Contributions to Service Level Agreement (SLA), Negotiation and Monitoring in Cloud Computing

Ali Alqarni

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

December 2022

Abstract

Cloud computing is a dynamic field of research, as the latest advances in the cloud computing applications have led to development of a plethora of cloud services in the areas of software, hardware, storage, internet of things connected to the cloud, and 5G supported by the cloud networks. Due to ever increasing developments and the subsequent emergence of a wide range of cloud services, a cloud market was created with cloud providers and customers seeking to buy the cloud services. With the expansion of the cloud market and the presence of a virtual environment in which cloud services are provided and managed, the face to-face meetings between customers and cloud providers is almost impossible, and the negotiation over the cloud services using the state-of-the-art autonomous negotiation agents has been theorized and researched by several researchers in the field of cloud computing, however, the solutions offered by literature are less applicable in the real-time cloud market with the evolving nature of services and customers' requirements. Therefore, this study aimed to develop the solutions addressing issues in relation to negotiation of cloud services leading to the development of a service-level agreement (SLA), and monitoring of the terms and conditions specified in the SLA. We proposed the autonomous service-level framework supported by the autonomous agents for negotiating over the cloud services on behalf of the cloud providers and customers. The proposed framework contained gathering, filtering, negotiation and SLA monitoring functions, which enhanced its applicability in the real-time cloud market environment. Gathering and filtering stages facilitated the effectiveness of the negotiation phase based on the requirements of customers and cloud services available in the cloud market. The negotiation phase was executed by the selection of autonomous agents, leading to the creation of an SLA with metrics agreed upon between the cloud provider and the customer. Autonomous agents improved the efficiency of negotiation over multiple issues

by creating the SLA within a short time and benefiting both parties involved in the negation phase. Rubinstein's Alternating Offers Protocol was found to be effective in drafting the automated SLA solutions in the challenging environment of the cloud market. We also aimed to apply various autonomous agents to build the new algorithms which can be used to create novel negotiation strategies for addressing the issues in SLAs in cloud computing. The monitoring approach based on the CloudSim tool was found to be an effective strategy for detecting violations against the SLA, which can be an important contribution to building effective monitoring solutions for improving the quality of services in the cloud market.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I wouldn't complete this PhD without the help and support of great and special people. First, I wish to thank my parents and my wife and lovely family. Their love provided my inspiration and has been my driving force. I have nothing but sincere gratitude for my supervisor Prof. El Rhalibi, Abdennour for the continuous support and especially for his patience. Also, I would like to thank my second supervisor Dr. Sudirman for his help.

DEDICATION

I would like to dedicate this thesis to my beloved homeland who gave me this great chance to be what I'm now.

PUBLICATIONS

ALi Alqarni, Abdennour El Rhalibi, Yuanyuan Shen (2018), ***Game-Theoretic Based Service Level Agreement for Cloud Computing***, the 6th IEEE International Conference on Multimedia Computing and Systems (ICMCS'18), Rabat, Morocco.

CHAPTER 1 INTRODUCTION

In this chapter, the research background will be discussed. The motivations and challenges to conduct this research work are discussed. The research aims and objectives will be delineated, along with a description of the scope of this study. The contribution of the research work towards the existing literature will also be highlighted. The structure of the thesis will be presented at the end of this chapter.

1.1 Introduction

The development of computing started from the mainframe which evolved into personal computing leading to the development of networked personal computing (Buyya et al., 2008; Pallis, 2010; Ogrpah and Morgens, 2008). Following this development, clients and servers came into existence leading to development of cloud computing through the use of the internet (Marston et al., 2011). Therefore, cloud computing is the latest development in computing systems, which also gave birth to the development of the Internet of Things (IoT), edge computing and 5G. The 5G wireless networks gave real impetus to the development of cloud services due to the need for collecting and analysing data from the IoT applications connected to the edge computing system which mediates the data processing and data storage near the applications in order to reduce the need for bandwidth. The data processed and analysed by edge gateways are sent to the cloud, or transmitted back to the IoT applications via the cloud. Hence cloud computing has a unique edge in coordinating different technological advances such as edge computing, growth of IoT applications and 5G networks (Denonno et al., 2019; Shu and Dustdar, 2016).

With the growth of technology and software development, the computing resources were developed to be delivered to the clients. In order to meet the customers' needs in the information technology sector, the diversity of the software was increased (Furht, 2010). In this way, the variety of services and products in the computing field has become available to the clients. In the past, the clients had to adapt to the existing set of limited computing resources in the market; and they used them to search for the services and products closely meeting their requirements (Pallis, 2010). In addition, the contractual affairs between the seller and the buyer used to be volatile in the past, and mostly the contracts used to be broken before reaching their maturity level (Spring, 2011). However, in the modern era of the computing, the services and products associated with cloud computing are being sold by professional sellers, and clients/customers have become knowledgeable clients with a greater degree of freedom in terms of choosing sellers and services depending on their needs (Tsai et al., 2010).

Moreover, the relationship between the clients and cloud computing providers continues to exist until the contract reaches its maturity level. Such relationships between the cloud provider and clients are termed as service level agreements (SLA) which carry legal weightage (Wu and Buyya, 2012). The SLA was designed in order to ensure the quality of services and compliance of the clients and cloud service providers to the clauses of the contracts. Before signing the formal contracts, both clients and cloud service providers negotiate with each other to get the most out of the contract (Patel et al., 2009). Thus, negotiation is an important pillar to reach a viable SLA between the user and the cloud service provider, which enables both parties to define the parameters for security and quality of services which are required during the provision of services to the customers (Wieder et al., 2011).

In addition, most of the cloud providers provide the cloud services through the SLA based on the principle of "take it or leave it", which means the service providers only work for their vested interests for selling the services without caring about the customers' requirements. Against the backdrop of these poor conditions of SLA offered to customers in the market, the customised SLA can play the role to maximise the profits for both clients and providers. The tailoring of the customised SLA requires negotiation between the cloud provider and client. Furthermore, the automation of the customised negotiation procedure is another important aspect which can enable both parties involved in negotiation to manage the complex and dynamic cloud computing services. It also allows both parties to maximise their return-on-investment in the cloud computing market.

The current automated SLA frameworks are limited and are unable to consider the following factors during the negotiation process: the lack of use of intelligent agents with ability effectively to negotiate on behalf of the customers and clients, the timing of resource allocation leading to overheads, the dynamic nature of cloud computing due to changing cost and quality needs and customers' requirements (Sim, 2006; Wu et al., 2013). Due to these factors, the automated SLA framework cannot answer the following questions: which measures can be taken to balance the trade-off between the varying and conflicting quality of service requirements; which service provider could offer the best services; and how a client can make effective judgements/decisions in accepting or rejecting an offer or generating the counter offer (Linlin et al., 2013).

To address the above issues and questions, we propose the automated intelligent agent-based SLA framework involving five stages in its lifecycle: gathering stage, filtering stage, negotiation stage, agreement stage, and monitoring stage. The gathering stage is characterised by gathering the offers of the cloud providers and the customers' preferences. The filtering stage filters the best possible providers meeting the requirements of the customers. In the

negotiation stage, the automated agents negotiate the terms and conditions of the SLA on behalf of the cloud provider and the customers. Both parties enjoy equal freedom at this stage in terms of choosing their intelligent agents representing their interests during the negotiation process. In the agreements stage, agreement is reached between the customer and cloud provider. Finally, the monitoring stage, at this stage, will monitor the agreement based on the agreed rules and regulations governing the SLA agreement. The breach of the agreement will invoke penalties for the provider.

Notably, the development of intelligent agents for the negotiation stage is a challenging task, mainly because of the difference between negotiation events and working of intelligent agents to maximise the utility for the customer and service provider. Yan et al (2007) described different negotiation strategies such as competitive negotiation strategy and cooperative negotiation strategy. The effectiveness of these strategies in the negotiation process has not been tested and compared in the literature (Silaghi et al., 2010). Thus, the current work also aims to compare the different negotiation strategies based on the outcome of the negotiation. Therefore, the current research work provides a way forward to generate the next generation cloud computing in which the services will be automatically negotiated between the client and cloud provider, and negotiation process and the SLA of resources delivery will be monitored through the intelligent agents without involving human interference.

