MOBILIM : INTEGRATED LICENSE MANAGEMENT & ECONOMIC RESOURCE ALLOCATION FOR CLOUD COMPUTING

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

By

PANKAJ B. THORAT

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)

JUNE, 2011

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled "**MOBILIM: AN INTEGRATED LICENSE MANAGEMENT AND RESOURCE ALLOCATION FOR CLOUD COMPUTING**" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science and Engineering** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2010 to June 2011, under the guidance of **Dr. Anil K. Sarje, Professor,** Department of Electronics and Computer Engineering, IIT Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

Date: 9-06-2011

Place: Roorkee

(PANKAJ B. THORAT)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date:

Place: Roorkee

9/6/2011

(Dr. Anil K. Sarje)

Professor

Department of Electronics and Computer Engineering

IIT Roorkee.

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my heartfelt gratitude to my guide and mentor **Dr. Anil K. Sarje**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for his invaluable advices, guidance, encouragement and for sharing his broad knowledge. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. He has been very generous in providing the necessary resources to carry out my research. He is an inspiring teacher, a great advisor, and most importantly a nice person.

I am greatly indebted to all my friends especially Abhishek Bichhawat, who have graciously applied themselves to the task of helping me with ample moral supports and valuable suggestions.

On a personal note, I owe everything to the Almighty and my parents. The support which I enjoyed from my father, mother and other family members provided me the mental support I needed.

PANKAJ B. THORAT

Abstract

Cloud computing is a recent trend in IT that moves computing and data away from desktop and portable PCs into large data centers. It refers to applications delivered as services over the Internet as well as to the actual cloud infrastructure — namely, the hardware and systems software in data centers that provide these services. Cloud computing is on-demand computing in which the computing resources are owned and managed by a service provider and the users access the resources via the Internet. But cloud computing potential doesn't begin and end with the personal computer's transformation into a thin client. The mobile platform is going to be heavily impacted by this technology as well. License management and economic resource allocation is a major issue faced by cloud computing paradigm.

In this dissertation we propose MobiLim, a framework which addresses major problems related to license management and economic resource allocation in cloud computing. This framework named MobiLim is managed by independent software vendor (ISVs), who controls access to the resource provider resources. MobiLim provides a secure and robust license management solution as well as allocate resources economically to the customers. MobiLim acts as an intermediate entity between customer and resource provider which accepts customers request for the resources. If MobiLim is capable of allocating the resource to customer then it performs negotiation over price between customer and resource provider. If the negotiation is successful then MobiLim grants a valid license to the customer. A license allows the customer to gain access to the resource provider resource which customer needs. The proposed framework is modular due to which it is possible to make it more robust and secure with the help of advanced cryptographic techniques. The framework is tested with a set of customer requests and the results validate correctness of proposed framework.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Statement of the Problem

1.1 Introduction

A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers [1, 2]. A cloud computing platform dynamically provisions, configures, reconfigures, and de-provisions services as needed. Cloud computing is a paradigm in which computing resources such as processor, memory, software applications and storage are not physically present at the user's location. Instead, a service provider owns and manages these resources, and users access them via the Internet [3]. Unlike applications that are downloaded and installed onto the end user's devices, these applications run inside the cloud and can be accessed by any device running a browser.

Cloud computing is emerging as a new distributed system which aims to provide reliable, customized and QoS guaranteed dynamic computing environments for end users [3]. In a Cloud computing, different users demand various resources of service provider according to their requirement and are charged for that by the service provider. But before getting the access to the resources which are owned by some service provider, user needs to acquire a valid license depending on his requirement and budget. Cloud computing is usually offered with a pay per use model [4] in which you pay for just the cloud resources that a particular job requires. It is an on-demand service, users pay only for what they have used. The usage based billing model of cloud computing makes it preferable to the small and medium enterprises which cannot afford a dedicated infrastructure of this size. So, for the benefit of customers and service provider, resource allocation must be economic. The success of achieving this goal in proper time and to obtain higher quality of results in these dynamic and distributed environments depends on implementing an appropriate collaboration mechanism between service providers and customer.

1.2 Motivation

Cloud computing emerge as a hot topic since 2007 due to its abilities of offering flexible dynamic IT infrastructures, QoS guaranteed computing environments and configurable software services. As reported in Google trends [5] (Figure 1.1), Cloud computing (blue line), which is enabled by Virtualization technology (yellow line), has outpaced Grid computing (red line).

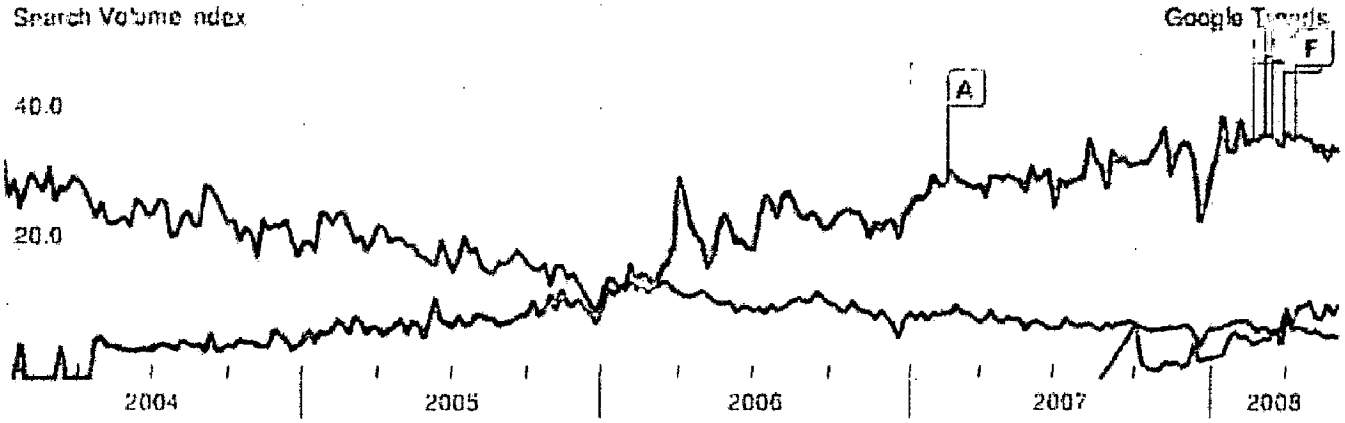

Fig. 1.1 Cloud computing in Google trends

"Cloud" computing – a relatively recent term, builds on decades of research in virtualization, distributed computing, utility computing, and more recently networking, web and software services. It implies a service oriented architecture, reduced information technology overhead for the end-user, great flexibility, reduced total cost of ownership, on demand services and many other things. A recent report by M. Armbrust et al. [6] states that license management is one of ten issues which needs to be tackled for the success of cloud computing. License management is the most important factor for the successful adoption of cloud computing for a service provider. The licensing scheme should be beneficial and compatible for both user as well as service provider. Service providers and consumers compete for providing and employing resources, trade handling in a fair and stable way is also a challenging task. The fairness must be in terms of price settings with right incentives and offering of cloud services should be supported by a suitable resource allocation, accounting, and pricing scheme. Economic resource allocation exploits the capability of resources efficiently and satisfies the user's reasonable requests. A sustainable economic cloud has two characteristics: it must allow

resource providers and resource consumers to make autonomous decisions, and both parties of providers and consumers must have sufficient incentives to stay and play in the market [7].

In this report, we propose and implement MobiLim, a novel model for integrated license management and economic resource allocation model for cloud computing, which issues licenses to users and provides a secure mechanism to manage the digital identity and authorization mechanisms required to describe the users, license issuers, and the cloud server by using cryptographic techniques. MobiLim allows the ISVs to manage the licenses in a distributed environment as well as its act as a resource broker whose job is to allocate the resources to the customer in a reasonable price depending on the resource availability and other parameters. Cloud computing is dynamic in terms of resource availability because at the same time multiple customers may request the resources or release them after use. The price of a resource depends on its availability, budget of customer and the deadline within which resources should be allocated. In cloud computing, there are different resource providers, each having various resources at various prices. The price of the resources depends upon the quality of service, availability of the resources and requirement of the user. There are multiple self-interested agents that supply or consume multiple types of resources, but the problems with these agents are

1) Consumers dynamically enter and leave the market.
2) Consumers have some bounded flexibility over when they require resources.
3) A single provider cannot satisfy consumers' resource requirements.

The first two characteristics are evident in current cloud platforms that are available to the general public, which allows them to execute tasks that may or may not have hard deadlines. As cloud computing environments becoming more commonplace and complex, users will have a difficult time integrating the various cloud services and ensuring their integrity. To tackle this problem there is a need of cloud service brokerages which will essentially negotiate the relationship between the end users and service providers and make it easier for businesses to manage their cloud services [8].

1.3 Statement of the Problem

The aim of the dissertation is to develop and implement a new technique for providing an integrated License Management and Economic Resource Discovery and Allocation model for cloud computing.

