1082
27	480	600	720	838	944	1017	1051
28	480	600	720	836	935	995	1019
29	480	600	719	834	924	971	987
30	480	600	719	830	910	946	957
31	480	600	718	825	894	920	927
32	480	600	718	818	876	894	899
33	480	600	716	810	856	869	872
34	480	600	714	800	836	845	847
35	480	600	712	788	815	821	823
36	480	599	708	774	795	799	800
37	480	599	703	760	775	778	778
38	480	598	697	745	756	758	758
39	480	597	691	729	737	738	738
40	480	596	683	713	719	720	720
41	480	595	674	698	702	702	702
42	480	593	664	682	685	686	686
43	480	590	653	667	669	670	670
44	480	587	642	653	654	655	655
45	479	584	631	639	640	640	640
46	479	579	619	625	626	626	626
47	479	574	608	612	613	613	613
48	478	569	596	600	600	600	600
49	478	563	585	587	588	588	588
50	477	556	574	576	576	576	576
51	476	549	563	565	565	565	565
52	475	541	553	554	554	554	554
53	474	533	543	543	543	543	543
54	472	526	533	533	533	533	533
55	470	518	523	524	524	524	524
56	468	510	514	514	514	514	514
57	465	502	505	505	505	505	505
58	462	494	496	497	497	497	497
59	459	486	488	488	488	488	488
60	455	478	480	480	480	480	480

Table 5.13: C=4, Require Time=60, Users Served Varying CPU

Servicetime	CPU=4	CPU=5	CPU=6	CPU=7	CPU=8
5	480.00	480.00	480.00	480.00	480.00
10	480.00	480.00	480.00	480.00	480.00
15	480.00	480.00	480.00	480.00	480.00
20	480.00	480.00	480.00	480.00	480.00
25	480.00	480.00	480.00	480.00	480.00
30	480.00	480.00	480.00	480.00	480.00
35	479.99	480.00	480.00	480.00	480.00
40	479.91	480.00	480.00	480.00	480.00
45	479.38	479.98	480.00	480.00	480.00
50	477.07	479.88	479.99	480.00	480.00
55	469.95	479.47	479.96	480.00	480.00
60	455.02	478.10	479.84	479.98	480.00

This is effectively the same as saying that resource augmentation can lead to improved cloudlet performance, namely, additional mobile device resources can improve the number of users served.

Finally, Table 5.14 shows the impact of varying the service requirement on the number of users served. The cloudlet resources were set to 4 cores. The user application arrival rate was 4 users per minute, and the maximum number of users that can be accommodated in the cloudlet was set to 20.

Table 5.14: C=4, Lambda=4, Require Time Varies

Tserv	Treq=40	Treq=50	Treq=60	Treq=70	Treq=80
1	480.00	480.00	480.00	480.00	480.00
5	480.00	480.00	480.00	480.00	480.00
10	480.00	480.00	480.00	480.00	480.00
15	480.00	480.00	480.00	480.00	480.00
20	480.00	480.00	480.00	480.00	480.00
25	480.00	480.00	480.00	480.00	480.00
30	480.00	480.00	480.00	480.00	480.00
35	479.99	479.99	479.99	479.99	479.99
40	479.91	479.91	479.91	479.91	479.91
45		479.38	479.38	479.38	479.38
50		477.07	477.07	477.07	477.07
55			469.95	469.95	469.95
60			455.02	455.02	455.02
65				432.94	432.94
70				407.65	407.65
75					382.64
80					359.50

It can be observed that the increase in the service requirement does not affect the number of users served by the cloudlet as long as the cloudlet service time is kept constant. This is understandable because the cloudlet is not providing additional resources by fixing the service time, irrespective of the demand of the mobile devices.

5.4.3 Average Number of Users in Cloudlet

We highlighted that the higher response time of the cloudlet results from the increased number of user applications in the cloudlet. To confirm this observation, we present data on the average number of users in the cloudlet for different values of the service time. Table 5.15 presents the cloudlet average number of users in the system as a function of service time and user arrival rate. The cloudlet CPU resources were set to 4, with a maximum user application capacity of 20 users. The service requirement of the mobile devices was set to 60 seconds. The presented data clearly show that the increased service time from the cloudlet results in an increased number of users in the cloudlet. Similarly, the increased user application arrival rate results in an increased number of users, even when the service time is kept constant. This increase in the number of users confirms our results of higher wait times as the cloudlet service time increases or the user arrival rate increases. The increased wait time results in an increased response time. To improve the response time from the cloudlet, the cloudlet must vacate the user applications earlier by adjusting the service time. This eventually improves the performance of the cloudlet.

To study the impact of the service requirement on the cloudlet average number of users in the system, we present the data in Table5.16. The data show the average number of users in the cloudlet as a function of service time and mobile application requirements. It can be observed that higher user requirements do not affect the number of users in the cloudlet as long as the service time from the cloudlet is kept constant. MobiCoRE

Table 5.15: Lambda=4, Require Time =60, Number of Arrival User Vary

Servicetime	**λ=4**	**λ=5**	**λ=6**	**λ=7**	λ=8	**λ=9**	**λ=10**
13.00	0.87	1.09	1.32	1.56	1.82	2.10	2.42
14.00	0.94	1.18	1.43	1.70	1.99	2.32	2.71
15.00	1.01	1.27	1.54	1.84	2.17	2.56	3.03
16.00	1.08	1.36	1.66	1.99	2.37	2.83	3.42
17.00	1.15	1.45	1.78	2.15	2.59	3.14	3.88
18.00	1.22	1.54	1.91	2.32	2.83	3.50	4.45
19.00	1.29	1.64	2.04	2.50	3.10	3.93	5.16
20.00	1.36	1.74	2.17	2.71	3.42	4.45	6.03
21.00	1.43	1.84	2.32	2.93	3.78	5.08	7.07
22.00	1.51	1.95	2.48	3.18	4.21	5.84	8.26
23.00	1.58	2.06	2.65	3.46	4.72	6.74	9.55
24.00	1.66	2.17	2.83	3.78	5.32	7.77	10.76
25.00	1.74	2.30	3.03	4.15	6.03	8.90	12.12
26.00	1.82	2.42	3.25	4.58	6.85	10.08	13.26
27.00	1.91	2.56	3.50	5.08	7.77	11.25	14.23
28.00	1.99	2.71	3.78	5.66	8.77	12.36	15.05
29.00	2.08	2.86	4.10	6.32	9.82	13.36	15.72
30.00	2.17	3.03	4.45	7.07	10.57	14.23	16.27
31.00	2.27	3.21	4.86	7.89	11.88	14.98	16.72
32.00	2.37	3.42	5.32	8.77	12.82	15.60	17.08
33.00	2.48	3.64	5.84	9.69	13.67	16.12	17.38
34.00	2.59	3.88	6.43	10.61	14.41	16.55	17.62
35.00	2.71	4.15	7.07	11.51	15.05	16.91	17.83
36.00	2.83	4.45	7.77	12.36	15.60	17.20	18.01
37.00	2.96	4.79	8.51	13.15	16.07	17.46	18.16
38.00	3.10	5.16	9.29	13.86	16.46	17.67	18.29
39.00	3.25	5.57	10.08	14.50	16.79	17.85	18.40
40.00	3.42	6.03	10.57	15.05	17.08	18.01	18.50
41.00	3.59	6.53	11.63	15.54	17.32	18.14	18.59
42.00	3.78	7.07	12.36	15.96	17.53	18.26	18.67
43.00	3.99	7.65	13.04	16.32	17.71	18.37	18.74
44.00	4.21	8.26	13.67	16.63	17.87	18.46	18.80
45.00	4.45	8.90	14.23	16.91	18.01	18.55	18.86
46.00	4.72	9.55	14.74	17.14	18.13	18.62	18.91
47.00	5.01	10.21	15.20	17.35	18.24	18.69	18.96
48.00	5.32	10.63	15.60	17.53	18.34	18.75	19.00
49.00	5.66	11.51	15.96	17.69	18.42	18.81	19.04
50.00	6.03	12.12	16.27	17.83	18.50	18.86	19.08
51.00	6.43	12.71	16.55	17.96	18.57	18.90	19.11
52.00	6.85	13.26	16.79	18.07	18.64	18.95	19.14
53.00	7.30	13.76	17.01	18.17	18.70	18.99	19.17
54.00	7.77	14.23	17.20	18.26	18.75	19.02	19.20
55.00	8.26	14.66	17.38	18.35	18.80	19.06	19.23
56.00	8.77	15.05	17.53	18.42	18.85	19.09	19.25
57.00	9.29	15.40	17.67	18.49	18.89	19.12	19.27
58.00	9.82	15.72	17.79	18.56	18.93	19.15	19.29
59.00	10.35	16.01	17.91	18.61	18.97	19.18	19.31
60.00	10.91	16.27	18.01	18.67	19.00	19.20	19.33

Table 5.16: C=4, Lambda=4, Require Time Varies

Tserv	**Treq=40**	**Treq=50**	**Treq=60**	**Treq=70**	**Treq=80**
1	0.07	0.07	0.07	0.07	0.07
5	0.33	0.33	0.33	0.33	0.33
10	0.67	0.67	0.67	0.67	0.67
15	1.01	1.01	1.01	1.01	1.01
20	1.36	1.36	1.36	1.36	1.36
25	1.74	1.74	1.74	1.74	1.74
30	2.17	2.17	2.17	2.17	2.17
35	2.71	2.71	2.71	2.71	2.71
40	3.42	3.42	3.42	3.42	3.42
45		4.45	4.45	4.45	4.45
50		6.03	6.03	6.03	6.03
55			8.26	8.26	8.26
60				10.57	10.57
65				13.26	13.26
70				15.05	15.05
75					16.27
80					17.08

transfers the partially completed applications to the mobile devices, irrespective of their service requirement. Therefore, additional requirements have no impact on the cloudlet system, and the number of users in the cloudlet system remain unaffected.

5.5 Conclusion

This chapter presented the methodology used for the collection of the data that have been used for the evaluation of MobiCoRE. We started by explaining the applied methodology. The data were presented in three steps. In the first step, we presented the data that have been used for the validation of the mathematical model. The data showed that the model and empirical results closely match. In the second step, we presented the data that were used for the extraction of the optimal parameters for the operation of MobiCoRE. In the final step, we presented the data that were used for the performance evaluation of the cloudlets under MobiCoRE.

The presented data are only a fraction of the total data collected for the experiments. We have not included all the data for the sake of brevity. In a subsequent chapter, we distill the collected data and present our findings in the form of a discussion.

CHAPTER 6 : RESULTS AND DISCUSSION

In this chapter, we present the data in graphical format and analyse in a cohesive and sequential manner to draw the logical conclusion. We start by verifying the correctness of the mathematical model. We derive the operational points of MobiCoRE where the best performance of the cloudlet can be achieved. Subsequently, we evaluate the performance of MobiCoRE using optimal parameters. Finally, we show that MobiCoRE is effective in enhancing the performance of cloudlets that have relatively large resources. The chapter is divided into three parts of MobiCoRE mathematical model validation, operational analysis,and performance enhancement of cloudlets using MobiCoRE framework.

The chapter is organized as follows. In Section 6.1, we present the model validation results along with a discussion on accuracy. Section 6.2 presents the parametric analysis and discussion on optimal parameter selection. Section 6.3 presents the performance analysis results and a discussion of MobiCoRE framework, and the chapter is concluded in Section 6.4.

6.1 MobiCoRE Model Validation

The first step of the results analysis is the validation of the mathematical model of MobiCoRE framework. In this section, we answer the following question: Does the developed mathematical model accurately represent the MobiCoRE framework? The answer to this question establishes the correctness of the framework, which has been used for data collection and for the remainder of the results and discussion. For this purpose, we implemented the mathematical model of MobiCoRE framework in Matlab to obtain analytical results. The empirical results were obtained from the real prototype implementation of MobiCoRE. The implementation details of MobiCoRE framework were explained in a previous chapter.

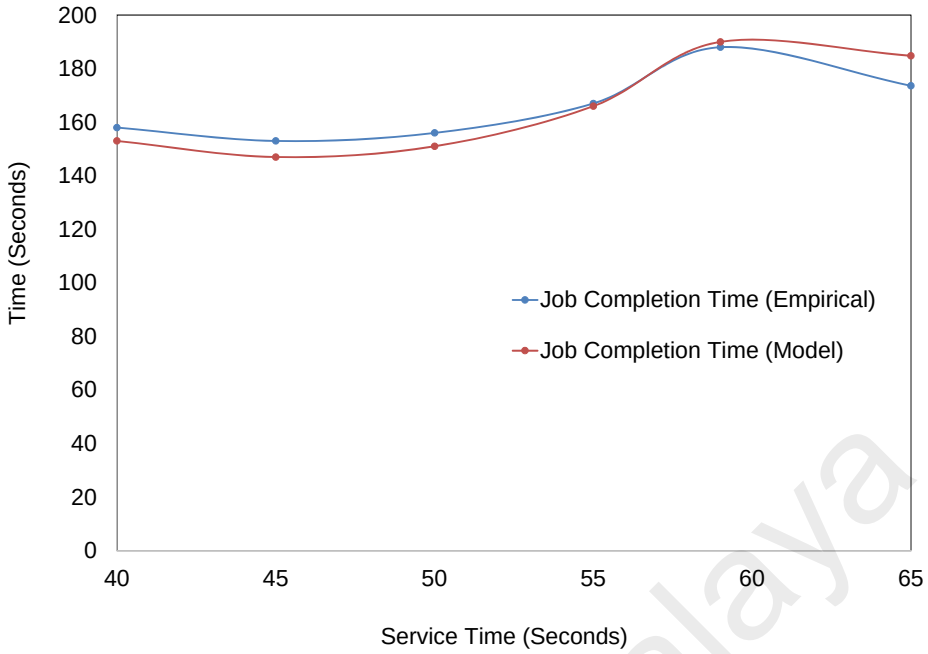

Figure 6.1: Task Completion Time

6.1.1 Application Response Time

We start by validating the application response time, as achieved by the MobiCoRE cloudlet implementation and the mathematical model. Figure 6.1 shows the average task completion time $\bar{T}$ for an arrival rate of 4 and varying service time $\bar{T}_s$. It can be observed that the empirical results almost perfectly match the model results, although the random values were generated over a small sample space of 60. The empirical results are within 6% of the analytical results, with most of the differences being less than 3%.

