TOWARDS AN EFFICIENT DISTRIBUTED CLOUD ARCHITECTURE

BY

PRAVEEN KHETHAVATH

Bachelor of Engineering in Electronics and
Communication Engineering
Osmania University
Hyderabad, AP, INDIA
2006

Master of Science in Computer Science
University of Northern Virginia
Annandale, VA
2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2014

TOWARDS AN EFFICIENT DISTRIBUTED CLOUD

ARCHITECTURE

Dissertation Approved:

Johnson P Thomas

Dissertation Adviser

Eric Chan-tin

Dissertation Co-Adviser

Subhash Kak

Mary Gade

LIST THE PUBLICATIONS YOU HAVE FROM THIS WORK

- Praveen Khethavath, Johnson Thomas. “Game Theoretic approach to Resource provisioning in a Distributed Cloud”, submitted at 28th IEEE International Conference on. Advanced Information Networking and Applications Workshops WAINA 2014(Accepted)
- Praveen Khethavath, Johnson Thomas, Eric Chan-Tin, and Hong Liu. "Introducing a Distributed Cloud Architecture with Efficient Resource Discovery and Optimal Resource Allocation". In Proceedings of 3rd *IEEE SERVICES CloudPerf Workshop 2013*
- Praveen Khethavath, Nhat, Prof. Johnson P Thomas. “A Virtual Robot Sensor Network (VRSN)”. In Proceedings of 2nd International Workshop on Networks of Cooperating Objects CONET 2011
- Praveen Khethavath, Johnson Thomas. “Distributed Cloud Architecture: Resource Modelling and Security Concerns”. In Proceedings of 3rd Annual conference on Theoretical and Applied Computer Science (TACS 2012)

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Johnson Thomas for his excellent guidance, patience, and providing me with an excellent atmosphere for doing research and throughout my thesis. His guidance helped me to successfully complete my research. For me, he was not only a respectable professor who led me on the way to do research, but also an attentive tutor who trained me to be a good teacher in my future career. I really appreciate everything he has done in the past years.

Besides my advisor, I would like to thank Dr. Eric Chan-Tin my co-advisor, for his support and guidance in my research. I greatly thank him for all the kindness, support and encouragement. I would also like to thank Dr. Subhash Kak and Dr. Mary Gade for their encouragement and insightful comments. I would like to also thank Dr. Tingting Chen for providing opportunity to work on her research.

I would like to thank all my committee members for being a support and guiding me to advance in my future goals.

I would like to thank my family for supporting me throughout my life and it is they who have made me who I am now.

NAME: PRAVEEN KHETHAVATH

DATE OF DEGREE: MAY, 2014

TITLE OF STUDY: Towards an Efficient Distributed Cloud Architecture

MAJOR FIELD: COMPUTER SCIENCE

Abstract: Cloud computing is an emerging field in computer science. Users are utilizing less of their own existing resources, while increasing usage of cloud resources. There are many advantages of distributed computing over centralized architecture. With increase in number of unused storage and computing resources and advantages of distributed computing resulted in distributed cloud computing. In the distributed cloud environment that we propose, resource providers (RP) compete to provide resources to the users. In the distributed cloud all the cloud computing and storage services are offered by distributed resources. In this architecture resources are used and provided by the users in a peer to peer fashion. We propose using multi-valued distributed hash tables for efficient resource discovery. Leveraging the fact that there are many users providing resources such as CPU and memory, we define these resources under one key to easily locate devices with equivalent resources. We then propose a new auction mechanism, using a reserve bid formulated rationally by each user for the optimal allocation of discovered resources. We have evaluated the performance of resource discovery mechanisms for the distributed cloud and distributed cloud storage and compared the results with existing DHTs, peer to peer clients such as VUZE and explored the feasibility and efficiency of the proposed schemes in terms of resource/service discovery and allocation. We use a simultaneous Auction mechanism and select a set of winners once we receive all contributions or bids. In a real world scenario, users request resources with multiple capabilities, and in order to find such resources we use a contribution mechanism where service providers will provide a contribution price to users for providing a resource. Users use our proposed auction mechanism to select the resources from the set of resource providers. We show that Nash equilibrium can be achieved and how we can avoid the problem of free riders in the distributed cloud. Network latency is an important factor when deciding which resource provider to select. We used treeple a secure latency estimation scheme to obtain network measurements in distributed systems. We developed a mobile application using distributed cloud which preserves privacy and provides security for a user. Distributed cloud is used for developing such an application where all the data needs to be close to the users and avoids single point of failure, which is the problem with existing cloud.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER1

INTRODUCTION

1.1 RESEARCH BACKGROUND

Cloud computing [1, 2, 3] refers to a novel way of computing over the internet where dynamically scaled shared resources are provided as a service over the internet to avoid costs of resource over-provisioning. Many companies are now relying and performing their operations in the cloud. The most popular of all the applications such as social networking, online gaming, email etc. are hosted on cloud. Current cloud architectures provide any thing as services for users. Cloud computing provides infrastructure as a service, software as a service and platform as a service as three main service models. However, with recent developments, the cloud is said to provide XaaS i.e. anything or everything as a service. X can refer to communication, storage, data, network etc. Cloud computing provides many advantages to industry as it saves a lot of time on installing and upgrading applications.

Cloud providers provide various instances of platform, infrastructure and storage as services to users. Cloud computing uses virtualization to provide these required resources to users dynamically in the form of virtual machines. Xen [4] is the most popular open source standard for hardware virtualization used by many cloud providers. Hyper-V, KVM, and Sun xVM are some of the other virtualization management tools commonly used. All these cloud providers provide resources in the form of various instances based on user needs. For example Amazon provides for different sizes of virtual machine instances – Small, Medium, Large and Extra Large.

Cloud computing is often misunderstood based on the technological aspects. Cloud computing is a combination of several concepts including virtualization, resource pooling, resource monitoring, dynamic provisioning, utility computing, multi-tenancy and elasticity.

Cloud computing is built over old technologies and was part of the evolutionary growth of science and technology. The main entities of cloud computing are service providers, physical resources, virtualized resources and end-users.

Distributed computing [12, 13, 14, 15] uses multiple autonomous computers over a network to solve computational problems as one single unit. Allocation of resources in distributed computing to solve a specific task is NP-hard [13]. Cloud computing and distributed computing share a lot of similarities which can be used to build a distributed cloud. However, a distributed cloud computing architecture requires a completely different environment to operate efficiently and securely. Content distribution and file sharing was made possible by peer-to-peer systems. Current peer-to-peer systems use a distributed hash table (DHT) for efficient query lookups. These systems are able to handle churn, node failures, flash crowds, and balance load efficiently.

1.2 MOTIVATION

With more people using cloud services, their machines are underutilized [5]. These unused percentages of resources can be efficiently used for both service provisioning and computation using the distributed cloud. Existing network bandwidth is limited and there exist only a single point of entry to data centers for a cloud user, resulting in high latency and network traffic. Due to its centralized infrastructure, existing cloud poses need of resource over provisioning, high energy consumption, and increases distance to end users.

While the demand for cloud resources continues to increase, machines like Desktops, PC's and small servers which can run virtualization software and can host multiple VM's on them are not part of the cloud. These substantial amounts of underutilized resources which could be integrated into the cloud, has motivated the idea of a distributed cloud which makes use of these resources for both computation and storage purposes.

To overcome the downsides of the existing cloud and to make use of underutilized resources of end users we propose architecture, a distributed solution which can handle all the services provided by the existing cloud. We can construct the distributed cloud by making use of these unused resources thereby avoiding investments on new data centers. The Distributed cloud [6, 7] is different from the existing cloud with geographically-distributed data centers.

1.3 OBJECTIVES

In distributed cloud computing there will be many resource providers, so users will be able to get the resources requested. Locating a proper subset of resources from a widely spread distributed cloud that satisfies users requirements would be significantly difficult. So efficient resource discovery mechanisms and resource allocation methods are needed. Resource discovery would be dependent on user requirements, and applications being used. Hence the resource discovery mechanism must be scalable and take account of node characteristics such as CPU, memory, operating system etc. So we need to define the characteristics of a resource which can be used for efficient resource discovery.

Apart from discovery of resources we must also consider resource allocation and stability of the whole system. We need to allocate resources based on application types such as high performance or high throughput computing and also consider the free riding problem. Since the distributed

cloud is constructed using end users shared machines, some users might use resources without offering their own resources and are referred to as free riders. Therefore we need an efficient strategic mechanism to assign resources. Game theory which is a strategic decision making mechanism would be very helpful for allocating resources to user‘s by maintaining stability of system i.e. allocation is fair.

We see distributed cloud computing as a real working model in next few years. Distributed cloud can also be used in industry to develop future technological applications. New innovative applications can be developed using the distributed cloud.

1.4 CONTRIBUTIONS

This dissertation topic studies distributed cloud architecture models, resource provisioning, networks measurements and applications of distributed cloud. The major contributions are as follows.

- **Distributed cloud model.** The main contribution of this thesis revolves under a new concept, namely, the distributed cloud. The proposed distributed cloud makes use of resources provided by users and allocates these resources to others as needed in a peer to peer fashion.
- **Resource modelling**. We model the distributed cloud resources. Resources and their attributes need to be properly identified.
- **A hierarchical model of distributed cloud architecture.** We developed a hierarchical model of the distributed cloud architecture and identified its problems which led to the development of a completely decentralized cloud model.

- **Implementation of a Distributed cloud model.** We implemented a distributed cloud model which is completely decentralized. We identified different problems for practical implementation of the distributed cloud model. The detailed architecture of the distributed cloud is also described. We also highlight the resource provisioning problem that needs to be solved to build an efficient distributed cloud.
- **Novel resource discovery mechanism.** The problems of using existing multi-dimensional range queries used in p2p systems were identified. We developed a new resource discovery mechanism for finding nodes with multiple attributes.
- **Multiple game theoretic approaches to solve the resource allocation problem in the distributed cloud.** We introduce an auction model by which requestors would ask for resource provisioning. In our model by introducing incentives we encourage providers to compete for providing resources and we consider participation of users for allocation of resources requested by a user. The system will then schedule resources such that the whole system would be stabilized. We described an incentive based auction mechanism which is utility driven. Hence the more resources users will provide, the more they can use the resources in the system thereby avoiding free riders. We also developed a Bertrand model for resource allocation.
- **Network measurements using VUZE.** We have developed a plugin for VUZE a bit torrent client to measure the real time latencies in a distributed network. These network measurements will be helpful to locate nodes nearby.
- **Mobile application.** We also developed an android application for secure and privacy preserving login mechanism using distributed cloud computing. We used a biometric scheme for login mechanism and used pallier encryption algorithm and its holomorphic properties to preserve privacy.

1.5 DISSERTATION OUTLINE

This dissertation is organized as follows:

- Chapter 2 introduces the basics of cloud computing, different architecture models of cloud, underlying infrastructure of cloud and different cloud services. We studied different properties of cloud which were helpful in identifying the characteristics required to build a distributed cloud.
- Chapter 3 presents different models of the distributed cloud architecture. We studied the economic aspects of cloud and advantages of having a distributed cloud which motivated us to the develop distributed cloud. We also discuss about the main challenges involved in developing a distributed cloud.
- Chapter 4 discusses the resource discovery mechanism of the distributed cloud. We use kademlia a P2P which is an existing routing protocol to develop our new routing mechanism. The system simulation model and implementation methods used are presented. We analyze our resource discovery mechanism and compare it with existing distributed systems.
- Chapter 5 introduces different game theoretic methods and the motivation in using game theory to address the resource allocation problem. In a distributed cloud users join and leave the network without any notification and there may be free riders in the system. Game theory is used to analyze the system and allocate resources effectively such that the system is stable and free riders are punished at the same time.
- Chapter 6 discusses the network measurements in distributed systems which would be helpful in analyzing the distributed cloud. We made use of the very popular distributed P2P bit torrent client VUZE to make these measurements.
- Chapter 7 describes an application that is developed using the proposed distributed cloud models. We developed a secure and privacy preserving scheme using the proposed distributed

cloud that can be used as a secure login mechanism for the android application we have developed.

- Chapter 8 concludes the dissertation and provides suggestions for future work.

CHAPTER 2

CLOUD COMPUTING OVERVIEW

2.1 INTRODUCTION

Cloud computing is becoming increasingly important in industry and academic research. Lots of researchers have provided their own definitions of cloud computing based on the applications they have developed. Among the many definitions, the most widely accepted and used is the one proposed by NIST.

NIST defined cloud computing [2] as follows "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Although there are many other definitions for the cloud, the NIST definition gives a more specific and clear definition along with all the essential characteristics, deployment models and service models of cloud. Some of the essential characteristics of cloud computing are:

- **On Demand Service**, where a consumer can get access to services and provision computing capabilities such as network, storage etc. without human interaction.
- **Broad Network access:** All the services are offered over the network and can be accessed using standard mechanisms. All these services can be used using heterogeneous platforms such as tablets, mobile devices, laptops etc.

- **Elasticity:** To a consumer all the services offered and capabilities seem unlimited and can be purchased at any time. Cloud capabilities can be rapidly scaled i.e. can be elastically provisioned and released automatically based on demand.
- **Scalability:** All services are scalable and scalability follows from the elasticity characteristic of cloud computing.
- **Resource pooling:** computing capabilities and resources of the cloud are available to serve multiple users at same time using a multi-tenant model, where different virtual machines are dynamically assigned to each user based on demand.
- **Measured service:** Cloud computing uses Pay-per-use scheme also known as utility computing. All the cloud services can be measured, controlled and provided with transparency between user and provider. Cloud computing uses a metering capability where users are charged based on the measurement of services used. This means the more a user uses the higher the bill. This has led to a lot of research on how to select the resources such that cost is minimized for the user or profit is increased for the provider.

2.2 CLOUD DEPLOYMENT MODELS

All cloud computing models should have certain essential characteristics. There are a wide variety of service models offered by the cloud and four deployment models. Clouds can be deployed in to four different models based on requirements and usage.

- **Public cloud** is the standard cloud computing deployment model in which the service provider makes resources such as applications, storage, platforms etc. available to users over the

internet. All the services will be provided at a fine grained price. Examples of public cloud are Amazon EC2 [48], Windows Azure [49], Sun cloud [50] etc.

- **Private cloud** is a restricted version of the public cloud. These clouds provide resources exclusively to a closed set of people in a single organization. An organization which requires privacy, security and wants control over their data will choose private clouds. Private clouds offer the same features as public clouds. Examples of private cloud include OpenStack [51], CloudStack [52] and VMWare [53].
- **Hybrid cloud** combines features of both public and private clouds along with required third party resources. Hybrid clouds are more suited for IT environments where part of data that should be completely under the control of the organization and the rest can be on public clouds. In the hybrid environment a user might run an application on his own physical machines and at the same time uses cloud services. Examples of hybrid cloud include popular vendors such as HP [55], IBM [54] etc.
- **Community clouds** share the cloud infrastructure by a specific set of organizations that share common missions, security principles and policies. Examples include IBM federal community cloud [56] for government organizations.

2.3 CLOUD SERVICE MODELS

All cloud providers provide services defined by service models such as Infrastructure as a Service (IaaS), Platform as a Service (Paas), and Software as a Service (SaaS). Recent developments have improved in the field of services offered by the cloud. Current state-of-art shows that anything as a service (XaaS) is the most generic way of defining services offered by cloud. Currently all cloud providers rely on huge data centers and are predominantly centralized.

- **Software as a Service** is a service model in which software's is delivered to the user as a service using a web browser over the internet. The user won't have access to the underlying infrastructure such as network, operating systems, storage platform etc. A user will only be able to configure the application. This model eliminates most of the security threats caused by users. Software as a Service is one of the most common and popular cloud based service model. Many companies or software providers use SaaS model to deliver their services to the users. Social media, web mail, google docs, games etc. are some of applications that fall into the Software as a service model based services.

- **Platform as a Service** provides a whole platform as a service to users and offers a highly integrated environment to build applications, test, deploy and host them. Users won't have access to the underlying network, operating system and storage. Users can only have control over their deployed applications and can manage the application environment by modifying configuration settings. PaaS mainly aims at providing the user an environment and all the facilities required to manage and develop an application. Google app engine [57], Heroku [58], Force.com [59] are some of the examples of services offered using the PaaS model.

- **Infrastructure as a Service** provides a whole set of resources which can be available in any real machine. The cloud makes all the processing, network, storage, OS and other computing resources available to the user such that user can run any arbitrary software, deploy any application etc. IaaS gives the user control over the whole infrastructure such as operating system, network, storage etc. other than the underlying cloud infrastructure. Some of the vendors who provide IaaS include Amazon EC2 [48], IBM [56], and Eucalyptus [60].

- **XaaS,** which can be read as anything as service is the result of emerging technology. In the percent scenario, the cloud is said to offer anything as service. Some of the other services that are offered over the cloud include Data, network, Information and policy management as service.

Any infrastructure that can be virtualized or any application that can be run on multiple instances or any service that can be shared can be a service provided by the cloud.

Basic cloud services can be seen as a hierarchical structure as shown below based on the level of access or amount of resources or capabilities available to user. When going from SaaS to IaaS access to underlying resources increase and moving from IaaS to SaaS ease of access increases.

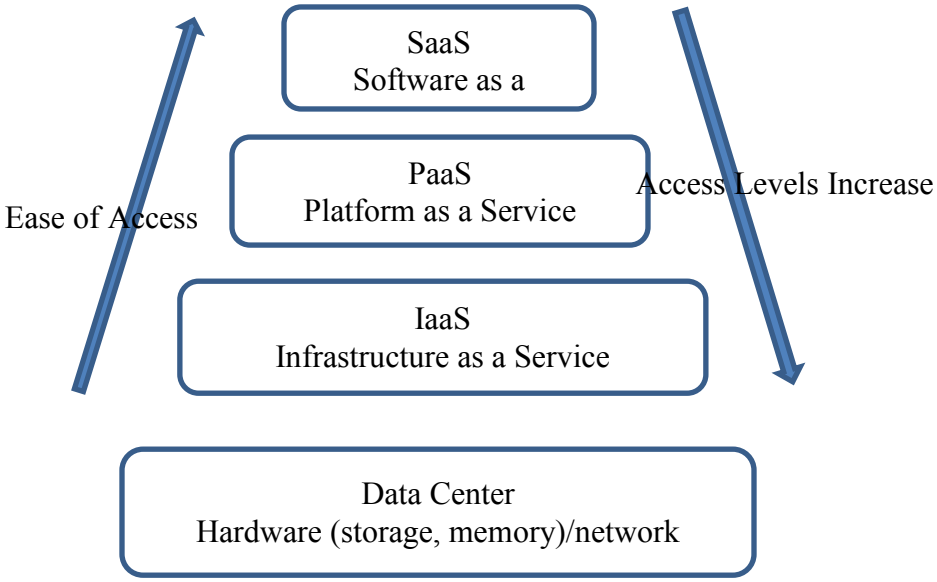

Figure 1 Cloud Service Model

2.4 BASIC CLOUD ARCHITECTURE

Cloud computing is a result of the evolution of previous technologies. Clouds will play a key role in the next decade. Amazon played a key role in development of the cloud by launching Amazon EC2 [48] to rent infrastructures to users. Virtualization was the key concept that made cloud computing possible.

Utility computing is the base for providing resources using grid computing. Grid computing is a parallel system in which a set of distributed resources are shared dynamically by users based on availability, capability and requirements. Cloud computing using virtualization provides resources in parallel to multiple resources which resembles grid computing. The main differences between a cloud and distributed computing are usage of centralized servers to manage the resources by cloud. Cloud computing uses the benefits of both grid and utility computing such as providing resources as requested and pay for services used. Cloud can be considered as next generation grid-utility model.

The basic components of a cloud computing architecture include datacenters, network and terminals or end users. Datacenters are facilities that provide huge computer systems and components needed along such as network, storage etc. The Internet is the main foundation for providing services to users. With improvements in technology, not only PCs but mobile devices, netbooks, tablets, PDA etc. become terminals, i.e. most of the services can be used using these terminal devices.

