5-2013

A Secure and Fair Resource Sharing Model for Community Clouds

Santhosh S. Anand
University of Arkansas, Fayetteville

A SECURE AND FAIR RESOURCE SHARING MODEL FOR COMMUNITY CLOUDS

A SECURE AND FAIR RESOURCE SHARING MODEL FOR COMMUNITY CLOUDS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

By

Santhosh S. Anand
Bangalore University
Bachelor of Engineering in Computer Science, 2001

May 2013
University of Arkansas

ABSTRACT

Cloud computing has gained a lot of importance and has been one of the most discussed segment of today's IT industry. As enterprises explore the idea of using clouds, concerns have emerged related to cloud security and standardization. This thesis explores whether the Community Cloud Deployment Model can provide solutions to some of the concerns associated with cloud computing. A secure framework based on trust negotiations for resource sharing within the community is developed as a means to provide standardization and security while building trust during resource sharing within the community. Additionally, a model for fair sharing of resources is developed which makes the resource availability and usage transparent to the community so that members can make informed decisions about their own resource requirements based on the resource usage and availability within the community. Furthermore, the fair-share model discusses methods that can be employed to address situations when the demand for a resource is higher than the resource availability in the resource pool. Various methods that include reduction in the requested amount of resource, early release of the resources and taxing members have been studied, Based on comparisons of these methods along with the advantages and disadvantages of each model outlined, a hybrid method that only taxes members for unused resources is developed. All these methods have been studied through simulations.

This thesis is approved for recommendation
to the Graduate Council.

Thesis Director:

_____

Dr. Brajendra Panda

Thesis Committee:

_____

Dr. Gordon M. Beavers

_____

Dr. Craig W. Thompson

THESIS DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this thesis when needed for research and/or scholarship.

Agreed _____
SANTHOSH S. ANAND

Refused _____
SANTHOSH S. ANAND

ACKNOWLEDGEMENTS

I would like to express my deepest and sincere gratitude to my advisor, Dr. Brajendra Panda for his continued support, patience and encouragement. His knowledge, direction and experience greatly helped me throughout my Master's program. I would also like to extend my gratitude to my thesis committee members, Dr. Gordon Beavers and Dr. Craig Thompson for their valuable guidance.

I would like to thank my family and all of my friends for their support and blessings. Lastly, I owe my deepest gratitude to Dr. Denise Airola for her continued support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Cloud computing has become the most discussed segment of today's IT industry. It is considered to be a grouping of existing technologies like grid and distributed computing that has changed the way traditional IT services are handled, accessed and paid for. Clouds offer services as virtualized resources that are accessed over the internet giving the end user freedom to choose, expand and pay for the services depending on the type and amount of service used.

The U.S National Institute of Standards and Technology (NIST) defines cloud computing as [1] :

> *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.*

The above definition can be summarized by considering clouds to be a facility (that can be operated by a third party) that offers computing capacity as services that are configurable, convenient and on-demand. The services are virtualized and scalable which allows enterprises to reduce spending on physical infrastructure and only pay for resources they use.

This thesis uses the term "community cloud" which is a type of cloud where infrastructure is shared by members with similar interest or goals. The term "cloud deployment" refers to the five different types of cloud models identified within the cloud computing model.

1.1 Research Motivation and Objectives

As a lot of enterprises have started to explore the idea of using clouds, concerns have started to emerge on cloud security and standardization [2]. These concerns mainly arise due to the loss of control experienced when an organization uses a vendor to provide its IT

infrastructure. Additionally, there are only a few vendors offering services with very limited explanation on how privacy and security is addressed. Furthermore, the cloud deployment models have not been fully explored since some of these deployment models can solve a few of the security concerns that arise from the cloud infrastructure.

The main objective of this thesis is to explore a specific cloud deployment model called Community Clouds that allows sharing of resources between members that have similar goals. Developing a secure resource sharing framework based on trust negotiations and finally, developing methods that allow for fair sharing of resources.

1.2 Research Questions

No single research project can consider all the concerns over cloud security and therefore, this thesis will only try to address the objective stated above. The following research questions guide this thesis:

- Can deployment models improve security?
- Can trust negotiations be used as a secure resource sharing framework?
- Can resource sharing be fair?

1.3 Scope of the Research

Concerns over the cloud infrastructure include access control, privacy, data storage, compliance, audit, portability and long term viability [3]. A broad approach is necessary to arrive at solutions which are beyond the scope of this thesis. Hence, this research only focuses on the community cloud deployment model that brings organizations with a common goal together that share resources amongst themselves over a cloud infrastructure. Two models, one for secure

sharing of resources and a second model that assures fair sharing of resources are developed for community clouds.

1.4 Thesis Structure

The thesis is organized as follows:

Chapter 2 discusses the cloud models, service models and benefits of cloud computing. It also lists some of the concerns and vulnerabilities of the cloud. Chapter 3 introduces community clouds, the need for community clouds and the approach taken to answer the research question described in this chapter. Chapter 4 presents a trust based negotiation framework for secure resource sharing. Chapter 5 introduces a policy language for the resource sharing model. Chapter 6 discusses a fair resource allocation model. Chapter 7 presents results of experiments conducted for the fair resource allocation model. Chapter 8 concludes the thesis and suggests areas for future work.

2. BACKGROUND

As many enterprises have started to explore the idea of using clouds, concerns have started to emerge around cloud security and standardization. This chapter describes cloud computing in detail before outlining some of the issues associated with it. The cloud model is represented in Figure 2.1 [4] and is explained in the following sections.

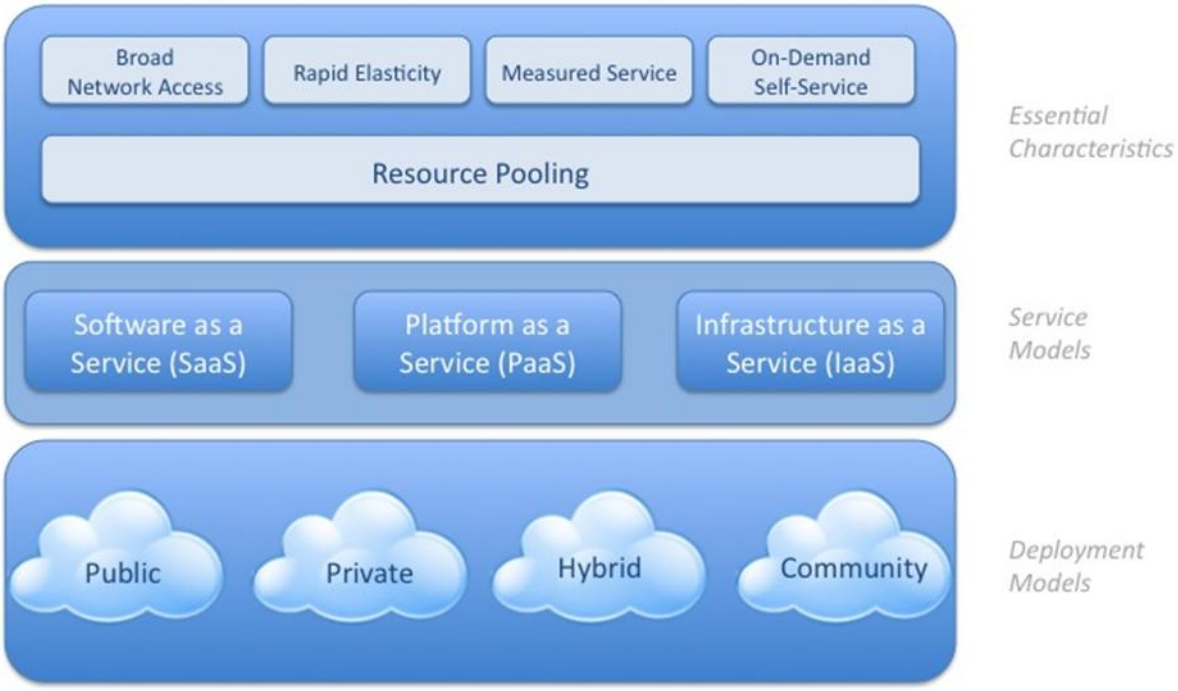


Figure 2.1: The Cloud Computing Model [4]

2.1 Characteristics of Cloud Computing.

The NIST definition of Cloud Computing [1] outlines five essential characteristics:

- **On-demand Services:** The model should be scalable to provide on-demand services without requiring human interaction with each provider.

- **Broad Network Access:** Services should be available over the network and should be accessible through all platforms such as mobile phones, laptops, and PDAs.
- **Pooling Resources:** The provider should be able to pool resources through virtualization to serve consumers with different physical and virtual resource needs.
- **Elasticity:** The capabilities available to the consumer should be scalable with provisions to rapidly release the purchased quantity at any time.
- **Measured Service:** Resource usage should be monitored, controlled and reported to provide transparency to the consumer. This allows consumers to pay for what they used.

2.2 Cloud Service Models

The service models in cloud computing fall into different layers of abstraction that are aimed at different market sections as shown in Figure 2.2 [5]. The NIST definition [1] on clouds describes these as:

- **Infrastructure as a service (IaaS):** This service model provides the consumer with virtualized instances of servers. The consumer does not have control over the underlying cloud infrastructure but can control software, operating systems, applications, and limited control over networking components like fire walls.
- **Platform as a Service (PaaS):** This service model provides the consumer with capabilities to deploy onto a cloud infrastructure applications created with tools or programming languages. The consumer does not manage any cloud infrastructure including networks, operating systems or storage but has control over the applications deployed.

- **Software as a Service (SaaS):** This service model provides consumers applications running on a cloud infrastructure. The consumer does not control any cloud infrastructure including networks, operating systems, storage, with limited configuration settings on the chosen application service.

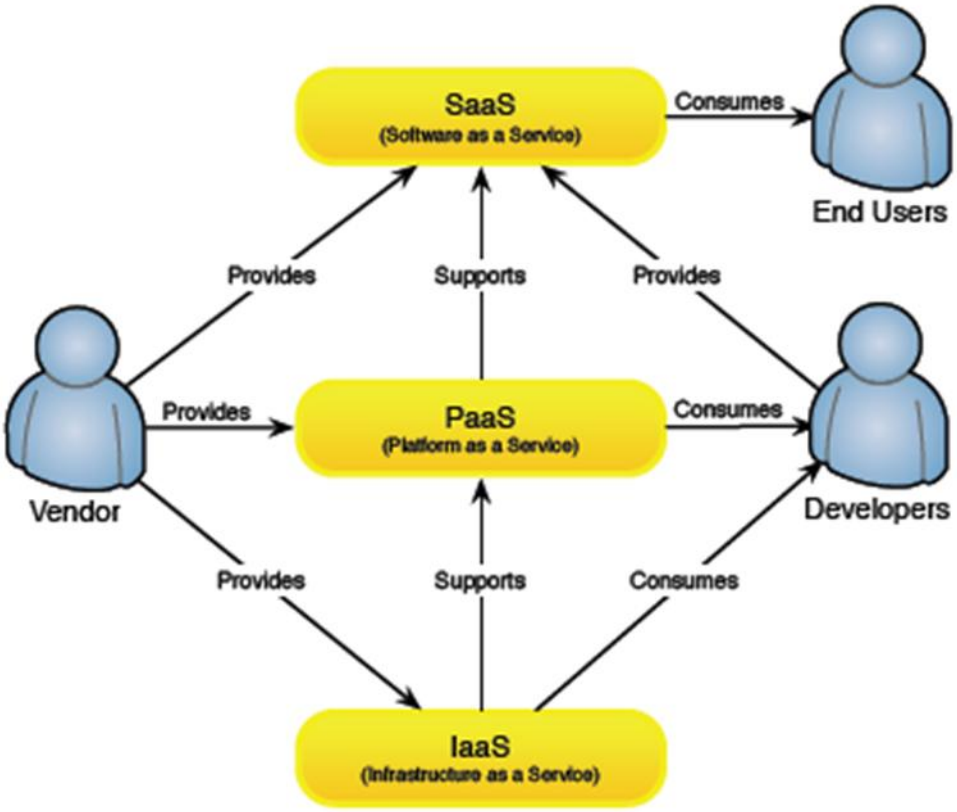


Figure 2.2: Cloud Service Models [5]

2.3 Cloud Deployment Models

The cloud infrastructure is defined by service models and can be classified as Public, Private, Community and Hybrid. Table 2.1 summarizes the deployment models of cloud computing.

