An SDN-assisted Framework for Mobile Ad-hoc Clouds

Venkatraman Balasubramanian

A Thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfilment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

In Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical and Computer Science (EECS)
University of Ottawa
Ottawa, Canada

January 2017

Table of Contents

List of Figures

List of Tables

Abstract

Over a period of time, it has been studied that a mobile "edge-cloud" formed by hand-held devices could be a productive resource entity for providing a service in the mobile cloud landscape. The ease of access to a pool of devices is much more arbitrary and based purely on the needs of the user. This pool can act as a provider of an infrastructure for various services that can be processed with volunteer node participation, where the node in the vicinity is itself a service provider. This representation of cloud formation to engender a constellation of devices in turn providing a service is the basis for the concept of Mobile Ad-hoc Cloud Computing. In this thesis, an architecture is designed for providing an Infrastructure-as-a-Service in Mobile Ad-hoc Clouds. The performance evaluation reveals the gain in execution time while offloading to the mobile ad-hoc cloud.

Further, this novel architecture enables discovering a dedicated pool of volunteer devices for computation. An optimized task scheduling algorithm is proposed that provides a coordinated resource allocation. However, failure to maintain the service between heterogeneous networks shows the inability of the present day networks to adapt to frequent changes in a network. Thus, owing to the heavy dependence on the centralized mobile network, the service related issues in a mobile ad-hoc cloud needs to be addressed. As a result, using the principles of Software Defined Networking (SDN), a disruption tolerant Mobile Ad-hoc Cloud framework is proposed. To evaluate this framework a comprehensive case study is provided in this work that shows a round trip time improvement using an SDN controller.

Acknowledgement

I would like to express my gratitude to my supervisor, Professor Ahmed Karmouch, for providing me patient guidance, support and motivation through my research to complete this thesis. His valuable technical help and encouragement were a source of inspiration throughout these two years and in the countless years to come.

I am grateful to my lab mates Heli Amarasinghe, Wijaya Ekanayake and Faisal Zaman. These guys have most definitely helped me with valuable suggestions to strengthen this research. The lab environment would not have been half as lively as it was without the contribution of these three.

I landed in Ottawa not knowing where to go, or how to live alone. Without the help from my brothers, room-mates and friends namely, Vinod Narayanan, Mohanraj Nandagopal, Akshaykanth Muppidi, Philip Benjamin George, Prassana Rousseau and Keyur Modi, it would not have been an endearing ride. The God in all the aforementioned people helped me live a salubrious life.

Last but far from least, I would like to dedicate this thesis to my father Dr.V.Balasubramanian and my mother Dr.N.Seethalexmy who have stood by me and for me throughout my career. I would be remiss if I did not mention the support from my brother in-law Siddharth K.S.Rangan and my sister Pooja Balasubramanian, to whom I owe everything I've learned about life. She'd be at peace with my one year old niece Ashlesha hearing the completion of this dissertation.

Acronyms

AP	Access Point	MAC	Mobile Ad-hoc Cloud
API	Application-Programming Interface	MANETs	Mobile Ad-hoc Networks
ARP	Address Resolution Protocol	ONF	Open Network Foundation
CSP	Cloud Service Provider	OVS	Open Virtual-Switch
CPT	Composed Participant Topology	OCS	Opportunistic Ad-hoc Cloudlet Service
DARC	Distributed Ad-hoc Resource Composition	OCR	Optical Character Recognition
DHCP	Dynamic Host Configuration Protocol	PaaS	Platform-as-a-Service
DHT	Distributed Hash Table	P2P	Peer-to-peer
DPID	Data-path ID	QoE	Quality of Experience
EPC	Evolved Packet Core	QoS	Quality of Service
FTP	File Transfer Protocol	RAN	Radio Access Network
GTP	GPRS Tunneling Protocol	RTT	Round Trip Time
GPRS	General Packet Radio Service	SDN	Software Defined Network
IaaS	Infrastructure-as-a-Service	TCP	Transmission Control Protocol
IP	Internet Protocol	UDP	User Datagram Protocol
ISP	Internet Service Provider	VPN	Virtual Private Network
ICMP	Internet Control Message Protocol	VARP	Volunteer Ad-hoc resource pool
KVM	Kernel-based Virtual Machine	VM	Virtual Machine
LAN	Local Area Network	WLAN	Wireless Local Area Network
LTE	Long-Term Evolution		

1 Introduction

Mobile Cloud Computing has gained significant attention over the years. [2] defines this paradigm as :

"*Mobile Cloud Computing at its simplest refers to an infrastructure where both the data storage and the 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 mobile computing to not just smartphone users but a much broader range of mobile subscribers*"

Striding by the principles of Cloud Computing, Mobile Cloud Computing rose to prominence. In [1] the important aspects of Cloud Computing with the Service Oriented Architectural specifications is shown. It defines the three layers of as-a-service paradigm that are also inherited by the Mobile Cloud Computing paradigm. These are:

(i) Software-as-a-service (SaaS) – This layer offers limited power to the consumer in-terms of customizing ability. The SaaS model fundamentally allows hosting web services and computer software application for users. A simple resource poor mobile device can potentially access SaaS via a web browser or any other vendor specific web based application. However, a SaaS user does not get the privilege to configure the underlying infrastructure such as a server, an operating system to name a few.

(ii) Platform-as-a-service (PaaS) – This layer provides services to host application, tools for development and other libraries to the cloud infrastructure. Subscribing to this service means the user receives an API from the provider to access the platform application or software

development. However, even with this service layer, users do not have the privilege to configure or modify the underlying infrastructure. Nevertheless, a PaaS user can configure the applications developed or the ones that are run on the platform

(iii) Infrastructure-as-a-service (IaaS) – This is the most integral layer that provides the processing, storage , networks and other computing resources that are provided according to the service characteristics of the cloud. IaaS users have the freedom to configure and migrate workload between resource provisioning entities. For instance, Amazon AWS provides virtual server instances and storage via an API which allows users to move workload to Virtual Machines within a datacentre. Additionally, users can choose the operating systems and what type of VMs they need for the task etc. Here, users get a portion of cloud control privilege by which they can customize the operating systems, processing and storage on demand. The key enabler is the virtualization technique. Different providers make use of hypervisors for provisioning. For instance, Amazon makes use of the Xen Hypervisor. Sun's Sun grid makes use of virtualization for Job Management System (Sun Grid).

The root layer is the IaaS that provides other two upper layer services. Although, large datacenters located in remote locations hosting these cloud services contain powerful computing and storage resources, service providers impose heavy burden in-terms of infrastructure cost and frequent time related issues. This has in many ways led to a new vein of Cloud Computing called Ad-hoc Cloud Computing. This paradigm breathes the same philosophy as Cloud Computing in terms of the service provided and additionally, a more closer-to-user solution that alleviates the infrastructure cost and time related issues. More recently, owing to the growth in the Mobile Device Technology, a subset of Ad-hoc Cloud Computing is being studied closely, this subset is known as Mobile Ad-hoc Cloud Computing.

1.1 Motivation

In today's accelerated growth of mobile device technology, there is a need to establish a firm ground for these devices to stay committed to application computation and completion. From [3] it can be inferred that the rate of mobile device usage has increased over the decades. Additionally, the growth of mobile application such as real-time gaming, face recognition, and music OCR also gives a similar picture. With an overall growth rate of 29.8% each year noticed in [4], by the end of 2017 there would be more than 4.4 billion mobile application users. Out of these, there are around one in four mobile applications that are downloaded once and never used again. These applications are primarily discarded due to the growing application needs that have gone beyond the mobile device capabilities.

Thus, even if the device is able to process its OS, the remaining resources are finding it difficult to process these intensive applications and resort to costly remote cloud services. Remote cloud services rely on large consolidated datacenters that provide compute and storage. However, these data centers represent a point of centralization that has serious shortcomings. It can end-up as a single point of failure in times of disasters or data center's geographical location is often- times out of limits for the customers making use of it. Moreover, public clouds have frequent issues such as infrastructure cost and high Round trip time (RTT) while considering time sensitive classes of applications. In [5][6][7] authors discuss many services and applications which ascertain using remote clouds as infeasible. These services range from experimental cloud services, shared services, and dispersed data services. These are services where applications are solely dependent on the time and place in which the applications need to be executed. Such place-bound activities are best addressed at the user level. This class of cloud computing that deals with the formation and deployment at the user’s level is known as Mobile Ad-hoc Cloud

(MAC). An MAC is a pool of device with high computational capabilities and is closer to the user. Therefore, it enables a low cost and a low latency environment that will form a potentially significant computational resource.

This low-cost computational environment is deployed over a network where all nodes cooperatively maintain the network. Hence, wireless local area networks (WLANs) and Mobile Ad-hoc networks (MANETs) are predominantly considered. For instance, a P2P network enabling a computational environment for mobile nodes could be referred to as Mobile P2P Cloud. In MANETs [8] users can form a wireless network at any place and at any time without any centralized administration. MANETS consist of arbitrarily placed communication devices. It is of great use when temporary connectivity is required. There are many features that differentiate cloud models in mobile ad hoc networks with public clouds, however, the most integral out of these are (i) Both consumer and provider nodes are mobile (ii) Service composition would change dynamically depending on the available resources (nodes).

For once imagine a music concert where a crowd has gathered for watching an artist perform. It's a common sight in such venues when artists are trying to enthrall the crowd by making the attendees present therein sing for them or interact with them through a mobile wave. Various interactive applications that are used at concerts not only play back pre-recorded notes but also convert audio to text or a music OCR for notes or lyrics viewing on the spot at the gig site. Some artists have also begun to call the use of smartphone application in concerts as the new applause [9].These applications are not only compute intensive but are also bound by place and time. What if devices present therein are able to provide compute and storage facilities to one another? A pool of idle intra-device resources put together would more likely provide a low-cost service in lesser time than a remote cloud. Thus, every device has the potential to act as a service

provider in mobile ad-hoc clouds. This has been a motivating factor in harnessing the idle device resources that are not completely capable of performing intensive computation but display the ability to collaboratively perform a compute intensive application execution. Just as a Cloud Service Provider (CSP) is an entity that is responsible for providing an Infrastructure as a Service (IaaS) to the consumers, in a mobile environment each device behaves as an IaaS provider. Noticeably, the services provided by such a cloud has constraints related to the network location of the consumer. That is incessant disruptions may eventually lead to an unsuccessful cloud computation event. Owing to the heavy dependence on the centralized mobile network, the service related issues in an ad-hoc cloud needs to be addressed. Therefore, a technology that hides the network heterogeneity as well as the network state is best utilized for addressing the issues in a mobile ad-hoc cloud. This is technology is known as Software Defined Networking.

Recently, software defined network (SDN) has become a widely accepted solution for network management. SDN with Mobile Cloud has provided an efficient mechanism for offloading. A fundamental misconception that should not be overlooked is that the Software defined networking in cellular networks cannot be considered as an extension of the Software defined networking in the internet. Although, conceptually it is similar the factoring of the control plane and data plane is however dissimilar. Hence, this is known as Software defined Mobile Networking (SDMN). As this takes into consideration the cellular networks, it primarily deals with Evolved Packet System architecture. Motivated by the benefits of SDMN, a disruption tolerant mobile ad-hoc cloud is proposed. The dynamic network configurability given by SDN is used to engage in a seamless mobile ad-hoc cloud computation to support the mobile IaaS cloud providers. Further, we introduce an algorithm for task scheduling to minimize the devices participating in an ad-hoc cloud service provisioning. In doing so, a devoted mobile ad-hoc cloud

service framework with SDN assistance is delineated that puts forth a significant contribution towards the Mobile Ad-hoc Cloud Computing research.

1.2 Thesis Objectives

The goal of this thesis is to develop a Disruption- tolerant Mobile Ad-hoc Cloud framework. It includes seamless mobility management with SDN and cloud service provider device minimization with optimal device selected from a number of devices available in the vicinity for task scheduling. Mobile Ad-hoc Cloud differs from traditional cloud computing services in many ways. It includes handling mobility related disruptions, volunteer node participation and capability of the mobile resource providers. The primary challenge with the existing Ad-hoc Cloud Computing framework was the inefficiency due to rapid movements. Therefore, our approach was to address key limitations of current strategies and propose a new IaaS mobile cloud service architecture. We outline our primary objectives as follows:

Spontaneity: Formation of an "on-the fly" cloud needs to be impulsive. It is essential because a spontaneous computing environment overcomes the time related issues of remote clouds. Minimizing end to end delay between mobile user devices ensures faster composition. Therefore, using the benefits of a key-based P2P composition algorithm, our objective is to provide a spontaneous IaaS cloud service for local mobile users.

Storage and Computation: Compute intensive applications not only seek a faster computing ecosystem but also require storage capabilities for the results to be stored and retrieved based on the consumers requests. Therefore, the objective of the proposed ad-hoc cloud service is to optimally allocate and manage resources during cloud composition and movement.

Management and Co-ordination: There is a need to ensure continuity of service. That is, once resources are composed and available for computation, there would be no disruption until the device moves out of the location. Additionally, the choice of the device selected for composition needs to be addressed as too many devices in a composition would lead to excessive synchronisation messages being sent between the devices. This leads to more stress in the system. Moreover, it becomes the responsibility of the manager who looks over the network to provide a seamless service between nodes. Therefore, along with deciding who participates in the composition, our objective is to use the strength of SDN in mobile ad-hoc clouds to enable a seamless resource provisioning. In the case of mobility, we identified the possibilities of handover within and across access networks and even across different technologies such as 3G/4G to Wi-Fi. Therefore, an SDN- based mobile ad-hoc cloud framework is defined that empowers the composed resources by extending the possibilities of user mobility behavior and maintaining a seamless connectivity to the ad-hoc cloud service.

1.3 Thesis Contribution

The objectives delineated in section 1.2, are essential to be met while providing a seamless mobile ad-hoc cloud service in wireless local area networks. According to the literature, mobile cloud approaches with acute impulsiveness provides the ability to access ad-hoc cloud services with minimum delay. Recalling all the aforementioned requirements, this thesis makes the following contributions.

- An **Infrastructure as a Service paradigm is designed for a Mobile Ad-hoc Cloud**: A framework designed to constitute a usable resource entity utilising the heterogeneous devices available in the vicinity while tapping onto their individual virtual resources. A

novel system architecture is built for spontaneous resource discovery and management of resources in scenarios where the existing infrastructure is inconvenient to be used like a crowded environment in a concert venue, or a University to a name a few.

- **Optimized Task Scheduling in a Mobile Ad-hoc Cloud:** In order to co-ordinate and manage between resources, considering the idle intra-device resources, a composition metric called Composition Score is introduced that puts devices in the vicinity in a usable form assisted by the Composition Algorithm. Further, task scheduling to individual devices by optimally choosing the best possible device is done using an optimal TSA algorithm for Mobile Ad-hoc Cloud. As a result, the inter-device synchronization and other networks overheads incurred are relatively reduced.
- **An SDN assisted Mobile Ad-hoc Cloud:** We exploit the prowess of SDN in mobile wireless networks for creating a disruption tolerant Mobile Ad-hoc Cloud. We recognize that, a seamless service for many network-based applications can be maintained if there is consistent user mobility across networks. Hence, our focus is to provide seamless handover with the SDN-enabled wireless networks and make the movement possible at the time of computation. Further, we re-model the data-plane with SDN wireless nodes that have local controllers which behave as a fallback mechanism in times of controller failure. An in-depth analysis of how the orthogonality of the data plane and the control plane can be beneficial is conducted. By virtue of this architectural independence between the two planes, Software Defined Networking principles provides a wide array of advantages like network programmability, flexibility ,virtualization to name few. A rigorous performance evaluation is carried out to determine the controller performance under stress. In this way, a logically centralized control mechanism to conveniently

handling user application requests for IaaS and orchestrate ad-hoc cloud resources with user mobility among different access networks is realized.

1.3.1 List of Publications

Based on the results of the research, two conference papers and one magazine paper have been produced:

Refereed Conference Publications:

- Venkatraman Balasubramanian, Ahmed Karmouch , "An Infrastructure as a Service for Mobile Ad-hoc Cloud", in Proceedings of The 7th IEEE Annual Computing and Communication Workshop and Conference, (IEEE-CCWC) , January 2017 **[Best Paper Award]**
- Venkatraman Balasubramanian, Ahmed Karmouch , " Optimization based on device selection in a Mobile Ad-hoc Cloud based on Composition Score", 2nd International Conference on "Communication System, Computing and IT Applications 2017" (IEEE-CSCITA 2017) , April 2017 [Accepted]
- Venkatraman Balasubramanian, Ahmed Karmouch , " Managing the Mobile Ad-hoc Cloud Ecosystem using Software Defined Networking Principles", in *The International Symposium on Networks, Computers and Communications (IEEE-ISNCC),* Marrakech , Morroco, May 2017 [Accepted]

Magazine Publication:

- Venkatraman Balasubramanian, Ahmed Karmouch, "SMAC : An SDN assisted Mobile Ad-hoc Cloud", IEEE Communications Magazine [Submitted]

1.4 Thesis Organization

The remainder of this thesis is structured as below:

In chapter 2, the background information is presented which shows the concept of Mobile Phone Virtualization that acts as an enabler for Mobile Ad-hoc Cloud Computing. Additionally, we show some state of the art mobile ad-hoc clouds solutions and a very closely related body of work called Mobile Device Clouds that utilizes the Mobile Ad-hoc Cloud Methodology. The architectures and challenges of present network environment is discussed which bring forth the novel principles of SDN that impact the mobile ad-hoc cloud. We analyze the ad-hoc cloud computing model in general and focus on the benefits of this paradigm. Then we introduce the concept of volunteer computing together with the existing ad-hoc cloud computing approaches to show the shared benefits of the two forms of computing that affects our chosen direction. Finally, we analyze the challenges in legacy networks that affects the mobile ad-hoc cloud computing paradigm and discuss the novel solution called software defined networks and software defined wireless networks along with its widely used implementation with the OpenFlow protocol.

In chapter 3, we identify the significance of mobile clouds placed closer to user by minimizing end to end latency. Then we introduce the IaaS based Mobile Ad-hoc Cloud Architecture. We further discuss the mobile ad-hoc cloud service framework with a job scheduling design that

facilitates the resource allocation and device optimization. Further, we look at some compelling drawbacks that renders such environment useless in absence of a technology that customizes the data-plane interactions.

In chapter 4, we describe a proposed solution to the limitations affecting the network in the mobile ad-hoc cloud service framework. We elaborate the functionalities of Software Defined Wireless Network that directly impacts the flexibility and adaptability of the network environment based on the data plane behaviour customization by the wireless network OpenFlow controllers. Here we analyze re-modelling of the wireless nodes and provide a complete hierarchical SDN controller framework that strategically enables a disruption tolerant mobile ad-hoc cloud.

In chapter 5, firstly, we describe scenarios of user mobility in a one-hop network with the architecture mentioned in chapter 3. Then we evaluate the performance of the proposed SDN assisted mobile ad-hoc cloud service framework using an emulated testbed to simulate the above scenarios. We discuss the outcomes of the evaluated test scenarios with both the environments based on the simulation results and compare the results with the seamless mobile ad-hoc cloud service with user mobility in a multi-hop network. Finally, we compare results in all the test-cases to predict the scalability of the proposed hierarchical framework.

In chapter 6, we conclude the thesis and gauge the future potential of our IaaS based mobile ad-hoc cloud service architecture.

2 Background and Concepts

2.1 Introduction

In this chapter, we describe the concept of Ad-hoc Cloud Computing and present how Cloud Computing served as an inspiration to build this concept. Although, at first the two paradigms might not seem to gel with one another, the elemental functionalities bring together the essence of computing in both paradigms.

2.1.1 Overview

Before we delve into the main idea, a background of the literature is provided that is requisite to understand the complete logical flow of the thesis framework. We first present a general study on the Mobile Phone Virtualization concept and it's suitability with the Mobile Ad-hoc Cloud paradigm. We then discuss, the Ad-hoc Cloud Computing framework and allied lines of research such as P2P Overlay Service Composition and Volunteer Computing concepts followed by a discussion on the intersection of these research topics that posits a familiar link to Ad-hoc Cloud Computing. Followed by how MANETs make use of a structured routing process with the help of Distributed Hash table (DHT) and how this helped in developing our design. Further, we describe several techniques mentioned in literature for Task Scheduling in Mobile Ad-hoc cloud services that play a crucial role in such models to minimize device synchronization problems. Lastly, we describe a number of bottlenecks with the traditional mobile ad-hoc network architecture (MANET) that act as the compelling factor for introducing the concept of software-defined networks to mitigate those bottlenecks.

2.2 Mobile Phone Virtualization

An increased use of smartphones in the past few years has showcased the omnipotence of the hand-held device taking over the traditional desktop and laptop computers. To this end, an important working tool has been developed in [10] for enabling a light weight resource abstraction inside the hand-held device. Fig 2.1 shows how the model is adapted in this research work. This phenomenon of abstracting the device hardware while maintaining the device performance is known as Mobile Phone Virtualization. The major benefits of using this technology is its light-weight nature and typical isolation characteristics as seen in traditional virtualization mechanisms. Other strategies that follow a somewhat similar approach is the para-virtualization schemes followed in [11, 12,13]. However, as [10] allows effective mechanisms to assist applications in directly using the hardware features from the VMs, we use the

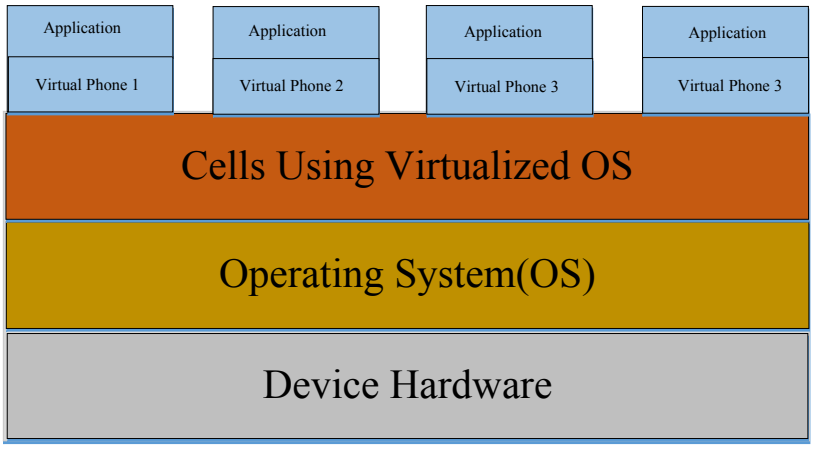


Figure 2.1 Mobile Phone Virtualization Usage Model

abstraction layer provided by the Cells [10] architecture.

2.2.1 Overview

The capacity to run multiple virtual phone instances on a single OS is essential to exercise multiple functionalities with resource resources. As the isolation between the Virtual Phones (VPs) is maintained, the processes running on one VP cannot be tampered with by the processes running on the other VP. Cells takes into consideration the limited screen size and allows view of a single VP screen at a time. Moreover, for the usage purpose a VP is created on a PC and downloaded into an android phone via USB. This provides a notable quality from a provider's perspective as the VPs in one providers phone cannot be reconfigured by any other. Therefore, the privilege rights differs based on the user and the device. For the case of usage in a Mobile Ad-hoc Cloud Computing paradigm, a shared access privilege can be pre-configured by a provider. One major benefit of Cells is its ability to prevent privilege escalation attacks for one VP. This ensures that the entire device is not compromised. In this way, Mobile Phone Virtualization concept becomes a major enabler of the Mobile Ad-hoc Cloud Computing framework.

2.3 Ad-hoc Cloud Computing

Moore's law has been proved right over the decades with the advancement of device processing and computation power. In [6] authors have suggested a method for harvesting resources from sporadically available devices. Gary McGilvary et al. believe that in a company environment when there are underutilized personal computers there is a need to harvest the resources that are left idle. In due course, the authors propose the ad-hoc cloud formation of these existing resources. The nature of tapping such resources poses similar elements to grid and volunteer computing. The model that the authors propose in [6] considers (i) Volunteer resources (ii) Lack

of trust (iii) Ensures continuity (iv) Low Interference and (v) Diverse Workloads. Hence, from the consideration of these factors for the ad-hoc cloud primarily certifies the set of non-exclusive, sporadically available hosts that are not like a dedicated cloud but are more unpredictable in nature. This model neither assumes a level of trust, nor does it provide a fixed relationship between end-users. The level of service continuity is maintained till the point of job failure at a particular node. Also, the interference level between executing processes is minimized. One important feature of this model is the diverse set of resource components which are harnessed, that is there is no clear description of what resources are/are not to be used, different memory and CPU processors would be offered for the job execution. Inspired by the traditional Cloud Computing Concept, i.e. where the resources such as processing, memory, applications and platforms are commoditized and delivered to cloud users, the Ad-hoc Cloud Computing paradigm adheres to the same.

According to the National Institute of Standards and Technology (NIST) definition [1] Cloud Computing (CC) is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service or service provider interaction". Therefore, the primal goal of ad-hoc cloud computing paradigm is to confirm its methodology with the traditional definition. For instance, the three well known pay-as-you-go aspects of Cloud Computing, as mentioned above, SaaS, PaaS, IaaS can be typically realized with this paradigm [6]. More specifically, having an Infrastructure in place provides the freedom of usability to the consumer, thereby allowing consumer to build a platform for developing software for a service.

As observed in [7] there is a separate class of services which could possibly make use of this type of computing mechanism. These are known as (i) Experimental Cloud Services, (ii)Dispersed Data intensive services , and (iii) Shared services. An example that elucidates these classes further could be individually seen as follows:

- Experimental Cloud Services: When we consider an environment that is used a test-bed system for testing bugs or an experimental deployment model of a large-test-cloud which could be used for sustained access like a research laboratory, it could fall under the experimental cloud services category.
- Dispersed Data intensive: When we consider, huge chunks of dispersed data, and moving data to the resource rich centralized cloud becomes expensive and inefficient. It is ideal that we move the data to a nearby closer to user environment for sufficient computational capability; this is termed as Dispersed Data intensive services.
- Shared services: When we consider, public service provisioning, as in users or organization wishing to freely share their applications, whether it is commercial or non-commercial, and where service providers do not want to pay the cost of running the services, its termed as Shared services

The aforementioned list, describes a multitude of services that do not require a high performing, expensive cloud platform but require a quick and a robust platform for providing necessary computational environment. Therefore, such services need to be hosted in a more dispersed, end-user volunteered resources i.e. resources donated by end-users. These resources are instrumental in producing this new class of cloud known as Ad-hoc cloud. Therefore, these benefits persuaded us to investigate this paradigm further.

A class of distributed computing that is built on the essence of offering compute and storage resources is known as Volunteer Computing. This concept was primarily used in the SETI@Home project [15] that used volunteers' compute and storage resources from their devices to help in extra-terrestrial research. Although, this form of computing is simple and popular it comes with additional challenges that are common in cluster computing and Grid Computing. For instance, minimizing overheads, maintaining scalability and churn are some of the major challenges this type of computing faces. Moreover, certain inevitable downfalls such as the provider donating resources may face a decrease in performance when the volunteer tasks are being executed. However, as with any form of computing there are trade-offs based on which a certain form is chosen. Most definitely, volunteer computing enables a substantially low cost and when combined with the ad-hoc cloud computing model it engenders a beneficial strategy for the aforementioned classes of applications.