The cloud architecture is a layered approach as shown in Fig. 2. The bottom layer consists of huge datacenters. Cloud computing uses virtualization to instantiate virtual machines on the fly based on demand. The Hypervisor manages these virtual machines. The physical layer together with virtual machines is also called infrastructure. These virtual machines are provided to the user as IaaS. The Cloud manager manages and keeps tracks of user requirements. A Client request is sent to the cloud manager who then provides respective services to the client.

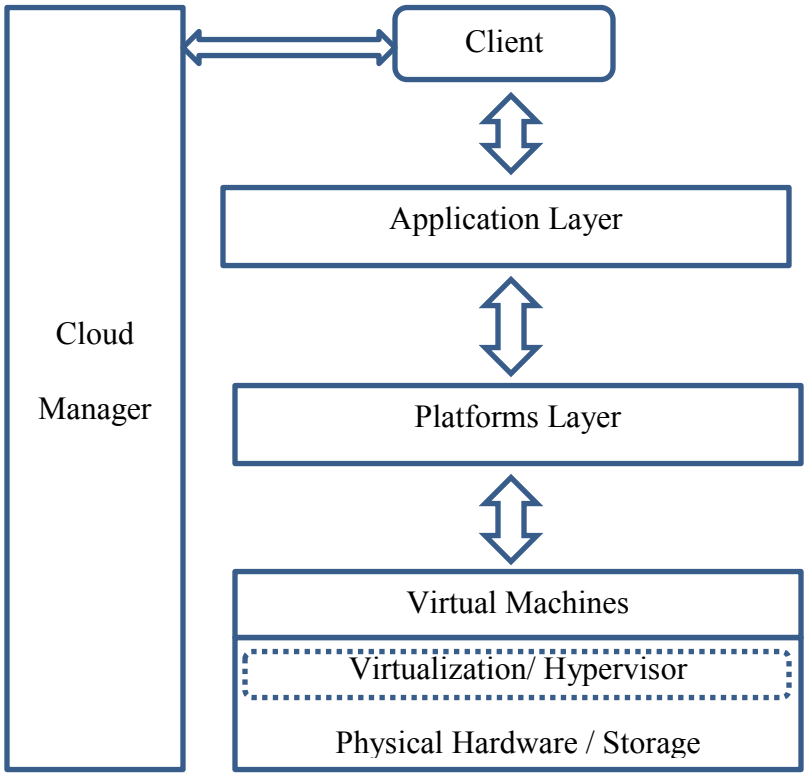

Figure 2 Cloud Architecture

There are many other modules such as service management, security management, policy management, data protection etc. that take care of the whole utility management and service provisioning. The Platform layer provides all the middleware and resources required for deployment of applications on top of virtual machines. Finally the application layer provides the SaaS or applications requested by the user.

2.5 SUMMARY

Although cloud computing has been very productive and popular there are many research challenges associated with cloud computing mainly because of security threats. There is a wide range of research that is going on cloud computing mainly focusing on new applications or

services a cloud can offer. Lot of research work is going on allocation of resources in an efficient way. Main concerns in cloud are the increasing number of users resulting in more energy consumption and heavy environmental pollution. Existing cloud architectures are centralized where all user requests are directed to a central server. There is therefore a single point of failure. Furthermore a single central point results in a bottleneck with high latency and network traffic. In the existing cloud model there is therefore only a single point of entry to the cloud for all users. Due to its centralized infrastructure, existing cloud over-provisions resources which incur high energy consumption, and high latency. Due to the increased number of users security has become more complex as the cloud is shared by many users.

CHAPTER 3

DISTRIBUTED CLOUD ARCHITECTURE

Distributed cloud computing refers to managing and provisioning of distributed resources. Furthermore, the distributed cloud is dynamic as resources may move in and out of the cloud. We propose a distributed cloud as resources and services provided by users in a P2P fashion, instead of huge data centers located in a single or multiple locations. It provides a completely decentralized mechanism for allocating and using resources, thereby avoiding a single point of failure. The motivation behind the distributed cloud is to make use of large number of machines like desktops, PC's and small servers which can run virtualization software on them.

Voluntary computing systems such as BOINC [27] are often mistaken as distributed cloud computing applications. Voluntary computing systems are completely different from the Cloud computing paradigm. In BOINC, the client installs a BOINC application which takes input from a central location and uploads the results. Currently BOINC applications are run on clouds as well.

Current cloud providers make use of multiple datacenters to handle the increasing number of users efficiently. If we expand this idea of multiple datacenters to millions of individual machines, we can build a distributed cloud using a peer-to-peer (P2P) model [29]. Such a peer-to-peer model of a distributed cloud can provide resources without any centralized architecture. Individual machines, such as desktops PCs, laptops, and servers, can run virtualization software to create multiple virtual machines (VM) and contribute to the distributed cloud. Distributed applications where data needs to be placed close to the end user can benefit from the P2P model distributed cloud, as a user can select peers closer to him. The Distributed cloud formed using

these machines provides storage and computation resources similar to the existing cloud, but in a distributed manner. The Distributed cloud is dynamic in nature as users join and leave the cloud.

Some of the key differences of a distributed cloud, existing cloud and voluntary computing are shown in Fig 3. As seen from table below we can see that distributed cloud uses features from both the existing cloud model and voluntary computing systems.

Distributed Cloud	**Cloud**	**Voluntary computing**
Distributed system	Centralized system	Distributed system managed by centralized entity
Virtualized environment	Virtualized environment	Runs applications
Connected using P2P	Centralized architecture	Centralized architecture
Unpredictable nature	Highly predictable	Unpredictable nature
Machines' provided by users	Datacenters	Machines' provided by users
Can form public, private or hybrid	Can form public, private or hybrid	Public
None	Amazon, Google, Azure, etc.	BOINC, SETI, Planet lab

Figure 3: Differences between different computing architectures

3.1 PROBLEM STATEMENT

When it comes to modeling, there is no specific cloud resource description [6] which describes the resources in a cloud. Computer networks and their resources can be described using many existing frameworks such as RDF (Resource description framework) [61] and Network description language [62]. Virtualization [4] technology can be applied on basic machines. The increase in cloud users makes existing desktops, laptops, and storage devices etc. are not used to their full capability. In order to make use of these machine we propose a distributed cloud where users can

share and use others resources in a distributed fashion. We need a distributed cloud model that can effectively make use of resources provided by the user. We can make use of these resources distributed across the globe.

Apart from the resource description framework we need an architectural model that describes the construction of a distributed cloud. We can use a hierarchical structure or a flat structure to form a distributed cloud. We first describe the hierarchical model and evaluate the advantages and disadvantages of the model. Then we describe a completely decentralized distributed cloud.

Traffic is the main concern with the existing cloud architecture. Hidden costs, high latency and need for high bandwidth to support more users are main concerns with the existing cloud. The advantages of distributed cloud include efficient energy usage, bandwidth conservation and low network capacity. The basic distributed cloud architecture is shown in Fig 4.

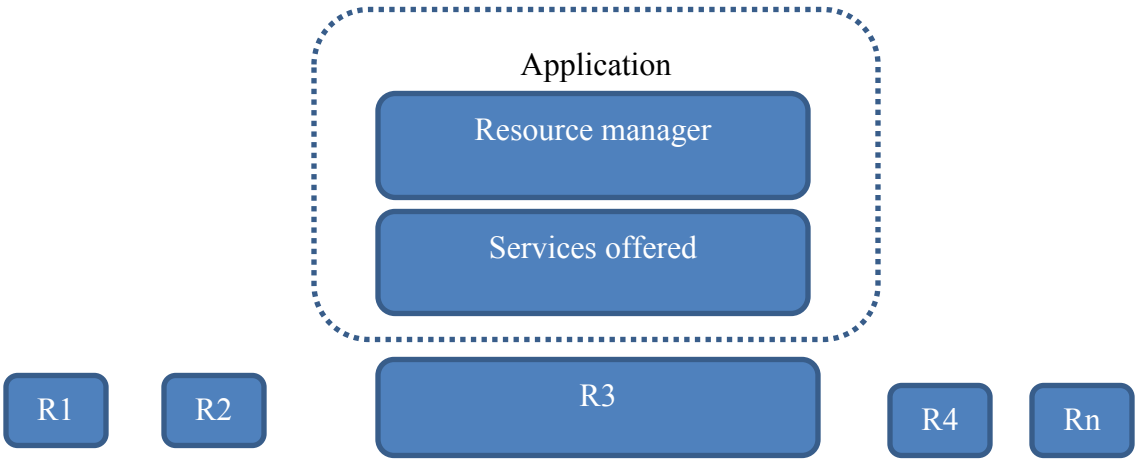

Figure 4 Distributed cloud

A client is the end user and uses applications to access the services offered by other users and to provide his own to others. The resource manager is responsible for managing the resources based on the attributes described by the resource model and provides security for the services offered. The resource manager is a black box as of now. Based on future requirements and modules identified we can add more layers to the system. In our model, the resource manager module is

responsible for finding resources using the algorithms we propose to discover and allocate resources. The service layer is used to identify and manage the types of services that can be offered i.e. this layer helps the user to choose the types of services that can be offered. It directly talks to the resource manager to validate the type of service required by a user and also updates the resource availability. R1, R2…Rn are the distributed resources in the cloud. Each resource will have the whole software running on their machines. It is similar to the peer to peer clients running on individual machines.

3.2 LITERATURE REVIEW

There are different implementations of cloud computing. Mobile computing [66, 67] faces the problems of resource scarcity, connectivity and mobility. Mobile cloud computing has emerged to tackle some of these problems and its architecture is shown in Fig. 5. Currently mobile cloud computing uses the existing centralized cloud architectures to run their applications on cloud providers. This takes care of the problems of resource scarcity and mobility, but connectivity remains an issue. Running mobile applications on the cloud brings out the new problem of latency. Mobile cloud computing can be defined as the ability run some of commonly used applications such as Gmail on a remote server, with the mobile device acting as a thin client connecting to these servers either over the network or internet.

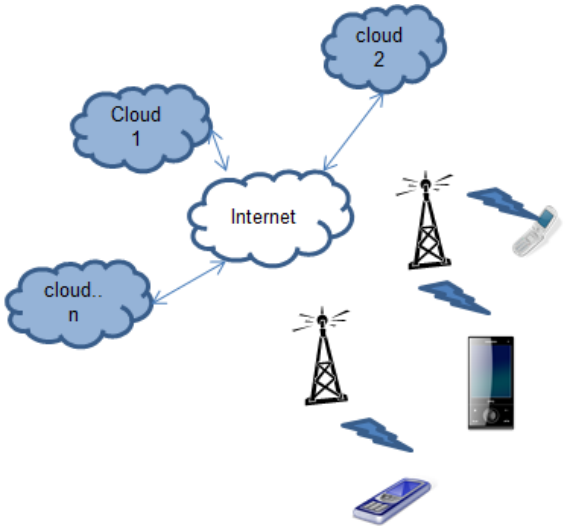

Figure 5: Mobile cloud computing

Mobile cloud computing schemes need high connectivity to data, on demand access to wireless to connect to and transfer to and from the cloud. With high latency, mobile devices run into battery issues and large processing times [68]. Use of the distributed cloud computing would be beneficial in this scenario. User can use mobile devices to do computation and run applications on nearby resource providers rather than connecting to the cloud data centers which may be more remote.

VANETS, Vehicular ad-hoc networks have been combined with cloud computing to form vehicular cloud computing. Drivers of vehicles can make use of mobile cloud computing, but face the problems discussed earlier of battery limitations and processing time [68]. One solution is to connect VANETS to the existing cloud architecture to form vehicular cloud computing [69] as shown in Fig. 6. However, the below architecture still suffers from high latency and connectivity problems. If the cloud resources are close by the performance of these networks would be improved. The use of distributed cloud architecture to run application close by the user would therefore be beneficial.

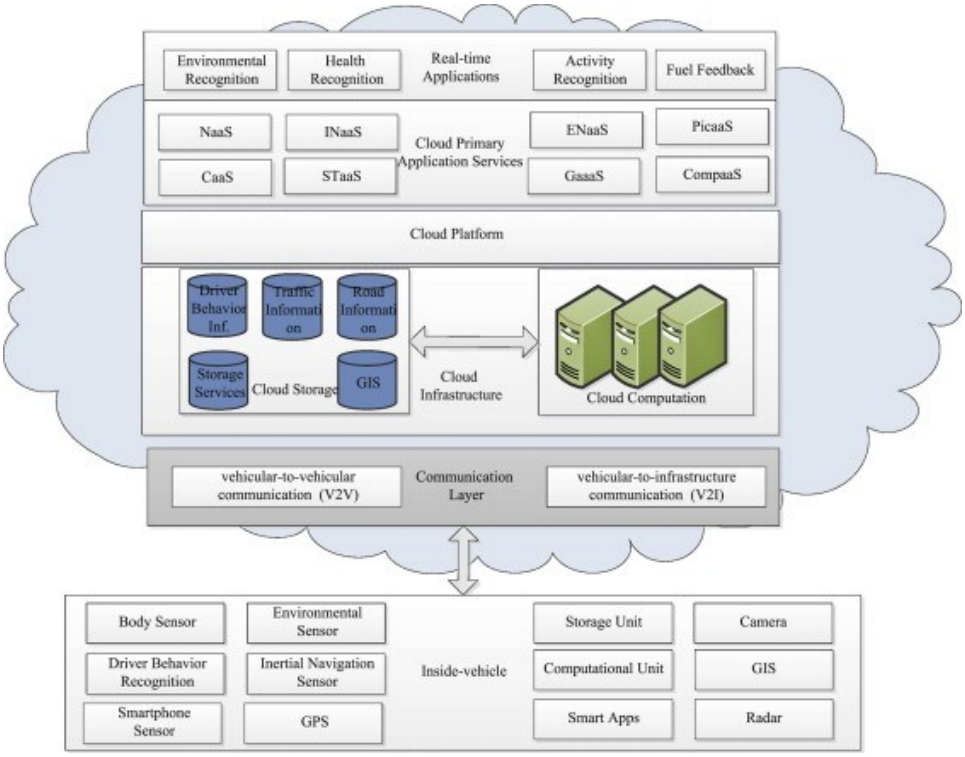

Figure 6: Vehicular cloud computing

All the current architectures make use of centralized cloud architecture. All these applications can be implemented using the distributed cloud computing model described in section 3.4 with high reliability and low latency. Future work looks into how to implement these mobile and vehicular cloud models using our proposed distributed cloud and how to make sure the data is secure and privacy can be preserved in these distributed cloud computing models.

3.3 RESOURCE DESCRIPTION FRAMEWORK

In the distributed cloud we have resources offering services across the world. These services are allocated to the user using a resource management system. We will create a resource model which defines the resources based on the services offered. The distributed cloud resource providers will

describe the resources and services they offer to the resource manager. Resource modeling describes the resource offered by the cloud and is used in describing cloud resource management and control of resources. The resources distributed have many characteristics which should be defined properly based on functionalities and services offered. These characteristics would be not only useful for identifying and analyzing user requirements properly but would also help to locate resources accurately as per need. The resource offered by each node can be modeled as shown in Fig 7.

The same resource model can be used for both the distributed cloud and cloud computing system. Each of the resources modeled are detailed as follows:

- The resourceID is a unique identifier for each node.
- The location of the node can be an important factor if localization is desired. The location of each node is represented by its network coordinates.
- Network coordinates maps the nodes to a predefined geometric space.
- Cores and RAM are used to identify resources at a fine grained scale like number of cores or gigabytes of memory.
- The bandwidth indicates how fast data can be stored or retrieved. This can be an important factor if huge amounts of data need to stored or obtained.
- The storage capacity is how much storage the node has allocated for the distributed cloud. Both the available and total capacities are reported.
- The availability denotes how busy the node is. If it is receiving a lot of storage requests, it will set its availability to low or none. This does not indicate the amount of storage capacity remaining; it can be used for load balancing.

- Operating system indicates the type of OS running on the virtual machine.

- Name denotes the type of service offered by resource. Ex: Java for PaaS

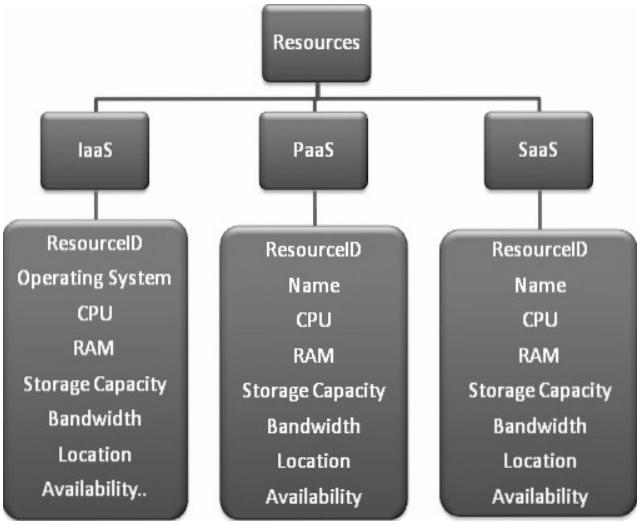

Figure 7 Resource description framework

All these parameters are defined for different types of service models. There are different types of services classified under these service models such as Storage as a Service, Data as a Service etc. Again actual services offered under these services may or may not use these parameters. For example if we consider Storage as a Service it only needs information such as resourceID, storage capacity, bandwidth, location and availability. The main goal of a resource description framework is to identify the attributes of cloud services.

3.4 HIERARCHIAL MODEL OF DISTRIBUTED CLOUD

Based on the service offered, the cloud infrastructure is composed of storage cloud, computing cloud, infrastructure cloud and data cloud. Using the hierarchical model shown in Fig 8, we can offer storage, data, computing and infrastructure as services to the users. We first identify users based on the area or locality and divide the whole system into zones and locations which will help us manage the resource efficiently.

The global resource manager is the first point of contact for the users and zonal resource manager is the layer till where users will be able to choose resources. The zonal resource manager is in charge of collecting information about the local resource managers. All the end nodes are connected to the local resource managers which keep track of information about nodes. The local resource manager is defined based on the location radius. We use a hierarchical model for the resource management system to effectively handle the high number of user requests and managing them in parallel based on the requirements. The user request based on user location and service requested will be routed to the zonal resource manager. The zonal resource manager based on the location assigns the task to a local resource manager. The local resource manager checks the requirements and based on resource availability and network bandwidth either assigns the resource in that location to the user or transfers the request to a different location from the same zone. Each service will be assigned with a unique VM except for storage. Multiple users can use the same storage based on availability. Using the hierarchical model, managing the resources would be more efficient.