Table 2.1: Cloud Computing Deployment Models (from [3]**)**

	Infrastructure Managed By[1]	Infrastructure Owned By[2]	Infrastructure Located[3]	Accessible and Consumed By[4]
Public	Third Party Provider	Third Party Provider	Off-Premise	Untrusted
Private/ Community	Organization Or Third Party Provider	Organization Third Party Provider	On-Premise Off-Premise	Trusted
Hybrid	Both Organization & Third Party Provider	Both Organization & Third Party Provider	Both On-Premise & Off-Premise	Trusted & Untrusted

[1] *Management includes: governance, operations, security, compliance, etc…*
[2] *Infrastructure implies physical infrastructure such as facilities, compute, network & storage equipment*
[3] *Infrastructure Location is both physical and relative to an Organization's management umbrella and speaks to ownership versus control*
[4] *Trusted consumers of service are those who are considered part of an organization's legal/contractual/ policy umbrella including employees, contractors, & business partners. Untrusted consumers are those that may be authorized to consume some/all services but are not logical extensions of the organization.*

- **Private Cloud:** When the cloud infrastructure is solely for a single organization. It may be managed by the organization itself or a third party.
- **Public Cloud:** When the cloud infrastructure is available for a large industry group. The cloud infrastructure is owned by the vendor.
- **Community Clouds:** When the cloud infrastructure is shared by several organizations with common concerns and goals. It may be managed by the organization or a third party.
- **Hybrid Clouds:** When the cloud infrastructure is a composition of two or more cloud types that are bound by standards or technology.

2.4 Security Concerns of Cloud Computing

The cloud model offer a lot of benefits which to be successfully utilized will need secure systems that protect data, privacy and resources. Security for clouds is not very different compared to traditional IT systems. However, since enterprises do not control services and the lack of clarity on security procedures raises new questions and challenges that need to be solved before an enterprise decides to adopt this model. In a recent survey [6] conducted by International Data Corporation (IDC) on challenges associated with cloud model, security was found to be the number one concern for most of the survey respondents. The results from the survey are show in Figure 2.3.

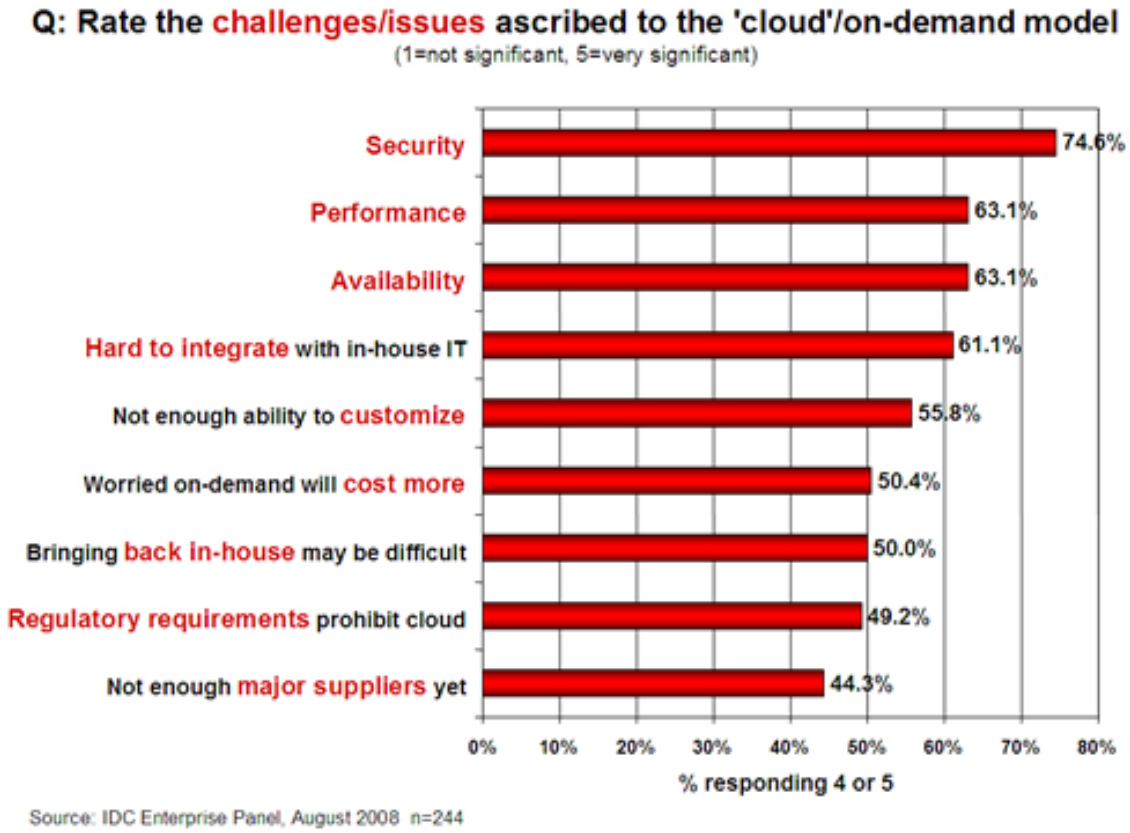


Figure 2.3: IDC Survey Results [6]

Knowledge of cloud architecture, technology, process, services, and deployment models is vital in specifying security models and identifying security concerns in cloud computing. As the providers and end users increase, standardization and security techniques will play an important role in helping organizations to reduce risks involved.

2.5 Vulnerabilities Identified in Cloud Computing

Recent incidents involving clouds have not helped the perception on cloud's security. This section outlines some recent incidents that explain vulnerabilities of cloud computing. These vulnerabilities range from outages to hacking attempts that inconvenienced end users and organizations using the services. Table 2.2 was published by the Open Security Foundation, a non-profit organization providing security information and list of incidents.

Table 2.2: Recent Cloud Incidents (from [7]**)**

Type	Date	Organization	What Happened?
hack	2012-01-21	DreamHost	DreamHost Database Hack Forces Mass Password Reset
outage	2011-04-21	Amazon Web Services	Companies left staggering or totally knocked out because of server problems in the Amazon datacenter
outage	2011-04-21	Sony	Play Station Network outages
outage	2011-03-25	Twitter, Inc.	Twitter Experiences Delays in Delivering to Facebook and SMS

outage	2011-03-25	Heroku	Heroku Users Experience HTTP 503 Errors
outage	2011-03-25	Twitter, Inc.	Twitter Experiences Tweet Delivery Delay
outage	2011-03-25	Heroku	Heroku Shared Database Experienced Hardware Failure
outage	2011-03-25	Heroku	Heroku Users Unable to Provision New Dedicated Databases

2.6 How is Security Different for Cloud Computing?

The security concerns over cloud environments are usually not very different from the traditional IT environment. However, the lack of standardization, technology involved in deploying clouds may present risks. Since infrastructure is leased from a third party, concerns on security, availability, and privacy arise as enterprises have limited control over the IT infrastructure. The specific security concerns are outlined below [3]:

- **Access Control and Management**

  Identity management, access control and privacy can be identified as challenges facing enterprises using clouds and vendors providing services, since resources are shared between several organizations and users.

  Identity of users and operations are important to prevent any malicious transactions. An access control model needs to be in place to prevent any unauthorized access or transactions that could allow users access data unintentionally or for any malicious reasons. The access control mechanisms need to be fine-tuned to employ dynamic,

contextual and credential based access requirements with strong privacy rules. They should also be easily manageable, interoperable with efficient privilege determination.

- **Data Storage**

  The main concerns with storing data on the clouds are integrity, availability and location. Data stored on clouds can include sensitive information such as [8]:

  1. Personally identifiable information: Information on identity, location of individual or other personal information such as credit card numbers, postal code, IP address.
  2. Sensitive Information: Information on race, sexual orientation, health, information related to job performance, personal information and financial information.
  3. Usage Data: Information collected from computers, cell phones, printers. Other behavioral information such as viewing habits for digital content, product usage history, most visited websites history.
  4. Enterprise Data: Information collected or generated through transactions important to business. Critical data on business, performance, sales, and technology should be safe, especially from the vendor who may have an un-fair advantage of accessing this data from enterprises that may be viewed as competition.

  Along with protection of data that is both private and sensitive as identified above, the availability of data as a service to enterprises is also important. Lastly, location of data can create challenges based on jurisdiction. Certain countries and state governments may have propriety laws on privacy, storage and access [9]. Many banks for example require the financial data to be stored in the home country [2].

- **Compliance, Audit and Monitoring**

  International boundaries and laws may affect ownership or control over data and services. Enterprises must be able to assess the risks involved not only over the control or ownership issues that may come under the laws of the host country but should be aware of any legal steps that it can take if there was a security breach. Any obligations to protect data should be observed at all stages of data handling and storage, irrespective of the jurisdiction [8]. Logs of all important negotiations and handling should be made transparent thus providing continuous monitoring [9].

- **Information Recovery, Retention and Destruction**

  Enterprises should be aware of the vendor's practice of backups and data recovery in the event of an accident or hardware failure.

  Retention policy on data, specifically after the end of a contract needs to be well defined. Furthermore, international laws on data retention should be studied if the vendor's services are located in a different country. These policies must be reviewed continuously to incorporate changes that may come about as business expands or other policies are revised. Finally, data destruction procedures and practices need to be studied. It is important to have a policy on data deletion that makes sure that data is deleted when specified and holding back any data by the vendor should be challenged as illegal.

- **Portability, Long-term viability**

  Portability can be an issue with data/services and security models if clients decide to change their vendors or if they wish to move everything in house. As cloud computing gets popular, more companies are providing cloud services. Long term availability of services from newer companies also raises questions.

- **Encryption, Key Management and Incident Response**

  Privacy, security are the main concerns over services. Hence, identifying key encryption techniques and implementing them is of prime importance. Similarly, protocols for Incident detection and response should be outlined effectively.

- **Virtualization**

  Risks involved with virtualization technology such as multi-tenancy, VM isolation, VM co-residence and hyper visor vulnerabilities need to be identified and understood [10].

3. APPROACH

This chapter outlines the approach taken in this thesis along with a brief description of the design used in the models that help answer the research questions stated in Chapter 1.

3.1 Can Deployment Models Improve Security?

Most of the security concerns of cloud computing discussed in the previous chapter arise due to loss of control, lack of standardized security measures and lack of trust when external vendors control services. The notion of a community where members with similar interest or goals coming together make them trusted members of the community. Several organizations form a community cloud when the cloud infrastructure is shared between them. The NIST defines community clouds as [1]:

> *The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.*

Community clouds address these concerns either by standardizing security, privacy and trust based on the community requirements or by sharing the resources of its members with each other, often with members being both consumers and contributors. Some examples where community clouds can be used to share resources over the cloud are: Government Organizations, Research Institutes, Universities and Hospitals. Figure 3.1 shows a university community with members being departments of the university. Each sub tree represents a different branch of the university.

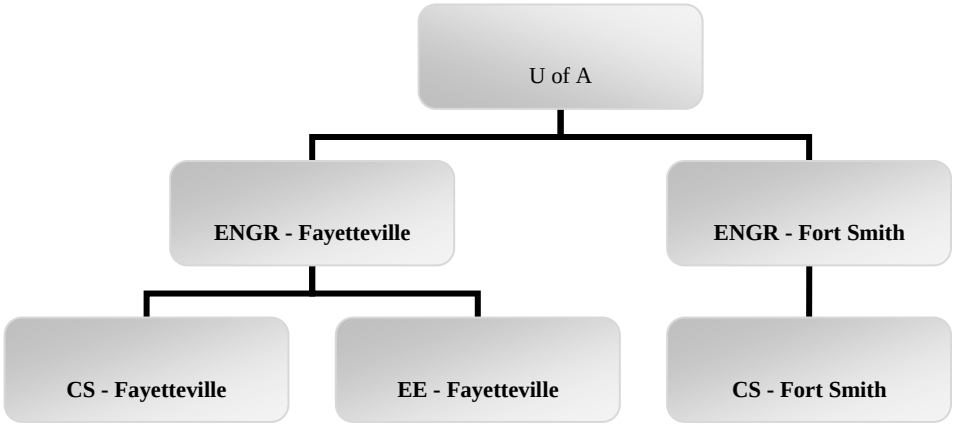


Figure 3.1: A Community within a University Setting

3.1.1 Advantages of Community Clouds

Community clouds not only have benefits of the cloud computing model but can provide additional advantages that are listed below:

- Ease of standardization since all members share a common goal. Hence, Access Control, Data Storage and Privacy are securely handled.
- Reduces redundancy and duplication of resources.
- Resource outage is minimal when members are vendors in the community since resources are not owned by just one organization.