6.1.1.1 Statistical Validation

We further validate the correctness of the model by comparing the results using statistical analysis. We have conducted paired sample t-tests for this purpose, and the results are presented in Table 6.1. The Pearson correlation in this case is approximately one, suggesting that the empirical results and MobiCoRE mathematical model results closely match. Similarly, the t-stat and t-critical values are greater than 0.005, with t-critical val-

Table 6.1: Paired Sample T-test: Task Completion Time

Pearson Correlation	df	t-Stat	P(T<=t) two-tail	t- Critical two-tail
0.965483791	5	0.237111308	0.821979158	2.570581836

ues being higher than the t-stat values, thus confirming that the two results are similar. A similar analysis has been conducted for the cloudlet response time where the mobile computation part has been ignored. The results are presented in Table 6.2 and show a near perfect match between the empirical and analytical results.

Table 6.2: Paired sample T-test: Cloudlet response time

Pearson Correlation	df	t-Stat	P(T<=t) two-tail	t-Critical two-tail
0.99834276	5	0.237111308	0.821979158	2.570581836

6.1.2 Average Number of Users

Similar to the trend exhibited by the application response time, a near perfect match was observed in the case of the average number of applications in the MobiCoRE framework $\bar{N}$, as shown in Figure 6.2. In this case, the empirical results have an average deviation of 0.21 from the analytical results. The near perfect match of the results suggests that the proposed MobiCoRE mathematical model can be used for the analysis of MobiCoRE.

Table 6.3: Paired Sample T-Test: Number of Avg User in the Cloudlet

Pearson Correlation	df	t-Stat	P(T<=t) two-tail	t-Critical two-tail
0.999292095	5	1.723121278	0.145475573	2.570581836

6.1.2.1 Statistical Validation

For the sake of completeness, we perform a paired sample t-test using these data, and the results are presented in Table 6.3. The Pearson Correlation indicates an even better match in this case, with a value of 0.99, thus confirming that the results are in good agreement.

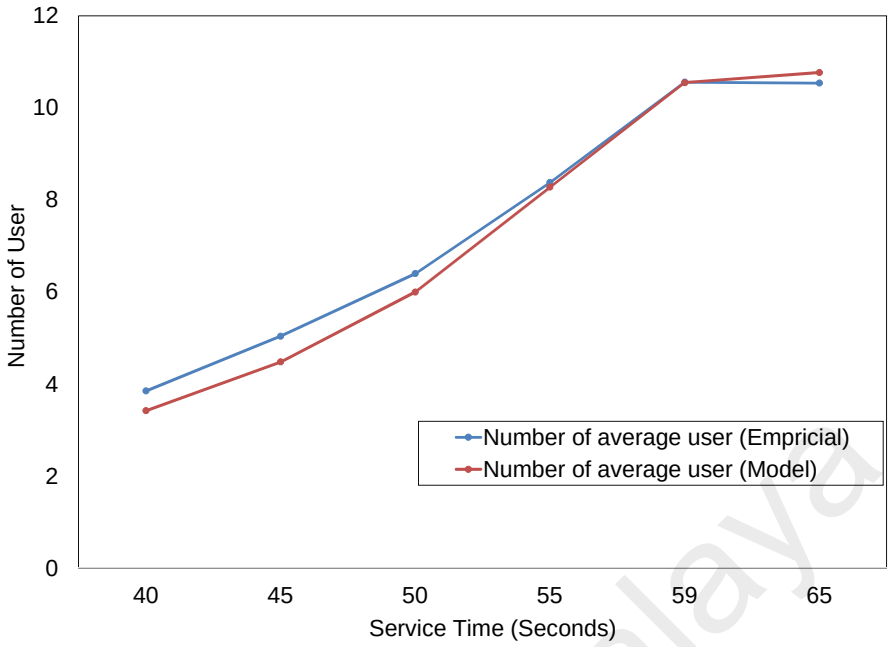

Figure 6.2: Average Number of Users in Cloudlet

6.1.3 Number of Users and Response Time

Finally, we present the results to verify the claim that the total response time for the service of a mobile application is dependent upon the number of users.

The results of one run of the implementation are shown in Figure 6.3. The graph shows the application execution time and the corresponding number of users in the cloudlet averaged over a minute for a Poisson arrival rate of on average 4 users per minute. The application requirement time was kept constant and equal to 65 seconds. It can be observed that although the application requirement as well as the service time are constant, the task completion time of the applications varies as a function of the number of applications in the system. This is because of the increased waiting time faced by the applications. The average task completion time for this experiment is 173 seconds, and the average time computed using the model for the same input values is 184 seconds. The error margin is less than 6%, which is acceptable for probabilistic models.

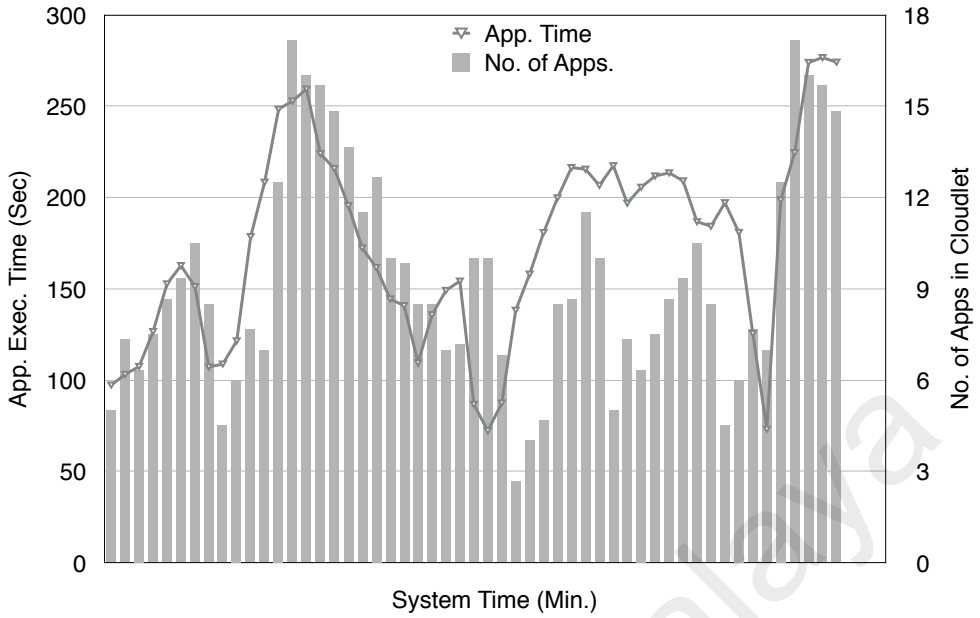

Figure 6.3: Implementation Exec. Time and No. of Apps.

The plot of the number of users exactly following the bar graph suggests that our claim is valid. It can further be noted that a higher number of users in the system at any particular time results in the impact of those users being experienced for extended periods of time after the event. Therefore, the number of users in the system must be kept under control at all times to ensure optimal performance. In other words, active user application management is required inside the cloudlet, which is a characteristic of MobiCoRE.

Having validated the MobiCoRE mathematical model, we now use the MobiCoRE model-generated results and perform the operational analysis of MobiCoRE. We use the analysis to identify the parameters for the optimal operational point of MobiCoRE.

6.2 MobiCoRE Operational Analysis

MobiCoRE maintains a desired service time such that the average service time of the cloudlet is set to an optimal point for a given arrival rate and given cloudlet resources. For this purpose, the average number of users that corresponds to the respective optimal

Table 6.4: Parameters and Value Ranges

Parameter	Tested Value Range
λ	$4 \rightarrow 10$ per minute
$\bar{T_s}$	$10 \rightarrow 80$ seconds
c	$4 \rightarrow 16$
K	20
$\bar{f}$	4
f^i	Generated using poisson dist.
$\bar{T_{req}}$	$60 \rightarrow 80$ seconds
$\bar{T_{req}^i}$	Generated using poisson dist.
T_{total}	Time of kth synchronization

service time in the system is maintained. In this section, we use the model presented in Chapter 4 to compute the optimal operational point for MobiCoRE for given arrival rates and resources.

We start by analyzing the task completion time and its components. Based on the observations of the task completion time, we further analyse the role of the average cloudlet utilization and the percentage of users served by the cloudlet on the optimal operation of MobiCoRE. We are interested in determining the condition for the optimal operational value for the average service time, independent of the application service requirements, which are not known a priori and which are difficult to estimate. The parameters and their respective ranges used in this analysis are shown in Table 6.4. Throughout the analysis, the service time requirements of the mobile device applications as well as the cloudlet-to-mobile-device speed-up ratio was generated using a Poisson distribution. Similarly, the number of users arriving per minute was also generated using a Poisson distribution with the desired λ as the average.

6.2.1 Task Completion Time

Figure 6.4 shows the average job completion time and its component times $\bar{T_m}$ and $\bar{T_c}$. The advantage of using MobiCoRE for different values of the service time is also shown. The value of the advantage is the difference between the job completion time

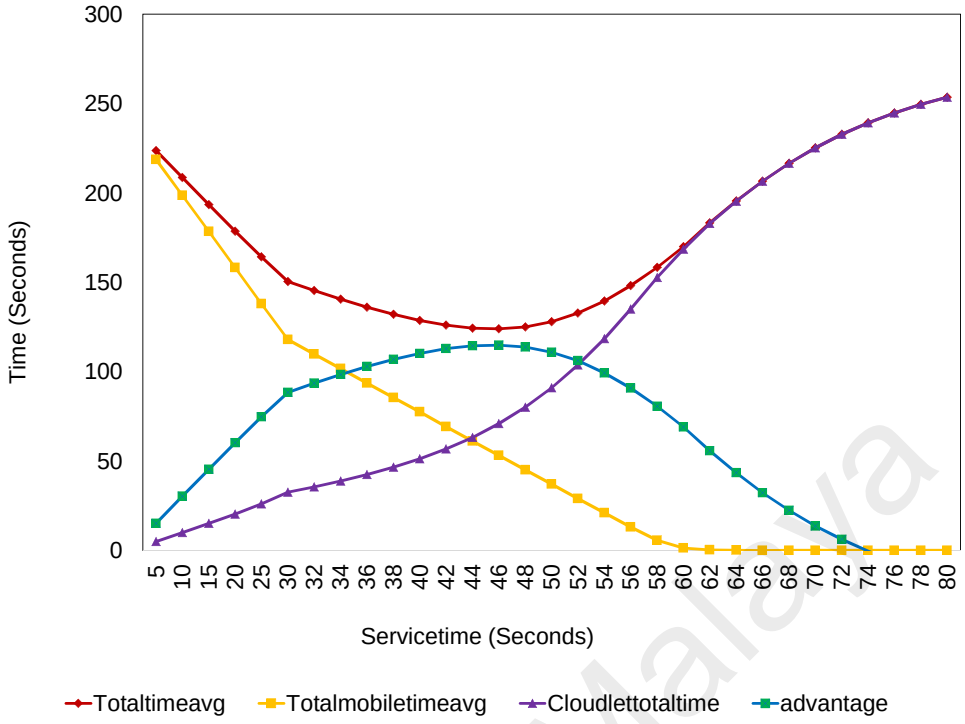

Figure 6.4: $\bar{T}$, $\bar{T_m}$ and $\bar{T_c}$ for varying $\bar{T_s}$

when the entire task is executed in the cloudlet and the job completion time when mobile resources are used to complete the application after obtaining the service time from the cloudlet. This value is plotted on the X-axis. Effectively, this is the benefit for the mobile device applications when a MobiCoRE-based cloudlet is providing the service. For this result, the parameters used are $\bar{T_{req}} = 60$ seconds, $C = 4, K = 20$, and $\lambda = 4$.

It can be observed that the task completion time initially decreases with increasing cloudlet service time because of the increased number of computations being performed by the cloudlet. However, after reaching a minimum value, the time starts increasing as a result of the increased number of users in the cloudlet and longer wait times. This suggests that there is an optimal service time and hence a number of cloudlet users whereby the cloudlet can provide the best service for the mobile devices. Two points are noteworthy in this graph. First, there is a service time for which the time spent on application execution by the mobile device is equal to the time spent by the cloudlet. Second, for a particular value of the service time, the maximum advantage is achieved for mobile applications

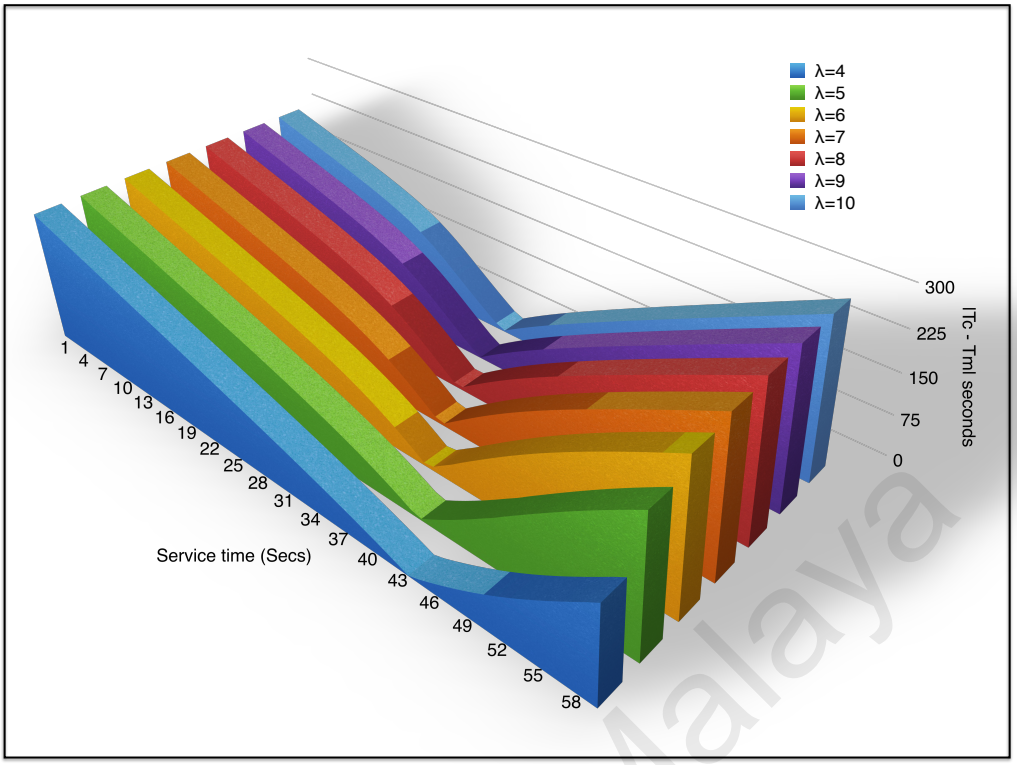

Figure 6.5: $\bar{T}_s$ for $\bar{T}_m = \bar{T}_c$

under MobiCoRE. We analyse both operational points in greater detail.

