DRAP: A decentralized public resourced cloudlet for ad-hoc networks

by

Radhika Agarwal

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.C.S. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

© Radhika Agarwal, Ottawa, Canada, 2014

Abstract

Handheld devices are becoming increasingly common, and they have varied range of resources. Mobile Cloud Computing (MCC) allows resource constrained devices to offload computation and use storage capacities of more resourceful surrogate machines. This enables creation of new and interesting applications for all devices.

We propose a scheme that constructs a high-performance de-centralized system by a group of volunteer mobile devices which come together to form a resourceful unit (cloudlet). The idea is to design a model to operate as a public-resource between mobile devices in close geographical proximity. This cloudlet can provide larger storage capability and can be used as a computational resource by other devices in the network. The system needs to watch the movement of the participating nodes and restructure the topology if some nodes that are providing support to the cloudlet fail or move out of the network. In this work, we discuss the need of the system, our goals and design issues in building a scalable and reconfigurable system.

We achieve this by leveraging the concept of virtual dominating set to create an overlay in the broads of the network and distribute the responsibilities in hosting a cloudlet server. We propose an architecture for such a system and develop algorithms that are requited for its operation. We map the resources available in the network by first scoring each device individually, and then gathering these scores to determine suitable candidate cloudlet nodes.

We have simulated cloudlet functionalities for several scenarios and show that our approach is viable alternative for many applications such as sharing GPS, crowd sourcing, natural language processing, etc.

Acknowledgements

Acknowledgements are due to the many people who have supported me, either directly or indirectly, in completing this work.

Firstly, I would like to acknowledge and thank Professor Amiya Nayak, my thesis adviser, for all his help, guidance, advice and support for the past one and a half year that I have worked with him.

Appreciation is also expressed to the staff, professors and graduate students of Ottawa-Carleton Institute for Computer Science, and Faculty of Graduate and Post-doctoral Studies for their help and efforts in providing an academic environment.

My deepest gratitude goes to Dr. Nishith Goel, Mrs. Nita Goel, Dr. Shivendra Goyal and my entire family in Canada for their advice, encouragement and unconditional love during my stay here.

I would like to thank my mother, sister and brother for their support, care and concern over the past few years of my education away from home. They have been a constant source of strength and comfort, even when I have not always shown my appreciation. And a big thank you to all my family and friends for cheering me up at all times. Thank you for believing in me.

Finally, I would like to dedicate this work to my father Late Dr. Pramod Agarwal. I hope I make you proud.

Thank you,

Radhika

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

For the past few years we have witnessed the burst of an enabling technology called cloud computing. Cloud computing utilizes powerful computing resources (e.g. networks, servers, storage, applications, and services) conveniently and ubiquitously. It is defined with a model for enabling convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned, leased, and released with minimal management effort or service provider interaction [40]. Many commercial cloud services have been deployed by the major players in the industry, as well as numerous researches have been conducted. During this process many non-believers have turned to boosters. The focus of the academia has shifted from arguing whether cloud computing can fulfill the expectations to designing more applications for it.

Another phenomenon that is impossible to overlook is the developing of advanced mobile devices such as smart phones, tablets, handheld gaming consoles, etc. For instance, the smart phones nowadays are equipped with dual core even quad core processors, gigabytes of memory, large screen, camera and a wide range of sensors. The era of wireless telephony began in 1918 with the development of mobile telephone on military trains between Berlin-Zossen, and in less than 100 years the research community is debating on the 5th generation of wireless systems.

While Mobile Cloud Computing is being considered the biggest trend in next 5 years [1], market predictions are increasing rapidly. Juniper Research predicted in 2010 that the cloud-based mobile market will generate annual revenue of $9.5 billion in 2014 (up from $400 million in 2009), at an average annual increase of 88 percent. Recently the IT research firm Gartner said, the worldwide market for cloud computing can be worth $206.6 billion in 2016.

Mobile devices connected via wireless networks can form a huge pool of resources. The evolution of cellphone to smartphone has given its users a mini computer on the tip of a finger. The operating system in a smartphones is capable of being programmed and form clusters.

A recent trend combining these two concepts gives rise to Mobile Cloud Computing (MCC), which broadly includes three different scenarios:

1. The first is the most common and traditional way, use of mobile phones to access the cloud services like Infrastructure as a service (IaaS), Platform as a service (PaaS), Software as a service (SaaS) etc. A hypervisor creates and runs a virtual machine depicting the device as a guest on the cloud. It is aimed to provide an anywhere, anytime rich user experience on mobile devices through internet [55].

2. The second scenario is to offload tasks of mobile devices to the cloud. Although the significant progress of mobile devices computing powers, it is still very hard for them to perform some compute intensive tasks such as voice recognition and image indexing, etc. With MCC, it is hoped we can further augment our experiences with mobile devices by utilizing the cloud services. It has also been

argued that energy constraint of mobile devices may also be relieved with the same endeavor [40];

3. The third scenario is that mobile devices can cluster together to contribute as a traditional cloud and provide services. This is motivated by the rich computing resources on modern mobile devices and their ability to act as perfect interface with human input.

With the advantage of internet mobile users use their smartphones not just for phone calls and texting but for much more processing and hardware intensive tasks, like web browsing, voice recognition, picture editing etc. Today when IT industry is on a very rapid progress, mobile devices are facing challenges with respect to their resources like storage, battery life, bandwidth, mobility, security etc.

Additionally, Crowd Sourcing is a developing term in the online community today. Inspired by outsourcing, crowdsourcing was introduced by Jeff Howe and Mark Robinson in a Wired Magazine article in June 2006 as a large community of potential workers with diverse skills and expertise, who can use their spare time to create content, solve problems, even do corporate R & D. Unlike traditional problem solving methods, this is an open invitation for any interested individual or group to submit a part of solution, which is later assessed for quality and if found satisfactory is added to the main solution. It is becoming a powerful mechanism to volunteers complete the task, successful applications like Wikipedia.com [6], Amazon Mechanical Turk (MTurk) [16], OpenStreetMap [45] are good examples.

In mobile communication a few tools have been developed to support the idea of crowdsourcing. mCrowd [63] is an iPhone based crowdsourcing platform; users can post and work on sensor-related tasks, such as image and object tagging, image data collection, textual queries, GPS location based queries etc. Rich sensors like image tagging, geo-location, traffic monitoring are well utilized in the tool.

The infrastructure issue being dealt in this work is Mobility and Fault Tolerance. Other issues could be Inter-operability, Privacy and Security, and Context Awareness.

1.1 Motivation

The motivation for this work comes from the idea of *cloudlet-based and resource-rich mobile computing* discussed by M. Satyanarayanan et. al in [57]. With the heterogeneity of sizes, resources and properties available in hand-held devices it is safe to say that most geographically connected areas are resource rich, enough to cater the needs of the locale.

"A cloudlet is a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices." [57] Owing to its physical proximity to the device, cloudlets offer one-hop away services with low-latency and high-bandwidth LAN access. Loss or destruction of a cloudlet is not catastrophic as it contains only the soft state i.e. the cached copies of data or code of the application originally hosted elsewhere.

Volunteer computing [56], also known as public resourced computing, practised with mobile devices to contribute to the formation of a pool of resources. These shared resources can act as a cloudlet to serve as an infrastructure for distributed computing. Such a design will enable resource-rich devices to donate supplies to the cluster, where in the resource-constraint devices could draw these resources to execute their applications.

BOINC [11] is the most commonly used volunteer computing infrastructure, where unused CPU and GPU cycles are offered by idle computers to perform distributed applications in diverse fields like mathematics, molecular biology, meteorology and many others those need huge amount of computing power. Similar system on mobile device has a lot of short term impacts on the device (battery drain, strained CPU, cache related problems) and has to deal with subject of mobility and connectivity.

There has been a recent discussion about free nation-wide wifi or super wifi which basically means that all wifi connected devices would be on one network. This will give the IT industry enormous opportunity to deal with challenges regarding latency and energy consumption. In such a setup, hop count will reduce significantly for peer-to-peer communication models.

Any given locale today has abundant mobile devices and if they are able to communicate with the network with very low cost, it will enable a lot of newer applications with a good deal of social and economic benefits.

1.2 Problem Statement

Design a completely de-centralized model, in which near geographic hand-held devices can be connected in a manner that they exhibit a strong communication channel. This connection be then leveraged to portion out a part of un-utilized resource to form a logical resource pool. Any other device that is a part of such a network should be able to consume the available resource in an efficient manner. This pool of resources should have the capability to act as a cloudlet to the devices in proximity.

Hand-held devices are mobile by definition and are available in various tools and technologies, thus the network formed is expected to be highly dynamic and heterogeneous. The algorithms developed should cater the needs of ad-hoc networks, provide fault tolerant mechanisms to deal with broken links and node failure, and show signs of scalability and stability.

To ensure the longevity of the cloudlet, working model should be low in cost and maintenance. It should have minimum setup time and low message complexity while building and maintaining the structure.

The mobile device should gracefully to able to fallback to a distant cloud if no local cloudlet is available in the vicinity of the device or if the cloudlet fails to provide needed resources. Figure 1.1 is an overview of such an ad-hoc network.

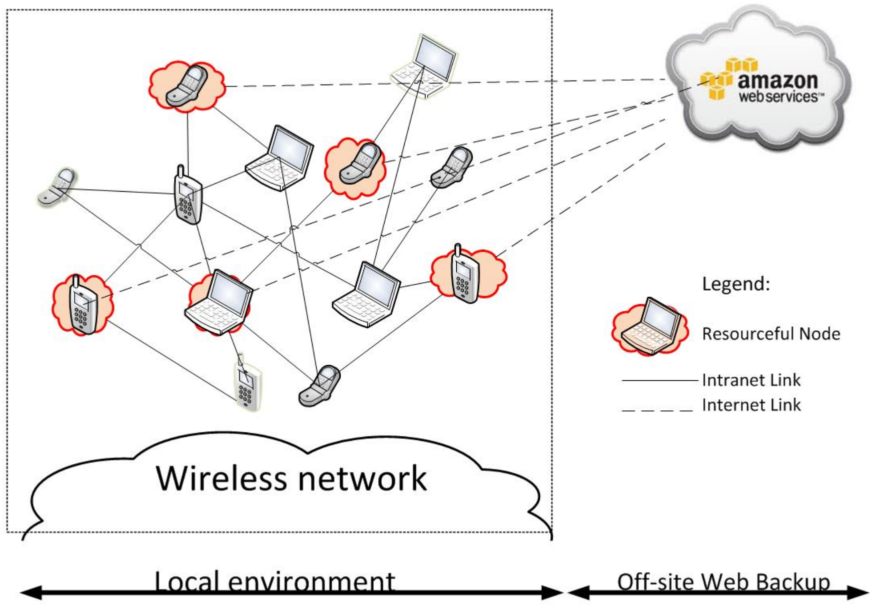


Figure 1.1: Ad-hoc network

1.3 Goals

The goals of this work are as follows:

- Provide motivations for the use of mobile devices in a cloud computing architecture by discussing usability of a de-centralized design.
- Reviewing and highlighting the various aspects of Mobile Cloud Computing, and its growth over the years.

- Discuss the need for middleware cloudlet and related issues.
- Designing an infrastructure for creating and maintaining a network that has capabilities to host a cloud server, has long life and low resource cost.
- Analyse the benefits of using cloudlet.

We aim for this infrastructure to mainly support distributed mobile applications in a small geographical proximity where mobile phones can share and pool resources, this does not take over many large-scale distributed applications that are supported by clusters today.

We do not aim to do the following:

- Develop a system / infrastructure that is fully ready for real-world deployment. This project is still at a proof-of-concept stage, and will be some way off from a real-world deployment. For example, our implementation only has support for local WiFi, but a real-world deployment implementation could possible support other forms of communications such as 3G mobile networks.
- Develop a system that is intended to completely replace a traditional mobile cloud computing.

1.4 Thesis Contributions

With this work we present a de-centralized Mobile Cloud Computing Server architecture, where mobile devices acts as servers of cloud services. The services are publicly resources and on volunteer basis. We discuss the need of such a design and deliver a scalable, fault tolerant and a stable architecture in a heterogeneous environment.

We propose an architecture (DRAP) for cloudlet middleware, to connect devices in the network and enable communication. It supports system features like scalability, redundancy, backups and trust. We also discuss that how DRAP can be implement to solve Mobile Cloud Computing applications.

We have designed several algorithms based on local information and created a virtual backbone to host a cloudlet. Our algorithm shows energy conservation in 32 out of 36 scenarios, with a maximum savings being upto 91% of radio signal energy.

1.5 Thesis Organization

This thesis is organised as follows: Chapter 2 gives the detailed description of Mobile Cloud Computing architecture, issues and related work. We present our high-level approach and propose our system design in Chapter 3. Chapter 4 describes our algorithms as needed in the system. Our empirical evaluation of DRAP is given in Chapter 5. Finally, we discuss lessons learnt, possible future work, as well as present our conclusion in Chapter 6.

Chapter 2

Background and Related Work

Mobile Cloud Computing integrates the cloud computing into the mobile environment and overcomes obstacles related to the performance (e.g., delay, battery life, storage, and network bandwidth), in heterogeneous and scalable environment. Mobile devices offload task to a remote location and become clients to the services provided while being connected over a wireless connection, refer Figure 2.1. MCC inherently comes with the concerns cloud computing like security, reliability and privacy but has the added characteristics pertaining to the mobility aspect. Over the years different architectures have been proposed by various research groups to effectively bring cloud services to hand-held devices.

This chapter is organised as follows. Section 2.1 gives the well accepted definition of MCC, discusses the concerns of MCC and its architecture.Section 2.2 describes the works where the mobile devices acts as the clients to cloud services and Section 2.3 accounts to a rather newer paradigm where the mobile devices participate in formation of the cloud itself.

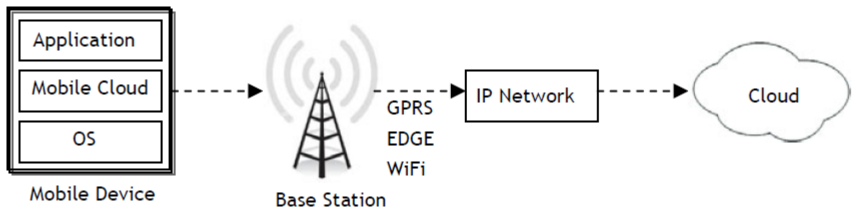


Figure 2.1: MCC Overview

2.1 Background

2.1.1 Definitions

The legal definition of MCC is defined by Mobile Cloud Computing Forum [2] 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"*.

The other two major definitions that correctly describe MCC are as:

"Mobile Cloud Computing is a rich mobile computing technology that leverages unified elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility to serve a multitude of mobile devices anywhere, anytime through the channel of Ethernet or Internet regardless of heterogeneous environments and platforms based on the pay-as-you-use principle" as by Sanaei et al in [55].

Aepona [68] describes MCC as *"a new paradigm for mobile applications whereby the data processing and storage are moved from the mobile device to powerful and centralized computing platforms located in clouds. These centralized applications are then accessed over the wireless connection based on a thin native client or web browser* *on the mobile devices."* Figure 2.2 is graphic illustration of this definition.

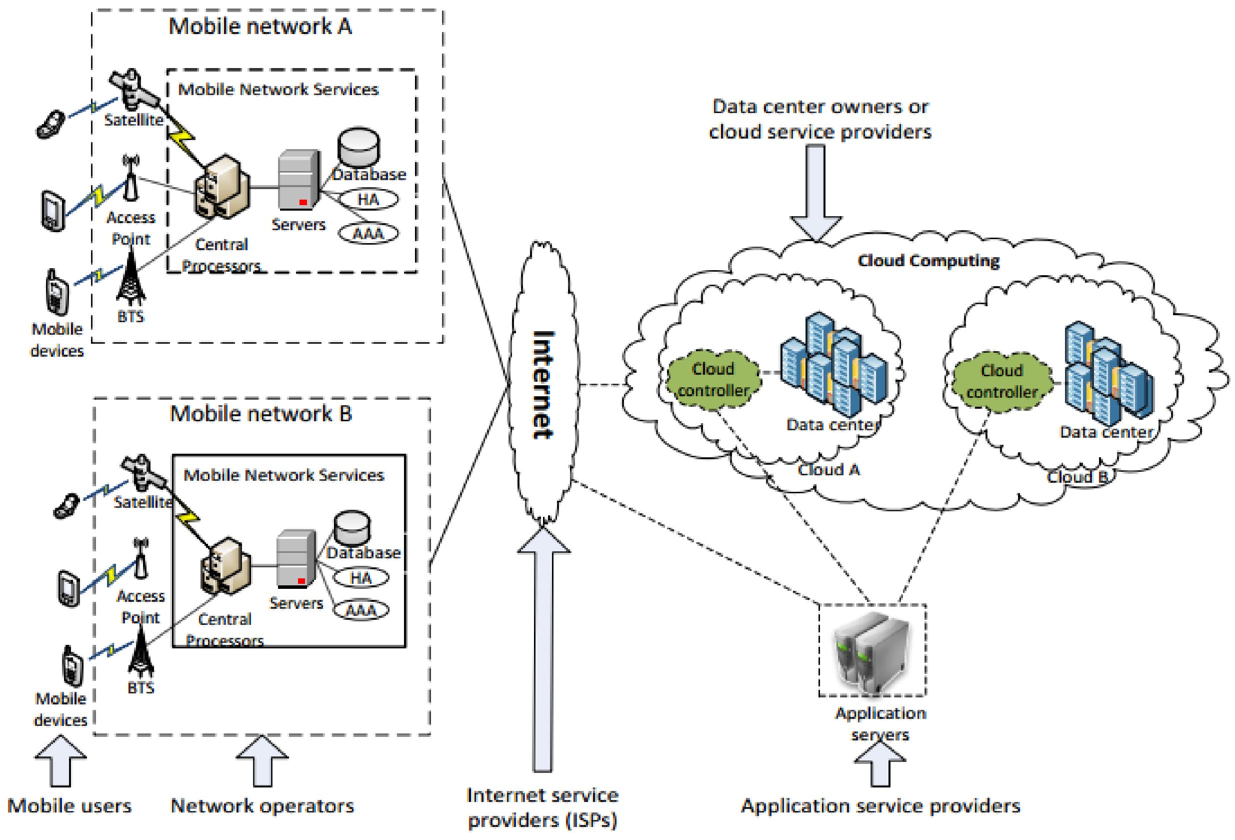


Figure 2.2: Traditional MCC Architecture. Source [25]

2.1.2 Concerns of Mobile Cloud Computing

Cloud Computing faces challenges as that of any other communication model. Different architectures of MCC may have different design concerns and impose different challenges to researchers, but there are some issues are crucial to almost all types of MCC. Here we discuss these matters and outline works aim to address them.

2.1.2.1 Privacy

Cloud storage is catering the ever-present need of more storage of data. Computer users are growing to rely on cloud storage consequently using of large hard-drives to save documents and multimedia is coming to end. This gives rise to a significant concern for cloud computing privacy. All applications which employ cloud computing often store some of the users data remotely. The concerns that companies can use or sell this information as well as concerns that the information could be given to government agencies without the users permission or knowledge.

Specifically with mobile cloud computing, the main privacy concerns come up with location-aware applications and services perform tasks for users which require knowledge of the users location. Instances of applications that finds nearby cash machine for the user, or ones which allows their friends and family to receive updates regarding their location, and vehicle tracking, parcel tracking etc. These applications do the GPS tracking of the user and provide its geographical information to the company providing the service.