In ad-hoc mobile clouds, mobile devices are connected over P2P protocol mechanisms like in [16] or via D2D communication strategies such as Bluetooth or Wi-Fi such as in [17]. The ad hoc cloud is formed using intermittently available unreliable infrastructure to form a distributed computing environment. However, as the use of an insufficiently available infrastructure such as a congested Wi-Fi link or a partially available access network can augment the overall ad-hoc network, there are many implementations that are based on ad-hoc mechanisms which make use of the residual available Wi-Fi infrastructure from a locality. For instance, a Key-Value store implementation called "Krowd" proposed in [3] follows this line thought. Using D2D, a source device can offload application data to closer mobile client devices. Therefore, in contrast to centralized cloud approaches, ad-hoc cloud computing paradigm contains key characteristics such as volunteering resources, lack of trust, lower interference and diverse workload [6]. In the

ad hoc cloud architecture proposed by [6], contains a volunteering set of ad hoc devices who participate for the processing of tasks submitted by cloud users over an ad hoc cloud server. They use reliable ad hoc devices to form small clouds for different applications within an organization to host virtual ad hoc cloud.

According to the authors of [18], ad-hoc clouds in general use existing IT infrastructure of organizations. The application data is then offloaded to the underutilized processing and storage devices within an organization. Following the same strategy, authors in [19] present a workload distribution approach among ad hoc devices within an organization. The fixed and reliable devices are grouped together to form virtual cloudlets. The key difference in this approach is that, the mobile users submit their tasks over an ad hoc cloud server interface which in turn passes decomposed subtasks in to the virtual cloudlets within the organization.

In ad-hoc clouds, tasks are partitioned to smaller subtasks and offloaded to other ad hoc devices at runtime. Then the subtasks are distributed among ad-hoc clients over D2D connections in a timely fashion based on a scheduling mechanism. [17] illustrates a meticulous job structure that incorporates the intermittent connectivity between devices. As it follows a collaborative approach towards task allocation and resource monitoring, that considers a PNP block. This block is composed of a pre-process program, n-parallel tasks and a post-process program. Another system model that follows an opportunistic approach is [20] where the job structure is decided based on the resource entity that is chosen for computation whether it's a remote public cloud, a mobile device cloud or a cloudlet.

Another body of work that replicates service provisioning similar to the aforementioned context is known as Service Overlay Networks. Research in this aspect has been a strong influence in the

development of device clouds or P2P clouds in general. As noticed in [21] a dynamic construction of an overlay in mobile networks involves movement of a node in and out of a network adding to the instability in the network topology. However, this model utilizes the effectiveness of cluster formation in well-known regions to deploy the overlay. [22] presents a P2P overlay discovery and composition that introduces a Service Oriented middleware architecture for QoS control and configuration of energy efficient composition graphs. The main agenda was to show resource depletion in mobile host. [16, 23] on the other hand provides a clear description of how P2P overlays on a MANET form a desirable combination for many services. It shows how multicasting can be achieved by a node outside of a P2P overlay that wishes to join a specific overlay.

Continuing the discussion on allied fields of ad-hoc cloud, a strong resemblance can be observed in Cluster computing and Grid computing. Authors in [25] show a scalable mobile ad-hoc cloud that involves a match-maker node that receives a request message each time a cluster goes void or the initiator does not find a suitable node in the divided cluster. The primary benefit of dividing the ad-hoc network into clusters is to reduce the overheads at the time of communication. Some clusters are fixed while others are dynamic. One clear assumption while considering clusters in a dynamic environment is that those nodes at a 1-hop distance become cluster members of the chosen cluster head. Node with the highest stability and life (in terms of battery, signal strength etc.) becomes the cluster head. Such is the similarity with Grid Computing where resources that are geographically distributed are combined to create a highly resourceful compute entity. Although, approaches for grid computing are localized within institutional or organizational boundaries, a novel usage is seen in [24] where authors emulate

x86 instructions on an iPhone mobile device by creating a Virtual Machine to demonstrate the feasibility of using mobile devices in a grid.

A typical approach of deployment of an intermittently available cloud can be seen in [26] where cloud is built on top of a Disruption tolerant network application layer. This behaves like an "overlay cloud" that elastically expands and contracts depending on the dynamicity of the network topology that may contain fixed or ad-hoc infrastructure. It is seen that ad-hoc devices can use the Cirrus Cloud if the user is equipped with a smart phone which can pick up traffic from a sensor network and can store data on the cloud. It also provides an advantage for the nomadic nodes to participate in providing cloud service with their own resources, thereby resulting in a service distributed over a dynamic topology. It could be a fixed or a mobile topology.

2.4 Mobile Ad-hoc Networks (MANETs) and its Synergy with P2P Overlay

With the augmentation of mobile devices, the research community is gradually shifting toward Mobile Ad-hoc Networks (MANETs). Implementing cloud computing per se in a MANET requires overcoming some of the technical road-blocks it posits. An integral part of communication in a MANET is the routing approach that is followed on the network layer (Layer 3). Further, in a Mobile Ad-hoc Network there is a need to address routing at the time of mobility while considering dynamic environments. One noticeable feature in these frameworks is the usage of a Distributed Hash Table mechanism. As the node location is refactored from it's identity DHT provides a scalable self-organizing system. Owing to the constant node mobility, limited transmission range and the requirement for spontaneity different DHT algorithms are prescribed for routing. In recent times, [3] has built a system framework that follows a Kademlia

[26] like approach with minor modifications for wireless networks. In P2P overlays protocols like Chord [27], Pastry [28], and Tapestry [29] have prevalent usages of DHT strategies for routing. For once, [27] does not consider the physical topology at the time of deployment which means the nodes which are assigned particular IDs (based on a pre-defined identifier space, *m*, 0 to $2^{(m-1)}$) might be a single hop away in the overlay but would be multiple hops away in the physical network. This affects routing in the sense that the Finger Table (the routing table in a chordal ring) maintained by a chordal node is arranged in an increasing order of IDs, therefore, proximity is not given any importance. [28] takes into consideration the physical proximity of the nodes in the overlay network and assigns ID in a random fashion from a circular logical space of 0 to $(2^{(128)}-1)$ .For routing it maintains three tables a main routing table, a neighbourhood set and a leaf set, based on the geographic distance the universal identifier is maintained to ensure guaranteed delivery of messages. [29] is a tree based P2P approach that employs a certain degree of randomness for routing. It makes use of a content's key and the node's logical ID to route a message. Further it follows a suffix based look up, such that the routing table levels have pointers that match the suffix of a particular level. These protocols are application layer protocols and rely on an underlying network layer protocols. However, there are protocols like Kademlia that can be directly implemented on the network layer. Moreover, owing to its independence from the node locality and node identification based on the randomly generated 160 bit node ID, this protocol has been used in popular systems like BitTorrent. The keys are the hashed content of the data (based on SHA-1 hashing scheme).The information is stored in *k* bucket, where each node has a list of *k*-buckets for nodes between a distance of 2^i and $2^{(i+1)}$ $(0 \leq i \leq 160)$ So, the <key, value> pair are stored in the nodes with IDs closest to the key. Extensive research on DHT based approaches has been carried out in [8] that shows the

suitability of various approaches and the trade-offs to be considered for making a decision on based on purpose and strategy.

2.4.1 Concepts

Logically, a distributed cloud infrastructure can be pictured as large dispersed individual computers connected over a network. These cloud frameworks have characteristics similar to P2P systems, some of which are observed in the previous sections. As we are dealing with such a system in a crowd-sourced mobile environment it is defined as a Mobile P2P cloud or Mobile Ad-hoc cloud. A Mobile Ad-hoc cloud harvests resources that are available in the vicinity. As mentioned previously, the mobile devices are responsible for playing the role of the cloud IaaS providers. The role of requesting a cloud service from the providers is of the consumer. Thus, the major actors in any cloud computing paradigm are the cloud providers and consumers.

Being the principal stakeholder, the consumer requires cloud resources to fulfill the application needs in terms of resources. Likewise in mobile ad-hoc clouds, due to the resource limitations in a mobile, compute-intensive applications require external assistance for execution. For instance, the tasks which cannot be processed locally and require a resource rich environment would need to be offloaded to the cloud providers. Thus, the consumer makes the IaaS request. Considering scenarios such as those with a high density of users (cloud IaaS providers) these requests submitted by the consumers are exposed to the IaaS providers. Once the IaaS request is received, the IaaS provisioning entity will discover cloud IaaS providers. These are resources whose ownership is with individuals that are available in the vicinity. The

major challenge is to turn this diverse collection of resources into a usable cloud infrastructure. A typical algorithm that would enable this usable resource entity can be called a composition algorithm. This composition algorithm performs the key functionalities pertaining to the services provided to the consumer. The consumers of IaaS have access to virtual resources available in the devices as explained below. Let's assume this IaaS is deployed over a wireless network formed with the assistance of an Access point (AP). Many volunteers (who wish to offer their VMs) may exist in the vicinity that provides a unique service to the consumer who requests the infrastructure. The physical resources are known as volunteer resources because of their ability to offer their VMs. For example, one user who is at the concert will have many friends or like-minded people who would be ready to offer their resources. Out of the many friends, the IaaS would select only the nearest devices. These friends (volunteers) will provide their device VM/VMs. For testing, one request is either dedicated to a single VM or could be a part of many VMs. In this way, the salient features of cloud computing i.e. on-demand self-service and service orchestration is realized with mobile ad-hoc cloud computing.

2.4.2 Mobile Phone Virtualization and Mobile Ad-hoc Cloud

As demonstrated in Fig 2.2. Our architecture relies on the Mobile-Phone-Virtualization concept. The potential of a virtualized ARM is mentioned in [10][11][13]. As observed in [10] the hardware virtualization approach for smartphones (Virtual Phones) has isolation and light-weight characteristics similar to the Virtual machines. In order to maintain generality, we refer to virtual phones (VPs) as virtual machines (VMs) of the devices. One device might have multiple background VMs/VPs each offered to different customers. The light-weight VMs from devices are harnessed to deploy IaaS. These VMs have adequate storage and compute capabilities. The process of obtaining a VM and the dependencies it should satisfy is

illustrated as follows: The first part of the IaaS algorithm is a composition that is responsible for discovery, selection and P2P formation. On discovery (1), the volunteer submits the details (2) of the VMs, node id, and the IP addresses to the IaaS. After this, routing and management is done with the assistance of the information (key, value) in storage. Concurrently, considering the dependencies the metafile is created. It uses the sub-task information (2a) and the resource information obtained from (2). The meta-file is retrieved (3a). It is then hashed and the keys are used for taking the meta-file (3b) to the correct device. It also has the location of the source file which is used by the VMs for downloading and processing the job (3c). After (1,2) , the virtual machines are composed(3) followed by (3b), the jobs are obtained with a get request from the user device (3c), processed, executed (4) and the results are sent back(5) after which the resources are released.

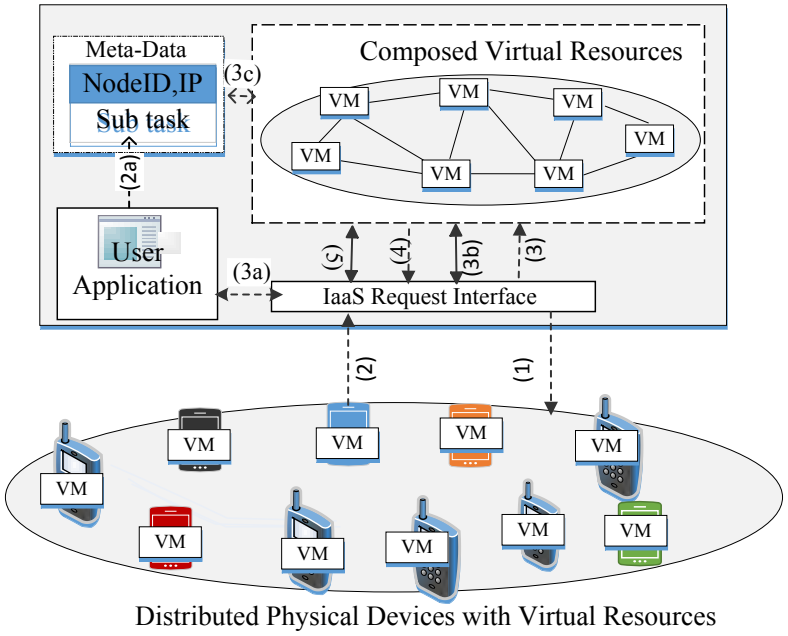


Fig 2.2. Mobile Ad-hoc Cloud Overview

In this way, by making the resources available to the cloud consumer, the ability to use the mobile ad-hoc cloud to execute any applications lies with the consumer. This conforms to the cloud IaaS paradigm as defined in chapter 1.

From the above concepts it is clear without a doubt that, a MANET is deployed over arbitrarily available communicational devices. In cases where the existing infrastructure is either disrupted or inconvenient to use a MANET provides a solution to form a network "on-the-fly". Likewise, in [3] the advantages of such networks are visible in scenarios where a crowd is gathered in a stadium or a concert venue. Hence, it can be posited that in circumstances where temporary connectivity is required MANETs are of great use. Similar in ideology, [25] works with the assumption that the match-maker nodes are statically present. It illustrates a Multi-hop Mobile Ad-hoc Cloud Computing framework that is formed using available local resources. It makes use of an intermediate match-maker node for service. However, it considers the producer nodes as the under-utilized resource rich nodes that help the consumer nodes via the match-maker nodes.

To elaborate further on its functionality, when a consumer request is made to the match-maker node, the match-maker has to keep a list of all the resource rich nodes at the location. As this match-maker can be many hops away the request made has to travel hop-by-hop to the match-maker. In this work authors divide the entire network into fixed sectors and give it a sectorID based on which the nodes presence and ability to form a mobile ad-hoc network is judged. This work does not show the consequence of maintenance of the computing environment at the time node movement but makes an implied reference to the inability of the nodes to maintain performance level at the time of mobility.

There are other closely related works that resemble in part the inflexibility of the network which impedes service in ad-hoc clouds. To address such inconveniences, there is a need to configure the data-plane elements to behave as desired. Therefore, a technology that hides the network heterogeneity as well as the network state is utilized to address the mobility related in a mobile ad-hoc cloud. This technology is known as Software-Defined Networking.

2.5 Software Defined Networking

As a mobile ad-hoc cloud is deployed over a MANET, the disruption is per-se pointing towards an inflexible network architecture. To eliminate inflexibility in the network infrastructure, a novel technology called Software Defined Networking (SDN) is introduced. More recently, the orthogonality offered by the SDN concept, the main objective has been to decouple the forwarding plane and the control plane of network elements and thereby making the network completely programmable. This, in turn, differentiates SDN from traditional networks such that with SDN the entire network system becomes a more flexibly manageable application based virtual entity. The openness and programmability of the network control plane given by SDN enable network operators and network administrators to manage their network functions avoiding the limitations in legacy networks which can be described as below.

2.5.1 Traditional Networks and Challenges

Traditionally, networks are composed of vendor specific networking devices such as switches, middle-boxes, routers etc. which direct network traffic based on the limited visibility within each device about the network. The internet that matured over time, large enough to provide a

global connectivity today are composed of such networks. In lieu of this there has not been a network architecture which evolved immensely to support the traditional organizational requirements.

However, these networks are neither adaptable nor flexible. That is, the network intelligence does not dynamically customize the switches and other network elements due to it's ossified nature. We consider certain factors such:

- **Manufacturers:** The primary challenge faced in network industry is choosing appropriate devices for designing a network. Different providers manufacture different types of network devices, some are run on their native proprietary protocols and services. This impedes the ability of mixing use of different vendor/manufacturer devices. Apart from that, the product service life cycle ends in a limited time because of the constant upgradation of newer devices; hence, the consumers are persuaded to go for the new services. Lack of a widely accepted standard and open interface among them are some of the factors that veil the ability of network operators to customize the network.
- **Rigidity and complexity**: Currently, the network technology consists of discreet sets of protocols built to interconnect hosts reliably based on metrics such as link speeds, topology, physical distance, etc. Due to business and technical requirements for delivery of reliable, better performing, secure and broad network connection, these networking protocols have grown rapidly. However, networks have become more complex and largely tied because of a large number of protocols involved to fulfill various business requirements. For example, each time a new system/device (or a virtual machine (VM)) is connected or disconnected from the system/datacenter, the network administrators have to configure network routers, switches, firewalls, authentication and access control lists, virtual local area networks (VLAN) and other

protocol based modifications. However, these configurations are highly dependent on the network topology construction, device capabilities in terms of storage processing, etc. On the other hand, now devices and servers have begun supporting hardware virtualization unlike before. VMs are migrated among servers in different locations to optimize and load-balance server workloads within datacenters

- **Impeding Network policy growth**: In current network architecture a plethora of protocols were invoked to define the behavior of flow paths within a network based on QoS, and other policies. As a result, if a new device is added to an existing network composed of middleboxes and network elements, a rigorous configuration is required covering multitudes of devices and mechanisms. The reconfiguration across the entire network when each time a new device or a virtual machine is added to the network would be an onerous task. This makes an overhead cost for the firm, and also adds intricacies with respect to the configurations to provide a consistent set of access, QoS, and other policies adaptably to the network. The failure to provide these services in time would leave many limiting consequences such as security attacks in the enterprise.
- **Scalability**: Usually in any cloud (whether it's data centre specific or peer-to-peer) it offers services like storage and processing to name a few to the consumer. As the number of users grows, the cloud resources needs prompt increase or decrease in services to meet the demand; including change of infrastructure and the network. However, the constraint is when each time a new consumer requests resources based on needs, entire or a part of the network needs to be modified and managed manually. This procedure involves various consumer who request services dynamically. It is obvious that such network scaling cannot be done with manual intervention to configure on-the-fly.

These drawbacks have caused a mismatch between the consumer requirements and other network capabilities. As a result, new standard with an open network architecture named Software Defined Networking was brought into picture.

2.5.2 Software Defined Network Architecture

In Software Defined Networking, the data-plane functionalities are independent from the control plane logic [30]. Figure 2.3 shows the general SDN architecture as proposed by Open Networking organization. It depicts that the system is divided into three main layers, application layer, a control layer, and network infrastructure layer. The core networking decisions are taken by upper layers; mainly the control layer is responsible enable communication between network infrastructure and applications.

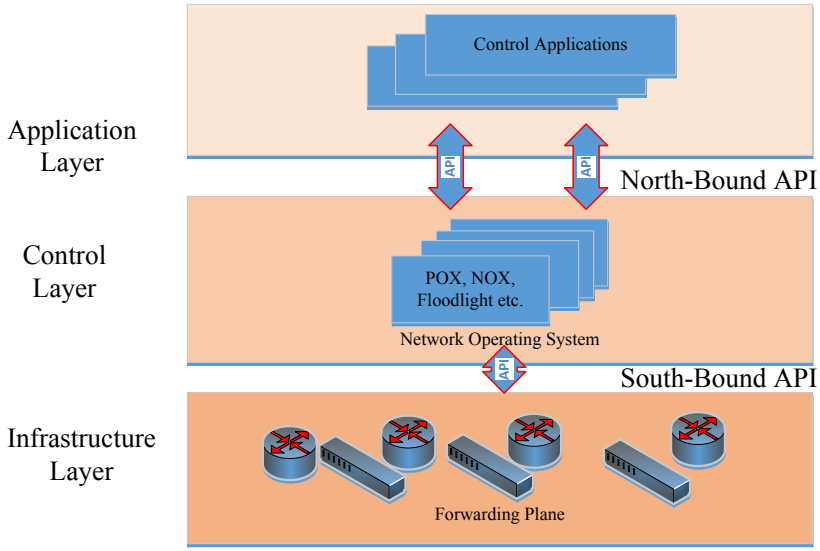


Figure 2.3: Three Layers of the SDN architecture [30]

In SDN, since entire network intelligence is logically centralized, the control layer has a global view of the entire network. This gives the applications to handle the network resources and applies policies considering the entire network as a single logical switch. On one hand, network operators can control the whole network from a single logical point through the controller simplifying both network design and operations. Also, on the other hand, SDN makes network devices simply controllable by hiding their internal processes and protocols, by merely letting the applications to access the network devices through network controllers.

The primary advantage of the large scale network operators is that, they can configure a network from a single logical point rather than configuring multitudes of network devices manually. This is mainly due to the control aspect of all devices that are managed by the SDN controllers. Also because of its centralized intelligence, network operators can handle/control the network in real time and deploy new applications to serve adaptably within a short period of time unlike traditional networks. With SDN, the IT administration does not have to wait till their policies or features are embedded in a vendor-specific device, instead, they can code the required feature as an application and run it via the controller. This is possible because SDN provides a standard API wherein a control algorithm provides the ability to implement typical network functions such as routing, access control, load balancing, QoS, path/route/energy optimization and many user other functions and policies. Ergo, SDN makes the entire network management easily feasible through intelligent orchestration and provisioning systems.

Open Networking Foundation has introduced SDN and provided guidelines for future SDN-based implementations. Most importantly the open API that is provided by them is useful in managing networks with multi-vendor devices that provides the services such as on-demand resource allocation, virtualized network management and secure cloud services etc. [30]. There

are several implementations based on SDN specifications. [31] shows that one such implementation is OpenFlow.

2.5.3 OpenFlow Protocol

OpenFlow has been considered as the most popular and majorly accepted implementation based on SDN specifications. OpenFlow is a control protocol that is used to define a secure communication between the control logic and the underlying data plane. With OpenFlow, the data plane network elements are commonly made compliant with the protocol hence are referred to, as OpenFlow switches. The OpenFlow protocol enables the update of OpenFlow switch flow table based on the application running on the control. The usage of the OpenFlow protocol and OpenFlow switch is depicted in Figure 2.4. It shows mechanism by which the OpenFlow [32] compliant switches communicate with the external controller. Inside the switch hardware, a flow table is maintained. The flow table contains three major fields namely header field, counter and a set of actions. The header field of the flow table is defined by the controller to classify flows of packets. Then the switch checks the header field against incoming packet to obtain the corresponding action. The action is also configured by the controller. In addition to that, there is a counter module which keeps dataflow statistics such as the packet count.

The OpenFlow switches are flexible and can be configured in different ways such that each flow that reaches the switch can be treated in separately based on the flow table. Most commonly the switches are configured such that, any flow that does not contain a matching rule in the flow table are either forwarded to the controller or dropped [30-32]. If the flow rule update request is forwarded to the switch, it is done by sending an event including flow header information using

OpenFlow protocol. The application running on top of the controller can handle an action for the flow by sending an action encapsulated in OpenFlow protocol back to the switch. With OpenFlow, such flow processing can be done either proactively or reactively [69]. In proactive processing, flows are installed into the switch flow table in advance before the arrival of any matching flow. In contrast to proactive processing, the reactive flow processing method requires each flow that either does not contain a matching rule or certain matching criteria are met, to be forwarded to the controller for processing. There are some trade-offs we need to consider as both the ways contain its own pros and cons based on the usage, however, due to the adaptability and flexibility offered by OpenFlow, a unique action for each flow can be defined.

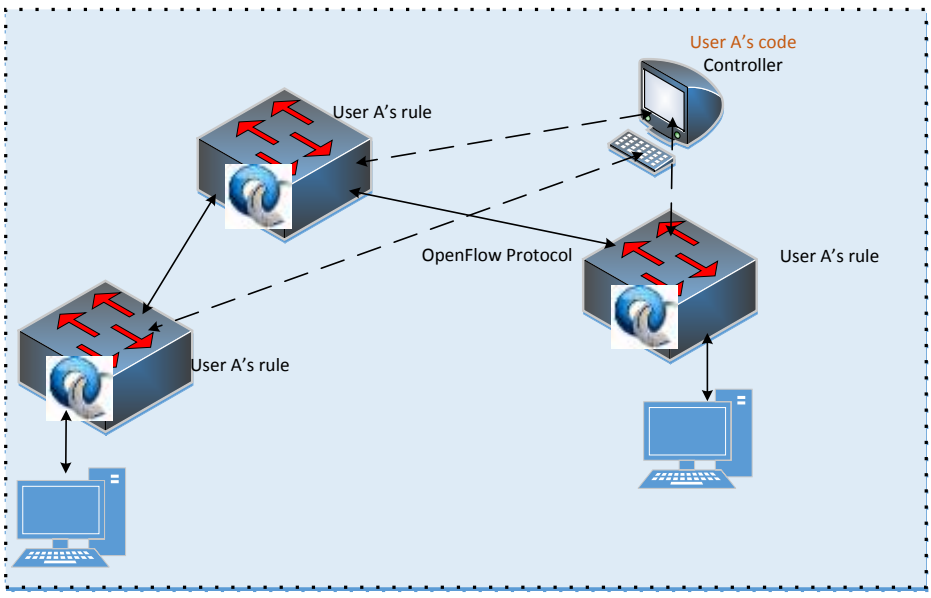


Fig 2.4 OpenFlow allows a remote software to have the control Intelligence [32]

Figure 2.5 shows the fields of incoming packet that is used to match against header field of the flow table in OpenFlow-1.0. If there are multiple match entries, a priority value will define the corresponding action for a particular packet. The flows are processed based on the physical layer (incoming switch physical port) parameters to the transport layer header portion of the incoming

packet.

Ingress Port	Ethernet Source	Ethernet Destination	Ether-Type	VLAN ID	VLAN Priority	IP Src	IP Dst	IP Proto	IP ToS Bits	IP ToS Bits	TCP/UDP Src Port	TCP/UDP DST Port

Figure 2.5: OpenFlow Primitives [33]

With further research in OpenFlow, the flow table characteristics were evolved supporting pipeline processing involving multiple flow tables. With pipelining, a matching flow can be forwarded for further processing in subsequent flow tables [33]. Then, the flow table is extended with additional parameters such as priority, timeout, and instructions etc. [69] also shows the essential details of rule installation.

2.5.4 Software Defined Wireless Network Architecture

The earliest effort in applying SDN principles in mobile networks in seen in [34]. Various tests conducted in [34] primarily, points out how a seamless movement between radio technologies can be maintained using SDN. OpenRoads architecture is divided into three layers. These layers are (a) flow layer for data path flow table maintenance, (b) slicing layer that makes use of FlowVisor . This FlowVisor layer appears to the switches as controller proxy, there are multiple guest controllers which are present on this layer that control the switch. (c) The experiments are carried out with NOS which has the network wide view of the topology and state in the network. Figure 2.6 shows the architecture where the visible transition from wired to wireless networks causes the data-plane to have SDN wireless nodes, or other wireless nodes along with the backhaul components in the network.

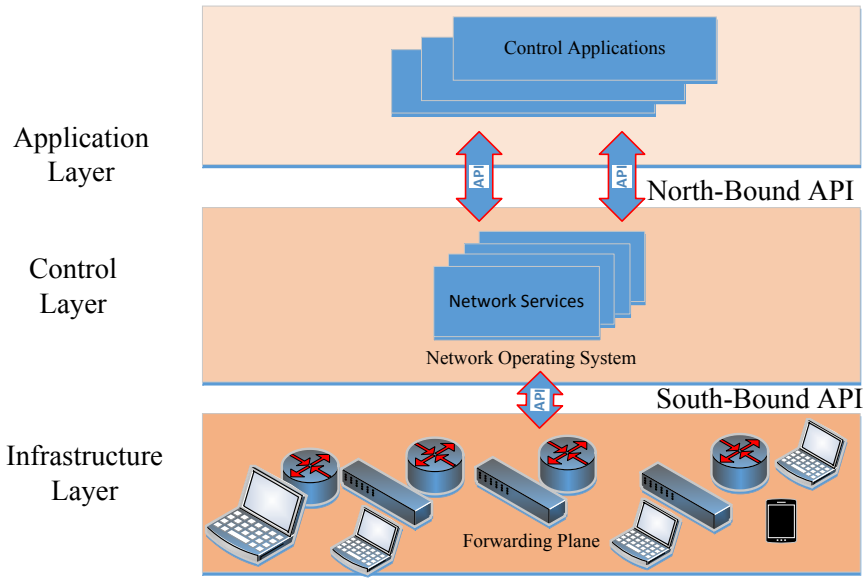


Figure 2.6: Three Layers of the SDMN architecture

In [35], authors explore the girth of SDN further by attempting to utilize its principles in wireless ad-hoc networks. It shows that application of SDN principles in ad-hoc networks has the potential to improve the limitations of such networks in terms of performance. It makes use of the AODV protocol over Wi-Fi to route the initiator's request among the nodes that become part of the ad-hoc network. This brings in a novel idea of having a local SDN module inside a phone that is inherited by authors in [68]. [70] Employs an SDN based Mobile Cloudlet where the usage of SDN shows how the centralized controller can oversee the traffic path and enable flexibility of traffic paths.