1.2. Motivations for the study

This study was motivated by the fact that cloud services are being developed and dispensed on a large scale in the market with the growth in the number of consumers. Most of the organizations have already shifted their database to the cloud datacentres. With the growth of

consumers, the cloud services providers are also increasing, which gives rise to the need for the robust negotiation strategies for consumers and cloud providers, so that both transacting parties can equally obtain benefit out of the deals (Venticinque et al., 2010). The market of cloud services providers is expected to grow concurrently with the growth of the mobile phone market in the near future, and it will be easier for clients to swap the cloud providers based on the changing requirements of consumers using the techniques such as hot-providers-swap (Alsrheed, 2014). The possibility of the quick and easy shifting is also possible due to the current format of delivery and dispensation of services from cloud providers to the clients. For example, the cloud providers are dependent on the use of the Open Virtual Machine Format for packing and distributing the cloud services such as virtual machine images (Petcu et al., 2013).

Based on the client-cloud provider shifting prospects, there is a need for development of the negotiation strategies which can help the customers and cloud providers to reach agreements quickly (Zheng et al., 2012). The negotiation strategies involving humans as transacting bodies are described as time-consuming, and are not tuned to reach the agreements with mutual benefits for the participating parties. This gives rise to the development of negotiation strategies leading to the development of SLAs between the client and the cloud-provider (Wilkes, 2008; Brams, 2003).

Though previous literature shows the presence of the SLAs for the web-based services between the web-service providers and clients (Moghaddam and Davis, 2014; Rinderle and Benyoucef, 2005; Hashmi et al., 2014), there are very few studies which have attempted to solve the issues between the clients and the cloud-providers through the development of the SLA in the domain of efficient delivery of cloud-services (Alsrheed et al., 2014). The SLAs designed for the web-services cannot be extrapolated to the cloud-services because of the variations in the modes of service distribution and delivery mechanisms. Additionally, the

development of different cloud services and the lack of full maturity of the cloud market demand dynamic approaches to negotiate the cloud services in the real market (Sim, 2011). Therefore, it is critically important to develop the SLAs for the delivery and distribution of cloud services between the client and cloud-provider, and those SLAs should reflect the dynamicity of the cloud market resulting from the changes in the cloud-services and requirements of the clients.

The agent-based negotiation can play a fundamental role in mediating the SLAs between the client and cloud-providers, as they are automated and smart to negotiate on behalf of clients and cloud-providers for creating the mutually beneficial SLAs (Sim, 2011). The automated agents offer the solutions to issues encountered during human-level negotiation strategies within seconds, which truly reflect the spirit of the dynamic cloud market where both cloud services and demands of clients are in a state of flux (Alsrheed et al., 2014). There are some studies which have attempted to develop the automated agents-based negotiation strategies, but they have mostly focussed on the development of agents, resource allocation, and pricing mechanisms rather than solving the negotiation and SLA-related issues in the real cloud market (An et al., 2010; Alsrheed et al., 2014). Therefore, there is a need to develop an SLA framework by capturing the real data from the cloud market. This study intends to develop an SLA framework which can be applied to the conditions in the real-time environment of the cloud market.

In addition, the issue of monitoring the SLAs is another domain which is highlighted by several scholars for the viability and tenability of the SLAs between clients and cloud-providers (Badidi, 2013; Wu and Buyya, 2012; Anithakumari and Chandrasekaran, 2015). The quality of services is often affected by the lack of monitoring services related to the delivery and distribution of cloud services as described in the SLAs (Badidi, 2013). The dropping rates of clients from the SLA or suspension of SLAs are reported due to the

frequent violations of the terms and conditions of SLAs, which is mainly attributed to the lack of a properly installed monitoring mechanism overseeing the quality and delivery of cloud services as stipulated in the SLAs (Wu and Bidi, 2012). Patel et al (2009) also argued that simple process of 'measure and trigger' may not work for effective enforcement of the SLA, therefore, a robust and well-managed monitoring mechanism is required to ensure the parameters governing the quality of service attributed are closely monitored. Hence, we believe that the development of a robust monitoring framework is fundamental in enforcing the SLA in order to preserve the quality-of-service parameters relating to the SLA. This study will solve the issue relating to the proposed SLAs between the consumer and cloud provider through developing the architecture of a monitoring mechanism which will ensure the efficient management of the SLA between the consumer and cloud-provider.

GENIUS platform is used to perform tournaments which involved the autonomous agents compiled by different researchers, which have the ability to negotiate autonomously over the given issues and exchange offers with each other using a bilateral variant called Rubinstein's alternating protocol. The utility (u) of the bid (b) is indicated by the equation 1 (Yaqub et al., 2011).

$$u\,(b) = \sum_{i=1}^{N} \omega_i V_i(x_i) \qquad (1)$$

Where u and b represent utility and bid, respectively, ω_i is the weight of *xi* issue; $\sum_{i=1}^{N} \omega_i = 1$ and $V_i(x_i) = eval\frac{(x_i)}{\max(eval(x_i))} \in [0,1]$ denotes the normalized values of i^{th} issue, which are shown in customer's evaluation and cloud provider's evaluation columns of Table 7.1. $eval$ is the evaluation function which determines the utility of each issue in the scenario? The

cloud providers assign the evaluation function based on the scale of their resources, while the customers do so by considering the business demands.

The evaluation values of issues along with weights (priorities) form the preference profile of the customer and the cloud provider. The business objectives are defined based on utility functions, and agents try to maximize the utility function to which they represent without revealing any data regarding the utility values of opponents. The PaaS domain in the given scenario is reasonably large which might contain more than fifty thousand possible offers exchanged between the customer and the cloud provider. This task is very daunting for the human brain to handle; however, the intelligent agent can handle it efficiently within a reasonably short period of time.

Yaqub et al. (2014) showed that burden of the bidding on the human brain appeared in the form of reduced utility, while Chen et al. (2013) revealed that CHUCK agent was able to negotiate the deal within 2 minutes. Yaqub et al [2014] demonstrated that human brain was able to negotiate over only 83 rounds without achieving the deal due to the time factor taken by the human mind to process information and time required to make a decision, whereas the agent-to-agent negotiations could complete thousands of negotiation rounds without breaking off the negotiation process and simultaneously increasing the convergence rate.

1.3 Aims and Objectives

1.3.1 Aims

The aims of this research are twofold:

1. To develop an agent-based SLA negotiation framework for cloud computing
2. To develop an SLA agent-based monitoring framework in order to improve the quality of cloud services for customers.

1.3.2 Objectives

a. To conduct a systematic literature review for consolidation of the knowledge in SLA negotiations and monitoring and establish:
    i. What are the current algorithms for SLA negotiations?
    ii. What are the current protocols for SLA monitoring?
    iii. What are the deficiencies in present ways and protocols?

The success of this objective will be measured through the development of a novel SLA framework involving negotiation and monitoring components, and a novel algorithm for the proposed agent to carry out negotiation at the negotiation stage of the framework.

b. To develop an optimal, novel, agent-based SLA framework

The Optimality will be defined in terms of maximum utility, resource and cost efficiency and achievement of desired levels. The techniques will be measured using multi-objective optimization based on Pareto, Nash and Kale optimality.

The success of this objective will be measured through development of SLA phases in the light of SLA frameworks used for other related technologies. In addition, the performance of proposed autonomous agent in the competition will measure the efficiency and effectiveness of the SLA framework.

c. To develop a functional, novel, agent-based SLA monitoring framework

The functionality will be defined in terms of ensuring the system's health, efficient tracking of operations and capability of detecting and reporting SLA violations.