The various technological goals for building a robust, secure licensing mechanism integrated with economic resource allocation for cloud computing includes the following:

- Establishing reliable interconnected network of digital authorization and identification among various software vendors, cloud infrastructures, licensing servers, and cloud provisioning servers.
- Software vendors must be able to describe their course of action in a license expression language which differentiates between the end client of the resources, independent of the hosting organization, or the organization that is configuring and installing the software.
- Enabling vendor to choose, preserve their existing licensing approach on the cloud, or adopt a usage- based or subscription-based model by allowing flexible pricing models, and assign the task of billing and pricing to some third party providers that are specialized in that field.
- Enabling the agencies to aggregate, resell and provide integrated software on the cloud, along with custom licenses and prices for reselling agreements.
- Providing a single interface which brings together multiple cloud services for billing and metering, monitoring, single sign on and value added services.
- Providing a cloud management tool to track, measure, monitor, and enforce policies across interactions. It can also be used for authorization and access management, creating and managing user profiles.
- Providing resources at some mutual agreeable price through negotiation between customer and resource provider.
- Billing should be usage based, i.e., pay per use.
- The architecture must be flexible enough to incorporate with existing software modules as well as improved framework.

1.4 Organization of the Report

This dissertation report comprises of six chapters including this chapter that introduces the topic and states the problem. The rest of the report is organized as follows.

Chapter 2 gives the background of cloud computing, services provided by cloud, its type. Besides this, it states various negotiation techniques used in e-commerce application and its limitations and brief literature review of related work including research gaps.

Chapter 3 describes the framework designed for integrated license management and economic resource allocation. It describes important modules of framework which includes negotiation strategy of MobiLim, cryptographic algorithm used in the design of MobiLim and working of customer, MobiLim server and server provider.

Chapter 4 gives the implementation details of the proposed framework, details of experiments performed.

Chapter 5 discusses the results obtained after negotiation based on various strategies of MobiLim.

Chapter 6 concludes the dissertation work and gives suggestions for future work.

Chapter 2

Background and Literature Review

2.1 Cloud computing

A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumer [9]. Cloud computing describes both a platform and a type of application. A cloud computing platform dynamically provisions, configures, reconfigures, and deprovisions servers as needed.

Cloud applications are extended to be accessible through the Internet. These cloud applications use large data centers and powerful servers that host Web applications and Web services. A cloud is a pool of virtualized computer resources. A cloud can:

- Host a variety of different workloads, including batch-style back-end jobs and interactive, user-facing applications
- Allow workloads to be deployed and scaled-out quickly through the rapid provisioning of virtual machines or physical machines
- Support redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures
- Monitor resource use in real time to enable rebalancing of allocations when needed
- Cloud computing offers immediate access to large numbers of the world's most sophisticated supercomputers and their corresponding processing power, interconnected at various locations around the world, proffering speed in the tens of trillions of computations per second.
- The concept incorporates infrastructure as a service **(IaaS)**, platform as a service **(PaaS)** and software as a service **(SaaS)**. Examples of SaaS vendors include Salesforce.com [11] and Google Apps [12] which provide common business

applications online that are accessed from a web browser, while the software and data are stored on the servers.

2.1.1. Types of cloud

As we focus on building the cloud, a number of models have been developed for deploying a cloud infrastructure [10].

- Public cloud: Public cloud or external cloud describes cloud computing in the traditional mainstream sense, whereby resources are dynamically provisioned on a fine-grained, self-service basis over the Internet, via web applications/web services, from an off-site third-party provider who shares resources and bills on a fine-grained utility computing basis [10]. The main benefits of using a public cloud service are: easy and inexpensive set-up because hardware, application and bandwidth costs are covered by the provider, scalability to meet needs, no wasted resources because you pay for what you use.

- Private cloud: A private cloud offers many of the benefits of a public cloud computing environment, such as being elastic and service based. The difference between a private cloud and a public cloud is that in a private cloud-based service, data and processes are managed within the organization without the restrictions of network bandwidth, security exposures and legal requirements that using public cloud services might entail. In addition, private cloud services offer the provider and the user greater control of the cloud infrastructure, improving security and resiliency because user access and the networks used are restricted and designated. The term has also been used in the logical rather than physical sense, for example in reference to platform as service offerings, though such offerings including Microsoft's Azure Services Platform [11] are not available for on-premises deployment.

- Community Cloud: A community cloud is controlled and used by a group of organizations that have shared interests, such as specific security requirements or

a common mission. The members of the community share access to the data and applications in the cloud.

Following is the table which compares between different deployments models [5].

Table 2.1: Deployment models of cloud.

<table>
<tr><th></th><th>Managed by</th><th>Infrastructure owned by</th><th>Infrastructure located</th><th>Accessible and consumed by</th></tr>
<tr><td>Public</td><td>Third party provider</td><td>Third party provider</td><td>Off premise</td><td>Untrusted</td></tr>
<tr><td>Managed</td><td>Third party provider</td><td>Third party provider</td><td>Off premise</td><td>Untrusted & trusted</td></tr>
<tr><td rowspan="4">Private</td><td rowspan="2">Third party provider</td><td rowspan="2">Organization</td><td>Off premise</td><td rowspan="4">Trusted</td></tr>
<tr><td>On premise</td></tr>
<tr><td rowspan="2">Organization</td><td rowspan="2">Third party provider</td><td>Off premise</td></tr>
<tr><td>On premise</td></tr>
<tr><td>Hybrid</td><td>Both organization & Third party provider</td><td>Both organization & Third party provider</td><td>Both Off premise & On premise</td><td>Both Trusted & Untrusted</td></tr>
</table>

- Hybrid cloud: A hybrid cloud is a cloud computing environment in which an organization provides and manages some resources in-house and has others provided externally. For example, an organization might use a public cloud service, such as Amazon Simple Storage Service (Amazon S3) [12] for archived data but continue to maintain in-house storage for operational customer data. Ideally, the hybrid approach allows a business to take advantage of the scalability

and cost-effectiveness that a public cloud computing environment offers without exposing mission-critical applications and data to third-party vulnerabilities.

2.2. Services of cloud computing

There are different types of cloud services such as infrastructure, storage, hardware etc. These services are delivered and consumed over Internet. We discuss some of these services in details below:

2.1.1 SaaS - Software-as-a-Service

It is a model of software deployment whereby a provider licenses an application to customers for use as a service on demand. Software or an application is hosted as a service and provided to customers across the Internet. This mode eliminates the need to install and run the application on the customer's local computers. SaaS therefore alleviates the customer's burden of software maintenance, and reduces the expense of software purchases by on-demand pricing. One example of SaaS is the Salesforce.com [13] a customer relationship management application.

2.2.2 IaaS - Infrastructure-as-a-Service

It is the delivery of computer infrastructure (typically a platform virtualization environment) as a service. Rather than purchasing servers, software, data center space or network equipment, clients instead buy those resources as a fully outsourced service. As the result of rapid advances in hardware virtualization, IT automation and usage metering & pricing, users could buy IT hardware, or even an entire data center, as a pay-as-you-go subscription service. The HaaS is a flexible, scalable and manageable to meet your needs examples could be found at Amazon EC2 [15], IBM's Blue Cloud project [16], Nimbus [17], Eucalyptus [18] and Enomalism [19].

2.2.3 PaaS - Platform-as a-Service

It is the delivery of a computing platform and solution stack as a service. It facilitates the deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers. The consumer uses a hosting environment for

their applications. The consumer controls the applications that run in the environment (and possibly has some control over the hosting environment), but does not control the operating system, hardware or network infrastructure on which they are running.

2.3 Literature Review

Cloud computing [1] is emerging as a new distributed system which aims to provide reliable, customized and QoS guaranteed dynamic computing environments for end users. The success of achieving this goal in proper time and/or to obtain the higher quality of results in these dynamic and distributed environments depends on implementing an appropriate collaboration mechanism between service provider and customer in the cloud. Moreover, this appropriate collaboration mechanism should include a negotiation technique to allow service providers to require cooperation from other providers mainly because they are unable to provide a particular set of services.

In the future years, with the increase in the number of cloud service providers and the number of users, there is a need of some third party who will regulate the resource allocation and license management. MobiLim is designed to manage licenses and allocates resources through negotiation. The following are the currently most commonly used licensing techniques which are:

- In system fingerprinting solution [19], a fingerprint of the user's system, i.e., the system on which user is going to access the resources composed of CPU identification numbers and other hardware specific identifiers, is sent to the ISV and the ISV creates a license key and sends it to the user which unlocks the software or utility for the requested hardware.

- In dongle solution, before the execution of the program a token is presented. Depending on the communication with this hardware token, the access to particular feature is granted and checks which feature is licensed. This means that on a single physical machine, the dongle can only be accessed.

The above mentioned license management schemes are not suitable for Cloud environment because the above license management mechanisms licenses are restricted to small IP ranges or single machines. These mentioned mechanisms do not support usage based licensing on an arbitrary machine. The Hardware dongle solution is not applicable for distributed environments of cloud computing. Server based licensing mechanism [20] uses IP address for authentication which is not useful for the cloud environment because licenses should be mobile. The problem of allocating networked resources in dynamic environment, such as cloud computing platforms, where providers strategically price resources to maximize their utility, where both providers and consumers are "selfish agents", presents numerous challenges since the number of consumers and their resource demand is highly dynamic. There are numerous auction-based approaches have been proposed till now to tackle the problems occurred during the resource allocation.