The Architecture in [36] incorporates SDN in Device to Device Mobile cloud. In doing so, authors indicate effective spectrum utilization, system throughput and energy efficiency of network. However, authors fail to realize the three important functions namely, adaptability,

flexibility and traffic monitoring using SDN. D2D only deals with communication between devices in very close proximity and devices should be D2D compliant.

In [37] authors propose a framework for mobile offloading, however it follows the traditional data-centre specific offloading scheme. By doing so, authors want to show traffic management based on the user subscription without modifying the existing EPC architecture. As a prequel to this work [38] presented an architecture that made significant changes to the EPC architecture but maintained the in-network offloading architecture for delay sensitive applications.

In [39] the essential details of SDN in mobile networks is demonstrated. It elaborates on how a change in legacy network architecture can have positive impact in the performance of the EPC, however, it fails to make any reference to mobile ad-hoc cloud computation. The researches in software defined mobile networks and wireless networks are deep rooted and are moving towards expanding the capacity of the network by providing a wide variety of services. One such impacting research is the ability of the Software defined network principles to be adapted by ad-hoc cloud services that has the possibility to offer a seamless service. In complete coherence with this research we look at how service deployment and maintenance is possible in such scenarios in the following chapters.

2.6 Summary

In this chapter, we have analyzed the concept of ad-hoc cloud computing and its formation and benefits along with some trade-offs with conventional cloud computing .Then we emphasized on how clouds can support rich mobile service by addressing the intrinsic limitations of mobile

devices. We discussed different ad-hoc cloud computing approaches and how the shared benefits of volunteer computing is tapped. Based on the interest in mobile ad-hoc cloud in recent research works, we have described the advantages of formation of an on-the-fly mobile device cloud or a mobile ad-hoc cloud.

Motivated by these benefits, we conclude that the idle intra–device mobile resource when harnessed can be powerful and robust to satisfy certain classes of applications. We then explore the essential concepts on which our framework is built. In addition to that, we look at how a MANET has gained attention and discuss some of the limitations in existing network architecture that disrupts services in case of a MANET. And subsequently, we explained how SDN-based solutions can address the legacy network architecture challenges. Finally, we look at how SDN in wireless networks have improved performance and by using these benefits of SDN, we described proposed approaches for mobile ad-hoc cloud.

3 An Infrastructure as a Service for Mobile Ad-hoc Cloud

3.1. Introduction

In this chapter we further delve into the concepts that enabled us to build a Mobile Ad-hoc Cloud. Essentially, there are a certain class of applications that are solely dependent on place and time in which the applications need to be executed. Therefore, such applications need assistance from the closest available computation environment for providing Infrastructure-as-a-Service. The role of a mobile device in providing infrastructure and each device playing an integral role of a provider is highlighted in this chapter. We further describe the proposed IaaS provisioning entity. Additionally, we show how task scheduling is done in a Mobile Ad-hoc Cloud optimizing the number of devices that participates in a mobile ad-hoc cloud composition. In order to first understand the interplay of proximity with the provisioning of resources to a consumer we first look at the significance of mobile cloud proximity.

3.2. The Significance of Mobile Cloud proximity

The genre of mobile applications has shifted over the decades. This has not only obfuscated the mobile processing but has also rendered the intra-device resources to be useless. An obvious way to take responsibility of these heavy duty applications is to offload them to the cloud that has paved the way to the confluence of mobile computing to cloud. Nevertheless the cloud resources, although efficient in providing performance scalability, has not seen a positive change in the overall time taken for the services to reach the consumer This has put the research community to

find the optimal placement of the cloud infrastructure to aid the consumer in the lowest possible time.

A pioneering concept of Cloudlet was introduced in [40]. According to this concept mobile devices offload their workload to a local collection or cluster of computers that are connected to the Internet and are available for use by nearby mobile devices. This collection of computers is known as a cloudlet; it is beneficial in situations where mobile devices can connect as a thin client to the cloudlet (instead of connecting to remote servers in the cloud). Examples are coffee shops, shopping malls, stadiums, campuses, airports and train stations. In the cloudlet infrastructure, mobile users can execute specific, context-aware applications by a low-latency, high-bandwidth, and one-hop wireless connection to the cloud [40], thereby avoiding latency and reducing bandwidth costs. To this end in [41] authors show how the Gabriel architecture works in the cloudlet. The authors believe that for the task processing and computation to happen in a distant cloud and get a response there-in, latency increases. Therefore, with the cloudlet working close to the device, the processing of a heavy duty application like a cognitive assistance application is possible close-by with reduced latency. This work gives the reader an instance of Lego reconstruction done with the help of this framework and justifies the idea of an effective three tier hierarchy of a "mobile device-cloudlet- cloud" that enables faster query-response compared to the mobile device-cloud architecture

Among close proximity mobile cloud services, we can find another research direction called Edge Clouds. The concept of Edge Clouds uses small scale datacenter at the last mile closer to user locations. However [42] brings a novel research idea with mobile devices forming the "Edge Clouds". The authors in [42] show that a mobile edge cloud is primarily a resource-rich entity that performs a computation very similar to the traditional infrastructure cloud.

Nevertheless, the main advantage of a mobile edge cloud is that it can out-perform an infrastructure cloud in terms of both latency and battery power. This way the cloud service provider can maintain the required QoS for mobile users within a given network. A comparative study conducted by the authors suggested that between the traditional infrastructure cloud and a mobile edge clouds, there are classes of applications which require distributed data comprised of independent datasets that could be processed and executed faster at the nearby edge cloud than the traditional infrastructure cloud. Thereby reducing the overall latency compared to the task carried to the traditional infrastructure cloud.

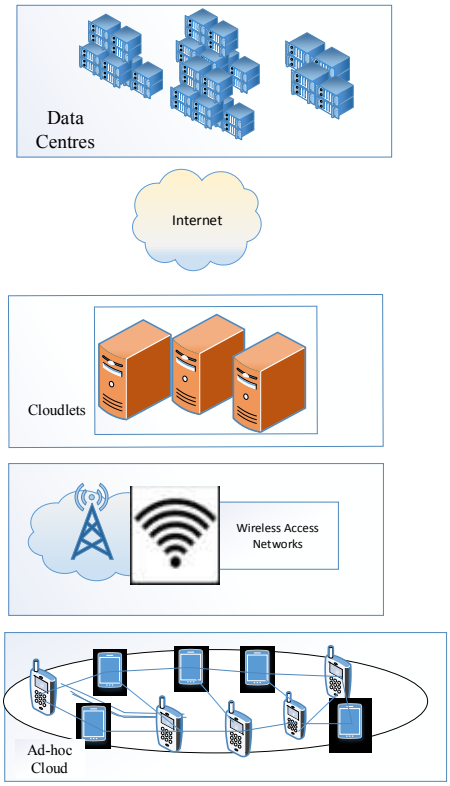


Figure 3.1: Three tier mobile cloud computing architecture

As elaborated in [40] [41], it becomes highly predictable that following a 2 tier approach makes the cloudlet the closest resource entity. We argue that provisioning of the ad-hoc cloud model

would provide a three tier approach that would have a better promise as shown in Fig 3.1. Moreover, in most of the closer proximity mobile cloud concepts such as ad-hoc clouds and cloudlets, the major benefit is the lower communication delay. Due to the availability of cloud resources in the nearest vicinity mobile devices can acquire infrastructure on demand for offloading mobile applications data. In ad-hoc clouds, the cloud resources are provided by the other mobile devices forming a mobile device cloud using device to device (D2D) communication methods. In the cloudlet model, the cloudlets are placed in the base station or aggregation nodes of wireless networks and hence the cloudlets can be accessible over the Wi-Fi or cellular links with minimum hops in the communication path.

Recapitulating all the aforementioned facts, in [20] authors come up with another approach. Understandably, the concept of mobile devices leveraging nearby computational resources for reduced execution time and consumption of energy, is essentially a step towards minimizing the overall latency and increasing the network lifetime. The authors in [20] provide an architecture that is run on a mobile device and discovers the capabilities of their environment based on which the computation offloading takes place. Based on the profiling of the task resources are allocated. The sub-division of the task takes place at the task profiler. The decision of whether the execution of the sub-task should be done locally inside the mobile, in the Mobile device cloud [43], at the cloudlet or the cloud is done at the offloading manager. The task once computed is brought back to the initiator by the routing and replication manager. In order for successful delivery within a given time, task can also be replicated in this module. A scheduler for maintenance of each sub task and to dispatch them to either local compute resources or move it to the forwarding manager for computation to be performed at the remote location is essential. For this, the architecture has a Task Scheduler which acts a maintenance block. Once the task is

at the forwarding manager, databases are updated about the neighbours and other exchanging histories. The stored information about historical contact summaries and other social information is observed for expected connection and inter-connection duration between devices. Although moving towards the ad-hoc mobile device realm involves intermittent connectivity and a high degree of unpredictability, the idea of opportunistic ad-hoc cloudlet service (OCS) is proposed in [44]. The three categories the authors talk about in are based on the following service modes 1. OCS (Opportunistic ad-hoc cloudlet service) with back and forth connectivity- In this case the user mobility is typically restricted to a specific area, in order to guarantee frequent meetings with the computation node and the service node. However, the mobility is comparatively higher than the Remote Cloud Service (Wi-Fi). 2. OCS with 3G/4G connectivity – Here, the offloading of the sub-task is done via the 3G/4G connectivity which demarcates more freedom in the movement of the mobile node. So without a Wi-Fi the offloading of the subtask depends on the ratio of the computed sub-task result and the sub-task received for computation 3. OCS with Wi-Fi In this category, the mobile node roams to another cell with Wi-Fi. In this case the communication cost is between Remote Cloud Service Wi-Fi and Remote cloud service using 3G/4G.

Although, many studies have referred to the cloudlet as a “proxy cloud”, it does not completely remove the latency related issues. Moreover, the time to discover a cloudlet is also heavily dependent on the available network infrastructure. This resulted in various approaches finding a more ubiquitous platform for resource poor devices to find a rich environment for computation. Research in [45] posited some preliminary results in their work. In their framework, the mobile nodes detect nodes in the vicinity that are in a stable mode that is according to their framework

only those nodes are chosen for provisioning that have the same movement pattern or in the same area. Therefore, if the nodes in the state are present, the provider is chosen for the application.

Similar to this research [46] present AMCloud that puts forth the advantage of harnessing the idle computing resources of mobile devices. Authors look at how the distributed framework of an ad-hoc cloud impacts the system management. Moreover, the research work is more inclined towards providing security and privacy issues in ad-hoc cloud computing. While looking to harness the local device resources authors have critically analyzed ad-hoc cloud into two modes:

1. Static Ad-hoc Cloud – This model can be defined in the same way as Gary McGilvary et al. have done in [3] wherein the static cloud shares the features of Grid and Volunteer Computing, it incorporates new features of elasticity and co-ordinated use of computing resources employees or users in an organisation.

2. Mobile Ad-hoc Cloud – This model can be evaluated in the same manner as Mtibaa et al. where in the idle computing resources of a mobile device owned by same or different individuals can be put to productive use . Various applications that authors target include ad-hoc multi-party gaming, object localization and tracking, multi-media content sharing and distributed environment monitoring.

In alignment with the same [47] provided a Mobile Ad-hoc Cloud management platform called PlanetCloud that dynamically configures an MAC. PlanetCloud is adapted from another task management platform by [48] called CyberX which is an autonomously managed virtualization layer. In this model however, author consider two nodes similar to a server and a client which they call:

1. A fixed control node – A cloud agent (CA) runs on this node. The CA manages the cloud that is formed and keeps track of all the resources joining the cloud. Owing to functionality of the CA, the deployment is carried out on a high capability node to manage and store the data.

2. Mobile Computing Node- A tenant agent (TA) runs on this node. It manages all the participants' local resource. Moreover, all other agents are connected who are involved in the formation of the cloud. Further, it synchronizes the local resource calendar with the global resource calendar on a CA.

In spite of this platform being too complex to execute, the simulation results show the high reliability in provision resources in an ad-hoc manner at the same time maintain the computation power of the devices in an MAC. Inspired by these works and striding by the endeavour to provide a low cost computational environment to a consumer we look at how a mobile infrastructure can be tapped with a layered architecture that exhibits ease of usage with very less complexity and at the same time considers the needs of a consumer.

3.3 The Architecture of the Mobile Ad-hoc Cloud

In this section, the details of the system framework depicted in figure 3.2 are discussed. Recall the concept in chapter 2 that elaborates how the mobile phone virtualization technology has augmented the development of mobile ad-hoc cloud paradigm. Striding by those facts we now look at the system framework that enables a low cost computational entity for situation centric applications.

3.3.1 System Architecture

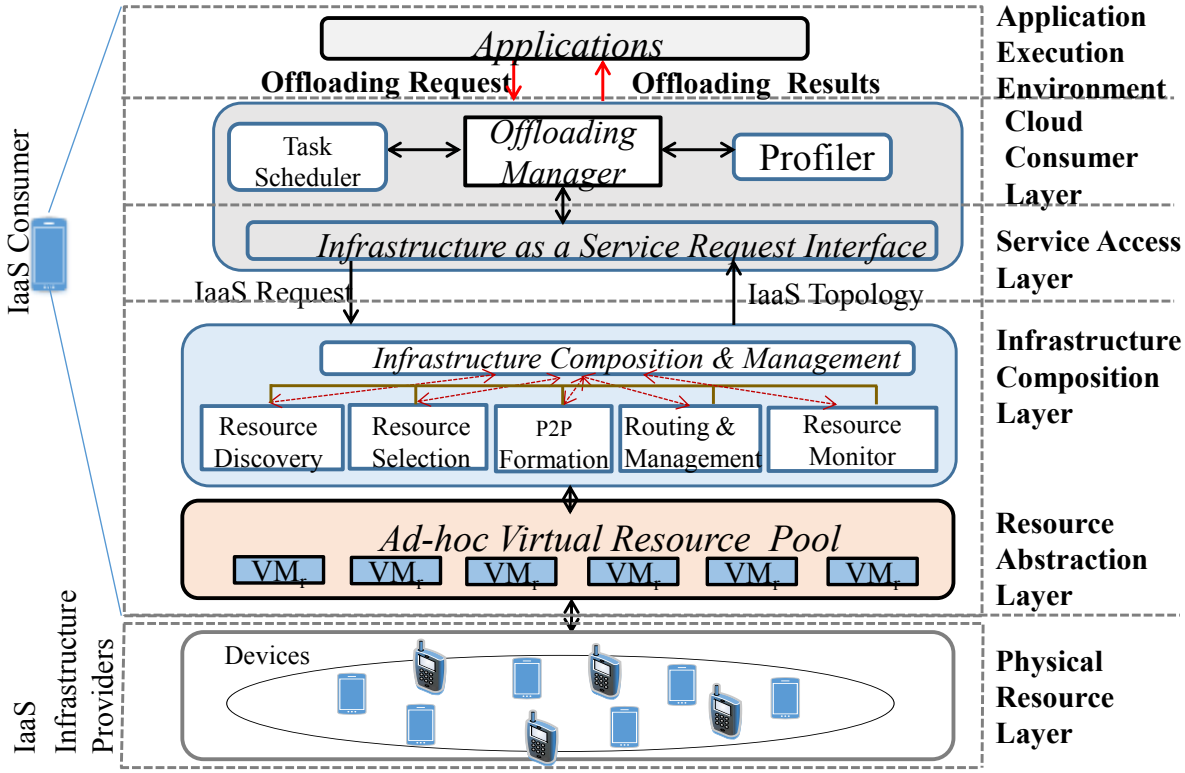


Figure 3.2: Mobile Ad-hoc Cloud Architecture

The architecture has two integral parts that are responsible for A.) IaaS request generation B.) IaaS composition and provisioning.

A. IaaS request generation

In this part, the request is generated. It consists of the following:

1. **Application Layer-** These are any user application that could make use of services in a crowd-sourced environment. The applications are agnostic to the provisioning mechanism and the interactions between the layers below.

2. **Cloud Consumer Layer** –This layer receives the mobile device application's offloading requests. The Profiler does the decomposition of heavy application tasks into light-weight jobs. It gives the information of execution profiles to the offloading manager. The decomposition process of a task is integral to decide the formation of an ad-hoc cloud. As we have seen in [49, 50] if we consider a Task (J) then decomposed set would look as follows.

$$J = \{j_1, j_2, j_3, \ldots, j_i\}$$

We will see in-depth the entire process of how the sub-task profile is affects the final cloud composition formation in the upcoming sections.

The Task scheduler constructs a queue by mapping the execution profile to the node profiles. The foremost goal in scheduling the jobs is considering the network parameters (3G, 4G, Wi-Fi or Wi-Fi-direct) at the time of offloading for the purpose of minimizing the cost. Further, there is a need to engender a mechanism that not only officiates minimizing devices in an ad-hoc cloud to reduce synchronization overheads but also choses the best available devices for the formation. The task scheduler follows an algorithm to schedule the decomposed jobs to compute/storage resources obtained after discovery. The Offloading manager accepts the information from the profiler and coordinates with the task scheduler input to queue these requests. The decision of whether to offload or not is made here. Once a decision to offload is taken, an IaaS request is made to the cloud provider. As the VMs of the devices are received and utilized by this module, the offloading manager behaves as a consumer. There is an obvious possibility of certain tasks that can be computed within the local device, therefore, the native capability of a device can by-default be estimated so much so that the previous task performances can assist with the

information of whether or not a device has enough resources to execute a given task/sub-task. Eventually, we will see with the help of a composition score how this is achieved.

B. IaaS Provisioning and Composition

In this part, the infrastructure composition is assembled and made available in usable form to the consumer. It consists of the following:

1. **Service Access Layer -** At this layer, the IaaS request interface is responsible for handling the IaaS request. This acts as a conduit between the cloud consumer and the IaaS provider. This is because the IaaS request cannot be made directly to the individual heterogeneous providers. Therefore, it is responsible for providing a set of service interfaces and resource abstractions (e.g. Virtual Machines) obtained from the vicinity to the consumer in a usable form. It is only concerned with the receiving of service requests and provisioning of services. In general, this layer could be defined as the uppermost layer in the IaaS provisioning mechanism.

2. **Infrastructure Composition Layer -**

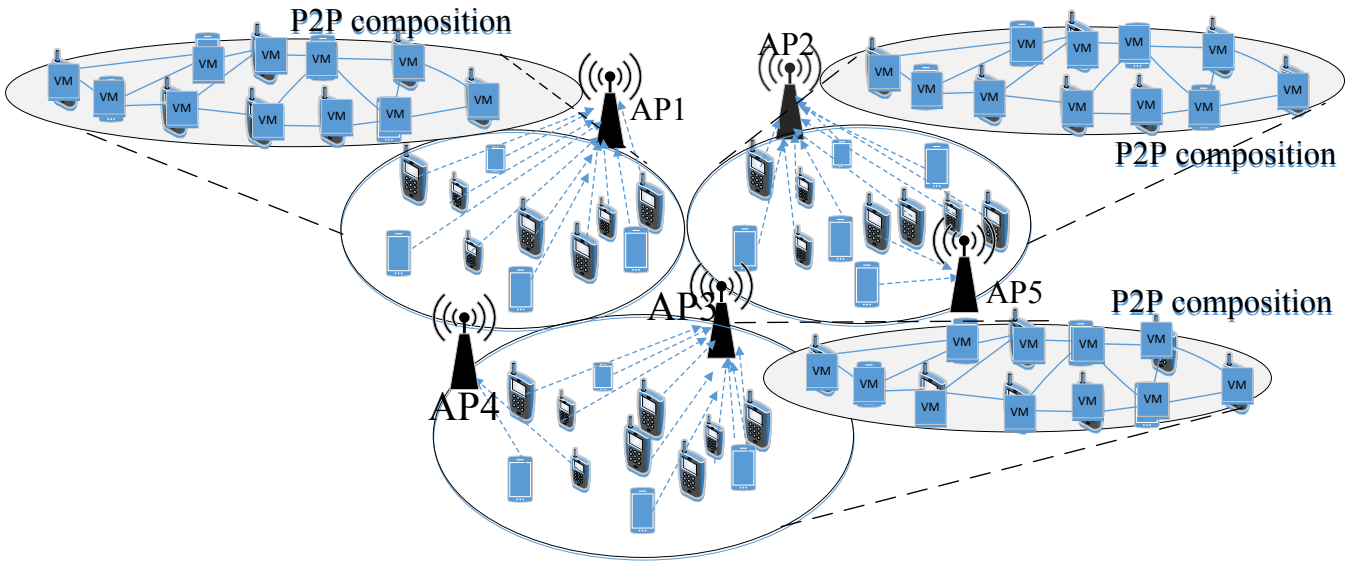


Figure 3.3: Multiple P2P compositions at different Access Points (AP)

This layer has the **Infrastructure Composition and Management** module which is the core of the architecture. This module is responsible for composition and management. It constitutes the essential functions of the architecture. This includes discovering resources, forming the physical layer and populating the ad-hoc virtual pool, the formation of the P2P network and managing the resources. As these resources are diverse in nature with different capabilities, a composition algorithm to unify them is proposed. Additionally, due to the heterogeneity, co-ordination between the resources is necessary. Thus, a key based routing mechanism is followed. This modeling approach allows easier resource management and spontaneous IaaS provisioning. Moreover, a composition strategy in IaaS provisioning is essential as the Service layer does not have the logic required for the unification of the disparate resources. This layer creates the composed resources from the ad-hoc virtual resource pool. The generated composition is the only view for the layers above. Each of these modules is explained below with their functionalities.

The **Resource Discovery** module is responsible for an examination of available resources in the vicinity. That is, it follows a publish/subscribe mechanism to search for the IaaS providers. It is the first step towards the deployment of the IaaS composition. The search involves discovering volunteers and populating the ad-hoc virtual resource pool. These volunteers together become part of the volunteer ad-hoc resource pool (VARP). Once discovered, the volunteers offer their intra-device virtual machines. As seen in Fig 3.3 , the composition can be formed close to any access points in a region where the resource discovery has been made. The figure elaborates presence of multiple Access Points (AP) and multiple compositions at respective AP.

The **Resource Selection** module optimally selects required virtual resources from the resource pool that is created. These are the VMs of the devices which satisfy the IaaS request. Once selected out of the volunteer pool, these are used as participants in the composition.

Once the participants are selected, the **P2P formation** module performs the composition of the selected device VMs. These VMs have an interface and a computing capability similar to the underlying device. We consider compute and storage services as our primary objective is firmly rooted towards this services. The composed participant topology (CPT) is formed by combining multiple virtual resources from the vicinity that were formerly part of VARP. How the composition of these virtualized resources takes place is elaborated e ventually in this chapter.

The **Routing and Management** module's role begins once the P2P network is formed. It accesses the storage that has the dependencies specific to a request and integral for the managing of the resources. For example, as seen in the previous section, a meta-file is taken into consideration that acts as a dependency. Once resources are composed, the requests need to be serviced with the assistance of IaaS providers that require co-ordination and management. It takes the decision about the route to take and the devices to be chosen when using the composed service. Hence, the routing and management algorithm makes use of a key that eventually takes the dependencies to the VMs. The IaaS algorithm comprises of this key based routing described later with an example and evaluation in chapter 5.

The **Resource Monitoring** module's interaction will involve frequent exchanges with the VARP, defined in the algorithm below that will be essential for recognizing failures and reconfiguring the CPT. Additionally, as CPT is a sub-set of VARP, it is also possible to adjust the configuration of the composition by joining new resources in the pool. In this chapter we

look at very few to no disruptions, thus, it is impertinent to delve into the node failures at this juncture. However, in chapter 4 we address this factor in its entirety.

3. **Resource Abstraction Layer**- This layer contains the mobile phone virtualization [10] components that the cloud IaaS providers use to provide and access the physical resources. It represents a collection of virtual resources collected from the volunteers forming the ad-hoc virtual resource pool. Here, the devices that offer their VMs/VPs have the same characteristics to the respective physical Node IDs in the Ad-hoc Virtual Resource Pool. Thus, in general, it could be said that the ad-hoc virtual resource pool is a combination of VARP and CPT. The cloud IaaS providers have control over these abstractions. There could be multiple such abstractions which the cloud IaaS provider can offer.

Owing to the isolation between these abstractions, flexibility in service orchestration is achieved.

4. **Physical Resource Layer** – Physical Resource Layer includes the physical devices obtained from resource discovery. This is the lowermost layer with hardware resources such as phones, tablets, and other physical computing infrastructure elements. These are the entities providing the virtual abstractions for computation. In other words, these are the cloud IaaS providers who own the virtualized resources.

3.3.2. Mobile Ad-hoc Cloud Composition Algorithm

The IaaS deployment requires the assistance of an algorithm in the mobile ad-hoc cloud. It is divided in to two, these are 1.) Composition algorithm 2.) Routing & Management Algorithm. The advantage of these two algorithms is to extend flexibility and simplicity. It achieves these

characteristics by orchestrating the discovered devices (shown in Figure 3.4) to satisfy the IaaS request. Additionally, the Routing& Management algorithm ensures co-ordination among the composed resources. Therefore, requests can be submitted at any time, and ad-hoc cloud can be formed on the fly.

A. **Composition Algorithm-**

All uploaded files are analyzed, decomposed and scheduled based on the network parameters discussed above. As shown in Fig 3.4 first, a cloud consumer will prepare the request. Once the service request is received, a resource broadcast is made after which resource information is obtained. The resources obtained any time later than *send(msg, t)* are considered to be evicted The resource information of Node ID, IP, and port from listener nodes is used later to form a P2P network, post the selection of resources and session establishment. A bootstrap construct is sent to the provider nodes. Once bootstrapped the VMs/VPs to form a P2P network. This is how the resources are composed.

B. **Routing and Management Algorithm**

This algorithm performs request-specific dependency retrieval and maps it to VMs where the computation can be processed. It makes use of a consistent hashing scheme for generation of the key. Thus, for a given value a corresponding key will take the dependency to the correct VM, where it is downloaded. For one request, consider an example of a meta-file formed with the resource information obtained assimilated along with the job information. This meta-file is the (that is stored in the key-value storage) value for keys generated. As the keys point to the meta-data values, if there are two similar keys then they'd be pointing to the same location from where the dependency needs to be downloaded. The unique node Ids that are bootstrapped to the

consumer device distinguishes between devices. Once composed the Routing & Management module armed with the information from the storage then informs the composed participants in the VM pool where the actual data exists. The individual VMs can then begin downloading the files for execution. Failed nodes can be determined by the resource monitoring module. Before we look at how resource monitoring takes place, there is a need to understand how the jobs are scheduled in a Mobile Ad-hoc Cloud and make a judgement about the complexities that need to be addressed.