The success of this objective will be measured by planning experiments to test the efficiency of monitoring components of the SLA framework in identification of violations.

d. To evaluate performance of the proposed frameworks; and comparison with existing state-of-the-art frameworks

i. Evaluation of cloud SLA negotiation using Genius multi-agent system platform

ii. Evaluation of SLA monitoring, using cloud simulation environment CloudSim and comparison against defined benchmarks

The success of this objective will be measured by carrying out the experiments on performance of the SLA framework's negotiation and monitoring components using GENIUS platform and CloudSim.

By achieving the stated objectives, the research project's aims will also be attained.

1.4 Contribution of the study

Our proposed work contributes to the existing literature on cloud computing literature in the following ways:

1. Cloud SLA negotiation is a "decision-making" process to resolve conflicts between a client and a service-provider. There are limited models and frameworks for SLA and negotiation. In addition, there is a scarcity of the automated intelligent based SLA frameworks in the literature. Therefore, in this research, we aim to propose a state-of-the-art novel "agent-based" SLA negotiation framework that will be functional in terms of enhancing service connectivity, responsiveness and reliability for cloud customers.
2. Furthermore, this research work is novel in terms of proposing a customised "agent-based" SLA monitoring framework that will employ the agent's ability of "negotiation", "competition" and "cooperation" in order to have an automated SLA management. The proposed framework will be more proficient in guaranteeing the framework's robustness and scalability, identifying, and reporting of SLA infringement.

3. Much of the research carried out in negotiation is focused on theoretical aspects of negotiation protocol and strategy; the practical aspects related to development of scenarios from the users and cloud providers perspective. This study will develop the scenarios in Platform-as-a-software (PaaS) and Platform-as-a-storage (PaaSt), and test them using the negotiating agents. Thus, this study will contribute a unique data set to the literature relating to users' and cloud providers' utility.
4. This research work will also develop the novel algorithm for the proposed negotiating agent in order to enhance its efficiency and effectiveness during negotiation in comparison with existing agents. In addition, protocol for implementation of the proposed SLA will be developed.
5. This study will use the evaluation simulator CloudSim to implement the Monitoring component of the proposed SLA framework for delivery of services from cloud provider to customers. Hence, the main contribution of this work will be towards development of an evaluation mechanism using an evaluation simulator.

1.5 Research Scope

Our research work is linked with five research areas including negotiation; cloud computing, game theory, intelligent agents and machine learning. The brief descriptions of these areas are illustrated below:

1.5.1 Cloud computing

The cloud computing has four main layers involving the Software-as-a-service (SaaS), Platform-as-a-service (PaaS), Infrastructure-as-a-service (IaaS), and Storage-as-a-service

(Staas). This work designs the SLA which will be applicable to resolve issues of service provision in Platform-as-a-software (PaaS) and Platform-as-a-storage (PaaSt) parts of the cloud computing. The SLA and negotiation strategies developed in our work mainly focus on the public cloud deployment models which emphasize promoting the effective negotiation between the cloud provider and the consumer.

1.5.2 Intelligent agents

Our work uses the intelligent agents to represent the cloud-provider and the consumer for negotiating the SLAs. The intelligent agents are useful in representing the preferences of their clients, and are based on the utility functions and negotiation strategies. Each intelligent agent taking part in the negotiation tries to increase its own utility during the negotiation process.

1.5.3 Game theory

Game theory is a useful approach for determining the level of conflicts between the competing utility-based agents. As mentioned earlier, during negotiation, each agent endeavours to maximize its utility, which can cause deadlocks or delays in the negotiation process during the conflicting interests of the intelligent agents. The game theory introduces concepts such as Nash equilibrium which denotes the point where both competing parties win the maximum benefit of out the negotiation (Yaqub et al., 2014). Negotiations conducted in this way aim to target the Nash equilibrium, so that both consumer and cloud-provider can benefit almost equally or to their satisfaction as a result of the negotiation. Game theory and related concepts will be introduced in later parts of this thesis.

1.5.4 Negotiation

The negotiation is mediated between two agents: consumer and cloud-provider in this research work. The Rubenstein bargaining model, which is also called Rubenstein Alternating Offers Protocol, as suggested by Rubenstein (1982) is used to conduct negotiation between the intelligent agents. Negotiation and Rubenstein Alternating Offers Protocol will be discussed in later parts of this thesis.

1.6 Research Methodology

The brief methodological tools adopted to conduct this research work are outlined in this section.

1.6.1 Reviewing

The existing literature in the cloud computing, negotiations and SLAs was scanned, carefully analysed in order to find the research issues in the domain of service level agreements in cloud computing. The state-of-the-art literature was also perused to identify the methodological tools adopted by the previous researchers to solve the research problems in cloud computing and service-level agreements.

1.6.2 Requirements for designing the frameworks

In the initial stages, all the stages necessary for developing frameworks for SLA and monitoring of SLA were determined, and included in the relevant frameworks presented in the later chapters of this thesis. In addition, the user requirements for each stage of the

frameworks were identified. The inputs and outputs were also identified for each phase of the frameworks.

1.6.3 Implementation of SLA and monitoring frameworks

We have implemented the SLA framework in the real world. The data from the cloud market were used to design scenarios which were negotiated by the state-of-the-art automated negotiation agents. The GENIUS (General Environment for Negotiation with Intelligent multi-purpose Usage Simulation) was used to conduct negotiations between agents.

1.6.4 Implementation of SLA monitoring framework

The parameters described in the SLA for measuring the quality-of-service attributes in the SLA were closely monitored through the implementation of the monitoring framework. The monitoring of quality-of-service attributes was conducted using CloudSim.

1.7 Structure of Thesis

Chapter 1: This chapter has presented the motivation of the research, the aim of objectives, research contribution to the existing literature, and overview of the research methodology. The scope of the research is also discussed.

Chapter 2: The technical background related to the cloud computing and SLA will be presented in this chapter.

Chapter 3: The related work will be discussed. The extensive discussion on negotiation frameworks, negotiation strategies are presented. We have also described the applications of

negotiation in cloud computing SLAs, and monitoring of cloud computing SLAs. The concepts related to simulation in monitoring and tools used for monitoring are also discussed.

Chapter 4: The autonomous SLA framework is developed with illustration of different stages and user requirements for each phase.

Chapter 5: The autonomous SLA framework was implemented using GENIUS. Negotiation scenarios will be developed and implemented in GENIUS.

Chapter 6: The novel SLA algorithm for the negotiation agent will be presented and implemented in this chapter.

Chapter 8: An adaptable monitoring framework for cloud SLAs will be developed and implemented using CloudSim.

Chapter 9: The thesis will be concluded in this chapter; including a discussion of the contributions, the limitations of research and future work.

1.8 Conclusion

This chapter has provided the description of research motivation. The limited knowledge available for negotiating SLAs and monitoring mechanisms in the area of cloud computing and SLAs motivated us to conduct this research. This research project aimed to design and implement an SLA and monitoring frameworks for efficient delivery of cloud services from cloud providers to the consumers in the real-time market environment. The contribution of this work is the development of a novel autonomous SLA framework and SLA monitoring framework for measuring the quality-of-service attributes. The research methodology covered the reviewing of literature, designing the SLA framework and monitoring framework

The next chapter will discuss the background to cloud computing and various services offered under the umbrella term of cloud computing. The concepts related to service level agreements, negotiation and game theory will also be illustrated.

1.9 Limitations of the study

This study, like any other study, has some limitations which should be kept in mind while interpreting and applying the outcomes of this study to improve the negotiation and monitoring of the cloud services offered by the cloud providers to the cloud customers in the cloud market. This study only benefited from the data from Azure and Amazon, and not from other cloud providers in the market. The other cloud providers may have different cloud services and offerings, which might affect the applicability of outcomes to other cloud providers in the market.

The impacts of fluctuations in demand-and-supply, changing trust levels between customers and cloud providers, emerging and novel cloud offerings and variations in cloud services across various regions are some of factors which may affect the negotiation outcomes between the customers and cloud providers. This study did not take into account the afore-mentioned factors while experimenting and interpreting results. Therefore, the negotiation strategies employed during this study should be optimized while incorporating the foregoing factors into the negotiation framework consisting of intelligent agents.