3.1.2 Importance of Community Clouds

Recent research suggests how the digital world can borrow properties from biological ecosystems. These systems are called Digital Ecosystems [11] that can be defined as distributed, adaptive, open, socio-technical systems with properties of self-organization, scalability and

sustainability inspired by natural ecosystems. Some properties like self-organization, self-management, scalability and the ability to solve dynamic problems are being studied as approaches to evolve the digital ecosystem. Among the many deployment models in cloud computing, community clouds have properties closely associated with digital ecosystems [5]. Communities of shared interest and concerns have a common goal and can make security and privacy implementation easier by introducing standards within the community.

Community cloud adoption will become a key success factor considering the volatile global economy, changing requirements, virtual resources which will bring together cooperation among competitors in specific areas where policy compliance and cost reduction becomes vital. Furthermore, organizations and departments that find themselves on common grounds by virtue of them working together for a shared goal like health care, education, research and governmental entities will also benefit by adopting cloud technology as a community.

The notion of pooling resources along with standardized security protocols reduce security concerns while adopting the beneficiary elements of clouds thus reducing redundancy and duplication of both efforts and resources.

3.2 Can Trust Negotiations be used as a Secure Resource Sharing Framework?

Several security models already exist for information and resource sharing between coalitions and organizations. Among the various research models proposed, Trust negotiations is a promising approach where trust is gradually built by the exchange of digital certificates that can be verified. While several trust negotiation models exist, this thesis extends the Trust –*X* frame work defined for peer-to-peer networks [12] for secure resource sharing within the

community. The different elements of this model along with the policy language are explained in Chapters 4 and 5.

3.3 Can Resource Sharing be Fair?

Fair resource sharing is an important need of the community. Although the members may be tied under common goals and interests, members are diverse and will always focus on their individual goals which may result in resource contention. Greedy members may want a larger share of the resource allocated to them while members with malicious intent will want to inconvenience other members they consider rivals. In order to encourage fair resource sharing in the community, a model is developed that makes resource usage and availability transparent to the members while taking necessary corrective measures to avoid situations where resources are depleted from the pool due to unnecessary demand.

3.4 Assumptions

In order to address the three research questions while not broadening the scope of the research, this thesis makes the following assumptions:

- A cloud architecture is assumed to be in place for the community where resources are virtualized and can be grouped together in a resource pool.
- A community cloud can be serviced by members sharing their resources with other members or through an external vendor. This thesis assumes that the members within the community share their own resources.
- Members within the community are organized in to a tree hierarchy to better express relationship among them.

- Each member is assumed to have a local security model already in place.
- A payment model is assumed to exist within the community that handles resource pricing and accounting functions.

4. SECURE FRAMEWORK FOR SHARING RESOURCES

The community cloud hosts resources through a shared resource pool with the ability to allocate these resources to other members on demand. The model provides a secure frame work for trusted communication of information on resources, polices and restrictions for resource sharing. The idea behind the model is for members of the community to regulate how and when to disclose information through digital certificates and policies with other members [13] to successfully complete a negotiation for sharing resources.

The approach of this model is based on trust negotiations [14] [15] [16] [17] where trust is built gradually as information is exchanged and verified. The model is an extension of the Trust – *X* framework, an XML based system [12] that separates negotiations into an introductory stage and disclosure stage. It also supports a novel idea of Trust Tickets that help speed up repeated negotiations. This model differs from the Trust –*X* framework in the following ways:

- Trust –*X* is a Peer-to-Peer framework. This model has been extended to fit a community cloud where information flow is regulated. Furthermore, the process of sharing will involve parent members of the hierarchy when negotiation is between members who are at different levels of the hierarchy.
- Resources in the community model are virtualized and scalable. This model allows resources to be expressed as a quantity.
- The policy language definitions were changed to better represent the community cloud environment. Resource definitions now allow resources to be defined under Saas, PaaS and IaaS following the cloud service models.

- The introductory stage now allows the requestor of the resource to share local policies and regulations. The goal is for both members to reach an agreement in the initial phase of negotiation.

4.1 Key Elements of the Model

This section describes the key elements that make the model. Since the model only describes a framework for trusted negotiations between members and not the system architecture of the community cloud itself, it is assumed that a cloud infrastructure is already in place within the community and each member has the ability to contribute and request resources from a virtualized resource pool.

4.1.1 Resources

Resources shared within the community can be varied that range from a single document to instances of software, servers and platforms. In this model resources refer to either a single resource or a set of resources. Furthermore, the resources can be categorized under SaaS, PaaS or IaaS based on the cloud computing definition for service models. A key element for resource scalability is to specify a measurable parameter that governs the capacity of the resource being shared. The measurable parameters are expressed in time, quantity or a range.

All resources available to be shared are grouped to form a resource pool. Availability of a particular resource in the pool does not guarantee its release to the requesting member. The member requesting the resource has to satisfy rules governing the resource release through a negotiation.

4.1.2 Members

Members are independent organizations or departments within an organization that form the community. They are diverse entities that vary in size, structure and complexities. Members can either be contributors or consumers of resources. The architecture of the community can be represented by a tree with each node representing a member. Figure 3.1 in chapter 3 shows a community formed in a university system. Each member is assumed to already have a local security model in place. Role Based Access Control (RBAC) [18] is a popular access control model that is widely used.

4.1.3 Policy

Policies are strict rules and regulations defined over the resources and members of the community. The policies are stored in a policy database and only disclosed when the negotiation for a resource starts. As described in the previous section it is assumed that a local security model is already in place within each organization with regulations and policies clearly defined. A negotiation is thus a method of gaining trust by the exchange of policies between members. A negotiation moves forward only when the previous policies are satisfied.

4.2 Flow of Information

Members are considered to form a hierarchy as explained before. Within the hierarchy, members who are not immediate children of the member controlling the resource cannot share resources directly but have to route the request through a parent. In Figure 4.1, node *H* and node *I* cannot share a resource directly but can share a resource through E. Similarly if $X = \{B, D, E,$

H, I) and *Y={C, F, G, J}* represent two sub trees. There cannot be any sharing directly between the two but will have to be through A.

The member requesting the resource is called "*requestor*" while the member controlling the requested resource is called the "*controller*" [12].

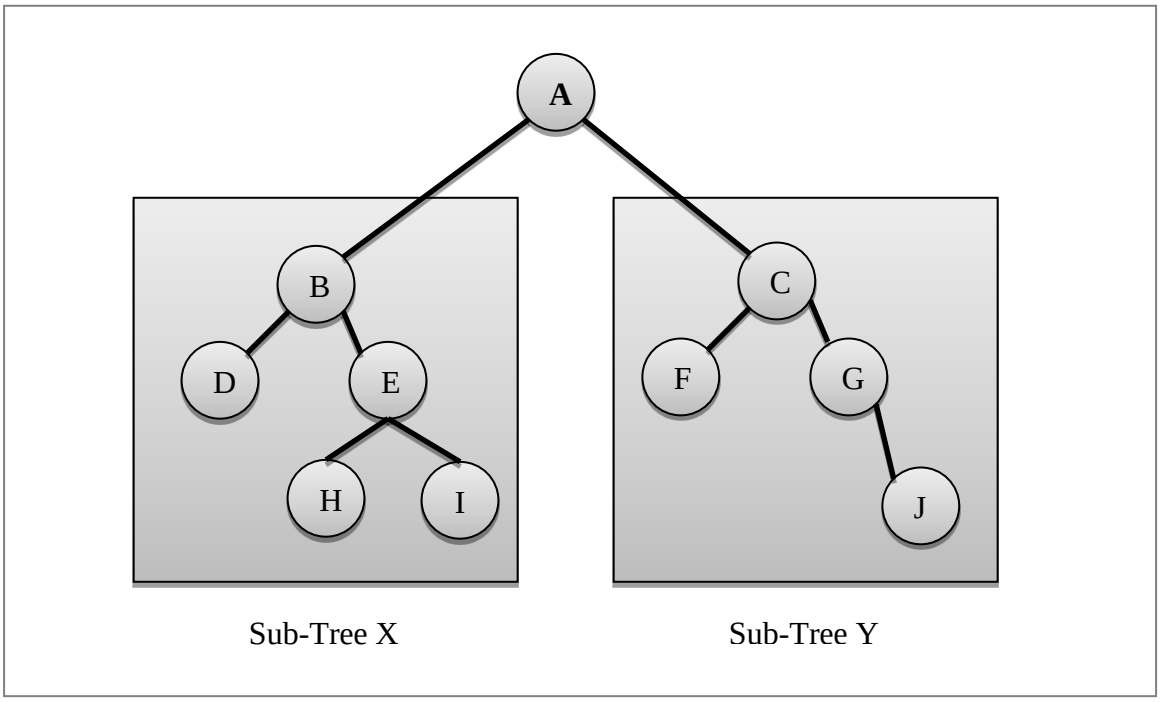


Figure 4.1: Flow of Information within the Community

4.2.1 Type of Resource Sharing

The controller can share a resource with its immediate children or children of its immediate children. Based on how the controller shares a resource, the process can be defined in two ways.

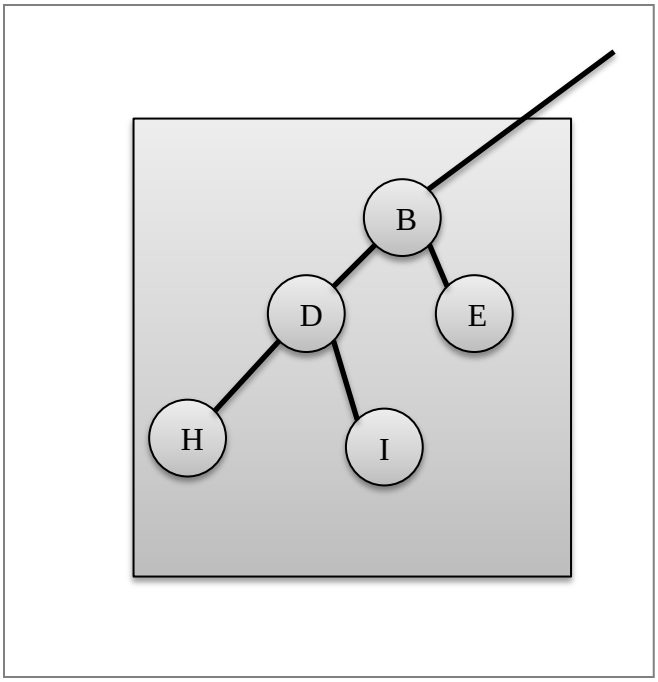


Figure 4.2: Types of Resource Sharing

- **Simple Share:** The process of sharing a resource between two members connected by a single edge. Example: In Figure 4.2 when members *D* and *H* share a resource.
- **Chained Share:** The process of sharing a resource between two members connected by more than one edge. Example: In Figure 4.2 when members *H* and *B* share an object.

4.2.2 Principle of Attenuation of Privilege

After a resource has been shared, the requestor is given a set of privileges over the resource. The requestor can now share the resource with a different member if there are no restrictions specified on further sharing of the resource. However, privileges governing the resource during the original share cannot be altered. The principle of Attenuation of Privileges states that a member cannot give away privileges it does not possess.

Example: In Figure 4.2 if *B* and *D* shared a data base table, with *B* granting *D* only read privileges, *D* can share the table with *H* but does not have any rights to grant any other privileges than read.

4.2.3 Principle of Automatic Allocation of Privileges

During any chained share, all privileges specified on the resource are also automatically granted to members who are parents of the requestor. The principle of Automatic Allocation of Privileges states that in a chained share, members between the requestor and the controller who are parents of the requestor are also automatically granted privileges on the requested resource that the requestor has acquired. This is also true for revocation of resources.

Example: In Figure 4.2 consider a successful negotiation between B and H. Since D is an immediate parent of H, D is also granted all privileges over the requested resource as H.

4.3 Trust Tickets

Collaborations within the community often result in repeated sharing of the same resource between the same members. This is particularly true in the cloud model where resources are scalable. In order to make the negotiations efficient and faster while being secure, Trust Tickets [12] are used to reduce the number of certificates and policies exchanged. These certificates are generated at the end of a successful negotiation which certifies that both members had successfully fulfilled all requirements to acquire the resource. If a member *M1* has a trust certificate with another member *M2* for a resource *R1* and if both *M1* and *M2* wish to share the same resource again, they need not go through the whole process of negotiation but share the trust certificate to acquire the resource again. Trust Tickets are means to prove that members

previously involved in a negotiation with the same resource are trusted entities. Each trust certificate is issued for a valid time period only and after which they have to go through the negotiation process again. The conditions for the validity of the trust ticket should be specified at the time the trust ticket is issued.

