Multi-Access Edge Computing Assisted Mobile Ad-hoc Cloud

Jay Kumar Bhuchhada

Thesis submitted in partial fulfillment of the requirements for the

Masters of Applied Science

In Electrical and Computer Engineering

uOttawa

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Dedicated to my dad Ashok, my mom Jayshree and my brother Raj

Table of Contents

Abstract

Mobile Ad-hoc Cloud offers users the capability to offload intensive tasks on a cloud composed of voluntary mobile devices. Due to the availability of these devices in the proximity, intensive tasks can be processed locally. In addition, the literature referred to in the text, distinguishes a specific class of application to be well addressed when processed at the user level. However, due to lack of commitment, mobility, and unpredictability of the mobile devices, providing a rich ad-hoc cloud service is challenging. Furthermore, the resource availability of these devices impacts the service offered to the requester.

As a result, this thesis aims to address the challenges mentioned above. With the support of Multi-Access Edge Computing, a mobile ad-hoc Infrastructure as a Service composition framework is proposed. An ad-hoc application server is designed to operate over the MEC platform to compose and manage the mobile ad-hoc cloud. The server uses the information provided by the MEC services to compose volunteer resources for a given request. As well, a heuristic approach for a multi-dimensional bin packing technique is considered, while extending the Euclidean distance for sub-tasks selection. In addition, to address the lack of resource availability, an architecture for MAC using SDN is proposed. The logically centralized controller works with the application server to migrate requests seamlessly from one region to another. Inspired by the benefits of the MEC, a mobility mechanism is introduced to address the movement of the participants. Finally, based on the evaluation, it was observed that the proposed MAC framework not only provided better use of resources but also provided a consisted and scalable service.

Acknowledgement

First, I would like to express my deepest gratitude to my supervisor, Dr. Ahmed Karmouch, for allowing me to learn from, work with and mentored by him. I will always remember his lessons and advice. This dissertation would not have been possible without his constant supervision and support. I would also like to thank Dr. Abdallah Jarray for contributing valuable suggestions and remarks associated with the work.

I want to thank my fellow lab mates Dr. Heli Amarsinghe, Ouassim Karrakchou, and Hassan Shaukat at the IMAGINE Lab. Their life teachings, encouragement, and inspirational talks changed the outlook of my life.

Further, I would like to thank my friends Satish Prabhu, Ramneek Singh, Luc Paquin, Caroline Meriam, Shivanghi Raghav, and Yu Zhu for standing beside me in every hardship and believing in me. They have and will always be my source of inspiration.

Finally, my sincere thanks to my family, my dad Ashok Bhuchhada, my mom Jayshree Bhuchhada and my brother Raj Bhuchhada, for supporting me in my every decision. I consider myself blessed for being raised in such a great family.

List of Acronyms

Acronym	Expansion
AP	Access Point
API	Application Programming Interface
ARC	Ad-hoc Request Controller
ARO	Ad-hoc Request Optimizer
AWS	Amazon Web Services
BOINC	Berkeley Open Infrastructure for Network Computing
BPP	Bin Packing Problem
CAGR	Compound Annual Growth Rate
CAPEX	Capital Expenditure
CC	Cloud Computing
CP	Combination Pack
CPU	Central Processing Unit
D2D	Device to Device
DC	Data Center
DCN	Data Center Network
ED	Euclidean Distance
E. ED	Enhanced Euclidean Distance
EPC	Edge Packet Core

Acronym	Expansion
ETSI	European Telecommunications Standards Institute
FF	First-Fit
FFD	First-Fit Decreasing
HDD	Hard Disk Drive
IaaS	Infrastructure-as-a-Service
ICN	Information Centric Network
IP	Internet Protocol
ISG	Industry Standard Group
KPI	Key Performance Indicator
LTE	Long Term Evolution
MAC	Mobile Ad-hoc Cloud
MCC	Mobile Cloud Computing
MEC	Multi-access Edge Cloud
MEO	Multi-access Edge Orchestrator
MEPM	Multi-access Edge Platform Manager
MMM	Mobility Manager Module
MTSO	Mobile Telephone Switching Office
NFV	Network Function Virtualization
OS	Operating System
PaaS	Plaform as a Service
PP	Permutation Pack
QoE	Quality of Experience
QoS	Quality of Service
RA	Request Allocator
RAM	Random Access Memory

Acronym	Expansion
RAN	Radio Access Network
REST	Representational State Transfer
RNIS	Radio Network Information Service
RTT	Round Trip Time
SaaS	Software as a Service
SDN	Software Defined Networking
SETI	Search for Extraterrestrial Intelligence
TTL	Time To Leave
VANET	Vehicular Ad-hoc Network
VBPP	Vector Bin Packing Problem
VC	Volunteer Computing
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

List of Figures

List of Algorithms

Chapter 1

Introduction

Rapid advancement in silicon technology has increased the popularity of computing devices. Currently, there are nearly 2 billion computers (e.g., individual and enterprise computers) and almost 9 billion mobile devices (e.g., smartphones and tablets) already connected to the internet [1,2]. However, computers owned by individuals and enterprises are estimated to remain idle for approximately 97% of the time [3], whereas mobile devices for almost 60% of the time [4]. Aggregating the idle computing capacity of these devices can result in an inexpensive, environmentally friendly cloud infrastructure.

Ad-hoc cloud computing is a form of distributed computing, which allows composing intermittent, nonexclusive, under-utilized device resources that are primarily used for some other tasks [5]. Unlike a data center cloud, resources in an ad-hoc cloud are not exclusively committed to offering cloud services, but often volunteer to process tasks for a short period. The volunteer devices, when used to compose an ad-hoc cloud, consist of mobile devices, also called participants, it refers to Mobile Ad-hoc Cloud (MAC) also known as mobile clouds [6]. In the places of mass gathering such as concert halls or museums, mobile ad-hoc clouds can be used for resource augmentation of a nearby mobile user application. Thus, improving the resource utilization of mobile devices.

Providing a reliable cloud service over mobile devices for resource augmentation, however, involve some challenges such as heterogeneous mobile resources, lack of commitment from the participants, mobility of the mobile user, limited resources availability, and unreliable wireless communication [7]. Nevertheless, these challenges are worthwhile to conquer in order to tap the tremendous amount of computing from devices available within the vicinity. Because of mobile cloud's deviation from a well-provisioned remote cloud-based computing, issues such as self-optimization and management, security, trust, and resiliency requires to be revisited [8].

The dissertation addresses the challenges of resource allocation in a spontaneously formed infrastructure composed of mobile devices. This chapter briefly outlines the motivation, objectives, and contributions of the proposed work and finally, the organization of the remaining chapters of the thesis.

1.1 Motivation

Over the past decade, the number of mobile devices increased dramatically; with almost half a billion devices added in 2016 alone [2]. These devices, in recent years, have enriched the user experience through the use of rich mobile applications such as Augmented or Virtual Reality (VR), and multiplayer online gaming. In addition, future trends in internet traffic estimate a significant increase in mobile traffic [9]. However, despite the growing demand for these rich applications from the end user, mobile devices are still lacking sufficient processing, networking capacity, battery life, and storage [10]. Even though next generation devices are assumed to have an adequate computing performance, a sophisticated energy and storage mechanisms still require an attention [11].

As a solution, researchers introduced a technique called computation offloading,

which partially or entirely offloads and executes intensive tasks of mobile application on an external infrastructure called the clouds [12]. One of the approaches to offer cloud resources is to make use of the remote cloud. However, the remote cloud often suffers from high access latency, packet loss, and sometimes limited reachability. As a result, using a remote cloud for a time-sensitive application is not feasible. Furthermore, Cloud Service Providers (CSPs) have limited access to optimize the end-to-end connection between the resources and the requester, when the cloud resources are accessed over WAN networks, e.g., LTE. Therefore, placing the cloud server at the edge close to an end user is considered to be effective in mobile cloud computing [13]. Also, based on the analysis performed by [16], there are certain classes of services and applications that are specific to the location and are usually executed at a particular time, for instance, recording and processing a video at a stadium or a concert hall. These applications are best addressed at the user level.

Mobile Ad-hoc Cloud leverages computing of nearby mobile devices to provide cloud infrastructure services to an end user. Since mobile devices remain idle for 60% of the time, every interested participant in the mobile cloud can be considered as a potential provider. The low-cost cloud environment can be spontaneously deployed over Mobile Ad-hoc Networks (MANETs) [6, 17–21] without relying on existing wireless infrastructure, e.g., LTE/Wi-Fi. MANETs are usually formed using short-range wireless Device to Device (D2D) communication, e.g., Wi-Fi Direct [18]. However, the use of short-range communication and lack of wireless infrastructure fails to discover potential neighboring devices that are further in the network, thus failing to offload the request for processing optimally. In addition, the absence of a central administrator forces cloud service requester to spend energy for creating and managing the mobile cloud [22]. Although some of the research works [24–27] include a centralized

composer; they fail to use real-time context information of the participants to compose the infrastructure. Therefore, there is a need for a centrally managed mobile cloud formation approach, which uses existing wireless infrastructure to compose and manage the volunteer resources for the requester using real-time context information.

Multi-Access Edge Computing or MEC places an edge server within the Radio Access Network (RAN) and collects the necessary radio, location, and bandwidth information of the users in real-time [14]. Edge applications, running on MEC makes use of this information to provide better QoS to an end user. Motivated by the benefits of edge clouds, in this thesis, an MEC assisted MAC has been proposed. The architecture proposes to use an ad-hoc application server, which is located within MEC. The server is responsible for creating and managing the MAC. Real-time information given by MEC is used for resource allocation, which optimally selects the resources for an IaaS request.

Noticeably, participants of MAC are mobile and have complete control over their resources. Thus, guaranteeing any application with uninterrupted service in the presence of mobility is challenging. Also, it is possible that some regions are overwhelmed by the incoming request and have a limited supply of voluntary resources, whereas others have vast pool resources sitting idle. Therefore, a technology that can seamlessly migrate the request as well as improve the participant's resource utilization within a region is required.

Recently, Software Defined Network (SDN) [28–31] has received ample attention as a widely accepted solution by academia and industries for efficient network management. The literature describes the benefits of centralized control in providing network flexibility and programmability over a traditional networking technique. One of the features of SDN is resource abstraction, which hides the underlying network heterogeneity, making it viable for resource and request migration in a cloud.

As a result, this dissertation proposes a centralized software control based on SDN to support requests migration, allowing to compose requested resources in a neighboring MEC region. Furthermore, real-time services provided by the MEC and rich network control of SDN is used to assist the mobility of a resource provider, which is a part of a composed MAC.

1.2 Thesis Objectives

The main objective of this dissertation is creating on-the-fly, stable, and scalable mobile ad-hoc IaaS. Unlike a traditional cloud infrastructure where resources are static and located within a data center, resources in MAC are mobile, which makes resource selection complex. The existing MAC formation frameworks are inefficient because of the limited use of context information to compose a resource-rich MAC service. Therefore, the objectives of the conducted research are to address the key limitation of current MAC formation strategies and propose a novel solution using emerging and state of the art technologies, namely, MEC and SDN. The objectives can be summarized as follows:

- **Centrally Managed MAC using MEC**

  Typically a MAC infrastructure is created and managed cooperatively by the mobile devices. However, mobile devices are constrained by their limited battery and CPU processing. More importantly, participant resource utilization, as mobile devices lack the global view of the network. As a result, using the benefits of MEC, the first objective of this desertion is to create an ad-hoc application server, which runs on the MEC platform and has a global view of all the volunteers within the region.

Perform Optimal Sub-Task Allocation

One of the essential characteristics of the participants involved in MAC is resource heterogeneity. Since each mobile device involved in MAC contribute unique resources, it is challenging to offload sub-tasks of a request optimally. As a result, a sub-tasks allocation mechanism is to be considered, which allocates sub-tasks based on the resources offered by each addition. In addition, the sub-tasks allocation helps to lower the number of devices used for the request.

Improve the Reliability of the Offered Mobile Ad-hoc IaaS

The mobile ad-hoc cloud often lacks the responsiveness of a traditional MCC in terms of guaranteeing and offering resources reliably. The challenges are mainly a result of the device mobility, lossy wireless channels, participant resource heterogeneity, and the lack of commitment towards an IaaS request. More importantly, the selection of resource-rich and stable mobile device ensure reliable service for a request. Consequently, as the third objective of this thesis, real-time information of the mobile devices, e.g., location and signal strength, are collected using MEC services. The real-time information provides the stability of mobile devices and can be used for selecting optimal devices for the MAC.

Improve the Scalability of MAC

In MAC, providing consistent resource availability is challenging. This is mainly due to a limited number of participants or lack of interest by the volunteers to offer resources. Therefore, using the benefits of SDN, the fourth objective of this work is to create a migration strategy that migrates requests from a resource-limited region to a neighboring resource-rich region. The global view and flow-based control provided by SDN are considered suitable for transparent request migration.

1.3 Thesis Contribution

The objectives mentioned above are essential for providing an edge cloud assisted mobile ad-hoc cloud. The literature states that cloud service placed at the edge of the network offers end-user low latency and high bandwidth. Additionally, using edge cloud server as a next-generation cloud service architecture has gained momentum among the research community. Hence, considering the mentioned requirements, this thesis details the following contributions:

- **MEC Assisted Infrastructure as a Service for Mobile Ad-hoc Cloud**

  The first contribution of this work is using the MEC server to assist and address the challenges faced by the mobile ad-hoc cloud. The study explains the benefits of offloading the IaaS composer logic to a centralized server called ad-hoc application server. The server assists in performing region-wide discovery, and optimally selecting the best devices for the application requests. Furthermore, it makes use of MEC services for composing and managing IaaS resources in real-time. Since the ad-hoc application server manages the MAC composition, the IaaS requester using the service is considered as a thin client. The experimental result shows a considerable saving of time and energy while composing a MAC with MEC in comparison to a MAC managed by a requester. Additionally, the results suggest using real-time services for MAC composition, as it helps to create a stable and churn tolerant mobile cloud infrastructure.

- **Enhanced Euclidean Distance based sub-tasks scheduling for MAC**

  Performance and the stability of a composed mobile ad-hoc cloud depend on the participants selected for the composition of an IaaS request. The resources contributed by these participants introduce complexity while composing because

of their resource and network heterogeneity. Hence, as a second contribution for this work, a heuristic approach for the sub-tasks scheduling algorithm using multi-dimensional bin packing is proposed. The heuristic uses Enhanced Euclidean distance for selecting devices while considering resource and network heterogeneity of the devices. The proposed model formulates the selection in two parts (i) node score normalization (ii) sub-tasks allocation using E. ED. The experimental results show a better distribution of sub-tasks when allocated using E. ED and results in higher request acceptance while maintaining a low device count.

- **Cluster Based Request Migration for a MEC Assisted MAC using SDN**

  One of the biggest challenges faced by any MAC framework is offering consistent IaaS resources in the presence of churn. More importantly, requests can completely consume the entire supply of each device in a region, rejecting future requests. Hence, as the third contribution of this work, a request migration framework using SDN is proposed. The proposed architecture uses a logically centralized SDN controller for managing a cluster of MEC regions, where a cluster represents a group of neighboring MEC regions. The control application located within the controller optimally selects and migrates the IaaS request to an access region, which satisfies the request requirement. The framework also defines two state collection mechanism: real-time and trigger-based. The results show improvement in the acceptance of IaaS requests in an access region when migrated using SDN. Lastly, the work defines a mobility management mechanism for MAC participants using MEC and SDN. The mechanism uses MEC services to update the SDN control application once a participant leaves the access region.

1.4 Thesis Organization

The remainder of this dissertation is organized into the following chapters:

- Chapter 2 provides background and overview of the technologies used in the dissertation to propose a novel work in an area of mobile ad-hoc cloud. First, the concept of cloud computing is introduced, which is followed by the introduction to MCC. Next, the chapter presents the benefits and drawbacks of wireless access technologies in the context of cloud services and server locations. This is followed by an introduction of various mobile cloud computing techniques that are available for resource augmentation highlighting architecture as well as the advantages, and drawbacks of each technique. A primary focus is given to ad-hoc cloud computing and multi-access edge computing. Finally, the concept of SDN, along with its widely used southbound API implementation, namely OpenFlow, is described to address the challenges affecting device composition in MAC.
- In chapter 3, the benefits of MEC are taken into consideration for providing low latency, on-the-fly, mobile ad-hoc cloud infrastructure. First, the chapter describes the related work in the area of mobile cloud formation. Next, it introduces the architecture of MEC based MAC along with the benefits involved using MEC services to compose the mobile ad-hoc cloud infrastructure. Furthermore, a sub-tasks scheduling algorithm is introduced, which is designed for MAC to perform resource allocation and optimization for a request. Finally, the chapter highlights the importance of using a multi-dimensional bin packing for sub-tasks scheduling and compares the advantages and drawback of various bin-packing techniques, namely: First-Fit Decreasing (FFD), First-Fit (FF), Euclidean Distance (ED), and proposed Enhanced Euclidean Distance (E. ED).

- Chapter 4 outlines the primary steps involved in transforming the MEC based MAC into a scalable mobile ad-hoc IaaS. First, the chapter highlights the key contributions of some of the work for increasing resource availability in the mobile cloud. Then, it introduces an SDN assisted MEC based MAC architecture for migrating IaaS requests whenever IaaS resources within an access region are insufficient. Furthermore, this chapter highlights inter-regional request migration strategy featuring various state collection mechanism that affects latency and network traffic. Finally, a description is given to assist the mobility of participants across regions in an ongoing mobile ad-hoc session using real-time services.
- Chapter 5 reports the evaluation results for the experiments performed in the proposed work. Initially, the chapter describes the setup and implementation used to evaluate and measure the performance of the proposed architecture for MAC. The outcomes of the test scenarios are given based on the simulation results conducted on a network emulator called Mininet-WiFi. In addition, results from the various heuristic approach for bin packing algorithms such as FFD, FF, ED, and E. ED are compared, demonstrating packing efficiency, requests rejection, and acceptance rate. Finally, the results of an SDN assisted MEC based MAC framework are highlighted, comparing the latency and network characteristics in a real time-time and trigger-based migration strategy. As well, a comparison is made of the acceptance of an IaaS request when migrated to a neighboring region.
- Chapter 6 provides a conclusion and summarizes the proposed mobile ad-hoc cloud service framework. Finally, this chapter highlights possible future work for enhancing the proposed service framework.

Chapter 2

Background and Concepts

2.1 Overview

Mobile devices can be utilized to create on-the-fly ad-hoc cloud Infrastructure as a Service. However, this kind of implementation typically lacks the desired QoS when compared to remote clouds. While ad-hoc cloud management frameworks have been developed and studied to a greater extent, the formation of a stable infrastructure using real-time information provided by an edge cloud is still in its infancy. Hence, there is a need for a MAC resource management framework capable of providing users with low latency ad-hoc cloud while improving the stability of the composed MAC.

This chapter presents the background details relevant to the proposal of this thesis. First, the concept of cloud computing is introduced along with its service model. Then, a description of mobile cloud computing is provided, while highlighting the importance of offloading mobile applications to a cloud. In addition, different forms of cloud available to an end user are highlighted based on computing resources, proximity, and latency. Ad-hoc cloud computing and multi-access edge computing is more emphasized, as they are used in the later chapter of the proposed work.

One of the significant hurdles in restricting resources for the IaaS request to an access region is the lack of resource availability. It is especially challenging when the user demand surpasses the available resources. Hence, in the latter part of this chapter, an explanation is given of the concepts and architecture of Software-defined networking and how it can improve scalability. Finally, a brief overview is given of the OpenFlow protocol.

2.2 Cloud Computing

Cloud Computing (CC) has been portrayed as the next-generation technology to provide computing to a resource-constrained device [35]. It inherits the fundamentals of Service Oriented Architecture (SOA), providing, three layers of services based on the user requirement. The National Institute of Standards and Technology (NIST) defines these services as, Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [32]. These services can be utilized by an end user at an affordable price [33].

- Software services are often used to host web services or applications and are generally accessed by a web browser or a web-based application. However, SaaS consumers typically lack the necessary privileges to design or modify the underlying infrastructure, e.g., the operating system.
- Platform services provide necessary tools and libraries to develop software applications for the cloud infrastructure. The platform provides API's to access these tools and libraries. Like SaaS, PaaS consumers are not allowed to configure the underlying infrastructure but mostly have control over the deployed software applications running on the platform.

- Infrastructure services are the most integral part of any cloud computing architecture. They provide computing and storage resources for other services such as the platform and the software. The IaaS consumers, unlike the SaaS and PaaS, are usually equipped with the highest degree of customization. They can choose the desired operating system, type of the virtual machine, type of service orchestrator, or a job management system to deploy PaaS or SaaS.

Furthermore, cloud computing offers on-demand and elastic resources that can be rapidly provisioned and released with minimal effort from a service provider [32].

2.3 Mobile Cloud Computing

With the increase in the popularity of interactive mobile applications (e.g., virtual reality) and innovations in mobile device technology, smart mobile devices (e.g., smartphones an tablets) have become a part of everyday life. Unfortunately, innovations in mobile device technology have been slow to adapt to the increasing demands of such mobile applications. Hence, many applications do not meet the resource requirements due to constraints such as [10]:

- **Inadequate Mobile Resources**: When compared to static devices such as desktop computers, mobile devices usually have limited CPU, RAM, and Storage space.
- **Finite Energy**: Although, battery life has increased over the past few years, the use of rich mobile applications exceeds the energy demands.
- **Unreliable Wireless Channel**: Mobile devices uses wireless technologies to

access the web, inheriting the problems associated with the wireless channel. These problems include varying bandwidth, delay, loss of signal, and packet.