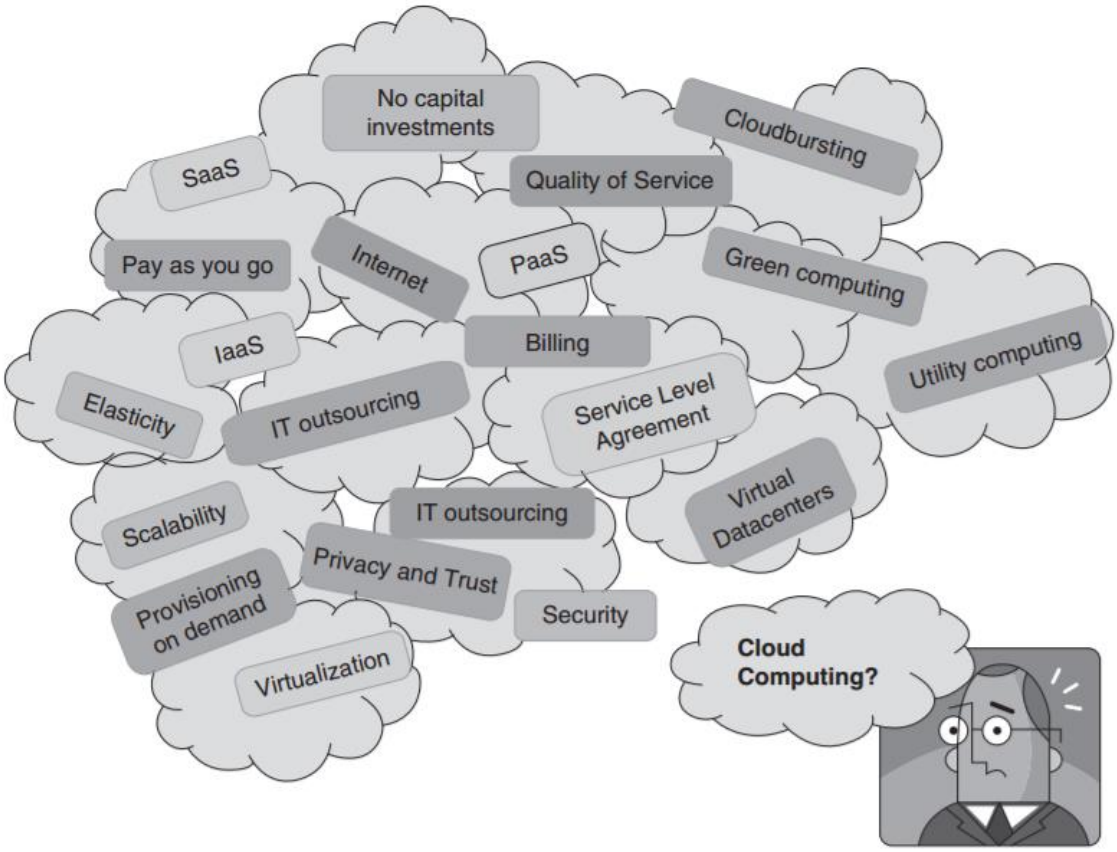
Also, Penalties taken based on breaking the agreement in the level of services provided. You need a clear definition in the contract and more expansion in terms of the type of penalties and procedures taken to compensate or cancel the service for the customer and the procedures for the cloud provider.

CHAPTER 2: BACKGROUND OF CLOUD COMPUTING AND SLA

The previous chapter presented the aims and objectives of this research work, which were related to the cloud-computing and the SLA. This chapter reviews the literature in the area of cloud computing, SLA and negotiation. The chapter has been divided into eight main sections. The first section focuses on the definition of cloud computing; the second section presents a brief history of cloud computing. The cloud computing layers are discussed in the third section, while the SLA along with its different phases is outlined in the fourth section. The overview of SLA metrics and parameters are introduced in the fifth section. The negotiation and game theory-related concepts are discussed in the sixth and seventh sections. The conclusion of the chapter is presented in the eighth section.

2.1. Defining the cloud computing

Nowadays, cloud computing has become a buzzword in businesses and the IT industry, which represents a variety of services, concepts and technologies (Buyya et al., 2008). It refers to IT outsourcing, utility computing, virtualized hardware-on-demand, software as a service (SaaS), platform as a service (PaaS), and several other services offered by the IT industry to their clients in the market (Patidar et al., 2012; Spring, 2011). Figure 2.1 shows the different notions one can imagine when it comes to defining cloud computing.



2-1: The cloud concepts and technologies which are provided under the umbrella term of cloud computing

Armbrust et al (2009) defined cloud computing in this way:

"Cloud computing refers to both the applications delivered as services over the Internet, and the hardware and system software in the datacentres that provide those services."

The above stated definition of cloud computing covers everything, ranging from the development and delivery and the underlying infrastructure, to the sophisticated software and applications as services. Therefore, it emphasises the concept of 'everything as a service often referred to as XaaS' under which datacentres, deployment platforms and IT hardware and so on are priced, measured and delivered to clients as a service (Marston et al., 2011). Buyya et al (2008) argues that the user requirements specified in the definition of cloud computing presented by American National Institute of Standards and Technology should be met by the cloud providers (Buyya et al., 2008):

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Similarly, Buyya et al. (2010) stressed the utility-oriented aspect of cloud computing in their definition:

"A Cloud is a type of parallel and distributed system consisting of a collection of interconnected 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."

In the next section, a brief history of the cloud computing will be presented.

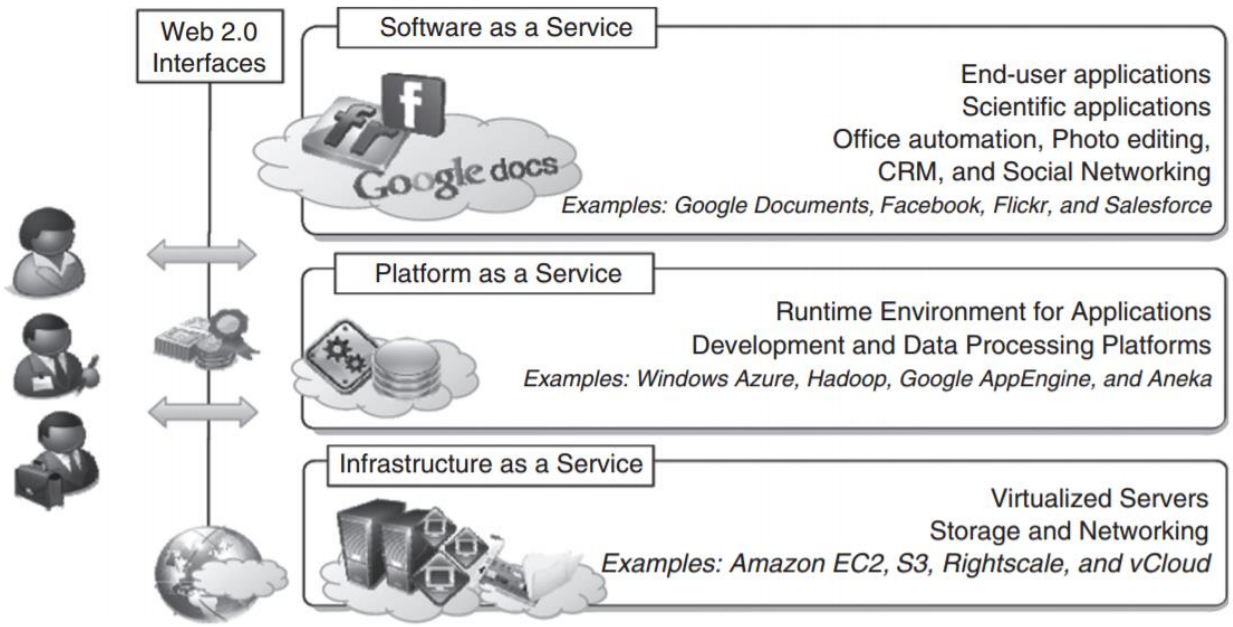
2.2. Brief History of Cloud Computing

Leonard Kleinrock had envisioned the spread of the computer networks in 1969 by saying that the day is not far away when the computing networks will become sophisticated and mature enough to turn into computer utilities like several other utilities such as telephone, water and electric utilities, and computer utilities will be provided to the users' homes and offices like any other service. This vision of the computing as a utility predicted the transformation of the entire IT industry in the 21st century. Later on, Douglas Parkhill had written a book entitled "The Challenge of the Computing Utility" in which he described the characteristics of cloud computing. With the advent of internet and web-2.0 technologies, this vision of Kleinrock was realised, because the Web 2.0 played a critical role in transforming Cloud computing into an attractive opportunity for developing computing applications; and the Internet was turned into an attractive and viable platform for service delivery (Xu, 2012; Furht, 2010). The virtualization component of cloud computing enabled the incorporation of