4.4 Resource Sharing Between Members

Sharing resources is an important and regular occurrence within the community. The resources shared can be of varied sensitivity that requires strict policies and hence it is very important to have a formal framework that facilitates secure resource sharing within the community. The sharing process between organizations begins when one organization requests a resource from another. This process includes communicating information on the resource, polices and verification of restrictions and other information. These collaborations may not be predictable and may result in security leaks and hence require an effective sharing process. In order to address this issue the model incorporates several approaches. In cases where resources need maximum security, strict policy definitions are used which could include limited privileges or inability to share with others. However, there can be cases where efficiency is crucial or when a requested resource does not have strict security requirements. In these case strategies that speed up the negotiation are adopted. The sharing process can be broken down in to into the introductory phase, certificate exchange and policy evaluation phase. The difference phases of the resource sharing process [12] are outlined by a flow chart in Figure 4.3.

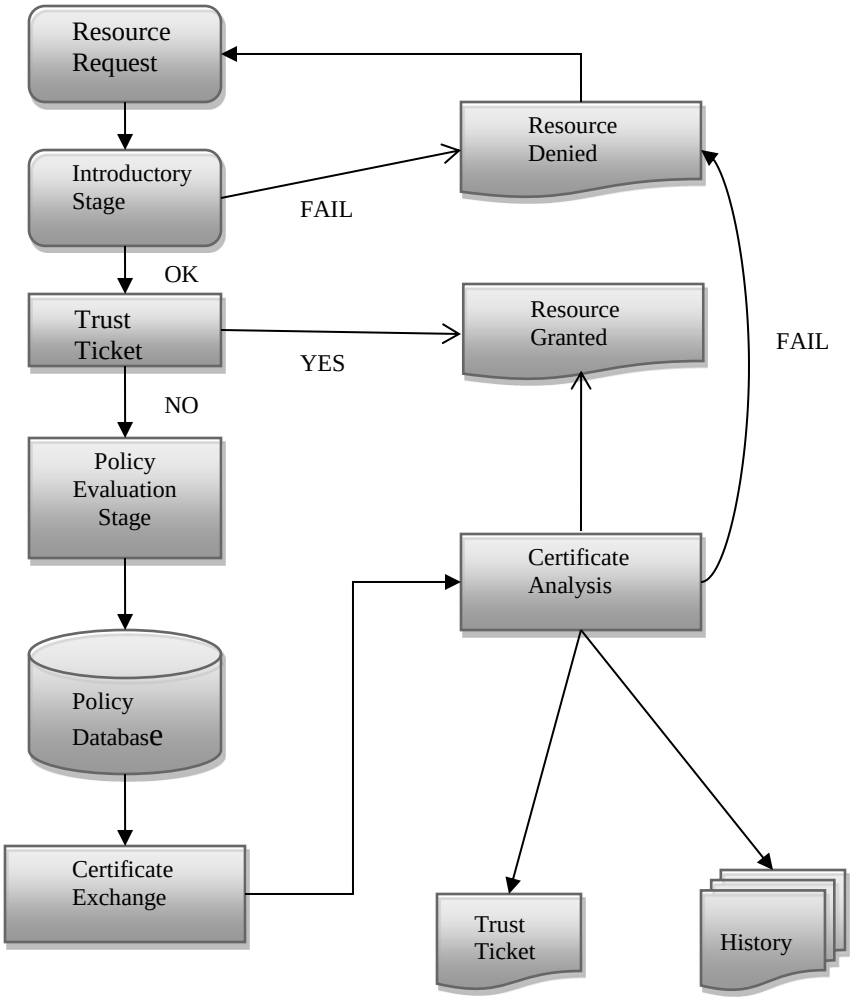


Figure 4.3: The Negotiation Process (from [12])

4.4.1 Introductory Stage

An important goal in the sharing process is not just enforcing strict rules on sensitive resources but also regulating any information on such resources. Consider a resource that can be accessed by US citizens only. Any requests for this resource from non US citizens should be verified at the initial stage before communicating all the other requirements that needs to be satisfied before the resource can be released. Introductory polices can be considered to be

prerequisite conditions that have to be satisfied at the start of a negotiation in order for the negotiation to continue. This helps in establishing an initial level of trust and speeding up subsequent steps in the sharing process. These rules do not guarantee the release of the resource but facilitate in the negotiation process by checking for trust tickets, identity of member and also any special requirements from the requesting member before the rest of the policies are disclosed.

4.4.2 Certificate Exchange and Policy Evaluation

Certificate exchange starts with the introductory phase and carries into the next stage where the controller of the requested resource communicates the required polices for the release of the resource. Policy evaluation is mostly governed by the sensitivity of the resource, history of previous negotiations between the members and trust tickets. If the requestor holds a valid trust ticket, there are very few steps to be executed as the members have already had a successful negotiation over the requested resource in the past. If a member requests for a resource that was declined earlier, polices will need to be satisfied on those phases of the negotiation that failed in the previous negotiation. The policies are disclosed gradually thereby building trust. Only after all the policies have been satisfied the resource is released. Every step of the negotiation is logged into history for future reference.

Consider a scenario where a university network shares resources through a community cloud. Let us consider that the Computer Science Department at location A wants an instance of a data base along with certain research data stored in one of the data tables from the Computer Science Department at location B. The database located at location B is governed by the following policies:

- Only US citizens can access the data.
- Only Computer Science Students can access the data.

After the request for the resource has been made, the Computer Science Department at location B will specify the above policies during the introductory stage. If the Computer Science Department can satisfy the above conditions, the negotiation process moves to the next stage where other conditions need to be satisfied before the resource is released. Figure 4.3 shows the different phases of a negotiation within the community.

4.5 Role of History

History is a service within the model that logs all important information about a negotiation that include member information, resource information and polices that were exchanged and verified. Trust can also be enhanced by checking history of previous negotiations between members. Furthermore log of negotiations provides accountability in cases where members disagree on polices they had initially agreed on. Lastly, the policies disclosed for a resource in various negotiations can be used to better understand and predict the sequence for future negotiations.

5. THE FORMAL FRAMEWORK

This chapter describes formal definitions for the different steps involved in the resource negotiation process that were discussed in the previous sections. An improved policy language was developed which is an extension of [12][16] that ties the different elements of the negotiation. The policy language is illustrated with examples.

5.1 Credentials and Declarations

Every member will disclose certificates during the negotiation that are means of conveying information. The certificates can be credentials or declarations. Declarations are statements used by members during a negotiation whereas credentials are statements that can be certified. Example of a credential can be a university id whereas an expression specifying the amount of a resource is a declaration. The credentials are verifiable and are not forgeable. Credentials and Declarations are represented as C_{name}, D_{name} respectively where *name* is the label of the credential or declaration.

Definition 5.1: A *policy condition* Con_n (n is an integer) on C_{name},D_{name} or a trust ticket is an expression of the form *element op expr*. Where Con_n specifies a condition and *element* can be an attribute of C_{name}, D_{name} or trust ticket. *op* can be a mathematical predicate such as <,>, =, <=, *!* = and *exp* is a resource expression or a constant compatible with the element.

Example:

Con_1 = *(write =”disabled”, year < 03)*

The above example specfies conditions over a data table where write permission is diabled and records before 2003 can be viewed.

Definition 5.2: (Credential): is a generic expression of the form $C_{name}(element\_list)$ where *name* is the label of the credential and *element_list* is a set of elements of the form *element_list="value_term",* where *value_term* is a value or a variable.

Example: $C_{profile}$(*name = "Alice", uid="xyz1234", Member="CS-Fay", role="Programmer"*)

The above credential identifies Alice as a programmer from the Computer Science department with university id xyz1234.

Definition 5.3: (Declaration): is a generic expression of the form $D_{name}(element\_list)$ where *name* is a label for the declaration statement and *element_list* is a set of elements of the form *element_list="value term",* where *value_term* is a value or a variable.

Example: D_{MySQL}*(hostname='cs_server', table='researchdata')*

The above declaration identifies a resource MySQL database with details about the hostname and a specific data table called "researchdata".

5.2 Essential Elements of the Community

In this section formal definition for the basic elements namely the community, members, resources and policies that were described in the previous chapter are explained.

5.2.1 Members

Members are identified by the sub-tree they belong to and their name. In Figure 3.1 CSCE Fayetteville is identified as Fay_{CSCE}

Definition 5.4: A *Community Hierarchy H* is a finite, non-empty set of organizations or nodes, $H = \{r\} \cup H_1 \cup H_2 \cup ... \cup H_n$ with the following properties:

- A designated node of the set, *r* is called the root of the tree.
- The remaining nodes are partitioned into $n \geq 0$ subsets $H_1, H_2, \ldots, H_n$ each of which is a tree.

For convenience, the following notation is used $H = \{r, H_1, H_2, \ldots, H_n\}$ to denote the hierarchy. The hierarchy in Figure 3.1 can be depicted as follows:

$H = \{University\ of\ Arkansas, H_{Fayetteville}, H_{Fort\ Smith}\}$

5.2.2 Resources

A resource *R* in this model can refer to a single resource or a set of resources that can be expressed in measurable parameters like time, quantity or a range. This allows the resources to be easily classified under the cloud service models (SaaS, PaaS, and IaaS). Examples for service requests based on the cloud service models are explained in the example section of this chapter.

Definition 5.5: (Share Term): A share term S_t is an expression of the form $S_t = (Con_n)$ where Con_n defines a range, quantity or a time limit. Time can be expressed as an interval or a

future date. Scalar quantities can be expressed as values for a specific parameter of the resource or as a range of values. A combination of time and scalar quantities can also be expressed with the resource.

Example:

1. Share Term Expressing Time Duration: S_t*(time = "24 April – 24 May")*
2. Share Term Expressing Quantity: S_t*(storage = "40 GB")*
3. Share Term Expressing a Range: S_t *(users = "10 - 15")*

Definition 5.6: (Resource Request): A resource request is a specific type of declaration used by the requestor to request a resource. It is a declaration of the form $R_{request}(R_{name}, element\_list)$ where $R_{name}$is the name of the resource and $element\_list$ is either a share term S_t or is a set of elements of the form *element_list="value term"*, where *value_term* is a value or a variable.

Example: $R_{request}(Black\ Board,\ version = 2.2,\ storage = "10GB")$

The above example is a resource request for an application called black board. The version and amount of storage required with the application is specified by the requestor.

Definition 5.7: (Disclosure Policy Rule) [12]**:**

A disclosure policy rule for a resource *R* is of the following forms:

- $R \leftarrow Pol_1, Pol_2, , Pol_n, n \geq 1$ Where $Pol_1, Pol_2, .., Pol_n$ consist of C_{name}, D_{name} or trust tickets and R is the name of the resource.
- $R \leftarrow Pol_{DELIV}$ Where R is the resource name and Pol_{DELIV} is the delivery policy terms for the resource.

Example: $MySQL \leftarrow (\{ Pol_2, Pol_1 \})$

The above policy rule for a MySQL data base specifies the requirement of two policies.

5.3 Resource Negotiation Examples

The scenario explained in the following examples is for a University System sharing resources over a community cloud as show in Figure 3.1.

5.3.1 Example of a Simple Share

In this example let us consider the Department of Computer Science wants a resource for a course taught in the department from the Department of Engineering. The Engineering Department is the controller and the Computer Science Department is the requestor. The resource is an application called blackboard that lets professors distribute and manage course material, exams, research papers and home work. Additionally, each of the departments may have their own regulations and policies that need to be satisfied. Let us consider that the controller will only release the application for courses that are being taught in the current semester. Furthermore, IEEE and ACM membership is required for sharing research papers available through the application. The requestor may also have specific requirements to enforce policies on data, access or may need additional features enabled on the application like have permission to create accounts for his Teaching Assistant. Figure 5.1 identifies the different steps in the negotiation.