Such type of applications simultaneously has broad appeal and brings significant apprehensions. One technique sometimes used to lessen concern is to make data submitted either spatially or temporally imprecise. This is called location cloaking[19]. The cost of location cloaking, of course, is that it can reduce the quality of desired service delivered by the applications. For example, if a user is attempting to search a nearby cash machine and the request sent to the server by his mobile client is too imprecise he could receive results which are irrelevant or possibly miss relevant results. Thus, there is an interest in developing location cloaking methods which manage to alleviate privacy concerns and simultaneously reduce the negative effect on location-aware applications.

Suitable work in this area is suggested in [19], the researchers develop a framework and policies for location cloaking where uncertainty can be controlled to provide high quality and privacy-preserving services . Specifically, they focus their attention on location-based range queries (LRQs), in which a mobile user makes a request for information regarding objects or points of interest within a certain range of their location. They propose a format for an imprecise location-based range query (ILRQ) in which both the location of the user and the location of the returned objects are ambiguous. In addition they present techniques used to prevent trajectory tracking: attempts to infer the future location of users given information about their past locations.

2.1.2.2 Data Ownership

Another issue that arises from mobile cloud computing is the debate is on concept of data ownership. Who owns the data stored in a cloud system? Does it belong to the client who originally saved the data to the hardware? Does it belong to the company that owns the physical equipment storing the data? If a user purchases media using a given service and the media itself is stored remotely there is a risk of losing access to the purchased media. Also, what happens if a client goes out of business? Can a cloud storage host delete the former client's data? How long can the host hold the user data?

With the recent disclosure of the PRISM national security surveillance program 2013 [7] , operated by the United States National Security Agency (NSA), it became clear that US intelligence agencies have direct access to services such as Google, Facebook, Amazon and Microsoft, all of which can be described as being cloud services in one way or another. This accused the technology companies to have worked to undermine the security standards upon which the internet, commerce and banking rely.

The revelations have raised concerns about growing domestic surveillance, the scale of global monitoring, trustworthiness of the technology sector, whether the agencies can keep their information secure, and the quality of the laws and oversight keeping the agencies in check.

In July of 2009 Amazon remotely deleted and refunded copies of George Orwells 1984 from its users Kindle e-book readers [62], suggesting that this concern can develop even when the media is not completely stored remotely. They did this because they discovered that 1984 was not actually in the public domain and that the publisher of that specific e-book edition of the novel did not have the right to sell or distribute it. This provoked uproar among Kindle users and commentators. This action was compared to accidentally selling someone stolen property and then later breaking into their home to retrieve it.

This demonstrates that special precautions need to be taken with MCC to assure that incidents like this do not occur. Users should know exactly what rights they have regarding purchased content. The issues regarding data integrity, loss of data, data storage cost, and related contracts should be well read by the host and clients.

2.1.2.3 Data Access and Security

In addition to issues regarding privacy and data ownership there are the related issues of access and security. Since the idea of cloud related services rely on remote data storage and internet access in order to function at all then this significantly affects the user. If, for example a user stores all of their calendar and contact information online, outages can affect their ability to function from day to day. Such concerns have led some analysts to believe that MCC may be a trend which fails [38].

MCC is particularly vulnerable due to multiple points at which access can be interrupted. Reception and high speed availability can vary greatly for mobile devices.

In addition to this, particular services used may have downtime. Finally, there can be issues of data becoming locked in to a particular service. And cell phones are always vulnerable to being dropped, stolen, or lost, which can wipe out users' stored phone numbers and other information.

2.1.2.4 Resource Management

Issues related to resource management comes inherently with cloud computing because in theory, cloud computing services-based resources should be no different from the resources in your own environment, except that they live remotely. Self-organization i.e. coordinating resources in relation to the involvement of other mechanisms, such as resource replication, load balancer, and failover system is important. Allocating and releasing virtual resources into the available physical infrastructure in response to the starting, pausing, resuming, and termination of virtual resource instances is one of the major topics of cloud computing research. More related issues are creating and managing pre-built instances, such as virtual machine images, enforcing usage and security policies throughout the lifecycle of cloud service instances and monitoring operational conditions of resources.

2.1.3 Architecture

We embrace the broad concept of mobile cloud computing. This research classifies the works done for MCC in these two categories. We also introduce some sub categories to divide the previous works in finer granularity. We believe this is necessary due to two reasons:

1. The general concept of MCC is so broad that it demands a refined classification;

2. Each sub category distinct itself from others based on various attributes that

Table 2.1: Categories

Main Categories	Sub Categories
MCCC: Mobile devices are clients of cloud service	- Ordinary MCC (Cloud is servers through Internet) - Special MCC
MSCC: Mobile devices are both clients and servers in the cloud.	- Crowd computing, Crowd sensing - Crowd Sourcing - Vehicular cloud

will affect many critical aspects of MCC.

Our classification of literatures on MCC is summarized in Table 2.1.

2.1.3.1 Mobile Client Cloud Computing (MCCC)

In MCCC, mobile devices serve only as clients in the architecture, in other words, mobile devices are users of the cloud services. The cloud service provider could be a distant server cluster as in traditional cloud computing applications, or it could be a nearby machine or machine clusters referred by some as surrogate machines. The first case requires an architecture no different from traditional cloud computing. On the other hand, the second case usually has to employ new components to the architecture to address issues specific to it. In particular, since the surrogate machines are usually untrusted and unmanaged, it may require a trusted machine to ensure the security issues and manage resources on the surrogate machines.

2.1.3.2 Mobile Server Cloud Computing (MSCC)

For MSCC, mobile devices may cluster together to perform computing tasks, to collect and process sensor data. In this case, designing architecture becomes very hard and to the best of our knowledge little has been addressed in the literature. However, it is not hard to see the ultimate goal should be an adaptive algorithm to form different architectures on the fly when put in different situations.

2.2 Mobile Client Cloud Computing

Cloud computing is a very general term and can feasibly applied to a many different variety of practices. Mobile Client CLoud Computing refers to any practice where mobile devices are clients of cloud service. This work makes a distinction between ordinary MCC and special MCC. Any mobile device when utilizes the internet in order to make use of a specific resource on-demand manner could be labelled as ordinary Mobile Cloud Client. There are multiple such applications that run on mobile phones today. Also, it has the potential to save mobile client energy by creating systems that can offload a part or the entire computation on virtual machines, referred as special MCC. Other possible ideas exist to utilize different resources a cloud can offer like Dropbox [9] for cloud storage, Cloudinary[8] for image processing, AgroMobile [49] facilitating agricultural world.

2.2.1 Ordinary MCCC

2.2.1.1 Mobile Services in MCCC

Mobile cloud services are deployed to complement a mobile application platform and help accelerate and enrich mobile application development. Mobile cloud services are often a discrete set of high-value middleware capabilities designed to be consumed by mobile applications. Such system are designed to create easy and automatic service configuration to offer services that can be used by mobile clients.

This idea was conceived by [54] that included a collection of low-level facilities that can be either invoked directly by applications (primary proxy) or in the case of disconnection compose more complex services. MSCs will establish transient proxies for mobile devices to monitor the service path, and support dynamic reconfiguration. The advantages of this model are that the model addresses the disconnection issue and can maintain the QoS at an acceptable level.

RESTful Services

Representational State Transfer is an architectural style for large-scale software design. It was term coined by Roy Fielding [29]. REST involves reading a designated Web page that contains easily to parse XML file. The XML file describes and includes the desired content. REST is often used in mobile applications, social networking Web sites, mashup tools and automated business processes. The REST style emphasizes that interactions between clients and services is enhanced by having a limited number of operations. Flexibility is provided by assigning resources their own unique universal resource indicators

Cloud computing services by nature are distributed and evolutionary in terms of users and resources. So, the use of web-based RESTful APIs by consumers is a logical solution for the remote consumption of data processing services. [20] discuss the strategies for implemting MCC applications using RESTful web services. The author claims that by leveraging mobile applications that use these services it will be more easily possible to create solid cloud applications.

Platform-agnostic MCCC

The vision of cloud computing is to provide services related to all aspects of the computer stack. With increasing diversity of service models (XaaS) on multiple mobile device frameworks it is trivial to overlook the demand of agnostic nature both in application development and cloud architecture.

In [42] the authors develop a Domain Specific Language (DSL) based methodology to facilitate agnostic application development paradigm for cloud mobile hybrid (CMH) applications. These Cloud-mobile hybrid as a collective application that has a Cloud based back-end and a mobile device front-end. The DSL is capable of providing abstractions over certain special mobile and cloud functions such as location and power awareness, enabling developers more flexibility and shield from the heterogeneities of each of the platforms

2.2.1.2 Elastic Applications of MCCC

Infrastructure as a service demands resource elasticity in an on-demand fashion. Cloud resources can fadeout if it is overwhelmed with unexpected peak demands. Elasticity can guarantee robust and scalable architectures in general and on the cloud in particular.

Amazon Elastic Compute Cloud

Amazon EC2 [59] provides resizeable computing capacity in the *Amazon Web Services (AWS)* cloud. It is deployed on a framework in which physical devices and hardware, such as servers and storages are virtualized as a resource pool to provide computing storage and network services users, in order to install operation system and operate software application (*Amazon Machine Images*). The virtual computing environments called instances and various resources are called instance types. *Amazon Elastic Block Store volumes* store persistent data in multiple availability zones within firewall enabled *security groups.*

Elastic Weblets

Zhang et al. [74] introduced the concept of weblets as the sub divisions of system that have significant features of portability. The framework supports the idea of elastic applications and is intended to be used on mobile devices. Any weblet can easily be switched between both mobile and stationary devices through network. One major concern the authors deal with is the requirement of security for these weblets. *Trustworthy weblet containers* (VMs) on both device and cloud are build to authenticate, manage and authorise weblets between multiple initiations. It follows an assumption that the cloud manager and application manager is trustworthy.

Such a framework enables a wide spectrum of applications on a mobile device and ease the application develops to create ultra-efficient software for a resource constraint mobile device.

2.2.1.3 Energy Efficient Offloading

MAUI System

Microsofts MAUI (Mobile Assistance Using Infrastructure) [23] is a partitioning approach to save energy. It enables fine grained energy aware code partitioning and VW migration. It uses the benefits of managed code environment maximize energy savings with minimal burden on the programmer. It selects a local or remote server to offload the application on runtime whether or not should the application be offloaded. Also, which method should be remotely executed is decided. It also takes in concern the current connectivity constraints of the mobile device. Developers annotate which methods can be offloaded and at the time of execution, if there is a remote server available. The architecture of MAUI is illustrated in Figure 2.3.

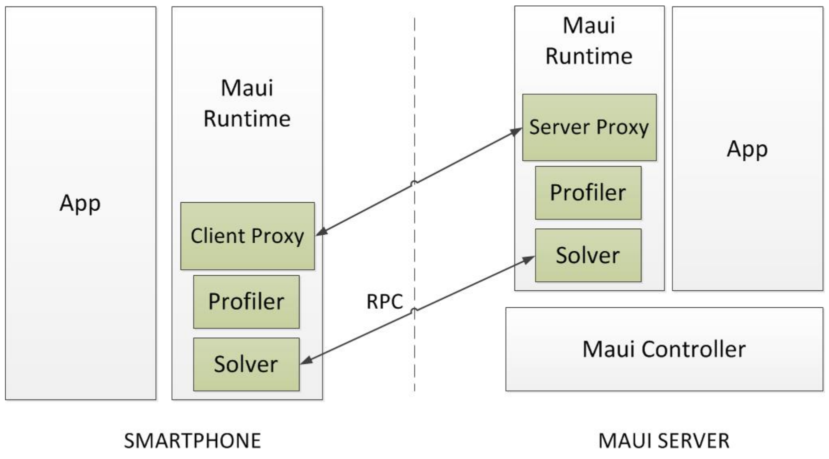


Figure 2.3: MUAI architecture. Source [23]

2.2.1.4 Middleware Frameworks for MCCC

Middleware approach is often suggested to handle the interoperability issues, and eases the use of processing intensive services from mobile phones. They can monitor the context of a mobile device that is connected to a cloud service, and dynamically adapts the services so as to make them more resource efficient, reliable and cost efficient

Giurgiu et al [32], presented a middleware framework that automatically and dynamically distributes several components of an application between a mobile and a serve in order to optimize different objective functions such as interaction time, communication cost, and memory consumption. The framework is developed on a two-step approach to optimally partition the application on the local device and remote server. They propose two types of partitioning algorithms: ALL and K-step. In ALL step the best partitioning is computed offline by considering different types of mobile phones and network conditions. The K-step partitioning is computed on-the-fly, when a phone connects to the server and specifies its resources and requirements. R-OSGi [52] for service discovery and service registry .AlfredO [53] is the deployment tool used that supports the physical distribution of the application between a client (phone) and a server. Notice that the optimization in this work only focused on the client side and is implemented under the assumption that the servers resources are infinite.

2.2.2 Special MCCC

Virtualization of physical infrastructure and migration of software to surrogate hard is special Mobile Cloud Computing.

2.2.2.1 Cyber Forage (Cloud is surrogates 1-hop away)

Balan et al. [14] first introduced the concept of Cyber Foraging: cyber foraging uses opportunistically discovered servers in the environment to improve the performance of interactive applications and distributed file systems on mobile clients. The idea is to dynamically augment the computing resources of a wireless mobile client by exploiting nearby compute and data staging servers. The distinctive nature of cyber foraging is that instead of offloading task to a trusted cloud by means of Internet, it offloads tasks of mobile clients to a surrogate machine within LAN. This attribute poses new and completely different challenges than ordinary mobile cloud computing:

- The surrogate is untrusted and unmanaged, unlike the cloud services in ordinary MCC;
- The surrogate has to be discovered and configured on the fly;
- The surrogate with the most appropriate resources for the task has to be correctly selected;

- It is harder to achieve scalability than in ordinary MCC, since the computing resources within LAN is bound to be limited;
- Data staging may be required to achieve adequate performance.

Despite these new challenges, it is appealing compared to ordinary MCC because the surrogate is usually 1-hop away from the mobile clients, which accounts for shorter latency and larger throughput.

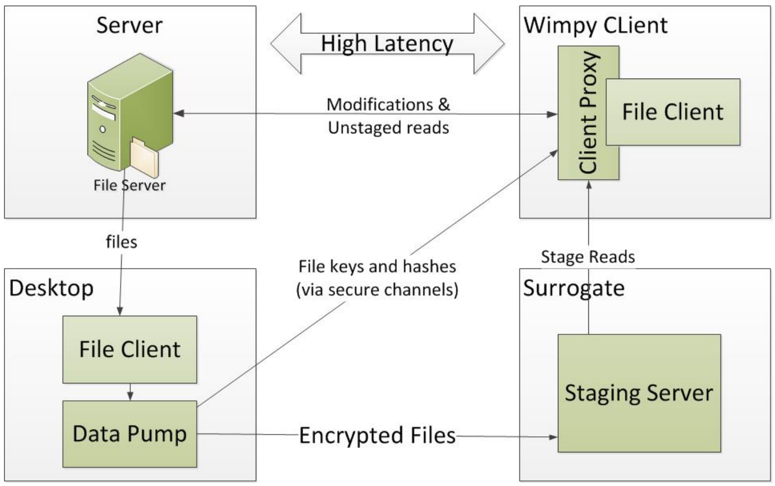


Figure 2.4: Cyber Foraging. Source [14]

We use the concept cyber foraging for all application schemes of augmenting mobile clients computing resources by offloading compute intensive tasks to surrogates within wireless LAN. Besides the original work that brought up this concept, many other works can be grouped in this category. In [14], the authors built an initial implementation of data staging architecture as shown in Figure 2.4. The architecture is split across four computers: the file server, the surrogate, a home desktop machine and the mobile client. The mobile client and the surrogate are located closely while the file server is distant. The request for data is preferably directed to the staging server if the requested data is stored in the surrogate machine; otherwise the request is forwarded to the distant file server. Although rudimentary, this architecture is not only heuristic but also shows one very important result for cyber foraging, which is that untrusted computer can facilitate secure mobile data access. Since the surrogates are untrusted, this architecture uses end-to-end encryption to ensure privacy and secure hashes to guard against malicious modification of file data. The mobile client initiates data staging by specifying a list of files to the desktop machine, the desktop machine then reads these files from the file server, encrypts them with per-file keys, generates a secure hash of the data, and sends the encrypted files to the staging server. It also sends the keys and hashes for the staged files to the mobile client. The mobile client can use the keys to decrypt the files and verifies that it has not been modified using the secure hash.

The authors also built a prototype for using surrogates for remote execution called Spectra [30]. It is claimed that Spectra can monitor the current resource availability and dynamically determines the best remote execution plan for a given application. However, the criteria to evaluate the available execution plans for an application was not mentioned, nor were comparative results provided.

Similarly, Chroma [13] uses pre-installed services reachable via RPC to offload computation, using the history based approach to predict future resource demands. It is developed to improve Spectra be reducing the burden on developers. It uses 'tactics' as meaningful way of partitioning the application code, specified by the programmer. It uses a brute-force method to chose the best or near best tactic.

2.2.2.2 VM-based Surrogates

Goyal and Carter's System:

GnC [33] identifies that cyber foraging is most useful for applications that can be divided into loosely-coupled components; otherwise the cost of communication between the client and the surrogate will offset the performance or energy benefits of using a surrogate. Based on this belief, they rely on application writers to partition applications at development time instead of a scheme for automatic application partitioning. They also make such assumption:

- applications are split in a way that mitigates client-surrogate communication,
- the surrogate is connected to the Internet via a high-speed network,
- pre-existing trust relation between surrogate and client devices.

The system has a service discovery server, where XML-like descriptor is used by the surrogates register themselves. The client sends request to service discovery server and receives the IP address and port number of an appropriate server. Shell scripting has been used for code mitigation, established applying Public Key based Authorization. Form the standpoint of mobile cloud computing, the main contribution they made is using virtual machine technology to implement surrogates, which stays closely in line with the concept of cloud computing. Virtual machine technology allows a single surrogate machine to run a configurable number of independent virtual servers; it also makes isolation, resource control, scalability, cleanup easier. Figure 2.5 shows a typical client-surrogate control flow. As most of other systems developed this work also uses static computers as surrogates, but is a good example on surrogate discovery, remote execution control, and live mitigation of code.

2.2.2.3 CloneCloud

CloneCloud [21] is a history based profiling tool built on the concept of partition of applications to the remote server in cloud and dynamic VM migration.

In essence, this architecture still embraces the idea of offloading tasks to a nearby

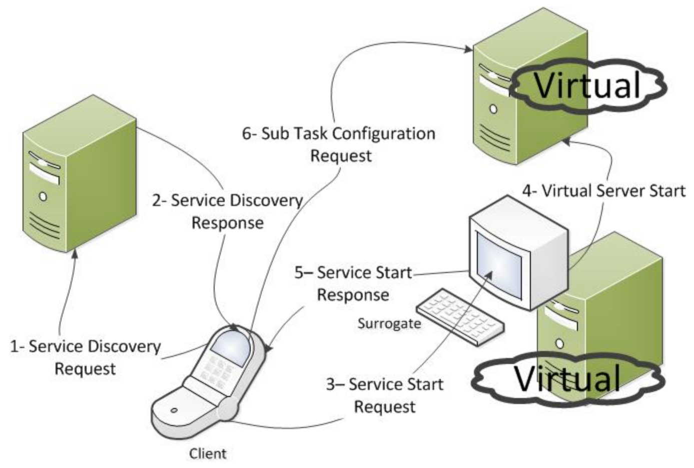

Virtual
6- Sub Task Configuration Request
2- Service Discovery Response
4- Virtual Server Start
5– Service Start Response
Surrogate
1- Service Discovery Request
3– Service Start Request
Virtual
Client

Figure 2.5: Client-Surrogate Control Flow. Source [33]

computer, the novel part compared to previous works lies in that the tasks are executed in a cloned whole-system image of the device. This approach keeps the environment configuration overhead on the surrogate machine to a minimum. Since the smartphone is virtually cloned, it only needs few or no modifications to the applications in order to run on a surrogate. Additionally, since the clone is a whole-system image of the device, it can serve as a backup when the smart phone is lost or destroyed. The high-level system model is shown in Figure 2.6.