6.2.1.1 Service Time at $\bar{T}_m = \bar{T}_c$

Figure 6.5 shows the absolute difference between the time that the mobile device spent on the computation and the time spent by the cloudlet as a function of cloudlet service time for different values of λ. If can be observed that for every λ, the difference becomes approximately zero, which is the point where $\bar{T}_m = \bar{T}_c$. For example, for $\lambda = 4$, the above condition is satisfied for a cloudlet service time of 44 seconds. The average time taken by the cloudlet to complete $(44/60) * 100 = 73\%$ of the application is equal to the average time taken by the mobile devices to complete the remaining 27% of the application. It can also be observed that the difference sharply increases after reaching the minimum value. The sharp increase can be attributed to the sharp increase in the waiting time for larger values of service times. This indicates that a small decrease in service time can result in a significantly higher advantage in terms of time. This observation is further

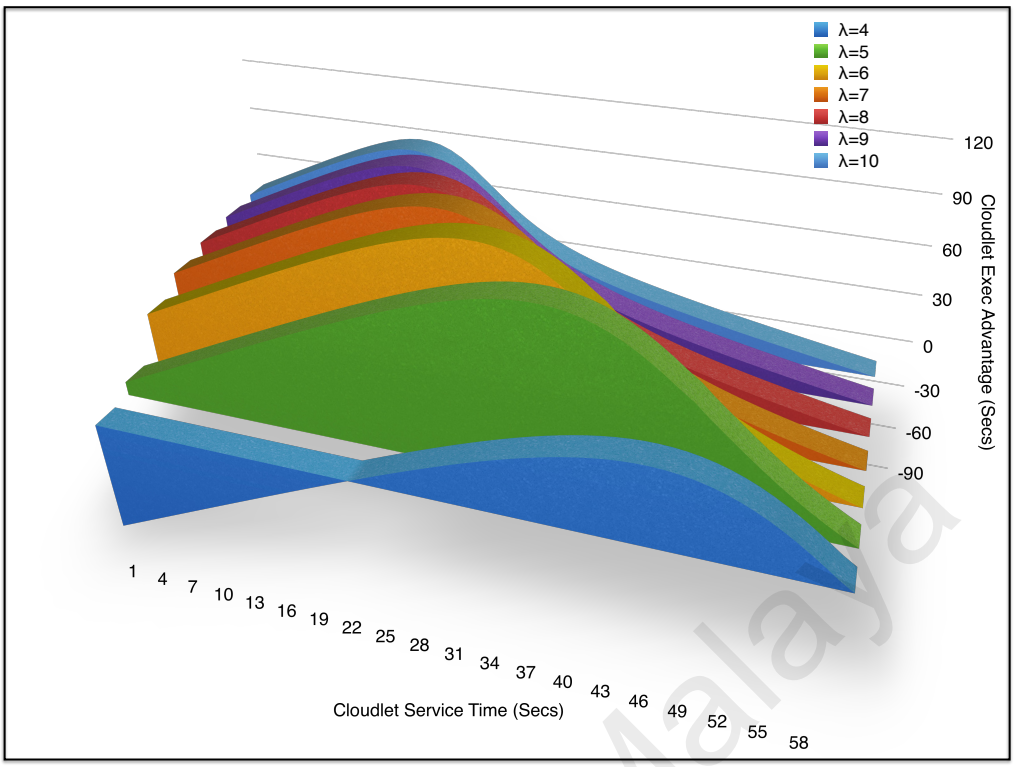

Figure 6.6: Improvement in Total Time for Maximum Advantage

confirmed in a subsequent section.

6.2.1.2 Service Time at Max. Advantage

Similar to the service time whereby the mobile and cloudlet times are equal, there is a service time for all values of λ where the maximum benefit of using MobiCoRE is achieved, as shown in Figure 6.6. The graph shows the advantage resulting from the use of MobiCoRE in terms of service time as a function of service time from the cloudlet for different values of λ. It can be observed that there is a specific service time for all values of λ whereby the maximum advantage is achieved. Further, the service time corresponding to the maximum advantage resulting from the use of MobiCoRE decreases with increasing arrival rate λ. The service time whereby the maximum benefit is achieved for an arrival rate of $\lambda = 4$ is 45 seconds, whereas for $\lambda = 10$, the value decreases to 18 seconds.

It can also be observed that the decrease in advantage is near exponential for ser-

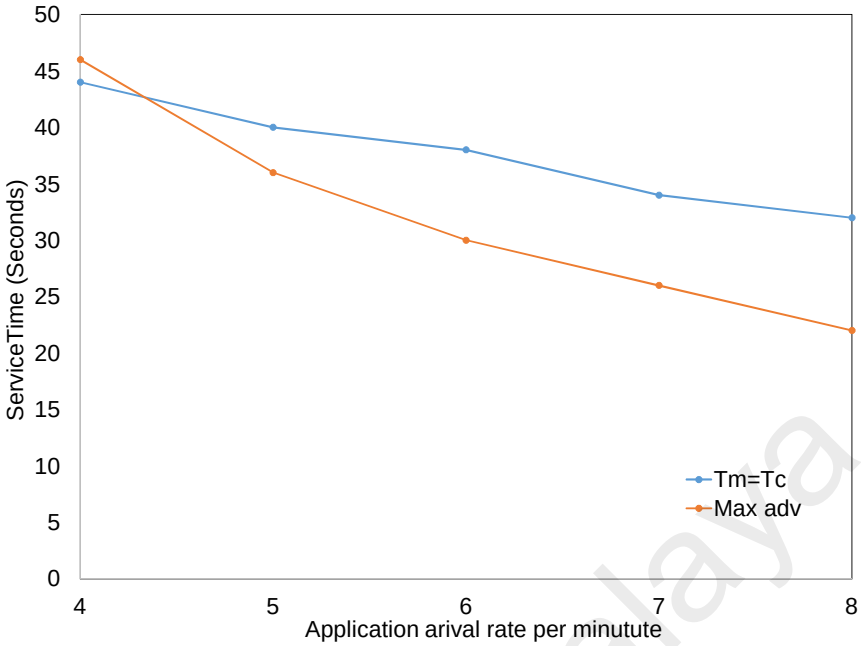

Figure 6.7: $\bar{T}_s$ for Max. Adv. and $\bar{T}_m = \bar{T}_c$

vice times higher than the service time that provides the maximum advantage. This confirms the earlier observation that the waiting time in the cloudlet increases sharply for higher cloudlet service times. Therefore, a small reduction in the cloudlet service time can significantly improve the cloudlet performance. This observation will be revisited in Section 6.3.

6.2.1.3 Comparison of Service Time at $\bar{T}_m = \bar{T}_c$ and Min. Task Time

Figure 6.7 shows the average service time values where the maximum advantage is achieved as well as the values where the cloudlet and mobile times are equal as a function of λ. It can be observed that the increasing value of λ decreases the values of the average service time in both cases. This is because the cloudlet must vacate the mobile applications earlier from the system to accommodate more users while keeping the wait time low. Furthermore, the decrease in the average service time for the maximum advantage point is sharp compared to the decrease in service time for $\bar{T}_m = \bar{T}_c$. This shows

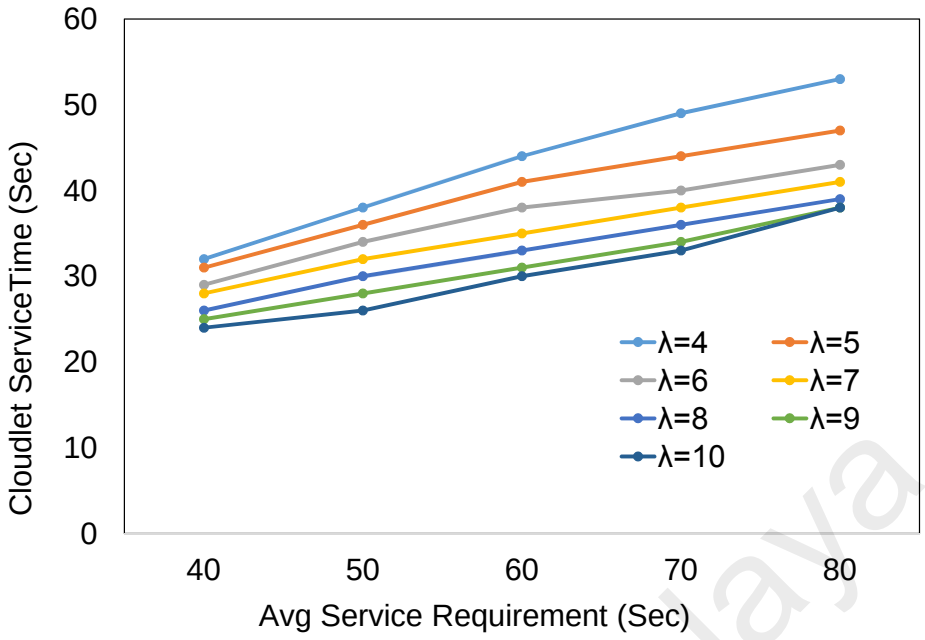

Figure 6.8: $\bar{T}$ for Varying $\bar{T_{req}}$

that for $\bar{T_m} = \bar{T_c}$,the cloudlet does more work (leaving less work for the mobile device) compared to the maximum benefit point. Consequently, at $\bar{T_m} = \bar{T_c}$, the load on the mobile device is reduced, which is desirable. Therefore, with reference to the load on the mobile device, it is more desirable to operate the cloudlet at the point where $\bar{T_m} = \bar{T_c}$. We further analyse the cloudlet average service time for the condition $\bar{T_m} = \bar{T_c}$.

6.2.2 Analysis of Service Time at $\bar{T_m} = \bar{T_c}$

Figure 6.8 shows the cloudlet average service time at $\bar{T_m} = \bar{T_c}$ plotted against the average service requirement of the mobile devices for different values of the arrival rate. It can be observed that the average service time varies almost linearly with the service requirement. This means that the cloudlet average service time at $\bar{T_m} = \bar{T_c}$ is dependent on the value of the application requirement. This can be observed from equation 6.1, which has been derived from $\bar{T_m} = \bar{T_c}$ and can be solved iteratively to compute the required cloudlet service time for different values of the required service time such that the above

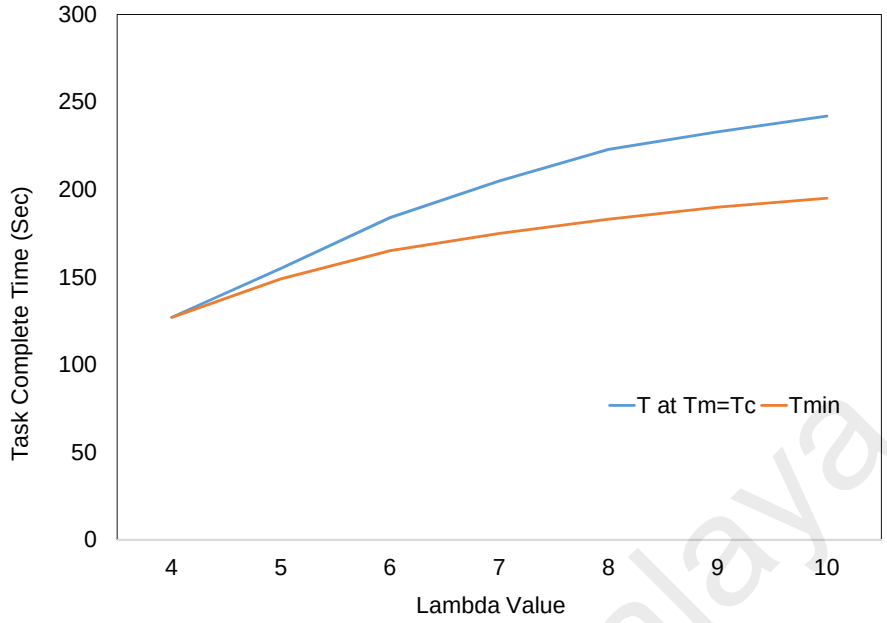

Figure 6.9: Comparison of $\bar{T}^{\bar{T}_m=\bar{T}_c}$ and $\bar{T_{min}}$

condition is met.

$$\bar{T}_s^{\bar{T}_m=\bar{T}_c} = \frac{\bar{T_{req}} * f - \bar{T_w}}{1+f} \tag{6.1}$$

Although $\bar{T}_s^{\bar{T}_m=\bar{T}_c}$ is an important operational point for MobiCoRE-based cloudlets, its dependence on the service requirement $\bar{T_{req}}$ makes it a less useful parameter because $\bar{T_{req}}$ is not known by the mobile devices nor the cloudlet, even at the time of operation. In addition, the task completion time at $\bar{T}_s^{\bar{T}_m=\bar{T}_c}$ is not a minimum, and a smaller completion time can be obtained by the MobiCoRE-based cloudlet, as shown in Figure 6.9. The figure shows the minimum $\bar{T}$ (average task completion time) and $\bar{T}$ for the service time of equation 6.1 for varying arrival rates. Other parameters are set to $\bar{T_{req}} = 60, c = 4$ and $K = 20$. The difference between the two values indicates that the minimum task completion time is not achieved at $\bar{T}_s^{\bar{T}_m=\bar{T}_c}$. We now turn our attention to the average service time where the average task completion time is minimized.

6.2.3 Analysis of Cloudlet Service Time at Minimum Task Time

We start the study of the service time at the minimum task time by observing its dependence on the service requirement. Figure 6.10 shows the average task completion time under MobiCoRE for varying cloudlet average service times and variable mobile application service requirements. The cloudlet resources and user arrival rate were set to $c = 4, K = 20, \lambda = 4$. It can be observed that the optimal operational point is not affected by the service requirement of the mobile applications. For all values of $\bar{T_{req}}$ and the given cloudlet resources, the minimum task completion time is achieved at the same average service requirement of $\bar{T_s} = 45$ seconds. The only affect of $\bar{T_{req}}$ is the constant increase in the task completion time, which is also obvious from the trendline equations, where the only significant change is in the constant factor.