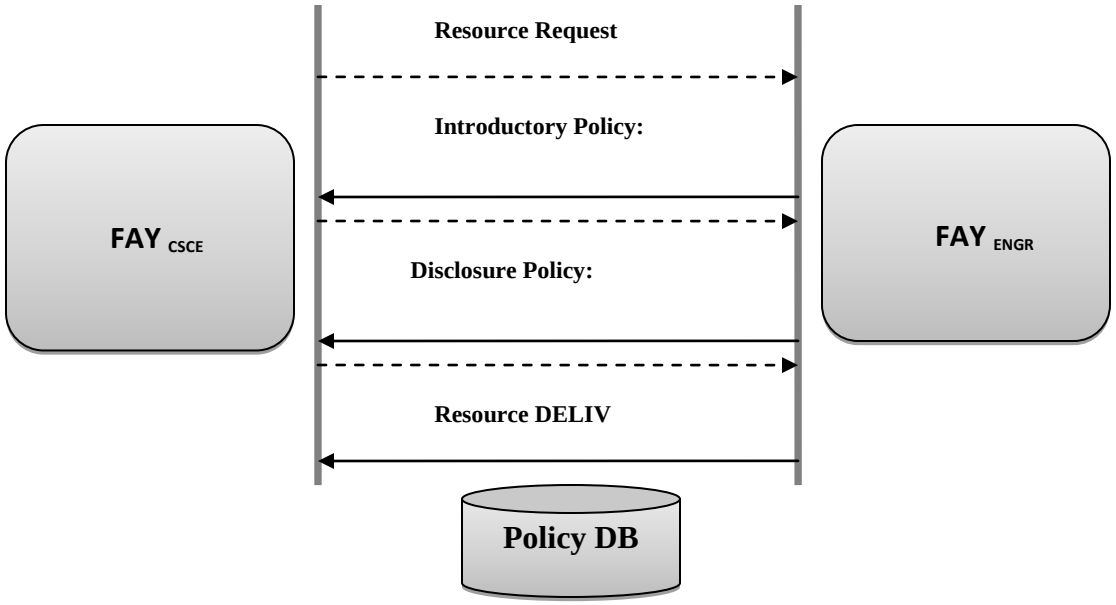


Figure 5.1: The Negotiation Process

The policy requirements are summarized below.

Resource Request: The negotiation process begins when a request for the resource is made.

$\boldsymbol{Fay_{CSCE}}: R_{request}(Black\ Board,\ version = \text{"2.2"})$

Introductory Policies: During the introductory stage the controller will check if the professor for whom the resource is being requested is a faculty member of the department.

$\boldsymbol{Fay_{ENGR}}: Pol_1 = D_{id}(name, university_{id})$

$\boldsymbol{Fay_{CSCE}}: C_{\boldsymbol{id}} = (name = \text{"John Doe"},\ university\_id = \text{"54321"})$

Custom requirements and policies over the resource and its usage are submitted by the requestor at this stage. If any of the requirements fail during this stage the negotiation fails and the resource is not delivered.

$\boldsymbol{Fay_{CSCE}}: R_{request}(Black\ Board,\ version = 2.2,\ permission = \text{"}create, update, delete\text{"})$

Disclosure Policies: If all of the introductory policies were successfully evaluated the negotiation moves to the disclosure policy stage. In this example there is only policy to be satisfied.

$\boldsymbol{Fay_{ENGR}}: Pol_2 = D_{membership}(ACM, IEEE)$

$\boldsymbol{Fay_{CSCE}}: C_{\boldsymbol{id}} = (ACM_{id} = \text{"1234"},\ IEEE_{id} = \text{"5678"})$

If the disclosure policies are successfully evaluated the resource is delivered. Trust Tickets may also be issued for this negotiation.

5.3.2 Policy Examples

In this example let us consider that a member in the community requests the use of a MySQL table called "cloud_research". The members identify each other with certificates. After the initial policies are satisfied, a set of disclosure policies must be satisfied which are summarized below.

$\boldsymbol{Pol_1}$ is a resource certificate that specifies that requestor wants access to a MySQL table name "cloud_research". The initial policy requires the requestor to be identified. $\boldsymbol{Pol_2}$ requires a certificate identifying the requestor. The policies on this data table further require that the user must be a US citizen. After these credentials are identified the resource is delivered.

$\boldsymbol{Pol_1} = (\{\ \}, MySQL \leftarrow R_{request}(MySQL, table = \text{"}cloud\_research\text{"}))$

$\boldsymbol{Pol_2} = (\{\ \}, MySQL \leftarrow C_{id}(name = \text{"}John\ Doe\text{"}, university\_id = \text{"}u12345\text{"})$

$\boldsymbol{Pol_3} = (\{Pol_2\}, MySQL \leftarrow C_{citzenship}(nationality = \text{"}US\ Citizen\text{"})$

$Pol_4 = (\{Pol_3, Pol_1\}, MySQL \leftarrow DELIV$

The disclosure policies can have an empty policy pre-condition and a policy rule as seen in Pol_1 and Pol_2 whereas Pol_3 and Pol_4 have policy pre conditions that need to be satisfied before they can be revealed. The notion of policy preconditions is to prevent unnecessary disclosure of policies especially when sensitive resources are involved in the negotiation. In the above example Pol_3 is disclosed only if Pol_2 is satisfied and further Pol_4 needs Pol_3 and Pol_1 to be satisfied for the resource to be released.

5.3.3 Example of a Chained Share

If the Engineering Department of Fort Smith campus wanted to use a resource from the Engineering department at Fayetteville, trust is established by following a regulated information flow where the resource request and certificates identifying the members are exchanged. The process starts when the Engineering Department at Fort Smith requests for the resource from the University of Arkansas system. This chain of events is depicted in Figure 5.2.

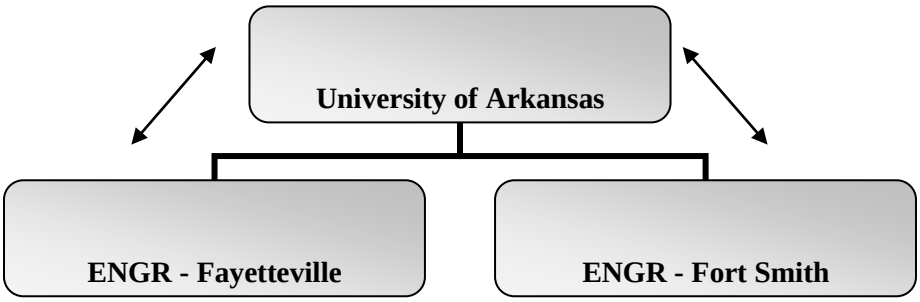
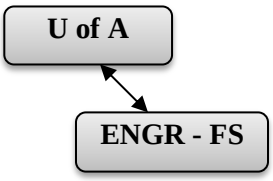
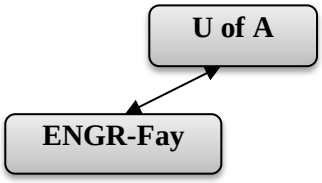


Figure 5.2: A Chained Share Example

The resource in this negotiation is a LAMP Stack (Linux, Apache, MySQL, and PHP). The chain of information exchange is depicted in Table 5.1 which only explains the resource request along with the identity exchange. The disclosure polices will start only after the flow of information is complete.

Table 5.1: Trust Establishment in a Chained Share

	$\boldsymbol{FS_{ENGR}}: R_{request}(LAMP,\ PHP\_version = \text{"5.3"})$ $\boldsymbol{FS_{ENGR}}: C_{id} = (Department =$ "*ENGR*", *university_id*="ABC123")
	$Pol_1 = (\{C_{id}\}, R_{request}(LAMP,\ PHP\_version = \text{"5.3"}))$ Where, $C_{id} = (Department =$ "*ENGR*", *university_id*="ABC123")

5.3.4 Examples of Resource Requests

The cloud service models defined in Chapter 1 classifies services in to SaaS, PasS and IaaS. The examples below explain resource requests for each of the service models. Since resource in this model refers to either a single or a set of resources, representing resources as software, platform or as virtual infrastructure is made easy.

Example for Software as a Service (SaaS):

$R_{request}(Black\ Board,\ version = \text{"2.2"})$

The above example is a resource request for an application called Black Board. The request also specifies the version of the software requested.

Example for Platform as a Service (PaaS):

$R_{request}(LAMP, PHP\_version = \text{"5.3"}, OS = \text{"RedHat 7.1"}, MySQl\_version = \text{"5.1"}, apache\_version = \text{"2.2"})$

The above example is a request for a platform called LAMP which denotes a web solution stack of software consisting of Linux, Apache, MySQL and PHP. The request above specifies the version of each of the software.

Example for Infrastructure as a Service (IaaS):

$R_{request}(SERVER, OS = \text{"CentOS 6"},\ Memory = \text{"16GB"},\ Hard\_drive = \text{"300GB"}, IP\_addresses = \text{"10"})$

The example above is a request for a virtual instance of a server mostly for hosting purposes with specifications listed for operating system, RAM and hard drive.

6. THE FAIR-SHARE MODEL

This chapter describes a model for fair sharing of resources within the community. During periods of high demand when multiple members request for the same resource the new requests can be denied due to unavailability of the resource in the pool until members holding the resource have finished their task. This may encourage members to request higher resource amounts than what they need in anticipation that the resource may not be available at a later stage. To overcome the problem of resource starvation during high demand, a fair-share model is proposed.

The main objective of the fair-share model is twofold. First, the community must be aware of the resource availability in the pool and current usage of the resource. Second, measures should be applied to improve the situation when demand is higher than resource availability. In order to correct the situation the model introduces a threshold. When the resource consumption reaches the threshold value necessary corrective measures are employed to improve the situation. The corrective measures include decreasing the amount of resource requested for new requests, reducing the time a resource can be allocated and enforcing a tax on the resource. These methods are discussed in the following sections.

6.1 The Model Architecture

The main components of the model are the resource pool, members, local decision module (LDM) and the global decision module (GDM). The main objective of this model is to track resource allocation and usage while constantly updating this information to the community. The resource pool is a virtual grouping of resources that are available to be shared within the

community as described in the previous chapters. The members of the community are either consumers or contributors to the resource pool. The local decision module tracks resource usage for a member. This information is fed to the global decision module which then tracks the resource usage across the community. The global decision module keeps track of the total resource usage of all members and thus the total usage of a specific resource within the community. If the demand is high and the resource availability is low, it can take specific measures to help keep supply available for a longer time. Figure 6.1 displays the architecture of the model.

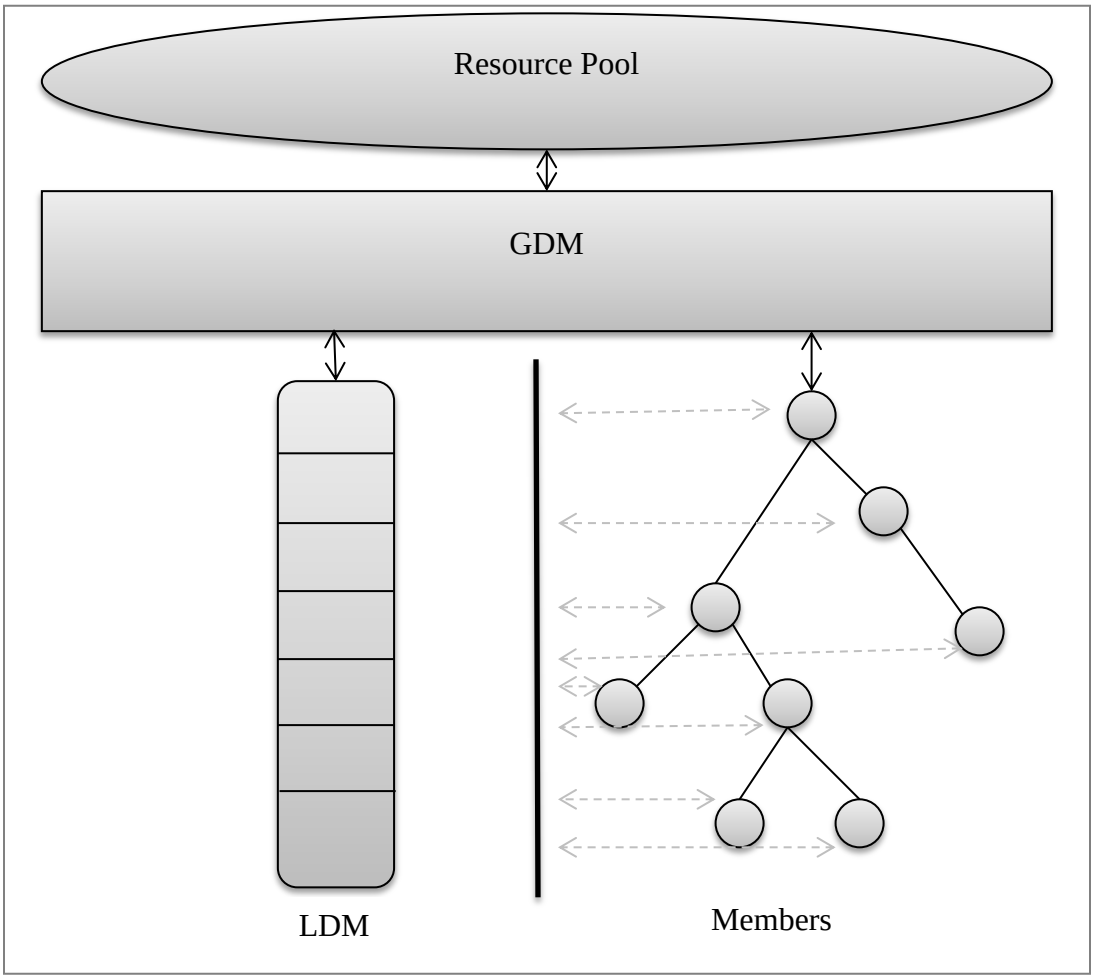


Figure 6.1: Resource Allocation Model Architecture

6.2 Economy in the Community Cloud

An economy within the community is thought of as comprising sustainable resource pricing and payment methods. The economic model can be borrowed from real world examples where resources are available for a cost that can be consumed for a price. Various economic models have already been developed for grids which are explained in [19][20][21]. The Community Cloud model assumes that such a system is already in place.