Mobile cloud computing, over the past few years, has gained significant attention from both academia and industries. It has introduced a new paradigm for mobile users to offload application data from mobile devices to a powerful, dedicated, and centralized platform located in the clouds [34]. Authors in [35] define MCC as:

"Mobile cloud computing at its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and MC to not just smartphone users but a much broader range of mobile subscribers"

Figure 2.1 shows the general architecture of MCC. The application offloading within MCC can be performed either by using a cellular network through base stations (e.g., LTE) or through Wireless Local Area Network (WLAN) (e.g., Wi-Fi). There are two forms of cloud servers available to the users, namely public cloud servers (Tier-2) and edge cloud servers (Tier-1). Public Cloud servers are usually provided by large service providers such as Amazon, Google, and Microsoft. They are designed to provide a large computing resource pool. However, due to their massive size, they are geographically placed further away from the crowded areas and hence often result in higher communication latency.

On the contrary, edge cloud servers can either be provided by the wireless service provider (e.g., Cloudlet, MEC, and Fog) or by the proximate users (e.g., ad-hoc cloud). Since edge cloud servers are physically located closer to the end user, they provide ultra-low communication latency. As well, being closer to the edge allows the

application developer or service provider to collect context information such as signal strength, location, user information, and bandwidth. The information can then be used by application or services to provide the end user a better quality of experience. However, due to physical limitations, proximate edge servers have limited computing resources.

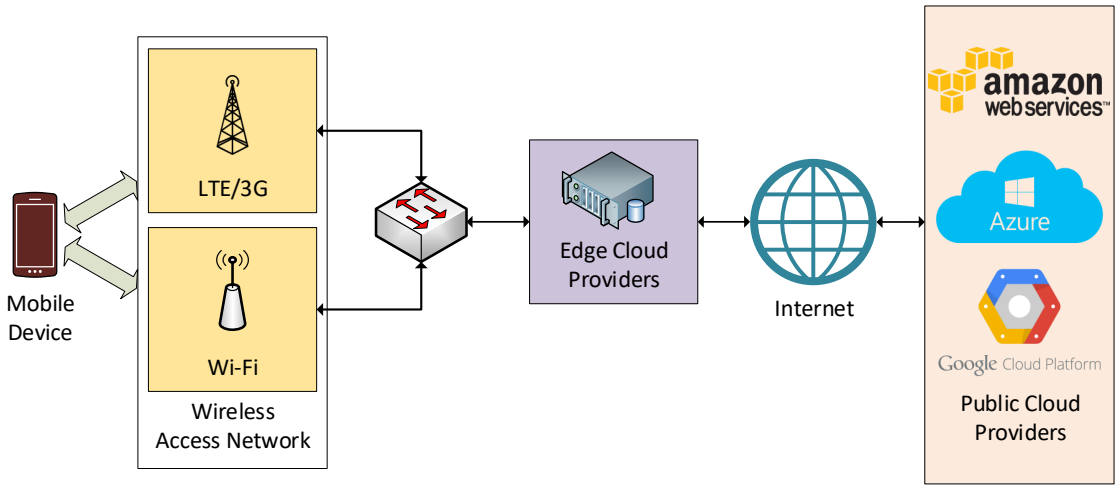


Figure 2.1: General MCC Architecture [35]

2.4 Mobile Cloud Computing Models

In recent literature, several offloading strategies have been proposed, which considers different objectives. Among them, offloading mobile applications with minimal latency is observed to impact the most in terms of end-user quality of experience. However, applications that rely on large centralized data centers for offloading and processing give preference to energy saving and execution time over latency. Architecture such as MAUI [36] and CloneCloud [37] propose offloading strategies that allow maximum energy saving with minimal interference from an application programmer. The benefits of the remote cloud servers not only include access to inexpensive computing resources but also

include highly available resources with a wide range of infrastructure (IaaS), platform (PaaS) and software (SaaS) services. Although the mobile network offers near ubiquitous coverage to access cloud services, it suffers longer round-trip time (RTT) when compared with Wi-Fi [36]. Additionally, most of the activity for offloading takes place within the home network with more than 54% of application data transferred using Wi-Fi [2]. Since cloud connectivity over WAN introduces a communication bottleneck, remote clouds may not be suitable for interactive, real-time, and multimedia applications [38]. Hence, using a remote public cloud for augmenting resource-rich mobile applications may not be an ideal solution. A plausible solution to the problem of a remote cloud is to use edge cloud [39].

Edge cloud computing refers to a distributed resource infrastructure where compute nodes are in a proximal location to the end user. This proximity to the end user allows service providers and application developers to leverage storage and processing capabilities within an access region. Facilitating such an intermediate computing layer between end users and the remote cloud not only ensures low latency communication but also provides higher bandwidth. Based on the type of device acting as a compute node, communication protocol, and services offered, the intermediate layer can take different forms. [40]. There exist two kinds of edge cloud implementation a) Infrastructure and b) Ad-hoc Cloud [13]. Infrastructure-based edge clouds refer to cloud resources that remain static and are provided by the wireless service provider, for instance, Cloudlet, Multi-Access Edge Computing, and Fog Computing. Ad-hoc cloud, however, refers to a shared infrastructure where nearby devices voluntarily participate in forming a cloud on-the-fly. When the devices used to form shared infrastructure consist of mobile devices, such kind of implementation refers to the mobile ad-hoc cloud. Figure 2.2 summarizes the above-mentioned approaches.

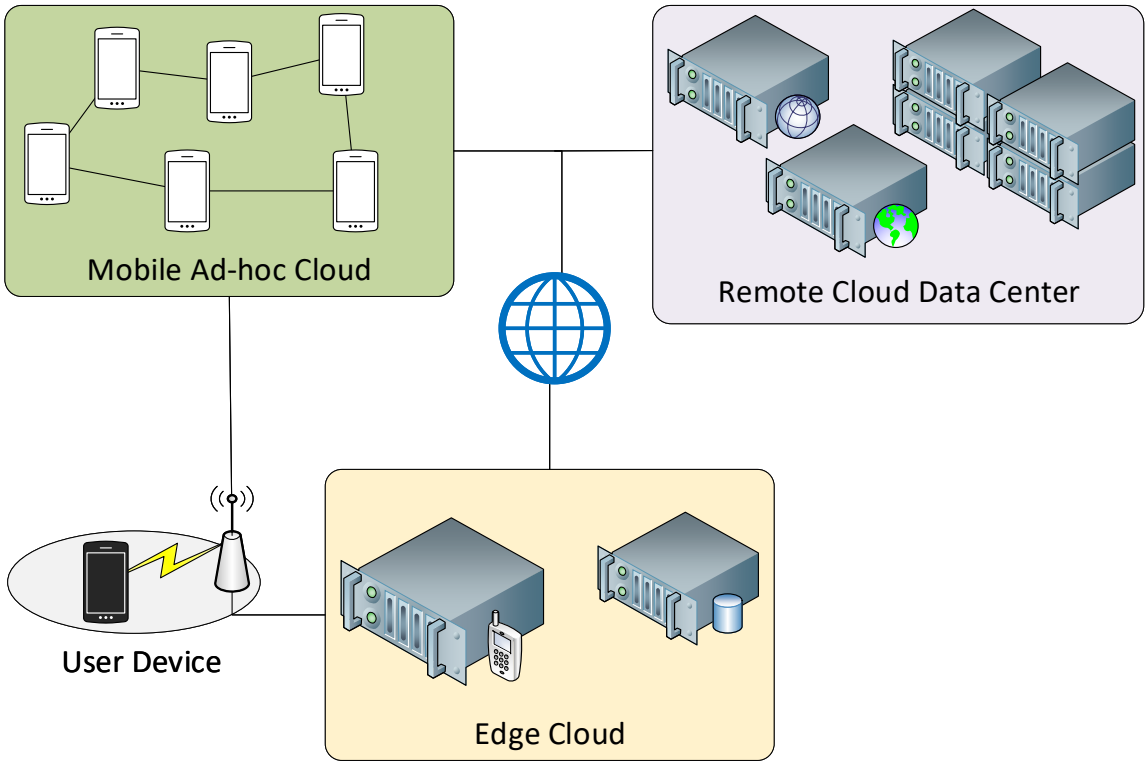


Figure 2.2: Mobile Cloud Computing Models [13]

2.4.1 Ad-hoc Cloud Computing

In recent years, sharing has observed an exponential growth and has been the subject of interest to stakeholders across the world [41]. More importantly, a positive user attitude towards sharing the commodity has fueled the growth further. Popular services include ridesharing (e.g.Uber, Lyft), room sharing (e.g.AirBnB), knowledge sharing (e.g., TaskRabbit), and many more. Unlike any other sharing service, compute sharing relies on crowdsourcing, which dynamically provisions resources. It is completely different from the traditional cloud computing that uses dedicated servers to provision user required resources. The use of intermittently available devices primarily used for something else refers to ad-hoc cloud computing. Both the client and the server include participating

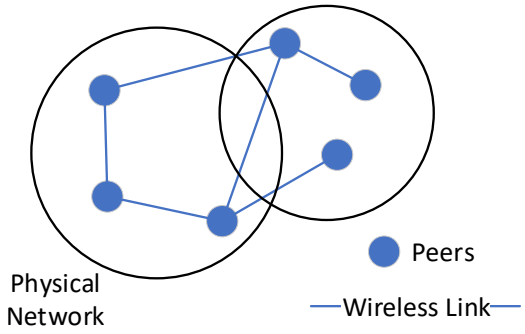
devices. More importantly, the participating device in the ad-hoc cloud does not establish any relationship with an end user, nor are the resources committed [16, 42]. Thus, device resources are often used for a small fraction of time at a specific location. Once, the resources in the ad-hoc cloud are discovered; the client device offload tasks partially or fully based on its application requirement. Similarly, volunteers that act as providers often receive some kinds of rewards or incentives in exchange for offering ad-hoc cloud service [24, 25].

While harnessing the local device resources, ad-hoc cloud computing can be classified into two categories based on the type of devices involved [24]:

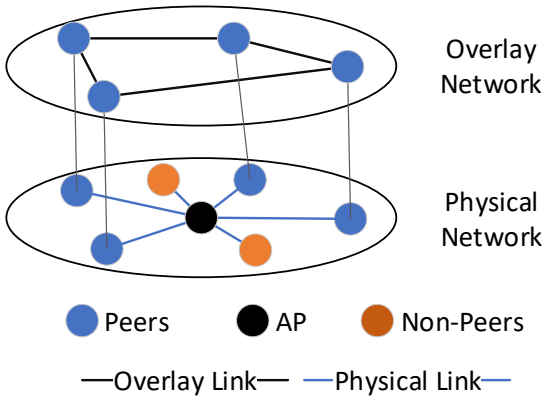
Static Ad-hoc Cloud harness computing from underutilized static devices owned by a user or an organization such as desktop computers and enterprise servers. The inspiration for static ad-hoc cloud originates from the other distributed computing architecture such as Volunteer Computing (VC) [43] and Grid computing [44]. The concept of VC has received recognition for the project such as BOINC (e.g., SETI@home [45]), which uses volunteer devices compute and storage resources to help with extraterrestrial research. This form of computing is comparatively simple in terms of design as it makes use of the client-server model [46]. Although, static ad-hoc cloud shares some of the characteristics of grid and VC, it provides new functions such as rapid elasticity, coordinated use of resources, and supports a wide range of applications.

Mobile Ad-hoc Cloud harness computing from underutilized mobile devices available locally that are willing to share their resources. The motivation for providing cloud resources over MAC is the fact that the majority of personal mobile devices (e.g., smartphones, tablets) are not always used to its maximum potential. Also, proximate users can take advantage of the idle resources of these devices in the areas where network connectivity is not available or poor and share amongst each other.

Based on the type of communication involved, a MAC can be formed in two ways [8]. **Infrastructure-Less MAC** often referred to mobile ad-hoc networks (MANETs), where mobile peers communicate with each other without any physical existing wireless infrastructure. Figure 2.3a shows high level view of MANET. Short-range wireless technologies using Device-to-Device (D2D) communication such as Wi-Fi direct [47], LTE direct [48], and Bluetooth are often used to form MANETs. Nevertheless, monitoring and detecting a failure in MANETs is extremely challenging due to network congestion, limited energy of mobile devices, and lack of resources.



(a) Mobile Ad-hoc Network

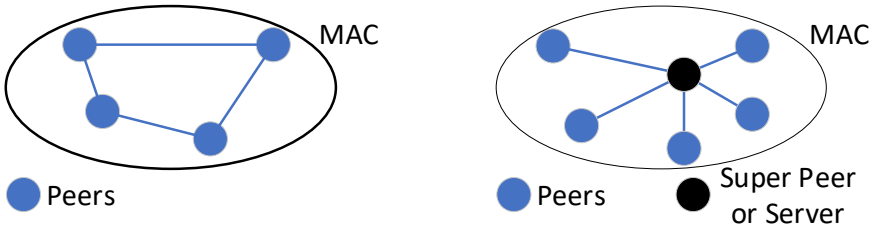


(b) P2P Overlay Over Infrastructure Network

Figure 2.3: Types of Mobile Ad-hoc Communication [8]

In a **Infrastructure-Based MAC**, mobile peers use the existing wireless network such as Wi-Fi to exchange messages with each other. The messages are relayed using the access point as a gateway, allowing peers to discover other peers over large geographically distance. Also, the P2P overlay network built over an infrastructure based wireless network is widely used to provide popular cloud services [22, 23]. Multicasting [49] [21] can further add devices that reside outside the overlay network. Figure 2.3b shows the high-level view of P2P overlay over a Wi-Fi network. Since infrastructure-based MAC allows discovering peers at a considerable distance, the proposed work uses this technique to compose MAC.

In remote clouds, an orchestrator is responsible for allocating the client requests and for providing resources meeting the user requirement [12]. However, orchestration in a mobile ad-hoc cloud is different. In fact, based on the degree of decentralization, P2P model in a MAC can be classified as *Pure* or *user-drive* and *Hybrid* or *server-driven* [50]. Figure 2.4 show types of MAC based on orchestrator. In a *pure* MAC model, a centralized server is absent, the peers or mobile devices themselves are responsible for composing and managing the mobile cloud. However, peers often have limited battery life and compute capacity, which affects the service availability, impacting the scalability of the ad-hoc cloud. Also, in a peer-managed infrastructure, it is challenging to manage the churn (devices entering and leaving) in an ad-hoc cloud as the peers lack the global view of the entire network. A *hybrid* model addresses some of the challenges faced in a peer-managed infrastructure by allowing a centrally located entity to manage the mobile cloud. This entity is often referred to as a super-peer or ad-hoc server. The ad-hoc server has responsibilities to pass meta-information that allow other peers to connect with each other. However, the location of the super-peer or ad-hoc server impacts the service offered by a mobile ad-hoc cloud [19].



(a) Pure (b) Hybrid

Figure 2.4: Types of MAC based on Degree Decentralization [50]

Deciding whether to offload a task to the nearby ad-hoc cloud or to process locally often consider communication delay, processing delay, and queuing delay [51]. However, a requester using a mobile ad-hoc cloud for offloading has to deal with a set of challenges associated with the use of mobile devices. These challenges include lack of trust among the participants, unique resources and network conditions, voluntarily resources, and low interference [5]. Moreover, device selection based on the participant's resources can significantly affect the energy and the execution time [18]. User mobility, poor network connection, and lack of motivation are some of the other causes that can disrupt the ongoing ad-hoc session [20]. Since network conditions of the participants decide the stability and the quality of service offered to the client device [27], a task scheduling mechanism is required to offload the tasks.

Despite the reduced communication delay, the mobile ad-hoc cloud suffers from limited compute resources and unreliable resource availability. This is mainly due to user mobility, change in network conditions, as well as abrupt entering and leaving the network. Additionally, places of public gathering (e.g., shopping centers, concert halls, and museums) are the only places where the infrastructure can be composed. Hence, additional support can be taken from other MCC implementations to strengthen the limitations of the ad-hoc cloud.

2.4.2 Cloudlet

Cloudlets are computers or a cluster of computers connected by an inter-network which possess resources like a data center but on a minuscule level. Authors in [52] initially proposed the concept, where servers deployed in business premises such as coffee shops or doctor's office act as a cloudlet node. Compute placed one hop away from the user offer several benefits such as low WAN latency, higher bandwidth, and high service availability [52]. Moreover, when the connection to a remote cloud goes down, a cloudlet can be used as an anchor point [29]. The hierarchical design of the cloudlet allows offloading of intensive tasks to an aggregation point or directly to the public clouds [53]. Figure 2.5 shows the 2-tier architecture model [54].

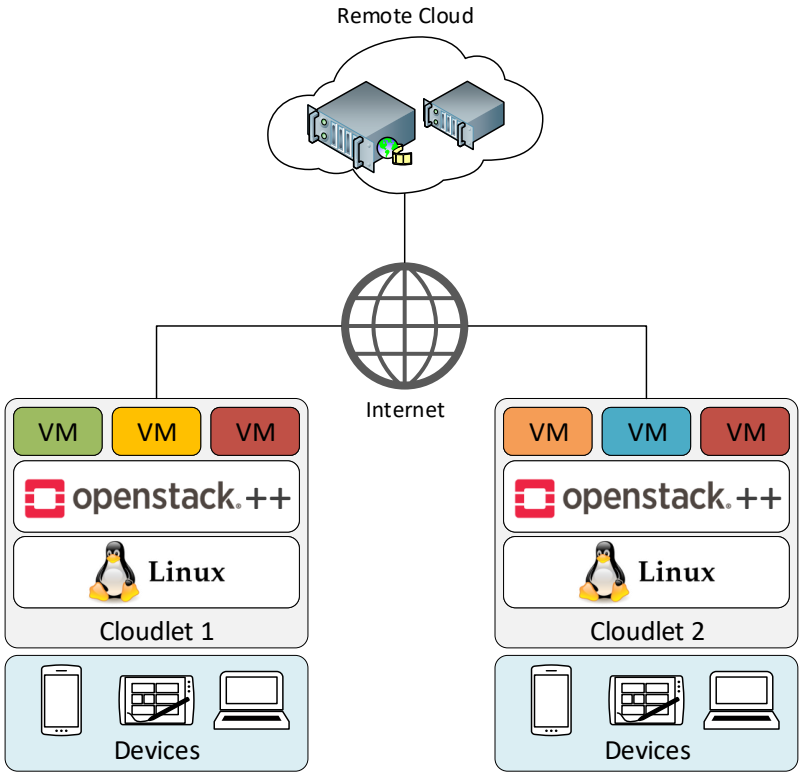


Figure 2.5: 2 Tier Cloudlet Architecture [54]

The size and resource capacity of each cloudlet can vary from a single server to the racks of a server. Additionally, the location of the cloudlet determines the user demand, bandwidth, latency, and services offered by a cloudlet server. Thus, the placement of a cloudlet server requires an extra effort in terms of planning and design [55]. Recent research [56] demonstrates that cloudlet performance outperforms the centralized cloud only when the maximum number of hops to reach the cloudlet is less than or equal to two hops. However, authors in [57] argue that cloudlets are generally not connected to the mobile network such as LTE and thus, they lack network operator-provided context information such as signal strength, bandwidth.

2.4.3 Fog Computing

Fog computing (FC) shown in figure 2.6, is a decentralized and highly virtualized infrastructure platform that provides compute, storage, and networking services between an end device and a remote cloud. It is mainly designed to support sensor devices in the Internet of Things (IoT) which require a low latency context-aware network. Unlike other edge cloud implementation, Fog Computing Nodes (FCNs) are not exclusively located at the edge of a network [58]. They can be placed one hop away within the access point, or they can be placed at an aggregation point. As well, FCNs contains a wide range of devices and are not limited to devices such as routers, switches, gateways, access points, and set-top-boxes. Thus, FCNs can communicate over different IP protocols such as Wi-Fi and non-IP communication protocols such as Zigbee, Near Field Communication (NFC), or Bluetooth low energy (BLE).

The abstraction layer of the fog computing architecture layer hides the heterogeneity of FCNs from an end user [59]. Additionally, it provides a set of functions for resource allocation, device management, security, and monitoring. The fog orchestrator receives

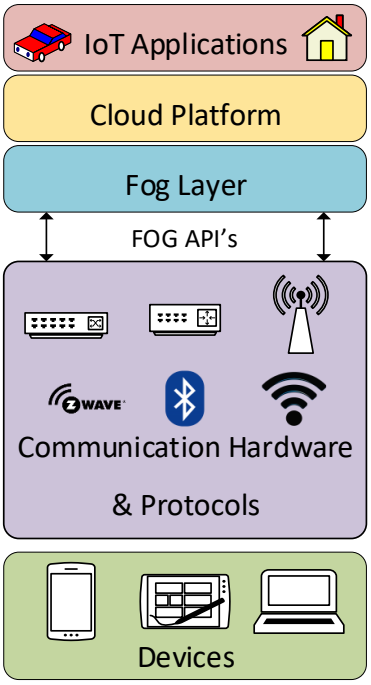


Figure 2.6: Fog Computing Architecture [58]

the user requests, which typically consist of parameters such as QoS; minimum compute resources, latency to name a few. Resources abstracted from nodes in the region using Fog APIs are matched against the requested requirements and based on availability, the most suitable device is chosen for the tasks. However, connecting every component with fog introduces network management problems, particularly the issue of providing service on top of a wide variety of fog nodes running independent and propriety protocols. [60].