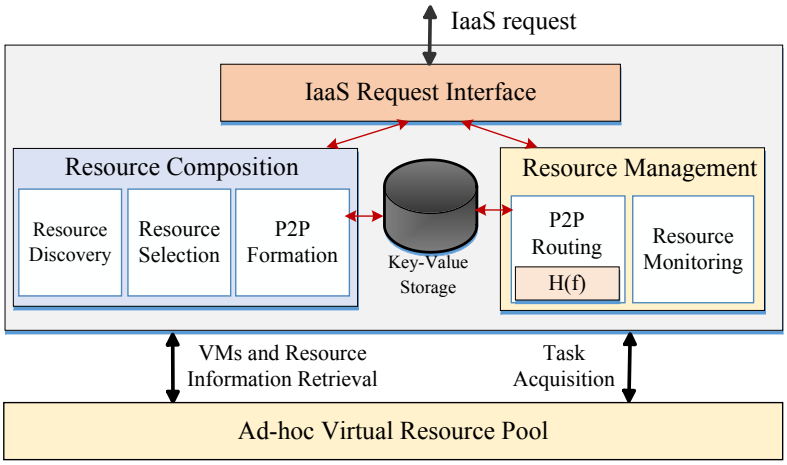


Figure 3.4 . Element interactions and dependency retrieval

Performance evaluation of the IaaS algorithm is done in chapter 5 wherein we go in-depth into the algorithm construction and implementation.

3.4 Job Scheduling

It has been established before that the Mobile Ad-hoc Cloud computing paradigm enables a computation environment closest to the user . However, by following this strategy there are more

complex unprecedented problems such as constant device movements, disruptions in the external device to name a few that needs to be addressed.

Hence, a vital process is the scheduling mechanism in a Mobile Ad-hoc Cloud. Ostensibly , there occurs synchronization problems and intra-device disruptions that can breakdown task execution in the mobile ad-hoc cloud. It is clear that every node has enough resource to execute a certain number of sub- tasks that falls within it's resource capabilities but there has to be a strategy which allocates the sub-tasks taking into consideration the resource limitations of the device.

Further, the strategy should also ensure minimum devices participating in execution to reduce the synchronization overheads. In doing so, it would lead to a faster ad-hoc cloud composition formation, that in turn provides quicker task execution in a Mobile Ad-hoc Cloud. There are different ways of scheduling tasks based on the serialization or parallelization of sub-tasks. We now look at different scheduling mechanisms.

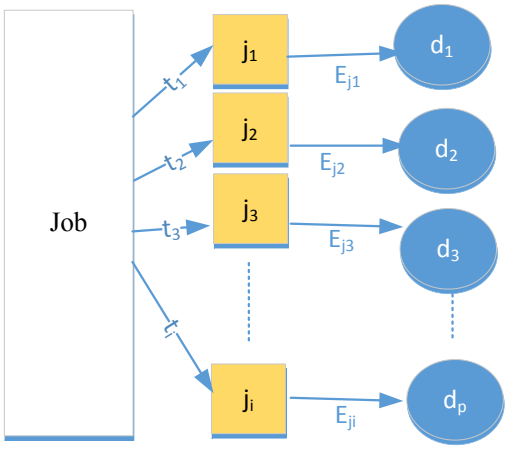


Figure 3.5 : Job Scheduling Model

3.4.1 Overview of Schedulers

Schedulers and Scheduling Algorithms in general are prescribed based on the need. Some algorithms look at scheduling from the perspective of the profilers output whereas some models are defined based on the connectivity and contact durations with the assumption of a profiled set of tasks already present. As the focus here is to look at the complexities of the scheduling mechanism after the formation of an MAC and not to figure out the best combination for a scheduling algorithm, we recognize that in an MAC it becomes abundantly difficult to handle device synchronization due to a large participation of devices in the vicinity. Fig 3.6 shows an instance of composition formation using P2P links. It is of significance to know that the task dissemination at the offloader has to follow some stringent rules for an optimal execution.

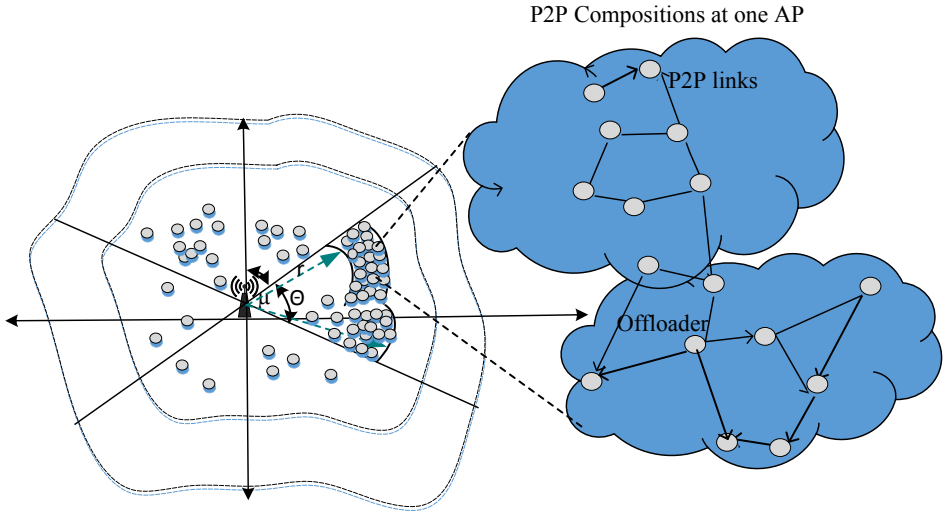


Figure 3.6: Task Dissemination by offloader

Research in [51] shows that it is important to minimize the number of devices in collaborative Mobile Cloud Computing environment because of the extra traffic incurred. Similar to this view,

[52] looks at the task success rate approach and considers the contact durations and intermittent connectivity to offload traffic in a mobile ad-hoc cloud environment.

In [53] the authors use the method of convex optimization to determine the computation amount to different computing entities opportunistically. They make use of the statistic property of intermittent contact rates and exploit the contact patterns for finding this optimal value. Based on the same premise [54] follows a prediction based offloading plan to the central cloud by reducing decision making overheads. [43] elaborates an offloading scheme in an intermittently available device cloud. However, it fails to minimize the number of devices available to address the data-traffic and synchronization issues in such scenarios.

Although, differing in design and methodology from the above approaches, it is our endeavour to strategically prepare our model that follows a decentralized strategy to minimize the number of devices participating in a mobile ad-hoc cloud.

3.4.2 Optimization based on Composition Score

As seen in Figure 3.5 in our model, we consider there is a Task (J) which is divided into n sub-tasks (j). These sub-tasks need to be distributed among the devices as shown in Fig 3.7 based on proper selection to engender an effective ad-hoc cloud. We look at a scenario, when there is no dependency among each job and are parallelly executable. Total number of sub-task is given as:

$$J = \{j_1, j_2, j_3, \dots, j_i\} \quad \text{.....(1)}$$

And total sub-tasks after splitting is given as:

$$J' = |j_i| \quad \text{.....(2)}$$

Let's assume the time taken to profile one sub-task be T_s . Therefore time taken to process n sub-tasks is

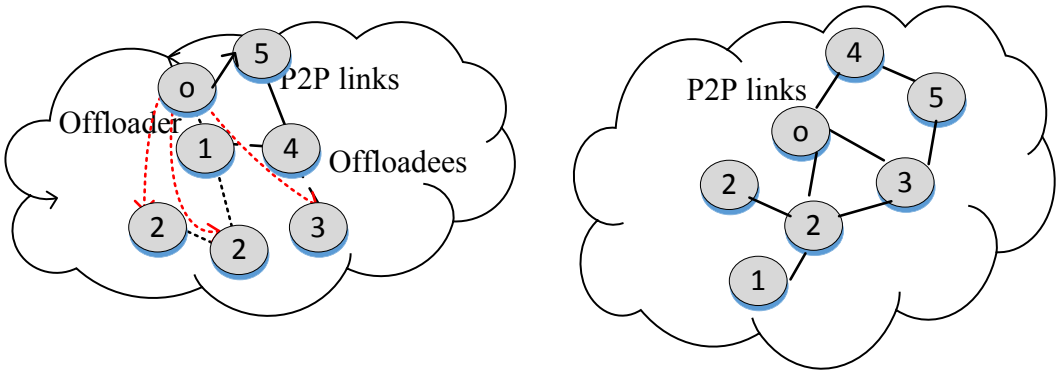
$$T_s \{for\ n\ sub - tasks\} = n * \ T_s \quad \text{.....(3)}$$

Additionally, at the offloader time taken to form or initiate a mobile ad-hoc composition (T_{mac}) is given as:

$$T_{mac} \ = \ T_b + \ T_c + \ T_s + T_d + T_{rep} \quad \text{.....(4)}$$

Where $T_{b/c/d/rep}$ are the time taken to broadcast, compose, disseminate, and obtain a response from devices after execution.

Clearly, this time relies more on the execution of the sub-tasks in another device more than the other parameters. Therefore, T_{rep} could intuitively show how fast or how slow the task was



(a)

Choice of devices from the resource pool post discovery

(b)

Final Composition

Figure 3.7: Composition formation (Units inside represent the C_s)

executed in a device, given the network performance was constant for an event.

Now, let's assume each sub-task has an execution time on each resource to be E_j^n. Consider p devices $d_1, d_2, d_3, d_4 \ldots\ldots\ldots d_p$ are obtained in T_b as seen in Fig 3.5; then each of these devices will have shareable resources of CPU(C),RAM(R),STORAGE(S).

Based on these parameters we model the Composition Score (C_s).

Now, let's look at the Composition Score (C_s). It's a score that takes into consideration the devices capacity, popularity as well as stability. Higher the composition score of a node better are that node's chances of getting accepted in the composed participant topology of the consumer.

A minimum threshold composition score called Min_{cs} is calculated. The popular nodes maintain a value of (C_s) above this score.

Consider a regular two dimensional Euclidean plane as shown in Fig.3.6. For simplicity, we consider only one Access Point (AP) that assists in fetching the number of devices for a P2P composition. For such a scenario we have,

α – As the value given to a node since its arrival and/or last failure after its arrival

T_{dept} - As the time of departure and T_{arr}- As the time of arrival

Q_k - Each service can have ' k ' number of QoS criteria such as service delivery and accuracy

w_q - Each QoS factor is given a pre-defined importance or weight.

$P_i^n(t)$ - Popularity factor of the device who has served `i' number of requests at time `t'

Now, let ` $o$ 'be resources provided with $D_i^o$ being the device resources (` o ' is the CPU (C), RAM (R), STORAGE (S)) .and satisfies ` i ' number of requests. Together these factors give the Composition Score:

$$\text{Composition Score } (C_s) = \alpha \left(T_{dept} - T_{arr}\right) + \sum_{k \in K} \left(\sum_{i \in I} P_i^n(t) + (D_i^o * w_q Q_k) \right) \quad \ldots. (5)$$

We are concerned with situations where the time of departure is greater than the arrival, to ascertain stability of the device in the region at the time of composition formation. Fig 3.7 shows one instance of such a scenario, the composition scores are the units mentioned inside the circles. It can be noticed that the devices that have equal composition scores but are at a farther distance are not chosen while forming the composition. We ignore the case of same composition score and at the same distance for the moment.

Observing the device signal strength (S_n)having a direct impact on its popularity in the neighbourhood, we define $P_i^n(t)$ as

$$P_i^n(t) = \sum_{n \in Z} S_n * P_i^n(t-1) \quad \ldots\ldots\ldots\ldots (6)$$

$(t-1)$ shows the popularity at time before ` $t$ '. Where $n \in Z$; ` Z ' is the nodes in the vicinity of a device. We realize from Fig.3.6. devices moving in and out of the region affects its stability and popularity at any instant of time.

3.4.3 Problem Description

Lending computational resource to an MAC shows significant improvements in the execution performance as seen in [55], however, it also incurs additional costs. For example, as the number of devices increases in a composition, the traffic between the devices is certainly going to add extra stress to the system. Therefore, the major goal of this work is to reduce the

offloadee ($nodes\ participating\ in\ a\ composition$) choices made by the offloader ($nodes\ initiating\ the\ composition$) . The offloadees are chosen based on the composition score (C_s) computed on each node according to the factors enumerated in the previous section. As C_s increases for a device, it implies that the best available device execution takes least amount of time.

We now look at the design of the model shown in Fig 3.5.

Consider multiple jobs in set

$$J = \{J_1, J_2, J_3, \dots, J_i\} \qquad \text{.....(7)}$$

We have seen j_i be the sub-tasks generated for each Job in time t_s , as we know from Eq. 2, we have $J = n * j$. Thus, the objective beckons us to assign j_i number of sub-tasks to minimum number of mobile devices with the best possible Composition Score(C_s). This problem can be handled from the perspective of a $Packing\ Problem$ owing to the strong sense of NP-hard complexity it possesses. Simplistically, if each of the task subsets is considered as items in a general packing problem and each device to be considered as potential bins to be packed, this complexity reduces to a great extent. Our model builds around the composition score C_s and involves the following steps:

1) Based on the task subsets created for one particular job we have an ordered set of $j_{\mathrm{i}} \in J$; $J = \{j_1, j_2, j_3, \dots, j_i\}$ such that the sub-tasks are in an ascending order that is $j_{\mathrm{n+1}} \geq j_{\mathrm{n}}\ s.t.\ \{j_{\mathrm{n+1}}, \dots . j_1\}$ of size assigned on each device. Additionally, for our evaluation we also look at an unsorted set and ordering of the set in descending order i.e. $j_{\mathrm{n+1}} \leq j_{\mathrm{n}} \quad s.t.\ \{j_1, \dots . j_{\mathrm{n+1}}\}$
2) Mobile devices may be ordered or unordered according to their composition scores with scores ranging from $(1,2,3 \dots p\}$. As observed later in the performance

evaluation ordering of mobile devices give a better mapping logic as the primary purpose is to assign the quickest resource to the most intensive sub-task.

3) Assigning of the sub-tasks is done one after the other. C_s can give a broader highlight of the devices popularity, the resource capability and the stability (based on the presence of the device in the region). For instance, the device index 1 having a composition score of 7 units provides better resource than the device index 7 with a composition score of 1 unit.
4) We select a mobile device based on this C_s and make that score as the device index. In this way 'p' gives a picture of highest composition score in the set mentioned in step (2). We then check the minimum composition score (signifying highest task completion time) that is available and assign the least value in the ordered set J to that device. This means based on the composition score, the number of requests that can be taken for execution by a device can be predicted. For reducing complexity, we assume the sub-tasks taken by a device equal to the C_s index. For example, a device with a C_s index of 5 can process 5 sub-tasks.
5) Once, the minimum C_s index is found and assigned, the remaining sub-tasks are assigned to the oncoming devices with C_s Index greater than the Min_{cs} values.
6) To elaborate the algorithm further, consider a device pool of 5 with a sub-task set of 5 $j_1, j_2, j_3, \dots, j_5$. Ostensibly, the device with a minimum C_s Index in that pool would be 1 and that device is assigned 1 sub-task. Now, the second device has a C_s, say 3, this means it can take 3 sub-tasks for processing. The remaining 1 task in the set can go to one more device which means we have reduced the search space to just 3 devices.

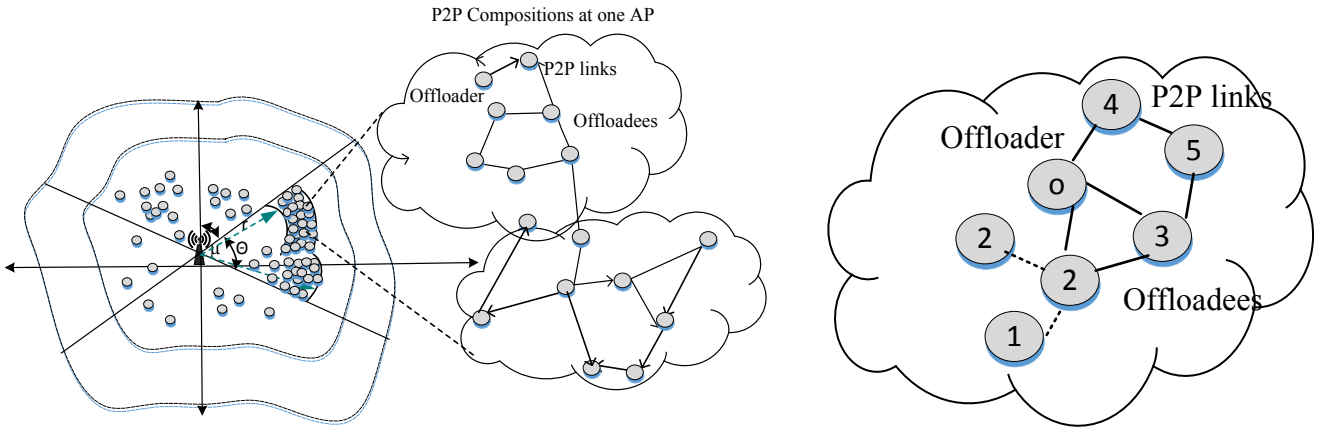
7) As we are considering scenarios where in the devices are actively volunteering , the C_s values are bound to stay above 0 owing to the stability and popularity factors of the devices in the vicinity.

3.4.4 Linear Programming Strategy

Taking cue from the composition model seen above, we now look at our main objective to reduce the number of devices needed for the execution of Task. In order to do so, we make use of Linear Programming to formulate our model. Consider task $\boldsymbol{J}$ that represents a task which needs to be offloaded. The number of devices needed to perform the execution of this task is variable. A task $\boldsymbol{J}$ can be divided into $\boldsymbol{j_{n+1}, \ldots . j_1}$ parallel sub-tasks such that $\in \boldsymbol{J}$ .Let's assume the composition formed has $\boldsymbol{n}$ nodes represented by a set N. The Objective function is to use the least number of devices available in N which can perform the task within an agreeable response time.It is governed by two decision variables, $\boldsymbol{y_j^n \; and \; x_n}$ given by ,

$$x_n = \begin{cases} 1 & ; \text{ if the node n} \in \text{N is selected to form the composition} \\ 0 & ; \text{ other wise} \end{cases}$$

$$y_j^n = \begin{cases} 1 & ; \text{ if the sub} - \text{task j is assigned to the n for execution} \\ 0 & ; \text{ other wise} \end{cases}$$



(a) Initiation with volunteer help **(b) Final deployment**

Figure 3.8: Complete Composition deployment at one Access Point

The Objective Function can be defined as,

$$min \sum_{j \in J} y_j^n * x_n * \frac{E_j^n}{C_s^n} \; ; \quad \forall \, n \in N \qquad \ldots\ldots\ldots(8)$$

Where, E_j^n is the task execution time taken by device $\boldsymbol{n}$ to execute sub-task $j \in J$.

Our primary goal of minimizing the devices comes with an implied advantage. That is a greater C_s shows a better task execution time E_j^n . Therefore, the ideal case of having C_s as infinity implicitly proves least amount of execution time.

The objective function is governed by following constraints:

1) All the sub tasks $j \in J$ should be offloaded in one shot, or it should wait for the devices to be available. In other words incomplete offloading of sub-tasks means rejection from execution. If $\eta(J)$ represents the total number of sub-tasks to be executed then it can be defined as:

$$\sum_{n \in N} y_j^n = 1 \; ; \quad \forall \, j \in J \qquad \text{……..(9)}$$

Eq. 6. Shows that for a given sub-task j, there occurs only one device where it gets executed unless there is a rejection. To elaborate sub-task rejection let's look at an example. Say n= 3, if there are three sub-tasks j_1, j_2, j_3 and j_1 is assigned to device 1 then that means it cannot be assigned to device 2 or 3 in the pool of n devices unless the device rejects the sub-tasks due to resource unavailability. We further discuss this issue while evaluating the performance of the scheduling algorithm.

2) The selected node should have the minimum available shareable resources for hosting the sub-task. The shareable resources that a device can volunteer include CPU(C),RAM(R),STORAGE(S)

   a. The device should have enough spare CPU for sharing with incoming task. This can be defined as ,

$$\sum_{j \in J} d_j^c * \; x_n \leq D_n^c \; ; \quad \forall \, n \in N \; ; c \in C \qquad \text{……(10)}$$

   b. Likewise, the device should have minimum Storage. This can be defined as ,

$$\sum_{j \in J} d_j^s * \; x_n \leq D_n^s \; ; \quad \forall \, n \in N \; ; s \in S \qquad \text{……(11)}$$

   c. Similarly for minimum RAM, this is defined as ,

$$\sum_{j \in J} d_j^r * \; x_n \leq D_n^r \; ; \quad \forall \, n \in N \; ; r \in R \qquad \text{……(12)}$$

d. The composition score of the device should be more than the threshold value. It is defined as ,

$$\sum_{j \in J} y_j^n \leq Min_{cs}^n ; \quad \forall n \in N \qquad \ldots\ldots(13)$$

e. Sub-Tasks will be hosted only on the devices which are chosen to form the composition. Hence, more generally it can be said that,

$$y_j^n \leq x_n \qquad \ldots\ldots(14)$$

In this way by prioritizing compute capacity, we find the optimal number of nodes that should be participating in a mobile ad-hoc cloud composition. We will find out how this happens in Chapter 5 when we look at different ways that show the importance of the Composition Score for modelling ad-hoc cloud composition.

3.5 Summary

A low cost and a closer placement of resources for consumers enhances end user application service experience due to a number of factors such as availability of high bandwidth links and low latency. From the perspective of realizing the IaaS paradigm in a mobile ad-hoc cloud, these requirements have been identified to be critical and beneficial due to cost improvement. Further, if the service providers are the ones who are closest to the consumer then it engenders a faster execution of application. Therefore, in this chapter we have presented the concept of Mobile ad-hoc cloud with every device playing the role of a service provider. With the proposed mobile ad-

hoc cloud framework, mobile devices that require infrastructure for processing an application are provided on demand based on the mobile application request. Additionally, the devices in the vicinity are those like-minded people who are interested in volunteering their services. Once such volunteers are discovered the Mobile Ad-hoc cloud composition can be deployed. We also handle propose a novel optimization framework for minimizing the devices who are part of the composition in order to reduce the cost incurred in terms of synchronization overheads and other data traffic related issues. In doing so, the proposed mobile ad-hoc cloud model holistically enables making available heterogeneous resources in the vicinity in a usable form and at the same time this resource composition maintains a high quality owing to the composition score metric.

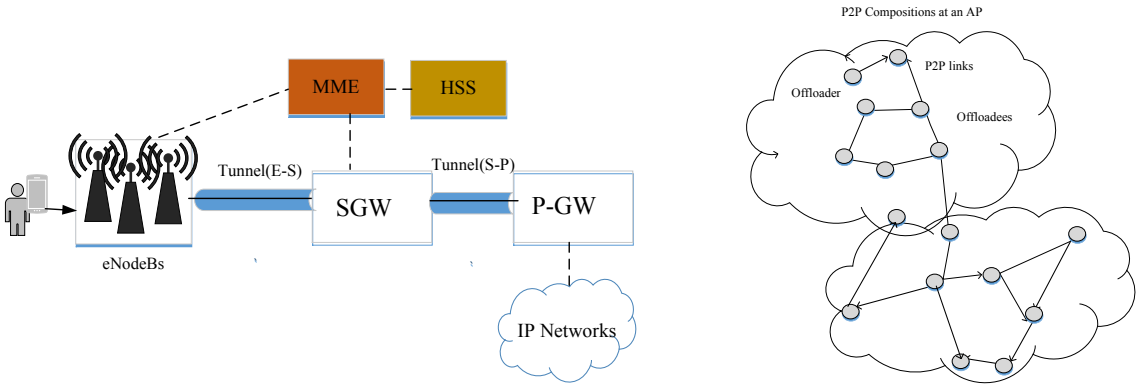
4 An SDN-assisted Disruption Tolerant Mobile Ad-hoc Cloud

4.1 Introduction

In this chapter, we describe the compelling force of Software Defined Networking technology that has empowered the wireless networks. We show some elemental changes in the architecture mentioned in the previous chapter that directly attacks the programmability characteristic of the SDN architecture and brings forth the flexibility and better manageability to the P2P composition. Further, this chapter describes how to control traffic of mobile users from one access network to the other using a network controller. We then exploit the orthogonality offered by the SDN concept and address our objective of handing mobility thereby providing a disruption tolerant mobile ad-hoc cloud that seamlessly assists the consumer while using a service.

4.2 Conflict in design of Mobile Networks and Ad-hoc networks

As we have observed in the previous chapters when the nodes begin to move based on random waypoint traces, the ad-hoc cloud composition dislodges itself and data packets drop. Now consider the P2P network discussed in chapter 3, owing to the conflicting designs of a P2P network with respect to a Mobile Network there is a need to have a rain-check on the routing mechanism in Mobile Networks. Figure 4.1 shows the conflicting designs of the two networks. A P2P network is designed to take advantage of the close proximity characteristic which enables a low latency environment. Therefore, when a consumer of a P2P composition belonging to one Access Network emits a certain packet, the routing is directed through the hierarchical mobile



(a) Design of Mobile Network **(b) A P2P network**

Figure 4.1 Conflicting designs of P2P network and Mobile Network

network. That is, the routing is directed through the centralized gateway and the user traffic moves up the levels before being directed to the closest peering device. Secondly, the innate network overheads and the delay caused in this process where the network state's need to establish connections in the mobile network renders the whole point of having a P2P application useless. Moreover, while we have observed constant packet drops due to random movements in the peering traffic, it becomes imperative that we explore ways to prioritize seamless services to a consumer and provide a good quality of experience. To address this scenario we envision that customizing the data-plane with a controller that manages the behavior of elements present therein is advantageous. To this end, we introduce the concept of Software defined Networks.

4.3 SMAC: An SDN assisted Mobile Ad-hoc Cloud Architecture

Remodelling the traditional mobile network is most definitely a costly affair. On the same note, there are intricate protocols for enabling communication and maintaining the state of the network

across various core entities like tunnels for routing information even in times of device mobility. As we see at the time of composition formation, the primary advantage of having a P2P in the first place is because of its non-reliance on the already congested network infrastructure. That is, if we consider a case like a University Access network, which is often, congested during a semester when there is high influx of student activities for video processing, application executions etc., we cannot fully rely on the existing infrastructure. Thus, making use of SDN principles with a minimal usage of the Wi-Fi access point we propose an SDN assisted Mobile Ad-hoc Cloud architecture. Essentially, the control mechanism can be deployed at the network administrator of an organisation or a crowded venue. It is our belief that the access point aggregation sites are closest to the peers who are part of the composition. This not only reduces latency but also provides a better network performance. The SMAC architecture consists of the following components:

Control Plane – There are three integral components in the control plane that handle the intelligence of the SDN controller to install appropriate flow rules into the switch to perform calculation of path, monitoring mobility and routing. It also maintains an information database that has network location of all the SDN-Wireless Nodes obtained via the Mobility Control Interface. The main component of the control plane is the software based SDN Controller.

It maintains a global view of the network consisting of several switches, Access Points and SDN wireless Nodes. The closely coupled application layer is where the routing decisions are made and in due course the mobility is monitored. The node responsible for forming the composition is known as the offloader. The offloader communicates the routing list to the controller that maintains the logical network map.

At the **Mobility Monitor** module**,** the requisite rules that are required to be followed by the data-plane are formulated. It is responsible for monitoring the whole network and primarily maintains the network topology. Every rule has an <*Action*> *and a* <*Match*> field. In-order to match against the header of a packet in a particular traffic flow, the <*Match*> *field* will be used. Once a rule is matched, the data-plane elements are requested to follow a particular action. A typical instance of managing the route of a packet with a rule < Match: {ip , src_eB = 10.0.0.2, dst_eB= 10.0.0.3}, Action = output: 6633>, shows that from source enodeB to destination enodeB the packet would be forwarded to the output port number 6633.