Currency within such economies is a virtual ticket, token or convenience points which are means of obtaining resources from members. The local currency can only be used within the community and also facilitates a payment model such that it can be exchanged for real cash or convenience points defined within the community. The currency is earned when a member shares a resource or it is spent when a member obtains a resource. The price of a resource depends on demand for the resource, amount of resource requested or resource type.

All transactions within the community are handled through the community bank [20] which is a secure service that handles payment and accounting functions in real time for every member. It maintains member accounts and resource usage in a database. This is shared across the community such that parent nodes have access to accounts of child nodes and child nodes have access only to their own accounts. The bank handles individual accounts by either crediting or debiting the virtual money as and when resources are used or shared. It can provide total service cost by compiling resource usage and pricing records from the bank for all members. The community bank along with history provides a powerful means of tracking member's actions in the community.

6.3 Fair Sharing of Resources

A serious situation within the community is resource contention which can lead to members being denied a resource when the demand for the resource is higher than the total resource available to be shared through the resource pool. This could lead to members requesting higher amounts of a resource than what they need in anticipation that the resource may not be available if needed in the future. Although there is a notion of a community, individual members are focused on their self-goals to maximize their own gains thus a community can have members that are greedy or selfish. Greedy members can act individually or group with other members and have tendencies to request higher amounts of resources than what they need. Members can also request higher amounts of resources to inconvenience other members who they may see as a rival. All of the above scenarios can lead to a higher demand of resources when there may not be a true need for the demand to be high. In order to manage a situation when the demand for a resource is higher than its availability in the resource pool, a strategy called fair share is developed. The main objective of this process is to share resource usage and availability information within the community and employ methods that manage resource usage.

6.3.1 Definition: Fair Share

Fair-share is a resource allocation method developed by the author that maximizes resource usage across the community by balancing resource supply and demand through the following strategies:

- The Resource Availability Plan
- The Share Management Plan

6.3.2 The Resource Availability Plan

One of the key objectives of the fair-share strategy is to track resource usage by members and current availability of a resource in the pool. The resource availability plan makes this data transparent to both members and the system. With this plan, the members are aware of the total amount of resource available in the pool and the amount currently being consumed. This information is vital for the members since they can make informed decisions about their own requirements based on what is available in the pool. This information is also used by the GDM described in Figure 6.1 to put into action the share management plan if the demand for a resource is higher than its supply. The resource availability and usage are defined with the following terms:

Definition 6.3.2.1: $share_{own}(M_i, R)$ for a member *M* and resource *R* is the total amount of resource that the member has contributed to the resource pool. The sum of these share amounts of all contributors for *R* is the total amount of resource *R* in the pool.

Definition 6.3.2.2: $share_{allocated}(M_i, R)$ for a member *M* and resource *R* is the total amount of resource allocated to a member from the resource pool.

Definition 6.3.2.3: $share_{current\_usage}(M_i, R)$ for a member *M* and resource *R* is the amount of resource that a member is currently using from the allocated amount.

The above definitions are for share amounts defined for an individual member. However, the resource availability information available to the GDM is the sum of these amounts across all members.

Definition 6.3.2.4 the resource availability plan defined for the community is of the form $\{ share_{allocated}(M,R),\ share_{current\_usage}(M,R) \}$ Where

$$share_{allocated}(M,R) = \sum_{i=1}^{n} share_{allocated}(M_i,R)$$

(*M* represents members who have contributed the resource *R* to the pool.)

$$share_{current\_usage}(M,R) = \sum_{i=1}^{n} share_{current\_usage}(M_i,R)$$

Hence, the resource availability plan for a resource in the community is a specification of the total amount available in the pool and the total consumption of that resource.

6.3.3 The Share Management Plan

The other key component is the corrective measures that the GDM employs to avoid denial of resources to the members when the demand is higher than resource availability in the pool. In order to prevent members from requesting higher amounts of resources than needed or prevent selfish members with malicious intent to inconvenience other members by requesting higher amounts this strategy introduces a threshold.

A threshold is a resource amount that is lower than the total amount of a resource in the pool. On reaching the threshold limit the system triggers a series of corrective steps to delay the depletion of available resources in the pool. Various corrective methods are studied and the advantages and disadvantages of each method are compared. The methods include reducing the amount of resource that was requested, forcing members to release the resource early and lastly, taxing members for the resource requested. The methods are outlined below and simulated results are explained in the next chapter.

6.3.3.1 Method 1 – Reducing the Requested Amounts

In this method after the threshold value for a resource is reached, any new requests for the resource is still processed but the requested amount for new resource requests is reduced. The idea behind this method is not to deny the resource to the requestor but still provide a slice of the available resource until the usage falls below the threshold value.

The advantage of this method is it successfully delays the complete consumption of the resource from the pool if demand continues to climb. However, the disadvantage in this method is that it encourages members to request for higher amount of a resource in anticipation that the resource may either be reduced or delayed, if demand is high.

6.3.3.2 Method 2 – Early Release of the Resource

In this method after the threshold value for a resource is reached, any new requests for the resource is still processed but the requestor is forced to release the resource early. The idea behind this method is to provide the full requested amount of the resource to the requestor but is forced to release it early until the usage remains above the threshold. This way the requestor can complete important tasks before releasing the resource.

The advantage of this method is it successfully delays the complete consumption of the resource from the pool if demand continues to climb. However, the disadvantage in this method is that it encourages members to request for higher amount of a resource in anticipation that the resource may either be available only for short durations or delayed, if demand is high.

6.3.3.3 Method 3 – Enforcing a Tax

In the last method a tax is imposed on all members requesting and consuming the resource if the threshold is reached. The idea behind this method is to allow members to reassess the utility of the resource they are consuming or requesting with respect to the increase in price. An assumption is made that members may be encouraged to release the resource partially or completely based on the new price for the resource. Members not completing important tasks or those willing to wait will either release the resource or wait for the resource until the tax is removed.

The above scenario is successful in delaying the complete consumption of the resource from the pool provided a few members either release the resources they were holding or new members delay their requests for the resource. The disadvantage of this method of taxation is in the assumption. If members do not release or delay their requests, the price of the resource would go up for all members using and requesting the resource and more importantly the situation does not improve which may result in the resource being depleted from the pool.

Based on the advantages of Method 1 and Method 2, a hybrid method is developed which continues to impose a tax when the threshold is reached but also reduces the resource amount or forces early release of the resource as discussed in Methods 1 and 2 for new requests. Although this method did improve the situation during high demand situations, it continued to have disadvantages since this method encourages members to request higher amount of resource. This further helped selfish members or members with malicious intent to inconvenience members they considered rivals by initially requesting higher amounts of resource so that their rivals had to pay higher price for lower amounts of the resource or for a lesser time.

In the last scenario the method adopted is similar to the hybrid method but in this case a tax is imposed only on members who have already requested the resource but are not utilizing it fully. All newer requests are forced to be released early or the request amounts are reduced like in Methods 1 and 2. This scenario is the best choice since it only taxes members who are holding large amounts of resource but utilizing only a part of it. If there is a true requirement for the resource, members pay tax for underutilized resources. However, if there is no true requirement for holding the resource, members have no incentive to hold on to the resource and pay a tax while the resource is still available in the pool without tax for other members. Even if the members continued to hold the resources, Method 1 and 2 would delay the resource from being depleted in the pool. This method ensures a slow depletion of the resource from the pool and thus improves the situation while enforcing a tax on underutilized resources which may result in their release, thus improving the situation further

All the above methods were studied and compared through simulated data which is explained in the next chapter.

6.4 Summary

This chapter described the main goals of the fair-share model which are providing resource usage and resource availability information to the community and methods to address situation when demand for a resource is higher than its availability in the pool. Several corrective measures were studied outlining the pros and cons of each while keeping in mind that demand for a resource could also be high due to greedy members within the community. The hybrid model which taxes members for underutilized resources while employing Methods 1 or 2 for

newer requests is considered the most efficient model as it encouraged members to give up resources they are not using while correcting the high load situation for newer requests.

The amount of tax enforced on a member can vary depending on the type of resource, amount of resource and contribution of the member to the community. These rules can be decided by the members of the community and is beyond the scope of this thesis. Additionally, unused resource amounts that trigger a tax can also be decided by the community.

The table below outlines the advantages and disadvantages of all the methods in the share management plan.

Table 6.1: Advantages and Disadvantages of the Various Methods Studied in the Share Management Plan

Method	Advantages and Disadvantages
Method 1: Reducing Requested Resource Amount when Threshold is Reached:	**Advantages:** • Successful in delaying the complete consumption of the resource from the pool if demand continues to climb. **Disadvantages:** • Encourages members to request for higher amount of a resource in anticipation that the resource may either be reduced or delayed if demand is high.
Method 2: Forcing an Early Release of the Resource when Threshold is Reached:	**Advantages:** • Successful in delaying the complete consumption of the resource from the pool if demand continues to climb. **Disadvantages:** • Encourages members to request for higher amount of a resource in anticipation that the resource may either be available for short durations or delayed if demand is high.
Method 3: Enforcing a Tax Resource when Threshold is Reached:	**Advantages:** • Successful in delaying the complete consumption of the resource from the pool if demand continues to climb.

	Disadvantages: • An assumption is made that few members requesting or already holding the resource will release the resource when the threshold is reached. • Encourages members to request for higher amount of a resource in anticipation that the resource may either be reduced or delayed if demand is high.
The Hybrid Method: Taxing Members with unused resources and employing Method 1 or 2 on New Requests when Threshold is Reached:	**Advantages:** • Successful in delaying the complete consumption of the resource from the pool if demand continues to climb similar to Method1 and Method2. • Tax is enforced only on members not fully utilizing the resource. • Such members have no incentive to hold on to resources they are not using and pay tax. **Disadvantages:** • There are no disadvantages on resource consumption due to this method.

7. SIMULATION AND ANALYSIS

The proposed corrective measures described in the previous chapter for the Share Allocation Plan were tested with simulated data that represent a high resource consumption scenario. The algorithms are controlled by changing the total amount of a resource in the pool, a threshold value, allocation time of a resource and the resource amount requested. The goal of the simulation was to study and compare all the proposed methods and analyze if they were successful in improving the situation. The simulations were created using PHP and MySQL.

7.1 Experimental Design

The important parameters that affect resource consumption are:

- Usage: is governed by the amount of resource and the time period for which a resource has been allocated or requested. Usage varies by each request.
- Resource Availability: Is the total amount of the requested resource in the pool.

In the simulated data depicting a high consumption scenario, the resource usage was compiled by changing the amount of the resource and the time requested for the resource. This was done by choosing random values from a fixed range. The arrival rate for requests could be altered. The above parameters were changed to get data that best depicted a scenario where the resource consumption quickly reaches the total amount of resource in the pool. The consumption was calculated based on the amount of resources being requested and the amount of resources that were released after the interval they were requested for was completed. The results were plotted on a graph shown in Figure 7.1. The X-axis depicts the time and the Y-axis depicts the consumption.

This simulated data was used to study the corrective methods described in the previous chapter. A threshold value is introduced in each of the simulations that trigger the corrective procedures. The resource availability in the pool and the threshold were kept constant while testing the different methods. In Figure 7.1 the amount of the resource in the pool is 70 units. The consumption reaches the peak when the resource in the pool is consumed.