An auction negotiates a mutually acceptable solution for the buyer and the seller [22] (it uses market forces to negotiate a clearing price for the auction item). The auction mechanism sets out rules for bidding, and allocates the computing resources to a certain bidder based on its predefined rule set. For example, in an auction, the auctioneer begins at the seller's reservation price, and solicits progressively higher oral bids from the audience until only one bidder is left. The winner claims the item, at the price it last bid. This auction carries the additional benefits that it makes bidder reservation prices publicly known and is efficient in the sense that it will give the object to the bidder who values it the most. Since resource demand and supply can be dynamic and uncertain, a distributed negotiation mechanism was proposed where agents negotiate over both a contract price and a de-commitment penalty, which allows agents to decommit from contracts at a cost. There are various reasons to prefer bargaining over auction mechanism some of them are:

1. Most auctions only allow negotiation for price, not other attributes (delivery time, loyal customer etc.).
2. Auctions usually are scheduled in advance and with time restrictions, e.g. some online auctions range from 1 hour to 1 week. Intrinsically, auctions need multiple buyers or sellers in order to work well, therefore needing some time for gathering participants. Some buyers/sellers may not want to wait until an auction opens or finalizes.

3. In some circumstances, non-attribute factors are important, e.g., trusteeships, friendships, etc. auctions cannot accommodate these factors.

Raiffa [22] established and compiled the mathematical basic of the negotiation models. He classified the different negotiation models in base to the characteristics of the environment and the negotiated goods. But it was just a mathematical model. Nipur et al. [19] propose a fault tolerant comparison internet shopping system BestDeal. The authors have conducted the simulation by launching nine shopping mobile agents where each has to visit five supplier sites to get the best deal for different products. Performance is measured in terms of execution steps as well as execution time of the simulation.

Faratin et al [23] applied and extended some existing model for service-oriented decision functions in bilateral negotiation between autonomous agents. It concentrates in many-parties many-issues, single-encounter negotiations with an environment of limited resources. Since computing services are qualitative in nature rather than quantitative, Faratin extends this model by adding qualitative values and associates fuzzy sets to them in order to express better the quality in the negotiations.

Venugopal et al. [25] introduced a bilateral negotiation Protocol, based on the Alternate Offers mechanism. The offers/counteroffers cycle of this paper is predefined finite, and in the Alternate Offer, sit can continue indefinitely until one or the other part decides to stop the negotiation. As cloud computing is a relatively new field, very less research has been done for tackling the issue of resource management, bargaining and negotiation.

2.4 Research Gaps

Till now there is no solution proposed which provides integrated license management and economic resource allocation. In the earlier licensing techniques, the contracts were made between the service provider and its customers in which customer buys the software and installs it on local resources in order to use it. But in cloud computing environment, this type of licensing is not possible and valid. In cloud computing, users use the services provided by service providers as needed. In order to preserve this business as a practice and requirement, licenses should be mobile i.e. a license must be location independent

and the user who has acquired the license must be able to use it from any device while ensuring that all legal restrictions are fulfilled. Also the resources should be provided in the reasonable price in accordance with market prices. Because customers are going to want to see an ecosystem of cloud computing services from multiple vendors that will allow them to dynamically allocate various jobs based on the capabilities and pricing offered through negotiation by the cloud computing service. When building an autonomous agent which is capable of flexible and sophisticated negotiation, two broad areas need to be considered.

- Which negotiation protocol will be used?
- What are the issues over which negotiation takes place?

There are various issues which need to be considered while choosing negotiation protocol. Negotiation with uncertainty is both the most challenging problem in the negotiation and the most practical problem for real cloud computing platforms. Bilateral bargaining considers only one type of uncertainty, such as a negotiation deadline [26] or reserve price [27]. In contrast, the negotiation between multiple agents in dynamic environments where there are multiple types of uncertainty that increases the difficulty of computing agents' rational equilibrium strategies. However, we think that the feature of time-dependence cannot reflect the real situations of online negotiation. There are two reasons.

- The Internet consists of a large number of heterogeneous sub-networks and their communication qualities are different from each other, which results into price offer that may not mean its real value and the same offer may mean different values for different agents.
- Time-dependent negotiations are inflexible because the deadline to close the negotiation process is determined in advance. The long preservation of negotiation will result in a time-consuming process and the short preservation may not achieve a satisfactory solution. A generic definition of online negotiation can be described as the time-independent process in which two or more participants search through the space of alternative ways to reach an acceptable agreement through Internet-based electronic interactions.

Chapter 3

Proposed Framework for MobiLim

The central idea of MobiLim is to accept the request of users/customers for the service and according to the requirements of the user specified in the license request token and the availability of the resources at that time, MobiLim server negotiates with the service provider. If the negotiation is successful then a license is granted to the customer. A license allows customer to access the resources on the resource provider site. In MobiLim there are three components: users/customer, resource providers, ISV that we have outlined below. The MobiLim client is used by the user for acquiring license before accessing the resources. MobiLim server is managed by ISV whose job is to issue license and negotiate between customer and service provider and third one is resource provider who is the owner of the resources. The basic architecture of MobiLim is shown in Fig. 3.1 and these three entities are explained below.

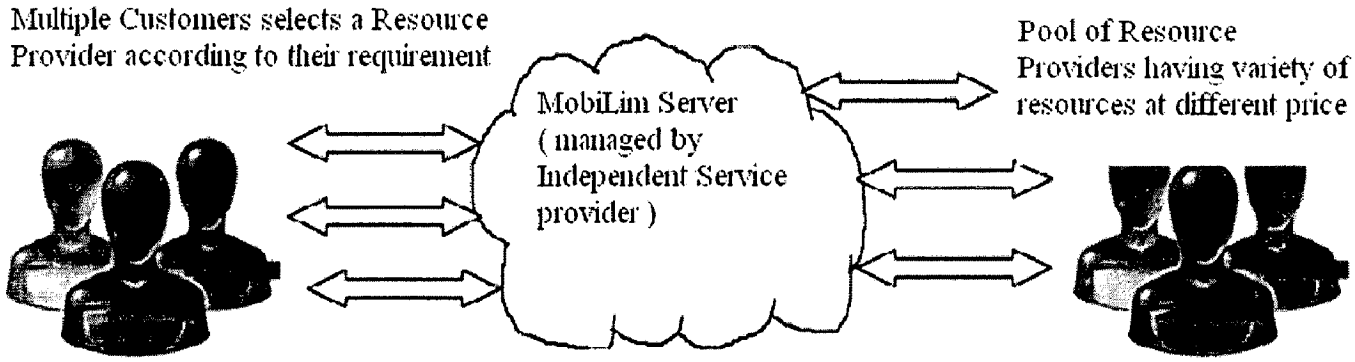

Fig. 3.1 Basic Architecture of MobiLim

3.1 Resource provider

Resource provider is the owner of resources and wants to sell his resources through MobiLim server. Resource provider submits his resource information like resource name, resource minimum prices, types of service, available number of copies of resource and maximum number of copies of a resource to the MobiLim server which is managed by

some independent software vendor. Different service provider owns different kind of resources like software, hardware and platform which they want to provide as a service to the customer. After issuing the license to customer, the service provider handles the request and allows the customer to access his resources.

3.2 MobiLim server

MobiLim server is managed by the ISV whose job is to negotiate between customer price value and price set by the service provider as well as to issue new licenses to new customers MobiLim sever maintains a database of various resources which belongs to different resource providers along with their attributes like its price, available number of copies and type of service. The MobiLim server is an entity that can legally issue software licenses, such as a vendor or reseller. Since MobiLim server has near-complete knowledge of the design of a provisioned software application, the MobiLim server is able to enforce a license's thresholds on unique usage metrics instead of the typical per-CPU or per-seat constraints. MobiLim server maintains a database that holds 3 tables:

3.2.1 Resource information table

This table stores the information about the resources. Resource information table has various fields which includes resource name, resource provider name, resource provider IP address, resource price, and number of instances or copies available of a particular resource.

3.2.2 Customer information table

This table stores the information about the requests arrived for license. Fields of this table are resource name, resource provider name, current IP address of the machine on which the user working, the customer accounts ID and usage time.

3.2.3 Granted license table

This table stores the information about the license requests which are granted by the MobiLim server. Fields of this table are resource price, resource name, resource provider name, current IP address of the machine on which the user working.

3.3 User/ customer

User is a customer who wants to access the resources which are owned by some service provider according to his requirement. User is at the client side of MobiLim who accesses the services offered by service provider when needed and pay for that according to pricing policies of the MobiLim server. User, who is a customer, has access to the client side of the MobiLim. User can choose the service and depending on the selected service, the request is redirected to the MobiLim server which handles it.

A request agent is generated by the client program at the users site which carries license request token for a given set of input data which consist of user requirements like name of service, budget, time for which he wants to use the service. The license request token is a file that can be transferred together with the input data to the compute site. We call this agent as license request agent who has all information that the license verifier needs in order to check the identity of user and validity of the software license. The lifecycle of license request agent is explained in details below.

3.4 License request agent lifecycle

In order to clarify the concept of license agent we outline the lifecycle of license agent in figure 3.2. An agent consists of encrypted license terms specification and input requirement of the user which are software-specific. The encryption is done by passphrase encryption algorithm [27]. When a user submits his requirements, a request agent is generated to encapsulate the submitted requirement by the MobiLim client and it migrates to the MobiLim server. After arrival of request agent at MobiLim server, it decrypts the request agent and extracts the requirements from the request agent. Negotiation takes place over the customer budget and resource price. Various checks are

performed depending upon the requirement of customer, if the negotiation is successful and it is possible to execute the job within customers deadline then a license is granted. MobiLim server licenses the job which enables it to execute at the service provider site. Before the job startup the license is evaluated again.