The **Path Calculator** calculates the best route for the packet to traverse. Based on the global view of the network, the information is used for routing traffic flows according to specific control policies. For instance, while moving from a source enodeB to a destination enodeB , say from region A to region B as seen in Fig 4.2 we look at multiple access points where packet traversal is based on an optimal route calculated by this module. Once, the composition nodes are in the data-base the controller pushes the required flows the SDN substrates in the data-plane. The users information is retrieved from the data-base every time a new request is obtained to enable offloading. This will be elaborated further while evaluating the performance of this framework.

The **Mobility Control Interface** module acts as the conduit between the data-plane and control plane. It plays the role of filling the data-base with information communicated by the offloader. That is, once a P2P composition is formed the information of the most reliable peers

are stored in the database so that the peering traffic can be detected. There are multiple ways by which devices can be discovered while forming a mobile ad-hoc cloud.

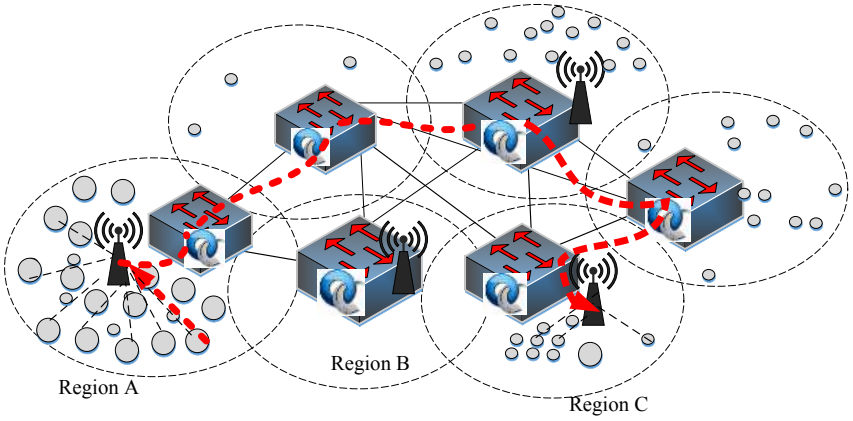


Figure 4.2 Path Computation

We believe a centralized device discovery by a controller would mean undue registration of UEs in the location irrespective of their participation in the mobile ad-hoc cloud composition. Therefore, we follow a distributed discovery of device, where in the consumer who wishes to form a cloud can do so by making a broadcast as observed in chapter 3. Fig 4.3 shows the SDN assisted framework for Mobile Ad-hoc Cloud.

Data Plane- It consists of components that work based on the rules installed by the controller namely; the backbone components (switches and radio elements) and the SDN wireless nodes. The SDN wireless-Nodes are those smart-phone prototypes that follow the principles in [35]. In addition these nodes inherit the capabilities of the virtual mobile smart phone architecture as seen in [10]. These nodes are involved in a comprehensive resource discovery mechanism such that they form a P2P composition with the available virtual phones over Wi-Fi. Thus forming a computational environment over the physical resource, every SDN wireless node has the

properties of a provider. The consumer who requests the infrastructure forms the composed participant pool.

We consider the LTE architecture radio elements namely eNodeB's (Access points (AP)) that are connected to the Open-Flow switches. Through the abstraction layer formed by the OpenFlow protocol and the device virtualization (kernel-level and user-level), the consumer receives the IaaS entities to form a composition.

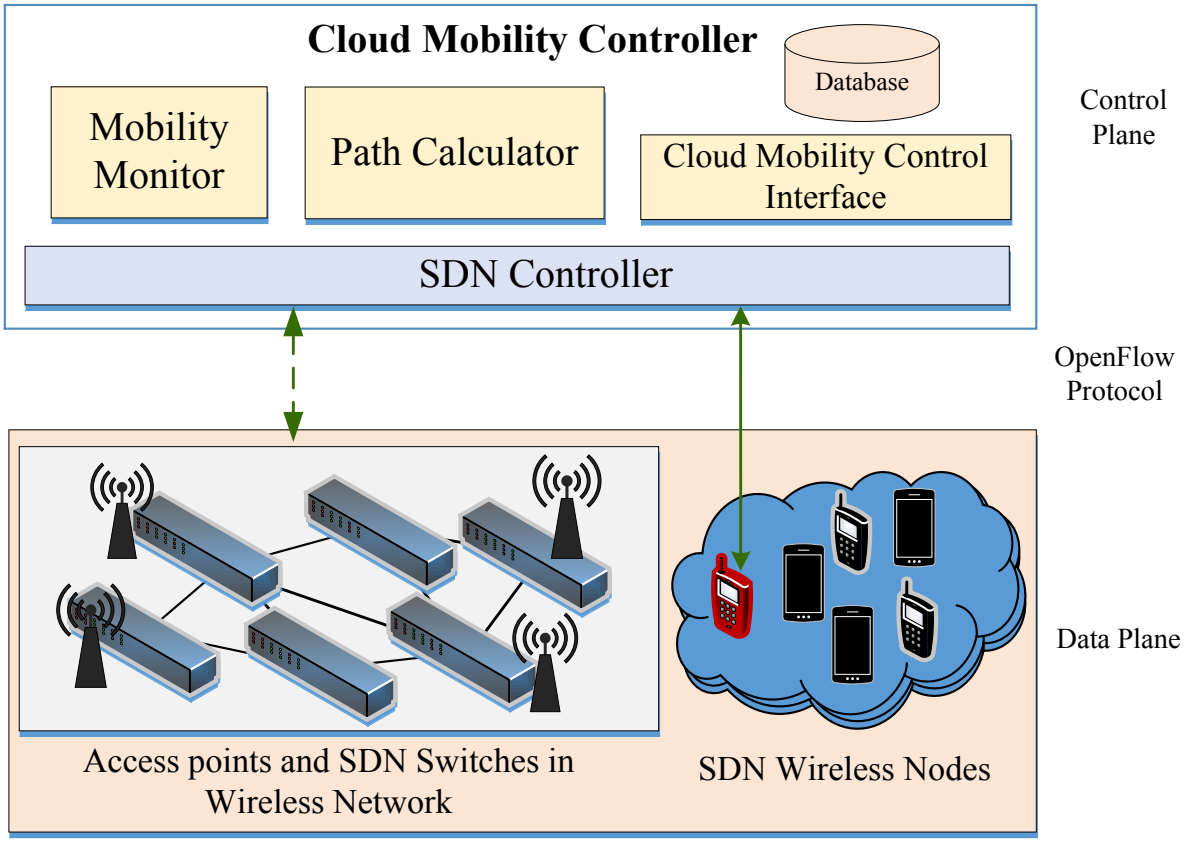


Figure 4.3: SDN assisted Mobile Ad-hoc Cloud

Thus, when the composition information is made available to the controller by the Cloud Mobility Control Interface, the database binds the entry of all the providers who are part of the composition. In due course, these SDN wireless nodes are visible to the controller. The SDN

wireless nodes maintain a routing table that helps in addressing the inter-node connections with the help of a local controlling agent with the P2P traffic detection application.

In this way, a twofold benefit is observed where in the Global View of the network map enables Mobility Monitoring and the P2P traffic is separately identified by the controller from other in-network traffic which ensures faster delivery of the ad-hoc cloud composition related traffic. Take for instance, if a node fails any SDN wireless node that is part of the composition can send an error message to the node that relies on the failed node's connection. Thus, the primary feature of an ad-hoc network is maintained by updating routes between source and destination assisted by intermediate nodes.

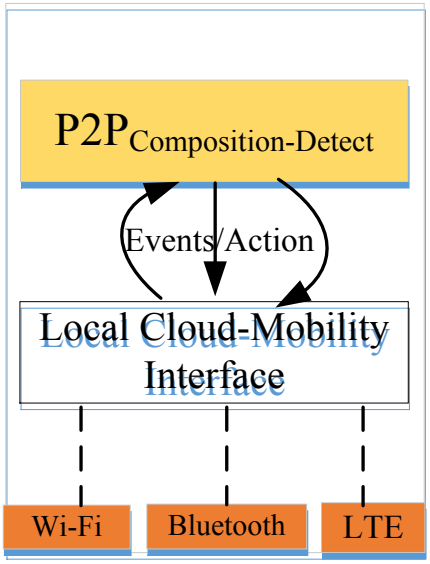


Figure 4.4 Local SDN interface

Figure 4.4 gives the functionalities of the SDN wireless Node, as seen in the example above the local $P2P_{composition\text{-}detect}$ application takes charge of the routing table. All the peering traffic is detected here by which the local Cloud Mobility interface informs the Control Interface. In doing so, the controller component will install or notify the switch to update flows. The central controller component primarily processes the events sent by the local interface that enables a

hierarchical communication. In this way, being the end point for an ad-hoc computing service the SDN wireless node will run its Local Interface so that the central controller can detect that node as well as the traffic that the node emits. This adheres to the primary features of the forwarding plane element at the same time allows the nodes to have a self-organizing network.

4.3.1 Designing an OpenFlow Based Wireless Network

Application of OpenFlow protocol to wireless networks essentially brings in a periodically communicating environment that makes the data plane adaptable to the topology changes. For example, in a completely mobile environment, it becomes imperative that the consumer nodes and the provider nodes both move. The adaptability offered by the OpenFlow enabled wireless network is that one can easily homogenize a heterogeneous environment consisting of disparate devices with varied mobility as the network states are hidden by the protocol.

In order to design an OpenFlow Enabled wireless network, the switches defined should be OpenFlow compliant. That is various functionalities like traffic management and measurement, network policy management like delivery flexibility in path changes to name a few are already provided in an OpenFlow switch. There are a multitude of trade-offs while defining the capabilities and functionalities of a switch. It is of significance to understand that in traditional mobile networks due to the ever increase in the number of consumers and devices moving there occurs a plethora of scalability issues along-with related mobility issues that act as bottleneck. As the primary purpose of this research is to identify the grounds where-in a seamless ad-hoc cloud composition can be maintained, our main focus is inspecting ad-hoc cloud composition related traffic and prioritizing such traffic to realize a disruption tolerant mobile ad-hoc cloud.

At the Control Plane, apart from maintaining the global network view, the applications that are built uphold the responsibility of listening to the events coming from the data-plane and installing appropriate forwarding rules in the switch. Further, the local interface in the SDN wireless nodes engender a self-organizing fall back mechanism in times of central controller failure. By doing so it our endeavour to not only reap the benefits of the mobile ad-hoc cloud composition but also augment the existing congested infrastructure to perform better. With all the aforementioned facts we now look at how Mobility is managed using Software Defined Networking principles.

4.3.2 Mobility Management using Software Defined Networking

As observed in chapter 3, the offloaders discover the offloadees for composition following the P2P composition metrics defined there-in. Once the composition is formed however, owing to the very nature of ad-hoc environments, the offloadees and the offloaders cannot be restricted to a particular location. Consider the Fig 4. where one of the devices that's part of access network belonging to Region A begins to move out and strays over to Region C access network. Irrespective of whether the node moved is an offloader or offloadee, the composition would dislodge itself rendering the entire composition process useless. Further, even minimal movements inside the region would also show a likelihood of packet drop in a legacy network as shown by researches [56] in the literature. Thus, there needs to be a proper mechanism for handling roaming of devices in a network and that provides end-to-end connectivity at all times. Ostensibly, there are plenty of mobility solutions available that incur high infrastructure related costs and do not integrate well with the routing process while considering mobile content [57].

We believe it is not only tedious but infeasible to follow those solutions and rather look for ways in which OpenFlow switch installation close to access points or even plausible aggregation sites can coexists with legacy routing process and assists in network abstraction. Further, OpenFlow has a secure channel established between the control and the data planes.

Thus, if we consider the same case as mentioned above, when a node moves out of a region the dynamics of the EPC is augmented by the veil of the controller. That is, in a conventional scenario, the mobile network follows a level by level EPC construct in a way that a node sending a packet traverses to the Packet Gateway via the eNodeB and the Serving Gateway.

The Mobility Management Entity (MME) exchanges periodic signals with the moving node and selects the Serving Gateway (S-GW) and Packet Gateway(P-GW) that serves in the node. Therefore, as the nodes move about incessantly new tunnels are established between elements that allow data transfer and assists in user mobility by encapsulating a packet at the source and decapsulation at the destination. The GTP or PMIPv6 tunnels are the legacy IP mobility providers as observed in literature; however it is our endeavour to work our way together with the present networks and SDN instead of modifying the state-of-the-art protocols or substituting OpenFlow protocol for a legacy protocol. It is observed that the pros and cons of substituting a new protocol in the mobile network will have high economic impact due to reasons that are totally inclined towards the infrastructure price. That is, re-modelling the infrastructure could per se point towards high economic infeasibility and purely defeats the purpose of utilizing the SDN technology to augment legacy networks by bringing flexibility to it and also empowering future networks.

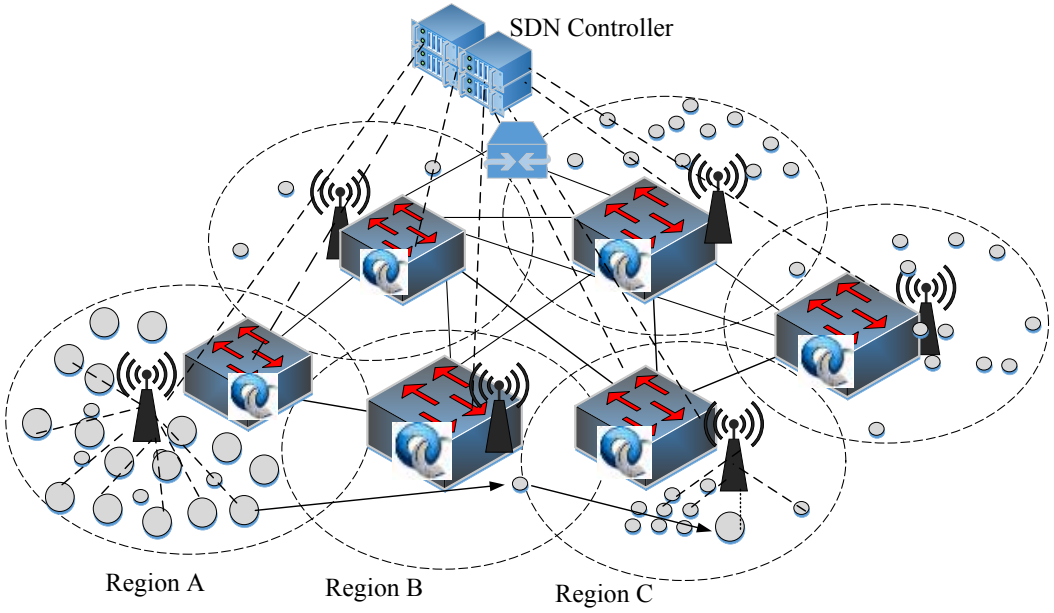


Figure 4.5 Mobility Management

To this end we realize a framework that adapts to the constantly changing ad-hoc cloud environment. Fig 4.1 and Fig 4.6 shows a typical scenario of how data transfer tunnels are formed at the time of handover. We will see in the following chapters how enodeB's (access points) are where the OpenFlow switching fabric is deployed. Studies like [37, 38] follow aggregation of access points , the enodeB's could be aggregated at different locations based on the deployment like a Mobile Telephone Switching Office (MTSO) . Once a node joins a network and begins sending and receiving packets, the tunnels are established between the gateways(S-GW & P-GW) and between enodeB's (or enodeB aggregation) with gateway (S-GW) based on the GTP protocol. A packet is encapsulated and passed through a tunnel, the tunnel Identifier or TEID definition in the GTP-U protocol is set to a particular destination. This destination information is put to use with the intervention of the SDN controller for enabling a seamless handover.

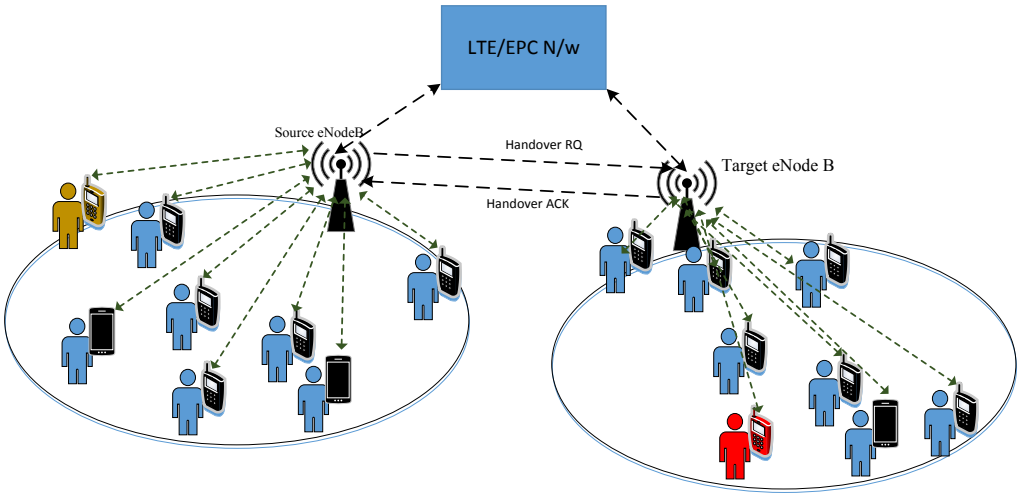


Figure 4.6 Interaction between users and the EPC component

As we know the controller has the visibility of the entire network, while providing a service, our major goal is to tolerate any kind of disruptions during offloading at the time of user's movement. While considering a specific handover scenario, like a node moving from Region A to Region C, the enodeB(source enodeB) at Region A decides to handover the node to the enodeB at Region C (target enodeB). In doing so the target location sends an acknowledgement to the source thereby switching the path of forwarding location. This acknowledgement is the handover ACK message that has the target location address like the tunnel ID, IP address etc. which will be used by the serving gateway to forward downlink packets. The mobility monitor will take note of the bearer information and constructs a new rule with a changed destination address action to be taken for the newly arrived event. In this time, the MME is notified of this change in path and the downlink packet to the target is updated.

The service offered by this framework does not measure the relevance of the signalling overheads of existing GTP approaches as the purpose of this model is to identify the ad-hoc cloud composition traffic and provide the consumer with the best possible quality of service. In

order to further evaluate between the need for a substitution of OpenFlow for GTP, it becomes imperative to closely understand the implications of this remodelling.

4.3.2.1 OpenFlow as a Substitute for Legacy Protocols: Second Approach

We have understood how OpenFlow can coexist with the GPRS tunnelling protocol, however, many studies have shown how OpenFlow can act as a substitute for the same. That is, the entire tunnel management, path switching management and other such related processes are handled only by OpenFlow protocol.

One such approach is MobileFlow [39], in this work authors begin to take the existing SDN concept of splitting the planes and have directly imposed it on the Mobile Network architecture. In doing so, they propose a MobileFlow Controller and a MobileFlow Forwarding Engine. Owing to its nomenclature, the controller handles all the EPC component related management and has the visibility of the entire network whereas the MobileFlow Forwarding Engine are software powered simple forwarding devices.

In Softcell[59] all the EPC components are changed to OpenFlow switches and instead of using the traditional GTP based approach, a tag based approach is followed and forwards traffic based on a hierarchical approach. In [60] authors follow a hybrid approach by replacing EPC components with switches and routers but use the MME features as is for handling signalling messages. In [61] authors propose another model that analyzes the separation of control and data plane with the use of extended OpenFlow protocol. It bears similarities to the MobileFlow and Softcell architectural strategies but has added functionalities of improved management by extending the OpenFlow protocol.

In [62] in order to address user mobility at the same time manage traffic between EPC elements using SDN principles. Here authors centralize the EPC control plane for addressing three procedure which they deem as the most integral procedures, these are:

1. Initial Attach – This is when a node joins the network to receive the services which involves authentication, registration and EPS bearer establishment. Once registered the node gets it's IP and the network location of the node is fixed.

2. Native technology handover- This is the X2 based handover technique with a relocation of S-GW. Here the node moves between enodeB's which are connected to different S-GW.

3. Inter-Mobility Management- This procedure consists of deletion of the PMIPv6 tunnel between previous serving and packet gateways and establishment of new tunnels between the two gateways.

However, this model bears resemblance to [59] in terms of addition of OpenFlow compliant SDN switches except for the addition of a centralized EPC controller for maintaining a global view of the complete network topology.

In general, although the aforementioned architectures provide the same flexibility and adaptability of the SDN technology it imposes heavy economical overheads for redesigning the existing infrastructure that are infeasible to a large extent.

Therefore, as these approaches mean complete remodelling of the existing design, we are motivated to provide a framework that is feasible and at the same time it is our endeavour to ground ourselves with an approach that calls for coexistence with the legacy features and proves beneficial to the posterity. Moreover, the very essence of SDN makes it responsible to enable an efficient framework for managing traffic. Hence in our work, we identify this traffic that needs

assistance from the management principles of SDN and augments the existing ad-hoc cloud performance.

4.4 Ad-hoc Cloud Composition Traffic Identification

As seen before once the ad-hoc cloud composition is formed by the offloader and notified to the controller, it becomes the sole responsibility of the controller to inspect every event that is being emitted in its purview. Moreover, learning the topology not only means the central controller has to make intelligent decisions, but it also means to keep listening to the local controller via the wireless Node interface. This means, if a node moves out of the location to another region, the wireless node has to detect this change in the ad-hoc cloud composition and update the SDN controller. For instance, if we consider multimedia application being used by a set of students in a university for processing a certain video and one of those in the composition moves out to another location, it becomes imperative that the video should be still processed by the node that has moved to another region inside the campus.

Consider the aforementioned scenario in the traditional mobile network, the P2P application traffic will have to traverse through different levels in the EPC before reaching the peer who is part of the P2P composition [63]. This in many ways degrades the need and purpose of forming a P2P composition in the first place. The primary benefit of having a closer-to-user P2P composition is to avail the low cost computational environment as well as reducing the time of getting the resources required for application processing [64]. Therefore, we leverage SDN concept to chalk out the P2P traffic and offload the same without going through the sub-optimal routing in traditional legacy network. This selectivity not only allows a low latency scheme but also keeps the composition intact throughout the service period.

As we are looking at a highly varying ad-hoc cloud composition, nodes may join and leave the network at any time, therefore, an important aspect of self-configuration needs to be addressed in-order to maintain the autonomy of the system. Moreover, every node has to have information of the peers who are becoming part of the composition in-order to perform topology specific routing and forwarding. Additionally, as the global view of a topology is maintained by the controller, a seamless disruption tolerant mobile ad-hoc cloud creation is inevitably possible.

4.5 A Disruption Tolerant Mobile Ad-hoc Cloud

For the testing purpose of the mobility we define three test cases that encompass an exhaustive study to show the compelling influence of Software Defined Networking principles. We have observed before the resilience of the composition will only allow for a seamless composition. Therefore, first let's look at the simplest procedure of two access networks that are the region of interest for the mobile nodes. We model the scenario considering two laboratory environments that are at a distance of five to ten minutes' walk from each other. In order to define their locations we make use of a 2-D Euclidean plane. In a practical scenario it could be an irregular ad-hoc environment, however for simplicity we consider a regular shaped network environment. As seen in Fig 4.6, the initiation happens at access point AP_a. Here, the offloader begins requesting for a composition as observed in chapter 3 and composes a homogeneous resource entity with the help of the volunteers in the vicinity to serve himself. Now, in a typical ad-hoc environment, the traffic inside the same access network can be easily maintained by the

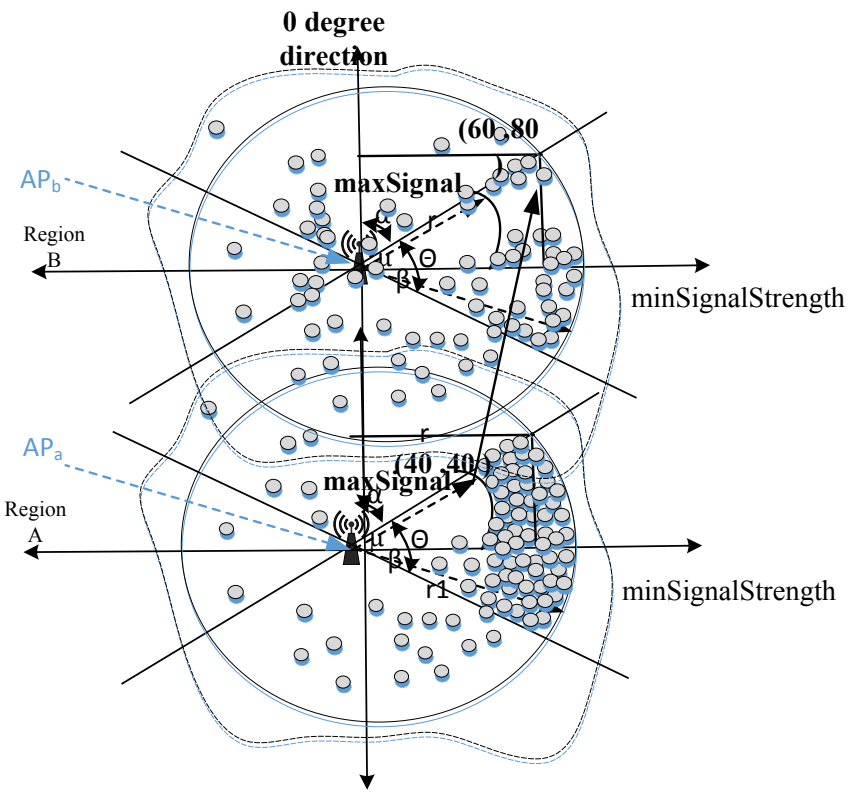


Figure 4.7 Test-Case 1 between two Access Networks

SDN wireless node module, however, as one of the wireless nodes moves out, to AP_b in region B, then the connection traverses through EPC tunnels. SDN controller is notified of the movement and the rules are preconfigured with the information from the initial packets of the connection with the initiator. At this point, the flow rules are installed into the switch and the data-plane elements are now aware of the P2P traffic.

The two fold benefit here is a seamless connectivity to the composition and separation of the P2P composition traffic from any other kind of traffic ensures lower round trip time. That is, every time packets from the composition are emitted and visible to the controller irrespective of the movement of the offloader or offloadee P2P packets will be redirected based on the associated

node destination. As seen in Fig 4.6 relevant information about the node's network location and associated source/destination information can be evaluated with GTP headers that is used by the EPC in the traditional mobile networks.

Recalling Fig 4.6, the network location of the node is assumed to be at the coordinates (40, 40), the mobility pattern is pre generated and the composition is initiated at the access point AP_a from where the node in question moves to a network location (60, 80) at access point AP_b. In this process, as soon as the node begins its transition, using the P2P information that the controller has is used to form matching rules. Typically a rule consists of processing priority, expiration time, counters and list of actions. The actions are output port information where the packets can be received. Apart from this a pattern field consists of information that states which packets belong to a particular flow and can comprise header values.