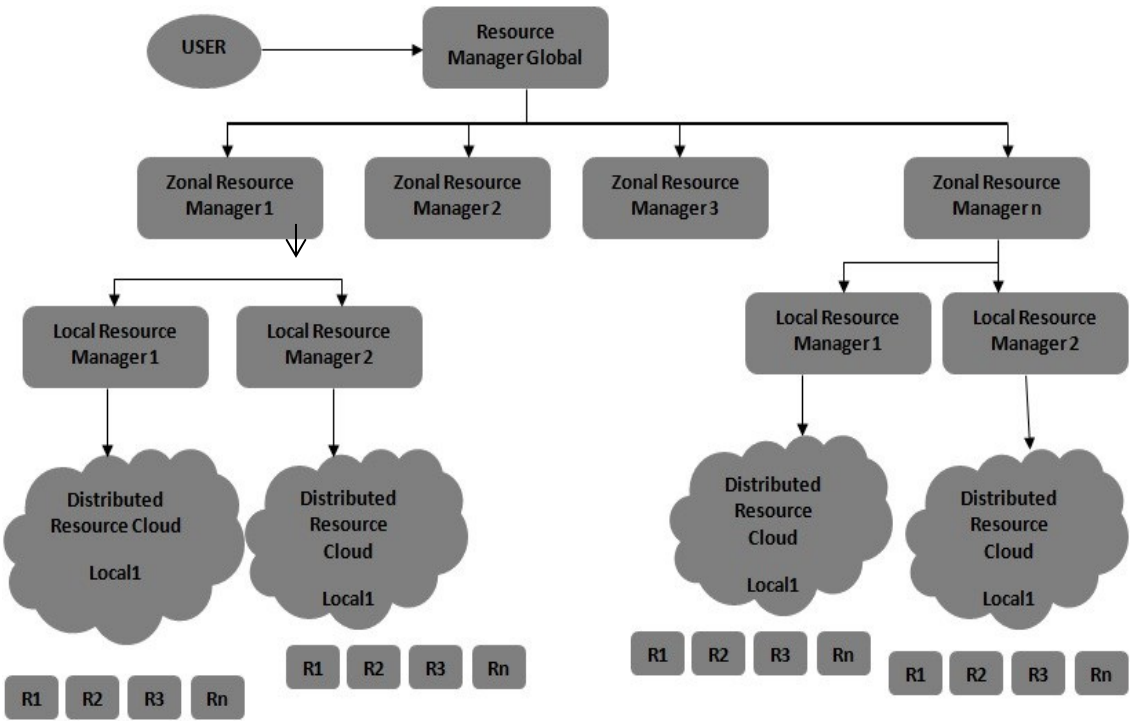

Figure 8 Hierarchical model of distributed cloud

We need optimizing techniques for allocation of resources. Optimizing techniques can be based on response time, throughput, and network traffic. To achieve the elasticity property of the cloud in a distributed cloud environment we need to use VM migration techniques. As long as a particular resource provider has enough resources elasticity is not a problem, but if the resources are not sufficient we need to look into virtual machine migration techniques to safely migrate resources to another resource provider.

3.5 DISTRIBUTED CLOUD ARCHITECTURE

Using the hierarchical model of distributed cloud we still have multiple points where the network is using centralized nodes. In hierarchical model of distributed cloud the super nodes act as

central points to the nodes under them. But since the load is uniformly distributed we need not worry about a single point of failure. The main problem using the hierarchical structure is how to elect nodes to be a zonal manger, local manager etc. If these are dedicated nodes similar to actual servers, the whole system needs to be maintained by a central authority. But if we want it to be completely distributed we need to find an alternative. To make the whole distributed cloud decentralized we developed a distributed cloud architecture that doesn't require any central authority to manage nodes.

Distributed cloud computing refers to managing, provisioning of distributed resources. We describe distributed cloud as resources and services provided by users in a P2P fashion, instead of huge data centers located in one single or multiple locations. Moreover it provides a completely decentralized mechanism of allocating and using resources, thereby avoiding a single point of failure. Users will be able to store data and perform computations on the proposed distributed cloud which should be efficient and a practical secure system.

In distributed cloud model shown in Fig 9, resource providers (RP) are distinctive i.e. these distributed cloud servers are individuals with resources to offer. Users of the distributed cloud need to discover these resource providers and request them for using their resources. Nodes in the distributed cloud can be both users and resource providers. The distributed cloud proposed uses a completely decentralized mechanism to discover and allocate resources. Users in the distributed cloud share resources in a P2P fashion.

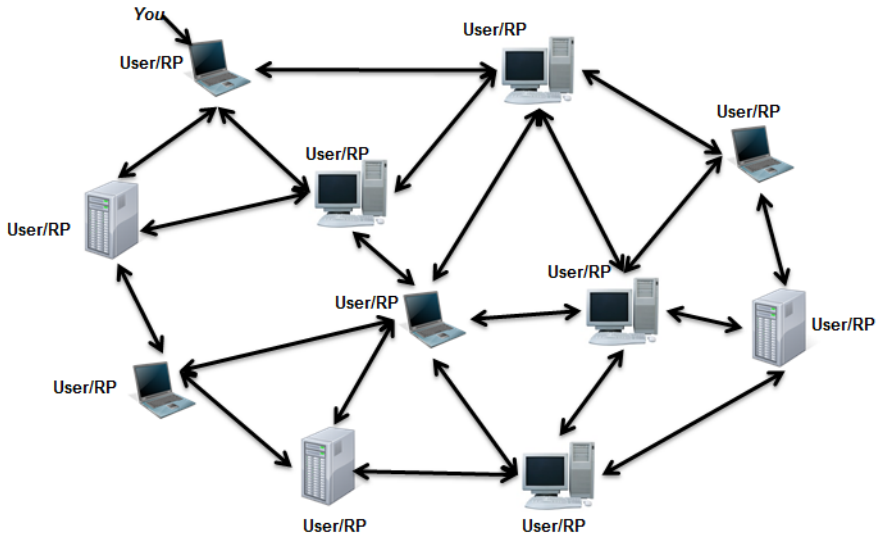

Figure 9 Distributed cloud model

The distributed cloud should have all the characteristics of the existing cloud architecture including proper management of resources, data security and privacy, and trust. The Distributed cloud provides scalability. Network constraints will be less using the distributed cloud which is internet based because data and resources distributed would be closer. Moreover latency would be reduced because resources would be chosen closer to the users. The distributed cloud would be free of cost as end users share their resources, but at the same time faces the problems associated with P2P systems such as free riding and performance issues.

In the distributed cloud model the user submits his job for the cloud in the form of a task to be performed and the required data resources that are present in the cloud. In Fig 10, we outline the execution flow for the distributed cloud. The user submits the job, data resources required and constraints over the job to the distributed cloud. A job may consist of multiple independent tasks or they may be dependent tasks. The workflow manager generates a workflow for the job based on the constraints and data and identifies the resources required to complete the job. It also will depict the work that can be done in parallel and hence can maximize the proper utilization of

resources. Once the resources requirements are identified, the resource manager will use our proposed resource discovery algorithm to find the appropriate set of resources. Based on the constraints, the proposed resource allocator will use game theory to allocate appropriate resources from the group of resources discovered. In this work we assume there is no collaboration between the resources discovered.

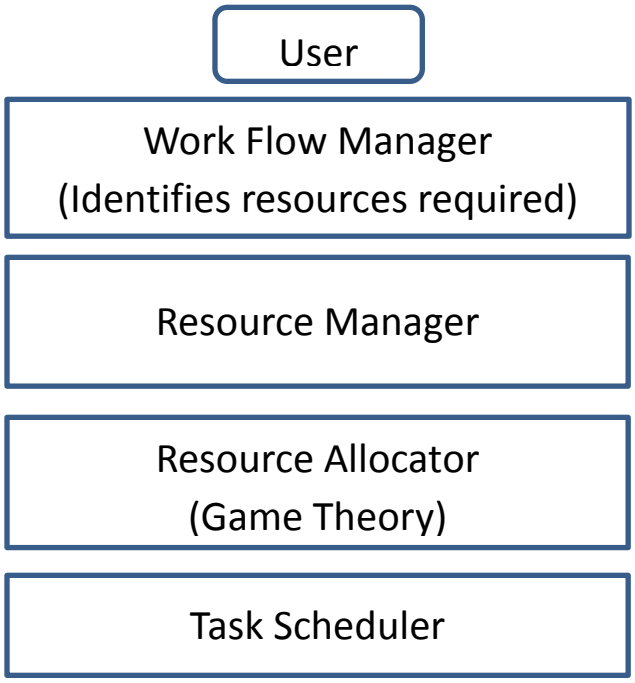

Figure 10 High level model of distributed cloud

Future work will focus on resource allocation for dependent tasks. Once the resources are allocated the task scheduler will schedule the tasks onto these resources efficiently based on the task constraints. If the tasks are dependent and require communication with other tasks we need to allocate the resources in a way such that overall resource utilization is maximized. We intend to work on this problem in the future.

CHAPTER 4

RESOURCE DISCOVERY MECHANISM IN DISTRIBUTED CLOUD

4.1 PROBLEM STATEMENT

The main goal of the distributed cloud is to evaluate and see if the system design can be practical in terms of efficiency. Designing and evaluating distributed cloud storage and computation is the main goal. Resource discovery in a distributed cloud model is a main problem. In current P2P technologies [9, 10, 11], using DHT's single attribute resources can be discovery. In a distributed cloud each resource is modeled with multiple attributes. Users request can include different attributes such as memory, cores, operating system, storage, availability etc. So we need a better mechanism to perform resource discovery.

To find a node we hash the IP address. But when we need to find a resource for computing we need to find resources which satisfy the user criteria. This problem of finding resources in P2P systems with multiple attributes is handled using range queries. Many range query schemes have been proposed for P2P systems. Since these range query schemes are built over existing architectures, they create a large overhead and moreover these are only used for data management. So, existing DHTs cannot be used for routing and resource allocation in the distributed cloud efficiently. A lot of work has been done on multi attribute range queries [19, 20, 21] in a P2P networks which are described in the literature review.

4.2 LITERATURE REVIEW

Distributed cloud computing makes use of underutilized resources and has its own advantages. The distributed cloud model provides all different services provided by the existing cloud in a distributed decentralized manner. Since users contribute these unused machines for the distributed cloud, they can make use of other resources that are offered by others. In distributed or grid computing a computationally intensive task is divided into multiple smaller problems and run on different machines. Such an approach can be implemented on this distributed cloud. Apart from file sharing that is similar to peer to peer networks, users can also store files on the distributed cloud.

Although allocation of computing resources in a distributed computing to solve a task is a NP-hard problem, there are models to optimize this allocation. Moreover in grid computing there is a centralized server which keeps track of other machines in the network. The Distributed cloud avoids the use of centralized servers to avoid single point of failure. The Distributed cloud allows users to select their own resources and store data in the resources they want. Although distributed computing and cloud computing share a lot of similar features, distributed cloud requires a completely different architecture to operate efficiently.

Peer to peer systems made file sharing and content distribution possible. Peer to peer systems make use of DHTs for efficient query lookups. Kademlia [9], a peer to peer, decentralized protocol is used by millions of people daily. In a network of size N users, it takes an average of $O(logN)$ hops to find a requested peer using kademlia. Kademlia has been shown to perform efficient query lookup. Every node in kademlia has a 160-bit node ID which is created randomly or generated from the IP address. Kademlia uses bitwise XOR metric for accelerated lookups. For each $0 \leq i < 160$ every node in kademlia keeps a list of information about other nodes of distance

between 2^i and 2^{i+1} from itself and these lists are called k-buckets. Kademlia uses four different RPCs; PING, STORE, FIND_NODE and FIND_VALUE. PING to check if node is online or not, STORE to save data, FIND_NODE to find k closest nodes given ID and FIND_VALUE works the same as FIND_NODE except that it can be used to retrieve a value given a key from STORE RPC. Routing is existing DHTs is as shown in Fig. 11.

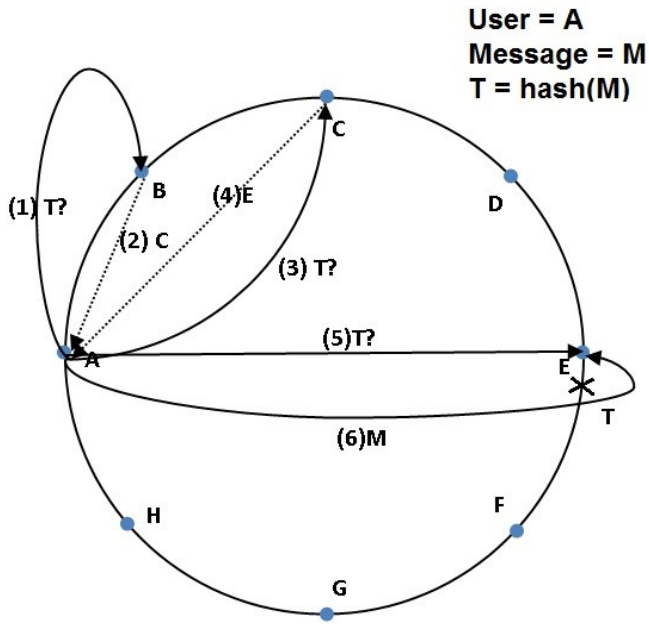

Figure 11 : Routing in a distributed hash table: 1) User A contacts B, the closest node to T in A's routing table; 2) B replies with C, the closest node to T that B knows about; 3) A contacts C; 4) C replies with E, 5) A contacts E and finds that E is the closest node to T; 6) A sends the file M to E to be stored.

Using existing DHTs for resource discovery is not possible as they only work for single dimensional queries which use one attribute. Moreover range queries are not supported by existing DHT's because of hashing. DHTs make routing simple if users want to find resources with an exact single attribute. In a distributed cloud, users specify multiple criteria for resources

and multiple resources may be available that meet user requirements. The problem then becomes identifying the best resource to be allocated. This requires range queries that satisfy multiple user requirements and then a set of best resources can be selected.

There are different range queries implementations used in P2P systems to find data in P2P systems with multiple attributes [19, 21]. In [19], authors used z-curves to partition data and arrange the subspaces into skip graphs. Query ranges are performed over these zones. Given a query range, a node computes superset of zones covering this range. In [21], authors used Hilbert space filling curves to map n dimensional data to a single dimension. In P2P systems these range queries look for a range of data whose locations are fixed and the data itself does not change. In contrast, resources in the distributed cloud are dynamic where resources enter and leave the cloud. The resource attributes (memory available for example) may change once a part of a resource is used. Existing DHTs cannot therefore be used for routing and resource allocation in the distributed cloud efficiently.

The Distributed cloud has been recently studied because of its benefits in sharing idle resources effectively. In [25], the authors merged voluntary computing and cloud and a new architecture was proposed, but this architecture makes use of a centralized mechanism to traverse the requests. Any user who wants to provide the resources should send a request to the centralized system and should accepts the rules and policies required by the cloud@home management system. In [24], the authors proposed a cloud model using P2P for storage but not for computation. They used a P2P system on the servers and used DHT to handle storage in cloud. In [26], the authors described a P2P cloud model which makes use of gossip-based protocol to manage a large P2P cloud. Though this model achieves building a P2P cloud, important things like resource discovery, user behavior and other P2P vulnerabilities are not mentioned. In a naïve way we can make use of multiple DHTs each for an attribute of resource to find the resource providers effectively. Large storage is required in order to store multiple DHTs and we need multiple

queries to process data from multiple DHTs i.e. the time required to find resources would be higher.

4.3 RESOURCE DISCOVERY IN DISTRIBUTED CLOUD

We use an efficient resource discovery model based on multi valued hash tables as it provides $O(logN)$ for finding a resource. This model uses kademlia for routing, lookup of nodes and uses a multi-valued hash table scheme for identifying resources needed with multiple attributes. We propose a new approach to resource discovery, having two different ID‘s, one to find a node and the other to find nodes satisfying user requirements. First we use the 2nd nodeID to identify the resources that satisfy user requirements and use the first nodeID to locate them. There is also a naive solution that can be used to perform resource discovery which is explained below.

4.3.1 NAÏVE SOLUTION FOR RESOURCE DISCOVERY

A simple and naïve solution to the problem of resource discovery in a distributed cloud is modifying the original Kademlia algorithm to handle multiple attributes (resources). The node ID can be changed to include resources information. Routing in Kademlia is performed using the XOR metric. Peers with similar first few bits in their IDs are “close” to each other. The resource information can be encoded into the node ID. Instead of the node ID being generated from the IP address only, the node ID is generated using both the resource information and node ID. However, for resource discovery to be possible, the node IDs has to be of some structure, such as

peers with similar resources are close to each other in terms of node ID space. The information representing the resources can be prepended to the original node ID, as follows:

$$nodeID = bits\ assigned\ for\ attributes + actual\ nodeID$$

Where + indicates concatenation. For example, we first decide the resource attributes needed and then assign bits for each attribute. If the two resources under consideration are processing power (CPU) and amount of memory, 3 bits can be assigned for CPU (for a maximum of 8 Ghz) and 5 bits can be assigned for memory (for a maximum of 32 GB). We then concatenate all these bits together and with the original node ID which is the hash of the IPAddress or hash of file name. If the original node ID size is *n* bits which is a constant and can be changed with increased number of users, the new node ID size is 3+5+n = 8+n bits.

$$nodeID = 3\ bits\ for\ CPU + 5\ bits\ for\ memory + actual\ nodeID$$

This new node ID is used to represent and locate peers. The routing table and routing mechanism stay the same. Each ID is now 8 bits bigger and the overhead is only 8, which is small compared to the original ID size of 160 bits. If user requires resources with fewer constraints only the required constrains are extracted from the nodeID and resources are retrieved. For example if user requires 5 resources with constraints such as 4 Ghz of CPU and 2 GB memory then all the nodes with CPU and memory greater than or equal to 4GHz and 2GB are retrieved. The nodeID size remains constant. If a user doesn't require memory then any resource with greater than 0GB would be considered. When a user requests for a node with a list of required resources, the user first calculates the first 8-bits, which indicate the resources needed. The peers who match these first 8-bits are located, and the remainder of the routing process is performed using the original Kademlia algorithm. Peers with the same first 8-bits are closer together and they all have the same available resources. The main problem with this solution is that it matters which resource is prepended first. In the example above, if a user only requires a certain amount of memory, many

query lookups are needed to discover all possible peers. Moreover, the size is fixed. If a user wants to share more than 32 GB of memory, the change is not incrementally deployable and the whole system needs to be updated. Finally, the IDs are not sorted. For example, an ID with 3 Ghz of CPU and 1 GB of RAM is located after an ID with 2 Ghz of CPU and 5 GB of RAM, in terms of the XOR metric.

4.3.2 RESOURCE DISCOVERY USING MULTI-VALUED HASHTABLE SCHEME

The resource discovery model proposed for distributed cloud computing is efficient as it provides $O(logN)$ for finding a resource. We evaluated this by finding the number of hops required to find a node. The kademlia scheme is used for routing and lookup of nodes. We introduced a new concept of distributed local multi-valued hash table to handle the problem of resource discovery in distributed cloud computing. Every node will have 2 ID‘s, one to determine the resource location and another to determine the attributes of a resource. We made use of kademlia‘s distributed hash table to locate a node efficiently. The 1st ID is determined as hash (IPAddress) and the 2nd ID as hash (attributes). In our approach each node along with its k-bucket maintains a multi-valued hash table. This multi-valued hash table gives information about nodes in its routing table. Each node constructs its own multi-valued hash table by storing key value pairs of <nodeID2, a set of nodeID1’s > in its routing table. The k-buckets and multi-valued hash tables are updated whenever a node receives messages and configuration changes such as change in attributes, nodes move in and out of network etc.

$$nodeID2 \rightarrow hash(set\ of\ resource\ attributes)$$

$$nodeID1 \rightarrow hash(IPAddress\ of\ node)$$

Resource discovery in this model is done by checking its multi-valued hash table and find nodes that match user requirements. If nodes are not found in the local table, a request to find nodes with user requirements is sent to the nodes in routing table. These nodes check their multi-valued hash tables and the process is continued until resources that satisfy user requirements are found. For example, if we consider two parameters, cores and memory we can have the 2nd nodeID as hash (cores, Memory). There might be many nodes with the exact cores and memory, so for each nodeID2 there might be a set of nodeID1's assigned as value. Hence we use 2nd nodeID to determine available set of resources and the 1st nodeID for node lookup and routing.

Consider an *n* dimensional space with *n* attributes. Given the resource requirements we can identify all the resources that can satisfy the requirement using *n* dimensional graphs. For example, if a user requests for a resource with 2GB memory and 2cores, from Fig.12, we can see that all the nodes from the shaded region satisfy user requirements since memory is greater than 2GB and cores greater than 2. So we find all nodeID2's which fall in this region. We use a multi-valued hash table algorithm to find all these nodes which can satisfy user requirements.