2.4.4 Multi-Access Edge Computing

Multi-Access Edge Computing (MEC), formerly known as Mobile Edge Computing is an edge cloud technology initiated by the European Telecommunication Standards Institute (ETSI). It facilitates cloud-like resources at the edge of the network, which aligns with the concept put forward by the cloudlet [61]. To accelerate innovations and advancement

in the area of edge cloud, ETSI created an Industry Specification Group (ISG) called MEC ISG. The objective of this group is to create an open platform across multi-vendor cloud providers by bringing together mobile subscribers, application developers, telecom, and service providers [62]. In this section, the concepts and the architecture of MEC are provided while giving insight into the MEC infrastructure.

2.4.4.1 Concepts

MEC has received quiet popularity as an edge cloud technology that can bring the benefits of the cloud service model to end users while satisfying the QoS required by interactive and real-time applications such as automated self-driving, augmented reality and virtual reality. MEC proposes to locate an edge server within the radio access network (RAN) [14]. Besides delivering resource augmentation to delay sensitive mobile applications, MEC aims to improve content delivery by remote caching at the edge server. Also, it can reduce the burden of increased traffic in back-haul and core networks by providing services at the edge.

MEC describes a standard performance metric that is used to deploy services on the MEC environment [63]. Application and service providers can gain access to the services to provide better QoE to the end user. The MEC includes two kinds of metrics, namely: functional and non-functional. Functional metrics include parameters that impact the performance perceived by the user, often referred to as Key Performance Indicators (KPI). Examples include latency, energy efficiency, bandwidth, packet loss, and jitter. Each parameter in the functional metric needs to be defined based on a per service requirement. Non-functional metric, on the other hand, describes the performance of the services deployed. It includes lifecycle management of a service, fault tolerance, Multi-Access Edge (ME) host load, and service availability.

To safeguard the requirements laid down by 5G consortium, MEC involves radical design changes in the radio access technology and the core network. It consists of the use of Network Function Virtualization (NFV), Information-centric Network (ICN) and Software Defined Networking (SDN) [62].

ETSI has defined a set of characteristics for a MEC framework which includes [61];

- Locally available compute resources
- Low latency due to the proximity
- Context-aware application for better QoE
- MEC service to provide RAN information in real-time

2.4.4.2 Multi-Access Edge Computing Architecture

ETSI proposed architecture includes three core entities [14], MEC Host, MEC Host Level Management and MEO System Level Management. Figure 2.7 shows the high-level view of the ETSI proposed architecture.

- MEC Host is the server located at the edge of the network which consists of all the functional elements of the MEC. The platform within the Host offer functions to run and steer traffic between the application and the services. As well, the MEC host provides the application with virtual infrastructure resources such as compute, storage, and network.
- MEC Host Level Management manages the running applications and services via MEC Platform Manager (MEPM). Additionally, it manages the virtual resources used by any application through Virtualization Infrastructure Manager (VIM). Moreover, during mobility, VIM can provision resources for application reallocation.

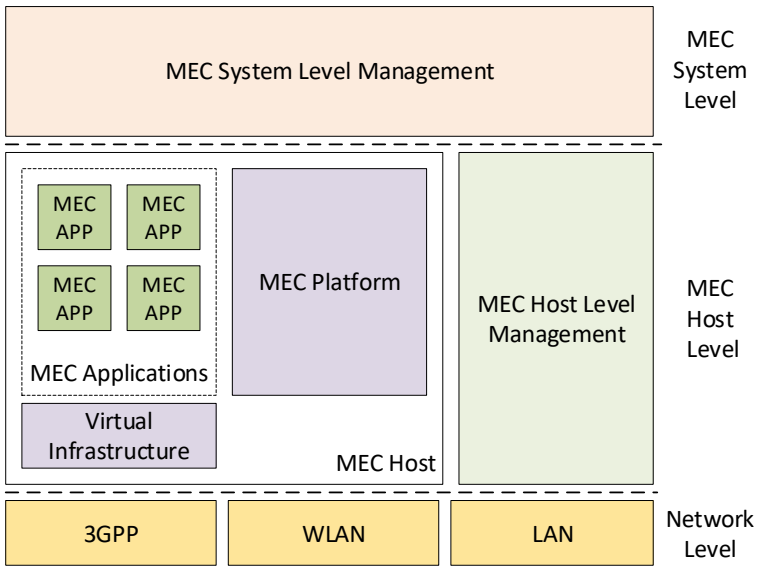


Figure 2.7: High Level View of MEC Architecture [14]

- MEC System level manages the entire MEC system, host, available resources, and services via MEC Edge Orchestrator (MEO). It maintains a global view and orchestrates the application lifecycle and ensures the integrity of the application.

2.4.4.3 MEC Application and Services

MEC application is an instance running on a virtual machine with the infrastructure provided by the MEC host. It makes use of MEC services and in some cases, can provide services to the MEC platform. Additionally, the application has certain constraints in the form of access latency, resources, and services required, that are fulfilled by MEO.

MEC includes services that are either offered or consumed by an MEC platform or an authorized MEC application. An application requires a subscription to consume these services. Currently, ETSI has defined four platform services such as Radio Network Information Services (RNIS), Location Service, User Information Service, and Bandwidth Manager Service. These services are a minimum requirement to provide

context awareness for any service or application running on MEC platform [63]. The service information is provided using standard REpresentational State Transfer APIs (REST APIs), namely: Radio API, Location API, User API, and Bandwidth API.

Radio Network Information Service

MEC provides edge applications with up-to-date information concerning the radio network conditions via RNIS. It can be used to optimize the behavior of applications that rely on network conditions. For instance, the authors demonstrate the use of radio analytics application to adjusts the video throughput of backend video server based on radio network conditions [64]. Additionally, RNIS can be used to detect changes related to user radio access bearers, which can be vital during mobility [65].

Location

The location service allows MEC application with the real-time location of the user within an access region. It can be used to recommend location-specific content such as popular videos, live feed from the stadiums or store specific flyers and promotions [64]. Furthermore, using real-time location to update neighboring regions during mobility can help to achieve handover without any service disruptions [66].

User Identity Service

User Identity service [67] allows the application to mention user specific traffic rules which can be used to deliver the required quality of service. Additionally, applications make use of tags to map a user or a set of users, which provide the linked traffic rules.

Bandwidth Manager Service

Bandwidth management service allows applications to adjust the bandwidth resource dynamically (e.g., change the size of the bandwidth). For instance, a single application can have multiple sessions running with each session having its bandwidth requirement.

2.5 Software Defined Networking

Software Defined Networking (SDN) [29, 30], is a novel platform designed to utilize and manage network resources optimally. It has brought several innovations in the field of network management [28]. These include:

- Decoupled control and data plane.
- Flow-based forwarding instead of a destination address.
- Network programmability.
- Data plane abstraction using standard communication protocol (e.g., OpenFlow).
- Centralized control using a software-based controller (e.g., Pox, OpenDaylight).

Unlike a traditional L2/L3 switch, SDN separates the core network control function from the packet forwarding and moves to a logically centralized server. Thus, the design of SDN-based network equipment is more straightforward as it acts as a simple forwarding device. The open and standard interface allows network operators to manage network functions while avoiding the limitations of a legacy network.

2.5.1 Legacy Networking and its Challenges

Traditional network architecture lacks the separation of the control plane and the data plane, which makes them rigid and inflexible to perform an upgrade or add a new feature. Hence, expensive specialized middleboxes are used to add new features to the network. As well, the placement of such devices involves intricate network design and should be placed strategically to avoid network failures. Also, the use of proprietary devices hides the knowledge of forwarding interfaces, which makes it extremely difficult to manage the network. The limitations of conventional network architecture are as follows [28]:

2.5.1.1 Use of Proprietary Network Devices

Network operators continuously seek new services and features that can satisfy rapidly changing business or user demands. However, their ability gets hindered by the vertical segments of proprietary devices that run specialized protocols and services. Moreover, the lack of widely accepted standard open interface restricts network operators from using multi-vendor network devices and services. Thus, operators cannot optimize the networking solution that is tailored specifically to their requirements.

2.5.1.2 Complex Network Configuration & Management

Networking technology currently supports a large set of protocols that are designed to interconnect devices over discrete distances, bandwidths, and topologies. However, the current configuration lacks necessary abstraction to configure a network when any new device needs to be added or removed from the topology. Furthermore, the use of low-level language for writing protocols restricts the user readability. Thus, network topologies are kept relatively static to avoid any service outages. On the contrary, virtualized service deployment is dynamic and changes as per the user requirement. VM migration is necessary to balance the server workloads across the data center, which can involve mapping a current namespace and routing address to the destination namespace and address. While existing networking solutions can deliver the required QoS per application, facilitating these resources still requires little human intervention. As well, the current network setup cannot dynamically adapt to changing constraints.

2.5.1.3 Lack of Scalability

Cloud infrastructure services such as process and storage, often provide elastic resources with rapid provisioning. As the demand for such resources increases, the IaaS

provider is required to add more supply in order to meet demand, thus, expanding the network. However, each time new requests need to service, the entire or partial part of the system needs to reconfigure. In addition, each user request is unique in terms of quality of service. Such constraints impose a challenge to implement with traditional network architecture. Specialized devices can be used to alleviate the problem, but they cannot be operationally or financially feasible for every enterprise. As well, in an ad-hoc cloud, an increase in the number of users can lead to inefficient composition and high communication overhead if not managed.

2.5.2 Software Defined Network Architecture

One of the fundamental principles of SDN is the separation of the control plane from the data plane, thus allowing network management from a logically centralized controller. As per the open networking organization, SDN consists of three layers: infrastructure, control, and application layer, as shown in figure 2.8. Due to the centralized decision, the uppermost layer is responsible for controlling the infrastructure.

The infrastructure layer is comprised of networking equipment such as routers and switches. The difference between conventional equipment and SDN-based equipment is the absence of the control software that decides the forwarding of the packet. As well, networking equipment is built using open and standard interfaces such as OpenFlow [68], which allows configuration and communication across multiple vendors. The control layer consists of a software-based controller that translates the application requirement and relays the control information to the forwarding device. In addition, the controller extracts the information about the network from the forwarding device and presents an abstract view to the SDN applications. Thus, having a global view of the network allows an application to define network-wide policies. The application layer contains

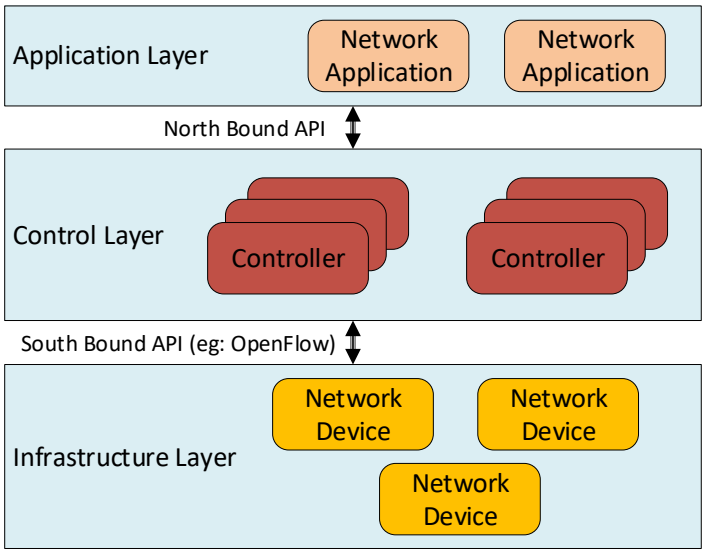


Figure 2.8: High Level Architecture of SDN [28]

SDN applications which are programs designed to implement the control logic. The application can be programmed to perform any networking function such as routing, switching, load balancing, or QoS optimizer. For instance, to implement the feature of routing, the application should be able to route traffic from one network A to another network B. To achieve this, the application collects the topology information from the controller, then decides based on this information and finally instructs the controller to issue routing specific flow. A significant advantage for a large scale network operator having an SDN based control is the ability to configure the entire network from a single logical point rather than setting each device manually.

2.5.3 OpenFlow Protocol

Southbound interfaces hold a crucial role in SDN for making a connection between the forwarding and the control plane. OpenFlow protocol developed by the Open Networking Foundation is one of the most studied and widely used southbound interfaces for SDN. It

defines a set of packet handling rules. Each rule contains a header, a counter, and a set of action field. The header field is used to classify the flow packet based on which an action is performed. The counter field keeps tracks of the flow statistics. The control-plane can be implemented with several OpenFlow SDN controllers such as NOX [69], POX [70] and OpenDaylight [71]. OpenFlow identifies network elements as switches. Internals of the OpenFlow switch running OpenFlow v1.0 [72] are shown in figure 2.9.

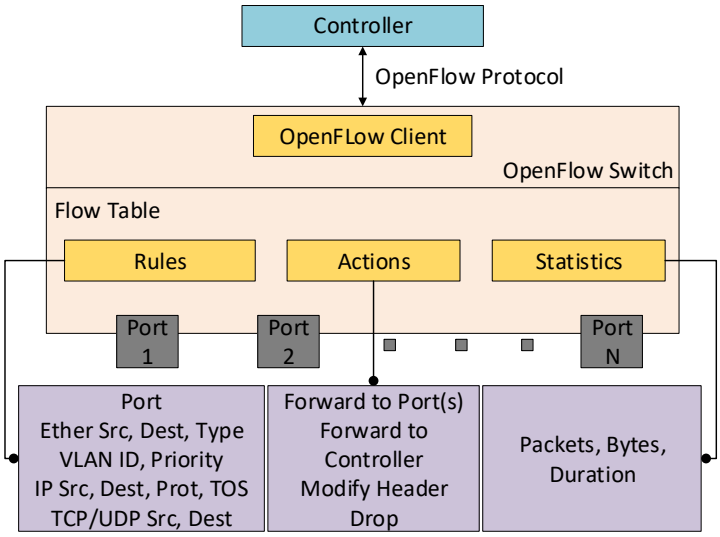


Figure 2.9: OpenFlow Switch Internal [73]

When a packet arrives at one of the switch ports, OpenFlow checks its internal flow table to determine the course of action. The decision is identified by matching an appropriate flow entry from the table. In the case of multiple flow entries that match the incoming packet, a priority value can be used to install a proper rule. If a matching rule is not available, the packet is either dropped or in most cases, a default action to forward it to the controller is used. OpenFlow controller uses two approaches to add flow entries, namely reactive and proactive. In a reactive mode, the controller installs new flow entries after receiving an unmatched packet for which no flow entry exists. On the contrary, the proactive method installs flows in advance before any matching takes

place. As well, in some cases, a hybrid approach is used that combines reactive and proactive flow generation that can install flow statically or dynamically based on a new packet. OpenFlow provides three sources of information to the controller. First, trigger-based events such as port change or link change are reported by the forwarding device to the controller. Second, flow statistics generated by the forwarding device, and third, in the event of an unknown PACKET-IN message, forwarding device send messages to the controller to determine the action. Thus, the use of open standard API not only makes it vendor neutral but also bring more innovations to networking technologies.

2.6 Summary

In this chapter, a comprehensive background study on MCC is presented, focusing primarily on a volunteer contributed resources and MEC. The rationale is provided for the importance of using an external cloud device in alleviating mobile application requirements. Also highlighted are the drawbacks of a conventional cloud deployment over mobile cloud computing. Edge cloud technology has recently emerged as an alternative solution to a distant cloud server. In this chapter, edge cloud is defined, and attention is given to various forms of compute node that is classified based on the following: server resources, location, the communication protocol used, and services offered to an end user. Focus is directed primarily to the concept of ad-hoc cloud computing, as well as a description of volunteer devices for creating on-the-fly ad-hoc infrastructure. In addition, attention is directed to the architecture and services offered by a multi-access edge computing platform. Research on network management has escalated with the introduction of the SDN paradigm. This chapter concludes with a comparison of the features and capabilities offered by SDN over traditional techniques.

Chapter 3

Multi-Access Edge Computing Assisted Mobile Ad-hoc Cloud

3.1 Introduction

The Mobile ad-hoc cloud provides the mobile user an IaaS to offload its application task on an intermittently formed infrastructure without relying on a remote cloud. However, lack of stability of the composed mobile cloud to service an IaaS request makes this kind of infrastructure, less desirable compared to remote clouds. Therefore, in this chapter, a novel MAC formation framework capable of forming a resource-rich IaaS composed of mobile devices has been proposed. The architecture integrates an edge cloud technology; namely, multi-access edge computing to form a mobile cloud for an IaaS request while using MEC services for sub-tasks allocation. In addition, the architecture is centrally managed much like cloud computing so that it can discover and carefully select infrastructure providers for an IaaS request while improving their resource utilization.

First, the chapter provides an analysis of the related work in the context of the mobile ad-hoc cloud. Then, a description is provided of the role of the MEC server

to create mobile cloud IaaS and an in-depth explanation of each functional element involved in the proposed architecture. Subsequently, the detail is provided of the MAC formation algorithm using a heuristic approach for multi-dimensional bin packing to select participants for an IaaS request optimally. Finally, an explanation is given of the effect of the proposed heuristic approach on sub-task scheduling for an IaaS request in the mobile cloud. Then a comparison is made with other heuristic approaches such as First-Fit, First-Fit Decreasing, and Euclidean distance.

3.2 Related Work

In this section, various research work conducted in the field of the mobile ad-hoc cloud is compared, mainly focusing on MAC formation strategies.

mCloud [25] proposes a mobile cloud processing and management framework using mobile devices located one hop away from the requester, referred to mDevs. The requesting device called *master* mDev is responsible for discovering, forming, and maintaining mCloud composed of *slave* mDevs. In addition, mCloud offers an attractive incentive to encourage mDevs to be part of the mobile cloud. However, the authors do not attempt to provide any sub-tasks allocation strategy that considers the resource heterogeneity of *slave* mDevs. Also, the framework relies on periodic discovery to monitor *slave* mDevs, which adds additional overhead to the network and increases the energy utilization of *master* mDev.

Ad-hoc Cloud as Service [20], introduces a protocol, namely C-Protocol to deploy mobile cloud MANETs. The protocol is designed to operate over mobile terminals in a peer-to-peer mode without relying upon any existing network infrastructure such as LTE. The publication describes several messages, e.g., $Cloud\_Setup$, $Node\_Tracing$, to

name a few, involved in C-Protocol in order to set up and manage the mobile cloud. Furthermore, C-Protocol helps to scale the mobile cloud by sending *NewP_Request* message to neighboring devices. Nevertheless, the proposed protocol does not consider any context information of the volunteer devices before allowing them to join the mobile cloud. As well, the use of mobile terminals for communication limits the range for discovering potential devices for the mobile cloud.

The authors in [18] propose a heterogeneity-aware task allocation model for the mobile ad-hoc cloud. The model makes use of various resource parameters such as device workload, number of cores, processor speed, and task size to reduce the execution time and the energy consumption of the compute-intensive task. The results achieved by the proposed framework reveal that incorporating heterogeneous parameters for task allocation guarantees shorter execution time and reduces the energy consumption of the offloaded tasks. However, the proposed work only considers the resource parameters of the volunteer device and completely negates the use of network information, failing to address the stability of the composed MAC.

Opportunistic Edge Computing (OEC) [27], introduces a centrally managed edge computing model created opportunistically using nearby volunteered resources. The proposed computing model uses the OEC broker to negotiate, create, and manage the OEC cloud. The OEC framework registers the participant resources based on their characteristics and provides them incentives as per their contribution, history, and engagement with the OEC. In addition, the OEC orchestration stack helps to separate the offered resources into slices enabling service providers to deploy their services on independent resources. While this thesis shares similar motives, the work differs from that of OEC in two ways. First, the contributor needs to register with the system to be a part of the OEC. Nonetheless, assuming that the registered device is still available for

offloading does not work well in scenarios where contributors are mobile. Second, this thesis proposes to manage volunteered resources efficiently by using the information from the MEC platform.

In summary, most of the work in this chapter considers the requester to form the mobile cloud infrastructure, which lacks the necessary information to select the volunteers for offloading. Furthermore, an additional aim is to improve the MAC formation by extending the previous contribution by Balasubramanian, Venkatraman from IMAGINE Lab on IaaS for Mobile Ad-hoc Cloud [22]. Previously, an architecture for MAC proposing to use composition score addressed some of the issues associated with the use of a mobile device as an infrastructure provider. The score considers the arrival and departure times of the providers, popularity, resources contributed, and the QoS when allocating the sub-tasks. While it improves the quality and stability of the resources offered, the IaaS requester is responsible for forming and consuming mobile cloud IaaS. Therefore, the current proposed work does not rely on the requester to form MAC; instead, it uses MEC. In this work, a centrally located ad-hoc application server runs the MAC formation logic on behalf of the mobile user. The server uses the information provided by the MEC to create a mobile cloud for an IaaS request.

3.3 MEC based MAC

This section introduces to a MEC assisted MAC formation framework for an IaaS request. The architecture strives to address the shortcomings of the MAC discussed in section 2.3.1 by employing the features provided by the MEC platform. In addition, the role of each entity associated with the architecture is defined, enabling to design a stable mobile cloud IaaS, resilient to changing network conditions and volunteer device movements.

3.3.1 System Architecture

Figure 3.1 shows the proposed architecture for the mobile cloud using MEC.

The architecture relies on three kinds of devices:

- **IaaS Consumer**: A requester device that runs the application and makes an IaaS request to offload the application data.
- **MEC Ad-hoc Application Server**: An application instance that runs on the MEC server and performs composition, management, and monitoring of IaaS resources.
- **IaaS Providers**: A pool of mobile devices that wishes to contribute their idle storage and compute resources to form a mobile ad-hoc cloud.

The following section explores the functioning of each device type in depth.