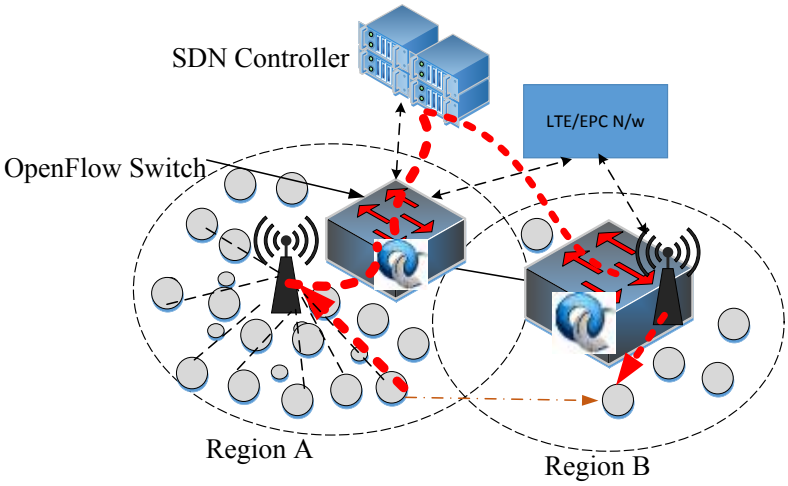


Figure 4.8. P2P Composition Traffic Management

Hence, if a packet that belonging to the traffic that has matching rules is found, then based on that an action to offload, redirect or just to forward can be performed. As our main goal is to cater to the service , only the traffic belonging to P2P composition is checked and sent to the desired location. In this way, instead of following the hierarchy of the EPC the composition service tolerates any kind of disruption. Fig 4.8 shows P2P composition traffic management using SDN. In the second case as seen in Fig 4.7 the initial case is extended to traverse to another access point AP_c , in-order to test the resilience of the cloud when there is continuous movement of the node between three access points in a region like between laboratories on a floor.

Therefore, the node that has moved till the coordinates (60, 80) is moved to the coordinates (80, 80).

In all the cases, the signal strength of an access point is considered to be ‘r’ for the max signal

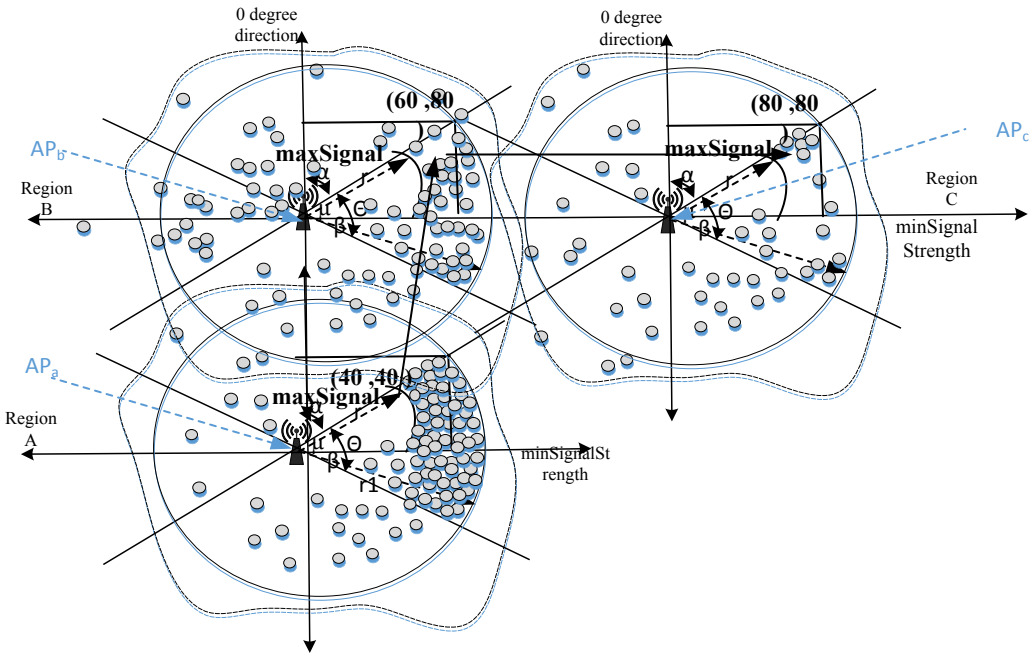


Figure 4.9 Test-Case 2 between three Access Networks

strength and as the node moves away from the point it gradually reduces to ‘r1’. The direction of motion is measured anti-clockwise from the horizontal axis. The vertical axis represents the 0 degree direction and the horizontal axis shows the reduction in signal strength as the node moves away from the centre. Fig. 4.9 is the case of back and forth motion. This is the case of computation being offloaded when a node moves out of the controller’s purview. That is, when a node starts computing a sub-task and moves out due to area limitation, it loses its connectivity to the composition itself. However, the controller maintains the node information in its database so that whenever the nodes comes back to the area, it can easily join the composition without any need to request for another composition link. Therefore, unless the offloader does not disband the composition, it maintains the ad-hoc cloud and a continued service is provided. Further, if the node computation is finished, the node can send the results back to the offloader whenever there is a possibility of a second meeting with the composition before the computation deadline.

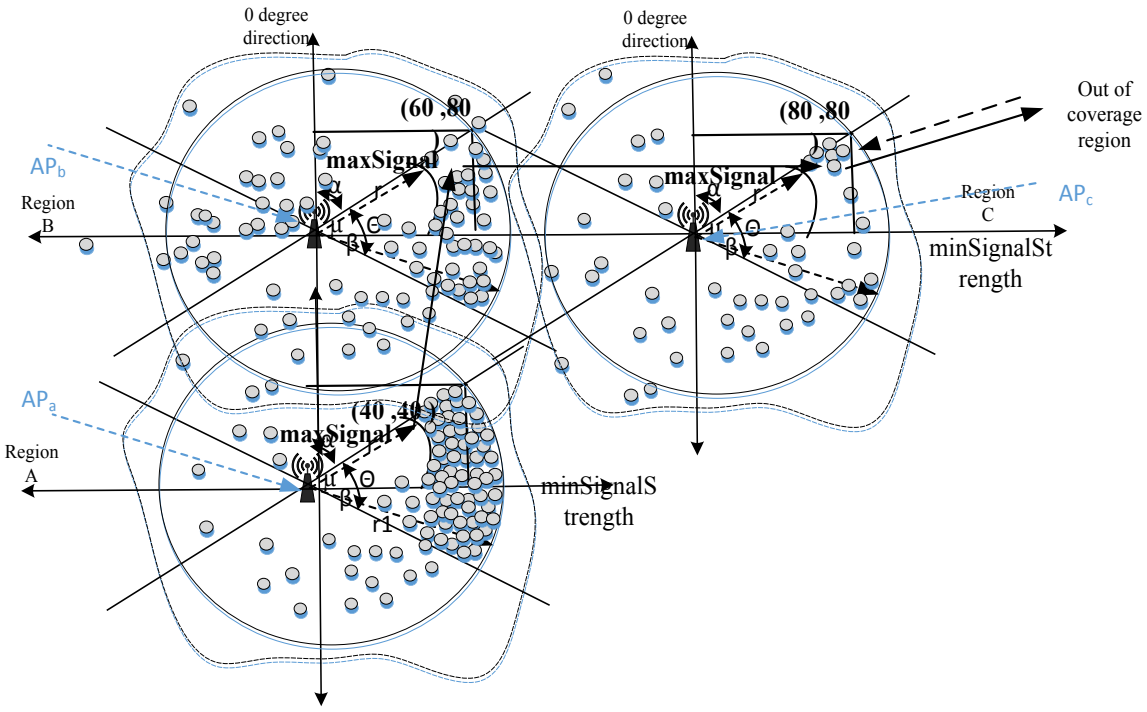


Figure 4.10 Test-Case 3 back and forth motion of a computation node

In both mobile handover and P2P composition maintenance , after a change in the network status, OpenFlow controller immediately deletes invalid matching flow rules in respective switches and install new flow rules instead of reactively processing flows to forward traffic between the mobile device and the ad-hoc cloud composition and vice versa till P2P traffic decision is taken. This way we could minimize controller load created by reactive flow processing until a decision is made. The pre-configured controller rules are left as it is within the switches. On the other hand, if service needs to be abandoned, the offloader can notify the controller to release the composition devices.

4.6 Summary

In this chapter we looked at how the entire SDN framework affects the mobile ad-hoc cloud composition. The centralized controller upholds the responsibility of monitoring the network wide topology, path installation and communication with the data-plane elements for P2P composition detection. We discuss the pros and cons of having the OpenFlow protocol co-exist with the legacy network to economize and provide a plausible solution for maintaining a disruption tolerant mobile ad-hoc cloud service.

Network control was done by accessing OpenFlow Controller at each network using the OpenFlow protocol that works well with existing legacy protocols. The very essence of OpenFlow protocol is brought out that demonstrates how the limitations of traditional networks can be overcome in terms of providing a lower latency.

The three test cases show how the network controlling component of the centralized controller identifies each network switch and other elements in the data-plane. These cases elaborate the dynamicity of the mobile ad-hoc cloud composition and suggest that roaming in a location barely matters if there is a controller monitoring the mobility of a device.

We see how maintaining a list of objects by the network controller is instrumental in the addition and removal of flows individually. After detecting events at the data plane, the network controller generated flow rules per switch based on the type of the event. In our case we have preconfigured flow rules for detecting usable composition traffic that can be separated from other traffic for faster delivery to the nodes in the composition. Different cases highlight different viewpoints and assists in exploring a variety of parameters as would be seen in the following chapter.

5 Performance Evaluation

5.1 Introduction

In this chapter, we present a complete analysis of the performance and behavior of the mobile ad-hoc cloud service. Initially, we describe the resource discovery and composition formation by showing how devices in the vicinity respond to a broadcast request individually. We then look at the composition formation, and optimization of device selection in an MAC. Eventually, we show that the seamless mobile ad-hoc cloud service orchestration can be maintained with SDN in their local networks. Then we analyze the outcomes of three use cases running in the user dedicated VM in different scenarios juxtaposing the usage of minimal devices in a composition and offloader movements.

5.2 Mobile Ad-hoc Cloud framework

To utilize the available resources in the composition, a consumer must submit a request to get the service. In the use case, the user application is considered to be a dummy task which is offloaded, however it could be any executable application. An option to request IaaS service through the IaaS request interface is realized. We evaluate the performance of the IaaS algorithms with the offloading of an application. The entire code is written in python. Consider a constant IP for a session. Two environments that support python well- Network Emulator for Mobile Universe (NEMU[65]) and Mininet[66] are used for emulation. Two of these environments are considered to create a heterogeneous space where the algorithm can be tested.

In NEMU environment, all nodes (VMs) of differing sizes are used for creating windows phone replicas. P2P network emulation within an environment height and width of 1000x1000 with small step changes every 5 seconds is used for modeling an ad-hoc network. VM images modeled between 256 MB to 512 MB with 1 CPU, that act as 12 nodes embedded onto NEMU environment. These 12 nodes are connected to an Access Point as shown in Figure 5.1. We first consider a scenario with one AP to avoid the case of congestion in local networks. The procedure starts from the initiator (host/offloader), first

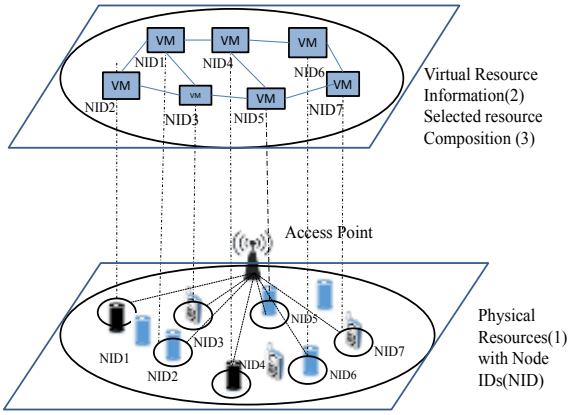


Figure 5.1 Composition and Management

discovering the devices and composing the VMs using composition algorithm. This is followed by routing and management done with the help of keys generated based on rendezvous consistent hashing.

These keys route the dummy tasks to nodes. Thus, the experiment starts first by offloading to 2 nodes, then to 4 nodes gradually, offloading is done to 12 nodes to check for performance with an increase in the number of nodes. As increasing by a single node was not producing any visible difference in performance, nodes are increased by a factor of 2. These nodes behave as providers (volunteers/offloadees). The composition of P2P network and its

routing & management is done with the IaaS algorithm (Algorithm 1). The sequence of the function call is shown in Fig 5.2. A bootstrap construct is sent to the provider nodes (line 8-10).

The resource information of Node ID, IP, and port from listener nodes (line 5) is used later to form a P2P network. For one request, consider an example of a meta-file formed with the resource information obtained assimilated along with the job information. This meta-file is the (that is stored in the key-value storage) value for keys generated (lines 3 to 7).

Algorithm 5. 1 : IaaS Composition and Routing Algorithm

```
1:Input  msg,T,msg1,host,,x,arrival, receive, key,value,ip,port,id
 //Composition initiation→resourceDiscovery(msg,T) by send(msg,t) "broadcast message with
   task  information with time"
 ninfo ← nodeInformation()
 sendto(msg1,ip)   > msg1 "Session establishment message"
//Maintain a database of resources :resourceDict[addr[0]]
2:begin Listener (nid,ip)
//once a ready offloaded files are queued and bootstrapping follows; btI← bootstrapI(id,initiator)
queue.poll () with parameters N, arrival and message   received
update() updating the array resourceDict[]
3:send(msg,t)          >msg " resource information within time t)
//for each meta-file a hashed: initiator.set(key,value) is followed; function hashMetafile(st,f) is
initialized
4: while (ninfo=True):
5:     data,addr←ninfo
6: end while
7: begin Session()
// Routing and Coordination inside →composition is made initProtocol←InitializeDARC()
8:    Exchange session acknowledgement
// For offloading use-case this protocol is initialized calling bootstrap();initiator behaves as
server <-initiator.bootstrap [(ip,port)].
9: for all t > T do    // t is the time of nodes arriving earliest, x  is the late replies.
10:      calc = t+x
11:   { N,arrival,receive}←queue.poll()
12:    updateInitiator()
13:     send(msg,t)    > with acknowledgement message
14:  end for
15: end
16:end
```

We evaluate the composition algorithm after the initiator discovers the VMs through IPs established for a session. During the experiment, we model dummy tasks which are small files (less than 100kb). These are used for offloading to have heterogeneous traces to engender a practical scenario. A total of six traces of differing sizes were used. The graph is normalized to local execution (value 100). The time is measured during the get-result request phase because getting back the serviced results would alone ensure completeness of the process.

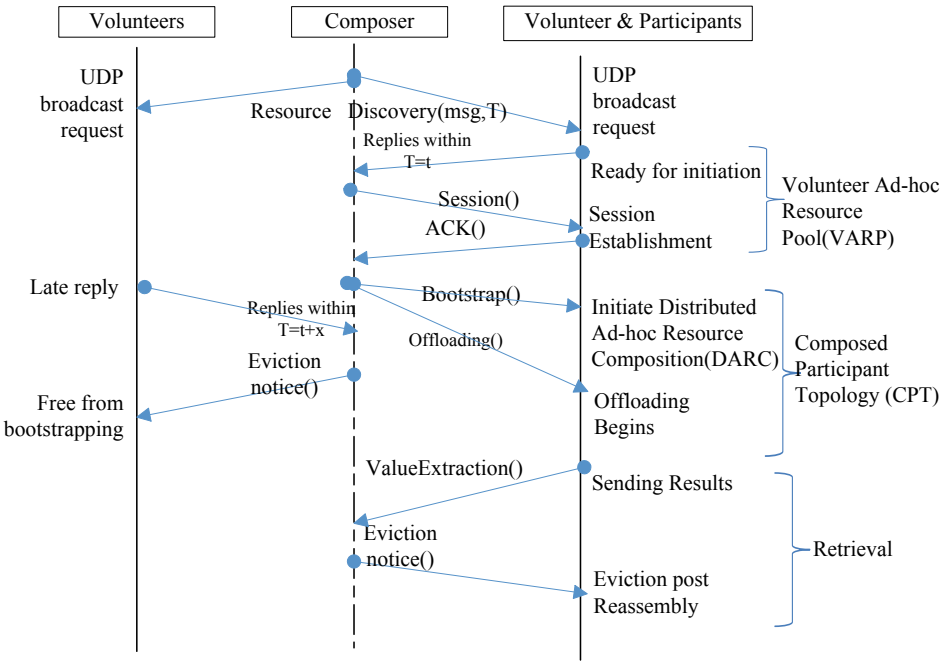
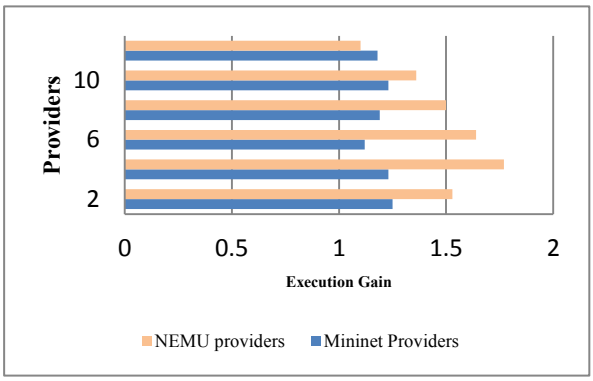


Figure 5.2 Sequence Diagram of Composition Algorithm

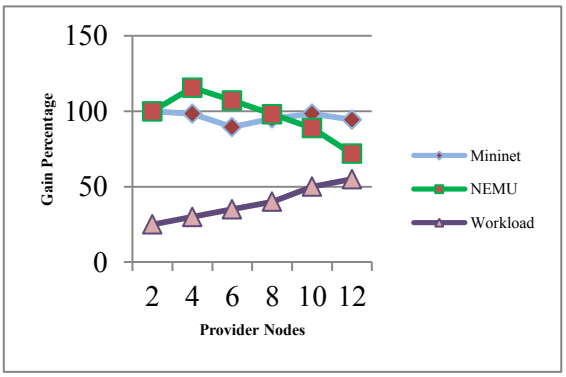
The performance results are shown in fig 5.3. Both NEMU and Mininet show approximately the same realism.

5.2.1 Evaluation of Mobile Ad-hoc Cloud framework

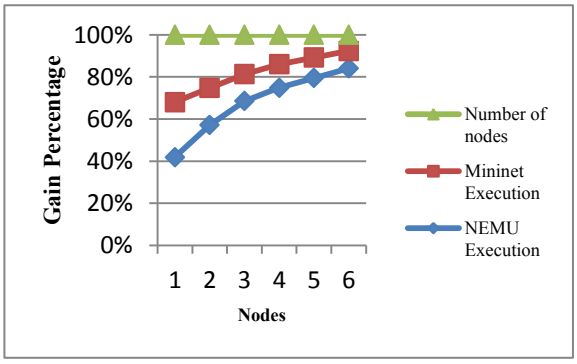
The results show that as the number of nodes goes on increasing for a particular task, the increase in performance stagnates after a point. It suggests that as overheads go on increasing (network overheads, lookups, and creation of P2P network, device overheads etc.) performance attains a stage where there is no increase or decrease but maintains the same level for the same task.



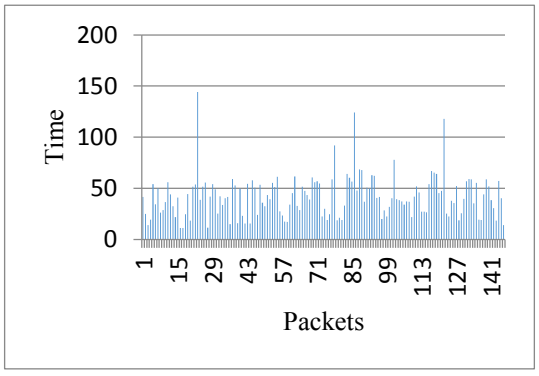
(a) Performance of both environment



(b) Degradation in performance



(c) Performance stagnation once the nodes are pooled



(d) Packet drop due to high mobility

Figure.5.3. Performance Analysis of Mobile Ad-hoc Cloud framework

This delineates the addition of overheads to the execution of tasks, which in turn causes the saturation. While offloading to 10 nodes i.e. going from 8 to 10, this change in slope of the graph is observed. It is observed in both the environments execution is better while offloading in most cases. In Mininet environment we model VMs mimicking windows phone replicas. The number of nodes is gradually increased same as before where dummy tasks are offloaded. Based on the work done by Canepa et al. [45] it can be learned that for small files using servers like Hadoop would degrade performance. It also provides an insight to the resource usage.

Pooling of resources, when not needed, results in performance degradation. Thus, resource usage should be based on the task's need. Assuming the preparation and offloading time in [45] together as the workload offloading time in our case, we observed a better performance in our system. The Hadoop server performance in [45] could be mapped to the degradation in performance observed after 10 nodes. In an ad-hoc environment, extra resources could be used by some other customers, also blocking more than the requisite resources means wastage of resources.

There were failures while offloading that caused over 20% packet drops. For one, during Offloader's device error the initiating host starts the process of offloading and does not set ("key", value) a value (meta-file). That is the provider bootstraps or the provider peer which is ready for processing the tasks keeps waiting but doesn't find a value. Secondly, when Device shuts down post offloading - In this scenario, the providers who are bootstrapped in the P2P network but do not respond once the host device has stopped. Lastly, during Late Replies the provider could be part of another consumer's ad-hoc cloud. However, once offloading begins then there will be another look up initiated for processing the task, if at all the host device is not

able to process the sub-task locally. Table 5.1 shows the simulation parameters used in the experiments.

S.No.	Parameter	Value
1.	Area	1000x1000 sq.units
2.	Channel capacity	2 Mbps
3.	Transmission Range	250m
4.	MAC Protocol	IEEE 802.11
5.	Packet Size	Upto 100Kb (6 different traces)
6.	Algorithm	Distributed Adhoc Resource Composition(DARC)
7.	Node Speed	0.5-4 m/s

Table 5.1. Simulation parameters

Noticeably, the pool of devices is formed first to observe composition formation at one access point. Moreover, what needs to be understood is if we can reduce the devices in the pool so that the sub-tasks can be processed faster some of these failures, if not all can be addressed. In the following sections, it becomes our endeavour to address these failures and improve the system performance. Firstly, we model a task scheduling algorithm prioritizing the node compute capacity in the next section.

5.3 Task Scheduling in a Mobile Ad-hoc Cloud

We follow. a strategy that prioritizes the node computing capacity to engender the best possible Mobile Ad-hoc Cloud environment. As it follows a general Packing Problem, we

strategically model the Bin Packing Algorithm to suit our need. In Algorithm 2, (line 1-2) the C_s sub-routine calculates the composition score. Then, (lines 4-11) after ordering the sub-tasks we perform task assignment. In (lines 23-30) the ordering of the sub-tasks is done based on an ascending fashion that is $j_n \leq j_{n+1}$ However, in order to evaluate the effectiveness of our

Algorithm 5.2: Task Scheduling using Composition Score

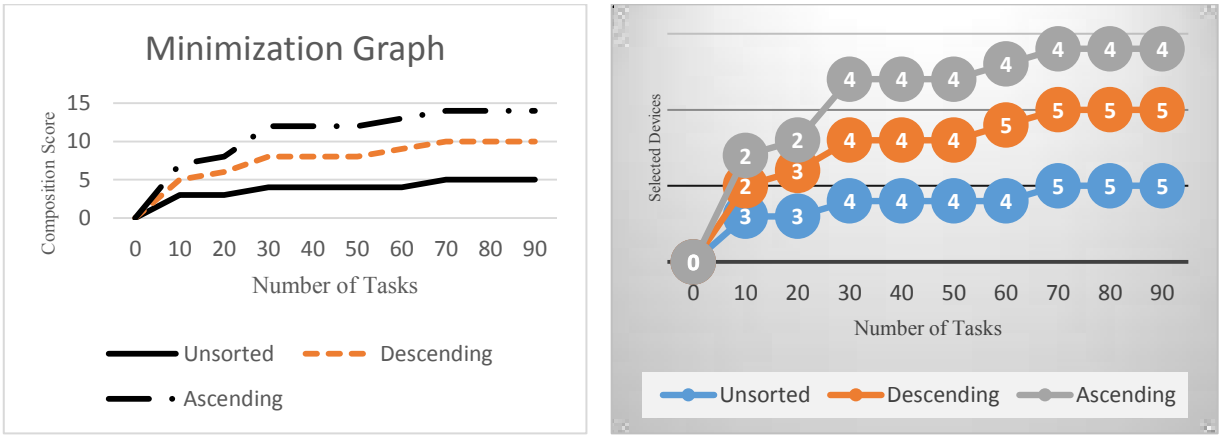
1: **procedure** COMPCAL // Calculate the Composition Score
2: **end procedure**
3: **procedure** UNSORTED
4: **Input**: *Jun* = {ji, *ji*+1.., *jn*}
5**: //** *Jun* unsorted list of sub-tasks
6: **Input**: $D = \{c_s^n \ ; \ \forall n \in N\}$
7: **Output**: list of devices selected for sub-task execution
8: **for** *jn* in J do
9: **while** D not empty and J is not empty do
10:
11: **if** Compcal(*jn*) $\leq C_s^n$ then
12: n ← *jn*
13: // mapping subtask *jn* to device n
14: else
15: Go to next device in the list *D*
16: **end if**
17: **end while**
18**: end for**
19: **end procedure**
20: **procedure** SORTINGASCENDING
21: **Input:** *J* = {*ji, ji*+1.., *jn*}
22**: Input:** $D = \{c_s^n \ ; \ \forall n \in N\}$
23: **Output**: list of devices selected for subtask execution
24: J = SortSubTaskAcc(*Jun*)
25: // Sorting Ascending order of Composition Score
26: i.e., C_s^j (1) $\leq C_s^j$ (2) $\leq C_s^j$ (3) $\leq .. C_s^j$ (*n*)
27: **for** *jn* in *J* do
28: **while** D not empty and J is not empty do
29:
30: if Compcal(*jn*) $\leq C_s^n$ then
31: n ← *jn*
32: // mapping subtask *jn* to device n
33: **else**
34: Go to next device in the list *D*
35: **end if**
36: **end while**
37: **end for**
38: **end procedure**

system we adapt the methodology in [51]. In Algorithm 3, our main sub-routine performs a similar computation, however, the desired structure of the sub-task set is $j_{n+1} \leq j_n$. Lastly, from line (6-14) we model the Bins as the devices in a composition and check for the C_s values before the assignment of the task. Now, we evaluate these algorithms. In order to configure the sub-task traffic and generate our topology at the same time we make use of a simulator built in-house to satisfy the experiment settings. The simulator is written in python. It is run on a computer with i5 processor, 8GB RAM with 200 GB storage. All integral modules are written in python including Sub-task traffic generator module and the topology generator. The topology adheres to the random Ad-hoc behaviour, wherein the nodes enter the composition and leave a composition at random instance of time. Using math module available in python, composition score of each node is calculated. Gurobi optimizer is used to calculate the solution for the proposed linear programming model owing to these libraries in python.

Algorithm 5.3: Adapted Bin Packing

1: **procedure** SORTINGDESCENDING
2: **Input**: *Jun = {ji, ji*+1*, .., jn}*
3: **Input**: $D = \{c_s^n \ ; \ \forall n \in N\}$
4: **Output:** list of devices selected for subtask execution
5: J = SortSubTaskAcc(*Jun*)
6: *//J* sorted list of subtask and Sorting of devices in Descending order of Composition Score
7: i.e., $C_s^j \ (1) \geq C_s^j \ (2) \geq C_s^j \ (3) \geq .. \ C_s^j \ (n)$
8: **for** *jn* in *J* do
9: **while** D not empty and J is not empty do
10:
11: **if** Compcal(*jn*) $\leq C_s^n$ then
12: n ← *jn*
13: *//*mapping subtask *jn* to device n
14: **else**
15: Go to next device in the list *D*
16: **end if**
17: **end while**
18**: end for**
19: **end procedure**

The descending order of C_s can be mapped to the ordering scheme in [51]. In Fig. 5.4 (a,b). we look at the number of devices that have been taken into composition from a pool of devices.