The trend shown in Figure 6.10 has been observed for all values of λ and c as long as K is set sufficiently large and the cloudlet operates under stable conditions. This observa-

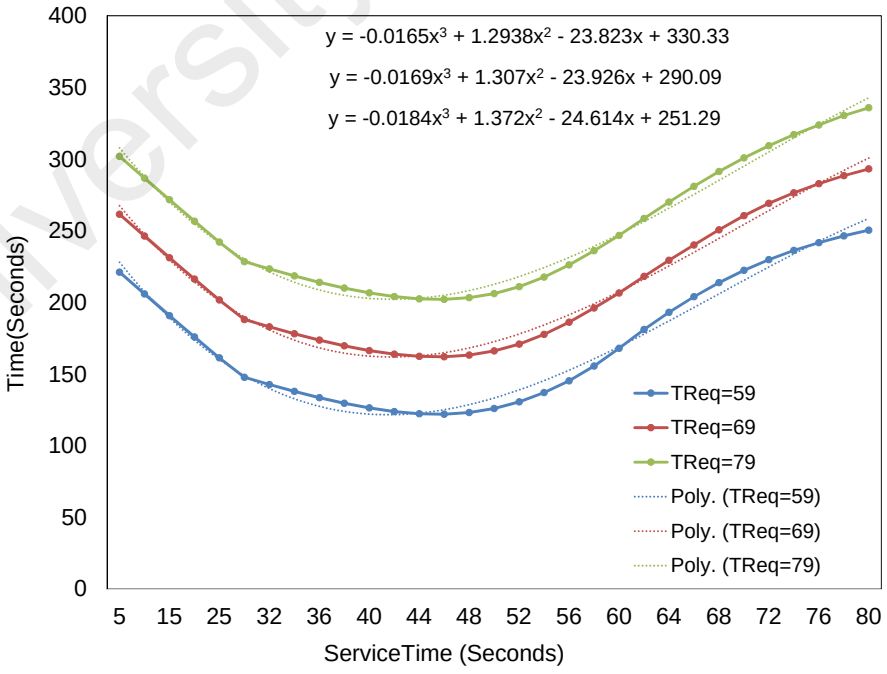

Figure 6.10: $\bar{T_s}$ for Varying $\bar{T_{req}}$

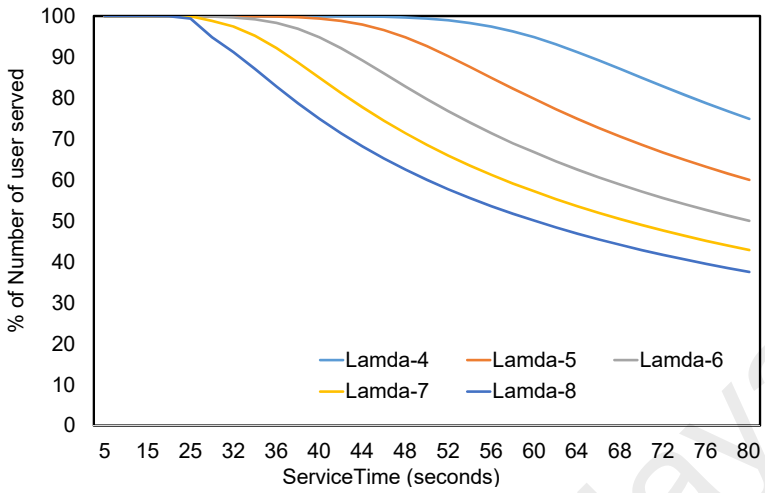

Figure 6.11: Users Served for Varying λ

tion leads us to the following question: Is there a specific behavior of the cloudlet system that can be used as a basis to derive an expression for the optimal cloudlet average service time? The answer to this question lies in the application rejection rate P_K and cloudlet utilization U. We now consider both variables.

6.2.3.1 Drop Rate and Minimum Task Time

To determine the optimal service time, we examine different parameters concerning cloudlet operation. We start with the application drop rate. Figure 6.11 shows the percentage of applications served for varying service times and different values of λ. It can be observed that as the service time increases, the number of users served also decreases. More importantly, we observe that application dropping starts at values that are close to the values of the maximum benefit for mobile applications from the cloudlet. We further analyse the user drop by observing the drop probability P_K.

P_K is the probability of a task being dropped by the cloudlet. This event occurs

when the number of tasks in the cloudlet is equal to the parameter K. Figure 6.12 shows the value of P_K as a function of λ at the average service time where the average task completion time is a minimum. It can be observed that the value of P_K is constant up to three decimal places. Note that the value remains constant for all values of the average service requirement. We use this observation to derive the expression for the optimal value of the average service time. Using equation 4.2, we obtain equation 6.2 as follows:

$$\begin{aligned} 0.999 &= (1 - P_K) \\ \implies 0.999 &= \left(1 - \frac{\lambda^K}{c^{K-c} c! \mu^K} P_0\right) \\ \implies \frac{1}{P_0} &= \frac{\lambda^K}{c^{K-c} c! \mu^K} 1000 \end{aligned} \tag{6.2}$$

Equation 6.2 can be solved iteratively using the known parameters of λ, c and K for the value of the only unknown μ. The average service time of the cloudlet for which

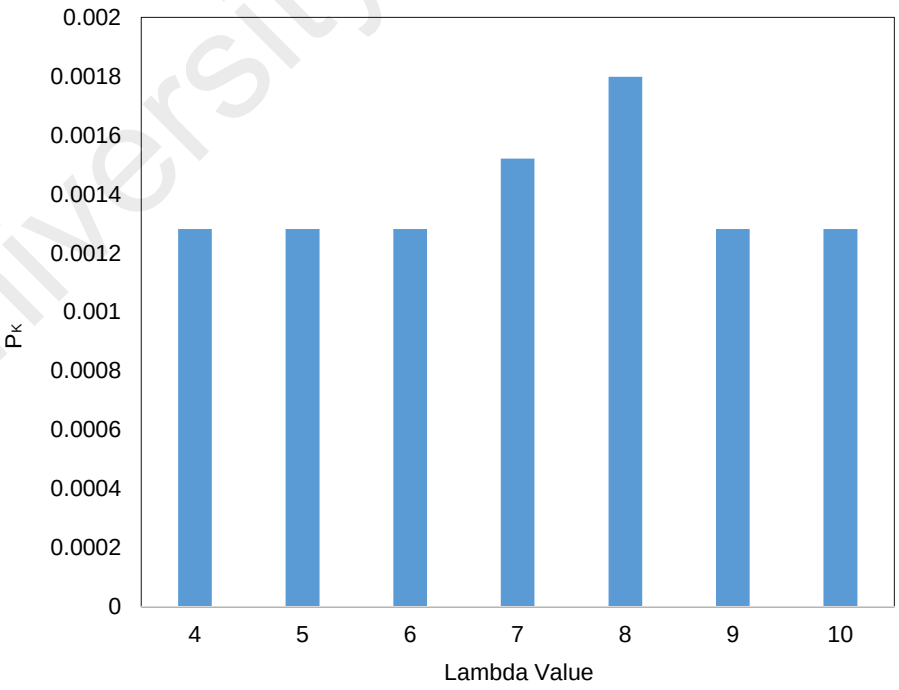

Figure 6.12: P_K for Varying λ

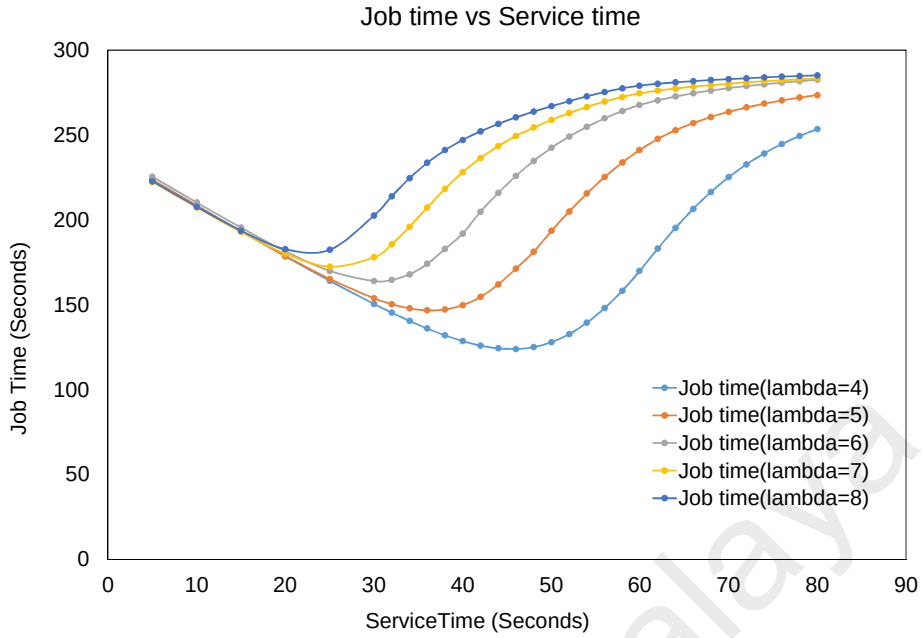

Figure 6.13: $\bar{T}_s$ for Varying $\bar{T_{req}}$

the condition $P_k = 0.001$ is satisfied can be computed using $\bar{T}_s = \frac{1}{\mu}$. Effectively, for any value of λ, c and K, we can compute the optimal cloudlet average service time for which the average task completion time is minimized independent of the task requirement. This result is of fundamental importance and can be used to tune the parameter $\bar{N}$ (which is a function of λ, c, K and μ) of the MobiCoRE-based cloudlet to determine the optimal operational point.

6.2.3.2 Cloudlet Utilization and Minimum Task Time

We now turn our attention to the second variable of cloudlet utilization U. We start by observing the optimal service time dependence on λ. Figure 6.13 shows the average task completion time under MobiCoRE for different values of the service time from the cloudlet for $c = 4, K = 20$, and $\bar{T_{req}} = 60$ and with varying λ. It can be observed that an increased value of Λ results in a reduced service time from the cloudlet for the minimum task completion time. This suggests that the optimal service time and the number of users

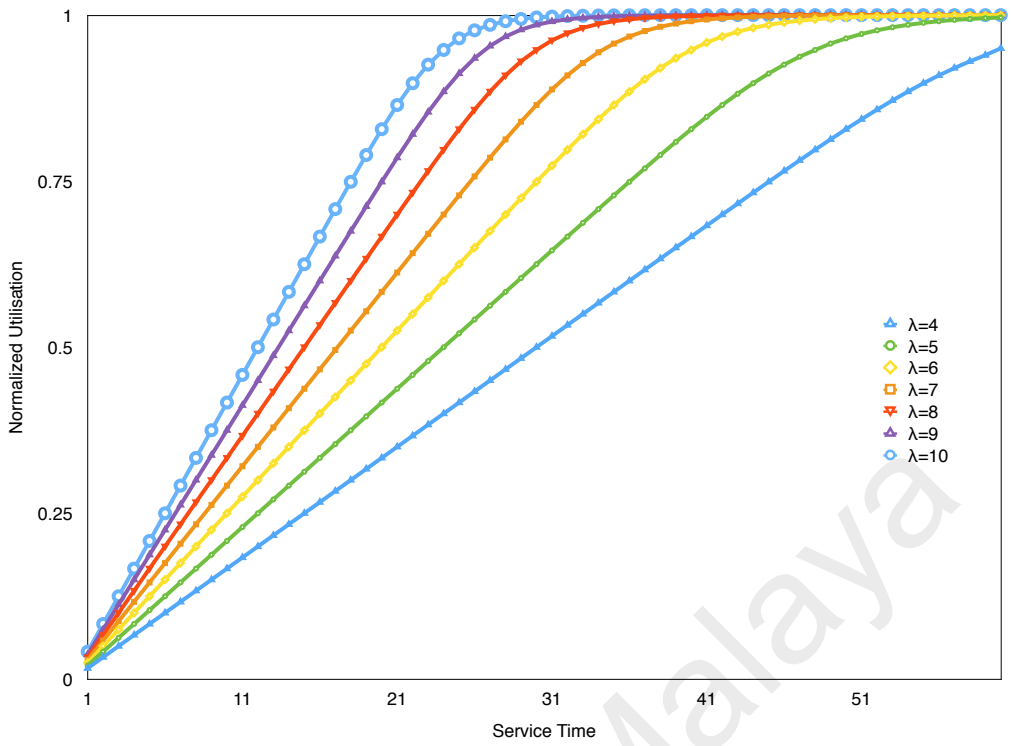

Figure 6.14: Cloudlet Utilization for Varying $\bar{T}_s$ and λ

in the cloudlet are inversely proportional to the mobile application task arrival rate. Furthermore, the task completion time increases with increasing arrival rate, thus suggesting that the number of users as well as the cloudlet utilization increase. This results in a longer wait time for the applications and a resulting increase in the task completion time.

Figure 6.14 shows the normalised utilization of the cloudlet for varying service times and different values of λ. It can be observed that increasing the service time results in increased cloudlet utilization, which eventually reaches the maximum value of 1. We further observe that the cloudlet utilization is almost constant at service time values whereby the maximum benefit for the mobile device is achieved. This can be observed in Figure 6.15, where the cloudlet utilization has been plotted for service times whereby the minimum task time is achieved. It can be observed that the cloudlet utilization is approximately 75% for the optimal service time. We use this observation to compute a simplified expression for the optimal service time that is conditionally valid and that can be used for

most of the realistic scenarios.