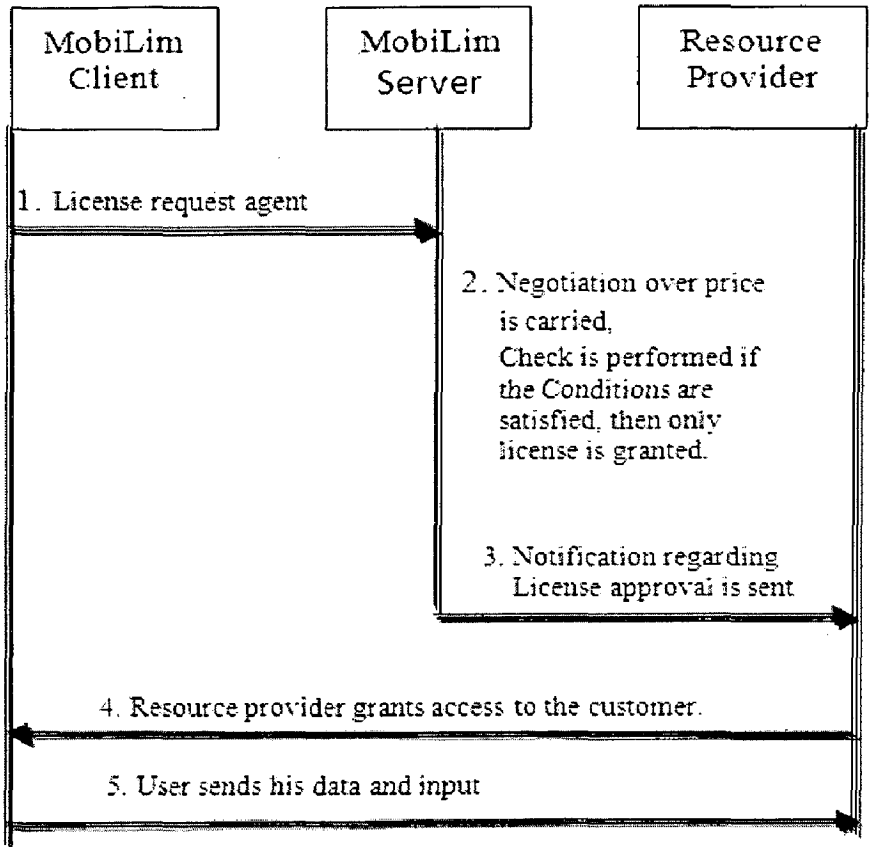

Fig. 3.2 License Agent Migration in MobiLim

User specifies his requirement called as license terms which are MobiLim server specific. It includes name of the resource, time for which user wants to access it. After submitting the requirements, the MobiLim client side program encrypts the requirement which includes the current IP address of the machine on which the user is working and license term specifications. License terms specifications are used by the ISV to decide the type of license to be issued to the user for this particular job. MobiLim client generates one agent which encapsulates the request token and migrates to MobiLim server. At the server site, these encrypted token is decrypted and server extracts the license terms specifications which includes customer budget and the user's identity from the license agent. The

MobiLim server which is implemented by the ISV uses this information to enforce the ISV's business policies.

When a request for the license arrives at MobiLim server, it checks the identity of user and compares it with the customer database. MobiLim server negotiates over the customer price and price set by the resource provider and if negotiation is successful then only MobiLim server grants a license to the customer else a notification is sent to the customer regarding unavailability of the resource. Depending upon the requirements of the user, he is charged and the billing record is stored in the database. After issuing the license to the user, MobiLim server uses its own passphrase to sign the request which is then redirected to service provider site. This signed request agent is act as a license which enables resource provider to identify the customer on receiving input data from customer. After arrival of request agent, service provider verifies whether the grated license is valid or not by decrypting the notification. If the license is valid, service provider allocates the resources according to specifications of request and notifies customer to send his input and data with the help of license agent. The MobiLim servers signed request is considered as a valid license.

3.5 Cryptographic Algorithms

Encryption is the process of taking data (called cleartext) and a short string (a key), and producing data (ciphertext) meaningless to a third-party who does not know the key. Decryption is the inverse process: that of taking ciphertext and a short key string, and producing cleartext.

3.5.1 Passphrase Encryption Algorithm

In the design of MobiLim, passphrase algorithm [27] is used to make MobiLim secure and robust. In this section, the description of passphrase cryptographic technique is explained in details that we have used in the MobiLim. This encryption algorithm uses the DES mechanism [29] with a key generated by MD5 [30] hashing the passphrase, combined with a salt string (random number, known as a *salt*, used to create the key.). DES is the archetypal block cipher — an algorithm that takes a string of plaintext bits and transforms it through a series of complicated operations into another cipher text bit string

of the same length. The MD5 Message-Digest Algorithm is a widely used cryptographic hash function with a 128-bit (16-byte) hash value. MD5 has been employed in a wide variety of security applications, and is also commonly used to check data integrity. DES allows the input string to be repeatedly MD5-hashed a number of times. The 16-byte MD5 string is then split into two halves, an 8-byte DES key and an 8-byte initial seed for DES.

3.5.1.1 Salt

A salt in password-based cryptography has traditionally served the purpose of producing a large set of keys corresponding to a given phrase, among which one is selected at random according to the salt. This has two benefits:

- It is difficult for an opponent to precompute all the keys corresponding to a dictionary of passwords, or even the most likely keys. If the salt is 64 bits long, for instance, there will be as many as 2^64 keys for each password. An opponent is thus limited to searching for passwords after a password-based operation has been performed and the salt is known.
- It is unlikely that the same key will be selected twice. Again, if the salt is 64 bits long, the chance of "collision" between keys does not become significant until about 2^32 keys have been produced, according to the Birthday Paradox. This addresses some of the concerns about interactions between multiple uses of the same key, which may apply for some encryption and authentication techniques.

3.5.1.2 Md5

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit little endian integers); the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with a 64-bit little endian integer representing the length of the original message, in bits. It is conjectured that it is

computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest.

3.5.1.3 DES

DES is a ***block cipher***--means it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size [29]. Thus DES results in a ***permutation*** among the 2^64 (read this as: "2 to the 64th power") possible arrangements of 64 bits, each of which may be either 0 or 1. DES is the archetypal block cipher — an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.

3.6 Negotiation Strategies

Two categories of market based models that are used for cloud resource management are commodities market models and auction models. In commodity market model, service providers specify their resource price and charge users according to the amount of resource they consume. But in other model, each provider and consumer acts independently and they agree privately on the selling price. In this model resources are considered as service provider agent and users are considered as consumer agent as shown in figure 3.3. All provider agents submit their minimum price for their resource (known as ask) and all consumer agents also submit their budget i.e. maximum price for the requested resource to the resource broker. The job of MobiLim server is to decide which resource to allocate to which user and at what cost using negotiation. The negotiation strategy conducted by MobiLim server is inspired from [31] which are explained below:

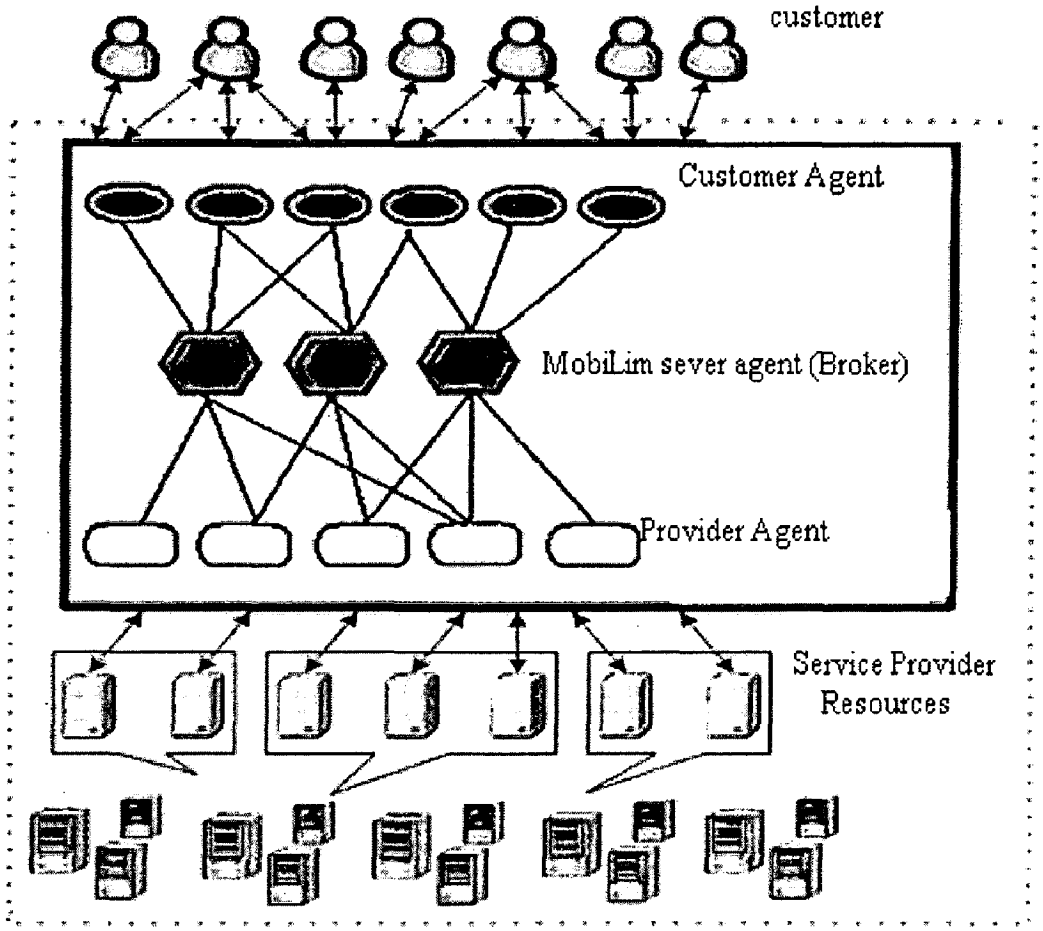

Fig. 3.3 Negotiation Model of MobiLim.