It aims to minimize the cost which could be execution time, energy consumption or network usage. It employs a *Dynamic Profiler* to collect data used in the costbenefit analysis, that is then fed in to the *Optimization Solver* to decide which part of the application needs to be migrated, such that the cost of migration and execution will be minimized. It reports test results on three applications; virus scanning, image search, behavior profiling.

2.2.2.4 VM-based Cloudlets

Satyanarayanan et al. [57] proposed VM-based cloudlets. The resource rich computers and computer clusters are called the cloudlets. The idea of cloudlet is analogous to

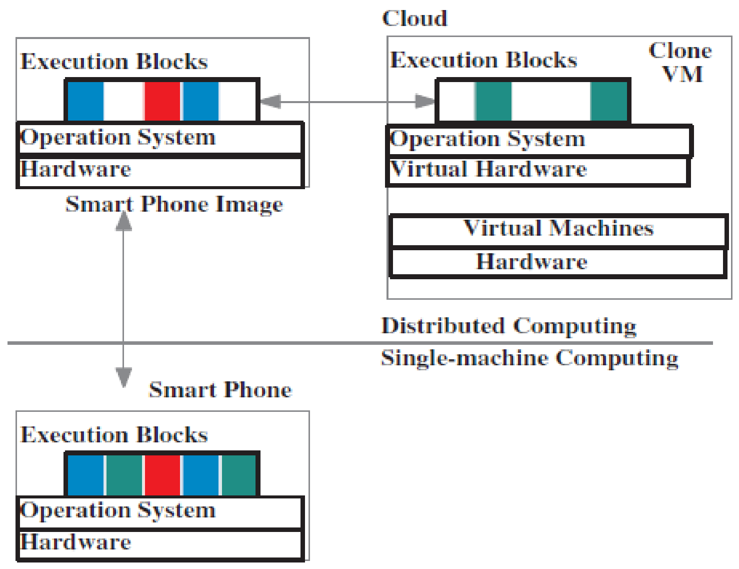


Figure 2.6: CloneCloud System Architecture. Source [21]

the idea of surrogate machines in the original cyber foraging scenario. As cloudlets are within the same wireless LAN with mobile devices, and the mobile user exploits virtual machine technology to rapidly instantiate customized service software on a nearby cloudlet.

The novel contribution of this work is that the authors identified rapid customization of infrastructure for diverse applications as a critical requirement of this architecture, and they presented a proof-of-concept prototype that suggests this requirement can indeed be met through VM technology. This is done by dynamic VM synthesis. Before offloading tasks to the cloudlet, the mobile device first transmits a small VM overlay to the cloudlet, the cloudlet is supposed to possess a majority of the popular base VMs, from which the VM overlays are derived. The cloudlet infrastructure applies the overlay to the base to derive the launch VM, which starts execution in the precise state from which the overlay was derived. Figure 2.7 shows the timeline for dynamic VM synthesis.

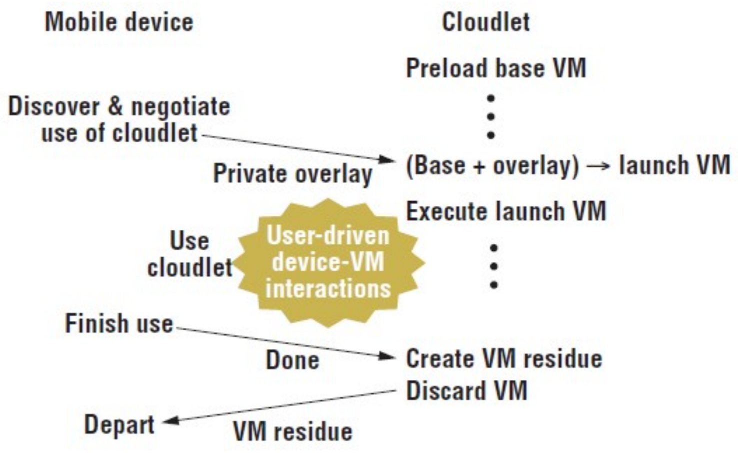


Figure 2.7: Dynamic VM Synthesis. Source [57]

2.2.2.5 Social MCCC

With the increasingly abundance of Social networks and ubiquity Cloud computing, consumers are looking for different ways to explore and interact with these evolving paradigms. Social networks are used to imitate real world relationships and allow users to share information and form connections between one another, essentially forming dynamic Virtual Groups.

Social Cloud [18] is a dynamically formed cloud leveraging the notion of trust formed through pre-established friendship with a social network. This enables resource sharing within social network groups. It is pointed out that linking trust relationships with appropriate incentive mechanisms (through financial payments or bartering) could provide very efficient and sustainable resource sharing mechanisms. It is suggested that evolution is deep and spread to all aspects of society (infrastructure, services, etc.). Dynamic and unpredictable global demands could be reached by such mechanisms of partnership.

2.3 Mobile Server Cloud Computing

So far in this work, we have only discussed instances of MCC where the mobile device serves as a client and some collection of non-mobile devices act as the server, the provider of resources. It is possible to invert this pattern and let mobile devices serve as the resource rather than the consumer. This scenario will called mobile server cloud computing where mobile devices are both clients and servers in the cloud.

2.3.1 Crowd Computing & Crowd Sensing

The long held dream of ubiquitous computing [67] is closer to be realized than ever due to the abundant availability of sensors and advanced mobile devices like the smart phones. Along with the Internet of Things, another trend that fuels the development of ubiquitous computing is the so called crowd sensing and crowd computing.

Many papers proposed similar concepts termed as *crowd sensing* [31], *crowd computing* [46] and *crowd-sourced sensing* [15]. The general idea of these propositions is to take advantage of the computing/sensing abilities of mobile devices, preferably combined with human interaction or intervention.Such systems could be very useful to achieve great parallelism and accomplish tasks by utilize such parallelism. Consumer-centric mobile sensing and computing devices, such as smart phones, music players, in-vehicles sensors, will feed sensor data to the Internet at a societal scale. Besides, given the potential of adapting powerful processing, storage and communication capabilities, these mobile devices can either serve as a bridge to other everyday objects, or generate information about the environment themselves. [46] analyse encounter traces to place an upper bound on the amount of computation that is possible in an opportunistic network of mobile devices. They also investigate a practical task-farming algorithm that approaches this upper bound.

Humans are usually involved in the loop. On one hand, the intelligence and mobility of humans can be leveraged to help applications collect higher quality or semantically complex data that might otherwise require sophisticated hardware and software. On the other hand, humans naturally have privacy concerns and personal preferences that are not necessarily aligned with the end goals of the applications. Also, the so called crowd sensing/computing is most likely undertaken in a participatory manner, which implies incentives scheme has to be devised, especially when active involvement of individuals is required, such as taking pictures. To the best of our knowledge, no thorough and dedicated discussion of this issue has been made in the literature.

2.3.1.1 Crowd Computing System of Heterogeneous Mobile Device

MapReduce System

Googles MapReduce [35] is an algorithm or a framework to process larger problems into smaller clusters that can be solved in a distributed parallel environment with multiple machines. Inspired by functional programming the MapReduce System has two procedures namely Map() and Reduce(). Map() performs the data processing by filtering and sorting, while Reduce() performs the summarization operations. It is a promising to leverage the large number of smart mobile devices connected to the internet. The limitation on resources on an individual device can be compensated for by the relatively small size of many of the tasks. [26] have created a system for controlling mobile networked devices to solve problems. It consists of several segments called the heterogeneous mobile device MapReduce system Figure 2.8

- master nodes which receives jobs, distributes them to nodes, aggregates results and returns the results.

- a client for a mobile device which receives works on and transmits solutions to sub-problems
- a browser interface which allows the user to submit problems and view results.

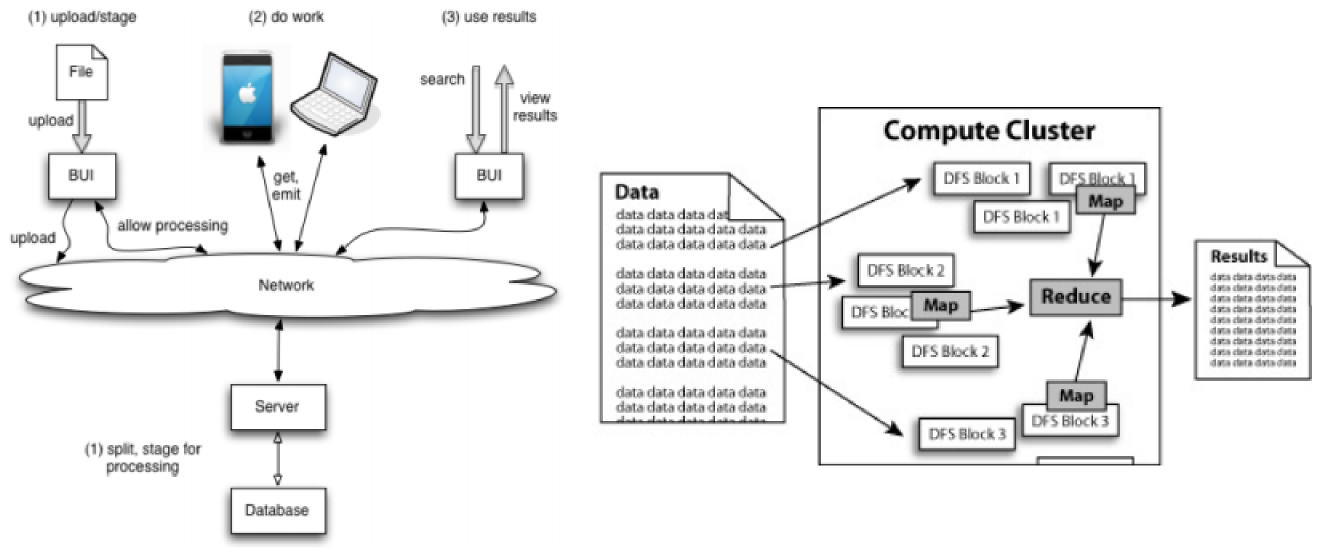


Figure 2.8: Map Reduce Architecture. Source [35]

2.3.1.2 Social Crowd Computing

Community-Based Crowd Computing

[46] shows an interesting result of crowd computing, which is social awareness can greatly raise the performance of crowd computing, in other words, achieve greater parallelism. This may point researchers to interest directions such as how to best exploit the community structure/attributes to achieve greater parallelism.

Twitter-Based Crowd Sensing

[15] takes an interesting approach for doing crowd-sourced sensing and collaboration, which is to utilize Twitter as the infrastructure to task/utilize the mobile devices.

The wide popularity and the enormous community behind Twitter are invaluable.

2.3.1.3 Crowd Sensing System

PRISM

Platform for Remote Sensing using Smartphones (PRISM) [24] describes a procedural programming language for collecting sensor data from a large number of mobile phones. It caters to the interconnected goals of generality, security, and scalability in MCC. PRISM allows application writers to package their applications as executable binaries, which offers efficiency and flexibility of reusing existing code modules. PRISM defines a push module that can push the application on the set of phones automatically on specified set of predicates. It enables timely and scalable application deployment while safeguarding a good degree of privacy.

2.3.2 Crowd Sourcing

2.3.2.1 Image Search

CrowdSearch

CrowdSearch [72] is an accurate image search system for mobile phones that is implemented on Apple iPhones and Linux servers. Automated image search is performed using a combination of local processing on mobile phones and backend processing on remote servers. Human validation is also performed using Amazon Mechanical Turk [16], where many of people are actively working on simple tasks for incentives (monetary rewards). Designing the models of the delay-accuracy cost behavior of crowdsourcing users, the work developed a predictive algorithm that determines which image search results need to be validated, when to validate the tasks, and how to price them. The algorithm dynamically makes these decisions based on the deadline requirements of image search queries as well as the behavior observed from recent validation results.

2.3.2.2 Information Service

Real-Time Trip Information Service

[12] builts a real-time trip information system in cooperation with a taxi operator that operates more than 15,000 taxis in Singapore. It describes an efficient algorithm to predict the trip time (accurate with mean error of 3 minutes) and taxi fare (accurate to approx. 0.76 USD). The optimization takes in account the region size, routes, weather conditions, history upto 21 months and data mining techniques. It describes the design, testing, iterative improvements, and real-world deployment of the system for providing passengers with accurate predictions.

2.3.2.3 Application Discovery

AppJoy

AppJoy system [71] is a recommendation system. It is designed to recommend applications by analyzing the installed applications on a system and its use, in a personalized approach. It is an understand approach based on all participants' application usage records. AppJoy employs an item-based collaborative lettering algorithm for individualized recommendations.

2.3.2.4 Decision Support

SignalGuru- Collaborative Traffic Signal Schedule Advisory

SignalGuru [39] is software service that leverages mobile phones detection for collaborative trafc signal schedule predictions and advisory. Its enables Green Light Optimal Speed Advisory (GLOSA), a traffic signal advisory to change signals based on such inputs. Also it can suggest an efficient detour that will save the drivers from stops and long waits at red lights ahead. SignalGuru leverages wind-shield mounted phones to opportunistically detect current traffic signals with their cameras, collaboratively communicate and learn traffic signal schedule patterns, and predict their future schedule.

2.3.3 Ad-Hoc Mobile Cloudlets

Satyanarayanan et. al [57] suggest the idea of using a nearby infrastructure rather than relying on the distant cloud. This infrastructure of a resource-rich computer or group of computers in the vicinity of client is called cloudlet. This initiated the idea of ad-hoc cloudlets where an infrastructure could be created in an ad hoc manner with mobile devices exposing their computing resources to other mobile devices. These cloudlets can offer services in cases of no or weak connections to the Internet and large cloud providers. Offloading to nearby mobile devices save monetary cost, because it largely cuts down on bandwidth usage. Moreover, it allows creating computing communities in which users can collaboratively execute shared tasks.

2.3.3.1 Interactive Applications

Fesehaye et al. [28] demonstrated how decentralised cloudlet infrastructure can lower data transfer delay and higher content delivery throughput than the cloud-based approach or the centralized cloudlet. They created experiments for file editing, video streaming and collaborative chatting in a distributed and in centralized routing schemes. In distributed the routing table was constructed and maintained by the cloudlets. The cloudlet as well as mobile node maintains specific tables to store user ID and cloudlet ID respectively. Whereas, in centralized routing, the central server was responsible for constructing and maintaining the routing table. The central server computes the routing table for each cloudlet and installs the forwarding tables into the cloudlets.

The results were under the assumption that the WiFi transmission range is larger than 250m restricts the system to work on technologies like Wi-Fi repeaters to achieve a desired coverage.

2.3.3.2 Hadoop Based

Hyrax [43] is developed on a centralized scheme by porting Hadoop Apache [60] on the master node, an open-source implementation of MapReduce, to Android smart-phones. Hyrax allows computing jobs to be executed on networked Android smart phones. The performance of Hyrax was poor compared with Hadoop on traditional servers.This is because the smart phones were much slower at that time, and also Hadoop was not originally designed and optimized for mobile devices. The systems assumes that the master node cannot fail, since it is a single point of failure for the entire cluster.

In [34], the authors describe a a discovery mechanism to find the nodes in the vicinity of a user wishes to use a local cloudlet instead of the main distant cloud. The application target is set once the system detects nearby nodes that are in a stable mode forming surrogates. Hadoop[60] is used on mobile device to distribute the processing tasks and storage. Communication is based on the Extensible Messaging and Presence Protocol (XMPP). The tasks are executed on a virtual machine acting as a protected space thus ensuring the security of device data. The implementation was tested using a OCR application.

The system does not consider the mobility, capability of devices, and privacy of neighbouring nodes. The authors suggest that the system saves energy and processing time than when executed on a single mobile device.

2.3.3.3 Component Based

Verbelen et al. [66] propose breaking down applications into components and offloading components instead of entire virtual machines. They describe an execution environment in which devices can join, share their resources and form a network Cloudlet. This suggest that this reduces the data transfer necessary to execute a distributed job, and allows greater control over how the application is divided among the available resources. However, the current approaches only focus on a single Cloudlet or resource aggregation and cooperation mechanisms among different EE but neglect the collaboration mechanism and synthetic model between Cloud center and Cloudlets. This approach also needs application developer's annotations to create or modify the application into a distributable component design.

2.3.4 Vehicular Cloud

[41, 47] proposed the concept of vehicular cloud, aiming to bring VANET and cloud computing together. We include vehicular cloud in our survey because we treat vehicular cloud as one special case of mobile cloud computing, since vehicles are also mobile agents. We also believe the research on vehicular cloud has great implication on crowd computing and crowd sensing, which also fall into the general concept of

mobile cloud computing. Last but not least, vehicular cloud is greatly motivated by the belief that vehicles on the road can network together to heighten drivers awareness of potential traffic safety risks, such as hazardous road conditions ahead, reckless drivers who share the same road, traffic accidents, etc. Thus the research of vehicular cloud has great significance since human life is concerned; this no doubt will give the research of this subject some extra momentum. Although promising as it seems, we believe vehicular cloud has a few critical issues to be resolved before it can be accepted and deployed.

2.3.4.1 Architecture of Vehicular Cloud

The first problem is the architecture of vehicular cloud. We believe for vehicular cloud to be widely accepted and adopted, the architecture of VC should be flexible. In other words, vehicles should form different types of networks among themselves in different situations. Although various types of road side infrastructures have been deployed, they are far from ubiquitous, which implies that when forming a VC, existence of road side infrastructures has to be taken into consideration. The simplest case of VC would be when all the vehicles are static, e.g., in a parking lot at the airport or shopping mall. In this case, the vehicles will simply form an ad-hoc network, and behave just as a conventional cloud. In a mobile environment, things change drastically and designing architecture becomes much more challenging. When infrastructures are presented, it is self-evident that the VC benefits from the interaction with the existing static infrastructure. In such a case, the road side infrastructures behaves like mesh routers in a wireless mesh network where mesh routers have minimum mobility, and vehicles behave like mesh clients, which can be static or mobile. Without infrastructures, mobile vehicles will form an opportunistic network, because of the high mobility of vehicles, the existence of a route from a chosen source to a destination at any given time cannot be safely assumed.

2.3.4.2 Privacy & Security of Vehicular Cloud

Secondly, we consider the privacy, security issues of vehicular cloud. In order for the VC vision to become reality, the problems of assuring trust and security in VC communication needs to be addressed [51]. It is highly possible to jeopardize the benefits of Vehicular Clouds as its services require a set of rich and diverse tools those are vulnerable to hacks. The self-organizing property makes it a formidable prospect that may make anti-social and criminal behaviour easier.

Given the fact that connectivity among vehicles can often be highly transient and a one-time event makes security even more challenging task. Traditional security methods implementation like password-based establishment of secure channels, gradual development of trust by enlarging a circle of trusted acquaintances becomes impractical for securing VC. For example, two vehicles travelling on a highway may remain within their transceiver range, or within a few wireless hops, for a limited period of time.

It has been discussed that scalability is one key requirement for providing cloud services. This poses another issue is in VC. Due to the huge number of vehicles registered in different countries and travelling long distances, well beyond their registration regions. The involvement of authorities in vehicle registration implies the need for a certain level of centralization. Communication via base stations is not enough, mainly because vehicles need to authenticate themselves not only to base stations but also to each other, which creates a problem of scalability.