We know from queuing theory that $\bar{\lambda} = \mu\bar{c}$. Given that the value of K is sufficiently large, we can use the above equation and the facts identified in Figure 6.12 and Figure 6.16 to derive the expression for the optimal service time as follows:

$$\begin{aligned} \bar{\lambda} &= \mu\bar{c} \\ \implies \lambda(1-P_K) &= \mu\bar{c} \\ \implies \frac{1}{\mu} &= \frac{\bar{c}}{\lambda(1-P_K)} \\ \implies T_s^{\bar{o}pt} &= \frac{c\bar{c}}{c\lambda(1-P_K)} \\ \implies T_s^{\bar{o}pt} &= \frac{cU}{\lambda(1-P_K)} \end{aligned} \tag{6.3}$$

Equation 6.3 gives a simplified expression for the computation of the optimal service time in the cloudlet under MobiCoRE, given K is sufficiently large. When the condition

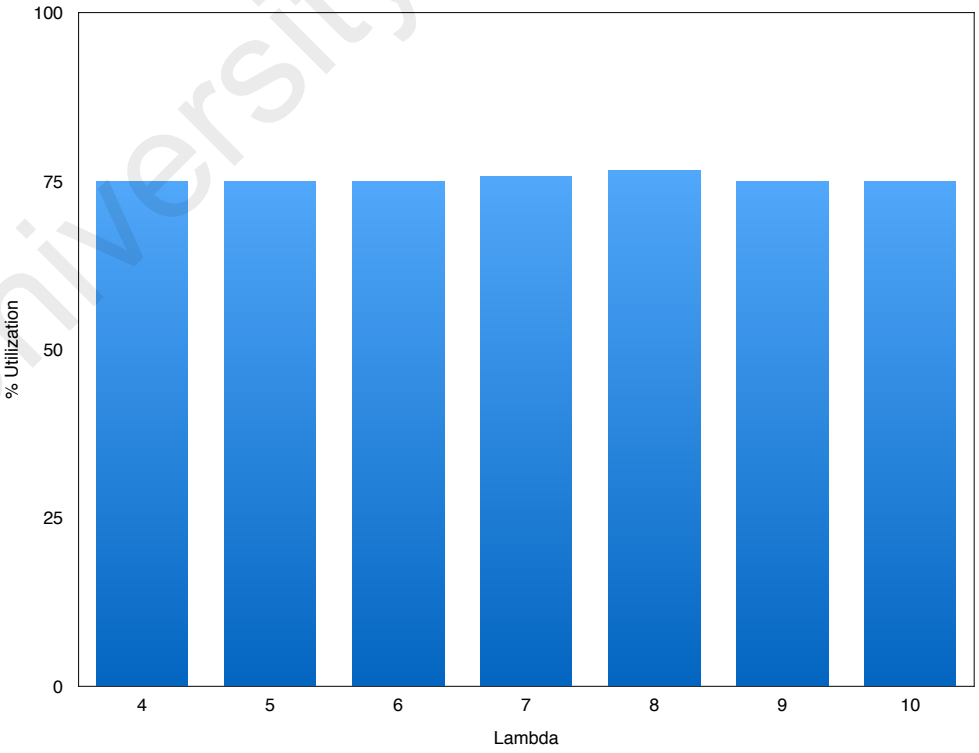

Figure 6.15: Average Cloudlet Utilization at Max. Adv.

on K is satisfied, we have $U = 0.75$ and $1 - P_K = 0.999$, as shown in Figure 6.16 and Figure 6.12, respectively. The value of K is considered to be sufficiently large as long as the value of P_K is less than 0.001. This effectively means that for given parameters of the CPU resources in the cloudlet and the user arrival rate, K should be set such that the users are not dropped by admission control. In other words, an optimal value is achieved only when the admitted user rate is adjusted according to the resources by blocking the users outside the system to achieve the desired average number of users in the system.

6.2.4 Discussion

In this section, we analysed the behavior of MobiCoRE-based cloudlets. We found that given the arrival rate and cloudlet resources, there is a service time value such that operating the cloudlet at this value results in equal time spent on the applications by the cloudlet and by the mobile device. However, this time is dependent on the service requirement of the applications, which is not known a priori. Further analysis showed that there is an optimal cloudlet service time value corresponding to a given arrival rate and cloudlet resources that results in the minimum average task completion time for the applications. Furthermore, this time is independent of the service requirements of the mobile applications. We derived a generalized expression for the optimal service time. We simplified this expression given the assumption that K is sufficiently large.

In the remainder of the chapter, we analyse the performance of MobiCoRE-based cloudlets using optimal service time values for different arrival rates and cloudlet resources.

6.3 MobiCoRE Performance

The performance of MobiCoRE can be measured using three metrics. First, MobiCoRE must ensure that any user that is admitted for service must obtain a benefit in terms of execution time compared to local task execution as well as cloudlet-based execution

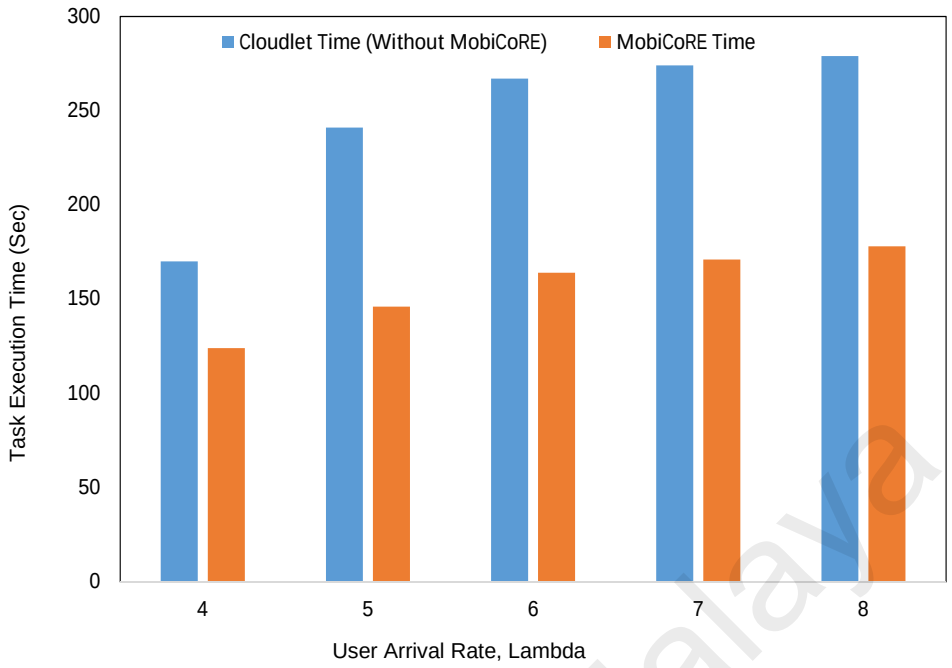

Figure 6.16: Execution Time Adv.

without MobiCoRE. Second, the number of computations performed on the mobile device should be a small portion of the total task of that mobile device. Finally, the cloudlet should be able to serve more users under MobiCoRE for the same level of resources. We show the effectiveness of MobiCoRE using the above metrics as well as through cloudlet utilization.

6.3.1 Task Completion Time Advantage

MobiCoRE uses mobile resources to enhance the performance of the cloudlet. Because of the use of mobile device resources, mobile devices vacate the cloudlet early, resulting in fewer devices in the system, which in turn reduces the waiting time in the cloudlet. Therefore, MobiCoRE should reduce the task completion time for mobile devices. Figure 6.16 shows the average advantage in terms of task completion time for mobile devices when the task is executed at an average service time of $\bar{T_{opt}}$ under MobiCoRE compared to the time when MobiCoRE is not used. It can be observed that for all

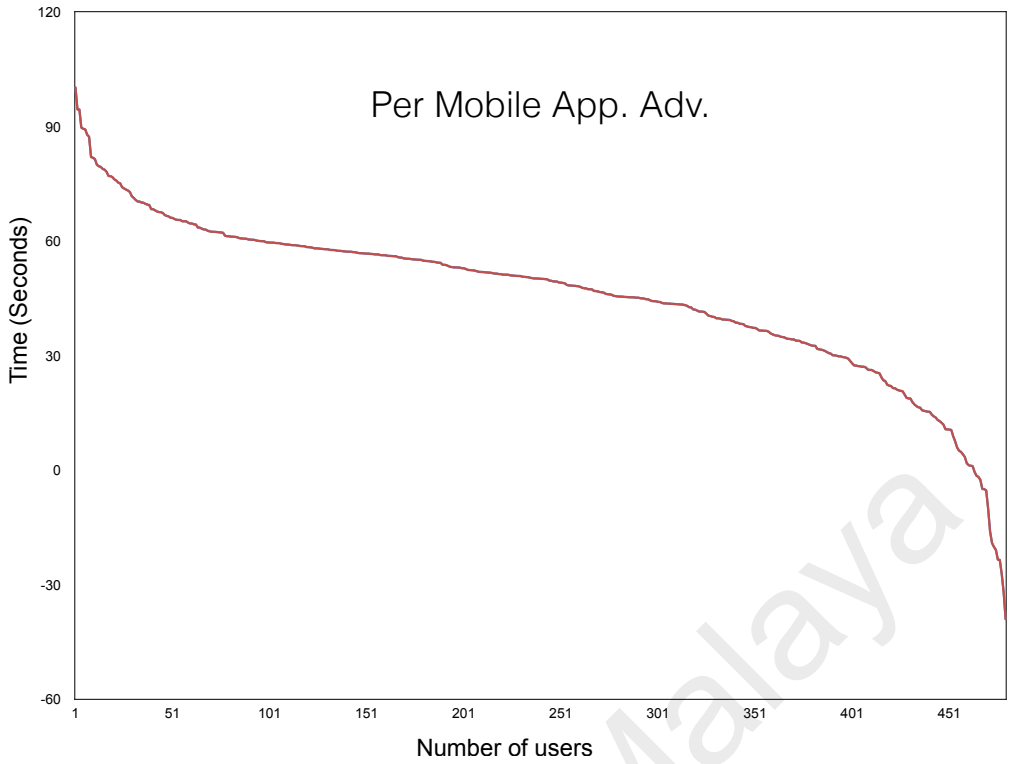

Figure 6.17: Per Device Completion Time Advantage

values of λ, MobiCoRE is able to significantly reduce the task completion time. It can also be observed that the advantage becomes constant after a certain value of λ. This is because the cloudlet operates at maximum utilization beyond this point, and increasing λ linearly increases the cloudlet waiting time.

Figure 6.17 shows the task completion advantage values for individual mobile applications for system parameters of $c = 4, K = 20, \lambda = 4$, and $\bar{T_{req}} = 60$. The actual service requirement and cloudlet-to-mobile-device speed-up ratios were generated using a Poisson distribution. It can be observed that $\approx 97\%$ of applications were able to complete their tasks faster under MobiCoRE compared to cloudlet-only execution at the optimal point, execution only by the mobile device, and execution by the full cloudlet.

6.3.2 Number of Applications Served

The primary advantage of enhanced cloudlet resources under MobiCoRE should be in the form of extra users served by the cloudlet. Figure 6.18 shows the percentage of

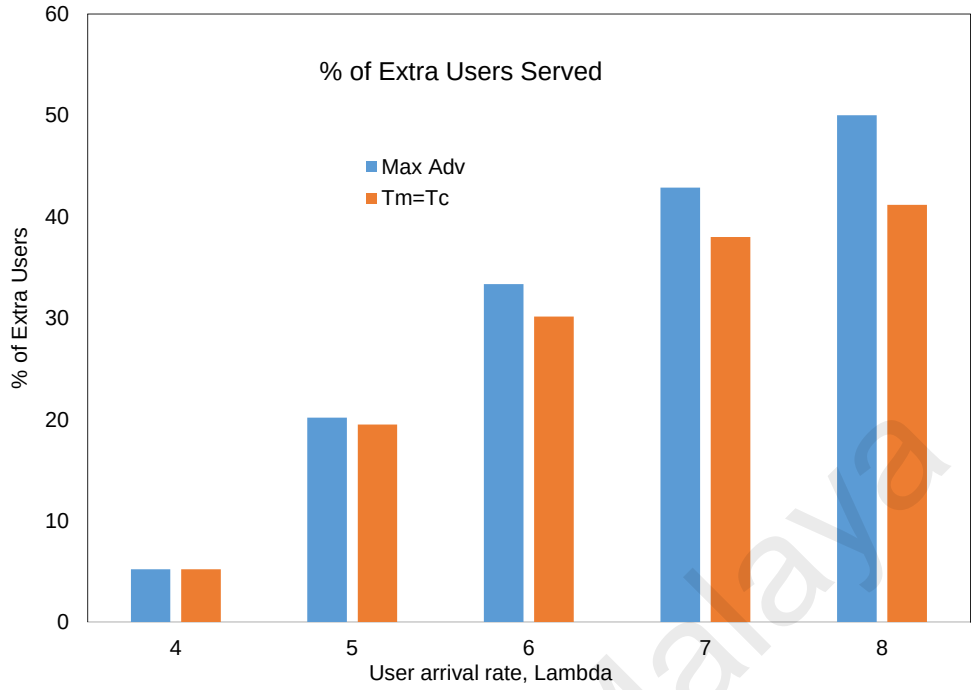

Figure 6.18: Percentage of Extra App. Served

extra users served when MobiCoRE was operated at the optimal service time as well as at the service time where $\bar{T}_m = \bar{T}_c$. It can be observed that MobiCoRE was able to accommodate up to 50% more users at higher arrival rates when operating at the optimal service time. On the other hand, the number of additional users served by the cloudlet was slightly reduced in the case where $\bar{T}_m = \bar{T}_c$. This is because fewer mobile device resources were used to ensure minimal overhead on the mobile device.

6.3.3 Cloudlet Response Time and Utilization

The resource sharing of the mobile devices with the cloudlet should result in an improved response time from the cloudlet. This can be observed in Figure 6.19, where the cloudlet response time-to-mobile device load ratio is plotted against the service times for varying λ. Higher values indicate better advantages of using the mobile device resources. It can be observed that there is a point of maximum benefit ratio for all values of λ. We observe that the service time for which the maximum response time improvement-

to-mobile resources used ratio is achieved is given by the expression $\frac{1}{P_0} = \frac{(\lambda-1)^K}{c^{K-c}c!\mu^K}1000$. We further observe that a load of 16% on the mobile device results in an up to 47% improvement in the response time of the cloudlet, which is a significant improvement in cloudlet performance.

Figure 6.20 shows the reduction in cloudlet resource utilization for varying service times under MobiCoRE compared to complete task execution by the cloudlet. We observe that for a majority of the service time values, the cloudlet utilization is reduced, which is intuitive. We also observe that the utilization is always equal to 75% for the optimal operation point, which is an improvement of 25% for most the cases where the cloudlet is subject to heavy loads. A reduction in cloudlet utilization below 80% to a more stable point results in an improved response time, which in turn results in a better performance of the cloudlet under MobiCoRE in serving mobile users.