MobiLim server maintains a list of various resources which belongs to some resource provider and is ready to rent them at some price. Resources which are rented on the basis of time are characterized by four-tuples R_i= (r_i, mc_i, cr_i, te_i), where for resource R_i, r_i is reserve price below which negotiation can't takes place, mc_i is maximum number of instances or copies of R_i , cr_i is number of instances or copies available of R_i, te_i is time to execute all the current jobs. The jobs i.e. a customer request for resource are also characterized on the basis of their service type. Jobs which are requested on the basis of time are characterized by four-tuples J_j=(b_j , d_j, t_j), where for job J_j, b_j represents the budget allocated to J_j , d_i determines job deadline by which the consumer desires the job to be finished, t_j represents the actual amount of time customer wants to access the utility. The cost of execution job must not exceed its allocated budget. The customer price is generated on the basis of two conditions which are explained below. The following

equations are proposed in order to generate proper value for the negotiation from both customer and service provider.

3.6.1 Determining Customer Price Based on the Number of Remaining Copies of Resource:

Request for a resource is accepted only if the resource can perform the job within its deadline and the minimum price of the resource (in form of $/hour) is less than or equal to the maximum value the customer can pay. For the resources which are rented on the basis of time, this is determined using is calculated by using proposed equation 1.

$$d_j - te_i - t_j \geq 0 \tag{1}$$

Where d_i determines job deadline by which the consumer desires the job to be finished, t_j represents the actual amount of time customer wants to access the utility, te_i is time to execute all the current jobs which are running on the resource.

The customer price is generated on the basis of number of remaining copies. If the resources are in demand, service provider can earn profit by increasing the resource price. The customer price for the resources which are rented on the basis of time is generated by equation 2 which is from [31].

$$customer\ price^{i,j,t}_{remaining\ resources} = \left[r_i + (b_j - r_i) * (1 - \frac{(cr_i^t - 1)}{mc_i})^{1/\alpha}\right] \tag{2}$$

In the equation 2, r_i is reserve price below which negotiation can't takes place, mc_i is maximum number of instances or copies of R_i , cr_i^t is number of instances or copies available of R_i, b_j represents the maximum budget allocated to execute the job. When $\alpha < 1$, the equation generates low consumer price value until the number of the remaining resources gets close to zero. On the other hand, when $\alpha > 1$ the equation starts with a consumer price value close to b_j.

3.6.2 Determining Customer Price Based on the Deadline:

In this method, if the resources are not available temporarily and user is able to wait for the completion of existing jobs then server generates a customer price value on the basis of deadline of job within which job must be completed. The remaining time to start a job which is rented on the basis of number of instructions is calculated by using equation proposed 5.

$$rt_{i,j}^{t} = [d_j - te_i - t_j] \quad (5)$$

If $rt_{i,j}^{t} < 0$ then job cannot be performed in its deadline. At the time of submitting the job the value of $rt_{i,j}^{t}$ is equal to rt_{max}^{t}. The customer price value is generated by customer agent using proposed equation 6.

$$Customer\ price_{remaining\ time}^{i,t} = \left[r_i + (b_j - r_i) * \left(1 - \frac{rt_i^t}{rt_i^{max}}\right)^{\frac{1}{\beta}}\right] \quad (6)$$

In the equation 6, r_i is reserve price of the resource below which negotiation can't takes place, rt_i^t is remaining time for executing job , rt_i^t job deadline, b_j represents the maximum budget allocated to execute the job.

3.6.3 Calculating the final customer price value

For generating the final customer price value, the values based on remaining time and remaining resource are combined. Final customer price value is obtained by using equation 7.

$$Final\ customer\ price^{i,t} = \lambda * customer\ price_{remaining\ resources}^{i,t} + (1 - \lambda) * customer\ price_{remaining\ time}^{i,t} \quad (7)$$

Where, $0 \leq \lambda \leq 1$.

$Customer\ price^{i,t}_{remaining\ resources}$ is a customer price value generated by customer based on the number of remaining copies of resource and $customer\ price^{i,t}_{remaining\ time}$ is a price value generated by customer based on deadline.

3.6.4 Determining the service provider price value

The price of the resource is generated by provider agent who aims at obtaining more profit. The price of the resource depends on the availability of the resources. If the workload is more and resource is busy then provider agent automatically increases the price of the resource. Initially the price of the resource is set to minimum reserve price but as the demand increases provider agent increases the resource price. The provider agent determines the resource price which is rented on the basis of number of instructions to be executed using proposed equation 8.

$$provider\ price_j^t = \left[r_j + (mp_j - r_j) * \left(\frac{st_j^t}{wl_j^t} \right)^{\frac{1}{\delta}} \right] \quad (8)$$

Where, $0 \leq \delta \leq 1$. wl^t_j is the workload of resource mp_j after the last allocation, r_j is the reserve price below and st^t_j is the current workload or start time of the new job if it is accepted at time t.

3.6.5 Auctioneers role

In each time unit, consumer agents and provider agents determine their resource price values and send them to the auctioneer. The auctioneer generates final price of the resource using equation 10 and then the trade occurs at that price.

$$price = (bid + ask)/2 \quad (10)$$

MobiLim server makes a database entry of successful deal and grants a license to the customer.

Chapter 4

Implementation of the Proposed Framework

Implementation

The proposed system model of MobiLim has been designed for implementation on Windows as well as Linux based mobile devices (laptops). For implementation we have used Java as a programming language and simulation is performed on Intel Dual core processor with 4 GB RAM and 320 GB disk is used. In order to study the efficiency of this approach, we simulated cloud environment in which there are three participants i.e. customer, MobiLim server, and service providers. We have designed three different programs for customers, MobiLim server and service providers. The customer program RmiClient.java is designed to take input requirements from the customer and encrypt those requirements with pass phrase algorithm and send it to MobiLim server. RmiServer.java program is designed to perform operations of MobiLim server which includes handling multiple requests from customer, maintaining the resource information of various resource providers, decrypting the request received from user, performing the negotiation over customer price and resource provider. If negotiation is successful and resource provider is capable of accepting customer request then MobiLim server grant a license to requesting customer. MobiLim server maintains a database which consists of three tables' resource information table, customer information table and granted license table. Resource information table maintains information about resources of the service provider, customer information table maintains information about request from each customer and granted license table stores successful transactions. After decrypting a request, MobiLim server searches the appropriate service provider in his database who has the resources demanded by the client and deal is taken place at some mutually agreeable price. Depending on the billing policies of the service provider customer is charged and informed whether the request is successfully granted or not. After negotiation, server forwards the request to that particular service provider and then communication takes place between Service provider and customer.

4.1 Algorithm for *RmiClient.java*

RmiClient.java program collects the user requirements encrypt them with pass phrase algorithm and send it to MobiLim server. The requirements of user are MobiLim specific which consists of following parameters:

- uti_name: Name of the resource i.e. utility.
- uti_time: Time for which user wants access to the utility.
- max_bid: Budget of the user i.e. maximum amount a user can pay to execute his job.
- del_dead: Maximum time for which a user can wait before the resource gets allocated.
- tpser: Type of service. (SAAS, IAAS, PAAS)

These requirements are taken as an input from the user and program adds address of the customer from which the user is accessing MobiLim server. After receiving input requirements from the client, client program encrypts the input requirement using passphrase algorithm. For this purpose, we have implemented a StringEncrypter subclass in the main class RmiClient.

4.1.1 StringEncrypter Subclass

StringEncrypter Subclass handles the encryption mechanism of MobiLim user which uses passphrase algorithm to encrypt the user requirements.

1. Call the StringEncrypter function and pass a string.
2. Obtain the iteration count c for the key derivation function.
3. Define salt which consists of random bits that are used as one of the inputs to a one-way function.
4. Generates a SecretKeyFactory object for the specified secret-key algorithm DES.
5. Returns a Cipher object that implements the DES.
6. Constructs a parameter set for pass phrase based encryption.

7. Initialize cipher with a key and a set of algorithm parameters. The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of opmode.
8. Encrypt the input requirement
9. Send the encrypted requirement to MobiLim server.