Figure 7.1: High Resource Demand Scenario

7.2 Method 1- Reducing the requested amounts

The graph below show the consumption of the requested resource before and after Method 1 described in the Share Allocation Plan was employed. The idea behind this method is not to deny the resource to the requestor but provide a slice of the available resource until the

usage falls below the threshold value. This is done by granting a reduced amount from the original amount of the resource requested. The threshold depicted with a dotted line was set at 30% below the total amount of resources available in the pool. Once the threshold was reached the resource amount for newer requests was reduced by 50%. The new resource consumption is depicted by the darker spline and the original consumption is depicted by the lighter spline.

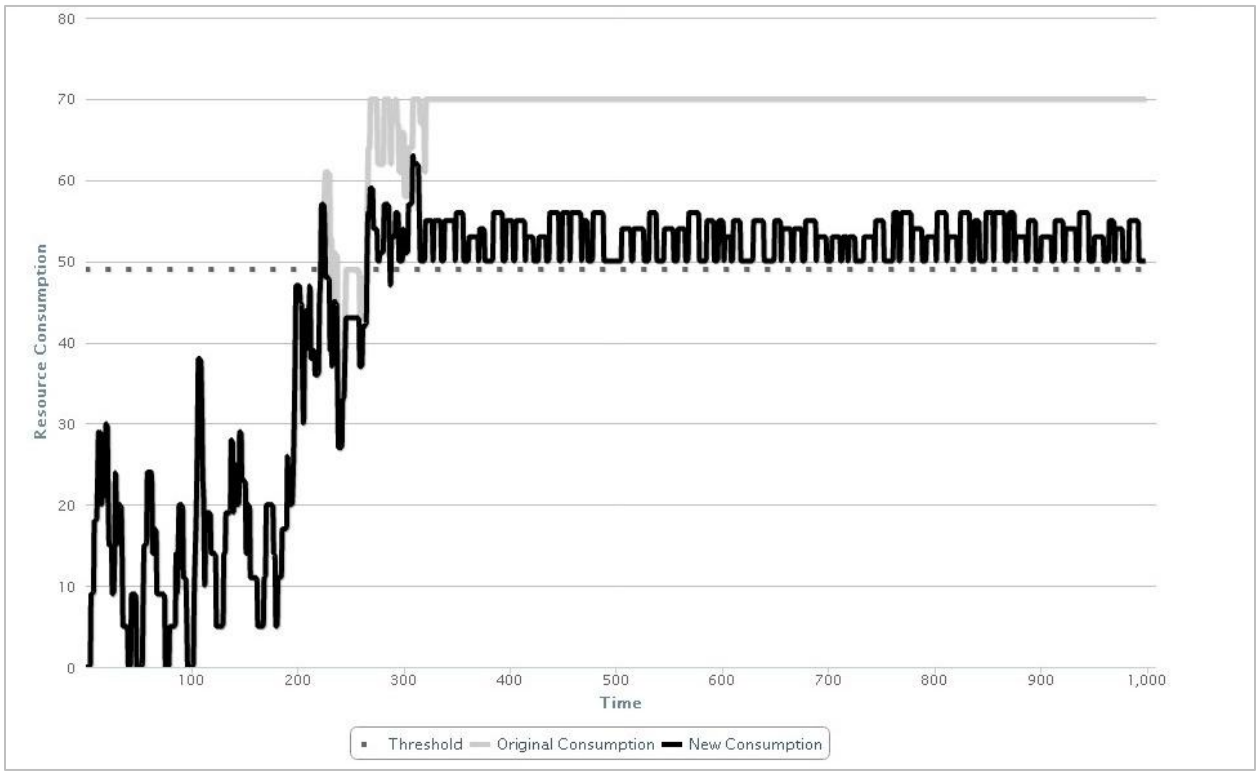


Figure 7.2: Method 1-Reduction in the Requested Resource Amounts

The graph clearly shows the consumption of the resource stayed well below the total available resource amount. The new consumption is based on the simulated data but consumption changes when Method 1 is triggered after the threshold is reached. Hence, before the threshold value is reached there is no difference in the consumption as shown in Figure 7.2. Once the threshold is reached the amount requested for the resource is reduced by 50% of the

original request which does not deplete the resource in the pool quickly and as resources already allocated are released the consumption stays below the total amount of resource in the pool.

7.3 Method 2- Early Release of the Resource

The graph below show the consumption of the requested resource before and after Method 2 described in the Share Allocation Plan was employed. The idea behind this method is to provide the full requested amount of the resource to the requestor but force the member to release it early. This way the requestor can complete important tasks before releasing the resource.

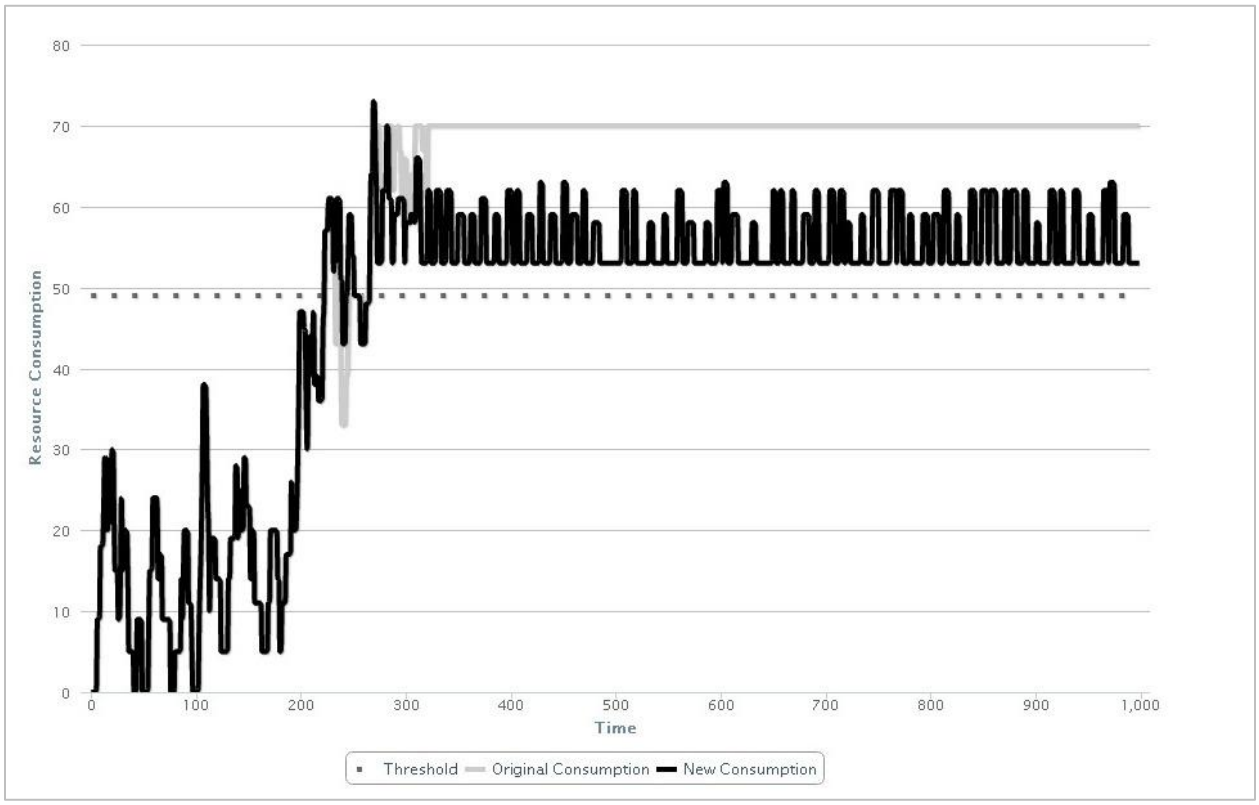


Figure 7.3: Method 2- Early Release of Requested Resources

The threshold depicted with a dotted line was set at 30% below the total amount of resources available in the pool. Once the threshold was reached, the requestor was only allowed to consume the resource for 50% of the original time that was requested.

The new resource consumption is depicted by the darker spline and the original consumption is depicted by the lighter spline. The graph above clearly shows the consumption of the resource stayed well below the total available resource amount for the same requests that caused the resource depletion. Method 2 was used on the simulated data which was triggered after the threshold was reached. Hence, before the threshold value is reached there is no difference in the consumption as shown in Figure 7.3. Once the threshold is reached the time requested for the resource is reduced by 50% of the original request which does not deplete the resource in the pool quickly since the newer requests are allocated for shorter time periods which can be observed by the thin spikes in the graph.

7.4 Method 3 – Enforcing a Tax

The graph below show the consumption of the requested resource before and after Method 3 described in the Share Allocation Plan was employed. The idea behind this method is to enforce a tax on members who are both holding and requesting the resource. An assumption is made that some of the resource holders will release the resource or members requesting the resource will delay their requests. In this simulation the consumption was reduced depicting the release of the resource by members when the threshold value was reached. The threshold depicted with a dotted line was set at 30% below the total amount of resources available in the pool. Once the threshold was reached the consumption was constantly reduced by 5% to depict a release of the resource by the members.

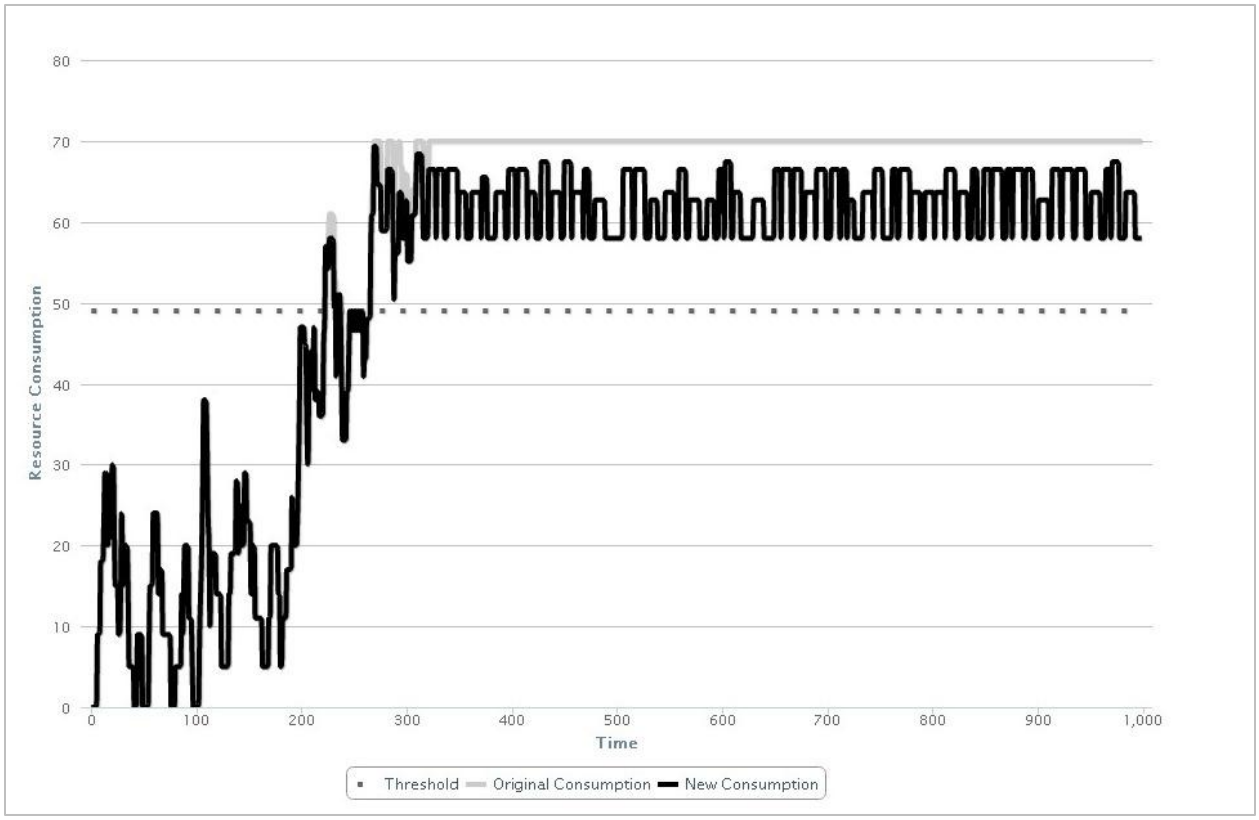


Figure 7.4: Method 3 – 5% Reduction in Resource Consumption when a Tax is Enforced

The new resource consumption is depicted by the darker spline and the original consumption is depicted by the lighter spline. The graph above clearly shows the consumption of the resource stayed well below the total available resource amount. However, the assumption made was that members constantly released resources that resulted in the consumption to drop by 5% every time. In order to simulate a more realistic scenario, the consumption was reduced by a random figure between 0 and 10% and at random time intervals. The resulting consumption is shown by the graph in Figure 7.5. The graph shows that if members constantly do not release the resource the consumption could touch peak amounts when the resource in the pool is exhausted.