flexibility in terms of developing enterprise systems, controls over quality and production, and customization (Stanoevska-Slabeva, K., & Wozniak, 2010). The service-oriented aspect of the cloud computing enabled the production of products familiar to the customer's abstractions. As a result of these developments, in 1990, the use of cloud computing was started to describe the utilities such as virtual private network (VPN) services and ATM networks (Jadeja and Modi, 2012; Zhang and Cheng, 2010).

2.3. Cloud computing layers

The cloud computing services are diverse in nature, and can be categorised into three main categories or layers: infrastructure as a service (IaaS), Platform as a service (PaaS), and software as a service (SaaS) [Weinhardt et al., 2009; Armbrust et al., 2010; Luo et al., 2011). These layers are interconnected with each other as shown in figure 2-2.



2-2: The relationship of three main service categories in cloud computing (Modi al., 2013).

2.3.1. Infrastructure-as-a Service (IaaS)

These services aim to deliver the infrastructure of the cloud computing in the form of virtual networking, hardware and storage to the clients. The cloud providers offer the virtual machine instance and computing on-demand on these virtual machines (Bhardwai et al., 2010). On request of a client, the cloud provider provides users with the infrastructure, users account, tools and interfaces to manage the software, configure the applications and software installed on the virtual hardware. The pricing is negotiated between the cloud provider and the user based on the hourly usage and characteristics of infrastructure required by the user. The virtual storage is provided to the user in the format of virtual space on the disk or object store. The virtual disc space is an important vehicle for offering continuous storage to the user, and complements the virtual hardware (Luo et al., 2011). However, the object storage is highly sophisticated abstractions for storage of entities instead of files. The virtual networking enables the connection between different virtual instances, and ensures the connectivity of virtual hardware with the private networks or the internet. The examples of IaaS involve the Amazon EC2, S3, Rightscake, and vCloud.

2.3.2. Platform-as-a-Service (PaaS)

These services constitute the next layer of solutions/services in cloud computing. They provide the on-demand and flexible runtime environments which are normally utilized by the host to execute various applications (Modi et al., 2013). The middleware platform is used to back these services, which are responsible for execution of the abstract environment in which applications are implemented and deployed. The cloud provider takes the whole responsibility of managing the applications' fault tolerance limits and issues of scalability; and the users are required to concentrate on the logics used to develop the applications and APIs and libraries of the cloud providers (Weinhardt et al., 2009). On one hand, this approach

puts constraints on the users to work within a controlled environment, and on the other hand it enhances the abstraction level used to leverage the applications in cloud computing (Villegas et al., 2012). The examples of PaaS include the Windows Azure, Hadoop, Google App Engine, and Aneka.

2.3.3. Software-as-a-Service (SaaS)

The collection of these services constitutes the top layer, and intends to offer the software, application and other related solutions on an on-demand basis to the clients (youseff et al., 2008). The cloud providers in this domain increase the scalability and accessibility of most of the desktop applications on an on-demand basis such as customer-relationship management software, photo editing, office automation, and document management. These applications are actually replicated on the cloud provider's hardware for making them scalable and accessible via browsing (Qian et al., 2009). The cloud provider shares these applications with multiple users or the single user on the request of the client. The SaaS layer is active in developing the social networking websites, which is responsible for generating the popularity metric based on the load sustained by website (Buyya et al., 2009). The prominent examples of SaaS include Facebook. Google Document, Salesforce and Flicker.

2.4. Service level agreement (SLA)

In this section, the SLA is defined and the components of the SLA are explained. This section also describes the lifecycle of SLA.

2.4.1. Defining SLA

The ‘service level agreement’ is defined by the researchers as a contract which is used to specify the details of agreement including the terms for quality of services promised by the cloud providers and expected by the clients, the terms and all cases of violations, arrangements for the provision of services and guarantees, and measures for the definitions of parameters governing the quality and level of services (Wu and Buyya, 2012; Xu, 2012; Liu et al., 2011). The SLA is executed between the cloud provider and another party which may be a broker negotiator, direct consumer of the cloud services or monitoring negotiator. The main purpose of the SLA is to provide the consumer/client with the descriptions of all the cloud associated services ranging from availability of the cloud services, through performance and quality to the billing information (Cai et al.., 2010; Arora et al., 2012). This concept officially defines the penalties and sanctions the cloud provider will pay in the event of non-provision of the promised services to the clients. SLA can be executed in multiple domains of IT related services such as internet, web-management, web-services, data centre management and networking (Maurer et al., 2012; Joshi et al., 2014). The descriptions of SLA in each case may differ depending on the nature of the services. The summary of definitions of SLA in the foregoing domains can be presented in table 2.1:

Table 2-1: SLA definitions in different IT related domains

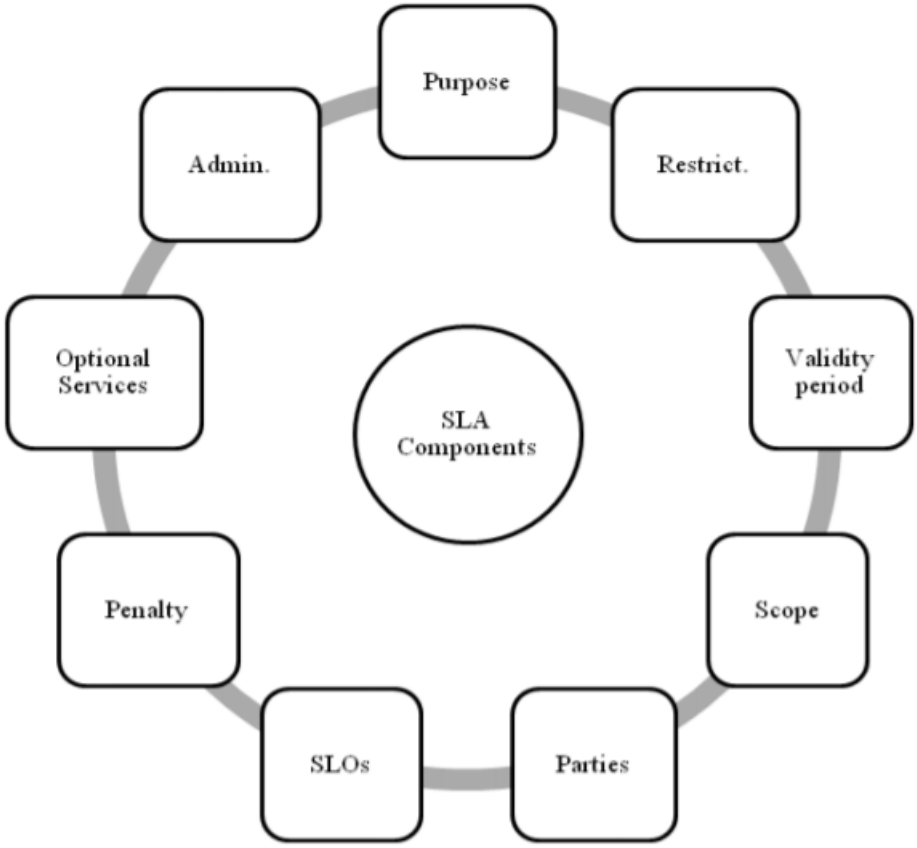
Domain	SLA definitions	References
Web Services	“SLA is an agreement used to guarantee web service delivery. It defines the understanding and expectations from service provider and service consumer”.	HP Lab (Jin et. al. 2002)