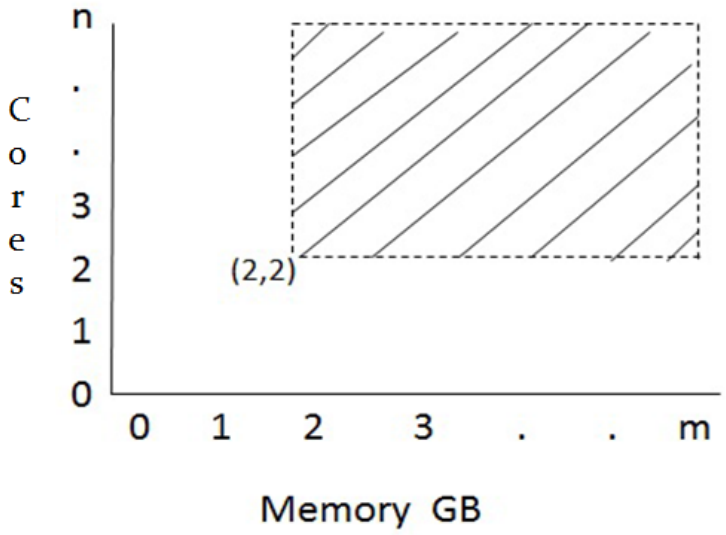

Figure 12: User request for a node with 2 cores and 2 GB memory can be satisfied by nodes in the shaded area.

Assuming there are two major attributes, cores and memory to find a resource we developed and evaluated the resource discovery algorithm shown in Fig. 13 which is explained below. When a user requests a resource with X cores and Y memory, the application layer based on a threshold creates a list of nodeID2‘s that can satisfy user requirements. The Threshold defines the acceptable range of resources which can satisfy user requirements. It is determined by the system. Each of these nodeID2’s in the list has cores and memory greater than or equal to user requirements in the X+threshold and Y+threshold region. This list is used to find the set of resources that can satisfy the user requirements efficiently. User starts by searching his routing table for nodeID2. If he finds nodeID2 he will get a set of nodeID1’s. If the number of nodes found reaches the thresholdCount the system stops additional querying for the resources. Otherwise the user will send a request to find nodes to other nodes and repeats the process until number of resources reaches the thresholdCount. ThresholdCount gives number of nodes to be found to make sure user has choices to select the best node which satisfies his requirements efficiently.

For example: user requests for 2 cores and 2GB memory

- Find all nodeID2 corresponding to user requirement based on thresholds x and y.

$$nodeID2 \rightarrow hash(2+x, 2+y)$$

- Find CheckList: find all other nodeID2‘s that can satisfy user requirements i.e. (cores, memory) as (2,3),(3,3),(3,2)...
- Check the nodes routing table to see if there are nodes with initial nodeID2. If found add them to final list.
- Check size of final list. If enough nodes are found start with resource allocation. if not found repeat the process for other nodes in the CheckList

- If there are not enough nodes that can satisfy user requirements look for the nearest nodes in the routing table and repeat the whole process.

Algorithm: Resource Discovery

```
//user asks for n resources with X-Cores and Y-Memory
nodeID2=hash(X,Y)//Find nodeID2
//Find all resources which can satisfy user
CheckList=FindAll(nodeID2);
For each nodeID2 in CheckList
    {
//user sends a message to all closest nodes in his k-bucket
    FinalList.find(nodeID2);
    If contains nodeID2
        {
        NodeID1=node.get(nodeID2)
//if a node contains NodeID2 get set of NodeID1's associated with NodeID2.
        For each node in NodeID1
        {
//n is number of resources requested. Threshold value is needed for load-balancing. We need
//to find adequate nodes so that we can perform our game on then to find best resources for
//user.The game is covered in section 5.3
        If(FinalList.size<n+thresholdCount)
        FinalList.add(NodeID1)
        Else
        {
            Stop Searching;
            Return FinalList;
        }
        End If
        }
    End If
    }
```

Figure 13 Resource Discovery Model using Multi-Valued Hash tables

Using the distributed cloud users can request for a single resource or set of resources. Since we used kadelmia [7] the whole process would be done asynchronously. Moreover since we use two different ID's there is no need to update the routing ID. User can use the first *n* resources found that meets the user requirement. But the resources found that match's user requirement may be far

away from the user or has large latency or low throughput. So in order to find the resources that best match the user requirements and also has low latency or high throughput or low incentives we use game theory to allocate resources effectively. We use auction games to avoid the free riding problem.

Since for the distributed cloud storage there is only one attribute, namely, storage, we implemented the distributed cloud storage model using the existing kademlia. For the distributed cloud computing there is more than one attribute. Hence, we implemented the distributed cloud computing model using a multi-valued hash table scheme by modifying kademlia. More details about the implementation are explained in the evaluation section.

4.4 EXPERIMENTAL ANALYSIS

4.4.1 DISTRIBUTED CLOUD COMPUTING

We have proposed a new mechanism to discover resources with multiple attributes in a distributed cloud. To determine if the proposed method works efficiently when compared to using the existing kademlia [9] for resource discovery, we simulated a distributed cloud computing model. The simulation made use of kademlia, a peer to peer protocol.

We wrote our simulator using java1.7. We made use of kademlia for routing purposes. We used the King dataset [17] to simulate the latency between nodes. Each node has different attributes such as id, memory, capacity, memory, cores, and availability. Each node maintains its own routing table and a multi-valued hash table. We used kademlia routing table in our simulator with each node maintaining 3 k-buckets with bucket size set to 1. We wrote a ping event to check if a

node is online. A PING event pings all the nodes in its routing table and verifies the availability of nodes. If the node doesn't get any reply for the ping the other node is considered offline. Each routing table and multi-valued hash table is updated when a new node joins and whenever a new node is discovered during the process of searching other nodes. After a node has assigned its resources to another node it deducts the resources assigned. It then updates its multi valued hash table and then informs about the change to nodes in its routing table. Once the processing time is finished and resources released, the node updates its new availability and multi valued hash table.

The new resource discovery mechanism was implemented by modifying the existing kademlia. The simulation network consisted of 500000 nodes which joined the network at the beginning of the simulation. Each node has a 32 bit unique ID generated randomly. Each node has a mean memory of 8GB; the minimum is 1GB and maximum is 16GB. Each node has cores which are randomly assigned from set of valid cores 1, 2,4,6,8 or 10. To make sure that simulations would be more realistic we are biased towards lower core values when assigning them. This is the assumption we used because in the real world most of the machines offered would be small machines with basic computation power i.e. lower number of cores and low memory. In our simulations we had 50% of nodes with 1 core, 25% of nodes with 2 cores, 10% of nodes with 4 cores, 8% of nodes with 6 cores, 4% of nodes with 8 cores, and 3% of nodes with 10 cores. We also made sure that most of the requests to the distributed cloud are more biased towards the lower values of cores and memory, i.e. users requesting nodes with lower computing power. In our simulations 80% of nodes requested 1 core and memory up to 4GB, 10% requested 2 cores and memory ranging 5 to 11GB, 5% requested 4 cores, 2% requested 6 cores, 1% requested 8 and 10 cores and all 4, 8 and 10 core machines requested 12-16GB memory.

Each node requests a resource or set of resources with randomly assigned capability. Requests are processed over a normal distribution with a time of 6 hours; the minimum is 1 hour and the maximum is 6 hours. After the requests are processed, the resource provider (RP) of respective

nodes selected will deduct the resources provided till the processing time is finished. Once the process is finished, the resource provider RP will change resource attributes based on the new availability. Each node stores a multi-valued hash table with the key being nodeID2 which gives information about the computing resources available with a node and value being a set of nodeID1's.

The simulation is run with all 500000 nodes joining at the beginning of the simulation. We ran our simulation with each node requesting "k" nodes for simulation. We used "k" as 1,2,3,5 and 10. We used different "k" values to find out how many nodes can be successfully found, so that we can allocate one of the best nodes from "k" to the requested user using our resource allocation game method. We calculated the number of searches/queries required to find a resource. This also infers that if searches/queries are less the processing time to find a requested resource is low.

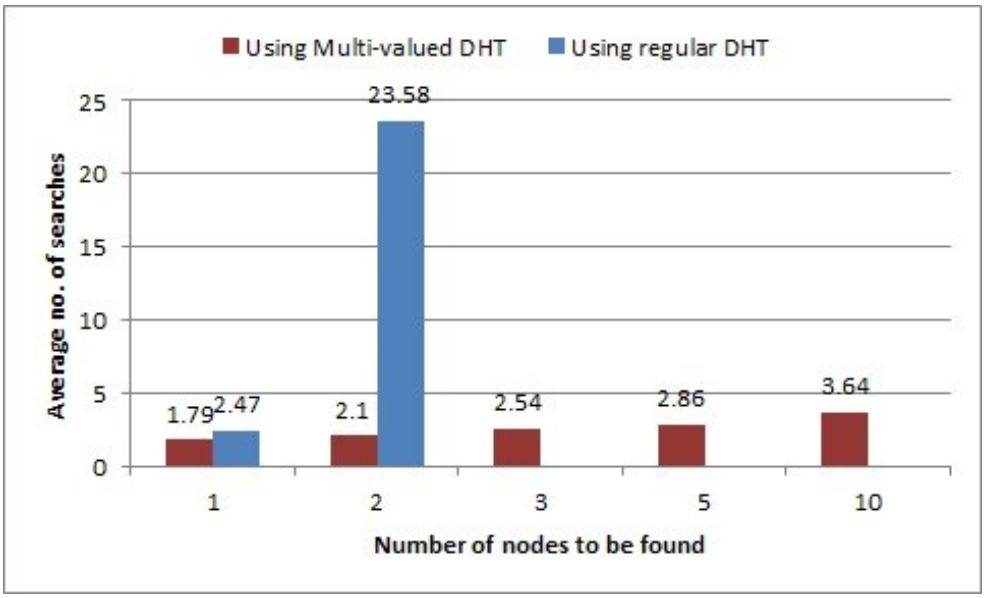

Figure 14 Average number of searches to find K nodes

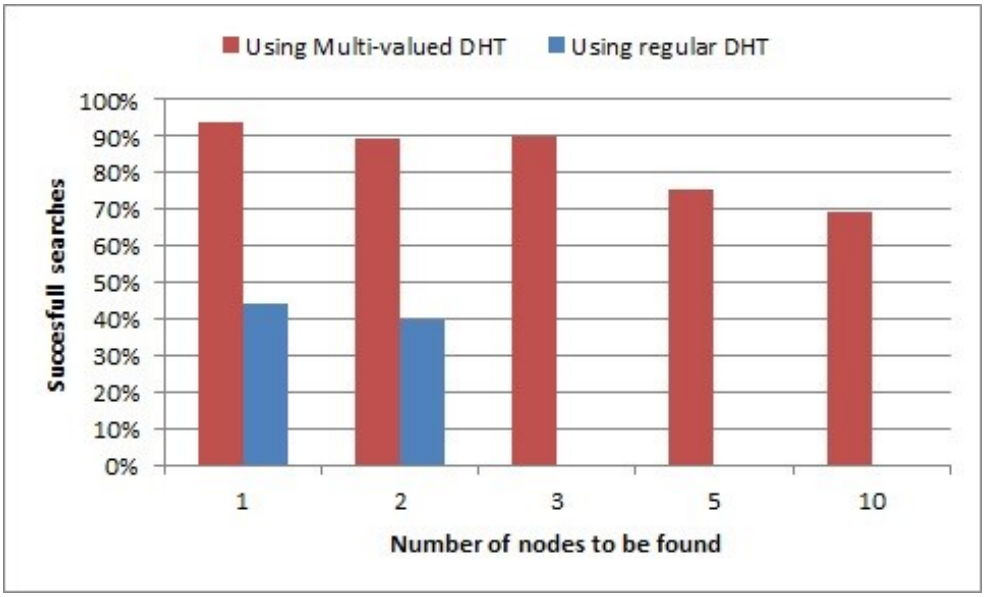

Figure 15 Percentage of Successful searches to find K nodes

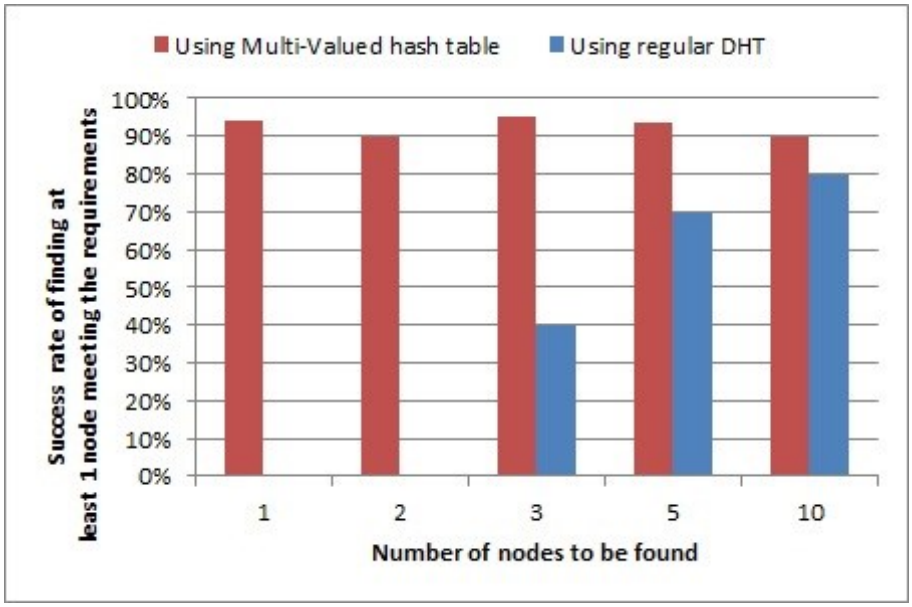

Figure 16 Percentage of successfully finding at least 1 node from incomplete searches

We implemented resource discovery using regular DHT and we observe that it took more than 25 searches to find a node compared to our multi-valued hash table scheme. We have set the search limit to 25 and then performed the resource discovery using the regular DHT. Fig. 14 shows the average number of searches taken to find “k” nodes using our multi-valued hash table and regular DHT. We see that the numbers of successful searches to find “k” nodes are a lot fewer using our

proposed multi-valued hash table scheme. The average number of searches for finding 10 nodes is 3.64, and from Fig. 16 we can see that more than 70% of time we can find 10 nodes which match user requirements within 4 search queries.

In Fig. 15 we compare the success rates of regular DHT's with a multi-valued DHT. We observe that the percentage of resources found i.e. successful searches is far higher using our resource discovery mechanism. We can see that the success rates for finding nodes is about 90% for k=1, 2, 3 and a little less for k=5 and for k=10. We also see that the success rate using regular DHT is lot lower. Using regular DHT, the success rate for finding k=3, 5 and 10 nodes is zero.

We also evaluated the percentage of finding at least 1 node from unsuccessful searches to finding 2, 3, 5 and 10 nodes. From Fig. 16 we see that the success rate for finding at least 1 node which meets the requirements using different "k" values is lot higher using our multi-valued DHT compared to using regular DHT. From these experiments we calculated the number of hops required to find a node in the network. Our scheme took an average of 2.4 hops compared to using kademlia which took an average of 2.6 hops to reach a node. From these results we observe that our proposed model of distributed cloud is practical and more efficient than existing schemes.

4.4.2 DISTRIBUTED CLOUD STORAGE

To determine whether our proposed concept of distributed cloud is feasible storage, where every user's machine can be a storage device for other users, we simulated a network of users and file storage. The peer-to-peer protocol used was kademlia. The network consisted of 10,000 nodes which all joined at the beginning of the simulation. Each node is assigned a randomly-generated unique nodeID. The size of each nodeID is 10 bytes, instead of Vuze's 20 bytes due to the smaller

network size. Each node randomly contacts another node which had already joined for bootstrap (obtain information about other nodes). The other parameters of the protocol are similar to what is set in the Vuze BitTorrent peer-to-peer client. The number of replicas for each file stored is set to 10, the bucket size of each routing table entry is 20. Every 10 minutes, a node pings another node in its routing table to determine if it is still alive. Every 30 minutes, a node picks a random nodeID and performs a FIND_NODE. This allows a peer to find other peers in the network.

Our simulation contains a few assumptions. We emphasize that the simulation is a proof-of-concept that a distributed cloud for storage is feasible and practical. Future work includes more realistic experiments. Each peer provides a mean storage of 50GB for the network; the minimum is 10GB and the maximum is 100GB. At the beginning of the simulation, each peer ``generates'' a random number following a Gaussian/Normal distribution with a mean of 50GB. With 10,000 peers in the network, this provides an average of ~500TB of possible storage for all users. Each peer also has an average of 100 files to store in the distributed cloud; the minimum is 1 file and the maximum is 500 files. The average size of each file is 1MB; the minimum is 1KB and the maximum is 100MB. The fileIDs, mapped on the same space as the nodeIDs, are randomly generated. Following Vuze's parameters, all files (metadata for Vuze) are deleted after 24 hours. This implies that every 24 hours, a node has to perform STORE operations to keep its files stored in the distributed cloud. Each file is replicated 10 times and stored at 10 different other peers. If a file is stored at least once, this is counted as a successful storage.

After storing its files in the cloud, every peer also has to search and retrieve the files. Searches for each file follow a normal distribution with a time of 24 hours; the minimum is 1 minute and the maximum is 48 hours. Every search also has a timeout of 30 seconds. After 30 seconds of searching and not finding the file, the search is considered to have failed. The peer only needs to find one copy of the file, not all the replicated copies.

The protocol described is very similar to a peer-to-peer file-sharing network. The major differences are storage is provided. A peer can store its file at another peer. Afterwards, a peer has to be able to search and find its previously stored file.

The simulation is run for 3 simulated days, with all 10,000 nodes joining at the beginning of the simulation. The King dataset is used to simulate the latency between peers. From the list of latencies from the King dataset, one latency is randomly chosen for the simulated latency between two peers. Each request/reply processed at a peer is also simulated to take 1 millisecond. This processing time is very likely an overestimate on the actual processing time. The average time to successfully complete a store operation is 1.9 seconds and the average time to successfully complete a search operation is also 1.9 seconds. The percentage of successful store operations is 98.98% and the percentage of successful search operations is 91.21%. The high success rates show that a distributed cloud can be used efficiently and effectively for distributed storage.

Since peers can join and leave the network at any time, churn is also simulated. Every peer has a 25% probability of going offline. If a peer leaves the network, it leaves for an average of 12 hours. Afterwards, the peer will rejoin the network after 12 hours. 12 hours is chosen as this is half a day. The times and success rates are similar to the non-churn scenario above. The average time for a store operation is 1.9 seconds and for a search operation is 1.9 seconds. The success rate for a store operation is 98.87% and for a search operation is 91.69%.

Based on the success rates obtained for store and search operations, we find our scheme more efficient and practical in terms of consideration for implementation in real time.

4.5 CONCLUSIONS

We proposed a distributed cloud computing and storage framework, and in particular we described the distributed cloud model, node architecture and a resource discovery model to find resources. We find our method took very few searches to find nodes for distributed cloud computing when compared to using regular DHT. Using our method, the success rates for finding nodes are 90% with searches less than 3, when compared to success rates of 40% or lower with number of searches greater than 25. We also find that, with churn, success rate for a store operation is 98.87% and for a search operation is 91.69%.

We see the distributed cloud as an emerging model in the next few years with a lot of research remaining to be done. We see from the evaluation that the distributed cloud is a practical model that will evolve over the next few years. As future work, we will explore more realistic experiments, such as file changes, more nodes, and more realistic times and parameters. We will look into adding churn to the distributed computing emulation.