(a) **(b)**

Figure.5.4. Device Minimization

5.3.1 Evaluation of Task Scheduling Algorithm

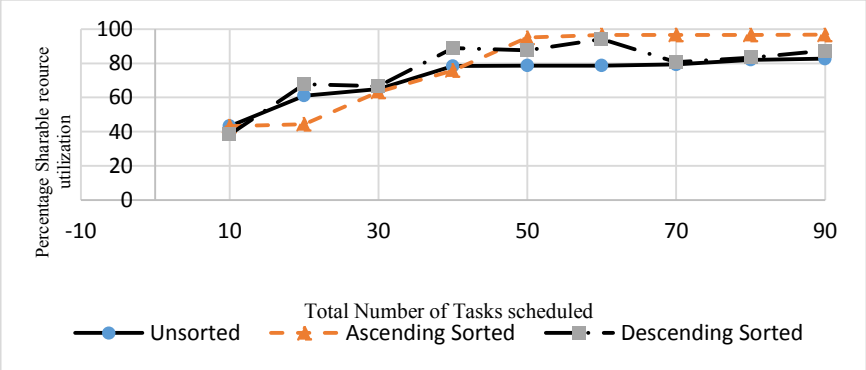
It is clear that, forming a sorted set of devices based on composition score index shows a better performance compared to an unsorted set. In this way the primary goal of defining a composition score is achieved. Also, following an ascending order of C_s enables best device selection scheme by minimizing the devices present in the composition. Fig 5.4 (b) represents number of devices used inside a composition inside the circles. Utilization of a resource based on the composition score is defined as the ratio of resource required for processing the task deducting from the total composed shareable resources to the total shareable resource. For instance, from a composition of N devices, that has shareable resource units of x, the resource required for processing is (x-y/x)\% where y units is the sub-task execution profile. In Fig. 5.5(a) device utilization is shown.

It provides a valuable insight with reference to the resource usage in [51]. Noticeably, [51] provides devices the advantage of executing a minimum of three tasks at all times.

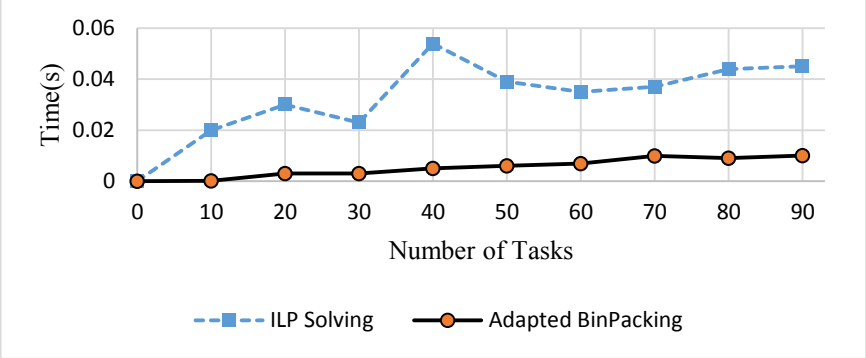
However, from a practical standpoint there could be times when the resources of a device are too low to execute even three tasks. Therefore, our model is developed with the assumption of a device being characterized by its resource potential. In due course, having a composition score provides our model the resilience and requisite performance benefits compared to [51]. In order to evaluate the mathematical model we analyze the time taken in Fig. 5.5 (b). Ostensibly, the agility and quick calculation offered by Bin packing algorithm makes it suitable for faster calculation. On the other hand, the ILP algorithm provides precise results but is time consuming. Additionally, as inferred from formulation, partial offloading or when the task apportioning does not fit within the resource constraints of the device, then the sub-tasks are rejected.

Although Fig. 5.5 (c) clearly shows the lowest rejection ratio for our system model when nodes are ordered according to descending C_s , it indicates an overall possibility of system breakdown when the node density reduces below a certain limit. For example, consider the case of descending order when the number of nodes in a composition goes as high as ninety devices, the rejection ratio reduces to a great extent. On the other extreme, when we look at the case when the node density is low say, between 10 to 20 the rejection ratio is high. This shows that if the node density reduces, then the rejection ratio increases but at the same time synchronization overheads decreases. Hence, this calls for a scenario wise trade-off, which is why, while we consider a volunteer computing scenario, we are looking at like-minded people who would be present for the initiator's service at least till the completion of application execution, like the case of a music concert in a stadium. If we are looking at scantily available nodes for formation of a composition, it does not suit the purpose for this framework is built. Moreover, while we

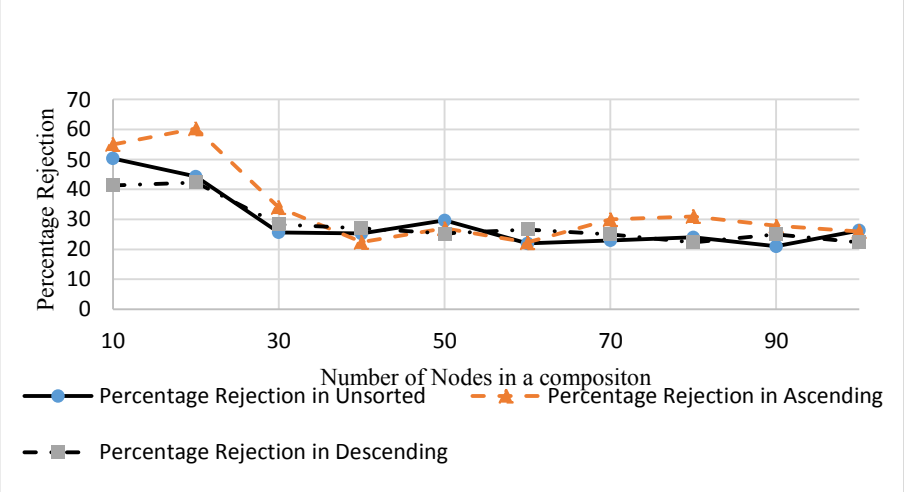
consider campus scenarios, this framework can benefit not only from volunteer presence but also from the vast area of coverage. Recall from the third case study in chapter 4, where the



(a) illustrates device utilization vs tasks scheduled



(b) Time taken by Bin Packing approach and ILP model



(c) shows rejection percentage by nodes not accepting when there is a conflict of interest

Figure.5.5. Performance Analysis of Optimization using Composition Score

movement of nodes in and out of the coverage region is observed, such cases can be avoided if the area will have a high node density in times of lecture hours.

5.4 SDN assisted Mobile Ad-hoc cloud framework

In order to verify the operation and analyze the proposed SDN design in terms of packet drops and packets delivered successfully we built a prototype and performed an emulation in Mininet. Taking cue from the use case of offloading we consider all the three cases mentioned in chapter 4 where in offloading an application is the priority. These are cases that require situation-centric computing environment creation. We consider text processing, but it could be any application. For the purpose of the test case, we consider two labs in the University campus where it takes approximately 5mins to 10 mins to walk. We simulate a topology with 20 stations connected to the Region A access network. The controller is managed by a computer in one of the labs. To simulate such a network the POX controller is used. The topology is built in Mininet-WiFi environment along with OVS switch created to use as a bridge between the wireless network and Ethernet network. The OVS switch version 2.0 is given remote control using OpenFlow version 1.3.

In Mininet-wifi, the behaviour is similar to Hostapd used to create a wireless network on the wireless network adaptor, where the Hostapd allows IEEE 802.11g access point management/authentication on wireless adaptor. As observed therein, creating a wireless network using Hostapd allowed wireless nodes to connect to the OpenFlow-based network. Communication flows through the wireless network was visible at the controller based on which

OVS switch and the OpenFlow-based application could handle those flows accordingly. Stations were modelled for detecting the broadcast made similar to the scenario seen in section 5.2. Out of the 20 stations, one station (UE/Consumer) broadcasts a request for providers and forms the composed participant topology. This topology is visible to the controller once the consumer informs it via the mobility controller interface. Initially, when the consumer attaches itself to the access points the UE with its MAC address,IP address and Ingress port information hits the switch. The switch considers this new flow rule and notifies the controller.

The controller matches this with the flow table entry and installs the rules in the OVS. After formation of the topology, the consumer updates the controller database with all the provider information (i.e. provider MAC, IP etc. To test and simulate a scenario of inter-region user mobility; we used Wi-Fi networks with different subnets. We have omitted the dynamics of the EPC component as it is beyond the scope of this research. For the handover process we switch the wireless network manually in the case of physical implementation i.e. migrate the mobile device VM to the destination wireless network in the Mininet environment. Similar to above mobile device handover scenario, during ad-hoc service migration with the user movement, we use the concept of Locater and Identifier separation for seamless migration across two access networks.

For each scenario a pre-generated movement pattern and traffic loads are developed using the traces in section 5.2. Our intention is not to compare the different mobility models but to ensure seamlessness of the ad-hoc mobile cloud. Figure 5 shows the topology as seen by the controller after the composition has been formed. As making use of Random Way point model and other such state-of-the-are model was not giving specific location based clarity due to a high mobility rate, all mobility scenarios are hand-crafted to evaluate the sensitivity of the case- study.

The simulation area is characterized by a 1000x1000 unit square. The mobility field is characterised by a pause time. Initially each node is stationary at the Access Point for‘t’ seconds, then begins to move to a random destination. The speed of the nodes is modelled for 0.5-2 seconds. On reaching the destination the node pauses for some time. For each case we simulate to an extent between 60s, 90s, 100s, 150s and 200s. Once the random node pattern is observed, in the next iteration a specific point of reference (like the coordinates (60, 80), (80,80) etc.) is chosen to check if the node is still communicating with the controller and the composition to evaluate the resilience of the composition at the same time analyze the software controller’s reachability throughout the area mobile ad-hoc cloud service deployment.

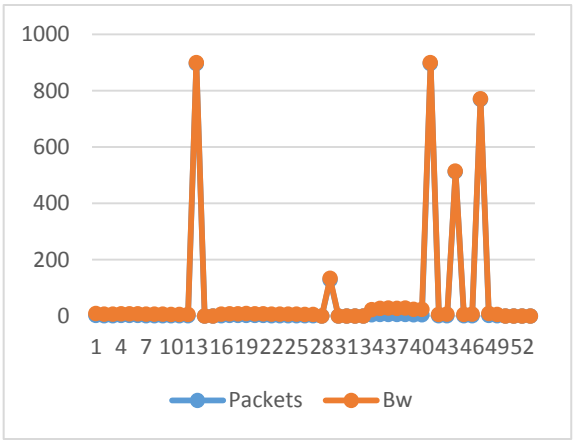
As performance is very sensitive to movement patterns, the scenarios are generated in such a way that it triggers sensitivity and shows how SDN integration customizes the data plane behaviour. When all the nodes are in range of each other there is no packet loss observed. This assures that at the same access point, even in times of high mobility rate the SDN model works without any disruptions. The challenge now is to evaluate the movement between access points and composition maintenance in time of nodes going out of region.

5.4.1 Evaluation of case studies

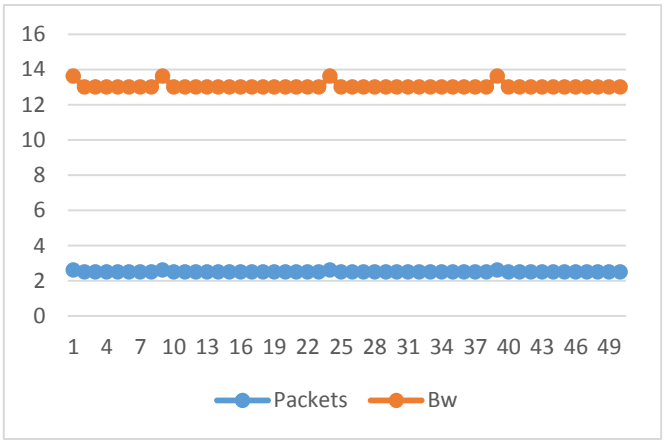
Case 1: At Region A – Handover to different Access Point covering coordinates (60,80):

Here, the establishment of the peer to peer composition takes place at (40,40) Then the node moves to another access point { AP_b covered at coordinates (60,80) }. If we compare all the node interactions at (40,40), it can be observed that between nodes moving in the same access point there is 100% delivery. This assuredly addresses the problem of packet loses we faced in

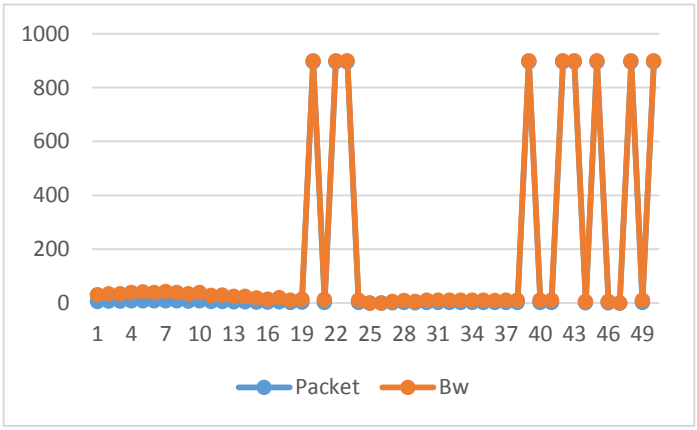
section 5.2. Although, there is intense bandwidth changes observed, based on our pre-generated mobility pattern we can evaluate the following:



(a) Bandwidth between Moving Nodes



(b) Bandwidth between Static nodes

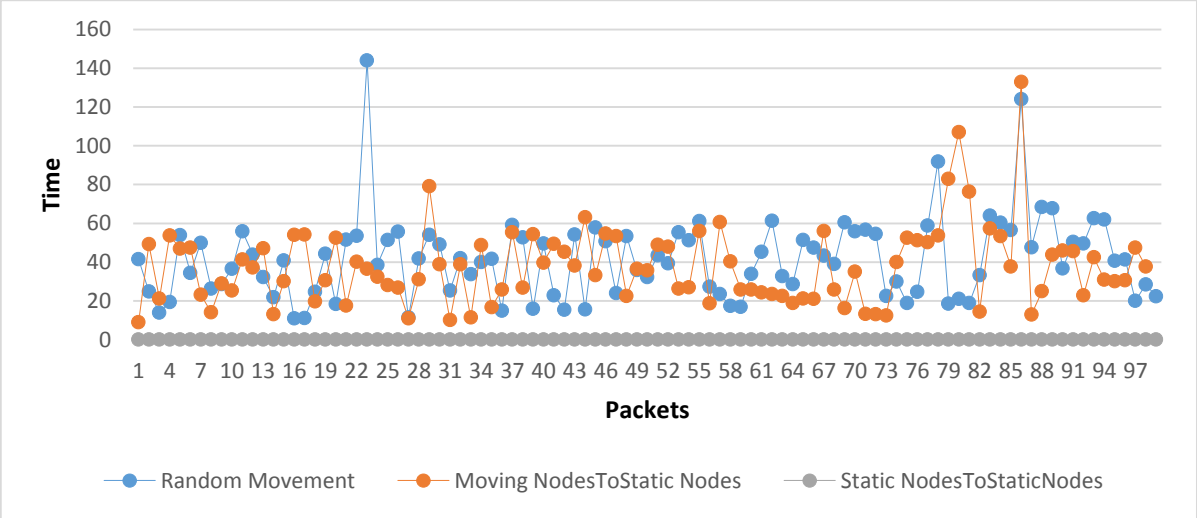


(c) Bandwidth between Moving and Static nodes

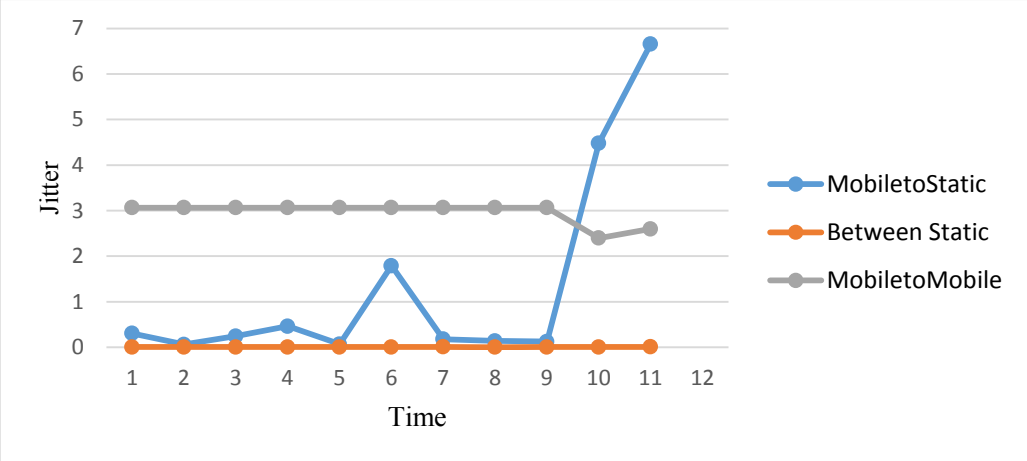
Figure.5.6. Case -1 analysis (Bandwidth Comparison)

In order to make the case more practically relevant the mobility is randomized before making a final decision of handover to coordinate (60,80). In cases when the nodes in motion are sending packets to the static nodes (we assume static nodes as the ones who are present at the same location as the initiation) there is a 2% packet drop. Fig 5.7 shows packet dropped. The jitter

incurred between mobile nodes is comparatively higher as opposed to the stationery case. In times of mobile nodes there are instances when the packets are received out of order due to high mobility rate. There is not much change in bandwidth between stationery nodes. However, in case of random mobility scenario the bandwidth drops to 0 Mbps intermittently based on the contacts left behind.



(a) Packet Drop in times of Random Movement



(b) Jitter Comparison

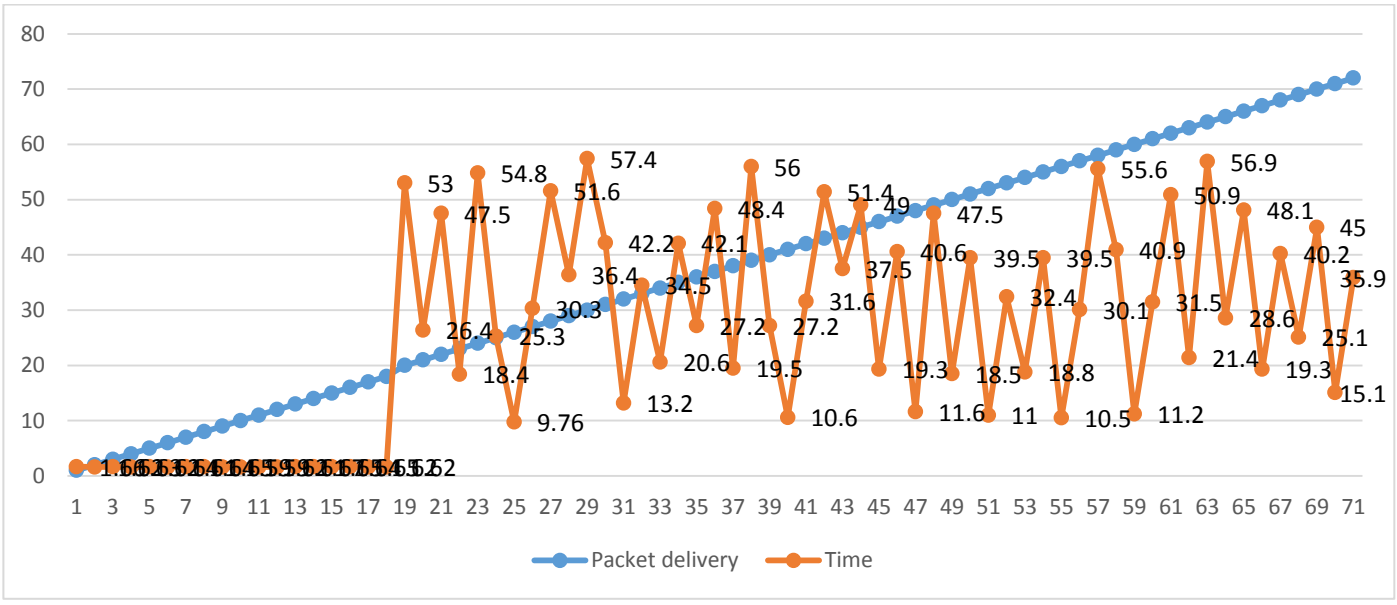
Figure.5.7. Case -1 analysis (Packet drop and Jitter)

A high average round trip time was observed due to some nodes being lost due to random movements. On an average around 37 ms was the observed RTT. The drop of packets can be attributed to the controller losing the node altogether. Sudden peaks of bandwidth correspond to a controller flow rule generation and update time. This is the worst case scenario which will be evaluated further in case 3.

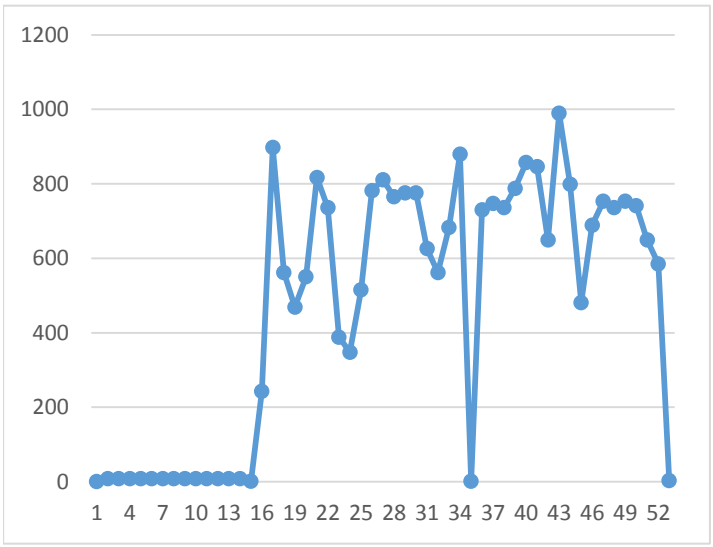
Case 2: From Region A - Region B – Region C – Access point handover (Based on clear definition of paths) :

Post composition, the offloader moves away. That is movement from AP_a, to AP_b followed by shifting to region B eventually settling down in Region C at AP_c When compared to the clearly defined motion between access points, we see that there is approximately 1% packet drop observed illustrated in Fig 5.7 (a),. However, it can be attributed to the minimum packet drop hardcoded in Mininet-wifi and not due to the handover. Bandwidth maintains a consistent approx. 8 Mbps till the node stays at the point of initiation, however, it begins to fluctuate as the movement begins. There are higher jitter rates observed compared to the stationary nodes case because of the handover. This case is similar to the first one except for the clear definition of Access points. This can be looked at from a perspective of student moving from one lab to another lab. The only difference seen was the intense fluctuation of bandwidth as observed.

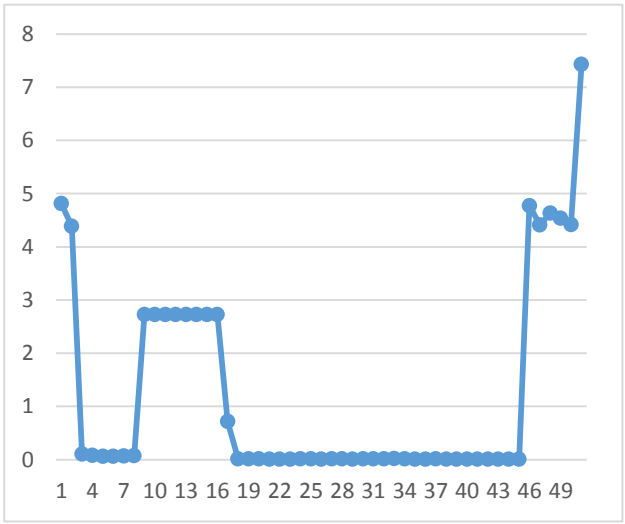
As seen in Fig 5.8(b) . This is because of the intermittent contacts to the APs. Also, the round trip time observed changed heavily while manual changes to the location was made but remained constant when the process was started. Average RTT was approximately 25 ms. The Region A is the place of initiation whereas the movement is unrestricted and follows a multi-hop path pattern. Starting at (40,40), node moves to (60,80).



(a) Packet delivery analysis



(b) Bandwidth Analysis

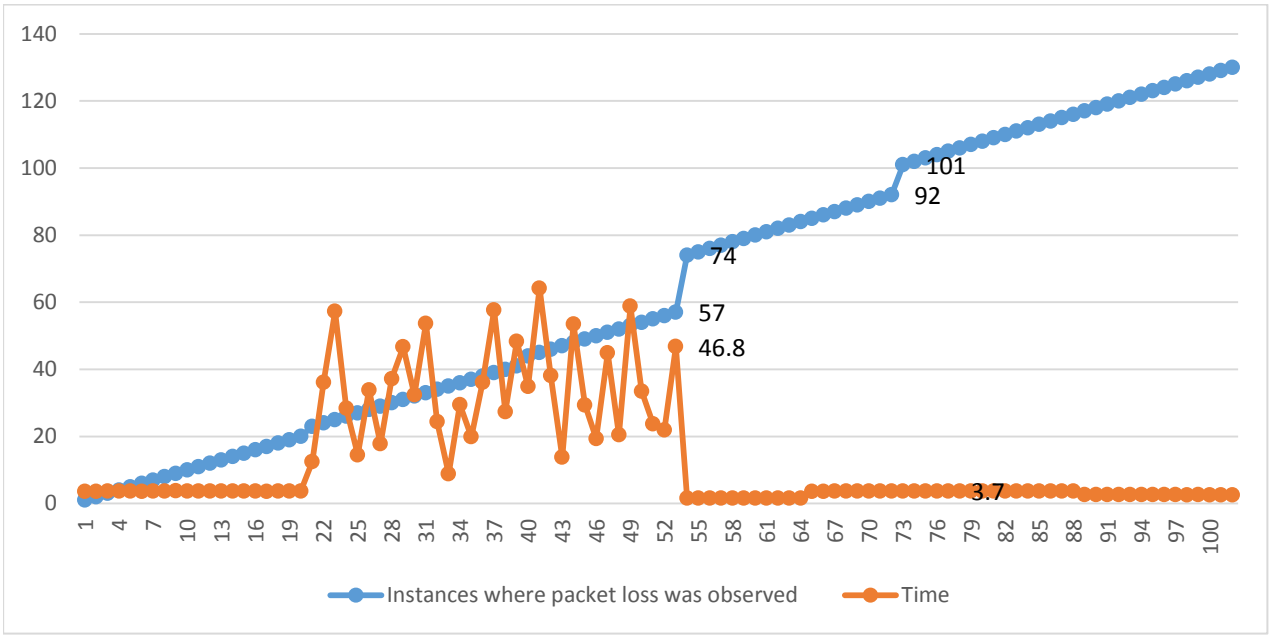


(c) Jitter Analysis

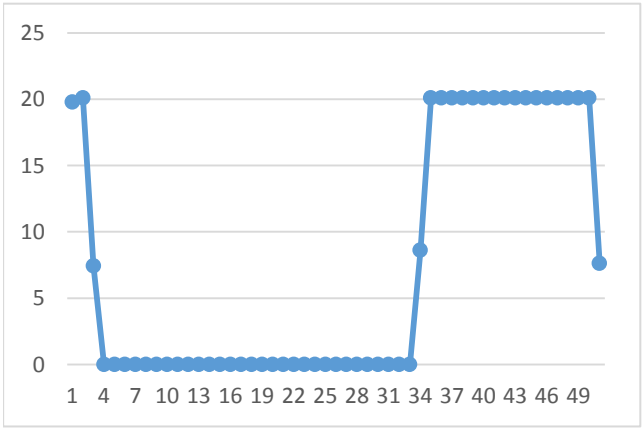
From coordinates (60,80) , the node is moved to (80,80). Jitter results were quite expected as seen in Fig 5.7 (c), whenever the access point recognizes the handover process i.e. at the time of transition, the jitter value changes greatly between the nodes moving across the

access points. Transition between Access Points is demarcated by high jitter, however, once the APs are reached jitter falls back to zero.