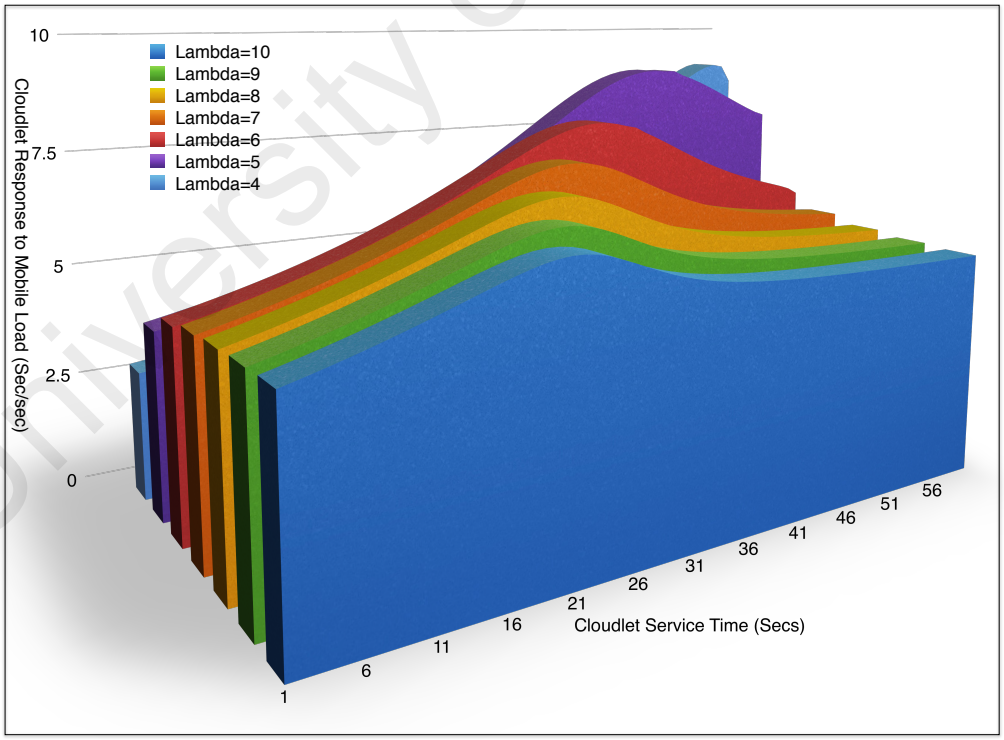

Figure 6.19: Cloudlet Response Time vs Mobile Load

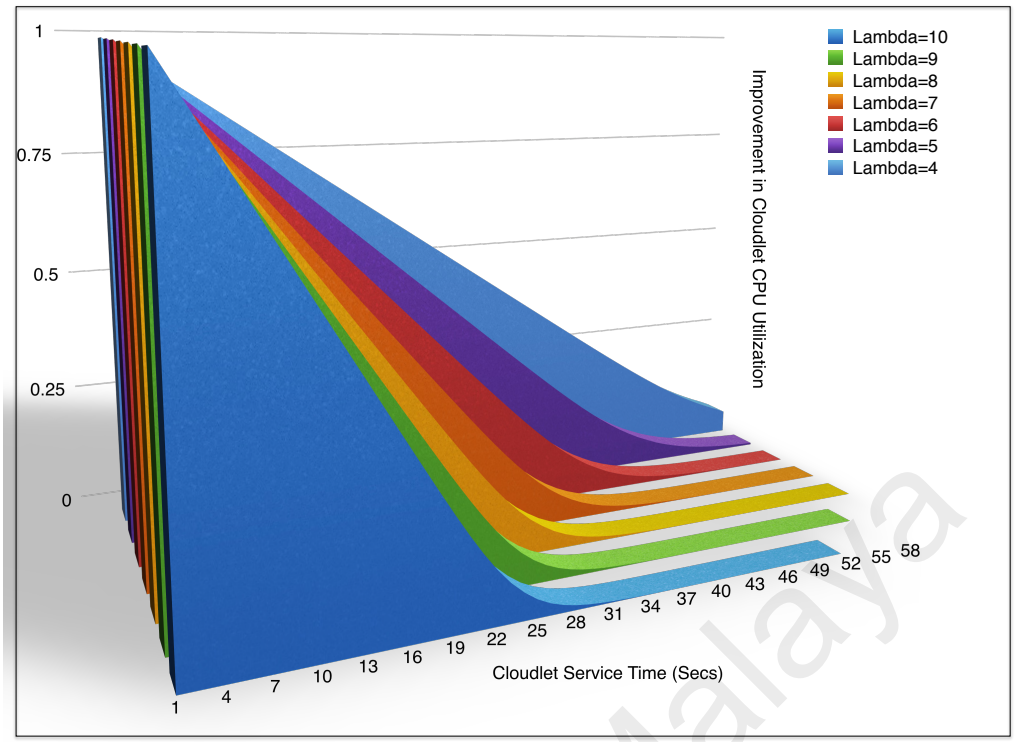

Figure 6.20: Improved Utilization vs Average Service Time

6.3.4 MobiCoRE Overhead on Mobile Device

Figure 6.21 shows the overhead of MobiCoRE on mobile devices when the maximum advantage is obtained as well as when equal time between the cloudlet and mobile device is obtained. It can be observed that equal time sharing between the cloudlet and mobile device results in a reduced overhead on the mobile device compared to the maximum advantage, especially for higher values of λ. Effectively, the cloudlet can choose between the two modes, depending on the application arrival rate. We note that in the case of equal time sharing, even at very high arrival rates, the mobile device must perform less than 50% of its own computations. This ensures that mobile device resources are not used beyond a certain threshold while ensuring an improved response time for the mobile device as well as enhanced cloudlet performance when 50% more users are served.

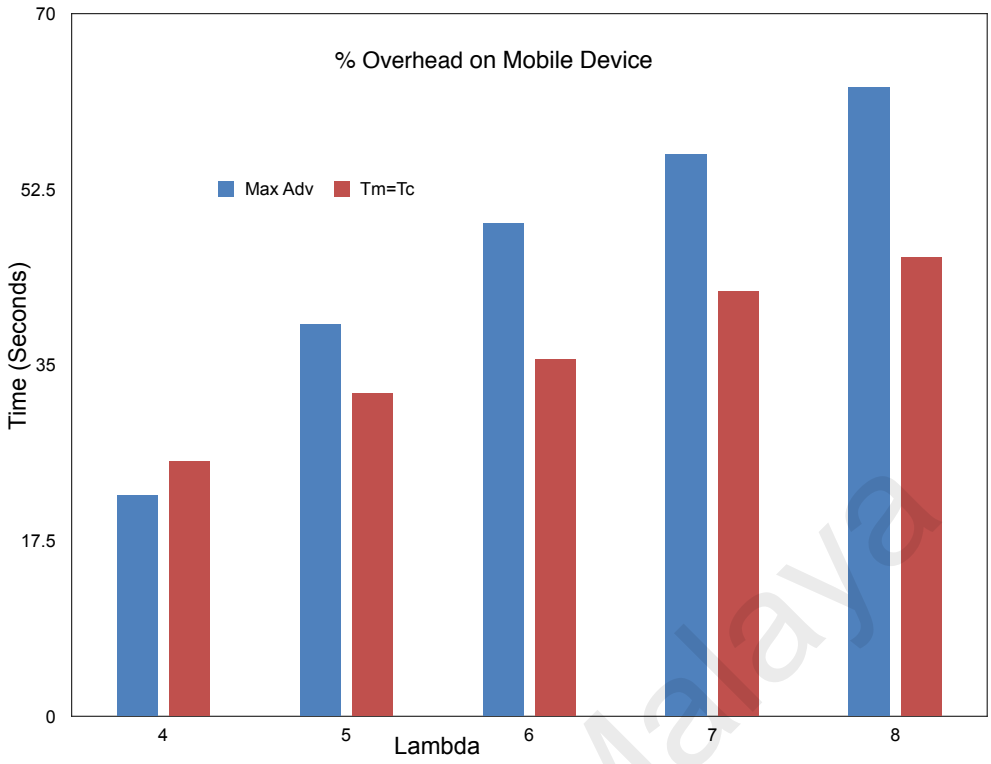

Figure 6.21: Execution Time Adv.

6.4 Conclusion

In this chapter, we validate the mathematical model of MobiCoRE framework. The data generated using the model were subsequently used to identify the optimal operational point for MobiCoRE-based cloudlets. The analysis also revealed that at the optimal operational point whereby the minimum average response time is achieved for the mobile applications, the value of the variables P_K and $\bar{c}$ are constant. These findings lead us to compute the cloudlet service time value for the optimal operational point for given values of cloudlet resources and user arrival rates. We derived an expression for such an operational point and found the the expression is generic and independent of the mobile device requirements. This expression can be used to tune any cloudlet to achieve optimal operation.Under the assumption of sufficiently large values of the allowable number of users in the system, the derived expression was greatly simplified to $\frac{\bar{c}}{\lambda}$, where $\bar{c}$ is the average number of CPUs used and λ is the application arrival rate. This simpler expression can be used to quickly tune the cloudlets when detailed parameters are not available. These

expressions are of great importance, and to the best of our knowledge, such results have yet to be derived for cloudlet operation.

Finally, we evaluated the performance of MobiCoRE. We considered the performance enhancement of the cloudlet in terms of the average response time and the number of users served. We also considered the impact of the improved performance of the cloudlet on the service received by the mobile devices. The results showed that MobiCoRE can accommodate up to 50% more users when operating at optimal service times and provide 50% time benefits to mobile users. At the optimal point, only 16% of the resources of the mobile devices are used to achieve the above-mentioned increase in the number of users served and the service time. The empirical analysis and statistical validation demonstrate that our proposed framework, MobiCoRE, positively and significantly impacts cloudlet performance by exploiting and orchestrating nearby mobile device resources without adding significant additional loads on the mobile devices.

CHAPTER 7 : CONCLUSION

This chapter presents the overall conclusion of this thesis and highlight the future research directions. We re-examine and highlight the research aim and objectives by revisiting again to ensure that we achieve through the work reported in this thesis. We also present the significance and important of this study and highlight the contributions in this thesis. Moreover, we list the scholarly peer reviewed ISI journal articles, conference and proceedings papers as an outcome of carried out research work throughout the research period. Finally, this chapter ends by discussion on research limitations and possible future works.

In Section 7.1, we describe how aim and objectives of this study are accomplished. Section 7.2 presents contributions of this research. Significance of this work is presented in Section 7.3. The scholarly ISI articles and conferences that are produced by this research are listed in Section 7.4 and future works are highlighted in Section 7.5.

7.1 Research Objectives Revisited

In this research, we aimed to achieve the performance enhancement of resource constrained cloudlet and at the same time the optimal user response time in cloudlet based mobile application augmentation in mobile cloud computing. In following, we re-examine the research aim and objectives whether or not this thesis achieve and fulfil the expected requirements.

Objective #1: Study and identify the recent cloudlet based mobile application augmentation approaches

The first objective was to study and identify recently developed cloudlet-based mobile application augmentation approaches and gain an insight into offloading in MCC to help us identify current problems in computation-intensive mobile application aug-

mentation. This research was conducted through an extensive and rigorous literature review by considering major research from well-established scholarly published databases worldwide. We used Web of Science, IEEE Explorer, Scopus ScienceDirect by Elsevier, Springer publications, Taylor and Francis and Inderscience and downloaded the most up-to-date research works and findings. We organised the literature and developed taxonomies for cloud-/cloudlet-based mobile application execution and cloudlet resource augmentation.

A number of research issues were identified in the literature review. The highlighted research issues demonstrated that a critical issue hindering the wide-spread deployment of cloudlets is ensuring that the cloudlets remain self-managed and easy to use and deploying computational entities while ensuring adequate user performance. The restrictions of the self-management and self-deployment of free resources resulted in limited resources in the cloudlets, with no possibility of enhancing the resources as required. This problem was subsequently addressed by the proposed research.

Objective #2: Investigate and analyse the limited and finite resources impact on cloudlet performance

The second objective was to investigate and analyse the impact of limited and finite resources on cloudlet performance as well as on the user service experience in cloudlet-based mobile application augmentation. We performed an extensive empirical study and developed a basic model for the cloudlets. The empirical analysis was used to investigate the service time, cloudlet response, resource utilisation and overall mobile service time to demonstrate that the problem of resource scarcity indeed exists and affects the user performance. The experiments helped us better understand the problem. Subsequently, a preliminary mathematical model was developed to better understand the extent of the problem. Using the model, we derived a relationship between the number of user applications in the cloudlet, the cloudlet resources and the expected response time from the

cloudlet. We were able to establish the number of user applications beyond which the cloudlet is unable to offer any advantage to the mobile devices.

Objective #3: Propose a performance enhancement framework for cloudlet in MCC for cloudlet based compute-intensive mobile application augmentation

The third objective was to propose a performance enhancement solution for cloudlets in MCC for cloudlet-based computation-intensive mobile application augmentation to achieve efficient computation by exploiting mobile device resources. As a solution to the identified problem, we proposed the MobiCoRE framework for cloudlets, which enabled the performance enhancement of cloudlets while simultaneously optimising the mobile user's overall response time. We described several building blocks and architectural components of the framework. In addition, we described the work flow in sections.

The proposed framework tunes the cloudlet service time to control the number of users in the system, which in turn keeps the cloudlet response time low, resulting in improved service time for the mobile users. This is achieved using the limited resources of mobile devices for completion of their own tasks. We modelled the proposed system using queuing theory and derived meaningful conclusions from the proposed model.

Objective #4: Evaluate the performance of the proposed framework

The final objective was to evaluate the performance of the proposed performance enhancement solution by considering several performance metrics such as efficiency, user service time and resource utilisation. To this end, we implemented the proposed framework and model in Matlab. The data were collected using the model and the implementation. The results were first used to validate the model, which showed a near perfect match with the results from the experiments. The model was used to derive the performance parameters for the proposed framework to obtain optimal operation. Specifically, we derived generic expressions, thereby linking the cloudlet optimal service time with the cloudlet resources and the user application arrival rate. The performance of the cloudlet

was evaluated, and we found that as many as 50% more users can be served with an up to 50% improved total response time if 16% of the resources of the mobile devices are shared with the cloudlet to augment the resources of the cloudlet.

7.2 Contributions

We now summarize the contributions of this research. First, we present the contribution of this thesis to the body of knowledge. Then, we present a list of scholarly articles that were published during the course of this research.

7.2.1 Taxonomy of Cloudlet based Resource Augmentation

We developed the taxonomy for the existing literature in the domain of cloudlet resource augmentation. We also produced and devised the taxonomy of cloudlet based resources in MCC by critically reviewing the state-of-the-art researches extracted from more than 100 scholarly articles. This classification and categorization is useful and significant contribution of this thesis for future researchers.

7.2.2 Outlined Open Research Issues and Challenges

In this thesis, we outlined and presented the research challenges and open issues. We analyzed and synthesized the present state-of-the-art literature of the cloudlet based resource augmentation for future research directions and also included the current limitations for cloudlet based mobile application augmentation research. This provides important research guidelines for young researchers in this domain.

7.2.3 Finite Resource Impacts Investigation on Cloudlet based Mobile Application Augmentation

We contributed to the body of knowledge by identifying the impacts of finite resource cloudlet based mobile application in MCC. We performed empirical investigation and comprehensive cloudlet behavior analysis of cloudlet performance. We investigated and

demonstrated significant cloudlet resource limitation effects by considering efficiency, throughput, and cloudlet response time. The empirical investigation and analytical study established that the limited resource of cloudlet negatively impact the cloudlet performance, which we identified as resource scarcity problem in cloudlet based mobile application augmentation in MCC.