CHAPTER 5

GAME THEORETIC APPROACHES FOR RESOURCE ALLOCATION

5.1 PROBLEM STATEMENT

The main goal of the distributed cloud is to alleviate, if not eliminate the disadvantages of a centralized system, namely, single point of failure and communication bottlenecks by allowing distributed computing and storage. In the distributed cloud, users are both the providers and users of the resource. Resources are discovered using the multi-valued resource discovery mechanism described in chapter 4. After a set (or range) of resources have been discovered, a subset of these resources are allocated based on certain criteria (performance requirements for example). Resource allocation based on user SLA (Service Level Agreement) is a NP-Hard problem and task scheduling on to the resources available is NP-complete. Hence optimized resource scheduling is achieved only for a particular user task or using heuristics. The user specified job may require the resources allocated for the tasks to cooperate among themselves to finish the task. Alternatively, the user specified task may be a completely independent task. In this work we only consider independent tasks where each resource or set of resources allocated perform independent work i.e. there is no need for cooperation among them. The proposed distributed cloud is user centered, where users do not pay money to use resources. Instead, we propose a barter kind of system where users provide resources and use resources. Hence, the question of fairness comes up. Free riders who use resources, but do not provide resources for others must be prevented. Hence after resource allocation the system must be stable. In our case, a system is said to be

stable when the allocation is fair, that is, users who provide resources for other users are more likely to get the resources they require.

5.2 LITERATURE REVIEW

Game theory has been used in existing cloud computing technology for resource allocation such that total payment is reduced for the user. The paper [29] talks about assigning tasks to resources in the existing cloud and gives an optimal way to allocate tasks to resources, but in a distributed cloud there is no central authority to manage resource provisioning. Moreover in the cloud architecture, resources can be defined and generated at runtime. Hence, based on tasks resources can be defined. In the distributed cloud we don't know about the availability of resources, therefore a game theory mechanism is used to create resources based on availability and latency. In [23] the authors used a prediction scheme to predict future bids and used bids to schedule resources which fit with bids. In our case there is no cost involved and we focus on utility for user and resource provider.

We introduce a novel incentive based mechanism and efficient resource allocation so as to avoid free riding in our proposed model of distributed cloud computing. Hence P2P technology and game theory can be used to develop efficient models for resource provisioning in the distributed cloud.

5.3 A GAME THEORETIC APPROACH FOR RESOURCE PROVISIONING IN THE DISTRIBUTED CLOUD

In the distributed cloud architecture, resource discovery and allocation are the main problems. In this chapter we address the resource allocation problem. Locating a proper subset of resources from a widely spread distributed cloud to satisfy users requirements is a significant challenge. This calls for efficient resource discovery mechanisms and resource allocation methods based on user requirements and applications.

Resources must be allocated based on application requirements such as high performance or high throughput while taking into consideration the free riding problem. In this chapter we use game theory as a strategic decision making mechanism for allocating resources to users while maintaining system stability. In particular we introduce an auction model for resource provisioning where incentives encourage users to provide resources. These incentives are in the form of a payback scheme where providers are given preference when they request resources. Hence the more resources users will provide the more they can use the resources in the system thereby avoiding free riders. The proposed game schedules resources such that the whole system is stabilized. We describe an incentive based auction mechanism which is utility driven. Once the resources are discovered users and resource providers' form a group and the game is performed among them.

After resource discovery part is done we use the first *n* resources found that meets the user requirement using this method. Discovering only the exact number of resources requested by a user is not sufficient. Although these may satisfy user requirements, they may be inefficient to use because of, distance from the user or large latency or low throughput. So in order to ensure

efficient allocation in the distributed cloud, we find resources that match user requirements but also has low latency or high throughput or low incentives. We use game theory to allocate resources effectively and to avoid the free riding problem as well.

Once the resources are found, resources are allocated based on game theory. Since resource providers (RPs) and users share resources similar to economic models where collaborative usage of resources is needed, game theory provides an appropriate model. An auction is a marketing mechanism we use for resource allocation in the distributed cloud. In a regular auction mechanism the bidders bid for a resource. In our proposed model RP's bid to sell or provide resources. In our proposed auction an auctioneer, the user, determines the resource allocation. The bidders who are RP's provide their resource. The players are both bidders and auctioneers. In this game there is no centralized computation, which fits in well with our proposed distributed cloud. Since the distributed cloud is based on resources supplied by users, it is free.

Both types of players have information of resource description and include CPU, memory, participation factor (which is defined later in the chapter), latency, and throughput. The auctioneer knows about the latency and throughput from measurements made by the RP's. RPs submits their contributions to the user. Contributions specify the amount of resources they are willing to provide along with the incentives they need for providing them. Incentives specify the resources the provider expects from the requesting user at some point in time. Each bid can be characterized as 4-tuple $b_i = <C_i, M_i, \delta, *, \alpha >$, where C_i is CPU available, M_i is memory available, δ is participation factor, * defines user specification which is latency or throughput and α determines the incentives required.

A question that arises is "can't a bidder submit a false bid?" For example a free rider might generate a bid with a high δ (participation factor) value. In this paper we assume that the underlying operating systems software generates the bid. This we assume is a secure piece of

code that cannot be tampered with. Based on communications (ping for example), the * values can be generated.

Based on the bids, the user calculates the utility function. Using this utility function, resources are allocated to the user. The Utility function (eq. 5.1 below) determines the utility which is the value of resources assigned to the user. Since our model of distributed cloud is free of cost, we have a *participation factor* (δ) assigned to each user, and incentives to encourage users to provide resources. The participation factor aims to avoid selfish behavior of users. It shows how much resources a user has contributed to the system. Participation factor is included in the system and user cannot cheat about the participation factor values.

In this type of auction the bidder with the highest contribution is not the winner. User *i* chooses the resources based on the bids and allocation of these resources to user is done rationally. Each user requesting a resource would not receive ideal resources if his participation factor is low. Users need to provide resources to raise their participation factor. Participation factor is based on the number of requests a user has satisfied. If '*N*' is the total capacity of resources provided by RP until a time *t* and '*Z*' is total capacity of resources available with the RP then participation factor is:

$$\delta = N/Z$$

The participation factor is always between 0 and 1. During the initial stages or if the user's participation factor is low he might receive resources whose utility value is small. This ensures that once the system is close to stability or users who are not free riders will most probably receive resources with maximum utility. Typically each provider submits contributions to maximize their participation factor. In the distributed cloud users will be requesting multiple resources for computation, so we will be using a simultaneous auction and there will be multiple winners.

A bidding profile is a vector of player's bids $b = b_1,b_2...b_i$. The bidding profile of user i is represented using b_i and the bidding profile of user i's opponent is defined as $b_{-i} = b_1...b_{i-1},b_{i+1}$.. User i chooses a resource based on the utility, where utility is defined as

$$Ui(b_i; b_{-i}, \delta) = (latency, \delta, CPU, MEM) - Cost$$

$$Ui(b_i; b_{-i}, \delta) = \left(\frac{Wt(CPU + MEM) * \delta_{user}}{latency}\right) - Cost_i \rightarrow (5.1)$$

Where W_t is the weight assigned for CPU and memory, δ_{user} is participation factor of user requesting resources and $Cost_i$ is the value of incentive α that is asked by the RP. Once the user receives all the contributions, the user will calculate utility values for all the bids using eqn. (1). If the user needs 'N' resources, the user will select 'N' winners whose utility is maximum. The cost $Cost_i$ for a particular RP is based on the amount of resources he has in his possession. $Cost_i$ is modeled in such a way that it is low for RPs with lots of resources to provide and vice versa. After the winners are announced, the user will have to pay incentives 'α' to the respective winners. Incentive for RP_i is given as

$$\alpha_i = \begin{cases} Cost_i & i \text{ is winner} \\ 0, & i \text{ did'nt win} \end{cases}$$

We describe this game by considering that if a RP had submitted a contribution to one user he won't submit anymore contributions until the results are declared. After the bidding is completed, utility is calculated based on user requirements, latency or throughput based on type of computing and participation factor. We calculate participation factor for a provider based not only on the resources provided by a particular provider but also on the quality of service. In a homogeneous system we assume only one user is requesting a resource at a particular point of time. Then the desirable outcome would be a bidding profile for which the utility would be maximized and is called the Nash Equilibrium for the game, from which no user can unilaterally deviate i.e.

$$Ui(b_i^*; b_{-i}^*, \delta) \geq Ui(b_i; b_{-i}^*, \delta), \forall i \in N$$

The users who provide their resources will be receiving some kind of incentives such as storage or resource‘s for computing. The incentives would be based on $Cost_i$. The whole process can be summarized beginning with users requesting a set of resources. The request is send out to all the nodes in the routing table and a set of resources which can satisfy the user would be discovered. Then we use game theory to select the resources such that it will best match user specifications and incentives would be awarded to the respective RP. All the users discovered will send the bids $b_i = <C_i, M_i, \delta, *, \alpha>$ to the user. The user then evaluates the bid using game theory and uses the resultant resources and pays the incentives to the winners. The whole process of resource discovery and playing the game would be automated. This way no user can modify their participation *factor* (δ) or none of the RPs can cheat about their availability and bids.

We claim that our proposed approach achieves stability, that is, allocation is fair. Users who provide resources for other users are more likely to get the resources they require.

Corollary 1: In a system where all resource providers have the same participation factor, load balancing is achieved, that is, stability is achieved.

Proof: Utility is based on δ and cost. δ is the same (see corollary 2 below). The only variable is cost. There are two possible cases:

- Cost that is higher than average. For a particular RP, a high cost, results in fewer resources being allocated. This reducesδ. This will drive down the cost, resulting in more resources being allocated and δ increasing.
- Cost that is lower than average. For a particular RP, a low cost results in more resources being allocated. This increasesδ. This will increase the cost, resulting in fewer resources being allocated and δ decreasing.

Corollary 2: In the absence of free riders, all resource providers will converge to a similar participation factor δ.

Proof: the Nash equilibrium states that the utility would be maximized for the game from which no user can unilaterally deviate. Utility is based on the participation factor. Therefore the RPs will have similar δs and in the limit will be equal.

This leads to the following important conclusion

Theorem 1: The proposed auction game achieves load balancing and Nash equilibrium in terms of resource allocation, that is, the system achieves stability.

Proof: Follows from the above two corollaries.

5.4 EXPERIMENTAL ANALYSIS

We simulated the simultaneous auction model using the king data set [17] with 100 nodes. There is no cost involved in searches for resource allocation. In other words, in the auction mechanism can be scaled to a large number of nodes and therefore simulating with 100 nodes is sufficient. The simulation results are therefore valid to a larger number of nodes. The king data set gives real time latencies between a set of DNS servers. This was used to evaluate Vivaldi [30] a decentralized network coordinate system. We implemented our auction game on a distributed cloud model. We simulated the distributed cloud by making changes to Kadelmia by adding our resource discovery model into it. We performed experiments using different participation factor values. We can see that only the nodes which satisfy user requirements would be selected for auction. Once *bids* are submitted, the user will select the nodes for which he can maximize his

utility. Fig.17 shows how cost or incentive values varies as difference between required and available resources increase.

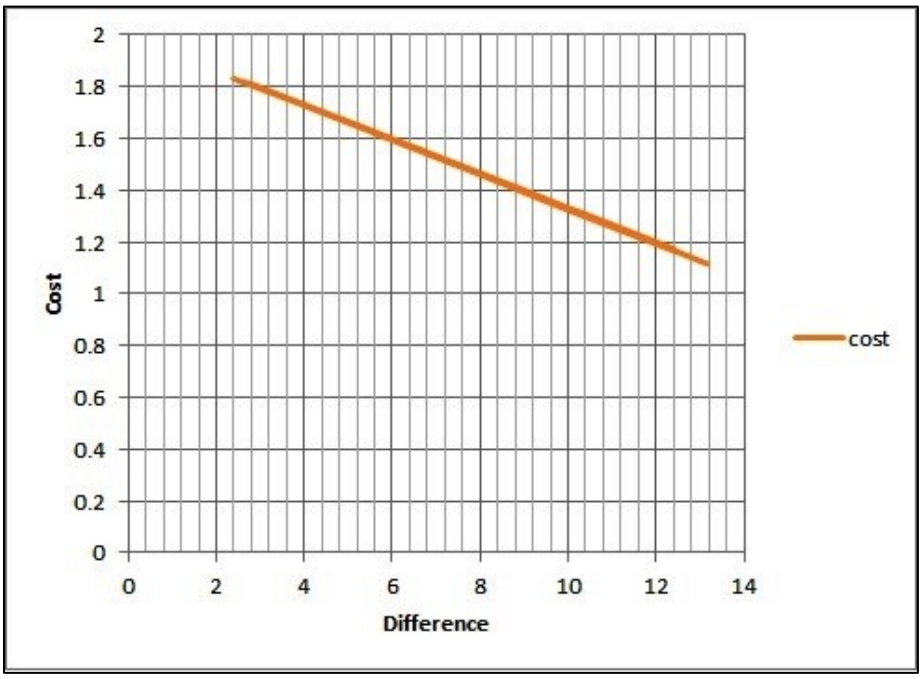

Figure 17 Cost (Incentives) dependency

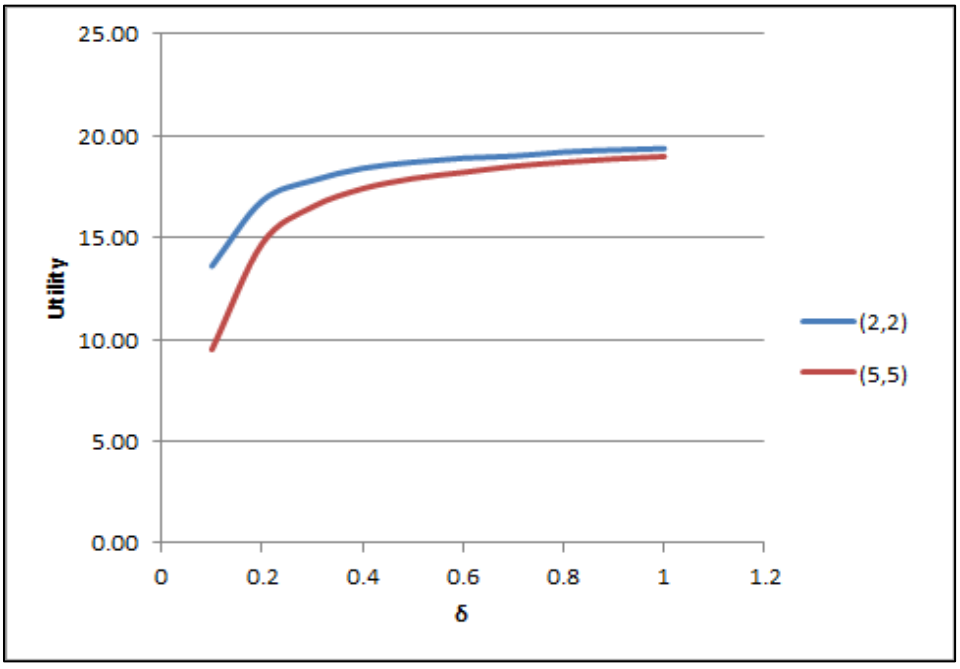

Figure 18 Utility with varying participation factors

We can see that as the difference between the available and required resources increases, cost decreases. This is similar to a buisness model where as supply incresases cost decreases. We ran simulations to see how the participation factor will affect the utility of a user. We measured utility for varying CPU and memory requested. In Fig 18 (2,2) means user requested a CPU 2GHz and memory 2GB and we can see that utility for nodes with lower participation factor is low for the same amount of resources requested. This also ensures that free riders will get resources only at a higher cost, than honest users.

From Fig.19 we see that with 10% free riders in the system the utility is a bit higher for free riders than with 20% and 50% free riders. These results show that with more free riders the utility of free riders is decreased. But for users who are not free riders the utility is almost constant as the participation factor is increased. Utility values with 20% and 50% free riders in the system are almost the same i.e. the utility values are low for free riders and constant for other users. Overall, utility for free riders is comparatively lower than regular users.

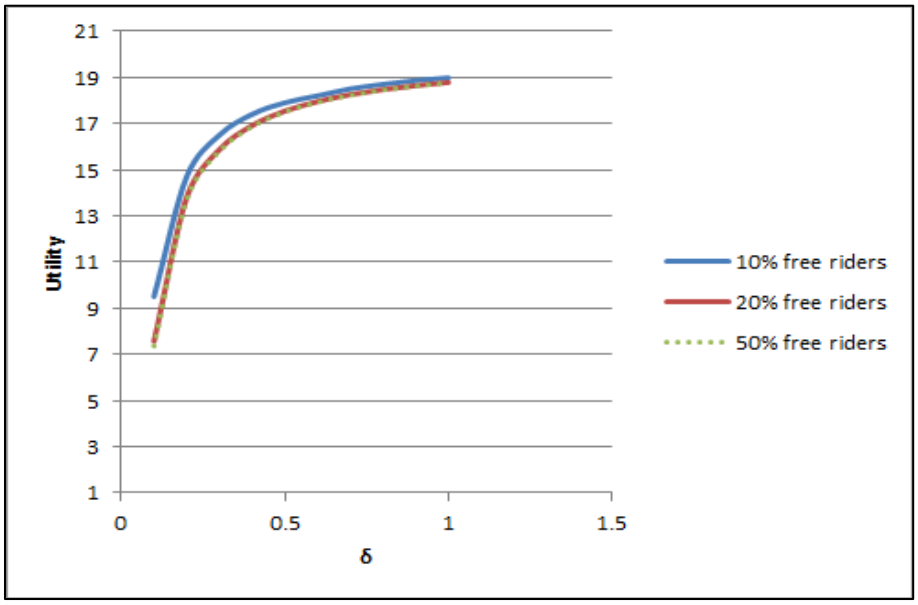

Figure 19 Percentage of resources utilized compared to participation factor.

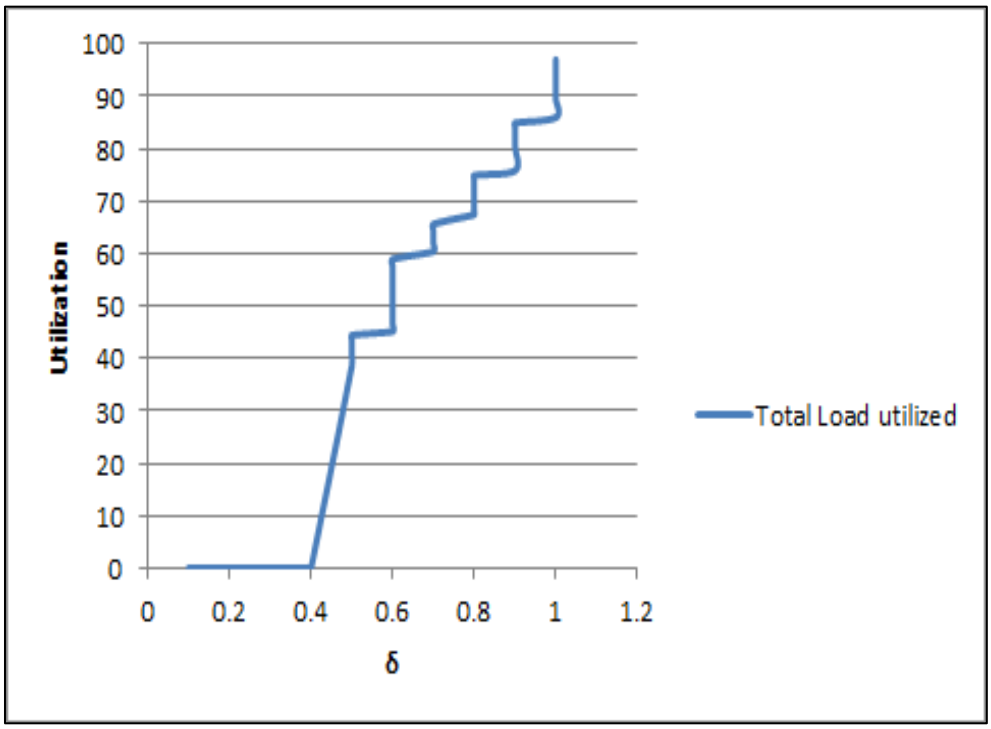

Figure 20 Percentage of resources utilized compared to participation factor over time.