2.4 Summary

Cost of using a cloud before real execution is an important metric in cloud computing. It measured in terms of processor cycles, memory size, storage size, input data size, communication traffic rate, execution time of the chosen surrogate. Decision to offload may depend on parameters like reduce energy consumption and latency, and increase performance. The major issues covered in literature are:-

Offloading could be **static or dynamic** in nature. Static offloading refers to code migration before start-time as implemented by Spectra [30], Chroma [13], GnC's [33]. Dynamic offloading is supported by MAUI [23], Huerta's [34] which refers to the offloading decisions at run time. This is more flexible as decision to offload depends on current conditions of the host and surrogate. However it creates overheads related to profiling and latency.

The trends in the research pattern favours to **VM migration** or offloading of Mobile Code over more traditional **client-server communication system**. Hyrax [43] and Huerta's [34] are only two projects based on grid and distributed programming that uses the pre-installed Hadoop framework. MAUI [23], CloneCloud [21],Kimberley [57] favour VM migration as it does not require re-writing the application and it also supports disconnected operations. Major disadvantage of the latter is the on going interactions and communications between the server and the client that may lead to network congestions.

Input data size relates to offloading granularity. **Coarse-grain** partitioning is when the whole application is mitigated to the surrogate machine, as in GnC's [33], Kimberley [57]. **Fine-grain** granularity is achieved when a part of the application is offloaded. It leads to large energy savings and less network bandwidth use. Systems like Spectra [30], Chroma [13], CloneCloud [21](Thread Level), MAUI [23](Method Level), Verbelen's [66](Component Level) use fine-grain strategy.

Two main aspects not dealt in an efficient manner are decentralized approach to surrogate discovery and fault tolerant mechanism to host applications on a distributed environment with multiple cloudlets. While most systems work on the idea of single surrogate, Chroma [13] incites the idea of multiple surrogates but does not deal with the possibility of fault occurrence. Also, Goyal and Carter's system [33] deals with the issue of surrogate discovery in a centralized approach. All surrogates register themselves to a service discovery server using an XML descriptor file.

As suggested by Fernando et al. in [27] a major challenge in mobile cloud is a decentralized solution to trade off between the possibility of energy drain with continuous scanning, opposed to missing out on encountering potential resources. Sanaei et al. [55] while talking about Mobile communication congestion issues suggests that a cognitive system within MCC, that likely pre identifies congestion issues and considers factors like MCC application types (e.g. data-intensive, computation-intensive,and communication-intensive), to determine the best actions to relay traffic, is imperative as a future research direction.

Chapter 3

Design of DRAP

In this chapter, we describe the requirements of a cloudlet, our proposed architecture for discovery and collection of services needed to host a server. We further describe our system features and network needs.

Our aim is to design a model that is scalable with reliability and high performance. These capabilities should be provided at relatively low costs compared to dedicated infrastructures like distant clouds servers or central servers for cloudlets. The cloud services come with the cost of huge bandwidth and pay-per-use service model, also central systems usually have a disadvantage of a single point of failure. With DRAP, we achieve cost benefits with respect to bandwidth, save radio signal energy and also remove any single point of failure. All the services with DRAP are public-volunteered, hence no service cost involved.

Further, we ascertain that DRAP has the potential to host mobile cloud computing applications locally with performance equivalent to distant clouds.

3.1 System Requirements

Cloud server may vary in resources depending on the categories of service it offers, like database as a service, platform as a service, etc. For hosting any cloud-like environment, the server has needs which may or may not be relevant to every cloud application. Following are the requirements of a cloud server on a mobile network:

- Storage: Cloud Storage requires maintaining, managing and creating backups of user data over the network. To the best of our knowledge, no significant research has been done to provide Cloud Storage services through mobile servers. This is mainly because mobile devices often have very limited storage abilities.

- Computation: Cloud servers require high-end processing units with unlimited capacities and an always-on service. Researchers [14], [23], [43], [57], [59], [66] have been looking into this area and have proposed different solutions for the same.

- Scalability: Servers should have the ability to adapt to workloads expanding and shrinking its services in an autonomic manner. It should at each point in time be able to supply resources to match the current demand as closely as possible.

- Response Time: While offloading tasks to distant servers, application response time is often a performance criteria. With cloudlets, we achieve faster execution by reduction in network latency due to one-hop information exchange.

- Connectivity: Mobile cloud computing relies on connectivity via wireless, this poses a major architectural threat to the system. In DRAP, we assume that all nodes are capable of intra-network (near field) communication. Also, a few nodes are expected to have internet connectivity.

- Security: In a heterogeneous mobile environment trust and security are open problems, DRAP includes a trust mechanism to deal with the same.

- Life-time: The cloud services must remain viable throughout and maintain a long life. In a mobile ad-hoc infrastructureless environment it is very difficult to predict the lifetime of the network. Network life-time can be inreased by saving radio signal energy, as it reduces node die-out.

- Fault Tolerance: Owing to the unstable nature of mobile networks, the system should tolerate the moving of nodes, breaking of communication links, unavailability of data server and system failure.

- Scheduling & Load Balancing: In order for a cloud to be truly on-demand and elastic while steadily meeting consumer needs, the cloud must be load- and resource- aware. This becomes even more critical in unexpected hit peak demand to the system. The system must be able to dynamically prioritize its tasks and resources on-the-fly based on priorities of the various workloads to ensure satisfactory service.

DRAP is based on general needs determined by the cloudlet environment. Different applications have varied needs in terms of size, speed and computational power. For example a crowd sourced application has huge dataset but does not require high response time sensitivity or complex computations, whereas image processing application will have small dataset, medium response time sensitivity but would need large computational power. Table 3.1 is a list of some of the cloud based applications and their possible resource requirements, this table is re-written from Soyata et al. [61]. In Chapter 4, we explain how application specific resource needs can be incorporated in the system and resource values can be set while creating the cloudlet.

Table 3.1: Cloud-based applications and their resource requirements

Application	Description	Database Size	Compute Resources	Time Sensitivity
Battlefield	Assist soldiers in the battlefield through real-time object recognition	HIGH	HIGH	HIGH
Natural Language Processing	Perform real-time speaker or speech recognition	LOW	MEDIUM	MEDIUM
Airport	Conduct real-time face recognition of known criminals	HIGH	HIGH	HIGH
Fire	Assist fire fighters with disaster in real-time	MEDIUM	MEDIUM	MEDIUM
Medicine	Accelerate medical research (e.g., recognizing DNA sequences in real-time from a microscope while the research is in progress)	HIGH	MEDIUM	MEDIUM
Archaeology	Recognize archaeological structures in real-time while the researchers are at the search site	HIGH	LOW	LOW
Surgery	Recognize issues (e.g., tumors) in real-time from a cloud-based medical database while the surgery is in progress	HIGH	MEDIUM	HIGH
Amber Alert	Identify criminals by searching through the FBI database to match a photo taken by a camera	LOW	LOW	MEDIUM
Social Network	Profile online users by searching through a database for marketing purposes	HIGH	LOW	LOW

3.2 DRAP Architecture

Every device that is a part of the cluster where we intend to host the cloudlet should have a system that makes the participating devices communicate with each other and also serve as a middleware between the application level and the Operating System. The DRAP middleware should fulfil requirements in following broad categories:-

Device Manager To detect the unused resources on the device and offer the same to the cloudlet, to decide whether or not it can serve the incoming requests depending on the current resource it holds.

Neighbour Discovery To be able to interact with other devices in the neighbourhood, every device should have a discovery mechanism that can detect the presence of another device and identify the service it can provide.

Cloud Controls A device can have many roles in the network, we need controls to manage them effectively. Tools to provide cloudlet service and perform task distribution needs to be setup while ensuring performance quality and initiating backups.

Heterogeneity The design of the middleware should be such that it is capable of providing services in a heterogeneous network. The network could be diversified in terms of device related (type/model, mobile OS) and application related issues.

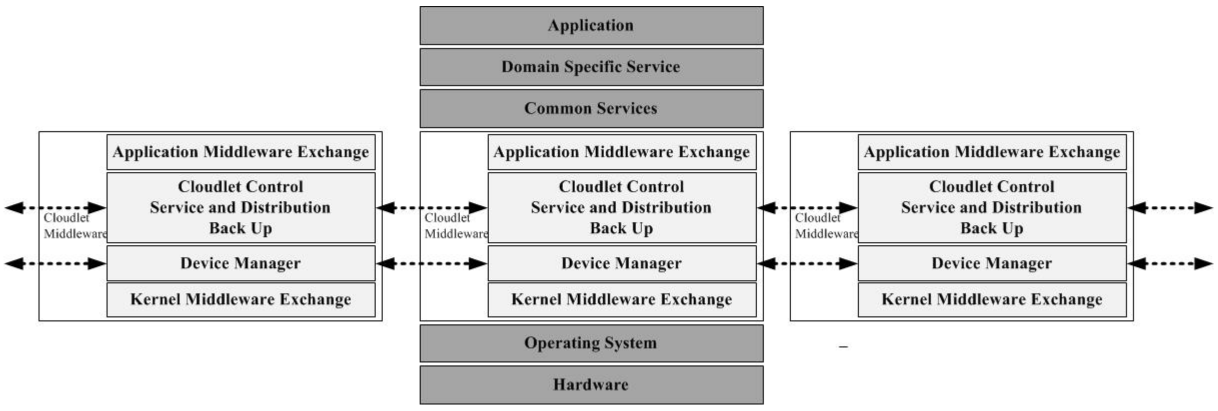


Figure 3.1: DRAP Overview

We propose such a middleware, that can be installed on the Operating System of a device and should be able to serve the applications running on the device. Figure 3.1 is the architectural overview of DRAP. The cloudlet middleware has four layers. Figure 3.2 demonstrates the component level design, grouped by the layers.

Layer 1 is **Kernel Middleware Exchange**. To design a 'write once run anywhere' middleware, we need an exchange layer that supports multiple Operating Systems. This layer is expected to re-configure the DRAP middleware according to the OS on the specific device. This cross-platform layer is responsible to adapt to the technology on the device and facilitate the cloudlet activities by providing kernel related information and perform tasks. Application files cache cleaning, CPU scheduling are some cloud activities that need OS support.

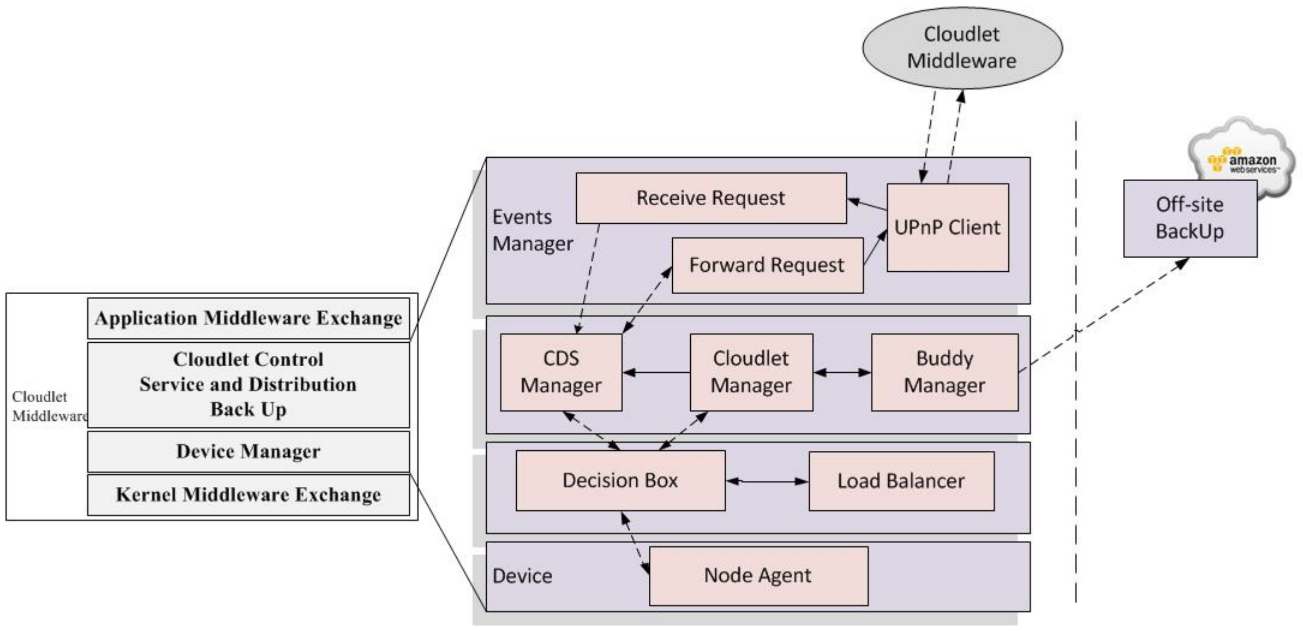


Figure 3.2: DRAP Architecture

Layer 2 is **Device Manager**. It is a set of tools that provide device support to the cloudlet through the **Node Agent**. It contains the information about the available resources, and data from different sensors on the device. **Load Balancer** acquires the information regarding the application presently running on the device and decides if it has enough resources to offer to the cloudlet. This information is then passed to the **Decision Box** which may accept or reject a service request made to the node. It is also the medium to create request in case the node agent is a cloudlet client.

Layer 3 is divided into 3 parts. Part 1 is the **Request and Service**. It allows the device to discover, communicate and initiate services between neighbouring devices. It receives/sends service requests from/to other devices. Part 2 is **Cloudlet Control**. It includes the CDS (Connected Dominating Set) manager, Cloudlet Manager and Buddy Manager. **CDS Manager** keeps information about the neighbouring devices, nearby resources and locations. **Cloudlet Manager** manages the activity the device contributes to in the cloudlet. It contains information related to the tasks being executed in the cloud. **Buddy Manager** logs the activities of the cloudlet, detects node failure. These three managers may or may not be active for every node at all times, the behaviour depends in the roles a device plays in the network. Part 3 is **Backup**. It is performed by the Buddy Manager. It initiates backup process for the cloudlet data if enough resources are not available in the network.

Layer 4 is the **Application Middleware Exchange**. This layer contains information regarding the resource needs specific to the hosted application. It would decide whether or not the application needs the cloudlet support/offloading. It can also act as the application server. The decision of this layer acts as the initiation of service discovery in the DRAP cloudlet.

3.3 DRAP Features

In this section, we describe the issues handled for efficient hosting and maintenance of cloudlet.

3.3.1 Fault Tolerance

DRAP should be tolerant to node failures, in other words after if a working node fails or moves out the network we should be able to recover the relevant data and restart the execution of the application in an efficient manner. In the Layer 3 of the architecture, we introduce a Buddy Manager. Cloudlet surveillance tasks are performed by the Buddy Managers. They have the following duties:

- perform computing operations parallel to the cloudlet node,
- store information upto a certain level of redundancy, so that in case of node failures data can be recovered or re-constructed,
- health checks cloudlet node; checking error logs, load levels, communication links,
- backup data to off-site servers
- share information with other Buddy Managers in the network, such that they can collaborate to detect errors and failures.

The level of redundancy expected in a DRAP is decided by the application hosted on the cloud and also the network itself. The applications that carry critical data or the data created after lot of computation needs higher fault tolerance. For example, history based prediction systems cannot afford to loose saved data. And, if the network is highly mobile, the system would require higher levels of redundancy because higher mobility leads to links break and thus the probability of loosing data increases.

The number of buddies available to every cloudlet node depends on the current resource pool of the network, density and dynamics. If the system requires high redundancy levels, it is highly risky to assign task to a cloudlet node that does not have enough buddies to back it in case of a failure.

3.3.2 Backup

Owing to the constraints of connectivity and chances of system's dysfunctional behaviour, the DRAP is subjected to periodic off-site system backups. We propose the use of Amazon Web service offering Elastic Compute Cloud [59] as a trustworthy resource backup system.

When an application requests cloudlet service, it signs up for AWS-EC2 services as it would do in case of no cloudlet. An Amazon Cloud 'instance' is started with each session, with the intent to overtake the execution of the local surrogates if the latter fails to provide service. During the normal cloudlet condition, the buddy nodes supporting the cloudlet periodically sends checkpoint data and task logs on the off-site cloud. In case of partial or complete failure of cloudlet,that is when the environment does not have enough resources to form a cloudlet, all requests from the application are directly send to the Amazon EC2. This gives a sense of reliability to the system.

3.3.3 Scalability

The nodes should be able to join and leave the network without any pre-requisite. The communications are set up on the fly with very low cost. The algorithm described in Chapter 4 deals with the extension of cloudlet by the addition of nodes and shrinking cloud size by nodes leaving.

Local cloudlets in proximity of each other should be able to communicate and share resources. Since we do not support any central system that should be held responsible for this communication it is necessary to have bridge nodes, that is one or more nodes that are common to both networks.

The distribution of duties is done in a distributed manner. With DRAP, we construct an overlay structure in the network by connected nodes set to expand throughout the broads of the network. Every node on this set is given the responsibility to watch its neighbours. The maintenance is done on a sub-net level but the connected nature of the dominating set binds it together to allow scalability across the network.

3.3.4 Trust and Incentive Mechanism

Public resourced clouds should have an incentive mechanism to encourage volunteers to offer service. This will boost the participation in the cloud, hence would bring out the true potential of the such a model. A formal credit-based mechanism should be implemented, where credits can be earned with nature and amount of service offered. The kind of application hosted should be deciding factor for the monetary or non-monetary benefits to the volunteer.

Marias et al. provide a comprehensive comparison of the important cooperation enforcement schemes in manets[44]. The token based scheme presented by Yang et al. in [73] well suits the requirements of our system as it works on promiscuous mode of the participating nodes and has a self-organized approach by exploiting full localized(one hop) design, without assuming any *a priori* trust or secret association between nodes. A system's secret key is distributed to the nodes in the network, and every node has a token signed by this key. Token has a validity period, after which it has to renew it, through its neighbours. It allows node verification and monitoring, and allows a fair node to collect credits and to renew its token less frequently. This trust relationship can be linked with social network groups to develop incentives. Social Cloud [18] leverage Facebook credentials to authenticate users, and then post prices and auction their resources.

3.4 Network Support

An ad-hoc mobile network is an infrastructureless network of mobile devices on wireless connectivity. The links between nodes change rapidly due to the freely moving nodes independently in any direction with any speed. As mobile ad-hoc networks are fundamentally different from wired/traditional networks due to its dynamic nature, the system needs a completely decentralized routing protocol and the participating nodes should be connected in an ad-hoc manner.

Since DRAP deals with mobile ad-hoc network, we now explain the apparent needs of the system.

3.4.1 Wireless Connectivity

Conventional WLAN networks are of two kinds, infrastructure mode and ad-hoc mode. Typically infrastructure mode is based on the presence of controller devices known as wireless access points. These devices combine three primary functions; physical support for wireless and wired networking, bridging and routing between devices on the network, and service provisioning to add and remove devices from the network. On the other hand ad-hoc is a way to have a group of devices communicate with each other wirelessly, without a central controller i.e. with no inherent relaying. This is as described in IEEE 802.11 [22] wireless ad hoc networks.

In such a setting links are influenced by the node's resources (e.g., transmitter power, computing power and memory) and behavioral properties (e.g., reliability), as well as link properties (e.g. length-of-link and signal loss, interference and noise). Since links can be connected or disconnected at any time, a functioning network must be able to cope with this dynamic restructuring. Another Wi-Fi standard, Wi-Fi Direct [10] enables devices to connect easily with each other without requiring a wireless access point and to communicate at typical Wi-Fi speeds for everything from file transfer to Internet connectivity. It allows pairing of Wi-Fi Direct devices in proximity to enable near field communication.