4.2 Algorithm for RmiServer.java

RmiServer.java program handles the functioning of the MobiLim server whose job is to negotiate between customer price value and price value set by the service provider as well as to issue new license to new customers. RmiServer.java program consist of a subclass StringEncrypter which handles cryptographic functioning of the MobiLim. For the purpose of negotiation, program consists of two sub modules generate_bid and generate_ask. The customer price value is generated using generate_bid function and price value from service provider side is generated using generate_ask function. When MobiLim server receives a request from customer it decrypts it using passphrase algorithm which works as follow:

1. Call the StringDecrypter function and pass a string.
2. Obtain the salt S for the operation. Salt consists of random bits that are used as one of the inputs to a one-way function
3. Obtain the iteration count c for the key derivation function.
4. Generate a secret key
5. Returns a Cipher object that implements the DES:
6. Constructs a parameter set for pass phrase based encryption.
7. Initializes this cipher with a key and a set of algorithm parameters. The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of opmode.
8. Call decrypt(String str).
9. Extract the input requirements.

After decrypting the user requirements, server extracts following information:

- uti_name: Name of the resource i.e. utility.
- cust_addr: IP address of the customer machine.
- uti_time: Time for which user wants access to the utility.
- max_bid: Budget of the user i.e. maximum amount a user can pay to execute his job.
- del_dead: Maximum time for which a user can wait before the resource gets allocated.
- tpser: Type of service. (SAAS, IAAS, PAAS)

MobiLim server stores this information into request_info table of database ISVDatabase. To perform this operation we have written a SQL insert query which performs the insertion operation in the database table. After storing request information in the table, MobiLim searches in the respro_info table of database ISVDatabase for checking whether requested resource is available or not and is it possible to execute the job within customer budget. For performing this operation, we have written a SQL select query which selects the resource provider having requested resource and cost of his resource is less than or equal to customers budget. Actual negotiation starts over providers specified resource price and customers budget, if the resource is available. MobiLim server extracts the resource provider information from respro_info table of the desired resource provider. The resource provider includes following information:

- resprov_addr: Address of resource provider
- min_ask: Minimum price set by Resource provider up to which resource provider is willing to sell his resource.
- copy_remng: the remaining number of instances remaining after aloocating them to customer. Initially copy_remng is equal to the value of maxm_copy.
- maxm_copy: Maximum number of instances of resource available at the resource provider.
- time_2_cur: Total time required to execute all current jobs. Jobs are queued if the value of time_2_cur value is zero.

All this information is collected in local variables and sends to generate_bid function which generates a final customer price for the demanded resource on the basis of two parameters which are:

- On the basis of remaining resources.
- On the basis of remaining time for executing job (this parameter is considered only when the request id queued due to temporary unavailability of the requested resource).

After generating a customer price by customer agent, the generate_ask function is called which generates resource price on behalf of resource provider. Based on the values generated by provider agent and customer agent, deal takes place and license is granted to customer. License allows customer to access the resource on the resource provider site. After issuing a license to customer, an entry is made which consists of following parameters:

- cust_addr: IP address of the customer.
- respro_name: Name of the requested resource provider.
- res_name: Name of the requested resource.
- final_price: Final price at which deal occurred.
- Req_DateTime: Time and date at which request arrived at MobiLim server.

After granting a license to the customer, MobiLim server will send a notification about customer request approval along with encrypted information related to customer and deal which includes:

- uti_name: Name of the resource i.e. utility.
- cust_addr: IP address of the customer.
- uti_time: Time for which user wants access to the utility.
- del_dead: Maximum time for which a user can wait before the resource gets allocated.
- final_price: Final price at which deal occurred.

4.3 Algorithm for ServiceProvider.java

Resource provider recover's above information by decrypting the notification sent MobiLim server with pass phrase algorithm and stores this information into his Respro_dbase table of database. Resource provider allocates the resources according to specifications of the request and sends a link to customer, which indicates that a resource has been allocated to the customer and customer can start utilizing it. Decryption algorithm work as follow:

1. Call the StringDecrypter function and pass a string.
2. Obtain the salt S for the operation. Salt consists of random bits that are used as one of the inputs to a one-way function
3. Obtain the iteration count c for the key derivation function.
4. Generate a secret key
5. Returns a Cipher object that implements the DES.
6. Constructs a parameter set for pass phrase based encryption.
7. Initializes this cipher with a key and a set of algorithm parameters. The cipher is initialized for one of the following four operations: encryption, decryption, key wrapping or key unwrapping, depending on the value of opmode.
8. Call decrypt(String str).
9. Extract the input requirements.

Chapter 5

Results and Discussions

SIMULATION AND EXPERIMENTAL RESULTS

The entities in cloud environment are users (resource consumers) and resource owners. Users have jobs for execution and are willing to pay for it. Also resource owners have computational resources and are willing to rent them for profit. We use resource consumer agents that work on behalf of the users and resource provider agents that work on behalf of resource owners. The consumer agents and provider agents are two intelligent entities having their own specific objectives. They interact with each other in form of a double auction protocol for obtaining their objectives.

As mentioned in the previous Section, consumer agents aim at executing jobs within their corresponding deadlines and with the minimum cost. Also the allocated budget for each job determines the maximum cost that a user is willing to pay for executing it. On the other hand, the provider agents aim at obtaining more profit. For this purpose, they try to sell their resources at higher prices. In MobiLim, consumers and providers prices are generated autonomously based on the basis remaining resources and remaining time. In this simulated system, consumers and providers modeled as two kinds of agents. Also we assume that there are three types of resources with the specifications presented in Table 5.1

Table 5.1: Resource types and specification

Resource provider name	Resource Name	Minimum price ($/hour)	Number of copies available	Maximum number of copies available
IBM	XEON Server	12	20	20
rackspace	Server	10	20	20
GOOGLE	GOOGLE DOCS	5	50	50

After submitting the request to the MobiLim server, server decrypts the request and negotiates between customer budget and provider price. Customer request is characterized by four tuples which are Resource provider name, Resource Name, Budget, Deadline. We have submitted jobs one by one and recorded the price at which deal took place for the corresponding resource. Consumer agent generates customer price on the basis of remaining resources and remaining time to start executing job.

5.1 Determining Customer Price Value Based on the Number of Remaining Resources

A customer agent generates customer price value for the requested resource based on the remaining resource. In this method the consumer price value increases, as the remaining copies of resources decreases. When there is more demand to the resources and resources are less, then the customer price value for the requested resource automatically increases. MobiLim server can control the values generated by customer agent by changing the value α. Figure 6.1 shows the different curves at α = 2, 3, 4, 5. In this method, the value of resource depends on the availability of resource.

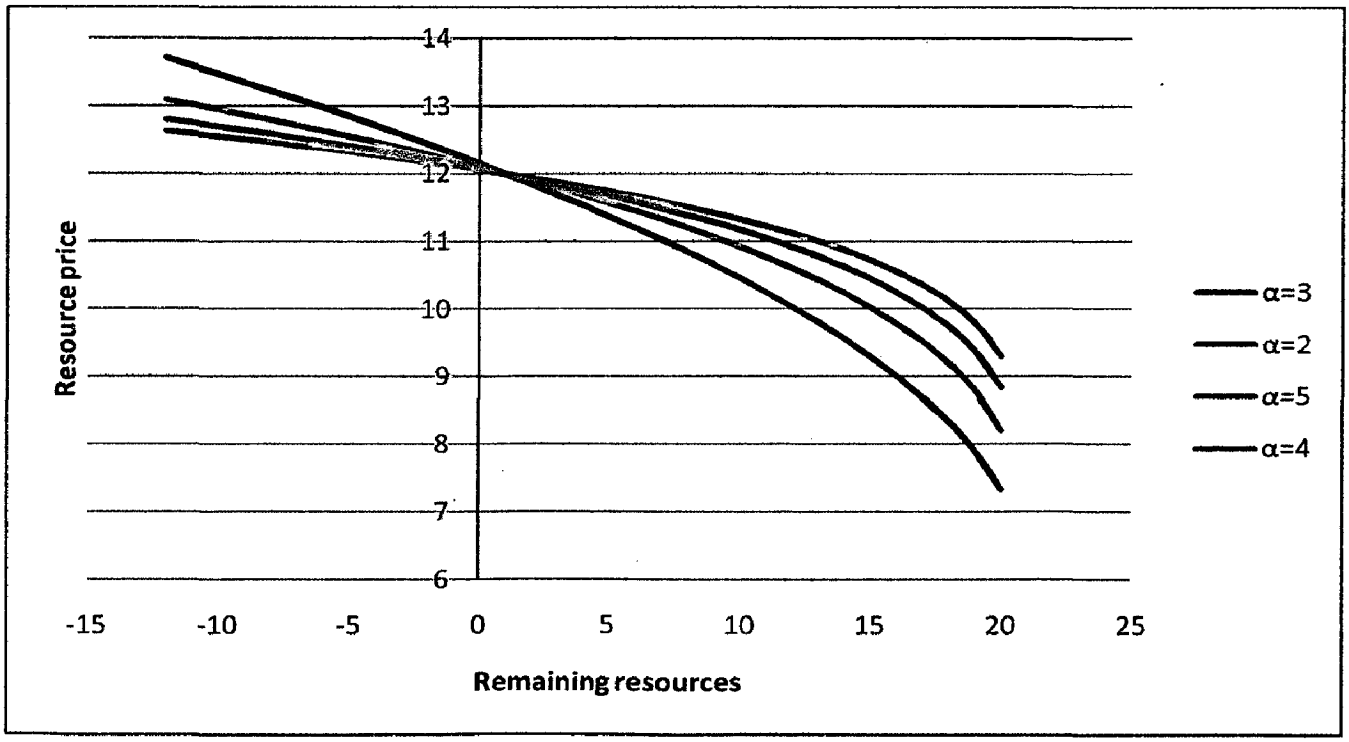

Fig. 6.1 Customer Resource Value Based on Workload

5.2 Determining Customer Price Value Based on the Remaining Time.

If the resource is not available and customer is ready to wait for some time then consumer agent generates customer price value for the requested resource on the basis of remaining time to start executing job. Customer agent can control the customer price values by using the value of β. Figure 6.2 shows the different curves with β = 1, 2, 3, 4, 5. In this method, the consumer price value increases as the remaining time to start executing job decreases.