3.3.2 IaaS Consumer

A user application that runs on a resource-constrained mobile device, i.e., a requester, benefits from a rapidly formed crowdsourced infrastructure, which often gets deployed in the proximity of the user. The application remains oblivious to the offloading, and it is agnostic to the resource provisioning mechanism. Since there are specific parts of the application that require execution locally, such as a task that involves sensors, an application needs to determine which tasks to offload. The offloading engine decides whether to offload these tasks or execute them locally. It receives the task information from the tasks profiler, which profiles and divides an application task into several independent light-weight jobs called sub-tasks. If an application T requires

augmentation; then equation (3.1) gives the decomposed task s_t , where s_t represents the number of sub-tasks to be offloaded. Each sub-task that requires offloading specify their resource requirement in the form of CPU, RAM, and Storage. The process of profiling and deciding which sub-task to offload is beyond the scope of this thesis, and an assumption that the decision to offload sub-tasks s_t is already made to an ad-hoc application server, where the provisioning mechanism takes place.

$$T = \{s_1, s_2, s_3.....s_t\} \tag{3.1}$$

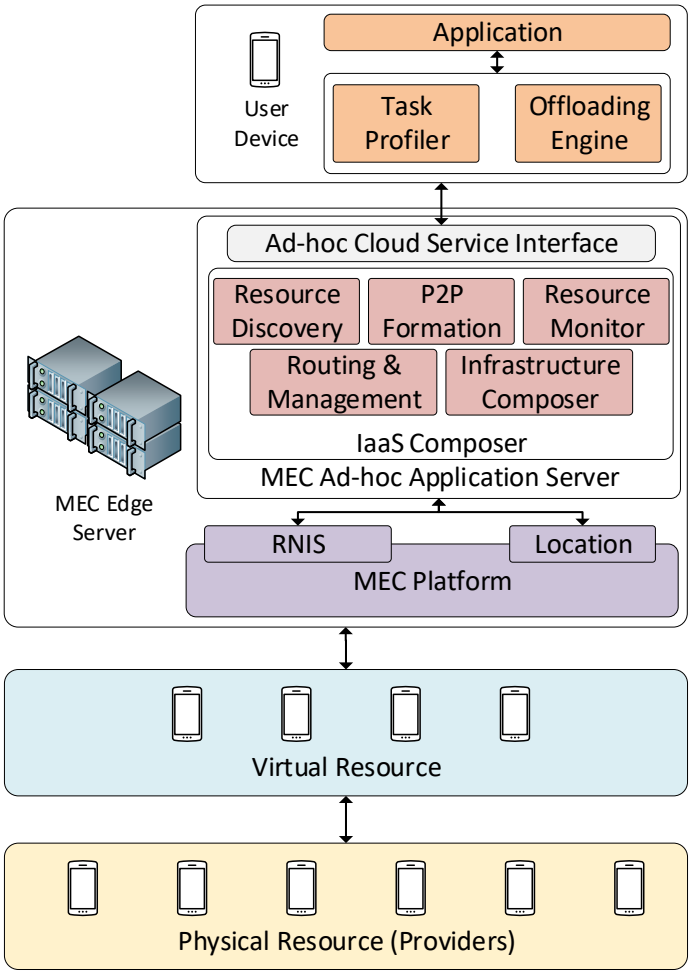


Figure 3.1: MEC Based MAC System Architecture

3.3.3 MEC Ad-hoc Application Server

Ad-hoc application server forms the mobile cloud for an IaaS request enabling the requester to offload its application on the participating devices. The elements defined within the ad-hoc application server to form a mobile cloud include discovery, selection, composition, management, and monitoring.

Once the requester completes profiling and decides to offload, it sends the sub-tasks resource requirement to the **Ad-hoc Cloud Interface**. The interface allows interaction between the ad-hoc application server and the requesting device. Furthermore, it hides the complexity of the underlying provisioning mechanism from the user application, making it completely transparent. Since the ad-hoc interface makes a resource request to the ad-hoc provisioning mechanism, logically it becomes the IaaS requester.

The request arrives at the IaaS Composer, where the infrastructure is composed and provisioned based on the requirement of the IaaS request. As users in MAC are mobile and are not bound to a given access region, it is essential to get their current idle compute and storage information before provisioning the request. Hence, **Resource Discovery** searches the access region to examine the available resources in the vicinity. The search involves discovering volunteers that are willing to contribute and desire to be a part of a mobile ad-hoc cloud. The information gathered during the discovery contains the volunteer's CPU, RAM, and Storage resource information, which when combined with the radio and location information provided by the MEC is useful to compose a reliable mobile ad-hoc cloud.

Upon discovery, the gathered resource information from volunteers, and the information received using MEC services are sent over to **Infrastructure Composer**, where a composition algorithm optimally selects the participants for the request. These

participants are later used to offload the sub-tasks. The algorithm involves a heuristic approach that uses multi-dimensional bin packing to perform sub-tasks allocation. The exact working of the algorithm is discussed in the following section of the chapter. Once the composition completes, **P2P Formation** bootstraps selected participants completing the composition of the mobile ad-hoc cloud, thus forming a P2P overlay network. Although the ad-hoc application server is responsible for forming the infrastructure, it does not participate in any sub-task execution. Instead, it remains to be a part of the P2P network, like a super peer. Figure 3.2 shows the composed P2P overlay network.

Routing & Management role begins once the P2P overlay network is composed. It is responsible for carrying sub-tasks specific dependencies that get assigned to individual devices by the infrastructure composer. It maintains key-value storage to map these dependencies on the participants and uses key-based routing to route the sub-tasks information.

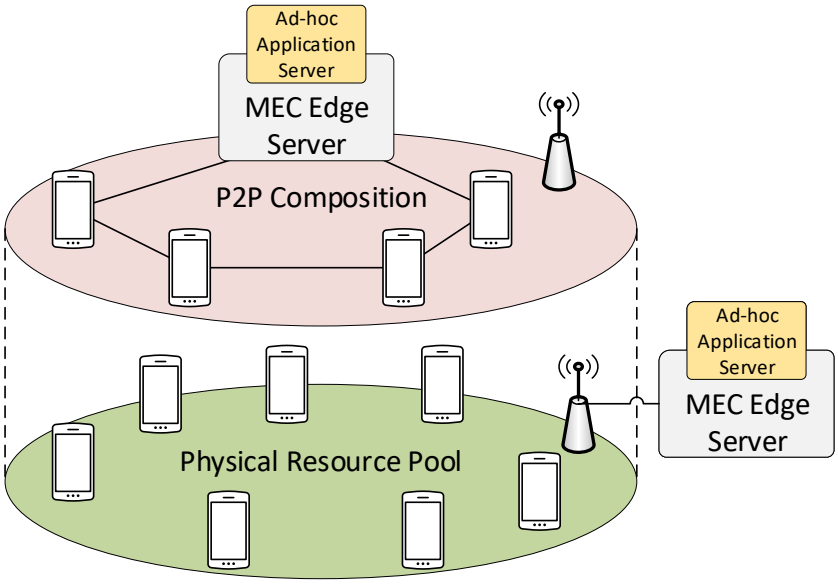


Figure 3.2: P2P Composition

Mobile ad-hoc clouds are prone to disruptions due to the highly unstable behavior of mobile devices; hence, **Resource Monitor** is used to record any fluctuations. It uses a portion of radio and location information provided by the MEC services to monitor participants that are part of the composed mobile cloud. Any failures in the mobile cloud such as participant leaving the access region or sudden loss of connectivity are reported to the infrastructure composer to reconfigure for the failed sub-tasks to continue the service. However, this chapter considers low (failures due to radio signal fluctuation) to no service disruption; modifying the network during failures is explained further in chapter 4.

3.3.4 IaaS Provider

IaaS providers in a mobile ad-hoc cloud consist of volunteer mobile devices that are willing to share their resources and are available for offloading application sub-tasks from an IaaS request. These devices provide the virtual abstraction, possess the ownership, and control the shared physical resource. In other words, these devices are the infrastructure provider for the mobile ad-hoc cloud. Since each device is different in terms of manufacturer, operating system, usage pattern and idle capacity, the set of resources contributed is unique from device to device. Thus, this makes the resources in MAC a heterogeneous ad-hoc infrastructure.

3.4 Sub-Task Scheduling

It has been established that the mobile ad-hoc cloud can be deployed on-the-fly using idle resources of the mobile devices in the region. However, there are a few challenges associated with the use of mobile devices for MAC while allocating sub-tasks on these idle

devices. These challenges include mobility, lack of commitment towards an IaaS request, unreliable wireless link, and resource heterogeneity to name few. Therefore, a scheduling mechanism needs to be used that can address the above-mentioned problems and allocate resources for the IaaS request without violating its requirement. The scheduler used for scheduling sub-tasks in this proposed work must follow certain objectives such as:

- It should allocate the number of sub-tasks considering the resource limitation of each device, as a single device may not be computationally resourceful enough to process all the sub-tasks in a given request.
- It should incorporate resource and network heterogeneity of each device while scheduling sub-tasks to ensure a stable mobile ad-hoc cloud
- It should ensure a selection of a minimum number of resource contributors for an IaaS request to limit the synchronization overhead in an overlay network. Additional overhead increases the response time of the deployed mobile ad-hoc IaaS.

Considering the objectives for a scheduler as mentioned above, bin packing is used in this research to perform sub-tasks allocation.

Sub-tasks allocation is like a Bin Packing Problem (BPP), where the volunteer devices represent bins, and the sub-tasks in a request represent items that require packing. The sub-tasks scheduler aims to not only allocate sub-tasks with the least number of devices as possible to reduce the communication overhead but also aims to improve the stability by selecting appropriate devices for the IaaS request. If it assumed that these sub-tasks, when additively placed in the same device such that they do not exceed the volunteer capacity, then the problem resembles a Vector Bin Packing Problem (VBPP). In VBPP,

each device or sub-task is characterized by a d-dimensional vector, with each vector corresponding to some resource parameter. For this research work, d corresponds to three, resembling CPU, RAM, and Storage.

VBPP is an NP-hard problem [74]. Even a single dimension when taken into consideration, the running time of this algorithm is adequately high. Hence a various heuristic technique has already been proposed. In this work, a comparison is made of two widely used variants of VBPP to allocate sub-tasks, namely FFD based heuristic, which uses greedy allocation, and dimension-aware heuristic using greedy and geometric allocation technique. The following section explains each variant in depth.

3.4.1 FFD Based Heuristics

In an FFD based variant, each item gets processed in order, and for each item, it attempts to place it in the first bin it can accommodate. If an already open bin cannot accommodate the item, it opens a new bin. The order of the items to be packed can be arbitrary in the case of First Fit (FF), and descending in First Fit Decreasing (FFD). The problem of VBPP in FFD based heuristic is solved by applying an appropriate weight function for each dimension such that a single scalar quantity represents the bin and item value. Items are then sorted in descending order in the case of FFD and placed in their appropriate bins. Authors in [75] describe two variants of FFD for the d-dimension vector. The first variant involves taking the product of each item given by equation (3.2) and the second involves summation of the product of the weighted function and the items given by equation (3.3). Weight factors $\alpha = \alpha_1, \alpha_2, ...\alpha_d$ allows for determining the importance of the vector based on its demand. However, FFD based heuristics use greedy allocation, which fails to pack items efficiently when the difference in the requirements within each dimension varies to a larger extent, i.e., if

the dimension vectors are highly co-related. For instance, some requests could be CPU intensive and low on Storage needs.

$$w(I) = \prod_{i \leq d} I_i \tag{3.2}$$

$$w(I) = \sum_{i \leq d} \alpha_i I_i \tag{3.3}$$

3.4.2 Dimension Aware Heuristic

Unlike the greedy allocation in FFD based heuristic, the dimension-aware heuristic technique takes advantage of co-relation between the d-dimensional vectors. The goal here is to fill each vector of the bin as equally as possible. For instance, if an item is CPU intensive and low on Storage, then the next item selected to balance the vectors selects an item that is high on Storage and low on computing. This process is repeated for each item until all items get packed or no more bins are available. One such technique mentioned in [76] is called Permutation Pack (PP) where the items to be packed are selected based on the current capacity of the bin. If the current order of the bin is $B1 > B2 > B3$, then PP finds an item such that it satisfies the order of $I1 < I2 < I3$. However, it may not always be possible that such an order of the items exists. Hence, another approach called Combination Pack (CP) relaxes the strict order of items by ignoring the order of smaller dimension vectors. Although PP and CP consider dimensional vectors of the bin, they use a greedy-based approach for selection.

The geometric equation-based approach described in [77], make use of Euclidean distance and the Dot product to select an item, such that it fills all dimensions equally without violating constraints of any given dimension vector. As well, both these

approaches make use of residual capacity $r(t)$ to select any item. The equation (3.4) represents the packing equation for dot product and equation (3.5) represents Euclidean distance, where α_d represents the weighted function for each dimension d. Additionally, the authors in [77] argue for using a bin centric approach instead of an item-centric, where a new bin is to be open only when no item can fit into a bin. Hence, at any given time, only one bin is open. Each time the largest item that equalizes the vectors gets selected for packing.

$$\sum_{d} \alpha_d I_d r(t)_d \tag{3.4}$$

$$\sum_{d} \alpha_i (I_d - r(t)_d)^2 \tag{3.5}$$

In the mobile ad-hoc cloud, resources provided by each device and the sub-tasks request resources can be highly co-related. Consequently, for this research work, a dimension-aware technique was utilized using a geometric equation based on Euclidean distance. Moreover, to maintain fewer mobile devices in the formed mobile ad-hoc cloud, a bin centric approach is to be used, as a new bin is opened only if no item requires further packing.

Nevertheless, some enhancements are required to make it suitable for the mobile cloud. As the resources are scarce in the mobile ad-hoc cloud due to user movement and lack of commitment, the selection algorithm should ensure that the mobile device selected has enough resources for an IaaS request. A mechanism is provided to select a mobile device that is equally optimal with respect to resources and network conditions. The device score, which is a normalized score of the device, is used to calculate and select the best device in the region. The next section explains the exact parameters involved to

calculate the normalized score and the way the scaling factor is determined. The factor relies on the popularity of the resource dimension in a region. If a given resource is in abundance, then the scaling factor tends to remain smaller, as each device can provide that resource. When a request demands a limited resource parameter, it increases the value of that quantity, leading to a higher scaling score. Since it is an extension to the Euclidean distance-based technique, the thesis work calls this approach as Enhanced Euclidean Distance (E. ED)

3.4.3 Enhance Euclidean Distance-based Resource Allocation

This section gives a detailed explanation of the heuristic approach based on E. ED. The approach describes the mobile ad-hoc cloud formation technique used to compose the infrastructure and steps for scheduling sub-tasks in an IaaS request using an E. ED-based bin packing algorithm.

Algorithm 1 explains the process of MAC formation. The process starts with a user making a request denoted by vm_{req}. It includes the number of sub-tasks that the offloading engine has determined to offload. If it is assumed there are s_t number of sub-tasks that require resource augmentation and each sub-task in s_t include their resource requirement, then it can be expressed as given in equation (3.6), where r represent the requester and C_r^t, R_r^t, S_r^t represents CPU, RAM and Storage respectively.

$$\begin{aligned} vm_{req} &= (s_1,\ s_2,\ s_3...,\ s_t)\ where, \\ s_1 &= (C_r^1,\ R_r^1,\ S_r^1), \\ s_2 &= (C_r^2,\ R_r^2,\ S_r^2), \\ &\vdots \\ s_t &= (C_r^t,\ R_r^t,\ S_r^t) \end{aligned} \tag{3.6}$$

Algorithm 1 MAC Formation

Input: client Request: vm_{req}

$vm_{req} \leftarrow$ **User Device make an IaaS request**
$resourceDiscovery() \leftarrow$ **Broadcast UDP message**
$vm_{recv} \leftarrow$ **Node Information**
$rssi, d_{AP}, P_r \leftarrow$ **Information from MEC**
for *node in* vm_{recv} **do**:
    $trustScore = avg(rssi, d_{AP}, P_r)$
    $N_p = \sum_{i \epsilon I} (\alpha_i * n_i)^2 * trustScore$
end for
$vm_{comp} \leftarrow eDBpp(vm_{req}, vm_{recv}, N_r, node_{ip})$
bootstrap vm_{comp} **with MEC Server**
Offload sub-tasks to vm_{comp}
getResults()

Upon receiving the request, the ad-hoc application server performs resource discovery by making a UDP broadcast. Mobile devices that are interested respond to the broadcast request, replying with their resource information. If there are n devices that are discovered and wish to participate, then vm_{recv} represents reply from all n devices given by equation (3.7), where C_p^n, R_p^n, S_p^t represent CPU, RAM, and Storage, and p is for the provider. Additionally, MEC platform API provides radio and location information of each device which is given by, $rssi_n$, d_n, P_n where $rssi$ represents received signal strength, d is the distance between AP and user device and P is the popularity score.

$$\begin{aligned} vm_{recv} &= \{n_1, n_2, n_3 \ \dots \ n\} \ where, \\ n_1 &= (C_p^1, \ R_p^1, \ S_p^r), \ (rssi_1, \ d_1, \ P_1) \\ n_2 &= (C_p^2, \ R_p^2, \ S_p^2), \ (rssi_2, \ d_2, \ P_2) \\ &\vdots \\ n_n &= (C_p^n, \ R_p^n, \ S_p^n), \ (rssi_n, \ d_n, \ P_n) \end{aligned} \tag{3.7}$$

The received signal strength (RSSI) of a device n given by $rssi_n$ represents the

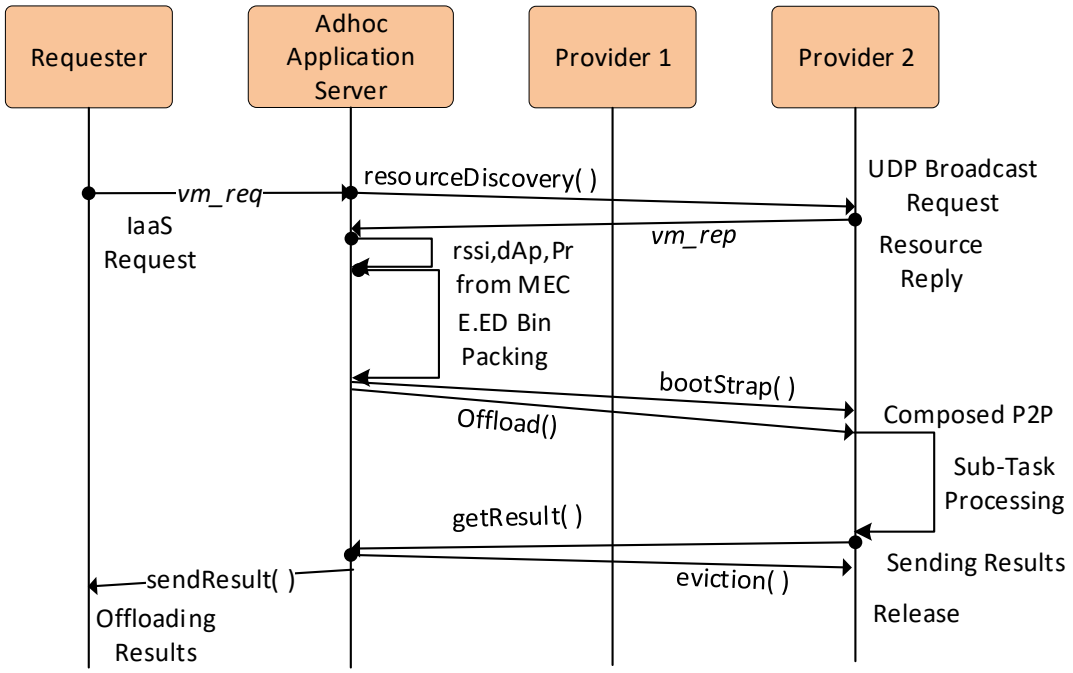


Figure 3.3: Sequence Diagram for MAC Formation

network quality experienced by a mobile device. Higher RSSI yields better bandwidth and provides low communication delay. d_n represents the distance between the access point and the device, which helps to localize the mobile device within the network. Devices that are closer to the center take more time to leave the region, giving enough time to complete executing the sub-tasks. It is quite common in ad-hoc infrastructure to come across devices that are unreliable and greedy, which never contribute but instead utilizes the resource. Therefore, in this research, a score given by P_n is used to judge the device reliability in the region. Each time a participating device completes the MAC IaaS session, the score increments, and vice versa.