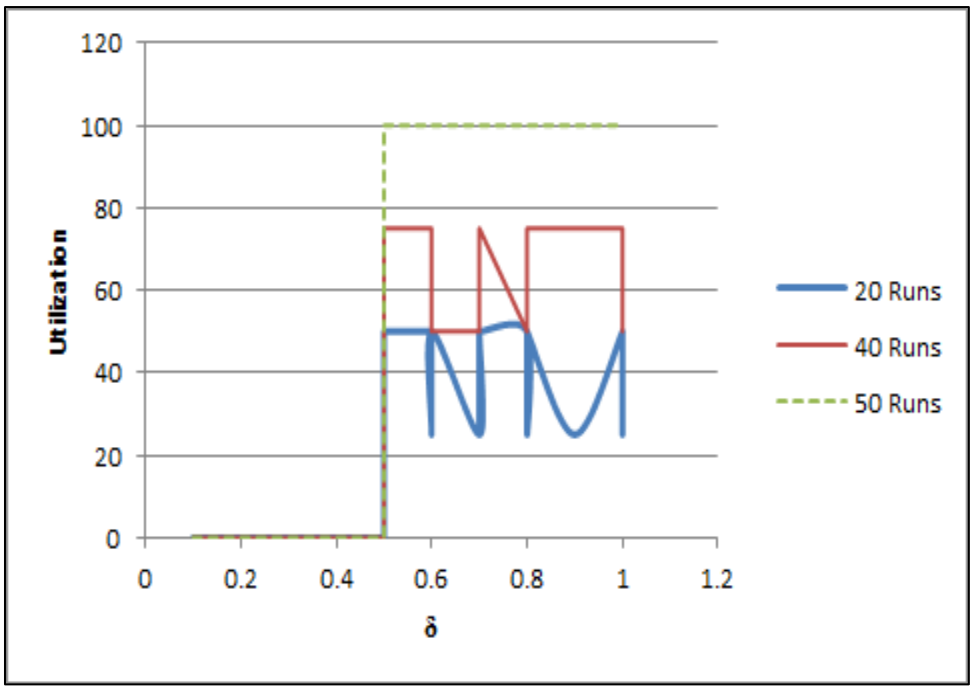

Figure 21 Utility vs Participation factor with free riders included

We calculated the percentage of resources utilized and Fig.20 shows that total resource utilization is more for resources with higher participation factor and we observe that users providing fewer resources will eventually provide more resources over time. This leads to system stability. Fig.21 shows that the percentage of resources utilized increases over time and after some time all the resources are utilized to the full. This shows that our scheme balances the load over time. We ran experiments with constant resources provided by everyone. We see that utilization is constant for all resources i.e. the load is balanced equally among all participants or users.

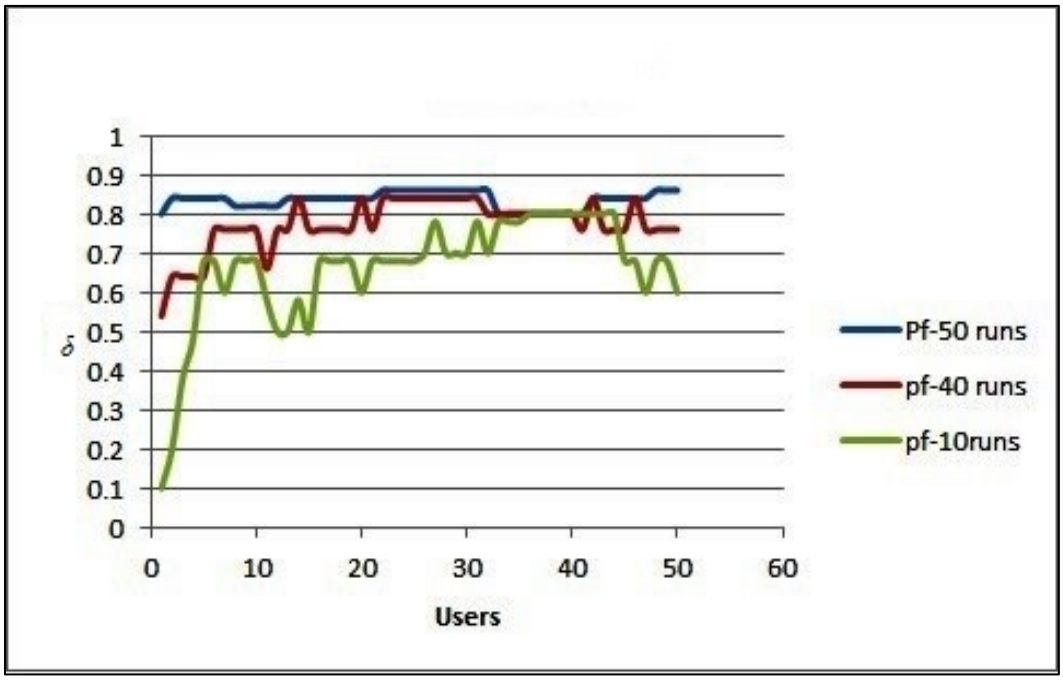

Figure 22 Participation factor for users over time.

We ran simulations with 50 users having varying participation factor values i.e. each user is assigned a participation factor value between 0 and 1. We assumed there were no free riders in this case. We see that the difference in participation factor values of all the resource providers are reduced over time. From Fig.22 we can see that initially after a few runs participation is widely distributed. But as the game proceeds we can see that all the resource providers end up with an almost similar participation factor.

5.5 CONCLUSIONS

We described a game theoretical model for resource allocation. We see distributed cloud as an emerging model in the next few years with a lot of research remaining to be done. We propose a game theoretic model for resource allocation. Results show that a game theoretical approach stabilizes the system and penalizes free riders using the participation factor for allocation of resources.

Our future goals include simulating a large distributed cloud for detailed validation. We plan to extend the game theory concepts presented here for a homogeneous system to a heterogeneous system, in other words, multiple users requesting the same resource or multiple users requesting different resources at the same time. We are working on using the Bertrand pricing mechanism to define the optimal price so that the system would yield maximum utility. Issues we need to look into would be how to allocate the resources when tasks are dependent on each other and need cooperation between the resources.

CHAPTER 6

BERTRAND GAME MODEL FOR ANALYSISNG RESOURCE ALLOCATION MECHANISM

6.1 INTRODUCTION

The distributed cloud is a completely decentralized system. There are no central agents working to allocate the resources. So we need a completely decentralized approach to allocate resources. A user would be a part of the distributed cloud. A user will get access to resources in the distributed cloud in a P2P fashion. The distributed cloud brings the benefits of both P2P networks and cloud computing systems. A user requiring resources will find the nodes using the P2P model and then uses the auction mechanism to select appropriate resources.

From an economics point of view pricing plays an important role in sharing/trading resources. We have used an incentive mechanism instead of price such that both user and provider benefits from the system. In our model demand is dependent not only on price but also the availability of resources. Since in a peer to peer model the main problem is free riding, we modeled the distributed cloud in such a way that free riders are unlikely to get resources. Free riders are the users who tend to use the resources and provide none. In our model price depends on the availability of resources, i.e. resource provider with more resources will provide resources for less price and demand for such providers increase.

Bertrand model is used to analyze the behavior of nodes and used to study interactions between nodes which can be either users or providers. Based on the Bertrand model we can make sure the system applies the appropriate auction mechanism on nodes. We use the Auction model described in chapter 5 for dynamic resource allocation in a distributed cloud. In this environment a user will request for resources and multiple service providers will compete to provide their resources. We formulate this resource allocation as an oligopoly market.

The main objective of this game would be to stabilize the system i.e. to make sure free riders are eliminated and all users provide resources. The decision of one resource provider will affect the outcome of others. Users decide to choose the best resource providers to maximize their utility. The best set of resources a user can choose to maximize his profit is called a best response. In our scenario resource providers competing can be assumed as firms in an oligopoly market. These resource providers compete with each other to provide resources. Resource providers and users are independent; therefore we can use the best response to find the equilibrium from both the user and resource provider point of view.

6.2 PROBLEM STATEMENT

In a distributed cloud users are completely distributed with no centralized authority to allocate resource, manage security etc. Multiple resources are available to be used. Users request and also provide resources for other users. All resources are identified based on the attributes that describe them. Some resources being provided or requested are single attribute resources e.g. CPU only, some have multiple attributes e.g. CPU, cores and memory based on the requirements of the user.

Users with tasks to execute on the distributed cloud will request for resources. Multiple service providers will compete to provide their resources which satisfy users requests. Since there are a

large number of users and their resources which are homogeneous we formulate this as an oligopoly market.

The main problems that need to be resolved are, firstly, how the user can choose resources such that user requirements are satisfied. Secondly, how can we allocate resources such that the utility of users are maximized. Users who don't provide any resources and just use the system to get resources for themselves should be punished. This is achieved by exacting a high cost for using the system i.e. not allowing free riders. A free rider is someone who uses resources but does not provide any of his own for others. The main problem would be to make sure the system doesn't allow free riders or punish them if there are any.

The main objective if the model is to attain system stability where all users participate in the system such that there is a balance between providing resources and using resources. To verify if users are providing resources or not we have included a participation factor associated with each user which measures a user's contribution to the system.

6.3RELATED WORK

Distributed cloud computing is an emerging model of cloud computing. Distributed cloud computing makes use of resources provided by users in a P2P fashion. A Distributed cloud can be implemented with no cost needed for maintaining a data center. Resources that are lying idle can be used to build a distributed cloud infrastructure. New users can join the system and users can move in and out of system similar to a P2P mechanism. The system would be scalable and can provide access to multiple resources without any single point of failure and central management.

Cloud resource pricing has been a topic for research recently. In [31], the authors proposed usage of game theory in a P2P cloud model to solve the free rider problem. The authors however divide the cloud into clusters supervising nodes maintain information. This forms a hybrid cloud which uses a centralized architecture among a cluster. Our model of distributed cloud is completely decentralized and there are no centralized points. In [33], the authors looked into cloud resource pricing competition among cloud providers. In [32], the authors used cournot equilibrium for strategic pricing of cloud resources where quantity is changed. In the cloud computing model increasing the quantity of resources without knowing the requirement means increase in costs for the provider to maintain them. In our model we show that by modeling the distributed cloud as an oligopoly market and using Bertrand competition, users compete to maximize their profits while solving the free riding problem.

6.4 RESOURCE ALLOCATION BASED ON BERTRAND GAME MODEL

6.4.1 NOTATIONS

- System - the distributed cloud
- ***N*** users
- Each user provides different number and types of resources
- Total number of resources provided by all users is ***r*** resources where ***r >=N***
- Participation factor $\boldsymbol{P_f}$ - ratio of resources provider has provided to other users to the total capacity of resources he has available.
- Auction factor $\boldsymbol{A_f}$: A_f is the threshold level of participation factor which is defined by the system. Auction factor determines which auction mechanism should be

used based on the users participation factor. The two auction mechanisms used are sealed first bid auction mentioned in chapter 5 and reverse auction or sealed last bid auction. In sealed last bid auction the lowest bid is considered winner. So free riders will get the resources with lowest bids i.e. for higher costs.

- Demand of user ***i, D_i*** **:** demand defines the need for his resources available.
- Profit π_{user} - Amount of utility gained by the user
- Profit π_{RP} - Total increase in participation factor and incentives obtained.
- ***u_r:*** Number/amount of resources requested by a user
- A user ***u*** has his own set of resources which he can provide to others ***w_p***
- Different resource providers charge a different price ***p*** for each resource.

Currently price is value of incentives

- Utility represents the motivation of players. In this game higher utility implies the strategy is most preferred one and vice versa.

6.4.2 RESOURCE PROVISIONING MODEL

We use the Bertrand model [22] to analyze the market and apply the auction mechanism. We consider a scenario in which there are "N" users and each of them provides "r" resources, where N <= r. At any given point all users might not provide resources. All the resources provided by a RP might be being used and they won't have anything to provide at time'*t*'. When a user requests a set of resources "u_r" the only thing that varies among the different resource providers is the price "*p*". All the different resource provider nodes vary in their capability and some of them might be closer to the requestor node so latency will be less. Here we try to analyze from both the resource provider's point of view and the user point of view.

In this model (Bertrand) RPs will have a fixed price which depends on the number of resources they are providing and the number of resources requested. In our model, the price quoted by a RP depends on the difference (r-u_r). We formulated our model such that cost of resource requested "u_r" decreases if (r-u_r) increases.

$$cost_{u_r} \propto \frac{1}{r - u_r}$$

This way we can make sure that RP's provide more resources to users and RP's who are not providing much resources will change their quantity to increase profits. Since cost decreases as the number of resources provided increases, the demand would be more for users who provide a lot of resources. RP's will try to maximize their total profit "π" and at the same time they will try to increase their participation factor P_f. The utility of the user depends on his participation factor, i.e. if the user's participation factor is low or below a threshold he will be awarded resources with low utility. For the user, utility is the total value of goods they received subtracted from the total incentives they paid. The user's strategy depends on his participation factor. If user's participation factor is less than a threshold value the system uses a reverse auction mechanism to make sure users are actively providing resources and not just using the resources.

6.4.3 GOALS

Different resource providers charge a different price *"p"* for each resource. Other factors such as latency, throughput may vary. The price quoted or amount of incentives required by a user *w* who provide resources, depends on availability. The scarcer a resource the provider has, the higher the price. Therefore price is a function of $1/(w_p\text{-}u_r)$ where *w* and *u* are different users. The goal is to

make sure that resource providers provide more resources to users. Resources that are cheap will be used faster and the participation factor of users providing resources will increase.

Resource providers:

- Aim to maximize total profit π.
- Aim to increase their participation factor P_f

Users:

- If a user's participation factor P_f is higher than the threshold or Auction factor A_f , the system will aim to maximize the user's utility
    - A regular auction mechanism is used
- If user's participation factor P_f is low or below threshold or A_f, the system will award him resources at a higher price resulting in lower utility
    - A reverse auction mechanism is used
    - Makes sure users are actively providing resources and not just using resources.

6.5 BERTRAND GAME MODEL

Definition: *Stable System: The system is said to be stable if all the users in the system have an almost equal participation factor i.e.*

$$P_{f1} \approx P_{f2} \approx P_{f3} \approx P_{f4} \approx P_{f5} \approx \ldots \approx P_{fN} \quad (6.1)$$

We define *Demand* to be a function *f* of prices provided by different providers to that provided by w_p. In other words, the lower the price, the higher the demand.

$$Demand\ D_i = 1/f(p_{wp}, p_{-wp}) \quad (6.2)$$

p_{-wp} gives the price of other resources available that are not provided currently.

The auction mechanism and all the terms are described in chapter 5. All resource providers will *bid* to provide resources. Price required or value of incentives is based on the availability of resources being provided and latency. Latency can be obtained when a user receives the bid using ping. In our initial model, although each resource is selected from only one provider, all the resources requested will be of the same type. Hence if *n* resources are need, *n* providers will be selected. All the resource providers with a resource will submit bids.

Let $P_i = \{p^1_i, p_i^2, ..., p_i^n\}$ be the bids submitted for a resource *i* by *n* providers where p^j_i is the price of resource *i* demanded by provider *j*.

If $P_f < A_f$ for a user, then using the reverse auction mechanism the successful bid will be p^j_i such that $p^j_i > p^k_i$ where $j \neq k$. In reverse auction mechanism the bid with least utility is assigned to the user i.e. user will get the resources with highest price.

For *m* resources, the winning bids $W_r(P)$ for user *r* requesting *m* resources will be the collection of each of the successful bids for the *m* resources. That is,

$$W_r(P) = \{p^a_1, p^b_2, ..., p^g_m\} \text{ such that } \forall i \leq m, \forall j > m, p^x_i \geq p^y_j \text{ where } x \neq y \quad (6.3)$$

If $P_f > A_f$ for a user, then the auction mechanism for the successful bid will be p^j_i such that $p^j_i < p^k_i$ where $j \neq k$

For *m* resources, the winning bid for user *r* requesting *m* resources will be the collection of each of the successful bids for the *m* resources. That is,

$$W_r(P) = \{p^a_1, p^b_2, ..., p^g_m\} \text{ such that } \forall i \leq m, \forall j > m, p^x_i \leq p^y_j \text{ where } x \neq y \quad (6.4)$$

Profit for a resource provider would be calculated as

π_{wp} = Total price received + increase in P_f.

Let C_i be the marginal cost of resource for resource provider i. Marginal cost is the unit cost for a resource.

Profit for a user n requesting i resources would be calculated as:

$$\pi_i(u_r) = \sum_{i=1}^{i} D_i/(p_i - C_i) \text{ if } P_f > A_f \quad (6.5)$$

$$\pi_i(u_r) = \sum_{i=n-i+1}^{n} D_i/(p_i - C_i) \text{ if } P_f < A_f \quad (6.6)$$

Definition (Nash Equilibrium): A price vector P_i^* is a Nash equilibrium for user i given P^*_{-i} such that user's profit is maximized by vector P_i^* i.e.

$$P_i^* = \text{argMax } \pi_i(p_1,p_2,p_3...p_n : P_{-i}^*) \text{ for all i=1,2,3…}N \quad (6.7)$$

The Nash equilibrium of the game is solution such that no player can increase his payoff unilaterally choosing another strategy. The Nash equilibrium can be obtained using the best response which is an optimal strategy. The best set of resources a user can choose to maximize his profit is called a best response.

$$BR(P_i^*) = argMaxP_i(P_i^*: P\text{-}_i^*) \text{ for all } I \quad (6.8)$$

Proposition: If $P_f < A_f$ there is no price vector P_i^* such that the user attains Nash Equilibrium.

Proof: From Eq. 5 we see that $\pi_i(p_1,p_2,p_3...pn) = \sum_{i=n-i+1}^{n} D_i/(p_i - C_i)$ if $P_f < A_f$

We know that price vectors are arranged in descending order such that $p_1 < p_2 < p_3 < ...p_n$ and number of resources requested $i \not\equiv n$, so there exist a set of price vectors P_i^* such that

$$\prod_i(P_i^*) > \sum_{i=n-i+1}^{n} D_i/(p_i - C_i)$$

6.5.1 ANALYSIS

In this Bertrand game model we will use the auction mechanism mentioned in chapter 5. If users participation factor is greater than the auction factor, than the regular auction mechanism mentioned in chapter 5 is used. If the participation factor of the user is less than the auction factor, a reverse auction mechanism is used. In reverse auction mechanism a free rider requesting a set of resources will get access to resources provided by resource providers which will give them the lowest possible utility, i.e. the free riders are punished by letting them have access to resources at a high price.

Therefore if a user's participation factor is below the auction factor, his profit would be minimized and therefore he doesn't attain Nash equilibrium. All users therefore try to increase their participation factor by providing more resources and thereby the system attains stability and avoids the free riding problem. Experimental analysis is similar to the results in chapter 5 where stability is achieved over the time by increasing the participation factor.

6.6 CONCLUSION

The Distributed cloud model is built on top of resources provided by individual users, firms etc. A Game theoretical model is proposed to solve the free riding problem. The Distributed cloud is modeled as an oligopoly market where users compete based on price. Participation factor is taken into account to decide the price of resources and any free riders requesting a resource would get them, but at a high cost. Users are motivated to provide resources for self-benefit and by avoiding

penalties. Since there is no maintenance cost, users with idle resources are encouraged to provide resources for their own self-interest.

CHAPTER 7

NETWORK MEASUREMENTS IN A DISTRIBUTED SYSTEM

7.1 INTRODUCTION

Treeple [28] is a centralized network coordinate. Trusted nodes, called vantage points, perform network-based path measurements to map the path from the vantage point to every node in the network. Vantage points build a tree of intermediate nodes and the per-hop network latencies. Every leaf of the trees built is a node. The network coordinate of a node is the path from the vantage point. To estimate the network distance between two nodes, the least common ancestor for the two trees or paths found and the estimated network distance is the sum of all the per-hop latencies. Although this is not the actual path taken by a packet between the two nodes, this estimate provides a bound on the error. With real Internet-based experiments, Treeple was shown to be accurate, efficient, and provably secure.