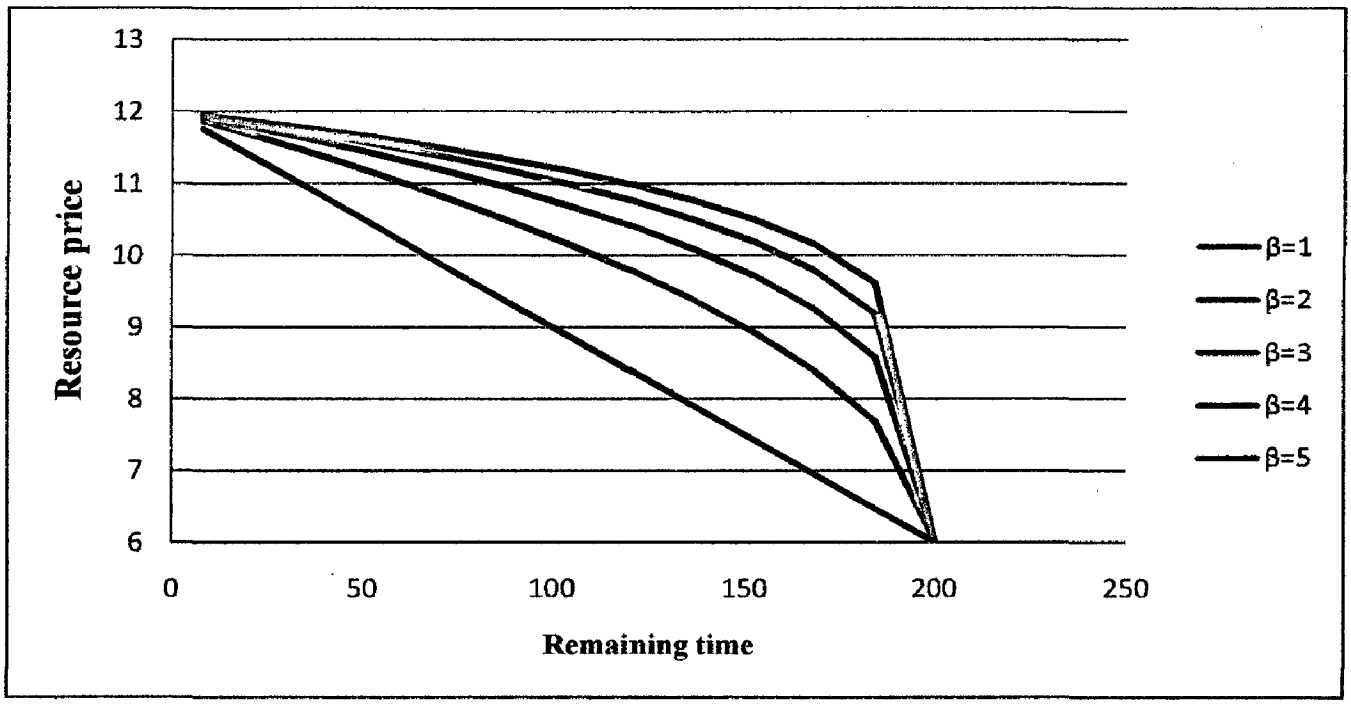

Fig. 6.2 Customer Resource Value Based on Remaining Time.

5.3 Determining Resource Provider Price Value Based on the Remaining Time.

The provider agent aims at obtaining more profit. For this purpose, it tries to sell its resource at a higher price. We assume that at the starting, the workload of resource is zero and the provider sets the price to minimum reserve price for accepting a job. After accepting a job it updates its workload (start time for a new job) and sets its price to the

maximum price, *mp*. Gradually, the workload of the resource is decreased and gets close to zero. By decreasing the workload, the provider decreases its resource price and in the case the workload is equal to zero, it sets the price to the reserve price. Provider agent generates the resource price from the service provider side. MobiLim server can control the provider value with the help of δ. Figure 6.3 shows the different curves with δ = .2, 1, 5. In this method, resource price increases, as the workload increases.

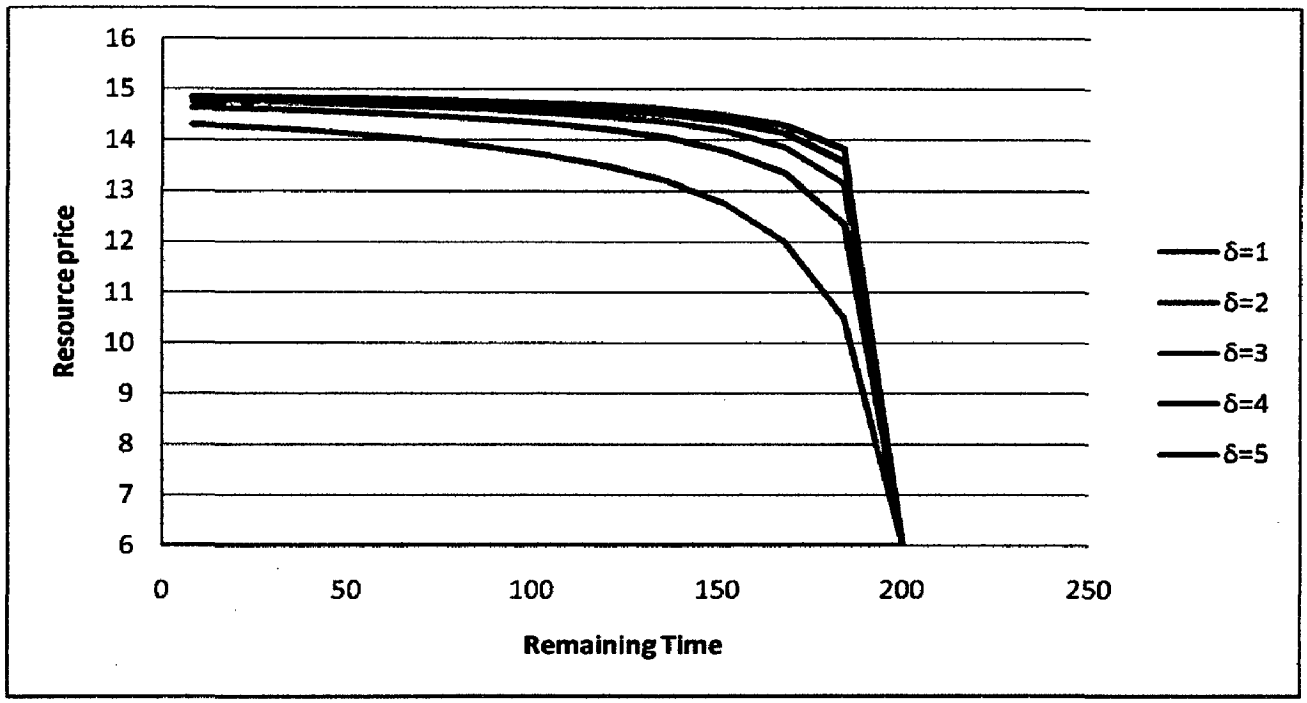

Fig. 6.3 Provider Resource Value Based on Remaining Time.

5.4 Determining Final Price Value Using Customer Price and provider price.

Final price value is an average of service provider price value and customer price value. Cust_price line indicates customer price, sp_price indicates resource provider price and final_price indicates the price at which deal occurred. Customer is put into a queue if the resource is not temporarily available and customer could wait for the resource until it gets freed. The final price value is derived from customer price value and resource provider price value. The graph is plotted between customer price value, resource provider price value and final price value in figure 6.4.