Networking	"An SLA is a contract between a network service provider and a customer that specifies, usually in measurable terms, what services the network service provider will supply and what penalties will be incurred if the service provider cannot meet the established goals"	Research Project (Jamakovic et al., 2013)
Internet	"SLA constructed the legal foundation for the service delivery. All parties involved are users of SLA. Service consumer uses SLA as a legally binding description of what provider promised to provide. The service provider uses it to have a definite, binding record of what is to be delivered"	Internet NG (Ron et. al.2001)
Data Centre Management	"SLA is a formal agreement to promise what is possible to provide, and provide what is promised".	Sun Microsystems Internet Data Centre group (2002)

According to Aljoumah et al. (2015), the SLA is used to enhance the professional relationship between the parties involved in the SLA, increase the acceptance level of customers, and to improve the service quality and performance of the services.

2.4.2. Components of the SLA

Several researchers agree with the components described by Jin et al. (2002) for an ideal SLA. The components described by Jin and his colleagues are given below (Figure 2.3.):



2-3: The illustration of different components of the SLA (Jin et al., 2002)

Purpose: This component defines the purpose and need of the SLA between the parties

Parties: In this component, all the parties to be part of the SLA are mentioned with detailed descriptions including their duties and portfolios.

Validity Period: Validity period is an important component which covers both the start date and end date of the SLA.

Scope: This component describes the nature and level of the cloud services agreed between the cloud provider and the consumer. This component also ensures that descriptions of the services should be adequate to inform the consumer about the procedures adopted to provide the services.

Restrictions: In this component, the cloud provider illustrates the necessary measures which can be undertaken to supply the required level of the cloud services to the consumer.

Service level objectives: In this component, service level objectives are mentioned in detail. All the services approved by the cloud provider and the consumer are mentioned, such as availability, reliability, quality-of-service parameters and performance. Day-time restrictions associated with each service and target for the service accomplishment within a specific period are also presented in this component.

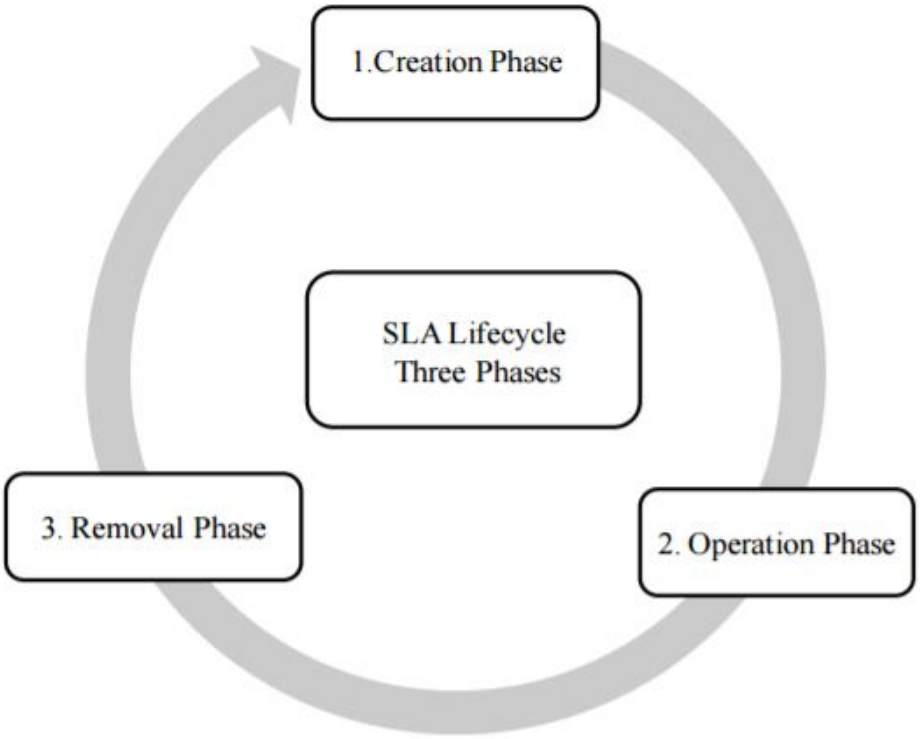
Penalties: The penalties are defined for the cloud service provider in the event of breaking the provisions of the SLA in terms of providing the required level of services. Similarly, the penalties are also described for the consumer in case he/she does not comply with the usage policies of the cloud services.

Optional services: The optional services are those services which are normally not required by the consumer, but may be requested in special circumstances.

Administration: This component covers all the procedures and steps taken by the cloud provider to supply the services and meet the targets and goals specified in the SLA

2.4.3. SLA lifecycle

The lifecycle of the SLA was defined by Ron et al. (2001). According to these authors, the SLA cycle contains three critical phases: creation phase, operation phase, and removal phase (Figure 2-4).



2-4: Description of three phases of the SLA lifecycle as described by Ron et al (2001).

Creation phase: In this phase, the consumers search for the cloud service provider matching with their requirements and needs for the specific service

Operation phase: In this phase, the consumer is provided with the read-only access to the description of services within the SLA.

Removal phase: In this phase, the SLA is terminated either on the expiry of the SLA or the violation of SLA terms and conditions. All the information and configurations associated with the SLA are deleted from the service systems.

Sun Microsystems Internet Data Center Group provided a more comprehensive six steps SLA lifecycle, which is explained in Figure 2-5.

Discover – service providers: In this step, the consumers assess their needs and requirements for a particular service, and locate the cloud service provider matching with these requirements.

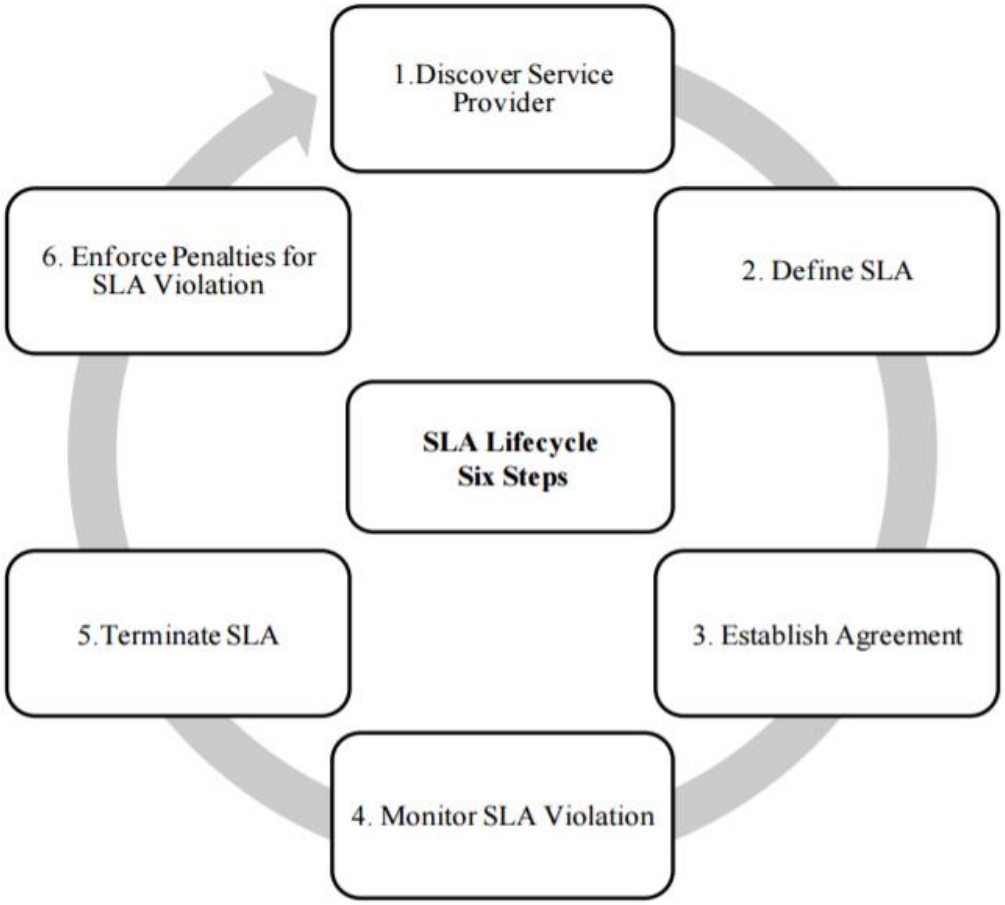
Define – SLA: In this step, definitions of the parameters and metrics of the services, level of services quality of service, penalties and parties are described. This stage allows the negotiation between the parties to reach the mutually agreed terms and conditions of the SLA.