Then, for each device in vm_{recv} a calculation is made of the normalized resource score N_p represented by equation (3.8). The trust score of a n^{th} device given by $avg(rssi_n, d_n, P_n)$ is the average of signal strength, distance, and popularity score. If

one of the components is lesser than the other two, it reduces the overall score. The scaling factor *alpha* of a i^{th} dimension depends on the sub-tasks requirement and the resource availability. Upon completing the selection process, the devices selected are bootstrapped, and offloading begins. Kademlia routing protocol is used to map the sub-tasks on the selected device in a structured P2P overlay network. Figure 3.3 shows the sequence diagram involved to form a mobile ad-hoc cloud.

$$N_p = \left(\sum_{i \epsilon I} \alpha_i * n_i \right) * trustValue_i^2 \tag{3.8}$$

Algorithm 2 Euclidean Distance Bin Packing

Input: vm_{req}, vm_{recv}, $node_{ip}$, N_p
Output: vm_{comp}

$C^r, vm_{index} \leftarrow 0$
$l_{vm} \leftarrow length(vm_{recv})$
$R^r \leftarrow M^r of vm_{recv}[vm_{index}]$
$vm_{comp}.add(node_{ip})$
while vm_{req} **do**:
    $subtask = min(\delta_{ed})$
    if $subtask \neq$ None **then**
        $R^r \leftarrow R^r - subtask$
        $vm_{req}.remove(subtask)$
    else if $subtask =$ None and $vm_{index} \leq l_{vm}$ **then**:
        $vm_{index} += 1$
        $R^r, M^r \leftarrow vm_{recv}[vm_{index}]$
        vm_{comp} .add($node_{ip}[vm_{index}]$)
    else if $subtask =$ None and $vm_{index} == l_{vm}$ **then**:
        $break$
    end if
end while
if vm_{req} is empty **then**:
    return vm_{comp}
else
    $CannotAllocate$
end if

Algorithm 2 explains the execution of E.ED Bin Packing. To initialize E. ED BPP, first, a device needs to be selected that can be used to compare sub-tasks, and then decide which one to allocate. If a current device is full and cannot allocate any sub-task, the next device present in the list of the normalized score is selected. vm_{index} point to the current device in use. Also, l_{vm} is initialized, which indicates the maximum number of devices available for the composition. As mentioned earlier, the residual capacity of the bin is used to calculate the Euclidean distance. As a result, the remaining capacity R^r to the maximum capacity M^r of the selected bin is initialized. The Euclidean distance (δ_{ed}) for each sub-task is calculated using the equation 3.9, where S^r represents the i^th dimension resource requirement of a sub-task.

$$\delta_{ed} \leftarrow \sqrt{\sum_{i \epsilon I} \alpha_i (S_i^r - R_i^r)^2} \tag{3.9}$$

A Sub-task that results in a minimum (δ_{ed}) without violating any dimensional resource constraint of a bin is selected to pack and removed from vm_{req}. The remaining capacity of the bin subtracts the sub-task value, adjusting the capacity to reflect the change. If none of the sub-tasks can be selected and if the vm_{index} is still less than the maximum number of bins l_vm, then a new bin or a device is selected and initialized with its remaining capacity. The process continues until all the sub-task in vm_{req} get allocated, or no devices are remaining to initialize. If all sub-tasks get allocated, it results in acceptance or rejection.

3.5 Summary

In the context of mobile ad-hoc cloud, providing a reliable and low-cost computation mobile ad-hoc IaaS with low access latency is essential due to the constant change in

network condition and mobile user mobility. Consequently, in this chapter, a description has been provided of the architecture of MEC assisted MAC, where the ad-hoc application server that resides on the MEC server provides a mobile ad-hoc IaaS by utilizing the resources volunteered by the proximal mobile devices. The requester making an IaaS request can utilize the formed infrastructure for processing any application or service on demand. Also, a novel heuristic-based approach for VBPP has been proposed, which uses enhanced Euclidean distance to address the resource and network heterogeneity amongst the participants. The heuristic schedules resources for the IaaS request while minimizing the number of participants involved in the composed mobile ad-hoc cloud. By making use of dynamic information offered by MEC services, the scheduling algorithm selects the participants that are resourceful and are closer to the access point, thus delivering a stable mobile ad-hoc cloud IaaS.

Chapter 4

SDN Assisted MEC based Mobile Ad-hoc Cloud

4.1 Introduction

In the previous chapter, an explanation was given for the architecture of MEC based MAC the benefit of using MEC to compose mobile cloud IaaS. The architecture allowed for offloading the composition logic from the requester to a centrally located application running on a MEC platform. However, lack of commitment, mobility, and availability of the participating devices are a few of the mobile cloud characteristics, which makes it challenging to provide consistent resources. Furthermore, mobility of the participants impacts the overall service latency offered by the composed mobile cloud. Therefore, in this chapter, to improve the service availability of mobile cloud IaaS, some changes are made to the existing architecture described in Chapter 3. The architecture make use of emerging and state of the art technology called SDN. It enables the migration of a request from a resource-depleted region to a neighboring resource-rich region within a cluster. Also, the architecture manages the mobility of the participants involved in MAC to continue serving the IaaS request.

First, an analysis is done of already proposed research work related to increasing the service availability of the mobile cloud. Then, the SDN assisted architecture for MAC is described and an explanation of the roles and responsibilities of each entity involved in the architecture. Also, a description is given of the request migration algorithm using region scoring to select the region that satisfies the IaaS request. In addition, a description is given of two state collection mechanisms, namely real-time and trigger-based, and collect the information of regions to decide the region of interest. Finally, insight is provided on how to manage mobility using SDN based on the information provided by MEC services.

4.2 Related Work

In this section, a comparison is made of various research work involved in improving the service availability of the mobile cloud.

In the previous chapter, it was observed how MEC services were leveraged by the proposed MAC formation algorithm to create a stable mobile cloud IaaS. Now, consider a scenario where a region is resource deficit because of either, the participant lacks the willingness to contribute, or there is a shortage of participants available in the region to serve the IaaS request. In either situation, the mobile cloud IaaS offered to the requester gets affected, degrading the QoS.

One of the techniques to increase the involvement of infrastructure providers is encouraging with some rewards or incentives. Researchers believe that offering incentives in a cooperative environment such as in the mobile cloud, motivate the providers to contribute their resources. The authors of [80] propose a double-sided bidding mechanism. The mechanism enables providers and requesters to place a bid as

per their willingness to share, and get paid or pay to execute tasks, respectively. Authors of AMCloud [24] propose a reward system that pays participants in the form of virtual credits for every successful contribution. These credits are tied to the reputation of the participants; thus, the higher the number of contributions, the better are the rewards, encouraging to participate more. A directory-based framework [81] is suggested to calculate retribution and rewards as per the energy spent and saved by the devices. The framework makes use of a global directory, which records the rewards even after a user moves to a different region, motivating participants to continue sharing without losing any earned rewards.

Another technique to improve the resource offered in an IaaS service is adding more participants from the neighboring region or network. C-protocol [20], a protocol designed to operate over mobile terminals in a peer-to-peer mode without relying upon any existing network infrastructure such as LTE. The protocol enables the participant of mobile cloud to add their neighbors by sending $NewP\_Request$ message. In SMAC [31], an SDN assisted MAC; the requester makes use of SDN and wireless access points to discover potential contributors across the multiple regions. Once the requester selects devices from the discovery, it sends the routing list to SDN, which add flows in the OpenFlow switches to connect the requester with the participants. In addition, SDN optimizes routing by selecting the shortest path for the requester to reach the participants of MAC. AMMADC [19], a Multi-hop Mobile Ad hoc cloud computing framework, describes a similar use of the multi-hop network to harness computing from the mobile infrastructure providers. The architecture divides the ad-hoc network in multiple smaller networks by clustering. The cluster restricts the cluster-specific message within itself, which helps to reduce the overhead in the network and allow it to scale.

Nevertheless, offering an incentive to the participants do not help to improve the

resource availability for IaaS, if the region is short of participants. The offered IaaS needs to add more participants from another region to increase resource availability. However, adding participants from other regions, as suggested by the work mentioned above introduces overhead due to inter-region communication and adds additional delay as the packet needs to take multiple hops to reach the requester. In addition, lack of control and limited information about the participants from other regions makes failure detection challenging and involves complex failure management. Therefore, instead of borrowing resources located further away from the requester to form a mobile cloud, it is believed that migrating the entire request to a neighboring resource-rich region is best suitable for cooperative clouds. In this work, several neighboring MEC regions are grouped into a cluster that is managed by an SDN controller. As mentioned by [19], clustering reduces the overhead in the network and manages the mobile cloud with minimal complexity. The specifics of the cluster, such as its formation, size, and type are beyond the scope of this thesis.

4.3 SDN Assisted MEC Based Mobile Ad-hoc Cloud

In this section, an SDN assisted architecture for the MAC is introduced to improve and scale the offered mobile ad-hoc IaaS. The architecture makes some changes to the proposed architecture mentioned in section 3.3 and addresses limited resource availability by employing SDN.

The proposed architecture has four design objectives. First, improve the service availability of the mobile ad-hoc IaaS by migrating an IaaS request to a neighboring region within a cluster. Second, hide the complexity of the IaaS request migration from the IaaS consumer. Third, improve the resource utilization of each MEC server

within a cluster and finally, manage the mobility of a participant that is a part of a composed mobile ad-hoc cloud. Thus, to meet the above-mentioned objectives, the proposed architecture have:

- Designed a mobile ad-hoc cloud service architecture using SDN that migrates the IaaS request within a cluster of MEC servers.
- Integrated a logically centralized SDN controller to discover and determine an optimal MEC server that uses region scoring to satisfy an IaaS request requirement.
- Designed a mobility management solution for the participant of MAC using MEC services to preserve the composed mobile ad-hoc IaaS session.

4.3.1 System Architecture

Figure 4.1 shows the functional elements of the SDN assisted mobile ad-hoc cloud. The architecture includes three primary entities, such as:

- **SDN based Network**: It includes OpenFlow switches and wireless access points that allow communication between MEC regions and the Ad-hoc Request Controller (ARC).

  MEC Region: A MEC server that runs an ad-hoc application server to compose mobile ad-hoc cloud for an IaaS request by consuming locally contributed resources.

  Ad-hoc Request Controller: Centralized SDN-based controller responsible for discovering, selecting, and migrating an IaaS request from one MEC region to another.

In the following section, the functioning of each entity is explored in detail.

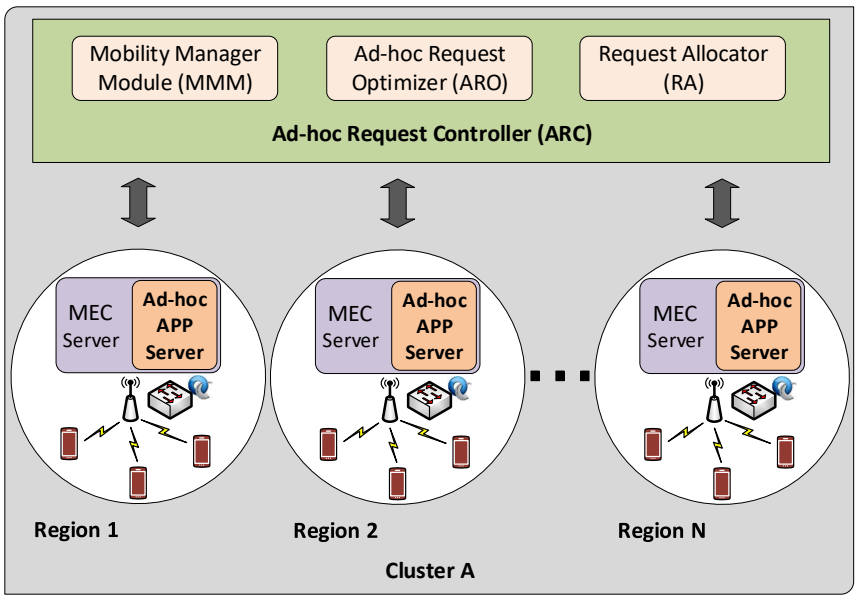


Figure 4.1: Scalable Mobile Ad-hoc Service Architecture

4.3.2 MEC region

A MEC region consists of a MEC server that runs the ad-hoc application server to provide mobile ad-hoc IaaS within an access region. It uses the MAC formation algorithm mentioned in Chapter 3 to compose locally available resources for an IaaS request. In addition, each MEC region is equipped with an SDN based network consisting of access points and switches and uses OpenFlow protocol to exchange information with the control application located in the ARC. In the case of limited resource availability, the MEC region with the help of SDN migrates the IaaS request to its cluster neighbors. The following section of the chapter explains the exact working of request migration. Furthermore, to avoid the increase in the delay due to migration, the IaaS request received from ARC is considered a higher priority over a local request. The neighboring MEC region can communicate with each other only if the control application adds specific flow, for instance, flows are added to allow servers to exchange offloading results once the processing of the migrated request is complete.

4.3.3 Ad-hoc Requests Controller

In an SDN network, an OpenFlow based controller allows customizing (i.e., add, remove, or modify) flows within the network dynamically with the help of OpenFlow based control applications. These applications use the OpenFlow protocol to define a set of actions (forwards or drop) for a certain set of flows. Hence, in the proposed architecture, **Ad-hoc Request Controller (ARC)** is defined as an OpenFlow based controller and modules within the ARC such as Ad-hoc Request Optimizer (ARO), Request Allocator (RA), and Monitor Management Module (MMM) is defined as OpenFlow based control applications. Figure 4.2 shows the service layer view of SDN controlled network. By centralizing ARC and using OpenFlow to discover network elements, ARC achieves a global view of the MEC regions present within the cluster. However, ARC lacks the local view of each region, i.e., it does not acquire the end device information, as it is only responsible for delegating the request to the selected MEC. As well, MEC monitors its region devices, and requests help only when the participant leaves the region. Furthermore, ARC can be physically placed on one of the MEC server or in a separate premise with minimum access delay. In this work, it is considered to be running on one of the MEC servers in a cluster.

ARC optimally migrates the resource allocation of an IaaS request, once received by a resource-deficit MEC region. The optimizer located within ARC, known as **Ad-hoc Resource Optimizer**, selects the region based on requested resources and the lowest link path to the originating region (where the requester is present). By communicating with both the wireless network and MEC, ARO can obtain the required information to select the region for migration. The algorithm to select the region is explained in the next section of this chapter. Once the selection is complete, ARC conveys the region information to the Request Allocator.

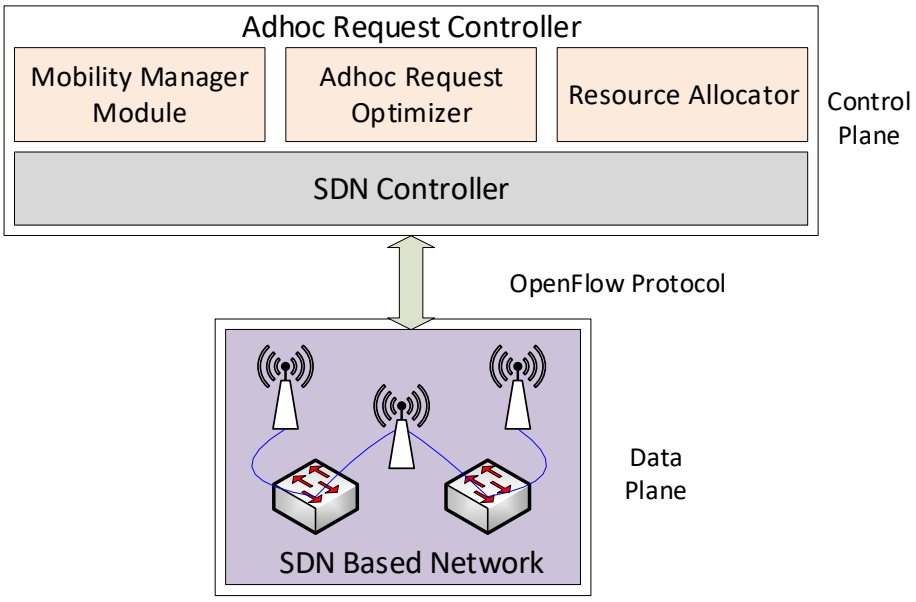


Figure 4.2: Service layer view of SDN Controlled Network

The **Request Allocator** is responsible for dispatching the IaaS requests to the ad-hoc application server of the ARO selected region. In addition, it ensures connectivity and delivery of the IaaS request results between both originating and selected regions by adding or modifying the flow rules.

Although ARC lacks local information for each region, it is still able to assist in the mobility of the participant when requested by the MEC region. **Mobility Manager Module** located within the ARC is responsible for preserving an ongoing session, i.e., when an infrastructure provider of the composed mobile ad-hoc cloud leaves the existing region. Resource monitor located within the ad-hoc application server reports the provider information to MMM. In addition, MMM adds flow in the OpenFlow switches, allowing to send the results back to the region of the composition. The exact working of how MEC services are leveraged for mobility management is explained in the following section.

4.4 Request Migration in an SDN based MAC

Resources in MAC highly depend on the interest and the availability of volunteer resources, and thus, a resource deficit region fails to provide mobile ad-hoc IaaS to offload application tasks. Unlike offering incentives or adding volunteers, as presented in section 4.2 to improve resource availability, request migration, migrates the entire request to the resource-rich region. In this work that the request migration provides two benefits for the mobile cloud. First, migration reduces the overhead and complexity of managing the inter-regional resource. The receiving ad-hoc server being solely responsible for serving the request. Also, any failures which interrupt the service are taken care of by its ad-hoc server and require no involvement from the originating ad-hoc server. Second, migration can help to load balance the regions within the cluster, allowing for the utilization of resources in each region optimally. It is highly possible that some regions receive a sudden burst of IaaS request, while some may have zero IaaS requests. Therefore, striving on the benefits involved with request migration; in this section, an explanation is given of the request migration strategy, and the region scoring is used to select the region of interest.

Prior to providing information on region scoring equation which determines the region of interest for request migration, an explanation is given of what information is collected and the methodology used to collect this information. State collection allows having updated information of a region such as their resources, latency to name a few. The resource information of a region measures if the requirements of the IaaS can be satisfied. The information includes total CPU, RAM, and Storage of all the participants in a given region. Also, link latency between the originating server where the requester makes an IaaS request, and its neighboring region, help to select a region with the lowest link

delay. The infrastructure providers for a region are volunteer mobile devices, and thus, it is vital to ensure a stable region for the IaaS request to avoid reconfiguration in the composed MAC. RSSI, location, and popularity of the participant within the neighboring region ensures the stability of the region. Based on the type of service expected, in this work, two systems are used: namely, real-time and trigger-based.

In a real-time system, a query message is issued at every short interval of time t_s to collect the state information of the regions within the cluster. The time t_s is adjustable and selected based on how soon the information needs to be collected. A shorter time t_s ensures live and updated state information but affects the network performance due to frequent exchange of the messages. Although longer time t_s involves a fewer exchange of messages, the collected information may be stale and cannot guarantee the current state of the region. On the other hand, a trigger-based system issues a query message only when a specific event occurs, such as an originating server request ARC for request migration. Although a trigger-based system guarantees the current state of the region, it waits until discovering the region information is complete. This additional wait time increases the overall service latency. Based on this scenario mentioned above, in a later chapter of this work, observation and evaluation are made on the effect of each state collection technique on delay and control message.

4.4.1 Selection of a remote MEC server for request Allocation

Initially, a requester makes an IaaS request to its local available ad-hoc server (referred to as an originating server). The request is represented by vm_{req} given by equation (3.6). Upon receiving the request, the ad-hoc server performs a local discovery using a UDP broadcast message to check the available resources in the region. If the resources discovered satisfies the request, they are allocated using the MAC formation algorithm

described in section 3.4. However, if the request lacks the resources and cannot be allocated, instead of rejecting the request, it is sent to the ARC. The ARC migrates the IaaS request to a resource-rich region available within the cluster. The IaaS request received by ARC is given by equation (4.1), where C_r^{total}, R_r^{total} and S_r^{total} represent the total CPU, RAM and Storage requirement of all the sub-tasks from an IaaS request.

$$vm_{req}^{ARC} = (C_r^{total},\ R_r^{total},\ S_r^{total}) \tag{4.1}$$

Consider there are q regions within a cluster. Each region in $q_j \epsilon q$ informs their resource and network state to the ARC. Let Q_{recv} given by equation (4.2) represent the state information received by the ARC for all regions q. Since ARC communicates directly with the region, i.e., ad-hoc application server and does not involve the direct participation of the infrastructure providers, collecting resources from each provider is not an ideal consideration. Thus, resources available in the region $q_j \epsilon q$ represents the total resources of CPU C_q, RAM R_q, and Storage S_q contributed by the volunteers of that region. Similarly, $rssi_q, d_q$, and P_q represent the average of MEC service provided information such as signal strength, distance from AP, and popularity of the provider devices in a region $q_j \epsilon q$ respectively. The above mentioned average information of the volunteers represents the overall stability of the region. Furthermore, ARC calculates link latency l_q of each region in $q_j \epsilon q$ to select the lowest link path for request migration.

$$\begin{aligned} Q_{recv} &= \{q_1, q_2, q_3\ \ldots\ q\}\ where, \\ q_1 &= (C_q^1,\ R_q^1,\ S_q^r),\ (rssi_{q1},\ d_{q1},\ P_{q1},\ l_{q1}) \\ q_2 &= (C_q^2,\ R_q^2,\ S_q^2),\ (rssi_{q2},\ d_{q2},\ P_{q2},\ l_{q2}) \\ &\vdots \\ q &= (C_q,\ R_q,\ S_q),\ (rssi_q,\ d_q,\ P_q,\ l_q) \end{aligned} \tag{4.2}$$

ARO located within the ARC uses the information Q_{recv} for region scoring to select the region for request migration. The mathematical expression used to calculate the score of the region represented as N_q given by equation (4.3). The objective of the scoring equation is to rank the region based on its resources, link latency, and stability of the region. Therefore, the region with the highest score of N_q that satisfies the IaaS request is selected. The equation calculates the score N_q in two parts, resource and the context score. The resource score uses the weight function β to calculate the importance of the resource that takes the resources of the self, the participating regions in the cluster, and the requirement of the request. The weight is then used to determine the value of the resource vector i of region q. The context score $region_q$ indicates the value of the region with the lowest link latency and the most stable participants.