7.2.4 Performance Enhancing MobiCoRE Framework

We proposed MobiCoRE, which is a performance enhancement framework for cloudlets that uses mobile devices for resource augmentation. The framework addresses the shortcomings of current solutions. Specifically, the framework ensures that the loads on the mobile devices are always a fraction of their own requirements from the cloudlet and that time benefits are always obtained for the mobile devices. The framework does not lead to issues concerning privacy and confidentiality because mobile application code and data are not shared with other mobile user devices in the network. No installation is required on the mobile devices. The performance analysis shows that MobiCoRE can significantly improve the performance of cloudlets as well as the service response time of mobile device applications.

7.2.5 MobiCoRE Framework Evaluation and Validation

We contributed to the knowledge by implementing, evaluating, and validating the performance of MobiCoRE framework . We also demonstrated its reliability, validity and significance. The results also verify that utilizing our proposed MobiCoRE framework provide significant performance enhancement of cloudlet as up to 50% , and served 50% extra user in the cloudlet system. The results ensure the feasibility of the framework and advocate the realization of objectives for this research.

7.2.6 Generic Expression Development

The most valuable contribution of this thesis is the development of generic expressions that link the optimal cloudlet service time with the cloudlet resources and the user arrival rate. The expressions are independent of the user service requirement, which is not known in advance. The expressions can be used to dynamically tune the cloudlet to obtain optimal performance under variable application arrival rates. To the best of our knowledge, our work is the first work to develop such expressions for cloudlets.

7.2.7 International Scholarly Publications:

The following list includes international research publications in high-quality publication produced by us during this research tenure.

Publication as only First Authors:

1. **Md Whaiduzzaman**, Mehedi Shookhak , Abdullah Gani, Rajkumar Buyya, "A Survey on Vehicular Cloud Computing", Elsevier Journal of Computer Networks and Applications, ISI (Tier1/Q1)and SCOPUS indexed publication, Impact Factor 2.23, Publised April, 2014

2. **Md Whaiduzzaman**, Abdullah Gani, Mohammad Nazmul Haque, Md Rejaul Karim Chowdhury, "A Study on Strategic Provisioning of Cloud Computing Services", Published Scientific World Journal , ISI indexed Tier1/Q1, Impact Factor 1.73, June 2014

3. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed , Rajkumar Buyya, "Cloudlet based Mobile Application Augmentation in Mobile Cloud Computing: Motivation, Taxonomy and Open Issues" , Elsevier Journal of Computer Networks and Applications,Under Review, ISI Tier1/Q1 and SCOPUS indexed publication,Impact Factor 2.23, August, 2015

4. **Md Whaiduzzaman**, Abdullah Gani, Nor Badrul Anuar, Muhammad Shiraz, Mohammad Nazmul Haque and Israat Tanzeena Haque, " Cloud Service Selection using Multi-Criteria Decision Analysis", The Scientific World Journal, Published, March 2014, ISI Indexed Q1/Tier1, Impact Factor 1.73

5. **Md Whaiduzzaman**, Anjum Naveed, Abdullah Gani " MobiCoRE: Mobile based Cloudlet Resource Enhancement for Optimal Task Response", IEEE Transactions on Service Computing,ISI and SCOPUS indexed publication, Revised, Tier1/Q1, Impact Factor 3.049, June, 2015

6. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed, "TOWARDS ENHANCING RESOURCE SCARCE CLOUDLET PERFORMANCE IN MOBILE CLOUD COMPUTING", Fifth International Conference on Computer Science and Information Technology, CCSIT-2015, Sydney, Australia, February 2015

7. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed, " An Empirical Analysis of Finite Resource Impact on Cloudlet Performance in Mobile Cloud Computing", Conference, CEET-2014, Kuala Lumpur, Malaysia, April 2014

8. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed, " PEFC: Performance Enhancement Framework for Cloudlet in Mobile Cloud Computing", IEEE & Conference, ROMA-2014,Kuala Lumpur, Malaysia, June 2014.

9. **Md Whaiduzzaman**, Abdullah Gani, " Measuring Security for Cloud Service Provider : A Third Party Approach", IEEE & ISI indexed International conference on Electrical Information & Communication Technology (ECIT), Khulna, December 2013

7.3 Significance of the Work

The research work presented in this thesis can be considered significant because of several key features briefly discussed as follows.

First, cloudlets represent the present and future of mobile cloud computing. Addressing the issue of resource scarcity for cloudlets can facilitate the wide-spread deployment of cloudlets. The proposed research has addressed the problem of resource scarcity in cloudlets while addressing known research issues concerning the problems and shortcomings of current solutions. Therefore, the thesis can be considered as a valuable research work.

Second, our presented framework, MobiCoRE address several shortcomings of exiting literature. For example, the framework does not lead to the issues of privacy and confidentiality because mobile application code and data is not shared with other mobile user devices in the network. Mobile device does not required extra software installation to use the cloudlet. The performance analysis shows that MobiCoRE can significantly improve the performance of cloudlet as well as service response time of the mobile device applications.

Third, and more important aspect is deriving and presenting generic expressions linking performance parameters of the cloudlet. Unlike prior research where empirical verifications demonstrated that the proposed solutions work under limited sets of scenarios, the proposed research derives generic mathematical expressions. These expressions can be used to dynamically tune cloudlets to achieve optimal operation points. The generic

nature of these expressions makes them applicable to any cloudlet or to similar systems, irrespective of the operating conditions. Therefore, the contribution of this research can be considered as a significant contribution to the body of knowledge in the domain of cloudlet resource augmentation using mobile devices.

7.4 Future Research Work

In this section, we enlist a limited set of possibilities of future work, that arise from this thesis.

1. The proposed research does not rely on application partitioning because a generic solution for dynamic partitioning is not available. However, if the functionally independent partitions of an application can be created, then the mobile device can work on one of the partitions in parallel with the cloudlet computation. In this way, by the time the cloudlet completes its part of the computation, the mobile device can most likely finish the remaining computation, thereby reducing the total response time by almost half. Exploring such a possibility remains a future research subject to the availability of dynamic partitioning algorithms.

2. We expect that the cloudlet will have the same platform as the mobile device (hardware and operating system) available as a virtual machine. This facilitates the transfer of states from the cloudlet to the mobile device. However, if such a setup is not available, this heterogeneous platform does not allow for state migration. The heterogeneous migration of tasks and integration into the proposed framework remains as future work.

3. A detailed admission control and service broker can significantly improve the admission of user applications into the mobile device. The integration of such a mechanism can improve the smoothness of the number of users and their service time in the cloudlet.

REFERENCES

Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., & Buyya, R. (2014). Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials*, *16*(1), 337–368.

Abolfazli, S., Sanaei, Z., Alizadeh, M., Gani, A., & Xia, F. (2014, February). An experimental analysis on cloud-based mobile augmentation in mobile cloud computing. *IEEE Transactiojn on Consumer Electronics*, *60*(1). doi: 10.1109/TCE.2014.6780937

Abolfazli, S., Sanaei, Z., Gani, A., Xia, F., & Yang, L. T. (2014). Rich mobile applications: Genesis, taxonomy, and open issues. *Journal of Network and Computer Applications*, *40*, 345 - 362. doi: http://dx.doi.org/10.1016/j.jnca.2013.09.009

Achanta, V. S., Sureshbabu, N. T., Thomas, V., Sahitya, M. L., & Rao, S. (2012). Cloudlet-based multi-lingual dictionaries. In *Third international conference on services in emerging markets (icsem)* (pp. 30–36).

Ahmed, E., Akhunzada, A., Whaiduzzaman, M., Gani, A., Ab Hamid, S. H., & Buyya, R. (2015). Network-centric performance analysis of runtime application migration in mobile cloud computing. *Simulation Modelling Practice and Theory*, *50*, 42–56.

Ahmed, E., Gani, A., Khan, M. K., Buyya, R., & Khan, S. U. (2015). Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications*, *52*, 154 - 172. doi: http://dx.doi.org/10.1016/j.jnca.2015.03.001

Ahmed, E., Gani, A., Sookhak, M., Ab Hamid, S. H., & Xia, F. (2015). Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. *Journal of Network and Computer Applications*, *52*, 52–68.

Albanesius, C. (2011, February). *smartphone shipments surpass pc shipments for first time. whats next?* http://www.pcmag.com/article2/0,2817,2379665,00.asp.

Ali, A. M. M., Ahmad, N. M., & Amin, A. H. M. (2014). Cloudlet-based cyber foraging framework for distributed video surveillance provisioning. In *Proc. of fourth world congress on information and communication technologies (wict)* (pp. 199–204).

Apple, O. (2015, Accessed on 7th July). *Apple notes.* Retrieved from `https://www.apple.com/itunes/charts/free-apps/`

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... Zaharia, M. (2010, April). A view of cloud computing. *Commun. ACM*, *53*(4), 50–58.

Bahtovski, A., & Gusev, M. (2014a). Cloudlet challenges. *Procedia Engineering*, *69*, 704–711.

Bahtovski, A., & Gusev, M. (2014b, May). Multilingual cloudlet-based dictionary. In *37th international convention on information and communication technology, electronics and microelectronics (mipro)* (p. 380-385).

Bayat, N., & Lutfiyya, H. (2014). Mc-skynet: Mobile-cloud dynamic partitioning for mobile cloud applications. In *Computers and communication (iscc), 2014 ieee symposium on* (pp. 1–7).

Bohez, S., Turck, J. D., Verbelen, T., Simoens, P., & Dhoedt, B. (2013). Mobile, collaborative augmented reality using cloudlets. In *Proceedings of the 2013 international conference on mobile wireless middleware, operating systems, and applications* (pp. 10–19).

Bohez, S., Verbelen, T., Simoens, P., & Dhoedt, B. (2014, April). Allocation algorithms for autonomous management of collaborative cloudlets. In *Ieee 2nd international conference on mobile cloud computing, services, and engineering (mobilecloud)* (p. 1-9).

Bohez, S., Verbelen, T., Simoens, P., & Dhoedt, B. (2015). Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets. *Simulation Modelling Practice and Theory*, *50*, 109–129.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging {IT} platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599 - 616.

Cai, W., Leung, V. C., & Hu, L. (2014). A cloudlet-assisted multiplayer cloud gaming system. *Mobile Networks and Applications*, *19*(2), 144–152.

Ceselli, A., Premoli, M., & Secci, S. (2015). Cloudlet network design optimization. In *Proc. of ifip networking.*

Chang, Y.-S., Fan, C.-T., Lo, W.-T., Hung, W.-C., & Yuan, S.-M. (2015). Mobile cloud-based depression diagnosis using an ontology and a bayesian network. *Future Generation Computer Systems*, *43-44*(0), 87 - 98.

Chen, S., Wang, Y., & Pedram, M. (2013). A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud. In *Globecom* (p. 2885-2890).

Chi, F., Wang, X., Cai, W., & Leung, V. (2014). Ad hoc cloudlet based cooperative cloud gaming. In *Cloud computing technology and science (cloudcom), 2014 ieee 6th international conference on* (pp. 190–197).

Christensen, J. H. (2009). Using restful web-services and cloud computing to create next generation mobile applications. In *Proceedings of the 24th acm sigplan conference companion on object oriented programming systems languages and applications* (pp. 627–634). New York, NY, USA: ACM.

Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on computer systems* (pp. 301–314).

Clinch, S., Harkes, J., Friday, A., Davies, N., & Satyanarayanan, M. (2012, March). How close is close enough? understanding the role of cloudlets in supporting display appropriation by mobile users. In *Ieee international conference on pervasive computing and communications (percom)* (p. 122-127).

Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on mobile systems, applications, and services* (pp. 49–62).

Duro, F. R., Blas, J. G., Higuero, D., Perez, O., & Carretero, J. (2015). Cosmic: A hierarchical cloudlet-based storage architecture for mobile clouds. *Simulation Modelling Practice and Theory*, *50*, 3–19.

El-Derini, M., Aly, H., El-Barbary, A.-H., & El-Sayed, L. (2014, April). Droidcloudlet: Towards cloudlet-based computing using mobile devices. In *5th international conference on information and communication systems (icics)* (p. 1-6).

Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems*, *29*(1), 84–106.

Fesehaye, D., Gao, Y., Nahrstedt, K., & Wang, G. (2012, Sept). Impact of cloudlets on interactive mobile cloud applications. In *Ieee 16th international enterprise distributed object computing conference (edoc)* (p. 123-132).

Flores, H., Srirama, S. N., & Paniagua, C. (2012). Towards mobile cloud applications. *International Journal of Pervasive Computing and Communications*, *8*(4), 344-367.

Gartner. (2013, Nov). *Gartner says smartphone sales accounted for 55 percent of overall mobile phone sales in third quarter of 2013.* Retrieved from `http://www.gartner.com/newsroom/id/2623415`

Ha, K., Pillai, P., Richter, W., Abe, Y., & Satyanarayanan, M. (2013). Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th annual international conference on mobile systems, applications, and services* (pp. 153–166).

Harney, E., Goasguen, S., Martin, J., Murphy, M., & Westall, M. (2007). The efficacy of live virtual machine migrations over the internet. In *Proceedings of the 2nd inter-*

national workshop on virtualization technology in distributed computing (vtdc'07), reno, nevada, usa (pp. 1–7). ACM.

Hoang, D. T., Niyato, D., & Le, L. B. (2014). Simulation-based optimization for admission control of mobile cloudlets. In *Communications (icc), 2014 ieee international conference on* (pp. 3764–3769).

Hoang, D. T., Niyato, D., & Wang, P. (2012, April). Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In *Ieee wireless communications and networking conference (wcnc)* (p. 3145-3149).

Huerta-Canepa, G., & Lee, D. (2010). A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st acm workshop on mobile cloud computing & services: Social networks and beyond.* ACM.

Jararweh, Y., Tawalbeh, L., Ababneh, F., & Dosari, F. (2013, Dec). Resource efficient mobile computing using cloudlet infrastructure. In *Ieee ninth international conference on mobile ad-hoc and sensor networks (msn)* (p. 373-377).

Jin, A., Song, W., Wang, P., Niyato, D., & Ju, P. (2015). Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. *Services Computing, IEEE Transactions on*, *PP*(99), 1-1.

Jindal, M., & Dave, M. (2014, May). Data security protocol for cloudlet based architecture. In *Recent advances and innovations in engineering (icraie), 2014* (p. 1-5).

Kara, N., Soualhia, M., Belqasmi, F., Azar, C., & Glitho, R. (2014). Genetic-based algorithms for resource management in virtualized ivr applications. *Journal of Cloud Computing*, *3*(1), 1–18.

Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2012). Cuckoo: a computation offloading framework for smartphones. In *Mobile computing, applications, and services* (pp. 59–79). Springer.