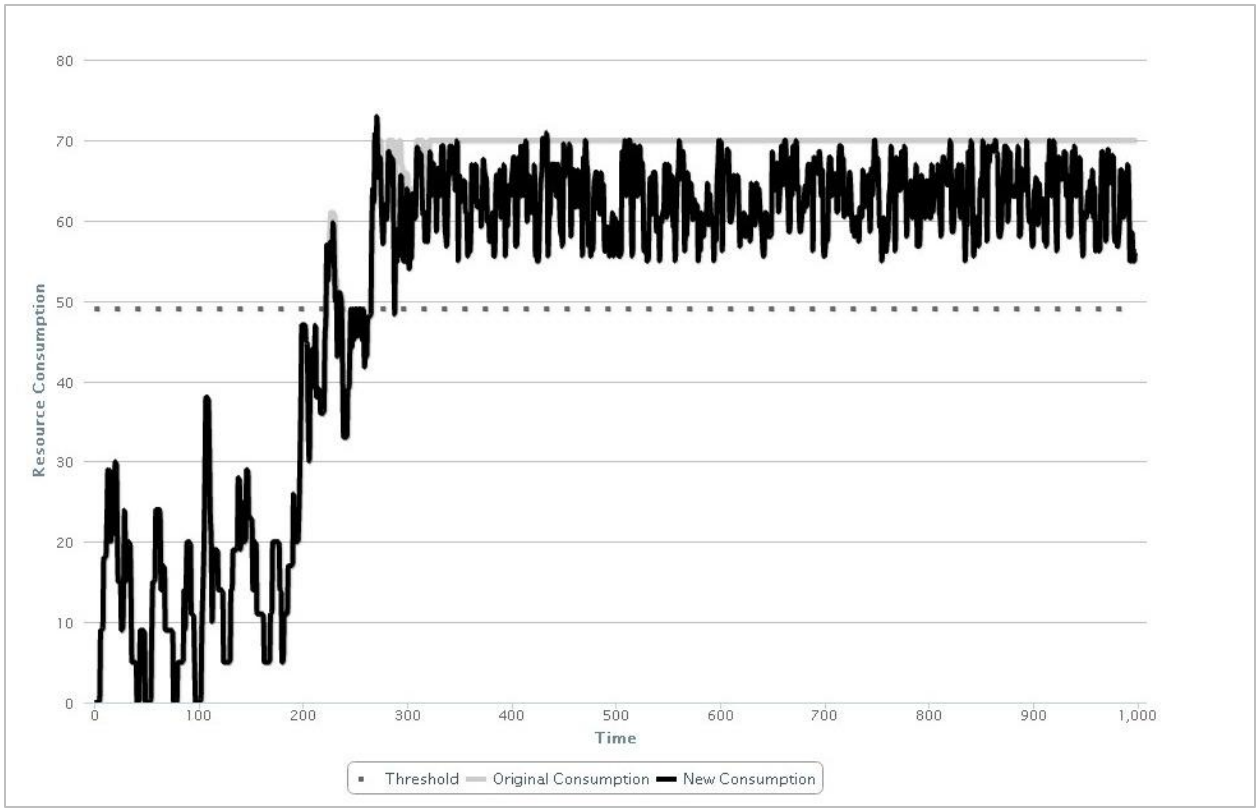


Figure 7.5: Method 3-Random Reduction in Resource Consumption when a Tax is Enforced

The above simulations were carried out on assumption that members with the allocated resource would release them once the Tax was enforced. Members delaying new resource requests were not considered in this situation. The purpose of this simulation was to study if the situation would improve if members released resource that was allocated to them. The graph shows it does help the situation but was not as favorable as Method 1 and 2. Furthermore since this simulation is based on an assumption that consumption will decrease, the results will be dependent on the number of members and amount of resource that was released.

7.5 Comparisons

The simulations can be changed by varying the threshold value which dictates when the corrective method will be applied. The simulations also depend on the amount of resource and time request for the resource allocation. While these parameters were chosen manually, the situation improved in every case. However, the scale of improvement in resource consumption can be decided by the community by either setting reasonable values or having a system that can predict high resource contention and decide the value for these vital parameters. Figure 7.6 shows an example of resource consumption when Method 1 was employed over off peak loads.

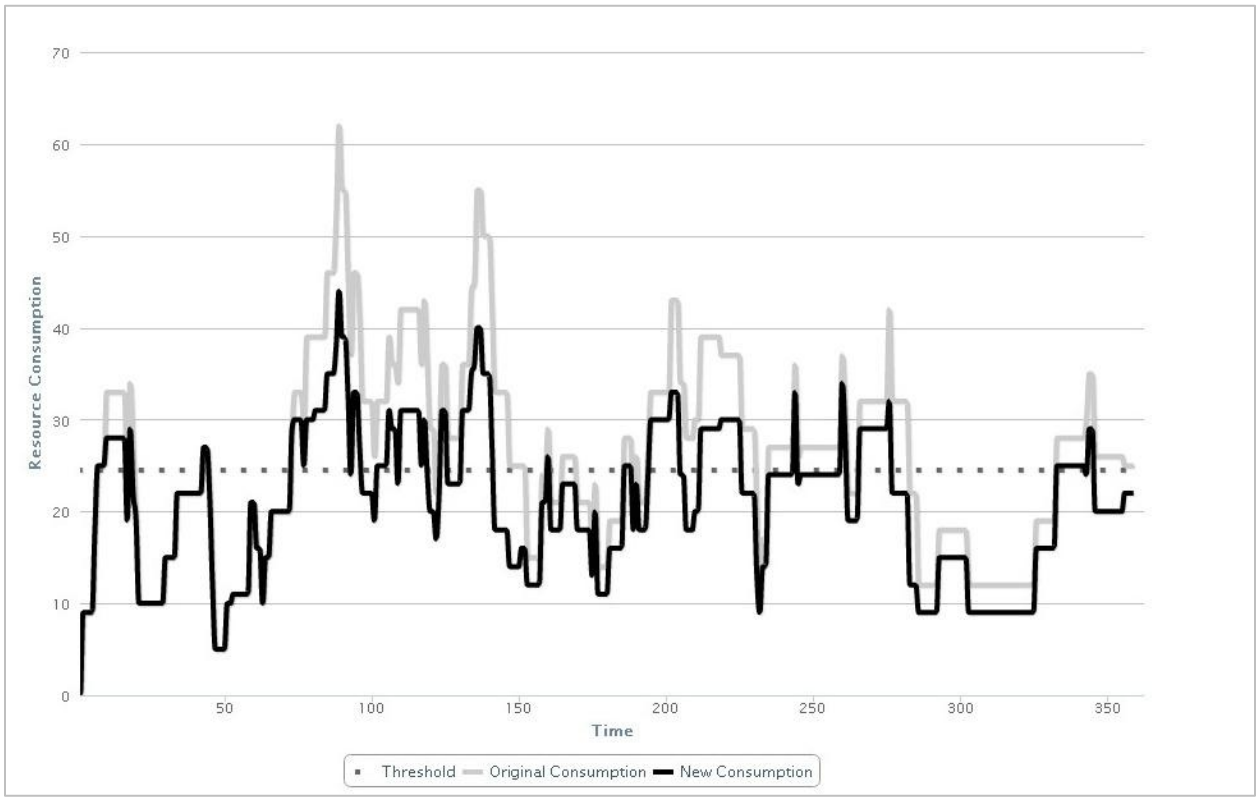


Figure 7.6: Method 1 Employed for Off Peak Loads

Based on the results derived from these simulations and the pros and cons of each method a hybrid solution is proposed that only taxes members who are not fully utilizing the allocated

resources and then employ Method 1 or 2 for newer requests as discussed in the previous chapter.

7.6 Conclusion

The experiments showed that Method 1 and 2 took different approaches in addressing the situation but were both successful in correcting the situation while making sure that newer requests were processed with some usage conditions without having to wait for the situation to improve. Method 3 also helped in improving the condition if the assumptions were true. The experiments help develop the proposed hybrid model.

8. CONCLUSIONS AND FUTURE WORK

8.1 Conclusion

Cloud computing has the potential to revolutionize how organizations conduct business. However, this technology faces new risks and challenges that need to be clearly understood and analyzed. This thesis focused on a deployment model of cloud computing called Community Cloud where organizations can share resources with each other. The opportunity for such a model does exist with changing business directions, demand and costs along with organizations that are already operating as a community. A security model that is based on trust negotiations is proposed that can form a standardized framework for sharing resources. Lastly, the fair-share model allows resource usage and availability to be transparent within the community while adopting methods to check for unnecessary increase in resource demand.

8.2 Future Work

The study of results from the fair-share model was based on simulated data due to the unavailability of any such data from the real world. In order to have a comprehensive understanding of these methods, simulations need to be run on data generated from a real world system.

The models proposed in this thesis can be extended to distributed operating systems, social clouds and other deployment models like the private and hybrid models of cloud computing.

REFERENCES

[1] P. Mell and T. Grance, “The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standarts and Technology,” 2011.

[2] K. Popovic and Z. Hocenski, “Cloud Computing Security Issues and Challenges,” *MIPRO, 2010 Proceedings of the 33rd*, pp. 344–349, 2010.

[3] J. Bardin, J. Callas, S. Chaput, and P. Fusco, “Security Guidance for Critical Areas of Focus in Cloud Computing,” pp. 0–176, 2009.

[4] NIST, “NIST Visual Model of Cloud Computing Definition.” [Online]. Available: http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html.

[5] G. Briscoe and A. Marinos, “Digital Ecosystems in the Clouds: Towards Community Cloud Computing,” *2009 3rd IEEE International Conference on Digital Ecosystems and Technologies*, pp. 103–108, Jun. 2009.

[6] I. D. Corporation, “IT Cloud Services User Survey.” [Online]. Available: http://blogs.idc.com/ie/?p=210.

[7] O. S. Foundation, “Recent Cloud Incidents.” [Online]. Available: http://cloutage.org/.

[8] S. Pearson, “Taking Account of Privacy when Designing Cloud Computing Services,” 2009.

[9] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, “Security and Privacy in Cloud Computing: A Survey,” *2010 Sixth International Conference on Semantics, Knowledge and Grids*, pp. 105–112, Nov. 2010.

[10] S. Ristov and M. Kostoska, “A New Methodology for Security Evaluation in Cloud Computing,” pp. 1484–1489, 2012.

[11] G. Briscoe and P. De Wilde, “Digital Ecosystems: Evolving Service-Orientated Architectures,” *Proceedings of the 1st international conference*, 2006.

[12] E. Bertino, E. Ferrari, and a. C. Squicciarini, “Trust-X: A Peer-to-Peer Framework for Trust Establishment,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 07, pp. 827–842, Jul. 2004.

[13] K. E. Seamons, M. Winslett, B. Smith, E. Child, J. Jacobson, and H. Mills, “Requirements for Policy Languages for Trust Negotiation,” *Proceedings Third International Workshop on Policies for Distributed Systems and Networks*, pp. 68–79.

[14] T. Yu, M. Winslett, and K. E. Seamons, “Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation,” *ACM Transactions on Information and System Security*, vol. 6, no. 1, pp. 1–42, Feb. 2003.

[15] L. Olson, M. Winslett, N. Seeley, A. Uszok, and J. Bradshaw, “Trust Negotiation as an Authorization Service for Web Services,” 2006.

[16] P. Bonatti and P. Samarati, “Regulating service access and information release on the web,” *Proceedings of the 7th ACM conference on on Computer and communications security*, 2000.

[17] A. C. Squicciarini, F. Paci, and E. Bertino, “Trust Establishment in the Formation of Virtual Organizations,” *Computer Standards & Interfaces*, vol. 33, no. 1, pp. 13–23, Jan. 2011.

[18] D. Ferraiolo and D. Kuhn, “Role-Based Access Controls,” *arXiv preprint arXiv:0903.2171*, pp. 554–563, 2009.

[19] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, “Economic Models for Resource Management and Scheduling in Grid Computing,” *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13–15, pp. 1507–1542, Nov. 2002.

[20] a. Barmouta and R. Buyya, “GridBank: a Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration,” *Proceedings International Parallel and Distributed Processing Symposium*, p. 8.

[21] S. Venugopal and R. Buyya, “An Economy-based Algorithm for Scheduling Data-intensive Applications on Global Grids,” in *Australia, Tech. Rep. GRIDS-TR-2004-11*, 2004.