With DRAP, all nodes have to be connected in an ad-hoc mode. We also need the nodes with active Buddy Agents to have internet connectivity to be able to communicate with Amazon Web Services for backup features.

3.4.2 Intra-cloudlet communication

Strong inter-node communication is needed to set up a working cloudlet. Information related to location, neighbourhood and resources is needed to host a server, and efficient communication links should be established to request services and avail them using DRAP architecture.

In this work, we use a dominating set based Routing Protocol. We have modified Ad-hoc Distance Vector Protocol [48] to create a Connected Dominating Set(CDS).

Connected Dominating Set of any topology is a collection of reliable nodes forming an overlay network. Constructing a virtual backbone in the network by using a CDS has been well researched a topology control mechanism [36]. Dominating set based routing is often considered an organised way to reduce control messages overhead [17]. It is also an efficient way to reduce multicast/broadcast related issues like receiving same message multiple times and wastage of bandwidth [64]. Saving on the number message passing between nodes also contributed to power management, and conserving energy of the nodes [70].

AODV is a reactive routing protocol. When a node will generate a route request(RREQ) message when it has to transmit traffic to a host node to which it has no route. RREQ will then be flooded in a limited way to other nodes. This causes a dynamic control traffic overhead and which results in an initial delay when initiating any communication. A route is added when the RREQ message reaches either the destination itself, or an intermediate node with a valid route entry for the destination. For as long as a route exists between two endpoints, AODV remains passive. When the route becomes invalid or lost, AODV will again issue a request. Other messages are RREP which is route reply message from the destination to the originator and RRER which is send after detection of a broken link, to notify the same.

3.4.3 Service Discovery

The mobile nodes require strategies to discover each other's presence on the network and establish functional network services. Due to the absence of any central unit, DRAP uses the directory-less approach, where service providers do not distribute their service descriptions onto other nodes in the network, but leave them stored on their own device.

In our approach, the management framework for mobile network is based on the Simple Service Discovery Protocol (SSDP)[50] a component of Universal Plug and Play (UPnP) by Microsoft Corp. UPnP is architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It works with wired or wireless networks and can be supported on any operating system which suits the heterogeneous nature of our environment. UPnP boasts device-driver independence and zero-configuration networking.

A device can dynamically join a network, obtain an IP address, convey its capabilities upon request, and learn about the presence and capabilities of other devices. A device can also leave a network smoothly and automatically without leaving any unwanted state behind. Universal Plug and Play leverages TCP/IP and the Web technologies, to enable seamless proximity networking in addition to control and data transfer among networked devices in the home and office.

SSDP protocol is used for announcing a device is presence to others as well as discovering other devices or services. HTTP over multicast and unicast UDP which are referred to as HTTPMU and HTTPU, respectively. A joining device sends out advertisement (ssdp:alive) multicast message to advertise its services to control points. They are the potential clients of services embedded into the device. UPnP has no central service registry. The other message of SSDP is search (ssdp:discover) multicast message sent when a new control point is added to the network. Any device that hears this multicast should respond to it with a unicast response message. XML is used to describe device features and capabilities. The aforementioned advertisement message contains a URL that points to an XML file in the network, describing the UPnP device is capability. Hence other devices, by retrieving this XML file, can inspect the features of this device and decide whether it fits their purposes. This XML description allows complex, powerful description of device capability.

3.5 Potential Applications for DRAP

Literature has many scenarios those can be benefited by Mobile Cloud Computing. We use the same scenarios and explain how DRAP could be a good alternative to central servers.

Sharing GPS/Internet data: It is more efficient to share data among a group of mobile devices that are near each other, through local-area or peer-to-peer networks. It is not only cheaper, but also faster. Rodriguez et al.[65] present a case study of a hiking party at Padjelanta National Park, which is a deserted land in the Arctic Circle lacking power access points and network coverage making it essential to save power and retrieve locations. The paper reports up to 11% energy savings by sharing GPS readings. Due the the infrastructure-less nature of DRAP, it can be efficiently implemented in such a case. Organised knowledge sharing could be achieved by managing ad-hoc nodes and saving power consumption. However co-location of most participants was low in the case study, the energy savings should be much higher (up to 40%) in a more social perspective like pubs, restaurants, and stadiums.

Sensor data applications: Since most mobile phones are equipped with sensors today, readings from sensors such as GPS, accelerometer, light sensor, microphone, thermometer, clock, and compass can be timestamped and linked with other phone readings. Queries can then be executed on such data to gather valuable information related to the surroundings. Such queries could be "What is the average temperature of nodes within a mile of my location?" or "what is the distribution of velocities of all nodes within half a mile of the next highway on my current route?" Sample applications for this are traffic reporting, sensor maps, and network availability monitoring [43]. To avoid data "hot-spots", i.e. smartphones uniquely hosting very popular data, could be avoided by following a de-centralized approach towards the application and using DRAP system where multiple smartphones can act as local servers.

Image and Natural Language processing: Optical character recognition (OCR) systems can be executed on multiple hosts connected in DRAP. In a real life scenario, this would be useful in a case of a foreign traveller who takes an image of a street sign, performs OCR to extract the words, and translates words into a known language. A similar scenario is given in [34] where a foreign tourist Peter is visiting a museum in South Korea. He gets interested in an exhibit, but cannot understand the description since it is in Korean. He takes a picture of the text, and starts an OCR application on his phone. Unfortunately his phone lacks the resources to process the whole text. Although he could connect to a remote server via the Internet, that would mean he use roaming data which is too expensive. Instead, his device scans for nearby users/devices who are also interested in reading the description, and requests sharing their mobile resources for the task collaboratively. Those who are interested in this common processing task create an ad hoc network with Peter and together, their local resourced cloudlet is able to extract the text, and then translate it to English and other languages as needed. This can be applied to many situations in which a group is involved in an activity together. Text to speech conversion could be done to help the visually impaired.

Crowd Sourcing: Mobile devices carry content like videos, photos, and music for single user but this data could be of interest for some other users, or multiple devices could share their bit to construct improved information. For example, Shazam[5] is a music identification service for mobile phones, that searches for similar songs in a central database. In the context of the mobile cloud, the searching could be executed on the contents of nearby phones using DRAP. Also video recordings from multiple mobile devices can be spliced to construct a single video that covers the entire event from different angles, and perspectives.

Lost child problem [58]: The 'Lost child' scenario takes place at a parade in Manhattan. John, a five year old child who is attending the parade with his parents goes missing among all the people, and his parents only notice he is missing after some time. Fortunately, a police officer sends out an alert message to all mobile phones via DRAP within a two mile radius, requesting them to upload all photographs they have taken in the parade during the past hour, to a local server that only the police has access to. With John's parents, the police officer searches through these photographs via an application on his phone. After looking through some pictures, they are able to spot John in one of the images, which they identify to be taken at a nearby location. Soon, the relieved parents are reunited with their child.

Chapter 4

DRAP Algorithms

In chapter we describe our algorithms to reach our aim of hosting a cloudlet server on a mobile ad-hoc network. First we describe the resources needed and then the creation of a virtual backbone and surrogate discovery. Later the maintenance of the cloudlet is discussed. We also discuss the sequence of activities if a node joins or leaves the network. Last section gives an example to follow the algorithms step by step and understand the working.

4.1 Background

Stability of the host is a vital issue in any cloud related environment. It become even more critical in mobile ad-hoc networks, given the irregular movement of nodes, addition, deletion and failure of nodes and related events. Other major issues are scalability, sufficient coverage and minimizing network delays and low energy cost .

4.1.1 Resources

For a cloud computing server the minimal expectation of resources are storage, memory, energy. In this work, we have given importance following characteristics of a mobile device.

- Available battery : Energy consumption is the most important constraint of mobile devices, and the batteries cannot replenish until plugged in. It becomes a significant characteristic to be considered.
- Available memory : Usually mobile cannot run memory intensive applications, so when forming a cloudlet we should cater to memory needs of the nodes.
- Available storage : Since in any cloud related application data storage indispensable, storage is the basic need.
- CPU (type, speed) : Response time and latency are always vital issues in designing any network design.
- Load : To understand the responsiveness of our resource nodes we consider the internal load on the device. This can be extended in future research to manage external load on the resourceful nodes to have a balanced distribution of client requests.
- Location : Location of the node is important in two aspects. Firstly, if the resource node is near to the topological center, the cost communication will be less. Secondly, if the resource node starts to move out of the network, the time it takes to reach the fringe is more, thus more time to transfer the data it holds.

Section 4.3.2 describes how these are resources utilised in decision making.

Table 4.1: Node Resources : An example

Node	RAM (in Mb)	Storage (in Gb)	CPU Speed (in GHz)	Battery Level (in %)
A	256	1	1.4	10
B	1016	2	0.2	73
C	256	4	1.2	68

4.1.2 Device Score

In a heterogeneous network, every node is not suitable to offer services to the cloud. We define a scoring algorithm to determine whether or not a particular node resource rich. Before scoring is done, it is important to understand that selecting a node which is a good source of resource X but has negligible amount of resource Y, is a bad decision. For example, in Table 4.1 nodes A, B, C, node A being the fastest processor should not an obvious choice for an eligible cloud node because it can fail any time due to battery drain. Similarly node B has better RAM and longer battery life but is slow CPU.

To avoid any such selection, the threshold value of each resource should be defined. This condition has an advantage to the participating nodes as they would not fail due to resource bottleneck and but at the same time could handicap the network as many nodes may not pass all threshold values and thus not qualify as resourceful nodes. In Section 5.1.3, we discuss the threshold values used in this work, and its calculation method.

4.2 Space Representation

Suppose at any time t, the given graph $G = (V, E)$ is connected i.e for any node $u, v \in V$. Therefore, set V is the universal set of mobile nodes; set D is the stable nodes in CDS; set C containing all nodes in V that are not included in D; set R is the resourceful nodes; set S is the set of nodes offering cloud services.

At $t = 0$; $C = V - \varnothing$

$D = \varnothing$

$R = \varnothing$

$S = \varnothing$

All nodes in G maintain an open neighbour set $N(v) = u|v, u \in E$, i.e. $N(v)$ consists of the set vertices adjacent to v.

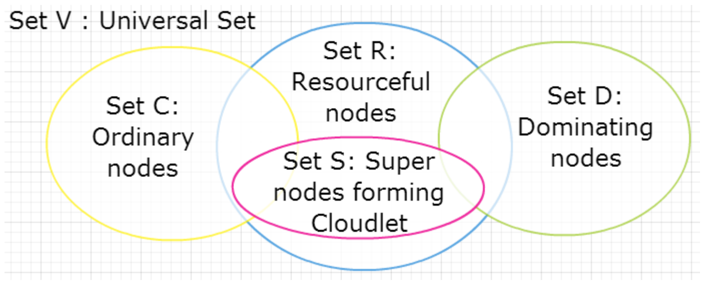


Figure 4.1: Space Representation

4.3 Algorithms

In this section we provide the description and pseudo codes of our algorithms. We have four algorithms in total. First algorithm is used to create a virtual backbone in the network, second contains the rules to score a device. After scoring we decide which nodes would be a part of cloudlet and who will serve as buddy nodes. Finally we describe our routing.

4.3.1 Creation of a CDS

We create a virtual backbone in the network for effective dominating set based routing. We implement the Connected Dominating Set based on a distributed and simple approach proposed by Wu and Li [69]. This algorithm is based on a marking process that marks every node in graph G with either $T(marked)$ or $F(unmarked)$. Any node which as atleast two unconnected neighbours is marked T according to the Algorithm 1. All nodes marked T form the connected and close to minimum dominating set D.

Every node entering u the network must multicast a beacon message to establish its presence and every receiving node v must process it and add to its neighbour set N(v) = N(v) + u. Node v sends its neighbour set to the new node. On receiving $N(v)$, any node that has at least two unconnected neighbours become a CDS node.

Ideally, this algorithm requires only local information and constant number of iterative rounds of message exchanges among neighbouring hosts. The dominating set includes all intermediate nodes of any shortest path, and an all-pair shortest paths algorithm only needs to be applied to the sub-network containing the dominating set.

This algorithm by the nature supports scalability. If a node joins as a leaf node, it will have get a CDS node in one hop distance, or otherwise it will become a CDS node itself if it has two unconnected neighbours. This also supports the merging of cloudlets via bridge nodes, as the common nodes will be serving as CDS nodes to both the networks, and thus form a communication channel resulting in a larger network. This would enable resources from both subgraphs be available to client nodes of the two smaller networks.

4.3.2 Election of Cloudlet Nodes

The first step in creation of a cloudlet is determining a set R; set of nodes those could be eligible to form the cloudlet. For this process, we use the Scoring Algorithm that helps the mobile node to analyse its resources. The significance of scoring of devices

Algorithm 1 Creation of a CDS

```
$States$ S = {ENTERING, ACTIVE, DONE}
$State_{initial}$ = {ENTERING}
$State_{termination}$ = {DONE}
v is the node id, N(v) is the neighbour set of v.

ENTERING:
  Spontaneously
  send($Beacon$) to all nodes in transmission range.
  Marked : $FALSE$
  become ACTIVE

ACTIVE:
  if receive($Beacon$) then
      N(v) = N(v) + sender
      send($N(v)$) to sender
  end if
  if receive($N_{sender}(v)$) then                ▷ Receive neighbours of a neighbour node
  result =unconnectedNeighbours($N(v), N_{sender}(v)$)        ▷ function call
      if result then
          Marked = $TRUE$
          send($v$) to $N(v)$
          become DONE
      end if
  end if

  function unconnectedNeighbours($N(v), N_{sender}(v)$)
      connected = False
      for all i : nodes in $N(v)$ do
          for all j : nodes in $N_{sender}(v)$ do
              if $i = j$ then
                  connected = True
              end if
          end for
          if $connected \neq True$ then
              return True
          end if
      end for
      return False
  end function
```

is discussed in Section 4.1.2.

The threshold values will directly reflect the resource pool created. The threshold values should be determined with respect to the resource specific needs of the application to be hosted in the cloudlet.

Every participating node calculates its 'Score' according to the Algorithm 2. If a node gets a Score it sends the same to the neighbouring CDS nodes in its one-hop distance, according to Algorithm 3. The nodes in set D (Dominating Set) collects all scores from its neighbour nodes and selects the node with maximum score as its 'Elected CloudCandidate' is added to set S. This information is multi-casted within the CDS, so that if any client node request a service the nearest CDS can decide and the forward it to the nearest CloudCandidate that can serve the request.

4.3.3 Buddy System

Once the node is elected as a the cloudlet node $v \in S$, it searches for a supportive buddies set B_v in the network. The Resourceful neighbour nodes of any cloudlet node qualifies to be the buddies to that node. In other words, any node which gets a score and enters the Resource Set R is nominated as the buddy (support) to its nearest cloudlet node. If there is no eligible node in the scoring result set, the dominator $d\ in\ D$ requests its neighbours to send a possible buddies, and the dominator d then selects the nodes nearest to the cloudlet node v in consideration.

Finally, the ordinary nodes maintain only relevant data from one hop neighbours and dominator set contains the routing information of all cloud nodes.

Algorithm 2 Scoring Algorithm

```
$States$ S = {ENTERING, RICH, POOR}
$State_{initial}$ = {ENTERING}
$State_{termination}$ = {RICH, POOR}
v is the node id, N(v) is the neighbour set of v.

ENTERING:
  Spontaneously
  result =ScoreNode(void)          ▷ Every node on joining the network calls this
  function
  if result then
      send($result, v$) to neighbouring CDS nodes ($N(v) \cap D$)
      become RICH
  else
      become POOR
  end if

  function ScoreNode(void)
      Score = 0
      if ($BatteryCharging == YES$) then
          Score = Score + 100
      else if ($BatteryLevel > 60$) then
          Score = Score + 60
      else
          return 0                             ▷ insufficient battery level
      end if
      if ($FreeStorage > Storage_T$) then
          $Score = Score + FreeStorage/FreeStorage_{max} * 100$
      else
          return 0                                  ▷ insufficient storage
      end if
      if ($RAM > RAM_T$) then
          $Score = Score + RAM/RAM_{max} * 100$
      else
          return 0                                  ▷ insufficient memory
      end if
      if ($NoOfCPU > NoOfCPU_T$) then
          $Score = Score + NoOfCPU/NoOfCPU_{max} * 100$
      else
          return 0                         ▷ insufficient processing units
      end if
      if ($CPUSpeed > CPUSpeed_T$) then
          $Score = Score + CPUSpeed/CPUSpeedmax * 100$
      else
          return 0                                   ▷ slow CPU speed
      end if
      return Score
  end function
```

Algorithm 3 CloudNode Election

```
$States$ S = {LISTENING, ACTIVE, DONE}
$State_{initial}$ = {LISTENING}
$State_{termination}$ = {DONE}
counter = 0
v is the node id, N(v) is the neighbour set of v.

LISTENING:
On being selected as a CDS node
  LocalMaxScore = ScoreNode(void)
  CloudletCandidate = nodeID

  if receive($Score_{sender}, nodeID_{sender}$) then
      if ($Score_{sender} > LocalMaxScore$) then
          $LocalMaxScore = Score_{sender}$
          $CloudletCandidate = nodeID_{sender}$

      end if
      counter = counter +1
      if $counter == count(N(v))$ then
          become ACTIVE
      end if
  end if

ACTIVE:
Spontaneously
  for all $i : nodesinN(v) \cap R - CloudletCandidate$ do
      $B_{CloudletCandidate} = B_{CloudletCandidate} + i$          ▷ buddy nodes
  end for
  send($CloudletCandidate$, 'isCloud') to all nodes in CDS
  send($B_{CloudletCandidate}$, 'your buddies') to CloudletCandidate
  send($CloudletCandidate$, 'you are buddy') to all nodes in $B_{CloudletCandidate}$
  become DONE
```

4.3.4 Inter-cloudlet communication

The implementation of CDS construction is done using some concepts of AODV routing protocol. The three AODV messages: RREQ, RREP and RERR, the nodes update the next hop, sequence number and the hop counts of their routes are employed. With this algorithm, every node will have a CDS node in its one-hop neighbour set or be a CDS node itself. If a RREQ is made from node A to node B, the CDS route with minimum hop count is selected as in Figure 4.2 .

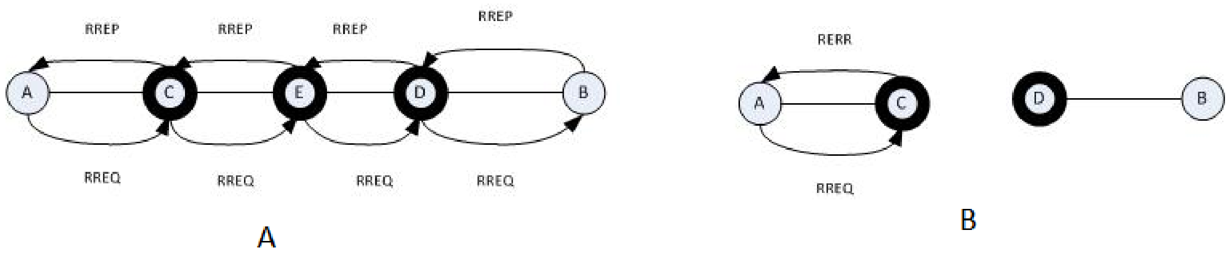


Figure 4.2: CDS based Routing

- If the source A is not a CDS node, it forwards the packets to a neighbouring CDS node C.
- This node C acts as a new source to route the packets in the reduced graph generated from the connected dominating set.
- Eventually, the packets reach a destination gateway, which is either the destination node B itself or a CDS node D of the destination host. In the latter case, the CDS destination forwards the packets directly to the destination node B.
- In the whole process of routing, if any route cannot be reached, the node sends an error message on the back route.