Treeple can be used for secure network measurements in a distributed system which in turn can be useful identifying the closest nodes in a distributed cloud. Latency estimation in distributed cloud is necessary and the network measurement techniques can be used to find the closest nodes efficiently and also can reduce communication overheads. The results from this work will be used in the resource allocation game as latency is a factor in resource allocation. In the distributed cloud latency can be obtained from the PING or we can use this mechanism. We can use this mechanism to show the user a list of providers based on the latency.

7.2 IMPLEMENTATION

We implemented a plugin for Vuze [16] and worked closely with the Vuze developers to make it available on the Vuze website. Vuze is a popular peer-to-peer file-sharing client, based on the BitTorrent architecture. The plugin has negligible overhead in processing and network traffic. We set up a server that collected the IP addresses of all the PlanetLab nodes. When the Vuze plugin first starts, it registers with the server which records the new IP address. The server then sends the list of current IP addresses to the plugin. The plugin then performs a ping network measurement to all the IP addresses and sends the latency information back to the server. The ping is performed every time Vuze starts and all the pings are spread over time and to random IP addresses each time.

We used planet lab nodes as these nodes are located all across the world. This closely resembles the distributed cloud where users can be any where in the world. We then deployed traceroute and ping scripts on PlanetLab that will periodically perform a traceroute and ping measurement to other PlanetLab nodes and the Vuze nodes (that have the plugin installed). This experiment was performed in July 2013. The Treeple dataset contains per-hop latency and the IP address of each router along the path from a node in PlanetLab to either another node in PlanetLab or a node in Vuze. Figure 1 shows the accuracy obtained by this new Treeple dataset.

Fig. 23 shows the CDF of relative errors obtained by using different k vantage points. The median relative error is about 0:15, which is comparable to that obtained by Vivaldi and the original Treeple paper.

Fig. 24 shows the median relative error over 30 days. The figure shows that Treeple network co-ordinates are very stable over time. Again, this reflects the result obtained previously. This leads us to conclude that the plugin and experiment performed work correctly.

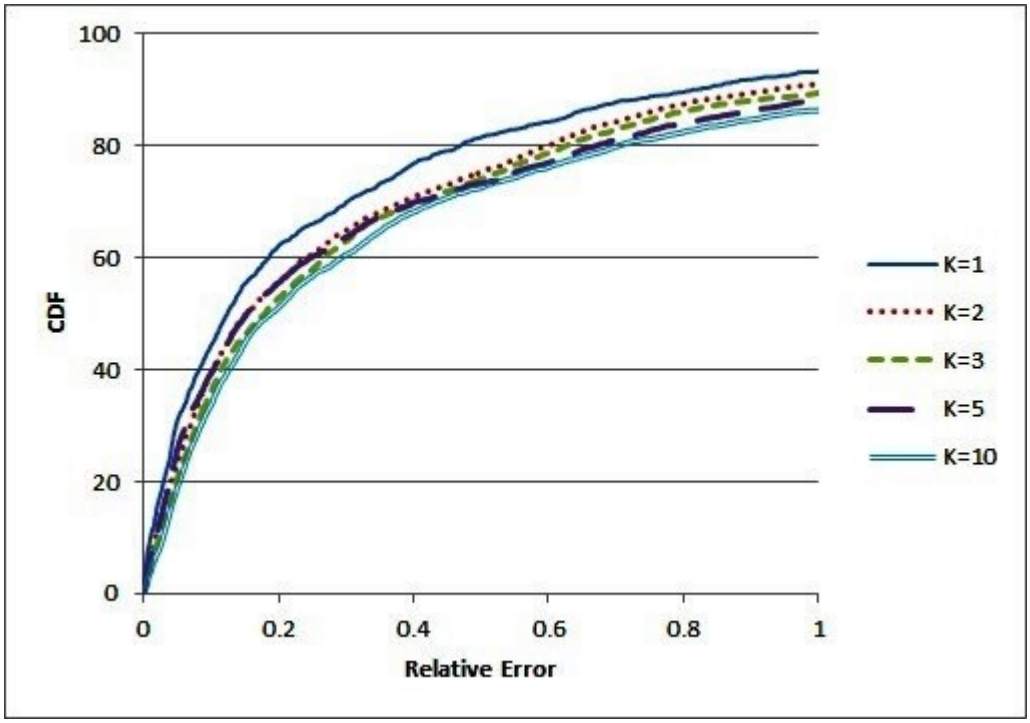

Figure 23: The CDF of relative error using different vantage points (*k*)

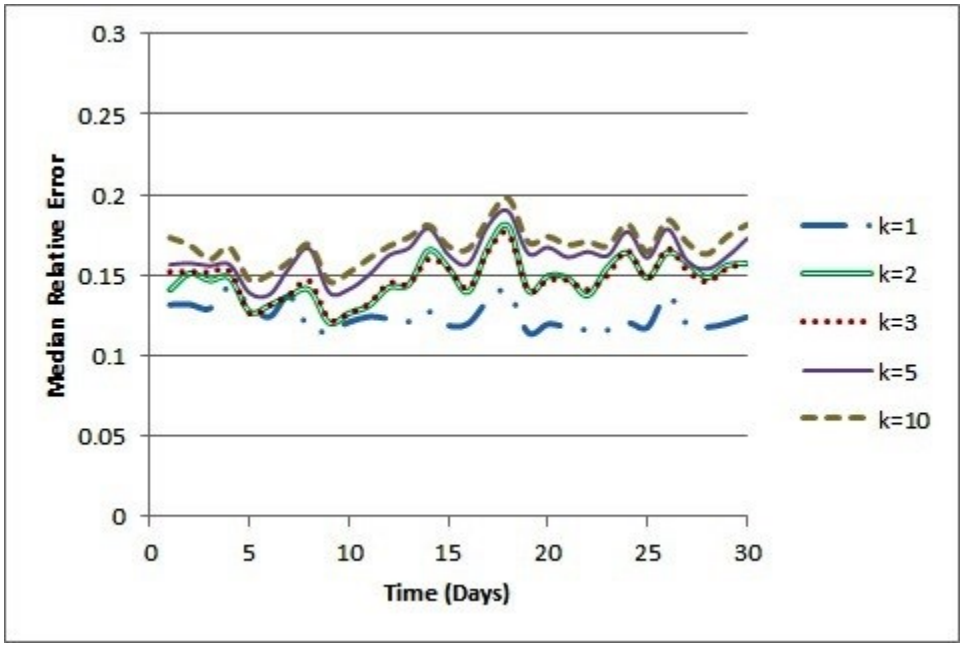

Figure 24: the stability of median relative error for different vantage points (*k*)

7.3 CONCLUSION

Treeple is a decentralized network coordinate system and can be used as an efficient and secure network measurement mechanism in distributed computing environments. We used it to make the network measurements in a distributed P2P systems and the data obtained was very useful. It was used for accurate geolocation based on latency and it can also be used in place of King data set. This mechanism can be used in distributed cloud computing environment for efficient latency estimation and can also be used to make decisions about routing so as to decrease communication overheads.

CHAPTER 8

USING CLOUD AND BIOMETRICS FOR SECURE MOBILE TRANSACTIONS

Distributed cloud computing can be of different types, such as public, private and hybrid. A distributed private cloud can be very useful as it can avoid huge data centers. Mobile payments are used to sell and purchase goods and services. The increase in the usage of mobile devices has increased the scope for using these devices for payments and other day to day activities. Using handheld devices for payments brings up lot of security challenges for mobile applications.

Mobile payments have become popular with the extensive use of user friendly hand-held mobile devices and easily available wireless networks [34]. Mobile payments are rapidly growing and are supported by many leading IT industries such as PayPal, Google, Apple etc. Although mobile devices are widely used for different purposes (e.g., mobile payments, mobile banking etc.), its low computation power cannot provide a highly secure environment. To enhance security and improve performance, cloud computing [1, 2, 3] can be used along with mobile applications.

Mobile applications for payment systems must be very secure to perform transactions and at the same time be efficient. Current mobile payment systems use a merchant agent (MA) who must be physically present in order to make a payment.

In this chapter we look at an application of the distributed cloud, in particular the use of the distributed cloud for developing a mobile payment scheme which can be used by all the markets. The distributed cloud can be used here for parallel computations to serve many users from the same area based on latency. The distributed cloud used here is private to the market i.e. only the

users of the market can have access to these distributed clouds. Since online shopping at a particular store requires high level of interactivity and low latency, the distributed cloud would be ideal for this scenario.

In order to make the shopping experience even more comfortable we propose a secure Global Market mechanism for making in store purchases and payment more secure without need of a cashier or a MA. We use a Biometric mechanism as it is more secure, safe and easy to use. We also show how it is related to the distributed cloud.

In this chapter, we propose a novel secure and privacy preserving biometric authentication scheme for mobile transactions and architecture using the distributed cloud. All the participating Stores will save their in store products information in the distributed cloud. Users scan the resources they want to buy using the mobile application and payment is made using the proposed secure scheme.

8.1 INTRODUCTION

All cloud providers provide services defined by their service model's Infrastructure as a Service, Platform as a Service, or Software as a Service. Using the distributed cloud we can have the entire biometric infrastructure from processing the biometric template to storing the templates done in parallel without a single point of failure. Characteristics of distributed cloud such as elasticity, broad network access makes the use of cloud for biometric services more scalable and cost effective.

With the advantages of the distributed cloud we also need to make sure that we preserve privacy and provide security for users. Performing user authentication is one of the important steps to

prevent unauthorized users. There are different authentication mechanisms [36] such as password based, token based and biometric based. Biometric authentication refers to identification of a user based on their characteristics or behavior. There are different biometric characteristics such as iris, face, fingerprint, voice recognitions that are being used. The cloud's advantages such as scalability and high computing power make biometric mechanisms more feasible and enhance the security needed for highly secure applications such as banking. Beside the advantages of biometric mechanisms there are increasing high concerns associated with privacy issues [37] as biometric information is highly sensitive. So we need to have appropriate secure privacy preserving mechanisms when using biometric features.

Different mobile payment applications use either a password based mechanism or token based mechanism to secure access for these applications. Any unauthorized user with access to these mobile devices and passwords can exploit these payment mechanisms. So we used a secure and privacy preserving biometric authentication mechanism for access to these mobile payment applications. Using biometrics for authentication involves the use of high computation power and can be time consuming. We have designed a light weight biometric authentication mechanism using the distributed cloud. Our proposed application does not need any merchant agent for a secure transaction and moreover our application ensures user authentication when any user tries to log in to our application. For privacy preserving log in we use biometrics since it is far more secure than traditional log in mechanisms. For secure transactions our application takes the entire responsibility for user authentication, merchant authentication, and payment.

8.2 RELATED WORK

Mobile devices face various challenges when it comes to M-commerce applications because of low bandwidth, low security, low computation power and highly complex device configurations.

All the mobile payments architectures revolve under how to perform transactions securely. Some of the technologies used for mobile commerce are NFC a short range communication protocol [42], Mobile wallet [ref] etc.

Mobile wallet [63] is an application residing on the mobile device which holds information regarding credit cards and can be used for payments also. Authenticating these applications using existing mechanisms is not sufficient as these devices are not protected against theft and password compromise. We need a more secure mechanism to authenticate mobile payment applications which resists theft of devices and other known attacks where user credentials are compromised.

There are many schemes for mobile payments which try to solve issues such as efficiency, security or privacy. A secure account-based payment protocol [38] for wireless networks makes use of symmetric key properties which reduces computations. The chapter also mentions how the protocol can be used to increase transaction security when compared to the SET [39] based approach. Use of a highly secure, multi-factor authentication scheme can be expensive in mobile devices. In [40], the authors proposed a multi-factor security protocol for wireless payments for J2MEE based clients and J2EE based servers. Author utilizes TIC full form code and SMS to authenticate the transactions. Author assumes the mobile device which has the TIC codes is secure. The only security for applications provided is username and password. S. Karnuskos et al [41], presented a secure mobile payment service as a part of the SEMPOS project to handle security and privacy of users. The SEMPSO project uses mobile network operators and data centers to be part of the payment mechanism. In [42] the authors utilized a NFC based solution for managing payments. Since there are many parties involved in the transactions, we utilized a cloud based approach. Secure elements are downloaded to the mobile device when a user wants to make payments and once the user authorizes the payment, the transaction is done and the secure element is deleted from the device for security purposes. Existing methods for mobile

payments are device specific as in [40, 41]. In [40, 41] the payment technology can be applied only for devices with J2MEE and NFC enabled devices. Existing payment techniques such as NFC based solutions or google wallet are similar to traditional credit cards and are provided for ease of customer use, but are still prone to thefts and various attacks. Our payment protocol uses the RSA based scheme to verify use and payment by the bank and we use a strong biometric authentication scheme to secure the mobile payment application.

Biometric authentication mechanisms have gained popularity as they are more secure and offer non repudiation. Biometric authentication mechanisms [43] have been proposed to secure mobile payments. The public key of the bank is used to encrypt credit card information and hashes of the finger print to send to the bank. The bank server has information regarding all users and their finger print information. This method doesn't protect the privacy of user. In [44], the authors specified an identity based encryption (IBE) and biometric scheme for secure data access in the cloud. The biometric scheme was used here to authenticate the user and IBE scheme was used to encrypt and decrypt the data. Ross and Othman [45] used visual cryptography to protect the privacy of user. They decompose the user's biometric template into two noises like images using visual cryptography and store them in two distinct databases. During authentication these images are laid together to create a temporary biometric template for verification. The main advantage of this mechanism is that the biometric information can never be exposed to the attacker using a single database. This method is practically very expensive as it requires maintaining two databases and moreover during authentication the search process is doubled. Barni et al. [46] designed a privacy preserving protocol for finger print identification using finger codes. This protocol uses a multi-party computations approach and makes use of hemimorphic properties of pallier cryptosystem. Feng et al. [47] used cancellable biometrics and proposed a three step hybrid approach for face template protection. All the biometric authentication mechanisms used are computationally very expensive because of the low computation power of mobile devices.

Our proposed biometric authentication mechanism makes use of distributed cloud services to perform computations and thereby reducing load on mobile devices. We used distributed cloud here as we need all the data close to the users. Moreover biometrics is very sensitive data, and we need to make sure we protect the privacy of the user. We propose a secure biometric authentication which is not only very fast compared to traditional password based authentication mechanisms, but also more secure and protects privacy of the user.

8.3 SYSTEM MODEL

This section describes the system model of our application. In this section we present an overview of our system, the main components of our system and flow among different components of the system. .

8.3.1 SYSTEM OVERVIEW

Using our mobile application any registered user (here by registered user we mean a user who already has an account in our mobile application) can log in to the application and purchase products from any participating store which subscribes to our application for mobile shopping. To purchase any product a user needs to scan the bar code of the product using their mobile phone. The scanned products will be added to the user account's cart. After selecting all the products the user wants to purchase, our mobile application let the user proceed towards checkout. To make a payment the user bank pays the amount to the merchant bank account. Our system secures the user log in to the application as well as the transaction from the user bank account to the merchant bank account.

8.3.2 SYSTEM COMPONENTS

The main components of our system are the mobile application, the cloud server, the user bank and the merchant bank.

- **The Mobile Application**

This is the main component of our system. Any registered user of our application can purchase items from a store and pay for the items using their mobile devices. To make this happen this component creates a connection with two other components, namely, the distributed cloud server and the user bank.

- **The distributed cloud Server**

The mobile application connects to a centralized server which holds information regarding the appropriate private distributed cloud. The distributed cloud server is used for two main purposes, authentication and mobile shopping and payment as shown in Fig. 25. We use the public distributed cloud for authentication purposes and private distributed cloud for inventory management and payment. For authentication where privacy preserving mechanisms which will ensure user privacy, we use the public distributed cloud. The data required for authentication would be placed on any of the nodes. We use kademlia, a DHT explained in chapter 4 to look for the user data stored. Since the data regarding the store items should be retrieved very fast and to maintain security of inventory, the private distributed cloud is used.

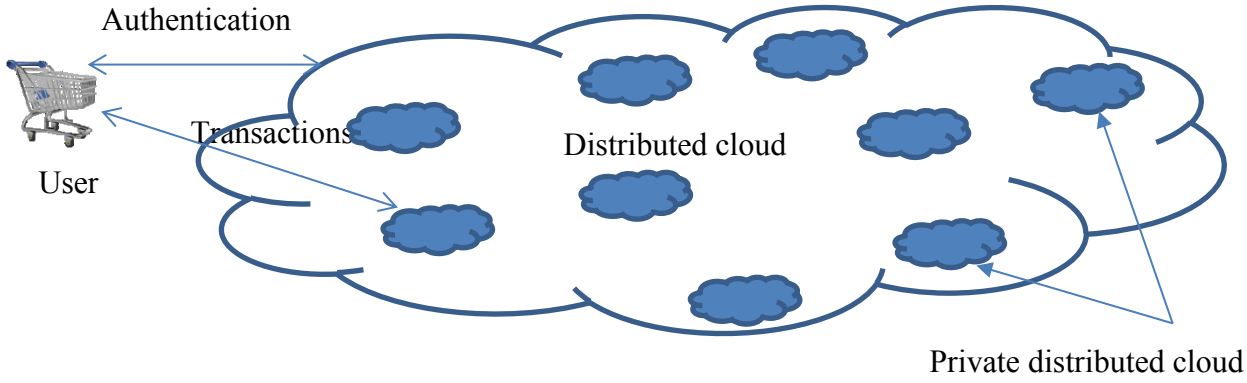

Figure 25: Public and private distributed cloud server roles

To purchase an item from a store the user need to see the item details from the selected store. The private distributed cloud server stores all the store details and the items' details of a store. The mobile application creates a connection to the private distributed cloud and fetches the information about a store and its items' details whenever it is required. The distributed cloud server is located near every store. All the private distributed cloud servers of similar stores keep in contact with each other to keep track of inventory and transactions. The main drawback of using a centralized cloud here is a single point of failure and latency. It takes a lot of time and bandwidth to support a large number of customers in multiple stores using a centralized cloud. Moreover to accommodate these many users the cloud should increase the bandwidth. The distributed cloud will divide the whole load into multiple smaller servers and thereby we can avoid single point of failure and need to increase bandwidth.

- **The User Bank**

The mobile application creates a connection with the user bank when the user proceeds with check out to pay for their purchase. The user bank creates a connection with the merchant bank and checks the authentication of the merchant account with the help of the merchant bank. If the merchant is an authenticated one then the user bank transfers the payment amount from the user account to the merchant account.

- **The Merchant Bank**

The user bank creates a connection to the merchant bank to start the transaction procedure. After the merchant authentication, the user bank transfers the payment amount from the user account to the merchant account.

8.3.3 FLOW OF THE SYSTEM

In this subsection we explain the workflow among different components of the system. Fig 26 shows the workflow among different components of the system. The system starts when a user log in to the mobile application. To log in to the application the application verifies user authentication first. We can have at each of the distributed cloud where user connects based on the location and has authentication verified. Each user when tries to authenticate will be routed to a close by distributed cloud resource. The data is retrieved using the distributed storage cloud mentioned in chapter 4 and authentication process takes on.

Here we assume that the user is a registered authenticated user and is logged in to the mobile application.

- The mobile application sends request for the user desired store details to the distributed cloud server.
- The distributed cloud server sends the store details to the mobile application.
- The User scans his desired products from the store by scanning the bar code of the products. The mobile application adds the product to the user's cart after the user scans the bar code of the product. All the data of particular store is stored in private distributed cloud server at that store. We use public resources only for authentication.