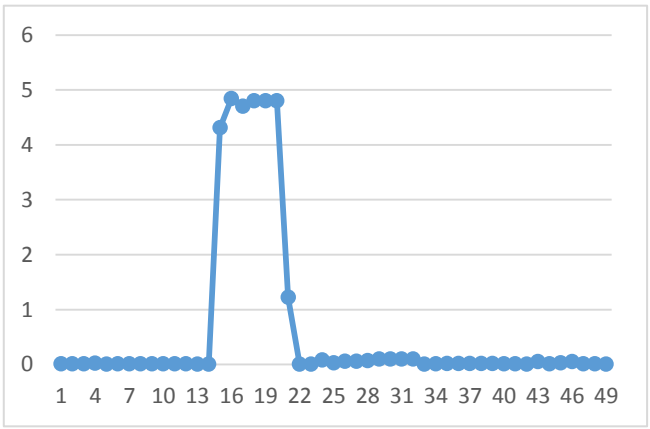
Case 3: From Region A- Out of Coverage Area- Coming back to Region A (Place of initiation)



(a) Packet delivery analysis



(b) Bandwidth Analysis



(c) Jitter Analysis

Figure.5.9. Case -3 Analysis

This is the case when the device goes out of coverage area and comes back. This delineates the controller's failure to reach for the node. However, logically it can be seen that when a user goes out of a particular access network space the reachability is lost. During the ping test the evaluation results show anomalous round trip time behavior. That is after considering multiple iterations of the manual handover, it was observed that after the handover occurs, if the node comes back to the initial Access point where in the composition was formed, the round trip time observed was lower however, it fluctuates while it moves inside the region. To put it conclusively, when the node moves out of the area and is back to any other AP the round trip time drops considerably. This is also the reason why the RTT in this case is lower than case 2. On an average the RTT was observed to be 13 ms compared to the 25 ms in Case 2.

At the Controller :

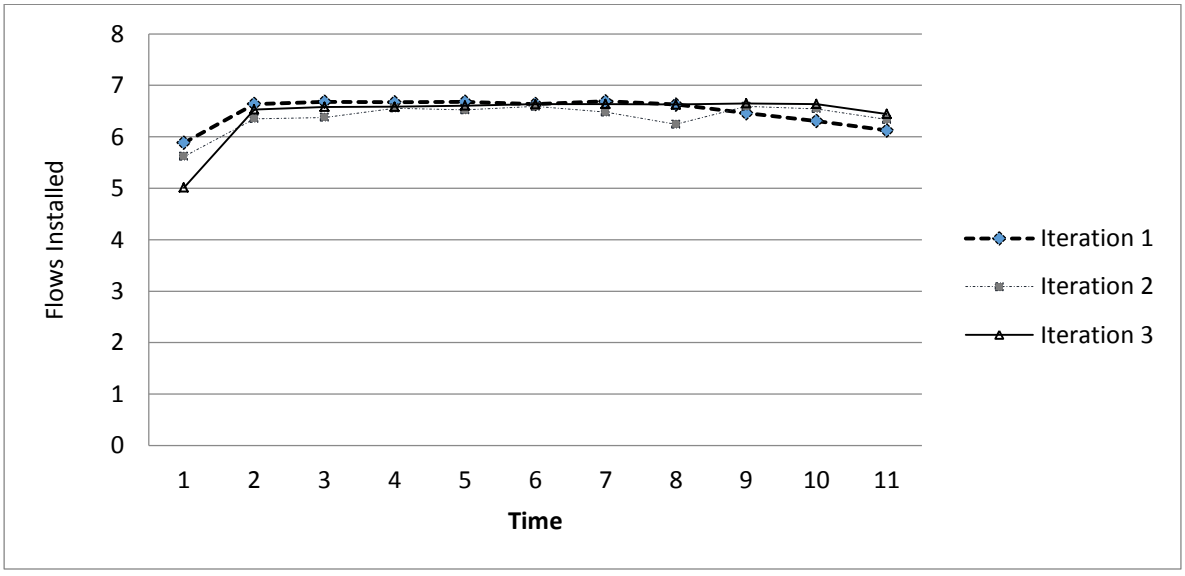


Figure.5.10. Controller performance

On an average the number of flows increases from 5.3 /ms. We consider 4 switches in the topology with four access points. A total of 10 tests were carried out. Fig 5.9 shows controller performance while in operation. A minimum response time on an average was noticeably between 6.4 flows to a maximum of 6.6 flows per ms. All the RTT was measured after the offloading rules are inserted into the switches. Thus, after the completion of detection of movement and new flow rule installation, there is a considerable delay due to network latency from switches to the centralized controller. This delay was minimal and was therefore omitted. As noticed in all the cases with the help of SDN controller, the RTT reduces to a large extent. Further, it is our belief that usage of SDN while considering the Mobile Networks in general can reduce the round trip time. Work done in [37] corroborates this fact when delay was significantly small while using SDN substrate. Therefore, it is a logical hypothesis to make, when we consider a variety of traffic in a mobile network other than P2P traffic but make specific rules for P2P composition traffic to be offloaded to nodes that are part of the composition, inevitably, the mobile ad-hoc cloud composition would provide a low latency environment. Therefore, combined with the benefit of maintaining the seamlessness, this model fortifies the reason and the purpose of having a disruption tolerant mobile ad-hoc cloud. Table 5.2. Shows the comparison of SDN and Non-SDN framework with emphasis on all the cases elaborated in chapter 4.

Control	**Round Trip Time**		
Without SDN Controller	42 ms		
With SDN Controller	***Case 1***	***Case 2***	***Case 3***
	37ms	25ms	13ms

Table 5.2: Comparison of Network Performance With SDN and Without SDN

5.5 Summary

In this chapter, emulated test environments were prepared and evaluated. The outcome of the all the proposed algorithms was observed, and also the test results were plotted considering different metrics. It is also noted that, inter-subnet seamless user migration can be successfully employed with SDN-based wireless networks. By detecting the node movements and round trip times, the packet drop was minimized using the presence of SDN controller. As a result, inter-access network mobile ad-hoc service relocation was carried out with minimal disruption . Except case 1, where the undesirable worst case scenario was analyzed, we observed a hundred percent packet delivery while the P2P composition is under the purview of an SDN controller. A longer RTT was observed while there was no controller which meant longer delays for ping packet reception. A number of improvements in the system were noticed from section 2 through section 4 that tackled most of the deficiencies in the system.

In doing so, the primary goal of having devoted ad-hoc cloud composition connectivity for a mobile user was achieved. As a result, it was observed that, the SDN-assisted mobile ad-hoc cloud service provides a seamless end-to-end connectivity during user mobility and disruption prone events.

6 Conclusion and Future Work

6.1 Conclusion

Mobile Ad-hoc Cloud computing has become a popular topic within the research community as a proven technology to provide computing resources "on-the-fly" to resource-constrained mobile devices. Although, there is a large number of application level services that are provided over the cloud, a resilient approach to allocate computing infrastructure for mobile devices using the ad-hoc cloud paradigm has not been extensively investigated.

IaaS for mobile computing provides computing resources closest to the user with a lower access delay and a guaranteed bandwidth. However, this delay can be further reduced by the mobile ad-hoc cloud computing paradigm. Once resources are composed in the mobile ad-hoc cloud, it is required to maintain the user location and mobility pattern. As a result, mobility aware ad-hoc cloud computation service control mechanism is required to reactively access the ad-hoc cloud service and maintain minimal service cost based on the location of the user. In this research work, we have proposed a framework for Infrastructure as a service for mobile ad-hoc cloud paradigm. Additionally, an SDN-assistance in the access networks and ad-hoc cloud composition based mobility management is proposed.

Initially, we provided the general definition of mobile cloud computing together with an introduction to existing technologies. Then, we have presented how ad-hoc cloud computing takes inspiration from the cloud computing paradigm and shares benefits of volunteer computing, grid computing and a comprehensive survey of mobile cloud computing approaches by examining their purpose, placement, proximity and usage. We have also summarized the

drawbacks of traditional networks and presented a motivation for software defined networks. We have described the benefits of software defined networks for ad-hoc clouds together with an introduction to OpenFlow protocol. Additionally, we have included existing approaches to handle mobility across SDN-based wireless networks to seamlessly manage end to end connectivity with the cloud.

We have focused on the main framework of IaaS for mobile ad-hoc clouds which provides computation services for mobile users. We leveraged the benefits of resource composition in mobile ad-hoc cloud in situation centric scenarios that are time sensitive and introduced IaaS composition algorithms giving importance to spontaneity and structured routing. A task scheduling algorithm was developed that is based on a Composition Score (C_s) metric that considers the popularity, stability and shareable resource capacity in a node based on which the nodes in the vicinity are chosen for forming a composition.

Finally, in pursuit of fortifying a seamless resource relocation mechanism in an ad-hoc cloud the Software Defined technology was introduced. In order to orchestrate ad-hoc cloud resources among requests a software controller was used. The ad-hoc cloud service controller was introduced as a logically centralized controller managing the entire topology of SDN wireless nodes. These nodes have pre-defined peering traffic routing table maintenance and composition detection capabilities that act as lower layer in a hierarchy of controllers. When user mobility is observed across regions, the monitoring module sniffs the movement and generates flows that are installed appropriately for managing the mobility. This inadvertently assists in staying connected to the point of initiation and maintenance a seamless service. The primary goal of having a disruption tolerant mobile ad-hoc cloud service framework to support seamless user

mobility across wireless networks without changing their network identity and thereby maintaining a transparent cloud access was achieved in this way.

Further, based on the experimental results, we conclude that a lower latency ad-hoc cloud environment is achieved with the assistance of software defined networking technology in comparison to a non-SDN environment. Additionally, most of the failures seen in the state-of-the-art mobile ad-hoc cloud framework is analyzed based on exhaustive case studies and has been overcome.

6.2 Future Work

Although, an improved system performance was observed, this research work comes with a set of difficulties. There are some more alternatives that need to be found to improve user experience in an ad-hoc composition. Firstly, in a mobile environment there is a need to consider the air-interfaces along with the total power consumption and energy efficiency for understanding the feasibility holistically. Further, microkernel implementations are a necessary support for building an agile arbitrary virtual system. A more robust microkernel base can demonstrate a higher level of strength in the ad-hoc cloud composition.

Secondly, while we consider back and forth computation, there is a need to specify a deadline for the mobile nodes to send the results back to avoid resource wastage as well as to keep the offloader waiting. This also points towards another important step of creating sections of compositions in a 2-D plane. For instance, the 0 degree direction can be made use of as a reference and each composition can be defined by the angular range based on the node's presence. Making the deadline/time-out specific to this angular range and massively assist in

reducing the time of wait for the offloader and the same time improve the composition entity as a whole.

Thirdly, SDN controller is a valued commodity that can offer more services that is delineated. One such feature is deep packet inspection. We have seen how offloading of P2P composition traffic can implicitly benefit the offloader by offering a low latency execution environment, however, considering other stray traffic in the core network and isolating it from composition would be achievable by deep packet inspection. Regardless of any other traffic in the network, the composition services would utilized to a full extent and the overall robustness offered by such an ad-hoc cloud would be also be commendable. We elaborate these directions further in the following sections.

6.2.1 Micro-kernel Implementation and Real World Traces

Typically a hypervisor, is a virtual machine monitor. It is used to run a de-privileged operating system. A kernel can be defined as the software that runs in a privileged mode. Therefore, usage of a hypervisor by itself would make the design purpose specific, whereas having a microkernel would bring in minimality and light-weighted-ness. Simply put, a hypervisor can run guest operating systems [67] whereas a microkernel needs only a minimal amount of code to run the most privileged mode of the hardware for building any arbitrary system especially in mobile phones. Additionally, a separate API for ad-hoc communications in android that taps onto the micro-kernel needs to be developed. Currently, android's security architecture defines a variety of permissions to access network resources and sensors so developing a separate API to support ad-hoc communication between these devices would be of value.

Moreover, in order to test the system on practical grounds, a small scale music festival like Ottawa Jazz Festival, wherein the stages are meticulously planned and sectored, can provide an in-venue evaluation of the system.

6.2.2 Creating Sections of Compositions

One plausible way to reduce the rejection ratio seen in Chapter 5 section 5.3 is to evaluate the node density before beginning of the composition. Therefore, creating sections of composition in an ad-hoc environment can not only augment the performance of a composition but would also enhance the resilience of an ad-hoc composition as the range specified for each section inside a sector can provide multitudes of benefits. That is two different sections may have multiple compositions that provide different services. Further, these compositions can have a predictive task scheduling to account for intermittent node movements in a location.

Apart from having a predictive intermittent look-out for nodes in the vicinity, an opportunistic cloud service mechanism can be inevitably developed with ad-hoc cloud being the closest resource entity followed by a cloudlet, an MEC and the traditional cloud respectively.

6.2.3 Deep Packet Inspection and Multi-Controller Network Management

Lawful interception of traffic in volunteer scenarios is beneficial for not only the consumers making use of the service but also for providers who can eliminate bottlenecks from the network in a quick and efficient manner. Deep packet inspection of P2P traffic can unload the core network on one hand and provide a further reduction in latency, thereby creating an ideal compute environment.

As our SDN framework is more inclined towards P2P traffic identification, there are many possibilities of detecting other EPC traffic alongside P2P traffic. Usage of realistic EPC components like OpenEPC for deployment of such service is certainly achievable in the future. Moreover, bringing more scalability by introducing multiple controllers at different locations can lead to a more dynamic framework. Further, in order to enable a large scale ad-hoc cloud service framework a distributed control mechanism can be used that provide a seamless service across multiple access networks spanning across different regions.

References

[1] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing."[Online,Accessed 20 January 2016], (2011): 20-23

[2] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang ,"A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches" in *Wireless Communications and Mobile Computing*, Vol 13, Issue 18, December 2013, pp. 1587–1611

[3] U. Drolia, N. Mickulicz, R. Gandhi, and P. Narasimhan, "Krowd: A Key-Value Store for Crowded Venues," *in Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, 2015, pp. 20–25,Paris, France .

[4] mobiforge.global-mobile-statistics-2013-section-e-mobile-apps-app-stores-pricing-and-failure-rates.

[5] O. Babaoglu and M. Marzolla, "Peer-to-Peer Cloud Computing," pp. 1–9, [Online Accessed-26 January 2017], 2011

[6] G. a. McGilvary, A. Barker, and M. Atkinson, "Ad Hoc Cloud Computing," *2015 IEEE 8th Int. Conf. Cloud Compu*t., pp. 1063–1068, 2015, Vancouver, Canada

[7] A. Chandra and J. Weissman, "Nebulas : Using Distributed Voluntary Resources to Build Clouds," *Proc. 2009 Conf. Hot Top. cloud Comput.*, vol. San Diego, no. San Diego, CA, June 2009., p. 2, 2009.

[8] S. A. Abid, M. Othman, and N. Shah, "A Survey on DHT-Based Routing for Large-Scale Mobile Ad Hoc Networks," *ACM Comput. Surv*., vol. 47, no. 2, pp. 1–46, 2014.

[9] Wave your Phone in the Air: How Technology is Changing Live Music-Craig Rosen. [online Accessed :23 January 2017], 2015.

[10] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh, “Cells- A Virtual Mobile Smartphone Architecture,” *Proc. Twenty-Third ACM Symp. Oper. Syst. Princ. - SOSP* ’11, p. 173, 2011,Cascais, Portugal

[11] K. Barr, P. Bungale, S. Deasy, V. Gyuris, P. Hung, C. Newell, H. Tuch, and B. Zoppis,”The VMware Mobile Virtualization Platform: Is That a Hypervisor in Your Pocket?”, *ACM SIGOPS Operating Systems Review*, Vol. 44: Issue:4, pp 124–135, Dec. 2010.

[12] C. Dall and J. Nieh.”KVM for ARM”. *In Proceedings of the Ottawa Linux Symposium*, Ottawa, Canada, June 2010.

[13 J. Hwang, S. Suh, S. Heo, C. Park, J. Ryu, S. Park, and C. Kim. “Xen on ARM: System Virtualization using Xen Hypervisor for ARM-based Secure Mobile Phones”, *In Proceedings of the 5th Consumer Communications and Network Conference*, Las Vegas, NV, Jan. 2008.

[14] Open Kernel Labs. OKL4 Microvisor, Mar. 2011. http: //www.ok-labs.com/products/okl4-microvisor [Online, Accessed 26 January 2017]

[15] D. P. Anderson, J. Cobb, E. Korpela,M. Lebofsky, and D.Werthimer ” SETI@home: an experiment in public-resource computing” *Commun. ACM*, 45:56–61, November 2002.

[16] A. Mawji, “A Framework for Peer-to-Peer Computing in Mobile Ad Hoc Networks,” Sch. Comput., vol. Doctor of, no. January, 2010, Queens University, Kingston,Canada.

[17] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, “Serendipity: Enabling remote computing among intermittently connected mobile devices,” in *Proc. 13th ACM Int. Symp.Mobile Ad Hoc Netw. Comput*., Hilton Head, SC, USA, Jun. 11–14, 2012, pp. 145–154..

[18] Hamdaqa, Mohammad, Mohamed M. Sabri, Akshay Singh, and Ladan Tahvildari. "Adoop: MapReduce for ad-hoc cloud computing." *In Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*, pp. 26-34. IBM Corp.,USA, 2015

[19] Gary McGilvary, Adam Barker, Malcolm Atkinson. "Ad hoc Cloud Computing: From Concept to Realization." arXiv 1505.08097 v2, 2015, Harvard, USA,

[20] Abderrahmen Mtibaa et al. " Towards Mobile Opportunistic Computing", 2015 IEEE 8th International Conference on Cloud Computing, New York City, NY, 2015, pp. 1111-1114 .

[21]. Y. Al Ridhawi, G. Kandavanam, and A. Karmouch, "A Dynamic Hybrid Service Overlay Network for Service Compositions", WORLDCOMP'11; July 18-21,2011,Las Vegas, USA

[22] E. Park and H. Shin, "Reconfigurable service composition and categorization for power-aware mobile computing," IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 11, pp. 1553–1564, 2008.

[23] A. Mawji and H. S. Hassanein, "Bootstrapping P2P Overlays in MANETs," *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, New Orleans, LO, 2008, pp. 1-5.

[24] M. Black and W. Edgar, "Exploring mobile devices as grid resources: Using an x86 virtual machine to run BOINC on an iPhone*," Proc. - IEEE/ACM Int. Work. Grid Comput*., pp. 9–16, 2009,USA

[25] A. Malhotra, S. K. Dhurandher and B. Kumar, "Resource allocation in multi-hop Mobile Ad hoc cloud," *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, Chandigarh, 2014, pp. 1-6

[26] D. Maymounkov, Petar ; Maziéres, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," *Proceeding IPTPS '01 Revis. Pap. from First Int. Work. Peer-to-Peer Syst*., pp. 53–65, Nov. 2002, NY, USA.

[27] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakr- ishnan. "Chord: A scalable peer-to-peer lookup protocol for internet applications." *In ACM SIGCOMM*, pages 149–160, September 2001, San Diego, USA.

[28] Antony Rowstron and Peter Druschel. "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems". *In IFIP/ACM In-ternational Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001, Heidelberg, Germany.

[29] B. Zhao, J.Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Comput. Sci. Div., Univ. California, Berkeley, Tech. Rep. UCB/CSD-01-1141, 2001.

[30] Open Networking Fundation. "Software-defined networking: The new norm for networks." ONF White Paper (2012)..

[31] 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* ,Vol.38, Issueno. 2 (2008): 69-74.

[32] OpenFlow Switch Specification of OpenFlow protocol 1.0: www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf . [Online, Access 23 January 2017]

[33] OpenFlow Switch Specification of OpenFlow protocol 1.3.2: www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.2.pdf . [Online,Access 23 January 2017]

[34] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "Open Roads: empowering research in mobile networks," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125- 126, January 2010.

[35] P. Baskett, Y. Shang, W. Zeng, and B. Guttersohn, "SDNAN: Software-defined networking in ad hoc networks of smartphones," 2*013 IEEE 10th Consum. Commun. Netw. Conf. CCNC* 2013, pp. 861–862, 2013, Las Vegas USA

[36] M. Usman, A. A. Gebremariam, F. Granelli and D. Kliazovich, "Software-defined architecture for mobile cloud in device-to-device communication," *2015 IEEE 20th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, Guildford, 2015, pp. 75-79

[37] J. Cho, B. Nguyen, A. Banerjee, and R. Ricci, "SMORE: software-defined networking mobile offloading architecture,", in *Proc. 4th workshop on AllthingsCellular*, pp. 2–7, 2014. Chicago, USA

[38] Banerjee, A., Chen, X., Erman, J., Gopalakrishnan, V., Lee, S., and Van Der Merwe, J. "MOCA: a lightweight mobile cloud offloading architecture". *In Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture* ACM, pp. 11–16, (2013), SOSP, Santa Clara, USA

[39] K. Pentikousis, et al., "Mobileflow: Toward Software-Defined Mobile Networks,"IEEE Commun. Mag., vol. 51, no. 7, Jul. 2013, pp. 44–53.

[40] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing" IEEE Pervasive Computing, vol. 8, no. 4, pp. 14–23, 2009.

[41] K. Ha, G. Lewis, S. Simanta, S and M. Satyanarayanan. "Code Offload in Hostile Environments". Carnegie Mellon University, CMU-CS-11-146, 2011.

[42] U. Drolia *et al*., "The Case for Mobile Edge-Clouds," *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Vietri sul Mere, 2013, pp. 209-215.

[43] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, "Computing in cirrus clouds: The challenge of intermittent connectivity*," in Proc. ACM SIG- COMMWorkshop MCC*, Helsinki, Finland, Aug. 13–17, 2012, pp. 23–28..

[44] Min Chen et al "On the computation offloading at Ad-hoc cloudlet: Architecture and Service modes", in IEEE Communications Magazine-Communications Standards Supplement, vol 22; pp 23-30, June-2015.

[45] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," *Proc. 1st ACM Work. Mob. Cloud Comput. Serv. Soc. Networks Beyond MCS* 10, vol. 16, no. 2010, p, CA, USA

[46] D. M. Shila, W. Shen, and Y. Cheng[Online], "AMCloud : Towards a Secure Autonomic Mobile Ad Hoc Cloud Computing System.", 2016.[Accessed 23 January 2017]

[47] Khalifa A, Azab M, Eltoweissy M. "Towards a mobile ad- hoc cloud management platform". *In Proceedings of the 2014 IEEE/ACM7th International Conference on Utility and Cloud Computing*, IEEE Computer Society, 2014, Washington DC, USA.

[48] Mohamed Azab and Mohamed Eltoweissy,"CyberX: A Biologically inspired Platform for Cyber Trust Management," *8th International Conference on Collaborative Computing*, Oct 2012, USA

[49] E. Cuervo, A. Balasubramanian, D.-k. Cho, A.Wolman, S. Saroiu, R. Chandra, and P. Bahl. "Maui: making smartphones last longer with code offload". *In ACM MobiSys*, 2010,NY, USA.

[50] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti "Clonecloud: elastic execution between mobile device and cloud". *In Proceedings of the 6th European Conference on Computer Systems* (EuroSys'11), pages 301–314, 2011.

[51] X. Wang, Y. Sui, C. Yuen, X. Chen and C. Wang, "Traffic-aware task allocation for cooperative execution in mobile cloud computing," *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, Chengdu, 2016, pp. 1-6.

[52] Y. Li, L. Sun, and W. Wang, "Exploring device-to-device communication for mobile cloud computing," *in Proceedings of IEEE ICC*, Sydney, NSW, pp. 2239–2244, June 10-14 2014.

[53] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," IEEE Commun. Lett., vol. 18, no. 10, pp. 1779 –1782, 2014.

[54] S.-H. Hung, T.-T. Tzeng, G.-D. Wu, and J.-P. Shieh, "A code offloading scheme for big-data processing in android applications," Volume 45 Issue 8, August 2015, Software: Practice and Experience, 2015

[55] A. Mtibaa, K. a. Harras, and A. Fahim, "Towards Computational Offloading in Mobile Device Clouds," *2013 IEEE 5th Int. Conf. Cloud Comput. Technol. Sci*., vol. 1, pp. 331 –338, 2013

[56] Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C., Jetcheva, J " A performance comparison of multi-hop wireless ad hoc network routing protocols". *In: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking;* Mobi- Com '98. New York, NY, USA:; 1998, p. 85–97.

[57] Xu, Q., Huang, J., Wang, Z., Qian, F., Gerber, A., and Mao, Z. M. "Cellular data network infrastructure characterization and implication on mobile content placement", *In Proceedings of*

the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems ,ACM, pp. 317–328., 2011,San Jose, USA

[58] 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.

[59] Jin, X., Li, L. E., Vanbever, L., and Rexford, J. "Softcell: Scalable and flexible cellular core network architecture". *In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies , ACM*, pp. 163–174, 2013,Santa Barbara, USA.

[60] S Shanmugalingam, P Bertin," Programmable mobile core network", *in Proceedings of the nineteenth IEEE Symposium on Computers and Communications (ISCC)*. IEEE, Madeira, Portugal, pp. 1–7, 2014.

[61] V. Nguyen and Y. Kim, "Proposal and evaluation of SDN-based mobile packet core networks*," EURASIP J. Wirel. Commun. Netw*., vol. 2015, no. 1, p. 172, 2015.

[62] S. Chourasia and K. M. Sivalingam, "SDN based Evolved Packet Core architecture for efficient user mobility support," *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, London, 2015, pp. 1-5.

[63] Chen, Y.-C., and Towsley, D. "On bufferbloat and delay analysis of multipath tcp in wireless networks". *In IFIP Networking ,IFIP*, 2014

[64] B.Venkatraman and A.Karmouch, "An Infrastructure as a Service for Mobile Ad-hoc Cloud", *in Proceeding IEEE-CCWC,* Las Vegas, USA, 2017.

[65] V. Autefage, D. Magoni, and J. Murphy, "Virtualization Toolset for Emulating Mobile Devices and Networks," *IEEE/ACM Int. MobileSoft*, 2016, Austin, USA

[66] Mininet : http://mininet.org/

[67] Secure embedded systems need microkernel [Accessed : 23 January 2017] [https://ts.data61.csiro.au/publications/papers/Heiser_05.pdf]

[68] I. Ku, Y. Lu and M. Gerla, "Software-Defined Mobile Cloud: Architecture, services and use cases," *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Nicosia, 2014, pp. 1-6.

[69] Reactive and Proactive Flow [Online, Accessed 24 January 2017] http://networkstatic.net/openflow-proactive-vs-reactive-flows/

[70] Lingxia Liao , Meikang Qiu, Victor C. M. Leung "Software Defined Mobile Cloudlet" Volume 20, Issue 3 , pp 337-347 Springer US, June 2015

[71] "SDN and Legacy Network Infrastructure" ,November 2013, http://www.enterprisenetworkingplanet.com/datacenter/sdn-and-legacy-network-infrastructure.html , [online, Accessed 26 January 2017]