4.3.5 Maintenance

A node movement can be seen as leaving from a sub-topology and entering in another sub-topology. Due to this concept all the changes in the topology can to divided in 3 categories Node Entering, Node Leaving, Node Failure.

Node Entering

When a node p enters the system, it enters as an ordinary node. First it sends beacon message to all nodes in transmission range. The next step is determined according to the result *set* $N(p)$, set of its neighbours.

- If $N(p) \cap D \neq \varnothing$ then p is added to the set of C of ordinary nodes. In other words node p is connected to a CDS node in one-hop distance.
- If $N(p) \cap D = \varnothing$ then using Algorithm 1, node p and its neighbours decide which node n where $n \in N(p)$ should be added to *set* D.

Node Leaving

If a node p leaves the system, the resulting topology may or may not remain a fully connected. If any node, if removed, results in two or more connected sub-graphs, then such a node is called bridge node. Only a dominator node $d \in D$ can be a bridge node. So if the bridge node leaves the network, $V = V \setminus d$ becomes disconnected, resulting in two or more sub networks, they would then have their individual cloudlets.

If a node suddenly disappears from the network, it is considered a node failure, but if the node accelerates in small velocity, then the time it takes to leave the network can be utilized to transfer the information on the leaving node to other stable nodes.

- If $p \in S$, then with the commence of the movement of p, the Cloud Agent will send the latest state to the buddy nodes and will then start transferring its data it holds. The most powerful(highest score) buddy node to $b \in R \cap N(d)$ takes charge of the tasks being performed and would also send update to the dominator node to re-configure the network and search for newer eligible cloud nodes and respective buddy nodes. The CDS nodes should update the cloudlet information throughout the overlay network.

- If $p \in D$, then it sends information to its one-hop neighbours. Each neighbours individually with the help its neighbour set $N(N(p))$ decides whether or not it belongs to the new creation of CDS using Algorithm 1.
- If $p \in S \cap D$, then it performs both actions as expected as a Cloud Agent and CDS agent separately. While the Cloud Agent transfers it state to buddy nodes, a new CDS node will be selected. The CDS node will now elect a CandidateCloud node using Algorithm 3.
- If $p \in B$, the movement of p will be accompanied by its announcement to the cloudlet node and its related other buddy nodes(working for the same Cloud Agent). Related buddy nodes may or may not demand the transfer of data depending on the relevance of the data held by the leaving node.

Node Failure

- If $p \in S$, then the failure of p will keep the dominator set unchanged, except the relations with p need to be updates in the all nodes of *set* D. The buddy nodes to p will make the process fallback to the last log or checkpoint, take charge of the tasks being performed, re-construct the data and send update to the dominator node to re-configure the network and search for newer eligible cloud nodes and respective buddy nodes.

- If $p \in D$, then the failure of p will be followed by the addition of a new node to D from N(p). Using Algorithm 1, the neighbouring nodes will locally make a decision about the new CDS node.

- If $p \in S \cap D$, then the failure of p would be in two steps, first the buddy to p would send notification to the CDS to resolve and find a new dominator to the now orphaned nodes, and next would perform a search for eligible cloud nodes and buddy nodes before presuming the tasks.

- If $p \in B$, then the failure of p will be sensed by other buddies to the same cloudlet node. They will try to re-construct the data and continue with the tasks. If this was the only buddy node to its cloudlet node c, then c should inform the CDS ask for another buddy node, until then c should halt all tasks in a safe state.

Figure 4.3 shows a possible sequence of activities followed by a network when a node joins or leaves a network.

When a node enters a it sends beacon/alive messagesin its transmission range. The network nodes receive the message and adds its to their neighbour set and set it as acknowledgement. All nodes now have the information regarding their one hop neighbours and two hop neighbours(neighbours of its one-hop neighbours). Using this information each node individually decide whether or not it is on the CDS. If on CDS, it allows CDS agent to find device scores of its one hop neighbours and elect a cloudlet node and buddy node.

When a node leaves the network, it triggers an event in the system. This event is caught by the listeners installed on the buddy nodes that pass this information to the nearest CDS node, the acting cloudlet node and its neighbouring buddy nodes to take action as per the status of the leaving node.

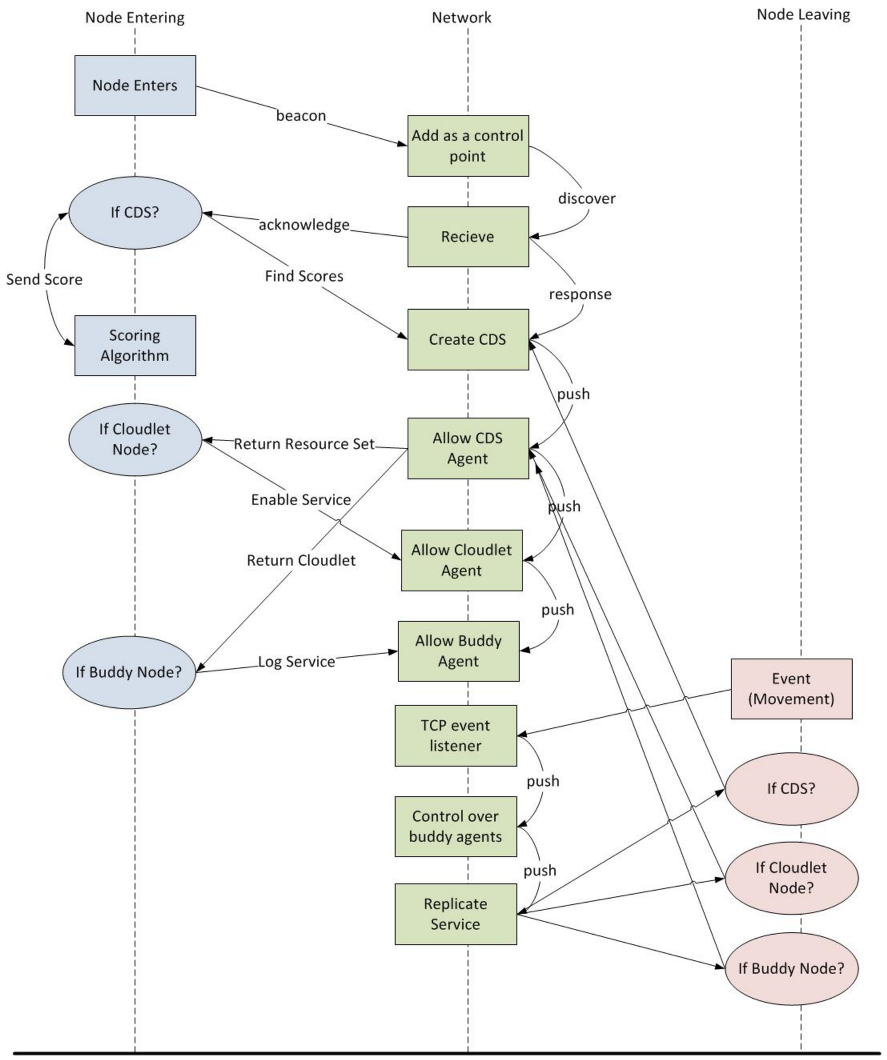
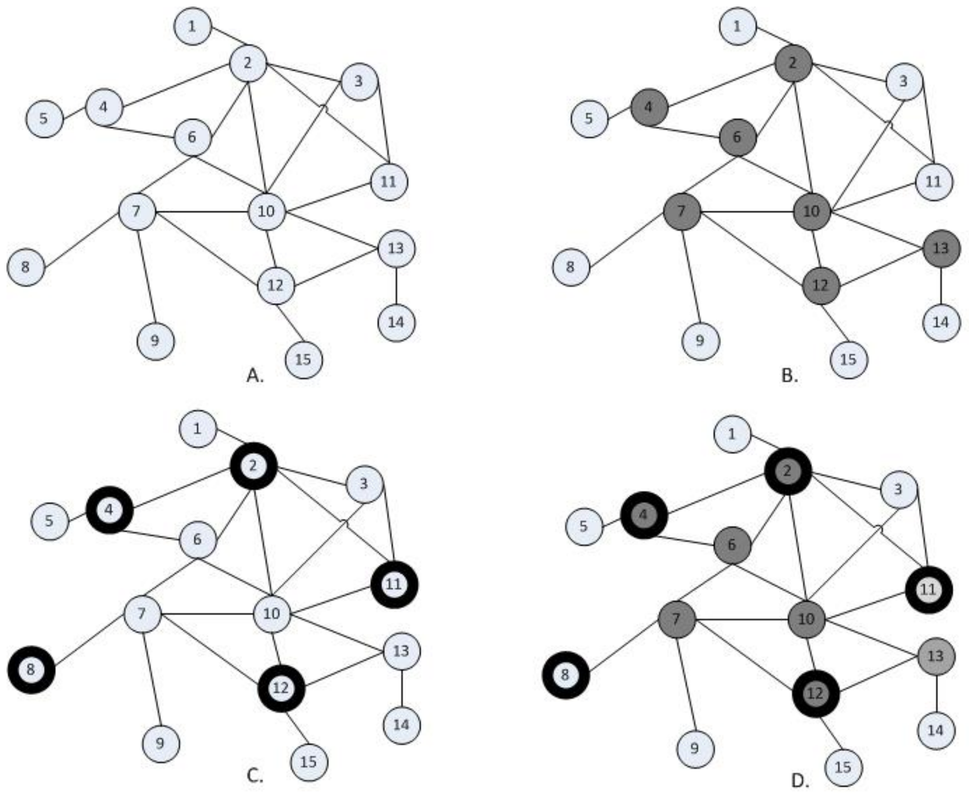


Figure 4.3: Control Sequence Diagram

4.4 An Example

Figure 4.4, is the first snapshot of a sample topology. In part A shows a graph of 15 nodes connected by bi-directional links.



Node	Score
1	311
2	324
3	224
4	434
5	193
6	NA.
7	178
8	398
9	312
10	122
11	411
12	372
13	NA.
14	344
15	NA.

Figure 4.4: Example Topology

Step 1. All nodes exchange their neighbour information with all neighbours. To the given graph, apply the following steps.

Step 2. Algorithm 1, any node with at least two unconnected neighbours becomes a CDS node. All colored nodes in Figure 4.4B are such nodes.

Step 3. By using Algorithm 2 each node determines its resource score, hence knows if it has enough resources to participate in the cloud formation or not. Figure 4.4 has a table for the scores of all nodes. 'NA.' means that the node failed to pass the threshold value for one or more resources.

Step 4. Every CDS node now collects the scores of all its neighbouring nodes and then elects its candidate cloudlet according to Algorithm 3.

Table 4.2: Node Score : An example

CDS Node	Neighbours	LocalMaxScore	CloudCandidate Node
2	1, 3, 4, 6, 10, 11	434	4
4	2, 5, 6	324	2
6	2, 4, 7, 10	434	4
7	6, 8, 9, 10, 12	398	8
10	2, 3, 6, 7, 11, 12, 13	411	11
12	7, 10, 13, 15	372	12
13	10, 12, 14	372	12

Table 4.2 shows the information on the CDS nodes. Every CDS node knows its one hop neighbours and their scores, it then makes a decision using the max score.

Step 5. The elected cloudlets in the example are marked as dark circles as in Figure 4.4C. Table 4.2 has the list of CDS nodes, their one-hop neighbours, the local max score for each CDS node (including their own score) and the Cloud-Candidate it selects.

Step 6. All CDS nodes have a common knowledge of the resource pool created in the network. Whenever a node wants to use the cloud services, it sends request to its nearest CDS node, which then determines if the request can be completed locally within its own network, or has to be outsourced. If the network has capabilities to serve the request, the CDS node determines which cloudlet nodes would be given the responsibility of the same, and will routes the request packets accordingly. Figure 4.4D shows the diagram of the CDS nodes and cloudlet nodes of this network.

Chapter 5

Experimentation

We present our simulation details in this chapter. It contains our hypothesis on network behaviour in different conditions and our evaluation results. We aim to study the network on the stability of the cloudlet in terms of its lifetime and node participation. We also study the energy consumption in hosting our model.

5.1 Scenario

To test the usefulness of CDS creation and understand the movement of nodes within the network, we simulate the algorithm for different types of mobile ad-hoc networks. The aim here is to visualise the working of the system and study its various aspects.

First we describe the environment used, and later discuss the experiments and observations. The results have been complied for total 144 experiments for different scenarios created by varying network parameters. It should noted that the network has an initial set-up, when nodes are fed into the network and edges are discovered. This time could vary depending on the number of nodes and speed.

Table 5.1: Simulation Parameters

Parameter	**Values**
Protocol	AODV
Simulation Area (m^2)	2000×2000, 6000×6000, 10000×10000,
Simulation Time	100 seconds
Number of Nodes	25, 80, 120, 200
MAC Protocol	MAC/802.11
Antenna	Omni Directional
Initial Energy per node	1.1 J
Energy Model	Wifi Radio Energy Model
Maximum Speed in (m/sec)	2,8, 15
Mobility Model	RWPM
Network Simulator	NS 3.13

5.1.1 Topology

The various topologies are created by implementing the Random WayPoint Mobility Model by Johnson and Maltz [37] for node movement. The movement of nodes is as follows: Each node begins by pausing for a fixed time. The node then selects a random destination in the simulation area and a random speed between 0 and some maximum speed. The node moves to this destination and again pauses for a fixed period before selecting another random location and speed. This behaviour is repeated for the length of the simulation.

Simulations are performed in 36 different node settings. The variations are executed on the area of the network, number of participating nodes and the maximum node speed. Simulation parameters are found in Table 5.1

5.1.2 Simulator Used

NS3 simulator [3] is used on Eclipse-CDT. ns-3 is a discrete-event network simulator, it is free software, licensed under the GNU GPLv2 license, and is publicly available for

Table 5.2: Threshold value for resources to be considered

Type of Resource	Value
Current Acceleration	static
Battery	60%
Free Storage	$1Gb$
RAM	$256Mb$
Number of CPUs	1
CPU speed	$1608MHz$

research, development, and use. We use the Wi-Fi physical channel standard 802.11b. The ad-hoc wifi mac is set up using the standard constant speed propagation model and Friss propagation loss model.

5.1.3 Benchmark Data

Passmark [4] is an online directory for hand-held devices on information related to CPU, memory, GPU, and disk benchmarks. It also contains the market share of all devices. As of January 2014, information of 2644 devices is present which includes smartphones and tablets of two most popular mobile Operating Systems namely Android and iOS.

The data for minimum threshold values required by a node is calculated for each resource by selecting the rounded value of the mode from the list of values available in the directory. Table 5.2 contains the values obtained and used in the simulations. This gives a fair estimate of the devices that could be present in proximity at a given time.

Every node in the network represents a actual hand-held device. Each device contains all resource, the value for each resource is randomly determined. A node is added to resource set only if it passes the Scoring Algorithm 2 as described in chapter 4, where the threshold values are from the table above. Figure 5.1 shows the nearly

60% of the nodes are resource rich as per minimum threshold criteria. Figure 5.2 shows a pie diagram of the what resource lacked in the other 40%.

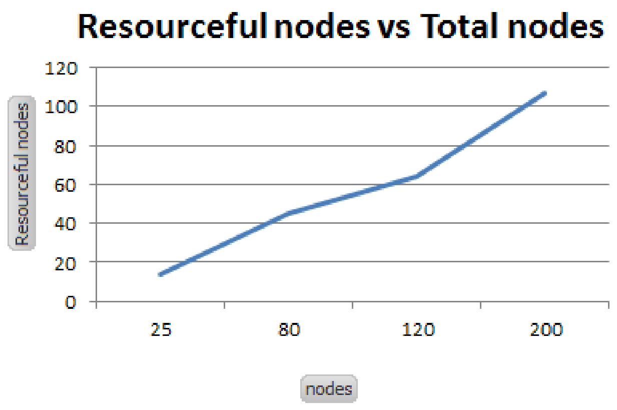


Figure 5.1: The graph shows that nearly 60% of nodes pass the threshold limits for all resources

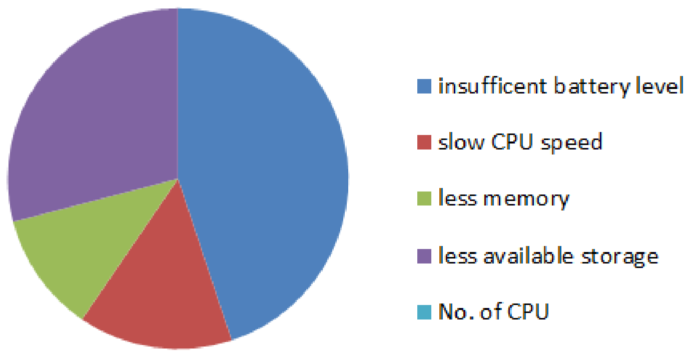
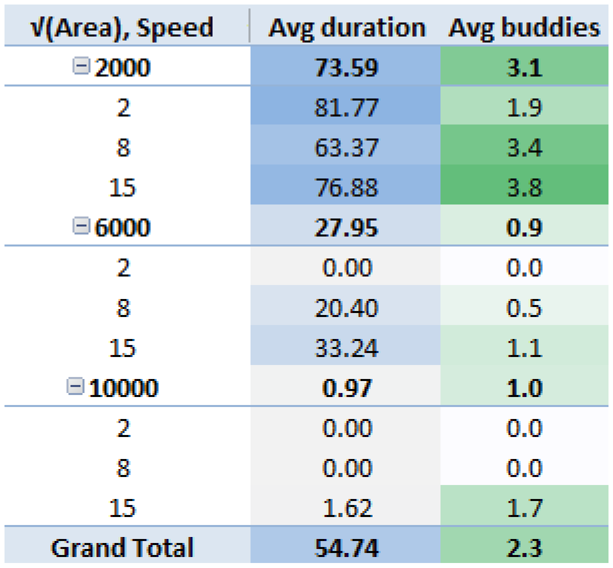


Figure 5.2: Reasons for resource lack in nodes

5.2 Stability Analysis

As mentioned earlier, tests are performed in 36 variations of the network. Network area is considered from $4 \times 10^6 \text{m}^2$ to $1 \times 10^8 \text{m}^2$. Maximum node speed is changed from 2 m/s to 15 m/s. Participating nodes range from 25 to 200 nodes. We have sorted the discussion in 4 groups based on the number of nodes in the network.

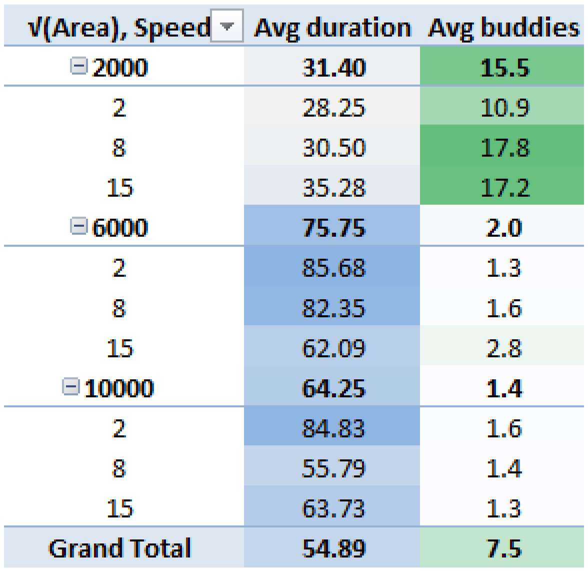


√(Area), Speed	Avg duration	Avg buddies
⊟ 2000	**73.59**	**3.1**
2	81.77	1.9
8	63.37	3.4
15	76.88	3.8
⊟ 6000	**27.95**	**0.9**
2	0.00	0.0
8	20.40	0.5
15	33.24	1.1
⊟ 10000	**0.97**	**1.0**
2	0.00	0.0
8	0.00	0.0
15	1.62	1.7
Grand Total	**54.74**	**2.3**

Figure 5.3: Trend followed by 25 nodes network

First group has **25 nodes**. Two nodes can communicate only if they fall in each others transmission range, so in case of very large network area, we do not expect many nodes to be able to transmit information to other nodes. Hence, cloud activity would be minimal or zero in a very large network. In a medium size network area, we expect some communication links which may increase or decrease depending on the movement of nodes. Experimental results show that the network shows no sign of cloud formation in case of very low node speed. Small area networks are ideal for less number of participating nodes. Communication links lives longer for low speeds but are in acceptable ranges even for a more dynamic network. Figure 5.3 shows the average time a node was serving as cloudlet node and the average number of buddy nodes over the period of simulation time 100 seconds over 25 nodes.