Once the selection completes, ARO sends the selected region information to the requester allocator, which adds or modifies appropriate flow messages to ensure requests delivery. The Selected region upon receiving the requests, allocate request using its local resources. When the device completes processing the sub-tasks, the selected server for migration sends the request results to the originating server. Finally, the results are sent to the requester. Figure 4.3 shows the sequence diagram of request migration. Since the request migration does not involve any requester interaction with the ARC, the entire process is kept transparent to the requester. Thus, allowing a scalable mobile ad-hoc IaaS to be established in this work.

$$N_q = \left(\sum_{i \epsilon I} (\beta_i * q_i \right) * region_q^2 \tag{4.3}$$

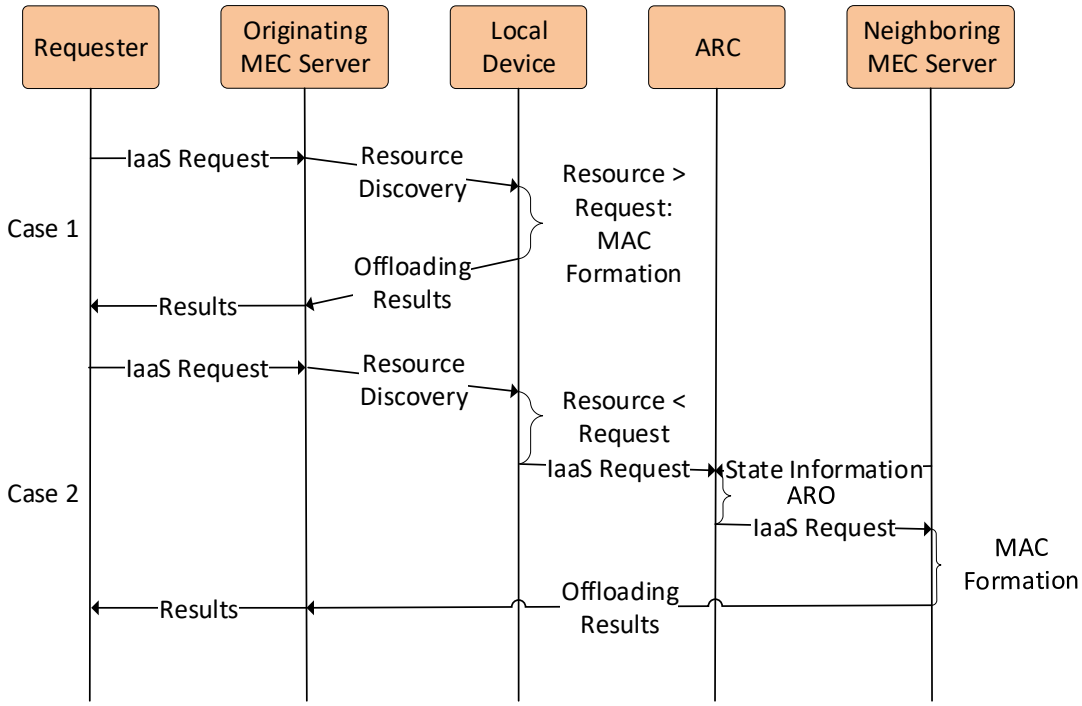


Figure 4.3: Sequence Diagram for Request Migration

4.5 Mobility Management Using SDN and MEC

This section focuses on how the mobility of the participant that is a part of a composed mobile ad-hoc cloud can be managed using SDN and MEC.

An essential characteristic of a mobile ad-hoc cloud is the lack of commitment from the participants to process the allocated sub-tasks. Since the participants are mobile and have control over their resources, there is no guarantee that they will remain in the same region until the sub-tasks execution completes. They can either decide to move to another region permanently or re-enter and re-connect with the current access point. Although most of the research work mentioned in this thesis avoids selecting such devices through optimization techniques, it is difficult to predict the change in the mobility of the participant. Another approach is to offer attractive incentives to the participants

and motivate them to remain in the region of operation. However, the possibility that the participant loses the wireless connection to the server still exists.

In most of the research work mentioned in this thesis, when the server loses the participant due to mobility or a connection loss, the server never receives a notification. Once the ad-hoc server dispatches the sub-tasks, it waits for time t_p to complete processing the allocated sub-tasks. If the ad-hoc server does not receive the results from the participant after time t_p, the server waits for additional time t_w, after which it reconfigures for the failed sub-tasks. In either case, longer wait time and reconfiguration both extend the service time by adding additional delay. Therefore, to solve the problem of mobility and abrupt change in the network conditions, this research aims to use SDN with the information provided by the MEC services.

The use case mentioned by ETSI in [66] inspires the work in this thesis to use real-time information for mobility. ETSI describes the use of radio network conditions to detect the need for reallocation in order to maintain service continuity. In addition, the radio and location information used to predict the target MEC host, assist in determining the right time to move the application state without taking additional time to set up the communication path. Motivated by the benefits mentioned in the use case, it appears that RSSI and location information of the participants can help to relieve the composed mobile ad-hoc cloud from sudden service interruptions. To limit the scope of this dissertation, application migration involved during mobility is not considered.

The RSSI provides vital information concerning the signal quality received by the device in an access region. The ad-hoc server uses this information to reconfigure for the failed sub-tasks if the participant loses network connectivity. Thus, this results in reducing the wait time for the reconfiguration. In the case of mobility, the ad-hoc

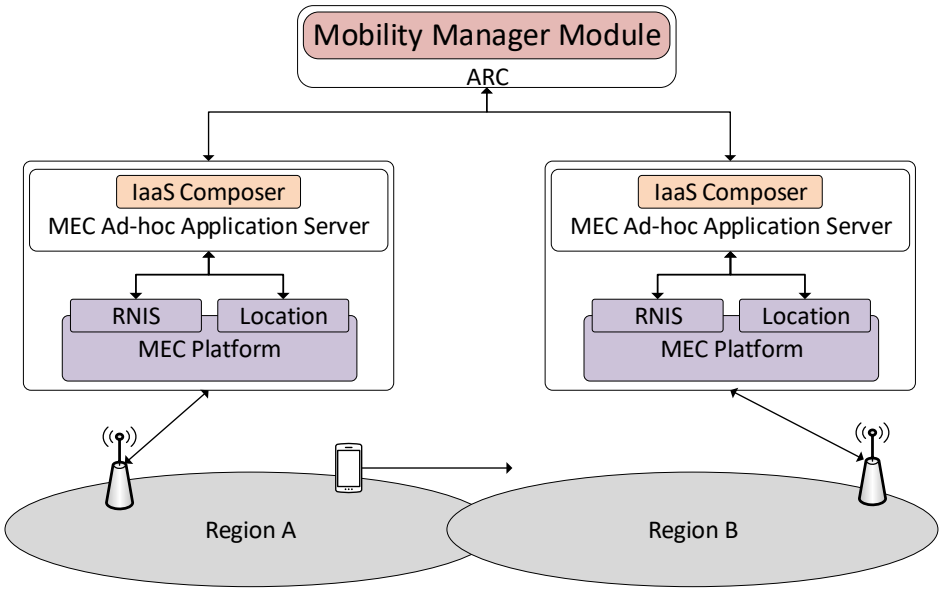


Figure 4.4: Mobility in a MEC-SDN based MAC

application server uses location and RSSI information to detect participant mobility. Instead of reconfiguring when the participant moves to another region, SDN bridges the connection until the offloading results are sent back to the region of composition. Figure 4.4 shows the signaling involved between SDN and MEC to assist mobility of a participant.

The resource monitor located within the ad-hoc application is used to monitor the participants that are part of the composed mobile ad-hoc IaaS. The resource monitor makes use of the services offered by the MEC to collect the signal strength and location information of the participants. If a participant shuts down due to connection loss or powers off, the resource monitor notifies the infrastructure composer to reconfigure. The infrastructure composer allocates the failed sub-tasks to the next best available participant. If no participant is available for the sub-tasks, the infrastructure composer performs resource discovery and repeats the process of MAC formation as presented in Chapter 3. The sole difference is that the request contains only the failed sub-tasks.

In the case of mobility, if the resource monitor observes a change in the signal condition and the location of the participant, it sends a notification message to a mobility manager module located in the ARC. The message includes device information such as the IP and MAC address. The mobility manager uses the information to discover a participant with the matching IP and MAC address. When the user moves to a new region, it detaches from the existing AP and associates to a new AP. To limit the scope, in this is it assumed that the participant does not re-enter to its original AP. During the association, the OpenFlow discovery protocol gets triggered, and requests flow from the SDN controller for an unknown IP and MAC address. The ARC receives the flow request and recognizes the participant, adding flows to connect the participant with the region of composition (where the mobile ad-hoc IaaS is composed initially). Once the participant completes processing the sub-tasks, it attempts to send the result back to the originating server. The flows added by ARC bridges the communication and thus results are sent back to the originating server. When the results are received, the originating server notifies MMM, and a flow mod message is issued to remove the flow, stopping the communication between both entities. Since the mobile device never lost a connection (ignoring the period between disassociation and association) with its original server, the mobile ad-hoc session remains intact. More importantly, the server does not need to wait for reconfiguration. Therefore, this thesis results in making the composed mobile ad-hoc IaaS disruption tolerant.

4.6 Summary

In this chapter, a mobile ad-hoc cloud architecture with SDN was proposed to provide a scalable mobile ad-hoc IaaS. A description is given of a centralized SDN controller within a cluster of MEC servers to allocate resources for an IaaS request. This allows migrating

a request from a resource-limited region to a resource-rich region without the involvement of the IaaS requester. The focus was placed on the request provisioning algorithm that uses region scoring to select the region of interest. This algorithm makes a selection based on the resources available, link latency, and other MEC provided information such as RSSI and location. As well as insights were given on ways to handle the mobility of the participants by taking inspiration from the use case proposed by the ETSI. The mechanism made use of MEC services to prepare the ad-hoc cloud in advance from any service disruption by notifying the ARC. Finally, the proposed work not only allowed for improvement with the request acceptance by migrating to other regions but also improved the load across the cluster.

Chapter 5

Performance Evaluation

5.1 Introduction

In this chapter, the evaluation of the proposed mobile ad-hoc cloud architecture is presented. Also, the performance and usefulness of MAC in the context of MEC are measured. Based on these results, the work demonstrates the use of SDN and MEC to compose a mobile ad-hoc cloud that provides the IaaS requester a stable, scalable, and low access latency infrastructure to offload its application data.

The chapter first highlights the platform and the scenario used to evaluate the proposed architecture. The evaluations are conducted in two parts; the first part investigates the role of MEC in the composition of a mobile ad-hoc cloud. A comparison is made between the performance of the sub-tasks allocation mechanism on a mobile phone and a MEC server. Then a comparison is made of the scheduling efficiency of the various heuristic techniques used for bin packing. The heuristics which are compared for evaluation include FFD, FF, ED, and the proposed E. ED. This is followed by an evaluation of the importance of using context information such as location and RSSI in the proposed heuristic to compose a stable MAC infrastructure.

The second part of the evaluation measures the scalability of the SDN based MAC architecture while addressing the resource availability in MAC. The effect of serving the IaaS request locally with limited resources is investigated, and an explanation is given as to why it is essential to address the resource availability. Subsequently, a comparison is made of the serving time of a request with a specific arrival rate when allocated locally and migrates to a neighboring region using SDN. Finally, the service access time required for a real-time and trigger-based is observed along with an evaluation given on the effect on network latency and request allocation.

5.2 Platform Setup

To evaluate the performance of MEC based MAC, the proposed architecture was developed within Python, which uses Mininet Wi-Fi as a network emulator. Network topology was constructed within the Mininet environment consisting of access points, switches, the wireless station as volunteers and requester. The access point is emulated to operate over 2.4GHz with a transmission range of 200m. In order to emulate this range within the Mininet environment, consideration is given to the Log Distance Propagation model with a system loss of 2 and an exponent of 3.6. Volunteers and requester are emulated using wireless stations. Each station consists of a single to dual-core CPU with each core operating at 1.0 GHz, 512MB to 1GB of memory, and 5GB to 20GB of storage space. Table 5.1 summarizes the simulation parameters used for the evaluation. In addition, the MEC server is simulated using a virtual machine running on Virtual Box with CPU of 4-Core, 8GB of memory, and 13.55GB of storage. MEC and AP communicate over a bridged interface created within the Mininet-WiFi environment. MEC and AP communicate over a bridged interface created within the Mininet-WiFi environment. The entire platform is set up on the physical machine

Type	Parameter	Value
Simulated Mobile Devices (Providers & Requester)	No. of Cores	1- 2
	Core Speed	1.0 GHz
	Memory	512MB to 1GB
	Storage	5GB to 20GB
Simulated MEC Server	No. of Cores	4
	CPU	1.0Ghz
	Memory	8GB
	Storage	135GB
Simulated Access Region	Transmission Range	200m
	Propogation Model	Log Distance
	Exponent	3.6
	Wi-Fi Band	2.4GHz
	Wi-Fi Channel	1

Table 5.1: Simulation Parameters

having an i7 Processor with 8 Cores (4 Physical & 4 Virtual) clocked at 3.70GHz, 13.5GB of DDR3 memory, and 1TB of Hard Disk Drive (HDD). Figure 5.1 shows the high-level view of the platform used for evaluating MEC based MAC.

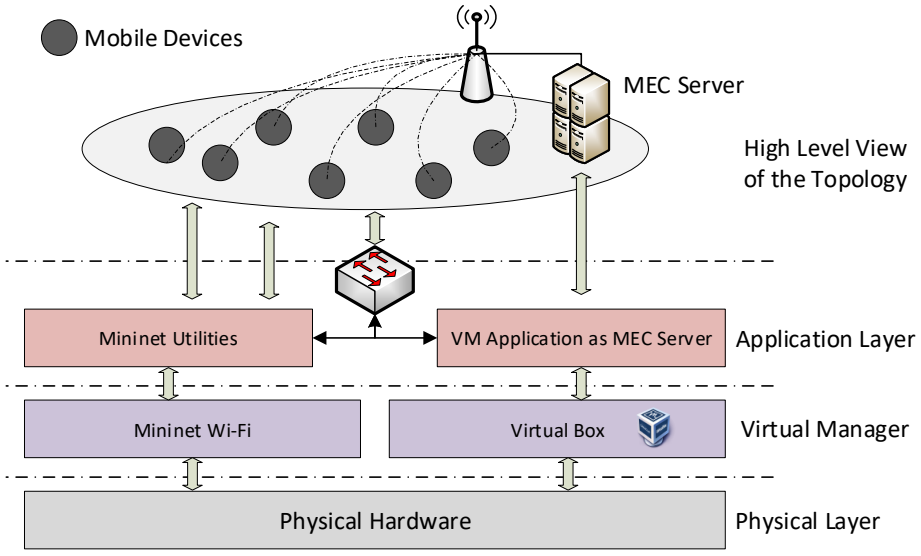


Figure 5.1: Platform Setup for MEC based MAC

Additionally, ETSI defines REST APIs for MEC to provide context information to the application running on the platform. However, in this research, the necessary function of APIs is created to emulate services offered by MEC using Python 2.7. These APIs provide radio and location information. An ad-hoc application server is constructed using Python, which runs as an application on the simulated MEC server. The server composes MAC for an IaaS request using the created API. Also, the server uses rabbitmq to queue multiple IaaS requests, allowing them to create queues with required latency and priority. The IaaS request that originates from the requester is pushed in the queue and remains there until time to live (TTL) of the request expires. While performing resource allocation for an IaaS request, in this research, consideration is given to an allocation to be successful only if all sub-tasks are allocated; otherwise, the request is declared as a failure. In addition, requests are allocated based on their arrival, and optimization in terms of batch processing is left for the future work. As well, to limit the scope of this thesis, each access region is assumed to have an application server running on MEC. The life cycle management involved with the application, such as instantiation, termination and migration is not considered within the scope of this thesis.

5.3 Sub-Tasks Allocation in a MEC based MAC

In this section, the proposed heuristic approach for sub-tasks allocation is evaluated and compared with other heuristic approaches such as FFD, FF, and ED. The evaluation begins when a requester initially sends a request to an ad-hoc application server. The request consists of the sub-tasks and their resource requirements.

Maintaining and managing a MAC is challenging, mainly due to the rapid movement of mobile devices. Consequently, there is a need to involve fewer volunteers

for the composition of a request. In addition, composing MAC with the least number of volunteers helps to reduce the churn in the network and increases the efficiency of the composed MAC [26]. Hence, as with the first case, the packing efficiency of each heuristic is evaluated. The evaluation aims to compare the number of volunteers (VMs) selected by each heuristic, along with the average percentage of sub-tasks and request rejected. Initially, the request with 10 sub-tasks is sent to the ad-hoc application server for allocation, and the number of sub-tasks is increased gradually to a maximum of 100 in the order of 10. Furthermore, each increase in the number of sub-tasks is iterated over 100 times, and for accuracy of the evaluation, the average of those 100 iterations is considered. In addition to a requester, 16 volunteers are initialized, and every volunteer is assumed to participate in the composition of MAC.

Figure 5.2a shows the average percentage of sub-tasks accepted by each heuristic for allocation. It is evident that the proposed heuristic approach using E. ED consistently allocates the highest percentage of sub-tasks when compared to other heuristics. This is possible due to three factors:

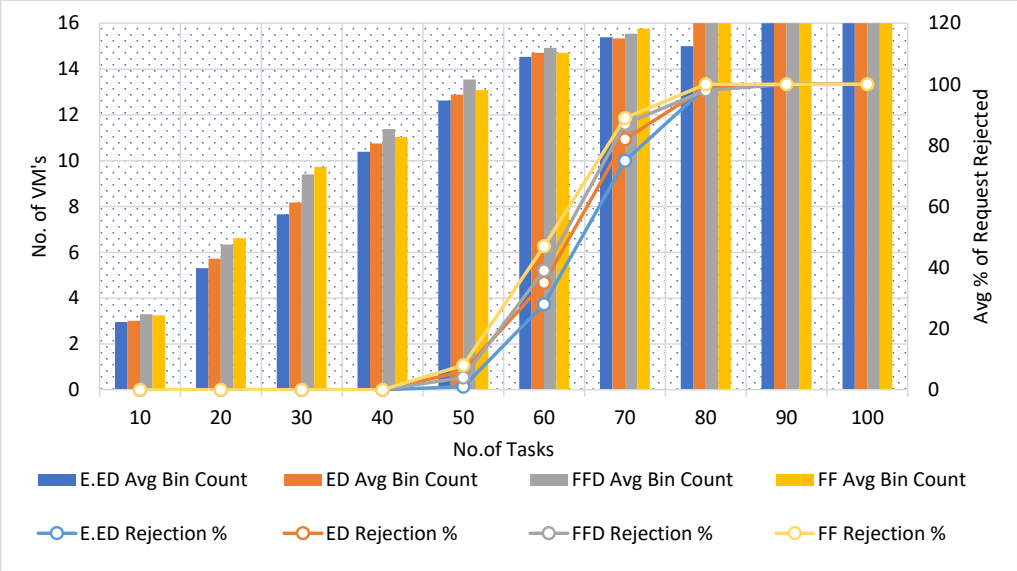
1. E. ED relies on the bin-centric approach for allocation, where a new bin or VM is only selected when no item or sub-task can fit.

2. E. ED uses the residual capacity of the VM to allocate sub-tasks (recall the calculation of Euclidean distance)

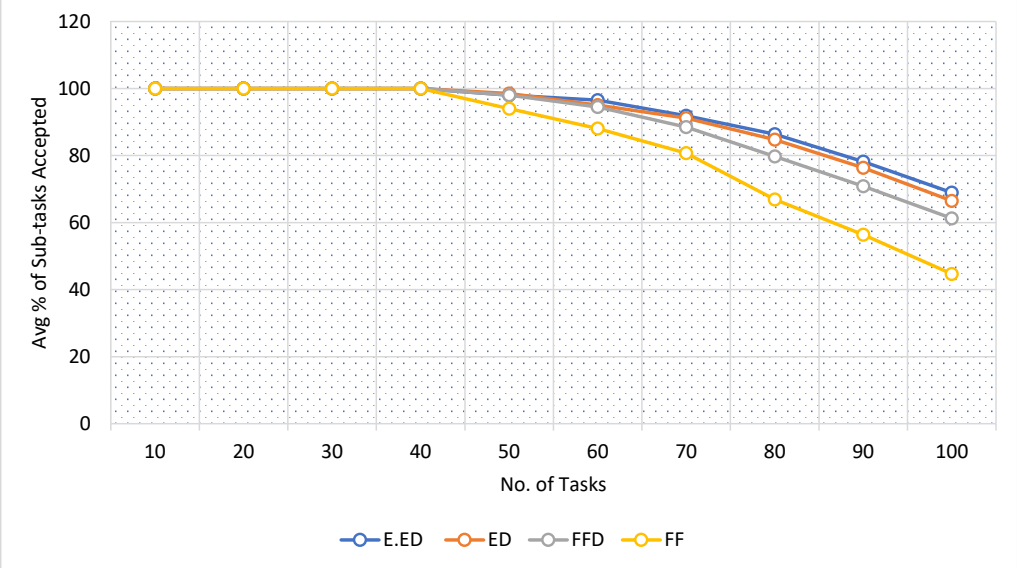
3. Most importantly, E. ED uses normalized scoring to select volunteers for sub-tasks allocation, as it allows for picking volunteers who are resource intensive and has the highest stability when compared with other volunteers

Since E. ED allocates the most number of sub-tasks from a request, the lowest percentage of requests rejected along with the lowest VM count is observed from figure

5.2b. Amongst the other heuristic approaches, FF and FFD provide the highest VM count with the highest rejection, as they allocate sub-tasks based on the order they appear rather than their residual capacity of the VM. Furthermore, it is highly possible that some of the VM resources allocated by FF and FFD are poorly utilized, which increases the number of volunteers involved in the composition.



(a) VM Allocation with Avg. Percent of Request Rejection



(b) Avg. Percent of No. of Sub-Task Accepted

Figure 5.2: Packing Efficiency of Various Heuristic Approaches

Next, an evaluation is conducted to measure the performance of each heuristic when subjected to increase resource availability within an access region. Unlike the previous experiment, the number of sub-tasks in a given request is fixed, and the number of VMs is varied. It initially starts at 10 and is increased by 1 after each iteration until it reaches 16. Each increase in the number of VMs is iterated over 100 times, and the average of 100 iterations is considered for evaluation.