Establish – agreement: in this step, the SLA template is created, filled in by the relevant parties, and agreement becomes operational.

Monitor – SLA violation: In this the cloud provider's performance and compliance to the terms and conditions of the SLA is measured.

Terminate – SLA: In this stage, the SLA is terminated either due to violation of the SLA or expiry of the SLA.

Enforce – penalties for SLA violation: In this phase, any violations on behalf of the contracting parties are penalties by invoking the penalty clause of the SLA.



BV

Figure 2-5: Six steps lifecycle of SLA proposed by Sun Microsystems Internet Data Center Group (2002).

Critically, the six steps lifecycle of the SLA is more reasonable and logical than the three-phase lifecycle of the SLA. This is because it provides better control in the establishment step where both parties (consumer and cloud provider) exchange the contract messages, negotiate and negotiate the level and quality of services, issues of needs/requirements which finally help shape the SLA benefitting all contracting parties equally (Blythe et. al. 2004). In addition, the monitoring of SLA violation helps all parties bound in the contract to adhere to the SLA terms and conditions in terms of achieving the service quality objectives (Gong et. al. 2003). These steps are not provided in the three-phase lifecycle. Therefore, the six steps

SLA lifecycle provides detailed information, and will be used to refer to the SLA lifecycle used in this research work.

2.4.4. SLA metrics and parameters

SLA metrics and parameters are employed to monitor procedures, enhance software procedure, and employ business policies in the domains of services where it is required to measure the satisfaction level of service quality objectives (Rima et al., 2009; Durkee, 2010; Buya et al., 2011). Previous researchers have acknowledged that it is hard to develop new parameters, which is why researchers deal with the rules for generating service parameters (Emeakaroha et al., 2012). Based on the SLA metrics and parameters, the information is collected to confirm the level of achievement of service quality objectives. The cloud computing services contain three layers of services – IaaS, PaaS, SaaS, and for each layer, there are different parameters which are employed to test the service quality in these areas. The parameters and metrics for each layer have been summarised in the table 2.

2.4.5. SLA metrics for IaaS

The clients using the infrastructure as a service use several significant metrics to negotiate the SLA while choosing the hardware for the cloud computing (Aljoumah et al., 2015; Paschke and Schnappinger-Gerull, 2006; Keller and Ludwig, 2003), as shown in Table 2-2.

Table 2-2: SLA Parameters for IaaS and their descriptions

Parameters	Descriptions
CPU capacity	CPU speed for VM (Virtual Machine)
Memory size	Cache memory size for VM
Boot time	Time for MV to be ready for use

Storage	Storage size of data for short or long term of contract
Scale up	Maximum of VMs for one user
Scale down	Minimum number of VMs for one user
Scale up time	Time to increase a specific number of VMs
Scale down time	Time to decrease a specific number of VMs
Auto scaling	Boolean value for auto scaling feature
Availability	Uptime of service in specific time
Response time	Time to complete and receive the process

Source: Aljoumah et al (2015)

2.4.6. SLA metrics for PaaS

The developers of the software using the PaaS do not need to depend on installing and organizing the hardware for development of cloud applications. The studies have recommended some significant PaaS associated metrics which can essentially be used by the developer to negotiate a better SLA with the PaaS suppliers ((Aljoumah et al., 2015; Paschke and Schnappinger-Gerull, 2006; Keller and Ludwig, 2003), as shown in Table 2-3)

Table 2-3: SLA Parameters for PaaS and their descriptions

Parameters	Descriptions
Integration	Integration with e-services and other platforms.
Scalability	Degree of use with a large number of online users
Pay-as-you-go billing	Charging based on resources or time of service
Deployment environment	Supporting offline and cloud systems
Browsers	Firefox, Explorer, etc.
Number of developers	How many developers can access the platform

Source: Aljoumah et al (2015)

2.4.7. SLA metrics for SaaS

The IaaS is provided by many suppliers such as Yahoo, Google and Microsoft in the form of social websites, calendar and mail. The significant metrics for IaaS to negotiate SLAs with the suppliers can be viewed in table 2-4 (Aljoumah et al., 2015; Paschke and Schnappinger-Gerull, 2006; Alhamad et al., 2010)

Table 2-4: SLA Parameters for IaaS and their descriptions

Parameters	Descriptions
Reliability	Ability to keep operating in most cases
Usability	Easy built-in user interfaces
Scalability	Used with individual or large organizations
Availability	Uptime of software for users in specific time
Customizability	Flexible to use with different types of users

Source: Aljoumah et al (2015)

2.4.8. SLA metrics for storage as a service

The clients gather data from different sources and need to store them in safe places for access in future. Suppliers like Amazon S3 have built powerful storage system for clients with ability to host data from millions of clients and guarantees that data can be used for different applications. The significant parameters associated with SaaS which can be used to negotiate SLAs with cloud providers can be viewed in Table 2-5 (Bianco et al., 2008; Aljoumah et al., 2015; Paschke and Schnappinger-Gerull, 2006; Alhamad et al., 2010)

Table 2-5: SLA Parameters for Storage as a Service and their descriptions

Parameters	Descriptions
Geographic location	Available zones in which data are stored
Scalability	Ability to increase or decrease storage space

Storage space	Quantity of units of data storage
Storage billing	How the cost of storage is calculated
Security	Cryptography for storage, transferring data, authentication, and authorization
Privacy	How the data will be stored and transferred
Backup	How and where images of data are stored
Recovery	Ability to recover data in disasters or failures
System throughput	Amount of data that can be retrieved from system in a specific unit of time
Transferring bandwidth	The capacity of communication channels
Data lifecycle management	Managing data in data centers, and using network infrastructure

Source: Aljoumah et al (2015)

2.5. Negotiation

Several scholars have proposed the definition of the negotiation (Alexander et al., 2015; Rosenschein and Zlotkin, 1994; Gelfand et al., 2011) in different disciplines, however, the common thread in all these definitions is that the parties involved in the negotiation process try to gain the maximum benefits. The definition offered by Thompson (2012) for the negotiation process is used for this work, according to which negotiation is "a decision process in which two or more parties make individual decisions and interact with each other for mutual gain. Negotiation is an integral part of the selling and buying process of cloud computing services, because the negotiation process allows the customer to discover the characteristics of the services offered by cloud providers and enable the decision to accept or reject the offers. Negotiation is performed on different aspects of cloud computing services such as the pricing, guarantees, features/characteristics of the services and quality of the services. Therefore, the negotiation process is based on three main aspects of the cloud

computing services: price, utilization and availability of the resources (Jayasankar and Ghulli, 2015).

2.6. Negotiation protocols

The negotiation protocols refer to the set of norms and rules which are obeyed during the negotiation process, and which are used to determine the fate of the negotiation process (Reaidy et al., 2006). They also provide a sequence of events/actions carried out during the negotiation process. (Kersten et al., 2004) Several negotiation protocols have been created until now, however, the most commonly used automatic negotiation protocols to negotiate a deal between the buyer and service provider are illustrated below along with their state diagrams.