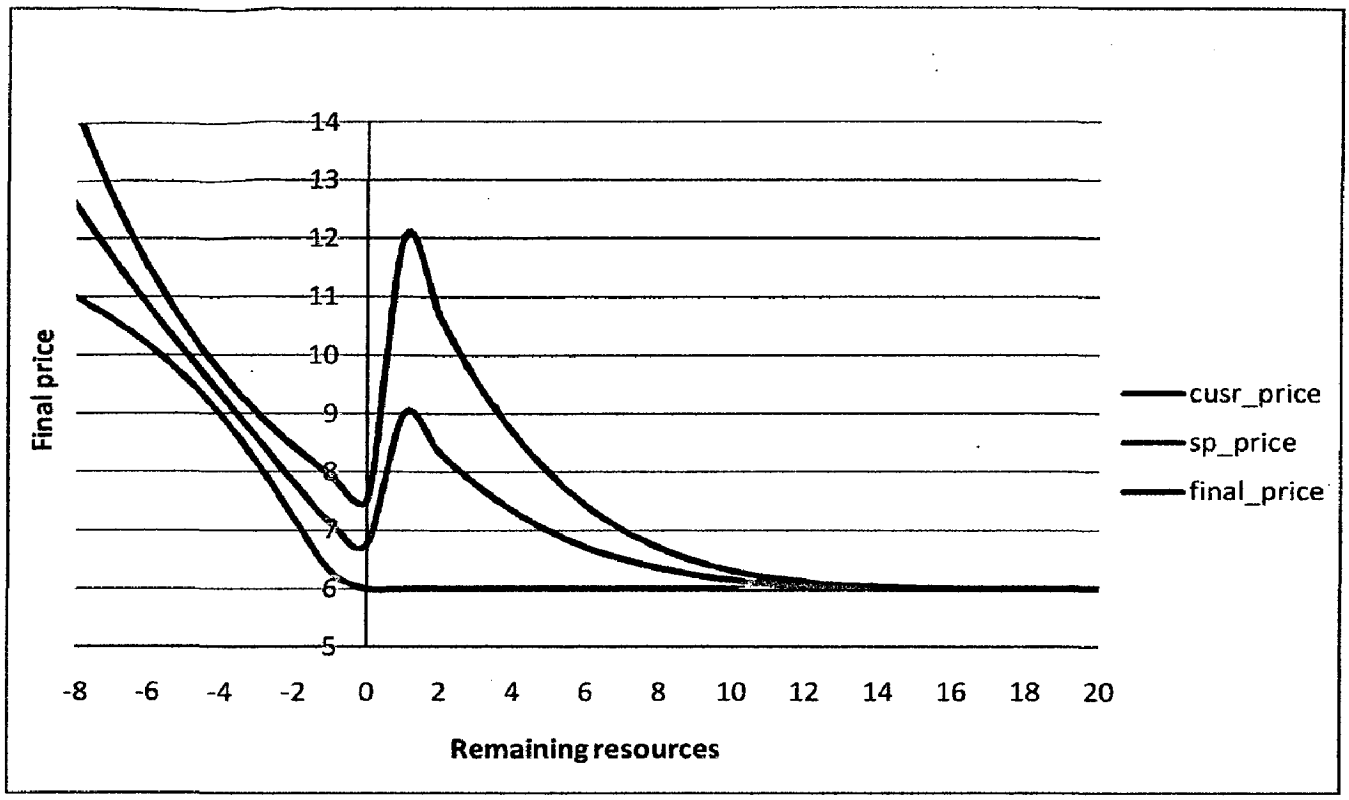

Fig. 6.4 Final Resource Value

If the negotiation is successful then a license is granted to the customer. Customer is charged for the requested resource according to final price generated by MobiLim server.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this report, we have proposed and implemented MobiLim, an integrated license management and economic resource allocation framework for cloud computing. MobiLim ensures security for both the customer and resource provider. MobiLim provides a licensing solution to manage the increased complexity of software licensing in the cloud computing and economic resource allocation for both resource owners and resource consumers. Our proposed license management scheme satisfies the requirements which are necessary for execution of licensing mechanism in distributed environment like cloud computing. MobiLim supports requirements like usage based licensing, access control, Billing and accounting and economic resource allocation.

In our proposed solution, we have considered three participants: users/customer, resource providers and ISVs. User is a resource consumer who needs resources and is willing to pay for them. Resource provider is a owner of the resources and willing to rent them. The independent software vendor job is to negotiate between customer and resource provider and license the customer to enable him to access the resources of the resource provider. This work is concentrated on a particular class of negotiation. In this context, customer agent who represents a customer requires a service to be performed within the budget of customer. Service provider agent represents service provider and try to sell the resources at maximum profit. Negotiation involves determining a mutually agreeable contract under certain terms and conditions.

For economic allocation resource, we developed two agents: provider agents which bid value based on its workload on behalf of the resource owner and consumer agents which generate bid value based on two constraints remaining time for bidding and remaining resources for bidding on behalf of the resource customer. An important feature of our license management architecture is the accounting and billing part, which allows a flexible license cost model.

6.2 Future Work

In the future the improvements can be done in the following areas:

- Implementation of licenses scheduling depending on the priority of the jobs. Priority will depend on the quality of service demanded by the customer.
- Implementation of some new pay per usage policies other than time.

REFERENCES

[1] Buyya, R., Chee Shin Yeo, Venugopal, S., "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," 10th IEEE International Conference on High Performance Computing and Communications, 2008. HPCC '08., pp.5-13, 2008

[2] Pankaj B. Thorat, Anil K. Sarje, "Programming Models for Cloud Applications and Systems: Requirements and Approaches," In the Proceedings of International Conference on Computing, ISBN no. 978-81-920305-1-7, pp 1-6, 2010.

[3] Foster, I.; Yong Zhao; Raicu, I.; Lu, S.; , "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, 2008. GCE '08 , pp.1-10, 2008

[4] Lizhe W., Jie, A., Kunze, M., Castellanos, A.C., "Kramer, D. Karl, et. al.: Scientific Cloud Computing: Early Definition and Experience," In the 10th IEEE International Conference on High Performance Computing and Communications, 2008. HPCC '08, pp. 825-830, 2008.

[5] "google trends", [URL] http://www.google.com/trends, accessed on May. 2011.

[6] Armbrust, M., Fox, A., Griffith, R, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California Berkeley, Tech. Rep. UCB/EECS 2009-28, 2009.

[7] P.Faratin, C.Sierra, and N.R.Jennings, "Negotiation decision functions for autonomous agents," International Journal of Robotics and Autonomous Systems, vol.24, pp.3–4, 1998.

[8] Paletta, M.; Herrero, P.; "A MAS-Based Negotiation Mechanism to Deal with Service Collaboration in Cloud Computing," Intelligent Networking and Collaborative Systems, 2009. INCOS '09. International Conference on , vol., no., pp.147-153, 2009.

[9] Dikaiakos, M.D., Katsaros, D., Mehra, P., Pallis, G., Vakali, A., "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," Internet Computing, IEEE , vol.13, no.5, pp.10-13,. 2009.

[10] Rimal, B.P.; Eunmi Choi; Lumb, I., "A Taxonomy and Survey of Cloud Computing Systems," INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on , vol., no., pp.44-51, 2009.

[11] "Window azure" [URL] http://www.microsoft.com/windowsazure/, accessed on May 2011.

[12] "Amazon Simple Storage Service (Amazon *S3*)", [URL] http://aws.amazon.com/s3/, accessed on May 2011.

[13] "Salesforce a CRM application", [URL] www.salesforce.com, accessed on May 2011.

[14] "Amazon Elastic Compute Cloud", [URL].http://aws.amazon.com/ec2/, accessed on May. 2011.

[15] "IBM Blue Cloud project" [URL] http://www.03.ibm.com/press/us/en/pressrelease/22613.wss/, accessed on May 2011.

[16] "Eucalyptus Project" [URL] http://eucalyptus.cs.ucsb.edu/, accessed on May 2011.

[17] "Enomolism Project", [URL] http:// www.enomaly.com /, accessed on May 2011.

[18] "Application Virtualization and Software Licensing: Best Practices for Software Vendors," [URL] http://www.softwaremag.com/linkservid/, accessed on May 2011.

[19] "Server-Based Licensing: How it Affects Your ROI," [URL] www.logixml.com/content, accessed on May 2011.

[20] M. Dalheimer, F. Pfreundt, "GenLM: License Management for Grid and Cloud Computing Environments," 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.132-139, 2009.

[21] H. Raiffa, "The art and science of negotiation. Cambridge", Mass: Belknap Press of Harvard University Press, 1982.

[22] P.Faratin, C.Sierra, and N.R.Jennings, "Negotiation decision functions for autonomous agents," International Journal of Robotics and Autonomous Systems, vol.24, pp.3–4, 1998.

[23] H. P. Nipur, K. Garg, "A Fault Tolerant Comparison Internet Shopping System: BestDeal by using Mobile Agent", International Conference on Information Management and Engineering, pp. 541-544, 2009.

[24] S.Venugopal, X.Chu, and R.Buyya, "A negotiation mechanism for advance resource reservations using the alternate offers protocol," 16th International Workshop on in Quality of Service (IWQoS 2008), pp.40–49, June 2008.

[25] N. Gatti, F. D. Giunta, and S. Marino, "Alternating-offers bargaining with one-sided uncertain deadlines: an efficient algorithm" Artificial Intelligence, vol. 172, pp. 1119–1157, 2008.

[26] B. An, N. Gatti, and V. Lesser, "Bilateral bargaining with one-sided two-type uncertainty" In Proc. of the 9th IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 403–410, Sep. 2009.

[27] "RFC 4226: HOTP: An HMAC-Based One-Time Password Algorithm," http://www.ietf.org/rfc/rfc4226.txt, accessed on May 2011.

[28] "RFC: 2405 The ESP DES-CBC Cipher Algorithm", [URL] http://www.ietf.org/rfc/rfc2405.txt, accessed on May 2011.

[29] "RFC 1321: The MD5 Message-Digest Algorithm", [URL] http://www.ietf.org/rfc/rfc1321.txt, accessed on May 2011.

[30] Izakian, H.; Ladani, B.T.; Zamanifar, K.; Abraham, A.; Snasel, V.; , "A continuous double auction method for resource allocation in computational grids," IEEE Symposium on Computational Intelligence in Scheduling, pp.29-35, 2009.