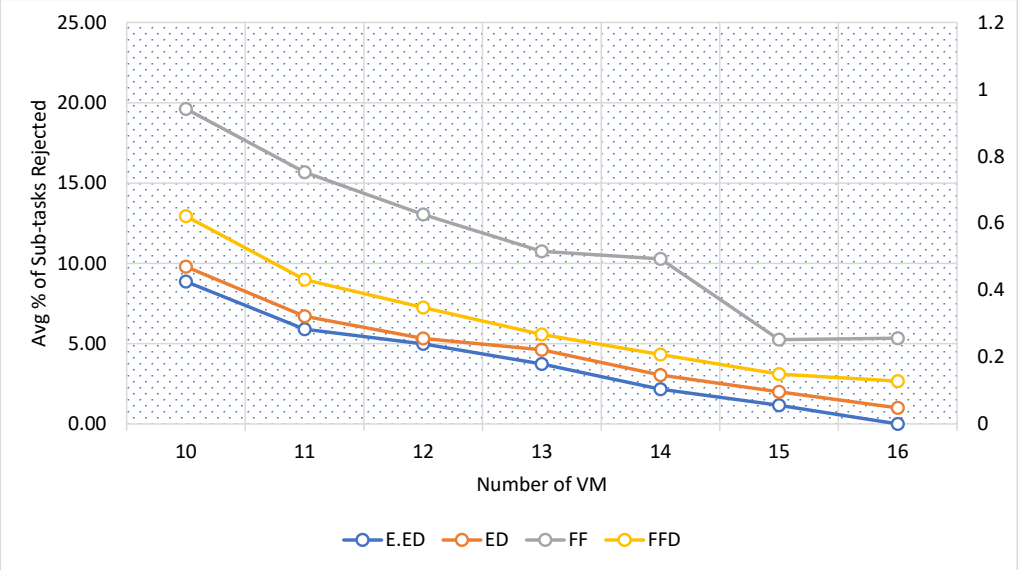


Figure 5.3: Avg. Percent of Sub-Task Rejection with Increase in No. of VMs

Figure 5.3 shows the comparison of each heuristic when the number of VMs increases while the number of sub-tasks is kept fixed. Since the number of VMs increases, the chances of selecting a better volunteer for sub-tasks increases. E. ED identifies the increase in the availability of the VM and uses the normalized score to identify the best volunteer. However, other heuristics fail to identify newer and better volunteers and use the first volunteer available for allocation. As well, other heuristics fail to optimize the resource utilization of the selected volunteers. Therefore, E. ED is observed to drop its rejection faster than any other heuristics.

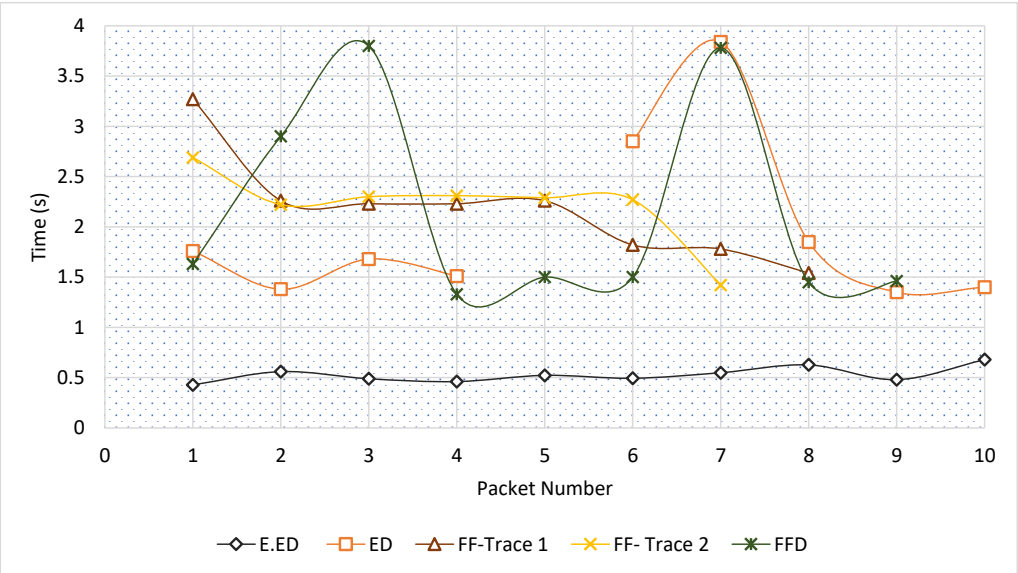


Figure 5.4: Ping Trace of Composed Nodes

Although these results show that the proposed heuristic has better request acceptance while selecting lesser volunteers, they do not provide sufficient information for the stability of the selected volunteers. As a result, in this work, such measurements were taken by introducing churn such that volunteers are allowed to leave and reenter the region. The mobility of the volunteers is based on the Random Way Point model, where each volunteer has its own set of minimum, maximum velocity, and positions. Figure 5.4 shows the ping response of volunteers selected by each heuristic for the composition . Clearly, E. ED results in a stable response because the algorithm makes use of information provided by the MEC in order to compose MAC. The normalized scoring allows for the selection of volunteers who are closer to AP and have the highest RSSI. Other heuristics do not incorporate the intelligence of closeness, and instead, allocate based only on the resources. Thus, there is no guarantee that the device remains or leaves the region. Apart from E. ED, all three heuristics are observed to face some interruption in the session (drop in the ping response), which means the volunteers selected for the composition may have left the region and entered again.

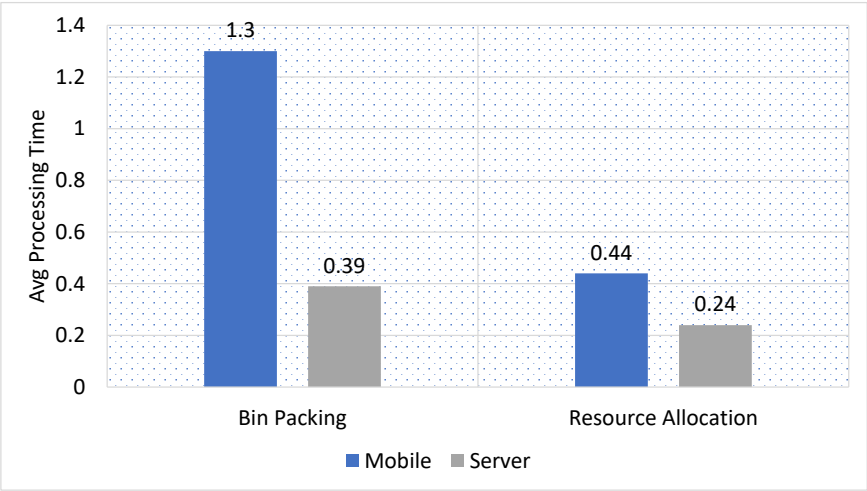


Figure 5.5: Comparison of Processing time on phone and a server

For the evaluation of centrally managed MAC, a measurement is taken of the time required to perform bin packing and sub-tasks scheduling on a mobile device and to compare these results with the simulated MEC server. The mobile phone used in this experiment is a Google Pixel 2Xl and runs Android 8.0 (Oreo). It consists of an Octa-Core processor with a 4 GB of memory and 64GB of internal storage. Figure 5.5 justifies the argument for using a MEC server for sub-tasks allocation. The figure shows that MEC consumes nearly 25% of the time to run bin packing and almost 50% of the time to run resource allocation compared to a mobile phone. This is because the tests involved to perform bin-packing and sub-tasks scheduling are process intensive. A faster CPU results in faster processing of tasks, while using less energy and time. Hence, offloading the composition logic on a MEC server results in faster and more energy efficient processing.

5.4 Request Migration in SDN Assisted MAC

In the previous section, the performance of the heuristic approach using E. ED was evaluated to allocate the resources for an IaaS request. However, when the resources requested by the IaaS requests surpass the resources a region can offer, they are either

rejected or queued until the resources are made available. In either case, the service availability of a mobile ad-hoc IaaS is compromised. In this section, the performance of the proposed SDN based architecture is measured, and evaluation is given on the impact of using SDN to migrate a request to a neighboring region.

The goal of the experiments mentioned in this section is to measure the service response time for an IaaS request under various conditions such as resource availability, mobility, and request migration. The experiment uses multiple requests with an arrival rate following the Poisson's distribution. The arrival rate is first set to 4 requests per minute, which is gradually increased by 4 requests per minute until it reaches 24 requests per minute. Each request consists of 10 sub-tasks, and they are allocated using the proposed heuristic approach.

Impact of Resource Availability on Service Time of IaaS Request

In MAC, volunteers contribute resources based on their interests and availability. However, insufficient resources either queue the requests until they are available or reject them right away. Thus, the impact of resource availability is evaluated by measuring the time required when queued until the requests are allocated and queued until the time to live of the request expires. Figure 5.6 shows the time response for both scenarios. Based on the observation, when the requests are queued, the time taken to serve the requests rises exponentially. The increase in time affects the QoS, and it is possible that the requester may withdraw the request. On the contrary, when the requests are queued, almost 40% of them were rejected. Thus, from the evaluations, it is evident that the performance of offered IaaS service is affected when the supply of resources fails to match the demand of the requests.

To manage the request in a resource-scarce region, an evaluation of the results from the request migration strategy is conducted using SDN. The topology used for the

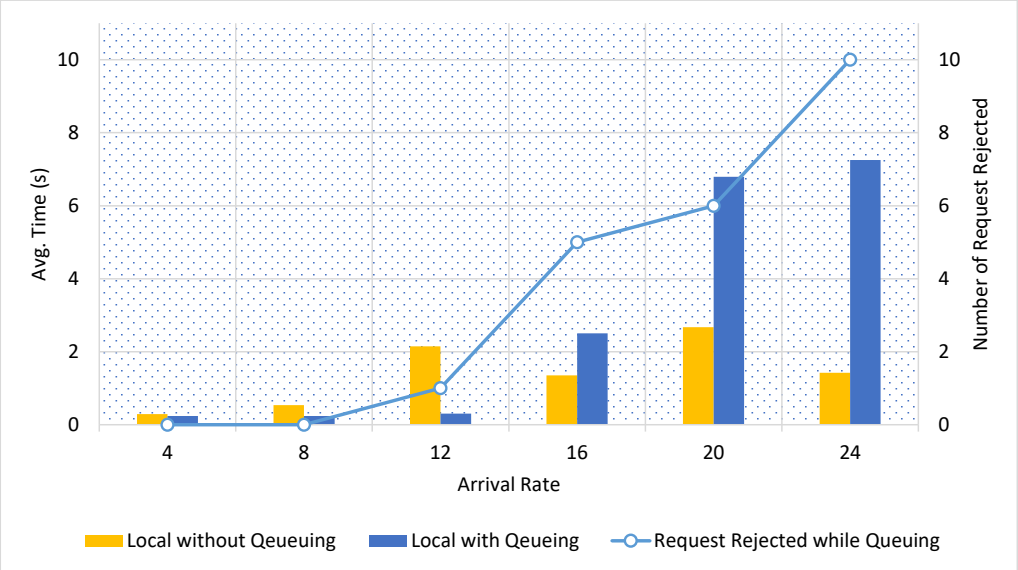


Figure 5.6: Local Broadcast in MEC with Request Rejection

experiment to evaluate request migration is shown in figure 5.7. It consists of 5 access regions, where each region connects to a centralized controller by an OpenFlow switch. Pox is used as a controller. In addition, each region contains 8-10 volunteers, and all volunteers are assumed to participate in the composition of MAC. The simulation parameters used for the access region and volunteers are referred to in table 5.1.

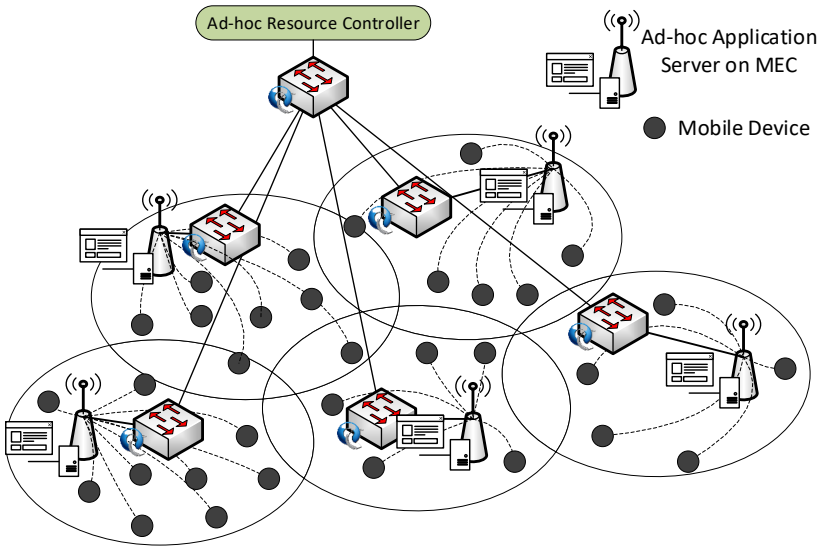


Figure 5.7: Platform Setup

Figure 5.8 evaluates and compares the overall response time for a two-state collection technique, namely real-time and trigger based. These techniques are used by the SDN control application to perform request migration. As observed, the trigger-based technique consumes 30% additional time when compared to a real-time system. This increase in time is due to the additional time required to perform region discovery, collecting state information, and setting up the flow entries. Since real-time technique collects state information every short interval, which in our case is 3 seconds (adjustable), there is no additional time required for region discovery or setting up the flow entries.

Figure 5.8: Comparison of Request Allocation in a SDN system with Non-SDN

In the next experiment, the impact of migrating a request is measured. In addition, a comparison is made of the overall response time for each request when allocated with and without migration. Figure 5.9 compares the response time for SDN based migration and local region allocation with queuing. Although the delay which was introduced due to migration using SDN, adds extra latency of nearly 40% as compared to locally served requests, the benefits of migration improve the service availability across the cluster.

Migrating the requests when resources are not available improves the request acceptance and decreases the overall response time caused as a result of queuing.

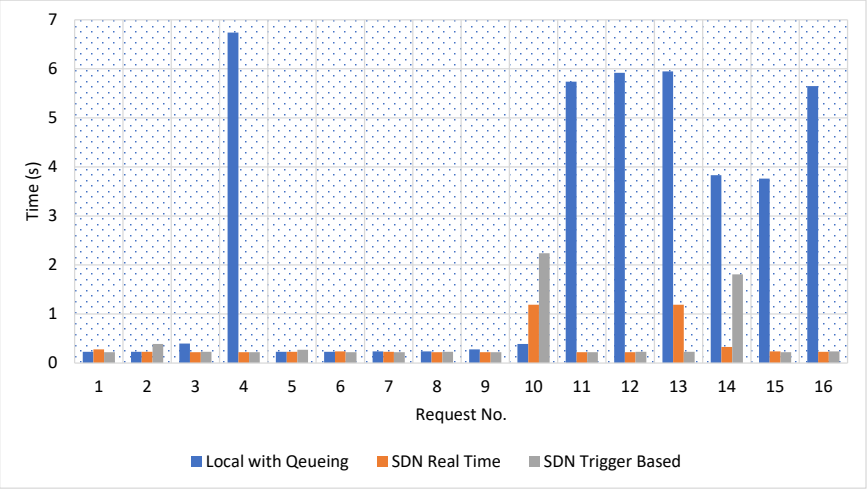
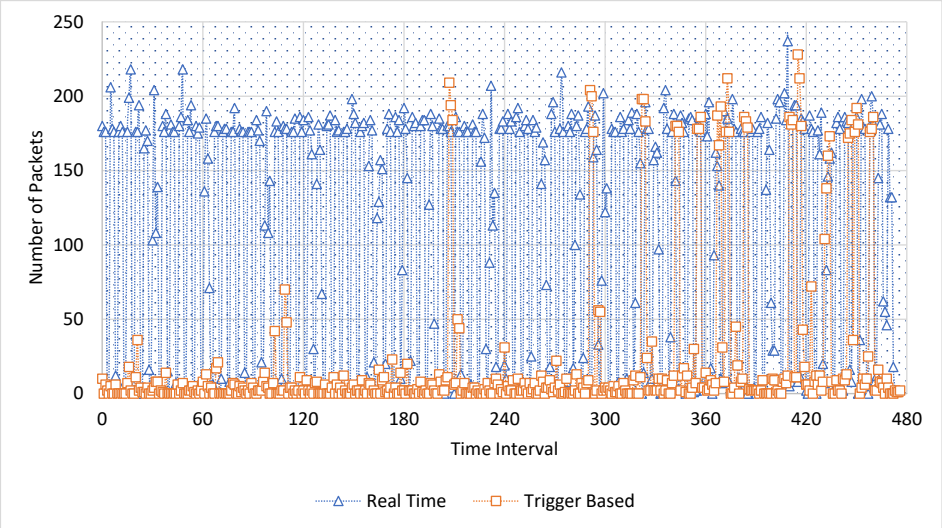
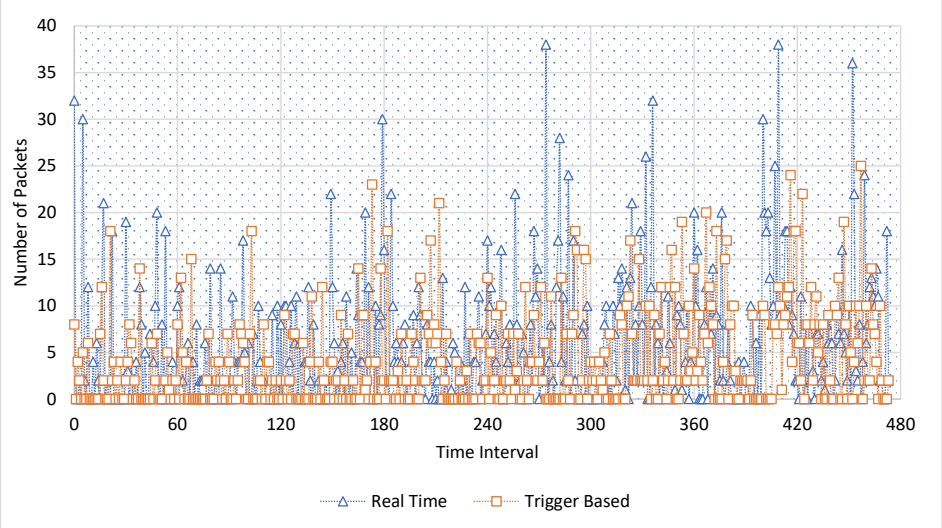


Figure 5.9: Comparison of Service Time per Request in SDN and Non-SDN System

One of the most significant drawbacks of keeping a near real-time system is the number of messages that need to be exchanged to maintain the state. This work measures these messages by monitoring Flow IN and Flow Mod/Out exchanged between the regions and the controller. Figure 5.10a illustrates Flow IN messages exchanged between regions and the controller. In a trigger-based technique, discovery is issued only when the request arrives at the controller. As a result, the number of packets accounted for is negligible when compared to a real-time system. On the contrary, to maintain the active state of the region, the discovery in a real-time system occurs at every short interval, leading to a vast number of message exchanges. Figure 5.10b reveals a similar response, showing the Flow Mod/Out message sent by the controller. These flows indicate the responses for a request migrated to a neighboring region. Based on the above-discussed evaluation, although a real-time system saves time over a trigger-based technique for request migration, it introduces overhead costs due to frequent message exchanges that may congest the network.



(a) Flow IN



(b) Flow Mod/Out

Figure 5.10: OpenFlow message exchange in a Real time & Trigger based system

5.5 Summary

The performance of the proposed MAC architecture was measured and evaluated in an emulated environment called Mininet-WiFi. It was evident that offloading the composition logic from a requesting device saves a considerable amount of time and energy. Also, it was apparent that the MEC server requires 25% to 50% for bin-packing

and sub-tasks scheduling when compared to a mobile device.

Subsequently, a comparison was made of the results from several experiments conducted on various heuristics for multi-dimensional bin packing. It becomes evident that when the resource vectors are highly co-related, there is limited knowledge concerning these vectors while packing results in an inadequate allocation. As well, it was clear that selecting devices for allocation based on normalized score ensures reliability and helps to create a stable mobile ad-hoc cloud IaaS.

In addition, an observation was made of the impact of device selection on the service time of the IaaS request. Also, an evaluation was conducted of the time taken to allocate the resources for an IaaS request when the composed MAC loses connectivity with the participating devices. Based on this evaluation, it was evident that the poor selection of devices that are not stable can lead to a reconfiguration of the composed MAC, thereby increasing the overall response time.

Finally, the measurements were conducted on the impact of migrating a request during limited resource availability and an evaluation of the response time of using SDN for request migration. Based on this evaluation, the overhead introduced during migration was observed negligible in comparison to local region allocation. Furthermore, it became clear that based on the network condition and required QoS, real-time, or trigger-based state collection can be used to perform requests migration.

To sum up, the evaluation conducted in this thesis shows that the proposed MAC architecture is feasible and offers consistent performance in the presence of an MEC and a centralized SDN controller. Although the simulated environment used to simulate the unpredictability of a mobile device in a mobile ad-hoc cloud represents only a part of the real environment, it is apparent that the proposed framework can address any untested behaviors.

Chapter 6

Conclusion and Future Work

In this chapter, a conclusion to this thesis is presented, highlighting a summary of the contribution and outcome of the proposed work. This is followed by a description of plausible future research directions to be adopted in the field of mobile ad-hoc cloud computing.