The second group has **80 nodes**. A fairly equal distribution of nodes can be expected in 80 nodes. A key feature to be noted in this group is the proximity of nodes. In a small size network, the nodes are expected to be close to each other, hence the average number of buddy nodes to the nodes in the cloudlet would be high.

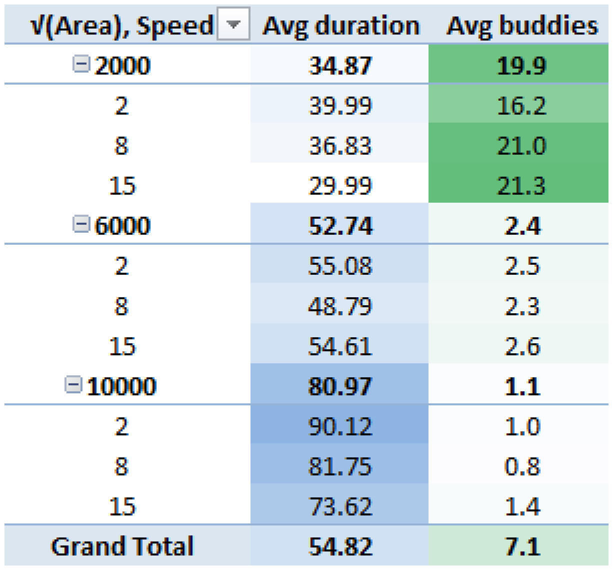


√(Area), Speed	Avg duration	Avg buddies
⊟2000	**31.40**	**15.5**
2	28.25	10.9
8	30.50	17.8
15	35.28	17.2
⊟6000	**75.75**	**2.0**
2	85.68	1.3
8	82.35	1.6
15	62.09	2.8
⊟10000	**64.25**	**1.4**
2	84.83	1.6
8	55.79	1.4
15	63.73	1.3
Grand Total	**54.89**	**7.5**

Figure 5.4: Trend followed by 80 nodes network

The ideal conditions in this group depends on the kind of application to be hosted in the network. The applications that are not very critical in terms on data but need a longer life time can be hosted in medium to large size networks, but the applications that cannot afford to loose any data should be hosted in a small area. Figure 5.4 shows the average time a node was serving as cloudlet node and the average number of buddy nodes over the period of simulation time 100 seconds over 80 nodes.

The third group has **120 nodes**. Since this group will be more crowded then the first two, the cloudlet is expected to be more stable. If a node fails or moves out of the cluster the network will have enough options to replace it. Also, it is important to know that the in CDS construction algorithm proposed by Wu and Li [69] has a characteristic that the size of the CDS increases with the increase in network area for fixed number of nodes and fixed speed. This characteristic is well established in the trends exhibited by the network in this work. Figure 5.5 shows the average time a node was serving as cloudlet node and the average number of buddy nodes over the period of simulation time 100 seconds over 120 nodes.



√(Area), Speed	Avg duration	Avg buddies
⊟ **2000**	**34.87**	**19.9**
2	39.99	16.2
8	36.83	21.0
15	29.99	21.3
⊟ **6000**	**52.74**	**2.4**
2	55.08	2.5
8	48.79	2.3
15	54.61	2.6
⊟ **10000**	**80.97**	**1.1**
2	90.12	1.0
8	81.75	0.8
15	73.62	1.4
Grand Total	**54.82**	**7.1**

Figure 5.5: Trend followed by 120 nodes network

The fourth group has **200 nodes**. Small network area are expected to have high population density, hence high number of buddies. Large network area will have very high lifetime of cloudlet node because of greater CDS size. Medium size network area is a good trade off between lifetime and support nodes. This case selection on the area of network would highly depend on the application to be hosted and the each category shows very significant properties. Figure 5.7 shows the average time a node was serving as cloudlet node and the average number of buddy nodes over the period of simulation time 100 seconds over 200 nodes.

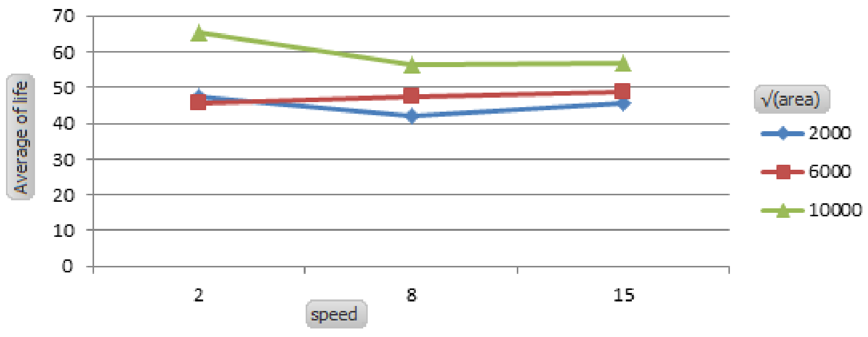
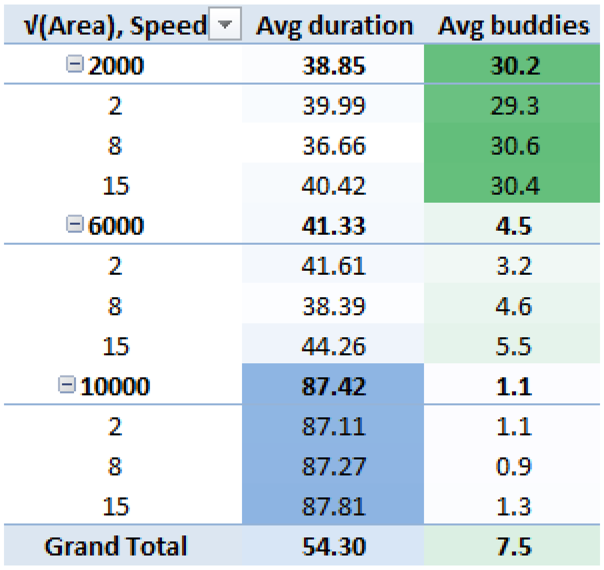


Figure 5.6: Average duration of a node w.r.t speed and area



√(Area), Speed	Avg duration	Avg buddies
⊟ **2000**	**38.85**	**30.2**
2	39.99	29.3
8	36.66	30.6
15	40.42	30.4
⊟ **6000**	**41.33**	**4.5**
2	41.61	3.2
8	38.39	4.6
15	44.26	5.5
⊟ **10000**	**87.42**	**1.1**
2	87.11	1.1
8	87.27	0.9
15	87.81	1.3
Grand Total	**54.30**	**7.5**

Figure 5.7: Trend followed by 200 nodes network

Note

With an exception of very sparse network (25 nodes in $10^8 m^2$), stability of nodes increases with increase in area. Figure 5.6 shows that the average duration of a node in cloud is higher for larger network.

5.3 Cost Analysis

Our major focus was on the amount of resources used in creating a network that is easy to set up and maintain. Prior works either use a central server to store information about potential surrogate devices or use a brute-force method to gather information of all resource rich devices, and then make the best possible choice. Central server becomes a single point of failure, and brute force method create a huge message complexity and may give rise to network congestion and be unacceptably slow. Also a significant amount of energy loss is observed in this process of surrogate discovery.

As discussed, the idea of virtual backbone is a solution to this problem. It is a spanning tree structure created by Connected Dominating nodes. Every Node in the network is connected to atleast one CDS node. The CDS nodes in turn store whether or not its non-CDS neighbours are a possible candidate for surrogate service.

The construction of CDS is an order of O(m) message complexity, where m is the total number of active links in the network. As the longest distance a message travels is one-hop. Every node that enters the network introduces it presence by sending messages to its one-hop neighbour. All nodes that receive this message adds the new node to its neighbour set and sends the new neighbour set to all its one-hop neighbours.

A brute force method of doing this is, starting from the client node, traverse the edges until all nodes are reached. If this brute force method is applied every time a edge is 'discovered', the complexity of the algorithm is O(mn), where n is the number of participating nodes and m are active communication links.

All CDS nodes hold information of the cloudlet nodes in the network. Any request made by a client node is acknowledged by the closest CDS node and it forwards the request to the nearest cloudlet node that has the resources to cater to the request. Since cloudlet nodes are selected by the CDS nodes, they are likely to be present in all parts of the network and cover all area. These two factors play an important role in providing quick response and saving energy.

Figure 5.8 shows the comparison of radio energy consumption for surrogate discovery in traditional method vs. DRAP. 32 out 36 scenarios indicate energy conservation by using our algorithm for surrogate discovery. Maximum conservation is observed in large networks with low mobility with as much as 91% of savings on radio signal energy.

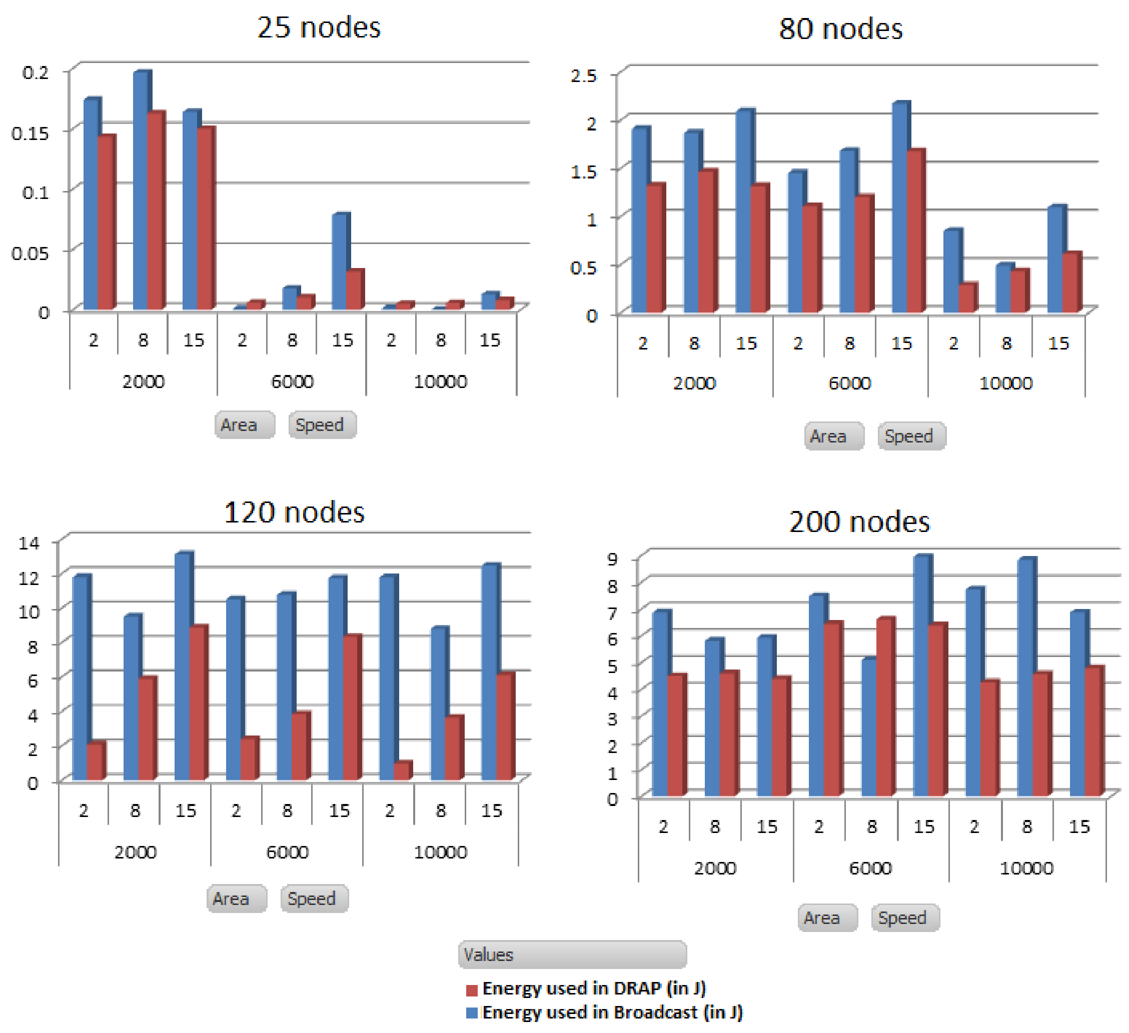


Figure 5.8: Total Radio Signal Energy utilized by CDS vs Broadcast

5.4 Dynamic Behaviour Analysis

Variations in results should be expected in real scenario as homogeneity of the network may vary, participating nodes can more or less resourceful as compared to the uniform distribution simulated in the experiments. The threshold values used in the experiments are calculated using the resources presently available in the industry. These values will change with the introduction of better and newer devices, and technologies.

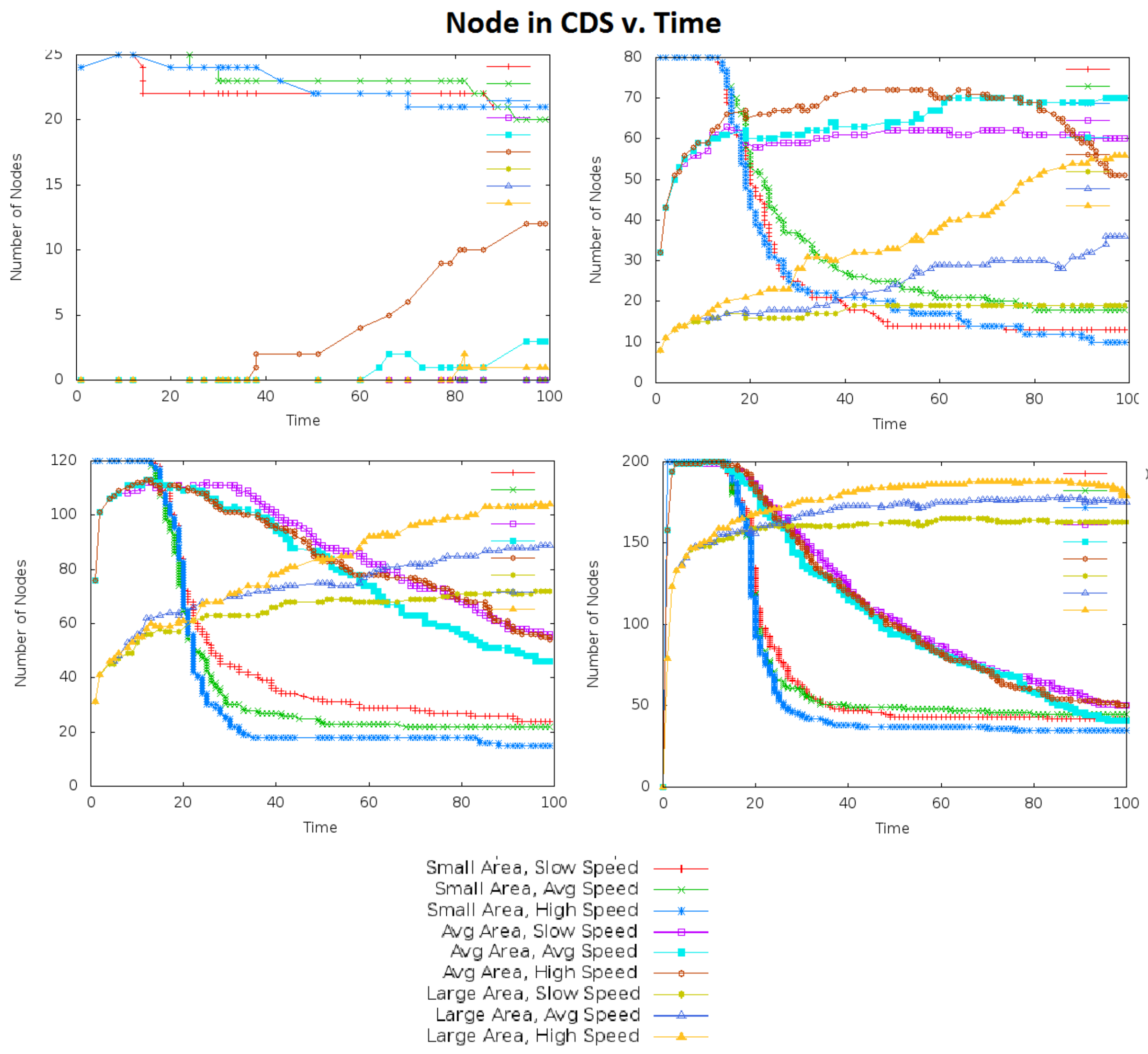


Figure 5.9: Average Nodes in CDS for different test scenarios

Figure 5.9 shows the number of CDS nodes in the network for simulation time 100 seconds. The CDS size i.e number of nodes on the dominating set increases with the increase in network area.

Very large CDS size would be inappropriate as it will have a high maintenance cost in terms of messages and energy as CDS nodes share common knowledge that needs to be update periodically. Ideal CDS size ranges from 30% to 60% of the total participating nodes.

Figure 5.10 shows the number of cloudlet nodes in the network for simulation time 100 seconds. More the number of cloudlet nodes, the chances of local execution of the request increase hence saving the cost of using a distant cloud.