Khan, K. A., Wang, Q., Grecos, C., Luo, C., & Wang, X. (2013). Meshcloud: Integrated cloudlet and wireless mesh network for real-time applications. In *Electronics, circuits, and systems (icecs), 2013 ieee 20th international conference on* (pp. 317–320).

Kommineni, S., De, A., Alladi, S., & Chilukuri, S. (2014). The cloudlet with a silver lining. In *Sixth international conference on communication systems and networks (comsnets).*

Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proceedings ieee infocom* (pp. 945–953).

Kristensen, M. D. (2010). Scavenger: Transparent development of efficient cyber foraging applications. In *Ieee international conference on pervasive computing and communications (percom)* (pp. 217–226).

Kumar, K., & Lu, Y.-H. (2010). Cloud computing for mobile users: Can offloading computation save energy? *Computer*, *43*(4), 51-56.

Li, Y., & Wang, W. (2013, Dec). The unheralded power of cloudlet computing in the vicinity of mobile devices. In *Ieee global communications conference (globecom)* (p. 4994-4999).

Li, Y., & Wang, W. (2014, April). Can mobile cloudlets support mobile applications? In *Infocom, 2014 proceedings ieee* (p. 1060-1068).

Litzkow, M., Tannenbaum, T., Basney, J., & Livny, M. (1997). *Checkpoint and migration of unix processes in the condor distributed processing system.* Computer Sciences Department, University of Wisconsin.

Liu, J., Ahmed, E., Shiraz, M., Gani, A., Buyya, R., & Qureshi, A. (2015). Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, *48*, 99 - 117. doi: http://dx.doi.org/10.1016/j.jnca.2014.09.009

Lumia. (2015, June). *Microsoft nokia lumia.* Retrieved from `http://www.microsoft.com/en-in/mobile/apps/lumia/`

Magurawalage, C. M. S., Yang, K., Hu, L., & Zhang, J. (2014). Energy-efficient and network-aware offloading algorithm for mobile cloud computing. *Computer Networks*, *74*, 22–33.

March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., & Lee, B. S. (2011). μcloud: towards a new paradigm of rich mobile applications. *Procedia Computer Science*, *5*, 618–624.

Miettinen, A. P., & Nurminen, J. K. (2010). Energy efficiency of mobile clients in cloud computing. In *Proceedings of the 2nd usenix conference on hot topics in cloud computing* (pp. 4–4). Berkeley, CA, USA: USENIX Association.

Modares, H., Moravejosharieh, A., Lloret, J., & Salleh, R. (2014). A survey on proxy mobile ipv6 handover. *IEEE Systems Journal*, 1-10.

Munteanu, V. I., Şandru, C., & Petcu, D. (2014). Multi-cloud resource management: cloud service interfacing. *Journal of Cloud Computing*, *3*(1), 1–23.

Nkosi, M., & Mekuria, F. (2010, Nov). Cloud computing for enhanced mobile health applications. In *Ieee second international conference on cloud computing technology and science (cloudcom)* (p. 629-633).

Park, S., Parwani, A., Satyanarayanan, M., & Pantanowitz, L. (2012). Handheld computing in pathology. *Journal of Pathology Informatics*, *3*(1), 15.

Pirozmand, P., Wu, G., Jedari, B., & Xia, F. (2014). Human mobility in opportunistic networks: Characteristics, models and prediction methods. *Journal of Network and Computer Applications*, *42*, 45–58.

Qi, H., Shiraz, M., Gani, A., Whaiduzzaman, M., & Khan, S. (2014). Sierpinski triangle based data center architecture in cloud computing. *The Journal of Supercomputing*, *69*(2), 887–907.

Qing, W., Zheng, H., Ming, W., & Haifeng, L. (2013, June). Cactse: Cloudlet aided cooperative terminals service environment for mobile proximity content delivery. *Communications, China*, *10*(6), 47-59.

Quwaider, M., & Jararweh, Y. (2013). Cloudlet-based for big data collection in body area networks. In *8th international conference for internet technology and secured transactions (icitst)* (pp. 137–141).

Quwaider, M., & Jararweh, Y. (2015). Cloudlet-based efficient data collection in wireless body area networks. *Simulation Modelling Practice and Theory*, *50*, 57–71.

Rahimi, M. R. (2012). Exploiting an elastic 2-tiered cloud architecture for rich mobile applications. In *Ieee international symposium on a world of wireless, mobile and multimedia networks (wowmom)* (pp. 1–2).

Rahimi, M. R., Venkatasubramanian, N., Mehrotra, S., & Vasilakos, A. V. (2012). Mapcloud: Mobile applications on an elastic and scalable 2-tier cloud architecture. In *Proceedings of the 2012 ieee/acm fifth international conference on utility and cloud computing* (pp. 83–90).

Rak, M., Venticinque, S., Máhr, T., Echevarria, G., & Esnal, G. (2011). Cloud application monitoring: The mosaic approach. In *Ieee third international conference on cloud computing technology and science (cloudcom)* (pp. 758–763).

Ravi, A., & Peddoju, S. K. (2013). Energy efficient seamless service provisioning in mobile cloud computing. In *Ieee 7th international symposium onservice oriented system engineering (sose)* (pp. 463–471).

Rawadi, J., Artail, H., & Safa, H. (2014, April). Providing local cloud services to mobile devices with inter-cloudlet communication. In *17th ieee mediterranean electrotechnical conference (melecon)* (p. 134-138).

Routaib, H., Badidi, E., Elmachkour, M., Sabir, E., & Elkoutbi, M. (2014, May). Modeling and evaluating a cloudlet-based architecture for mobile cloud computing. In *9th international conference on intelligent systems: Theories and applications (sita-14)* (p. 1-7).

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009, October). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*(4), 14–23.

Satyanarayanan, M., Lewis, G., Morris, E., Simanta, S., Boleng, J., & Ha, K. (2013). The role of cloudlets in hostile environments. *Pervasive Computing, IEEE*, *12*(4), 40–49.

Shi, Y., Abhilash, S., & Hwang, K. (2015). Cloudlet mesh for securing mobile clouds from intrusions and network attacks. In *The third ieee international conference on mobile cloud computing, services, and engineering,(mobile cloud 2015).*

Shiraz, M., Ahmed, E., Gani, A., & Han, Q. (2014). Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing. *The Journal of Supercomputing*, *67*(1), 84-103. doi: 10.1007/s11227-013-0988-6

Shiraz, M., Gani, A., Khokhar, R. H., & Buyya, R. (2013). A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys & Tutorials*, *15*(3), 1294–1313.

Shiraz, M., Whaiduzzaman, M., & Gani, A. (2013). A study on anatomy of smartphone. *Computer Communication & Collaboration*, *1*, 24–31.

Silva, R., Silva, J. S., & Boavida, F. (2014). Mobility in wireless sensor networks: Survey and proposal. *Computer Communications*.

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012). Cloudvision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *Ieee symposium on computers and communications (iscc)* (pp. 000059–000066).

Soyata, T., Muraleedharan, R., Langdon, J., Funai, C., & Ames, S. (2012, May). Combat: mobile-cloud-based compute/communications infrastructure for battlefield applications. *Proc. SPIE 8403, Modeling and Simulation for Defense Systems and Applications VII*.

Treurniet, J. (2014). A taxonomy and survey of microscopic mobility models from the mobile networking domain. *ACM Computing Surveys (CSUR)*, *47*(1), 14.

Verbelen, T., Simoens, P., De Turck, F., & Dhoedt, B. (2012a). Aiolos: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, *85*(11), 2629–2639.

Verbelen, T., Simoens, P., De Turck, F., & Dhoedt, B. (2012b). Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the third acm workshop on mobile cloud computing and services* (pp. 29–36).

Verbelen, T., Simoens, P., De Turck, F., & Dhoedt, B. (2013). Adaptive application

configuration and distribution in mobile cloudlet middleware. In *Mobile wireless middleware, operating systems, and applications* (pp. 178–191). Springer.

Verbelen, T., Simoens, P., De Turck, F., & Dhoedt, B. (2014). Adaptive deployment and configuration for mobile augmented reality in the cloudlet. *Journal of Network and Computer Applications*, *41*, 206–216.

Vu, L., Nguyen, P., Nahrstedt, K., & Richerzhagen, B. (2014). Characterizing and modeling people movement from mobile phone sensing traces. *Pervasive and Mobile Computing*.

Wei, X., Fan, J., Lu, Z., & Ding, K. (2013). Application scheduling in mobile cloud computing with load balancing. *Journal of Applied Mathematics*, *2013*.

Whaiduzzaman, M., & Gani, A. (2014). Measuring security for cloud service provider: A third party approach. In *IEEE International Conference on Electrical Information and Communication Technology (EICT)* (pp. 1–6). Khulna.

Whaiduzzaman, M., Gani, A., Anuar, N. B., Shiraz, M., Haque, M. N., & Haque, I. T. (2014). Cloud service selection using multicriteria decision analysis. *The Scientific World Journal*, *2014*.

Whaiduzzaman, M., Gani, A., & Naveed, A. (2014, April). An empirical analysis of finite resource impact on cloudlet performance in mobile cloud computing. Kuala Lumpur, Malaysia: CEET-2014.

Whaiduzzaman, M., Gani, A., & Naveed, A. (2015, February). Towards enhancing resource scare cloudlet performancein mobile cloud computing. Sydney, Australia.

Whaiduzzaman, M., Haque, M. N., Rejaul Karim Chowdhury, M., & Gani, A. (2014). A study on strategic provisioning of cloud computing services. *The Scientific World Journal*, *2014*.

Whaiduzzaman, M., Sookhak, M., Gani, A., & Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, *40*, 325–344.

Wu, Y., & Ying, L. (2015). A cloudlet-based multi-lateral resource exchange framework for mobile users.

Xia, Q., Liang, W., & Xu, W. (2013, Oct). Throughput maximization for online request admissions in mobile cloudlets. In *Ieee 38th conference on local computer networks (lcn)* (p. 589-596).

Yamato, Y., Muroi, M., Tanaka, K., & Uchimura, M. (2014). Development of template management technology for easy deployment of virtual resources on openstack. *Journal of Cloud Computing*, *3*(1), 1–12.

Yang, Z., Niyato, D., & Wang, P. (2015). Offloading in mobile cloudlet systems with intermittent connectivity. *IEEE Transactions on Mobile Computing*, *PP*(99), 1-1. doi: 10.1109/TMC.2015.2405539

Zhang, X., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011). Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks and Applications*, *16*(3), 270–284.

APPENDICES-A

International Scholarly Publications: The following list includes international research publications in high-quality publication produced during this research tenure.

Journal Publication as First Authors:

1. **Md Whaiduzzaman**, Mehedi Shookhak , Abdullah Gani, Rajkumar Buyya, "A Survey on Vehicular Cloud Computing", Elsevier Journal of Computer Networks and Applications, ISI (Tier1/Q1)and SCOPUS indexed publication, Impact Factor 2.23, Publised April, 2014

2. **Md Whaiduzzaman**, Abdullah Gani, Mohammad Nazmul Haque, Md Rejaul Karim Chowdhury, "A Study on Strategic Provisioning of Cloud Computing Services", Published Scientific World Journal , ISI indexed Tier1/Q1, Impact Factor 1.73, June 2014

3. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed , Rajkumar Buyya, "Cloudlet based Mobile Application Augmentation in Mobile Cloud Computing: Motivation, Taxonomy and Open Issues" , Elsevier Journal of Computer Networks and Applications,Under Review, ISI Tier1/Q1 and SCOPUS indexed publication,Impact Factor 2.23, August, 2015

4. **Md Whaiduzzaman**, Abdullah Gani, Nor Badrul Anuar, Muhammad Shiraz, Mohammad Nazmul Haque and Israat Tanzeena Haque, " Cloud Service Selection using Multi-Criteria Decision Analysis", The Scientific World Journal, Published, March 2014, ISI Indexed Q1/Tier1, Impact Factor 1.73

5. **Md Whaiduzzaman**, Anjum Naveed, Abdullah Gani " MobiCoRE: Mobile based Cloudlet Resource Enhancement for Optimal Task Response", IEEE Transactions on Service Computing,ISI and SCOPUS indexed publication, Revision, Tier1/Q1, Impact Factor 3.049, June, 2015

Journal Publications as co-authors:

6. Ejaz Ahmed, Adnan Akhunzada, **Md Whaiduzzaman**, Abdullah Gani, Siti Hafizah Ab Hamid, Rajkumar Buyya, "Network-centric Performance Analysis of Runtime Application Migration in Mobile Cloud Computing", Published in Special Issue on Mobile Clouds in Elsevier Journal Simulation Modelling Practice and Theory (Q-2, Impact Factor 1.159), July, 2014

7. Han Qi, Muhammad Shiraz, Abdullah Gani, **Md Whaiduzzaman**, Suleman Khan , "Sierpinski Triangle Based Data Center Architecture in Cloud Computing", Published in August 2014, in Journal of Supercomputing, ISI Indexed Q2, Impact Factor 0.917

8. Abdullah Gani, Golam Moktader Nayeem, Muhammad Shiraz, Mehdi Sookhak,**Md Whaiduzzaman**, Suleman Khan, "A Review on Interworking and Mobility Techniques for Seamless Connectivity in Mobile Cloud Computing ",Published in Journal of Network and Computer Applications ,ISI-Indexed Tier1/Q1, Impact Factor 2.23, November 2014

9. Muhammad Shiraz, **Md Whaiduzzaman**, Abdullah Gani, "A Study on Anatomy of Smartphone", Accepted for Publication in Journal of Computer Communication & Collaboration,Canada 2014.

Conference Publications:

1. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed, "TOWARDS ENHANCING RESOURCE SCARCE CLOUDLET PERFORMANCE IN MOBILE CLOUD COMPUTING", Fifth International Conference on Computer Science and Information Technology, CCSIT-2015, Sydney, Australia, February 2015

2. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed, " An Empirical Analysis of Finite Resource Impact on Cloudlet Performance in Mobile Cloud Computing", Conference, CEET-2014, Kuala Lumpur, Malaysia, April 2014

3. **Md Whaiduzzaman**, Abdullah Gani, Anjum Naveed, " PEFC: Performance Enhancement Framework for Cloudlet in Mobile Cloud Computing", IEEE & Conference, ROMA-2014,Kuala Lumpur, Malaysia, June 2014.

4. **Md Whaiduzzaman**, Abdullah Gani, " Measuring Security for Cloud Service Provider : A Third Party Approach", IEEE & ISI indexed International conference on Electrical Information & Communication Technology (ECIT), Khulna, December 2013