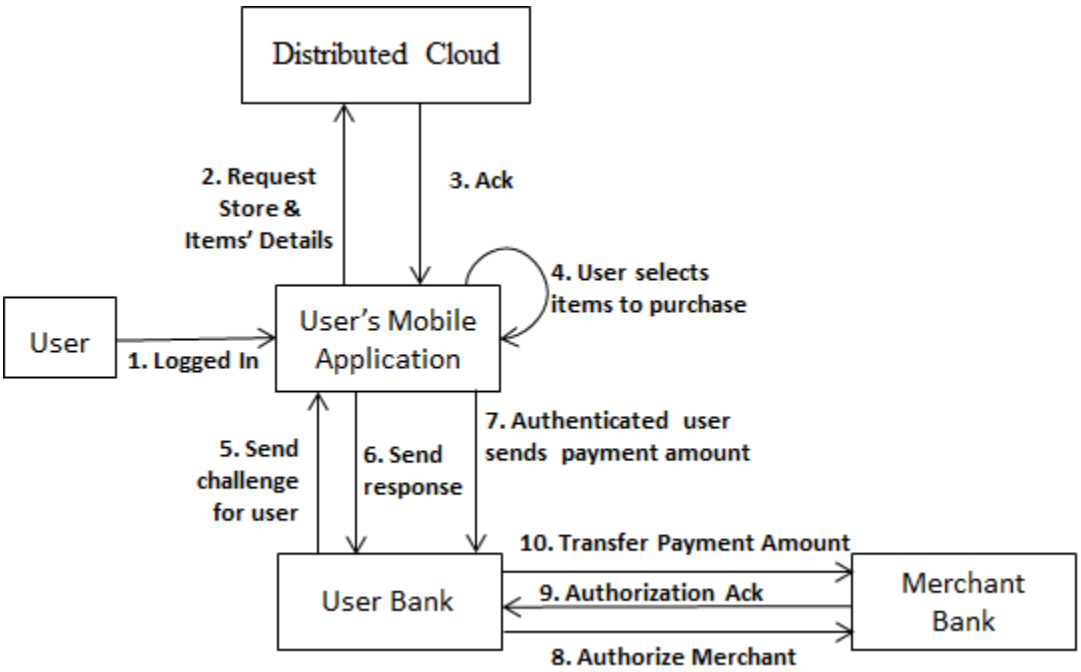

Figure 26: System Architecture

- When the user is ready to check out the mobile application creates a connection to the user bank.
- The user bank checks that whether the user is authorized. If the user is an authorized user then the bank creates a connection with the merchant bank to check the merchant authorization.
- If the merchant bank account is an authorized one then the user bank starts the transaction from the user bank account to the merchant bank account.
- The Merchant bank notifies the user bank of a successful transaction and the user bank sends notification to the user for successful payment.

If the user does no activity (e.g., does not use the application to proceed with the shopping) after logging in to the application, then the application logs out the user from the application for security purposes.

8.3.4 SECURITY MODEL OF THE SYSTEM

Mobile devices are prone to be lost or stolen frequently. Any adversary can access or misuse the mobile phone if it is stolen or lost. Since our mobile application is related to transactions from the user bank account, it should be secure so that no adversary can access the user account. Our application is secure in a semi-honest model [64]. In the semi-honest model, each party follows the protocol and adversaries may try to extract more information from observing the execution of the algorithm. The Semi-honest model is a standard adversary model in cryptography and it has been adopted by many previous works on privacy protection [65]. Intuitively, if privacy is guaranteed in the semi-honest model, it means that no adversary can efficiently obtain more information than the input and the output of the algorithm. In our application any semi-honest adversary who attempts to learn about the information between application and the cloud server will be able to observe nothing but encrypted data, since our mobile application sends only encrypted data to the cloud server.

In this model we are preserving user information privacy and bank details so that no malicious user can log in to the user account and access their bank account. To implement this we had provided two fold securities, so that only an authenticated user can use the application and the transaction from the bank remains secure always. To authenticate user log in to the mobile application we use user biometrics (e.g., user face image) as it is more secure.

8.3.5 IMPLEMETATION

We designed and developed an Android mobile application for secure mobile payments. To implement this application we used Java programming language on a computer with 3.33 GHz

Intel Core i5 processor, 4 GB RAM, 64 bit operating system. To test our application we use Google Nexus 7 with Android version 4.2.1. We implemented our system by developing an Android application and to do that we used Android SDK and Android Developer Tools (ADT). The main system components are connected to each other over the Internet. The communications between the cloud server and the application is realized by Java Platform, Enterprise Edition (Java EE) and Java EE APIs. we use javax.servlet package and use HTTP requests which includes the Java Server Pages (JSP) specification. To implement the communication between two banks and with the user we use javax.servlet packages. For extracting the feature vector from the user biometrics we use the PCA - based Face Recognition System package using Matlab.

8.4 CONCLUSION

We have designed and developed a mobile application for secure mobile payments. To make this application secure we focus on secure user log in to the mobile application and secure transactions from the bank. For secure log in we use biometrics based Authentication. We used the distributed cloud environment for both computation and authentication. Since mobile devices are low power devices privacy preserving authentication mechanisms cannot run efficiently on them. So we used distributed cloud computing resources to perform computations required for authentication and mobile payment.

CHAPTER 9

CONCLUSION

In this thesis we proposed a detailed architecture for the distributed cloud computing model. Distributed cloud computing can be used for applications which don't require huge data centers and applications which require data to be close to clients location. We developed a novel resource discovery scheme to work with multi-attribute distributed systems. The resource discovery mechanism developed works for finding resources required efficiently. We also noted that using our resource discovery mechanism, success rates for finding nodes are 90% with searches less than 3 which is a lot better than using existing mechanisms.

We developed multiple game theoretic approaches for resource allocation using game theory. Game theoretic algorithms were proposed and proven to be applicable for real world scenarios. The auction mechanism described in chapter 5 is used for resource allocation. We also developed the Bertrand model for analyzing the market and strategize the pricing for stability of the system. So using the Bertrand game we decide which auction game is to be used for a particular user based on their participation. We could see from the observations that the free riding problem could be solved and the system is stabilized using the above game theoretic approaches.

We have developed a network plugin for a distributed system client VUZE and made real time measurements useful to estimate the latency accurately. We used planet lab nodes to make sure our approach using Treeple network coordinates are stable over time. We use planet lab nodes as their location closely resembles a distributed cloud scenario. Treeple is a new scheme for secure latency estimation in P2P networks.

We also proposed a distributed cloud computing application for secure mobile transaction. In this mechanism users require distributed cloud resources to be available close to them in order to avoid any delays. Different markets can form distributed hybrid cloud for storage and management of inventory. Biometric authentication for mobile devices can be performed on available resources provided by regular users.

Future work includes more advance game theoretic approaches for resource allocation. We intend to work on coalition games where users can form coalitions to offer resources and in turn stabilize the system i.e. makes sure everyone is participating and profiting. There are many research challenges for designing of a distributed cloud environment such as data replication [18] techniques, incentive mechanisms etc.

Mobile applications for payment systems needs to be very secure to perform transactions while being efficient. We are working on developing a mobile application will provide both secure transactions and preserve the privacy of the user.

REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica and Matei Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing”, Technical report EECS-2009-28, UC Berkeley, 2009

[2] Peter Mell and Timothy Grance, “NIST definition of cloud computing, National Institute of Standards and Technology”, Special Publication 800-145, 2011, “http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf”, Retreived July 22, 2014

[3] Buyya, R. and Chee Shin Yeo and Venugopal, S., “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities”, Proccedings 10th IEEE International Conference on. High Performance Computing and Communications,. HPCC’08. IEEE, 2008.

[4] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Warfield, A.,“Xen and the art of virtualization”, ACM SIGOPS Operating Systems Review, Vol. 37, No. 5, pp. 164-177,2003.

[5] Fox, Armando. "Cloud computing-what's in it for me as a scientist." Science, Vol. 331, pp. 406-407, 2011.

[6] Endo, Patricia Takako, Andre Vitor de Almeida Palhares, Nadilma Nunes Pereira, Glauco Estacio Goncalves, Djamel Sadok, Judith Kelner, Bob Melander, and J-E. Mangs. "Resource allocation for distributed cloud: concepts and research challenges." IEEE Network, Vol.25, No.4, pp.42,46, July-August 2011.

[7] Khethavath, Praveen, Johnson Thomas, Eric Chan-Tin, and Hong Liu. "Introducing a Distributed Cloud Architecture with Efficient Resource Discovery and Optimal Resource Allocation.", Proceedings IEEE Ninth World Congress on Services (SERVICES), pp. 386-392. IEEE, 2013.

[8] Church, Kenneth, Albert Greenberg, and James Hamilton. "On delivering embarrassingly distributed cloud services.", Proceedings Hotnets-VII, 2008.

[9] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric.", Revised Papers from the First International Workshop on Peer-to-Peer Systems, pp.53-65, March 07-08, 2002.

[10] Stoica, Ion, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications.", IEEE/ACM Transactions on Networking, vol.11, no.1, pp.17-32, Feb 2003.

[11] Fletcher, George, Hardik Sheth, and Katy BÃ˝urner, "Unstructured peer-to-peer networks: Topological properties and search performance", Proceedings of the Third international conference on Agents and Peer-to-Peer Computing, 2004.

[12] Rimal, Bhaskar Prasad, Eunmi Choi, and Ian Lumb. "A taxonomy and survey of cloud computing systems.", Fifth International Joint Conference on INC, IMS and IDC, pages 44-51, 2009.

[13] Fernndez-Baca, David,"Allocating modules to processors in a distributed system.", IEEE Transactions on Software Engineering, Vol.15 No.11, pp.1427-1436, November 1989.

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

[15] Attiya, Hagit, and Jennifer Welch, *Distributed computing: fundamentals, simulations, and advanced topics,* John Wiley & Sons, 2004.

[16] VUZE, “http://www.vuze.com/corp/technology.php”, Retreived July 22, 2014.

[17] King data set, “https://pdos.csail.mit.edu/p2psim/kingdata”. Retreived July 22, 2014.

[18] V. Martins, E. Pacitti, P. Valduriez., “ Survey of Data Replication in P2P Systems.”, Technical Report, 2006, “ http://hal.inria.fr/inria-00122282/”, Retreived July 23, 2014.

[19] Yanfeng Shu , Beng Chin Ooi , Kian-Lee Tan , Aoying Zhou, “Supporting Multi-Dimensional Range Queries in Peer-to-Peer Systems”, Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing, pp.173-180, August 31-September 02, 2005.

[20] Gupta, Abhishek, Divyakant Agrawal, and Amr El Abbadi, "Approximate range selection queries in peer-to-peer systems.", Proceedings First Biennial Conf. on Innovative Data Systems Research CIDR. . 2003.

[21] Lawder, Jonathan K., and Peter J. H. King. "Querying multi-dimensional data indexed using the Hilbert space-filling curve." ACM Sigmod Record, Vol. 30 No. 1,pp.19-24, 2001.

[22] Dong-Qing Yao, John J. Liu, “Competitive pricing of mixed retail and e-tail distribution channels”, Omega, Volume 33, Issue 3, Pages 235-247, ISSN0305-0483, June 2005.

[23] F. Teng and F. Magoules, “A new game theoretical resource allocation algorithm for cloud computing”., Proceedings of the 5th international conference on Advances in Grid and Pervasive Computing, May 10-13, 2010.

[24] Ke Xu; Meina Song; Xiaoqi Zhang; Junde Song, "A Cloud Computing Platform Based on P2P," Proceedings IEEE International Symposium on IT in Medicine & Education, ITIME '09, vol.1, no., pp.427-432, Aug. 2009.

[25] Cunsolo, Vincenzo D., Salvatore Distefano, Antonio Puliafito, and Marco Scarpa. "Cloud@ home: Bridging the gap between volunteer and cloud computing.", Proceedings of the 5th international conference on Emerging intelligent computing technology and applications, September 16-19, 2009.

[26] Ozalp Babaoglu, Moreno Marzolla, and Michele Tamburini. 2012, “Design and implementation of a P2P Cloud system”, Proceedings 27th Annual ACM Symposium on Applied Computing (SAC '12), pp: 412-417, 2012.

[27] Anderson, David P. "Boinc: A system for public-resource computing and storage.", Proceedings Fifth IEEE/ACM International Workshop on Grid Computing, 2004.

[28] Chan-Tin, Eric, and Nicholas Hopper. "Accurate and Provably Secure Latency Estimation with Treeple.", Proceedings Network and Distributed System Security (NDSS), 2011.

[29] Wei, Guiyi, et al.,"A game-theoretic method of fair resource allocation for cloud computing services.", The Journal of Supercomputing, Vol.54 No.2, pp.252-269, November 2010.

[30] Dabek, Frank, et al. "Vivaldi: A decentralized network coordinate system.", ACM SIGCOMM Computer Communication Review. Vol. 34. No. 4. ACM, 2004.

[31] Qi, ShouQing, et al. "A Novel P2P Network Model for Cloud Computing Based on Game Theory", Proceedings IEEE International Conference on Computer Science & Service System (CSSS) , 2012.

[32] Wang, Bo, et al. "The Research on Cloud Resource Pricing Strategies Based on Cournot Equilibrium.", In Recent Advances in Computer Science and Information Engineering, pp. 253-259. Springer Berlin Heidelberg, 2012.

[33] Jin, Xin, Yu-Kwong Kwok, and Yong Yan. "A Study of Competitive Cloud Resource Pricing under a Smart Grid Environment." Proceedings IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), 2013. Vol. 1., 2013.

[34] Upkar Varshney, “Mobile Payments”, Computer, Vol.35 No.12, pp..120-121, December 2002.

[35] Sadeh, Norman. *M-commerce: technologies, services, and business models*, John Wiley & Sons, 2003.

[36] Clarke, Nathan L., and Steven M. Furnell. "Authentication of users on mobile telephones–A survey of attitudes and practices." Computers and Security, Vol. 24, No. 7, pp. 519-527, 2005.

[37] Salil Prabhakar , Sharath Pankanti , Anil K. Jain, “Biometric Recognition: Security and Privacy Concerns”, IEEE Security and Privacy, Vol.1 No.2, pp.33-42, March 2003.

[38] Kungpisdan, Supakorn, Bala Srinivasan, and Phu Dung Le. "A secure account-based mobile payment protocol.", Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Vol. 2, pp.35, April 05-07, 2004.

[39] Alia Fourati , Hella Kaffel Ben Ayed , Farouk Kamoun , Abdelmalek Benzekri, “A SET Based Approach to Secure the Payment in Mobile Commerce”, Proceedings of the 27th Annual IEEE Conference on Local Computer Networks, pp.136-140, November 06-08, 2002.

[40] Tiwari, A., Sanyal, S., Abraham, A., Knapskog, J. S. & Sanyal, S., "A Multi-factor Security Protocol for Wireless Payment-Secure Web Authentication Using Mobile Devices", Proceedings IADIS International Conference Applied Computing, pp.160-167, 2007.

[41] Harb, Hany, Hassan Farahat, and Mohamed Ezz. "SecureSMSPay: secure SMS mobile payment model.", Proceedings 2nd International Conference on Anti-counterfeiting, Security and Identification (ASID), IEEE, 2008.

[42] Pourghomi, Pardis, and Gheorghita Ghinea. "Managing NFC payment applications through cloud computing.", Proceeding 7th IEEE International Conference for Internet Technology and Secured Transactions (ICITST), pp. 772--777, 2012.

[43] Gordon, Michael, and Suresh Sankaranarayanan. "Biometric security mechanism in Mobile paymentts.", Proceedings 7th IEEE International Conference On Wireless And Optical Communications Networks (WOCN), 2010.

[44] Cheng, Hongbing, et al. "Identity based encryption and biometric authentication scheme for secure data access in cloud computing." Chinese Journal of Electronics, Vol. 21, No. 2, pp. 254-259, 2012.

[45] Ross, Arun, and Asem Othman. "Visual cryptography for biometric privacy." IEEE Transactions on Information Forensics and Security, Vol. 6, No.1 pp. 70-81, 2011.

[46] Mauro Barni , Tiziano Bianchi , Dario Catalano , Mario Di Raimondo , Ruggero Donida Labati , Pierluigi Failla , Dario Fiore , Riccardo Lazzeretti , Vincenzo Piuri , Fabio Scotti , Alessandro Piva, “Privacy-preserving fingercode authentication”, Proceedings of the 12th ACM workshop on Multimedia and security, September 09-10, 2010.

[47] Feng, Y. C., Pong C. Yuen, and Anil K. Jain. "A hybrid approach for face template protection.", IEEE Transactions on Information Forensics and Security, Vol. 5 No. 1, March 2010.

[48] Amazon EC2, “http://aws.amazon.com/ec2/”, Retreived July 22, 2014

[49] Microsoft Azure, “http://azure.microsoft.com/en-us/”, Retreived July 22, 2014

[50] Oracle sun cloud, “https://www.oracle.com/cloud/index.html”, Retreived July 22, 2014

[51] Openstack, “https://www.openstack.org/”, Retreived July 22, 2014

[52] Apache cloud stack, “http://cloudstack.apache.org/”, Retreived July 22, 2014

[53] VMWare, “http://www.vmware.com/cloud-computing/overview.html”, Retreived July 22, 2014

[54] IBM Hybrid Cloud, “http://www-01.ibm.com/software/tivoli/products/hybrid-cloud/”, Retreived July 22, 2014

[55] HP Hybrid cloud, “http://www8.hp.com/us/en/software-solutions/cloud-management/”, Retreived July 22, 2014

[56] IBM Federal cloud computing solutions, "http://www-304.ibm.com/industries/publicsector/us/en/promotion/!!/xmlid=218253", Retreived July 22, 2014

[57] Google App engine, "https://developers.google.com/appengine", Retreived July 22, 2014

[58] Heroku, "https://www.heroku.com/", Retreived July 22, 2014

[59] Sales force, "http://www.salesforce.com/", Retreived July 22, 2014

[60] Eucalyptus, "https://www.eucalyptus.com/eucalyptus-cloud/iaas", Retreived July 22, 2014

[61] Klyne, Graham, and Jeremy J. Carroll. "Resource description framework (RDF): Concepts and abstract syntax.", W3C Recommendation, 2004, "http://www.w3.org/TR/rdf-concepts/", Retreived July 23, 2014.

[62] Network description language, "http://sne.science.uva.nl/ndl/", Retreived July 22, 2014

[63] Zhao, Hao, and Sead Muftic. "The concept of secure mobile wallet.", Proceedings World Congress on Internet Security (WorldCIS), pp.54-58, 2011.

[64] Goldreich, Oded, *Foundations of Cryptography: Basic Applications*, Vol. 2. Cambridge University Press, 2009.

[65] Lindell, Yehuda, and Benny Pinkas, "Privacy preserving data mining.", Advances in Cryptology—CRYPTO, Springer Berlin Heidelberg, 2000.

[66] Fernando, Niroshinie, Seng W. Loke, and Wenny Rahayu, "Mobile cloud computing: A survey.", Future Generation Computer Systems, Vol. 29, No.1 pp. 84-106, 2013.

[67] Bahl, Paramvir, et al. "Advancing the state of mobile cloud computing.", Proceedings of the third ACM workshop on Mobile cloud computing and services, ACM, 2012.

[68] Shiraz, M., Gani, A., Khokhar, R. H., & Buyya, R. (2013), "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing", IEEE Communications Surveys Tutorials, , Vol. 15, No. 3, pp. 1294-1313, 2013 .

VITA

Praveen Khethavath

Candidate for the Degree of

Doctor of Philosophy

Thesis: TOWARDS AN EFFICIENT DISTRIBUTED CLOUD ARCHITECTURE

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2014.

Completed the requirements for the Master of Science in Computer Science at University of Northern Virginia, Annandale, Virginia in 2008.

Completed the requirements for the Bachelor of Engineering in Electronics and Communication Engineering at CBIT, Osmania University, Hyderabad, Andhra Pradesh, India in 2006.