6.1 Conclusion

The purpose of this dissertation was to determine the challenges faced in the field of mobile ad-hoc cloud formation and to propose a novel solution using state-of-the-art technologies. Initially, the research started with performing a literature review. The review identified some of the research challenges in the field of mobile ad-hoc cloud such as limitations of requester managed MAC, volunteer resource heterogeneity, lack of commitment, and scalability of the mobile cloud IaaS. Additionally, the literature identified the challenges of a remote cloud data center and analyzed the benefits of an edge cloud server.

The focus of the conducted research included developing a MAC formation framework that addressed the challenges involved in the composition of a mobile ad-hoc IaaS. The

main feature of the proposed work is the use of MEC and SDN in the context of MAC that provides a transparent, stable, efficient, and scalable service allocation mechanism. The following section presents a summary of the contributors involved in this dissertation:

- **MEC assisted Infrastructure as a Service for Mobile Ad-hoc Cloud**

  A mobile ad-hoc IaaS composition framework using the MEC server was presented, which offloaded the MAC composition, and management logic to an application located centrally on the MEC server. The centrally managed infrastructure was allowed to perform a region-wide discovery and used MEC services for selecting volunteers to compose resources for an IaaS request. Thus, the framework provided the requester with a transparent request provisioning mechanism. Based on the outcome of the proposed architecture, composing MAC using MEC employed greater stability and remained churn tolerant.

- **Enhanced Euclidean Distance based sub-tasks scheduling for MAC**

  A sub-tasks allocation algorithm, which composed MAC, was introduced to select the volunteer resources for an IaaS request. The algorithm involved a heuristic approach for a multi-dimensional bin packing, which was based on the use of Euclidean distance. The algorithm also considered resource heterogeneity of the volunteers and used the residual capacity of the resources for selecting sub-tasks to achieve improved allocation efficiency. In addition, node scoring was used for selecting volunteers to achieve stability of the composed mobile ad-hoc cloud. In order to evaluate the performance of the proposed heuristic with respect to existing heuristic, a series of experiments were conducted. Selecting fewer, stable volunteers with the lowest rejection was demonstrated throughout the evaluation process.

- **Cluster-based Request Provisioning for a MEC assisted MAC using SDN**

A framework providing a scalable request migration during a lack of volunteer resources was presented. The framework provisioned requests across multiple MEC servers in a cluster with the help of a logically centralized SDN control application. The application determined and selected an MEC region that satisfies the IaaS request requirement based on the region scoring. Also, a two-state collection mechanism was presented, namely real-time and trigger-based, which were used to collect the state information of the region. In addition, this research work highlighted the use of MEC services in conjunction with SDN to provide mobility assistance for participants that are part of the composed ad-hoc cloud. The outcome of the proposed work reduces the rejection due to limited resources and presents the impact of state collection on network and requests migration.

6.2 Future Work

The research work in the field of the mobile ad-hoc cloud has shown several possibilities to extend the proposed work. These include the following:

6.2.1 Extending Ad-hoc Service Availability

MEC offers a wide range of benefits to the applications running on the platform, including context awareness and ultra-low latency communication. However, it is not economically feasible to deploy the MEC server at every access region. Since the proposed work relies on the presence of MEC within the region, some fall back mechanisms need to be proposed to continue the mobile ad-hoc IaaS. The possible solutions to extend the mobile ad-hoc service availability include:

1. Place MEC server near the aggregation node.
2. Use a global orchestrator to provision the request.

In case one, a MEC server can be placed at the aggregation point, instead of at each access region. The placement allows the ad-hoc application to utilize resources from each region, providing a vast pool of resources. In addition, SDN can be used to manage the network when participants from multiple regions serve an IaaS request. SDN can allow routing and traffic management amongst the region using flow-based communication. Furthermore, an optimization algorithm can be proposed to allocate resources based on the requirements of the IaaS request.

In case two, a global orchestrator can be used to provision requests within a cluster of regions. The global orchestrator can be physically placed on any MEC server which is sufficiently resource-full to handle the workload of the entire cluster. All the requests, irrespective of whether the region is equipped with MEC or not, can be made to the global orchestrator. The requests are then optimally dispatched by the global orchestrator based on the availability of the MEC servers within the region.

6.2.2 Extending Mobility Management using MEC and SDN

Mobility management in the mobile cloud is challenging because of the unpredictability of the mobile device. In the current work, when a participant of a composed ad-hoc cloud leaves the access region, the resource monitor takes assistance from the SDN controller to keep the session alive. However, mobility assistance is provided only when the participant leaves the region. In addition, only the participants are considered for mobility management, and requester mobility still needs to be addressed. Thus, to address both problems, several variations in terms of mobility management can be proposed.

One of the approaches is to develop a dynamic mobility management system that uses machine learning to predict the mobile user path before it leaves the region. Machine learning has been used for a decade in the field of mobile networking to assist mobility [82]. MEC can provide meaningful context information to train the prediction model. The predicted region information for a mobile device can be sent to the global controller, which can add the flow in advance.

Another solution includes the use of hierarchical SDN control architecture. A local controller can be used to manage its local region, whereas a global controller is used to manage the entire cluster for inter-regional mobility. When a device attempts to move, the local controller, with the help of MEC services, detects the movement and sends device information to the global controller. The information includes the IP and MAC address of the device. The global controller uses this address as a matching field and adds flow in advance.

The above alternative can be used to address the device mobility in real-time, thus reducing the impact of latency.

6.2.3 Use of Optimization for Region Selection

In the current implementation of SDN assisted MAC, request migration algorithm used region scoring to determine the region of interest for an IaaS request. Nevertheless, the algorithm is based on a heuristic approach, and as with every heuristic, the solution to the problem may not always be an optimal solution. Thus, the selection of the region can be improved by using an optimization-based request migration strategy. The algorithm then guarantees the optimal placement of the request. Furthermore, additional constraints such as channel bandwidth, request load, and past allocation results can be included to improvise the selection further.

References

[1] How Many Computers Are There in the World?, Reference. [Online]. Available: https://www.reference.com/technology/many-computers-world-e2e980daa5e128d0.

[2] Cisco, Visual Networking Index. "Global Mobile Data Traffic Forecast Update, 2015-2020 White Paper." Document ID 958959758 (2016).

[3] Domingues, Patricio, Paulo Marques, and Luis Silva. "Resource usage of windows computer laboratories." In 2005 International Conference on Parallel Processing Workshops (ICPPW'05), pp. 469-476. IEEE, 2005.

[4] Karlson, Amy K., Brian R. Meyers, Andy Jacobs, Paul Johns, and Shaun K. Kane. "Working overtime: Patterns of smartphone and PC usage in the day of an information worker." In International Conference on Pervasive Computing, pp. 398-405. Springer, Berlin, Heidelberg, 2009.

[5] McGilvary, Gary Andrew, Adam Barker, and Malcolm Atkinson. "Ad hoc Cloud Computing: From Concept to Realization." arXiv preprint arXiv:1505.08097 (2015).

[6] Mtibaa, Abderrahmen, Afnan Fahim, Khaled A. Harras, and Mostafa H. Ammar. "Towards resource sharing in mobile device clouds: Power balancing across mobile devices." In ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, pp. 51-56. ACM, 2013.

[7] Yaqoob, Ibrar, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, and Sghaier Guizani. "Mobile ad hoc cloud: A survey." Wireless Communications and Mobile Computing 16, no. 16 (2016): 2572-2589.

[8] Mengistu, TESSEMA M., and Dunren Che. "Survey and Taxonomy of Volunteer Computing." ACM Journal of Computing Surveys (2019): 1-35.

[9] Cisco, Fixed and Mobile IP Network Traffic Forecast VNI. [Online] Available: www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni

[10] Satyanarayanan, Mahadev. Fundamental challenges in mobile computing. No. CMU-CS-96-111. Carnegie-Mellon University Pittsburgh School of Computer Science, 1996.

[11] M. P. Papazoglou, and W. Heuvel, "Blueprinting the cloud," IEEE Internet Computing, vol. 15, no. 6, pp. 74-79, 2011.

[12] Kosta, Sokol, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." In 2012 Proceedings IEEE Infocom, pp. 945-953. IEEE, 2012.

[13] Othman, Mazliza, Sajjad Ahmad Madani, and Samee Ullah Khan. "A survey of mobile cloud computing application models." IEEE Communications Surveys & Tutorials 16, no. 1 (2014): 393-413.

[14] European Telecommunications Standards Institute, "Multi-Access Edge Computing (MEC); Framework and Reference Architecture", ETSI GS MEC 003 V2.1.1, 2019. [Online]. Available: https://www.etsi.org.

[15] Mao, Yuyi, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. "A survey on mobile edge computing: The communication perspective." IEEE Communications Surveys & Tutorials 19, no. 4 (2017): 2322-2358.

[16] McGilvary, Gary Andrew., Adam Barker, and Malcolm Atkinson. "Ad hoc cloud computing." In 2015 IEEE 8th International Conference on Cloud Computing, pp. 1063-1068. IEEE, 2015.

[17] Lacuesta, Raquel, Jaime Lloret, Sandra Sendra, and Lourdes Pealver. "Spontaneous ad hoc mobile cloud computing network." The Scientific World Journal 2014 (2014).

[18] Yaqoob, Ibrar, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, and Muhammad Imran. "Heterogeneity-aware task allocation in mobile ad hoc cloud." IEEE Access 5 (2017): 1779-1795.

[19] Malhotra, Amarjit, Sanjay Kumar Dhurandher, and Bijendra Kumar. "Resource allocation in multi-hop mobile ad hoc cloud." In 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1-6. IEEE, 2014.

[20] Zaghdoudi, Bilel, Hella Kaffel-Ben Ayed, and Imen Riabi. "Ad hoc cloud as a service: a protocol for setting up an ad hoc cloud over MANETs." Procedia Computer Science 56 (2015): 573-579.

[21] Mawji, Afzal. A Framework for Peer-to-Peer Computing in Mobile Ad Hoc Networks. Queen's University, 2010.

[22] Balasubramanian, Venkatraman, and Ahmed Karmouch. "An infrastructure as a Service for Mobile Ad-hoc Cloud." In 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1-7. IEEE, 2017.

[23] Park, Eunjeong, and Heonshik Shin. "Reconfigurable service composition and categorization for power-aware mobile computing." IEEE transactions on Parallel and distributed systems 19, no. 11 (2008): 1553-1564.

[24] Shila, Devu Manikantan, Wenlong Shen, Yu Cheng, Xiaohua Tian, and Xuemin Sherman Shen. "AMCloud: Toward a secure autonomic mobile ad hoc cloud computing system." IEEE Wireless Communications 24, no. 2 (2017): 74-81.

[25] Miluzzo, Emiliano, Ramn Cceres, and Yih-Farn Chen. "Vision: mClouds-computing on clouds of mobile devices." In Proceedings of the third ACM workshop on Mobile cloud computing and services, pp. 9-14. ACM, 2012.

[26] Drolia, Utsav, Nathan Mickulicz, Rajeev Gandhi, and Priya Narasimhan. "Krowd: A key-value store for crowded venues." In Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture, pp. 20-25. ACM, 2015.

[27] Olaniyan, Richard, Olamilekan Fadahunsi, Muthucumaru Maheswaran, and Mohamed Faten Zhani. "Opportunistic Edge Computing: Concepts, opportunities and research challenges." Future Generation Computer Systems 89 (2018): 633-645.

[28] Kreutz, Diego, Fernando MV Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103, no. 1 (2015): 14-76.

[29] Baktir, Ahmet Cihat, Atay Ozgovde, and Cem Ersoy. "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions." IEEE Communications Surveys & Tutorials 19, no. 4 (2017): 2359-2391.

[30] Kim, Hyojoon, and Nick Feamster. "Improving network management with software defined networking." IEEE Communications Magazine 51, no. 2 (2013): 114-119.

[31] Balasubramanian, Venkatraman, and Ahmed Karmouch. "Managing the mobile Ad-hoc cloud ecosystem using software defined networking principles." In 2017 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1-6. IEEE, 2017.

[32] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011): 20-23.

[33] Tsai, Wei-Tek, Xin Sun, and Janaka Balasooriya. "Service-oriented cloud computing architecture." In 2010 seventh international conference on information technology: new generations, pp. 684-689. IEEE, 2010.

[34] Zhou, Bowen, and Rajkumar Buyya. "Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions." ACM Computing Surveys (CSUR) 51, no. 1 (2018): 13.

[35] Dinh, Hoang T., Chonho Lee, Dusit Niyato, and Ping Wang. "A survey of mobile cloud computing: architecture, applications, and approaches." Wireless communications and mobile computing 13, no. 18 (2013): 1587-1611.

[36] Cuervo, Eduardo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. "MAUI: making smartphones last longer with code offload." In Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 49-62. ACM, 2010.

[37] Chun, Byung-Gon, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. "Clonecloud: elastic execution between mobile device and cloud." In Proceedings of the sixth conference on Computer systems, pp. 301-314. ACM, 2011.

[38] Ha, Kiryong, Padmanabhan Pillai, Grace Lewis, Soumya Simanta, Sarah Clinch, Nigel Davies, and Mahadev Satyanarayanan. "The impact of mobile multimedia applications on data center consolidation." In 2013 IEEE international conference on cloud engineering (IC2E), pp. 166-176. IEEE, 2013.

[39] Ahmed, Arif, and Ejaz Ahmed. "A Survey on Mobile Edge Computing.", Accepted by 10th IEEE International Conference of Intelligent Systems and Control (ISCO 2016).

[40] Dolui, Koustabh, and Soumya Kanti Datta. "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing." In 2017 Global Internet of Things Summit (GIoTS), pp. 1-6. IEEE, 2017.

[41] Sharing Economy [Online]. Available: https://www.investopedia.com/terms/s/sharing-economy.asp

[42] Kirby, Graham, Alan Dearle, Angus Macdonald, and Alvaro Fernandes. "An approach to ad hoc cloud computing." arXiv preprint arXiv:1002.4738 (2010).

[43] Cunsolo, Vincenzo D., Salvatore Distefano, Antonio Puliafito, and Marco Scarpa. "Volunteer computing and desktop cloud: The cloud@ home paradigm." In 2009 eighth IEEE international symposium on network computing and applications, pp. 134-139. IEEE, 2009.

[44] Figueiredo, Renato J., Peter A. Dinda, and Jos AB Fortes. "A case for grid computing on virtual machines." In 23rd International Conference on Distributed Computing Systems, 2003. Proceedings., pp. 550-559. IEEE, 2003.

[45] Anderson, David P., Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. "SETI@ home: an experiment in public-resource computing." Communications of the ACM 45, no. 11 (2002): 56-61.

[46] Babaoglu, Ozalp, and Moreno Marzolla. "Peer-to-Peer Cloud Computing." (2011): 1-9.

[47] Asadi, Arash, and Vincenzo Mancuso. "WiFi Direct and LTE D2D in action." In 2013 IFIP Wireless Days (WD), pp. 1-8. IEEE, 2013.

[48] Tehrani, Mohsen Nader, Murat Uysal, and Halim Yanikomeroglu. "Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions." IEEE Communications Magazine 52, no. 5 (2014): 86-92.

[49] Mawji, Afzal, and Hossam S. Hassanein. "Bootstrapping p2p overlays in manets." In IEEE GLOBECOM 2008-2008 IEEE global telecommunications conference, pp. 1-5. IEEE, 2008.

[50] Milojicic, Dejan S., Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. "Peer-to-peer computing." (2002).

[51] Lu, Zongqing, Jing Zhao, Yibo Wu, and Guohong Cao. "Task allocation for mobile cloud computing in heterogeneous wireless networks." In 2015 24th International Conference on Computer Communication and Networks (ICCCN), pp. 1-9. IEEE, 2015.

[52] Satyanarayanan, Mahadev, Victor Bahl, Ramn Caceres, and Nigel Davies. "The case for vm-based cloudlets in mobile computing." IEEE pervasive Computing (2009).

[53] Jararweh, Yaser, Loai Tawalbeh, Fadi Ababneh, and Fahd Dosari. "Resource efficient mobile computing using cloudlet infrastructure." In Mobile Ad-hoc and

Sensor Networks (MSN), 2013 IEEE Ninth International Conference on, pp. 373-377. IEEE, 2013.

[54] Satyanarayanan, Mahadev, Rolf Schuster, Maria Ebling, Gerhard Fettweis, Hannu Flinck, Kaustubh Joshi, and Krishan Sabnani. "An open ecosystem for mobile-cloud convergence." IEEE Communications Magazine 53, no. 3 (2015): 63-70.

[55] Ceselli, Alberto, Marco Premoli, and Stefano Secci. "Cloudlet network design optimization." In 2015 IFIP Networking Conference (IFIP Networking), pp. 1-9. IEEE, 2015.

[56] Esehaye, Debessay, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. "Impact of cloudlets on interactive mobile cloud applications." In Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International, pp. 123-132. IEEE, 2012.

[57] Beck, Michael Till, Martin Werner, Sebastian Feld, and S. Schimper. "Mobile edge computing: A taxonomy." In Proc. of the Sixth International Conference on Advances in Future Internet, pp. 48-55. Citeseer, 2014.

[58] Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog computing and its role in the internet of things." In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16. ACM, 2012.

[59] Vaquero, Luis M., and Luis Rodero-Merino. "Finding your way in the fog: Towards a comprehensive definition of fog computing." ACM SIGCOMM Computer Communication Review 44, no. 5 (2014): 27-32.

[60] Yi, Shanhe, Cheng Li, and Qun Li. "A survey of fog computing: concepts, applications and issues." In Proceedings of the 2015 workshop on mobile big data, pp. 37-42. ACM, 2015.

[61] Patel, Milan, Brian Naughton, Caroline Chan, Nurit Sprecher, Sadayuki Abeta, and Adrian Neal. "Mobile-edge computing introductory technical white paper." White Paper, Mobile-edge Computing (MEC) industry initiative (2014): 1089-7801.

[62] European Telecommunications Standards Institute, "Multi-access Edge Computing (MEC); Terminology", ETSI GS MEC 001 V2.1.1, 2019. [Online]. Available: https://www.etsi.org.

[63] European Telecommunications Standards Institute, "Mobile Edge Computing (MEC); Service Scenarios", ETSI GS MEC-IEG 004 V1.1.1, 2015. [Online]. Available: https://www.etsi.org.

[64] Beck, Michael Till, Sebastian Feld, Andreas Fichtner, Claudia Linnhoff-Popien, and Thomas Schimper. "ME-VoLTE: Network functions for energy-efficient video transcoding at the mobile edge." In 2015 18th International Conference on Intelligence in Next Generation Networks, pp. 38-44. IEEE, 2015.

[65] European Telecommunications Standards Institute, "Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements", ETSI GS MEC 002 V2.1.1, 2018. [Online]. Available: https://www.etsi.org.

[66] European Telecommunications Standards Institute, "Mobile Edge Computing (MEC); End to End Mobility Aspects", ETSI GR MEC 018 V1.1.1, 2017. [Online]. Available: https://www.etsi.org.

[67] European Telecommunications Standards Institute, "Mobile Edge Computing (MEC); UE Identity API", ETSI GS MEC 014 V1.1.1, 2018. [Online]. Available: https://www.etsi.org.

[68] McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.

[69] Gude, Natasha, Teemu Koponen, Justin Pettit, Ben Pfaff, Martn Casado, Nick McKeown, and Scott Shenker. "NOX: towards an operating system for networks." ACM SIGCOMM Computer Communication Review 38, no. 3 (2008): 105-110.

[70] "POX Homepage," [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[71] "OpenDaylight Homepage," [Online]. Available: https://www.opendaylight.org/.

[72] Pfaff, B., B. Heller, D. Talayco, D. Erickson, G. Gibb, G. Appenzeller, J. Tourrilhes et al. OpenFlow switch specification version 1.0. 0 (wire protocol 0x01). Technical report, Stanford University, 2009.

[73] Goransson, Paul, Chuck Black, and Timothy Culver. Software defined networks: a comprehensive approach. Morgan Kaufmann, 2016.

[74] Lee, Sangmin, Rina Panigrahy, Vijayan Prabhakaran, Venugopalan Ramasubramanian, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. "Validating heuristics for virtual machines consolidation." Microsoft Research, MSR-TR-2011-9 (2011): 1-14.

[75] Panigrahy, Rina, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. "Heuristics for vector bin packing." research. microsoft. com (2011).

[76] Leinberger, William, George Karypis, and Vipin Kumar. "Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints." In Proceedings of the 1999 International Conference on Parallel Processing, pp. 404-412. IEEE, 1999.

[77] Ku, Ian, You Lu, Mario Gerla, Rafael Lopes Gomes, Francesco Ongaro, and Eduardo Cerqueira. "Towards software-defined VANET: Architecture and services." In Med-Hoc-Net, pp. 103-110. 2014.

[78] Gai, Keke, Meikang Qiu, Hui Zhao, Lixin Tao, and Ziliang Zong. "Dynamic energyaware cloudlet-based mobile cloud computing model for green computing." Journal of Network and Computer Applications 59 (2016): 46-54.

[79] Satyanarayanan, Mahadev. "The emergence of edge computing." Computer 50, no. 1 (2017): 30-39.

[80] Tang, Ling, Shibo He, and Qianmu Li. "Double-sided bidding mechanism for resource sharing in mobile cloud." IEEE Transactions on Vehicular Technology 66, no. 2 (2016): 1798-1809.

[81] Yousafzai, Abdullah, Victor Chang, Abdullah Gani, and Rafidah Md Noor. "Directory-based incentive management services for ad-hoc mobile clouds." International Journal of Information Management 36, no. 6 (2016): 900-906.

[82] Klaine, Paulo Valente, Muhammad Ali Imran, Oluwakayode Onireti, and Richard Demo Souza. "A survey of machine learning techniques applied to self-organizing cellular networks." IEEE Communications Surveys & Tutorials 19, no. 4 (2017): 2392-2431.