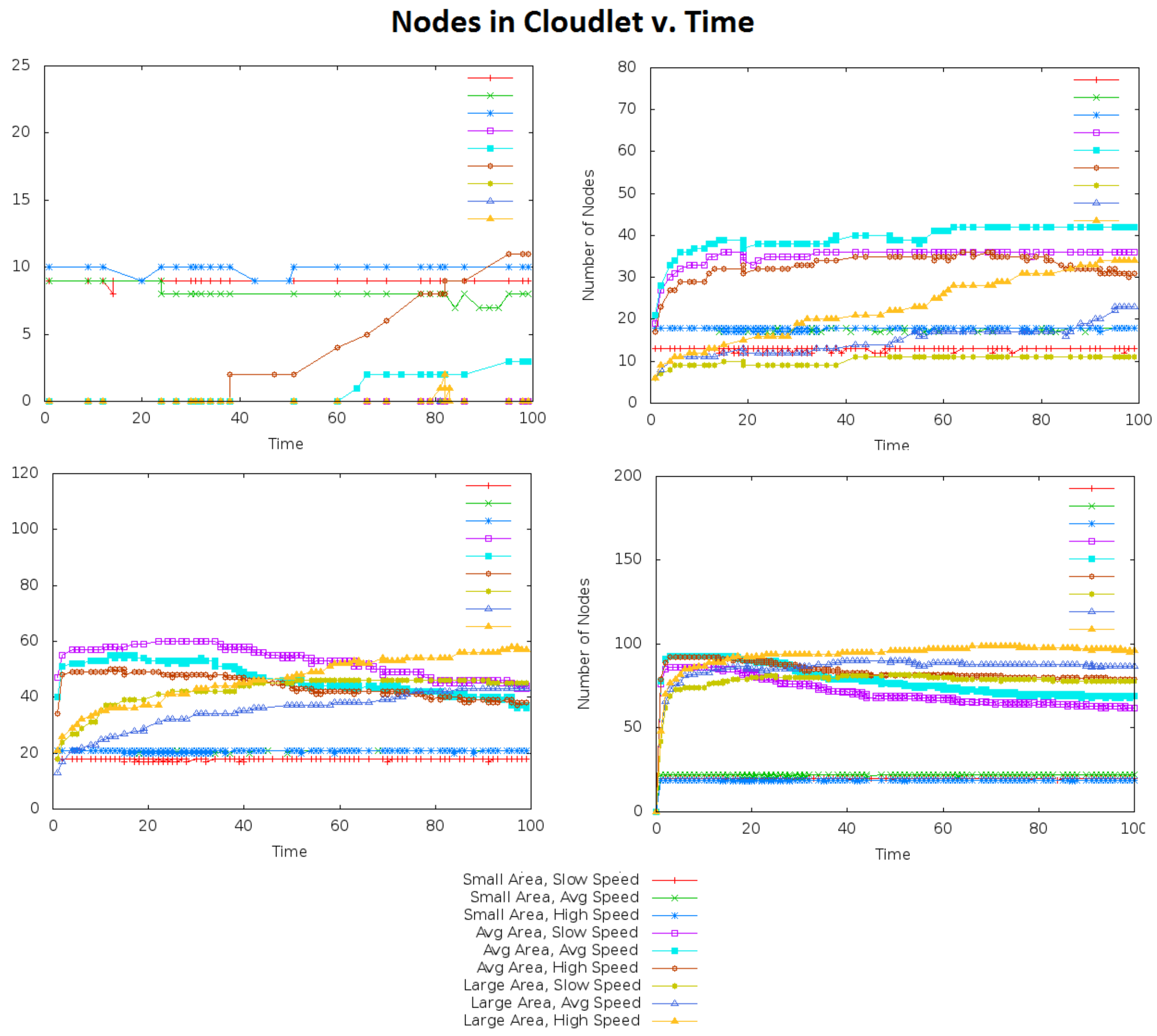


Figure 5.10: Average Nodes in Cloudlet for different test scenarios

Chapter 6

Conclusion and Future work

By applying the concept of volunteer computing to mobile cloudlets we are able to demonstrate the idea of self-sufficient neighbourhood. In real life, this work can be of great interest in place-bound activities, as they occur at a fix location, where the movement of nodes is reduced, connectivity is more stable, leading to fewer disconnections and faults. Also applications where many devices are expected to hold same information or perform similar tasks, this model could save time and resources. A de-centralized model for cloudlet with low cost has been designed and evaluated taking into account the conditions of community networks. We expect our results to be transferable to a prototype of a real community cloud system.

We have described, the requirements of a cloudlet, and an de-centralized architecture for surrogate discovery by creating a virtual backbone in the ad-hoc network. Each device in the network is analysed using an algorithm for resource scoring, so that the devices offering the service do not die out in the process. A four layer design has been discussed to demonstrate the components of proposed cloudlet middleware. Our system features characteristics of stable system with scalability and high performance. The algorithms hence developed provided a relatively low cost infrastructure.

To ensure reliability, we introduced buddy nodes in the network, they log the events on the cloud and take actions in case of service discontinuation. They are given the ability of channel the request to an off-site cloud in case of partial failure of the network. The procedure to maintain the network is discussed if a node enter or leaves the network. We have discussed the need of trust and incentive mechanism to invite more nodes to volunteer resources in the cloudlet.

Our simulation results show significant (32 out of 36 scenarios) cut in radio signal energy usage. We have present our hypothesis on network behaviour in different conditions and our evaluation results. Stability of the cloudlet in terms of its lifetime and node participation is studied and analysed.

Future Work

We anticipate the development of a prototype of our model and test it in real environment with heterogeneous settings to resolve different scenarios of mobile cloud computing. For example, we can host a cloudlet to solve a natural language processing task. The text can be divided into logical parts and each part could be assigned to a cloudlet node. The results from individual nodes can be combined to create the final result.

Predominantly this work can be extended to deliver a more stable cloudlet and higher network longevity by distribution of duties with respect to the load on individual device and network level. Division of tasks should be efficiently occur with in multiple surrogates, hence enabling more complex applications in cloudlets.

A better incentive mechanism should be developed for Mobile Cloud Computing billing to encourage mobile users to volunteer resources, keeping in mind that the hand held devices have a notion of resource poverty.

History-based model should be developed for de-centralized system where the we can study the patterns in which each node appears and disappears in a real scenario. This information can introduced to the dynamic networks and could be used in decision making of task assignment.

Bibliography

[1] "2013 Forecast On Cloud Computing Trends" Available at http://blogs.sap.com/innovation/cloud-computing/2013-forecast-on-cloud-trends-024331 by Folia Grace, Published on January 8, 2013, Last Accessed March 2014.

[2] [Online] http://www.mobilecloudcomputingforum.com/ Last Accessed January 2014.

[3] "ns-3 Documentation" Available at http://www.nsnam.org/doxygen/index.html Last Accessed March 2014.

[4] [Online] http://www.passmark.com/index.html Last Accessed January 2014.

[5] [Online] http://www.shazam.com/ Last Accessed January 2014.

[6] [Online] http://www.wikipedia.org/ Last Accessed January 2014.

[7] "The NSA Files" http://www.theguardian.com/world/the-nsa-files Last Accessed March 2014.

[8] [Online] http://cloudinary.com/ Last Accessed January 2014.

[9] Houston, Drew, and Ferdowsi Arash, "Dropbox." Available at https://www.dropbox.com/about (2008) Last Accessed March 2014.

[10] "Wi-Fi CERTIFIED Wi-Fi Direct: Personal, portable Wi-Fi that goes with you anywhere, any time, 2010". Available at http://www.wi-fi.org/discover-wi-fi/wi-fi-direct. Last Accessed January 2014.

[11] Anderson, David P. "Boinc: A system for public-resource computing and storage." In *Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing, 2004.* , pp. 4-10.

[12] Balan Rajesh Krishna, Nguyen Khoa Xuan, and Jiang Lingxiao, "Real-time trip information service for a large taxi fleet." *In Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services. ACM, 2011*, pp. 99-112.

[13] Balan Rajesh Krishna, Satyanarayanan Mahadev, Park So Young, and Okoshi Tadashi, "Tactics-based remote execution for mobile computing." In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Service. ACM, 2003*, pp. 273-286.

[14] Balan Rajesh Krishna, Flinn Jason, Satyanarayanan Mahadev, Sinnamohideen Shafeeq, and Yang Hen-I, "The case for cyber foraging.", *In Proceedings of the 10th Workshop on ACM SIGOPS European.* 2002, pp. 87-92.

[15] Bulut Muhammed Fatih, Yilmaz Yavuz Selim, and Demirbas Murat, "Crowdsourcing location-based queries." In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 513-518.

[16] Buhrmester Michael, Kwang Tracy and Gosling Samuel D., "Amazon's Mechanical Turk : A New Source of Inexpensive, Yet High-Quality, Data?" in *Perspectives on Psychological Science*, 2011.

[17] Chang Yu-Liang, and Hsu Ching-Chi, "Routing in wireless/mobile ad-hoc networks via dynamic group construction." In *Mobile Networks and Applications 5, no. 1, 2000* pp: 27-37.

[18] Chard Kyle, Caton Simon, Rana Omer, and Bubendorfer Kris, "Social cloud: Cloud computing in social networks." *In Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing)*, pp. 99-106.

[19] Cheng Reynold, Zhang Yu, Bertino Elisa, and Prabhakar Sunil, "Preserving User Location Privacy in Mobile Data Management Infrastructures," In *Proceedings of 6th Workshop Privacy Enhancing Technologies*, Springer, 2006, pp. 393412.

[20] Christensen, Jason H. "Using RESTful web-services and cloud computing to create next generation mobile applications." In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications ACM, 2009.*, pp. 627-634.

[21] Chun Byung-Gon, and Maniatis Petros, "Augmented Smartphone Applications Through Clone Cloud Execution." In *HotOS 2009*, vol. 9, pp. 8-11.

[22] Crow Brian P., Widjaja Indra, Kim L. G., and Sakai Prescott T., "EEE 802.11 wireless local area networks." *Communications Magazine, IEEE 35, no. 9* (1997): 116-126.

[23] Cuervo Eduardo, Balasubramanian Aruna, Cho Dae-ki, Wolman Alec, Saroiu Stefan, Chandra Ranveer, and Bahl Paramvir, "MAUI: making smartphones last longer with code offload." In *ACM Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, 2010.*, pp. 49-62.

[24] Das Tathagata, Mohan Prashanth, Padmanabhan Venkata N., Ramjee Ramachandran , and Sharma Asankhaya, "PRISM: platform for remote sensing using smartphones." In *ACM Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, 2010.*, pp. 63-76.

[25] Dinh Hoang T., Lee Chonho, Niyato Dusit , and Ping Wang Dusit, "A survey of mobile cloud computing: architecture, applications, and approaches." *Wireless Communications and Mobile Computing 2011.* Volume 13, Issue 18, pages 1587 1611,

[26] Elespuru Peter R., Shakya Sagun, and Mishra Shivakant, "MapReduce system over heterogeneous mobile devices." In *Software technologies for embedded and ubiquitous systems*, pp. 168-179. Springer Berlin Heidelberg, 2009.

[27] Fernando Niroshinie, Loke Seng W., and Rahayu Wenny, "Mobile cloud computing: A survey." *Future Generation Computer Systems 29*, no. 1 (2013): 84-106.

[28] Fesehaye Debessay, Gao Yunlong, Nahrstedt Klara, and Wang Guijun, "Impact of Cloudlets on Interactive Mobile Cloud Applications." In *Proceedings of IEEE 16th International Enterprise Distributed Object Computing Conference 2012*, pp. 123-132.

[29] Fieldin, Roy Thomas. "Architectural Styles and the Design of Network-based Software Architectures". *Doctoral dissertation in Information and Computer Science, University of California, Irvine*, 2000.

[30] Flinn Jason, Park SoYoung, and Satyanarayanan Mahadev, "Balancing performance, energy, and quality in pervasive computing." In *Proceedings. 22nd International Conference on Distributed Computing Systems, 2002*, pp. 217-226.

[31] Ganti Raghu K., Ye Fan, and Lei Hui, “Mobile crowdsensing: Current state and future challenges.” *Communications Magazine, IEEE 49*, no. 11 2011: 32-39.

[32] Giurgiu Ioana, Riva Oriana, Juric Dejan , Krivulev Ivan, and Alonso Gustavo, “Calling the cloud: enabling mobile phones as interfaces to cloud applications.” In *Proceedings of Middleware* 2009, pp. 83-102. Springer Berlin Heidelberg, 2009.

[33] Goyal,Sachin and Carter,John, “A lightweight secure cyber foraging infrastructure for resource-constrained devices.” *In Sixth IEEE Workshop on Mobile Computing Systems and Applications. WMCSA 2004.* , pp. 186-195.

[34] Huerta-Canepa Gonzalo, and Lee Dongman, “A virtual cloud computing provider for mobile devices.” In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond ACM, 2010.*, p. 6.

[35] Jeffrey Dean, and Ghemawat Sanjay, “MapReduce: simplified data processing on large clusters.” *Communications of the ACM 51, no. 1 (2008)* pp. 107-113.

[36] Jiguo Yu, Wang Nannan, Wang Guanghui, Yu Dongxiao, “Connected dominating sets in wireless ad hoc and sensor networks A comprehensive survey”, In *Computer Communications, volume 36, Issue 2, 1 January 2013*, pp 121-134.

[37] Johnson, David B., and Maltz David A., “Dynamic source routing in ad hoc wireless networks.” *Kluwer International Series in Engineering and Computer Science (1996)*: 153-179.

[38] Kharif, Perils of the Mobile Cloud, *BusinessWeek Online*, October, 2009.

[39] Koukoumidis, Emmanouil, Peh Li-Shiuan, and Martonosi Margaret Rose, “SignalGuru: leveraging mobile phones for collaborative traffic signal schedule advi-

sory." In *Proceedings of the ACM 9th International Conference on Mobile Systems, Applications, and Services, 2011.*, pp. 127-140.

[40] Kumar Karthik, Liu Jibang, Lu Yung-Hsiang, and Bhargava Bharat, "A Survey of Computation Offloading for Mobile Systems" in *Journal Mobile Networks and Applications* Volume 18 Issue 1, February 2013 Pages 129-140.

[41] Mahmoud Abuelela and Olariu Stephan, "Taking VANET to the clouds". In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM '10).* ACM, New York, NY, USA, 6-13.

[42] Manjunatha Ashwin, Ranabahu Ajith, Sheth Amit and Thirunarayan Krishnaprasad, "Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development," In *Proceedings of 2010 IEEE Second International Conference on Cloud Computing Technology and Science* , pp.496,503.

[43] Marinelli, Eugene E. "Hyrax: cloud computing on mobile devices using MapReduce". *No. CMU-CS-09-164. Carnegie Mellon University, Pittsburgh PA School of Computer Science* , 2009.

[44] Marias, Giannis F., Georgiadis Panagiotis, Flitzanis D. , and Mandalas K., "Cooperation enforcement schemes for MANETs: A survey." *Wireless Communications and Mobile Computing 6, no. 3* (2006): 319-332.

[45] Mordechai (Muki) Haklay,and Weber Patrick, "OpenStreetMap: User-Generated Street Maps," *Pervasive Computing, IEEE* , Oct.-Dec. 2008 vol.7, no.4, pp.12,18.

[46] Murray, Derek G., Yoneki Eiko, Crowcroft Jon , and Hand Steven, "The case for crowd computing." In *Proceedings of the second ACM SIGCOMM Workshop on*

Networking, Systems, and Applications on Mobile Handhelds ACM, 2010., pp. 39-44.

[47] Olariu Stephan, Hristov Tihomir, and Yan Gongjun. "The next paradigm shift: From vehicular networks to vehicular clouds." *Basagni, S. and Conti, M. and Giordano, S. Stojmenovic, I),(Eds), Mobile Ad hoc networking: the cutting edge directions, Wiley and Sons, New York (2012).*

[48] Perkins Charles, Royer E. Belding-, and Das Samir, "RFC 3561-ad hoc on-demand distance vector (AODV) routing." *Internet RFCs (2003): 1-38.*

[49] Prasad Shitala, Peddoju Sateesh K., and Ghosh Debashis, "AgroMobile: A Cloud-Based Framework for Agriculturists on Mobile Platform." in *International Journal of Advanced Science and Technology* Vol.59, (2013), pp.41-52.

[50] Presser A., L. Farrell, D. Kemp, and W. Lupton. "Upnp device architecture 1.1." UPnP Forum, 2008.

[51] Raya Maxim, Papadimitratos Panos, and Hubaux J-P., "Securing vehicular communications." *Wireless Communications, IEEE 13*, no. 5 (2006): 8-15.

[52] Rellermeyer Jan S., Alonso Gustavo, and Roscoe Timothy, "R-OSGi: distributed applications through software modularization." In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, pp. 1-20. Springer-Verlag New York, Inc., 2007.

[53] Rellermeyer Jan S., Riva Oriana, and Alonso Gustavo, "AlfredO: an architecture for flexible interaction with electronic devices." In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pp. 22-41. Springer-Verlag New York, Inc., 2008.

[54] Samimi FA, Mckinley PK, Sadjadi SM. "Mobile service clouds: a self-managing infrastructure for autonomic mobile computing services", In *Proceedings of the 2nd International Workshop on Self-Managed Networks,Systems & Services (SelfMan)*,vol. 3996,2006; 130141.

[55] Sanaei Zohreh, Abolfazli Saeid, Gani Abdullah, and Buyya Rajkumar, "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges." in *Proceedings of IEEE Communications Surveys & Tutorials, vol.PP, no.99,* (2013) pp.1-24

[56] Sarmenta Luis FG, "Volunteer computing." *PhD diss., Massachusetts Institute of Technology, 2001.*

[57] Satyanarayanan, Mahadev, Bahl Paramvir, Caceres Ramn, and Davies Nigel, "The case for vm-based cloudlets in mobile computing." In *Proceedings of IEEE Pervasive Computing, 8*, no. 4 (2009): 14-23.

[58] Satyanarayanan Mahadev. "Mobile computing: the next decade." In *Proceedings of ACM SIGMOBILE Mobile Computing and Communications Review 15*, no. 2 (2011): 2-10.

[59] Shankar Sushant. "Amazon elastic compute cloud." (2009).

[60] Shvachko Konstantin, Kuang Hairong, Radia Sanjay, and Chansler Robert, "The hadoop distributed file system." In *Proceedings of 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1-10.

[61] Soyata Tolga, Ba He, Heinzelman Wendi , Kwon Minseok and Shi Jiye, "Accelerating Mobile-Cloud Computing: A Survey." In *Proceedings of Communi-*

cation Infrastructures for Cloud Computing, ed. Hussein T. Mouftah and Burak Kantarci, 175-197 (2014),

[62] Stone Brad, “Amazon erases Orwell books from Kindle,” *The New York Times*, vol. 18, p. B1, July 2009.

[63] Tingxin Yan , Marzilli Matt, Holmes Ryan, Ganesan Deepak, and Corner Mark, “mCrowd: a platform for mobile crowdsourcing”,In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, November 04-06, 2009, Berkeley, California, Pages 347-348.

[64] Tseng Yu-Chee, Ni Sze-Yao, Chen Yuh-Shyan , and Sheu Jang-Ping, “The broadcast storm problem in a mobile ad hoc network.” In *Proceedings of Wireless networks 8, no. 2-3 (2002)* pp: 153-167.

[65] Vallina-Rodriguez Narseo, and Crowcroft Jon, “ErdOS: achieving energy savings in mobile OS.” In *Proceedings of ACM the sixth international workshop on MobiArch*, 2011., pp. 37-42.

[66] Verbelen Tim, Simoens Pieter, Turck Filip De, and Dhoedt Bart. “Cloudlets: Bringing the cloud to the mobile user.” In *Proceedings of the third ACM workshop on Mobile cloud computing and services* 2012., pp. 29-36.

[67] Weiser Mark, “The future of ubiquitous computing on campus.” In *Communications of ACM 41*, 1 (January 1998), pp 41-42.

[68] White Paper, Mobile Cloud Computing Solution Brief, *http://www.aepona.com/true-mobile-cloud-computing/* November 2010.

[69] Wu Jie, and Li Hailan, “On calculating connected dominating set for efficient routing in ad hoc wireless networks.” In *Proceedings of the 3rd ACM Interna-*

tional Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 1999, pp. 7-14.

[70] Xu Ya, Heidemann John, and Estrin Deborah, "Geography-informed energy conservation for ad hoc routing." In *Proceedings of the 7th ACM Annual International Conference on Mobile Computing and Networking* 2001., pp. 70-84.

[71] Yan Bo, and Chen Guanling, "AppJoy: personalized mobile application discovery." In *Proceedings of the 9th ACM International Conference on Mobile Systems, Applications, and Services* ACM, 2011., pp. 113-126.

[72] Yan Tingxin, Kumar Vikas, and Ganesan Deepak, "Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones." In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services* 2010, pp. 77-90.

[73] Yang Hao, Meng Xiaoqiao, and Lu Songwu, "Self-organized network-layer security in mobile ad hoc networks". In *Proceedings of the 1st ACM workshop on Wireless security (WiSE '02)*. New York, NY, USA, 11-20.

[74] Zhang Xinwen, Schiffman Joshua, Gibbs Simon, Kunjithapatham Anugeetha, and Jeong Sangoh, "Securing elastic applications on mobile devices for cloud computing." In *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009. pp. 127-134.