2.6.1. Fixed Price protocol

The state diagram for this protocol has been illustrated in Figure 2-6. The fixed price protocol is also referred to as the "take-it-leave-it" protocol. In this protocol, the seller sends only one offer-to-sell as a final offer, and does not involve the exchange of offers and counter offers. If the unique offer-to-sell made by seller is accepted by the buyer, the deal is done. However, if the buyer does not accept the offer of the seller, the negotiation is terminated. In this protocol, the seller holds the authority to withdraw the offer any time. This protocol is criticised in that it does not involve feedback from the customers, therefore, it is not beneficial for the bilateral negotiation process. In addition, it does not fulfil the gist of the negotiation process which aims to offer both parties opportunity to gain maximum benefit out of the negotiation process. Due to lack of constant interaction and communication in this protocol, this protocol is not used by most of the service providers.

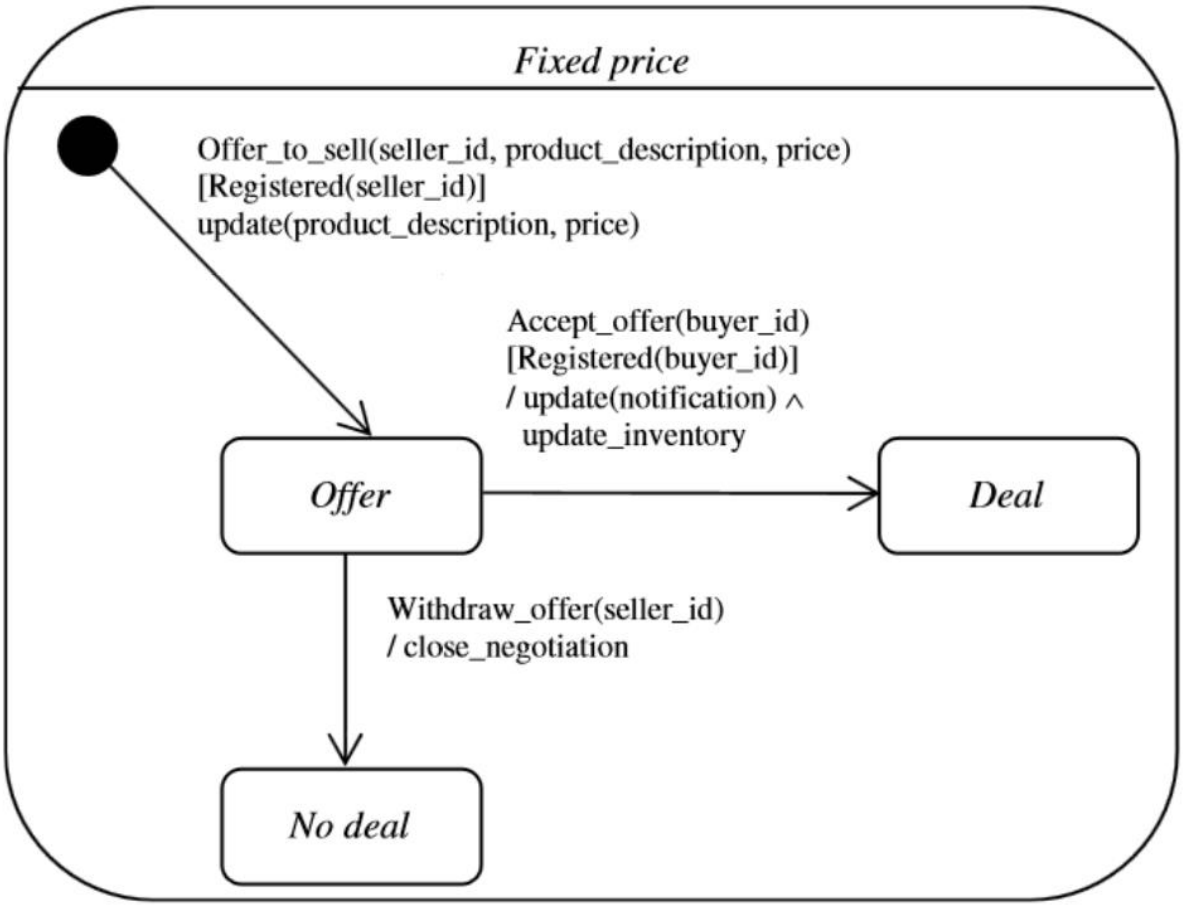


Figure 2-6: Fixed price protocol (Rinderle and Benyoucef, 2005)

2.6.2. English Auction

The state diagram of the English action protocol is given in Figure 2-7. In this protocol, the offer-to-sell message is created by the seller, and made open to many buyers. The buyers submit their bids, and each buyer receives an update message about the offer made by the previous buyer in order to respond to this message through counter-offer. If the inactivity period exceeds the set limit, the auction is closed automatically. The protocol contains the element of the 'hierarchical state *auction closed*' containing deal or no deal components. The deal takes place if the offer received is higher than the reserved price. However, there is no deal if the highest offer received is less than the reserve price.

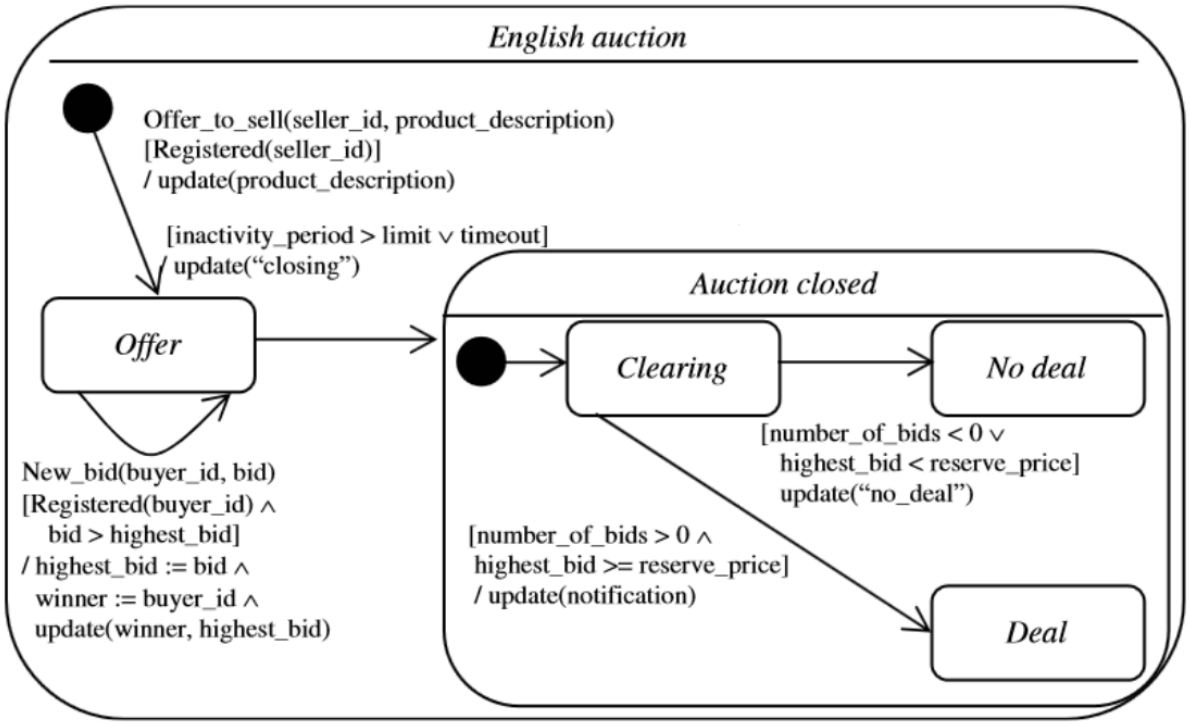


Figure 2-7: English auction protocol (Rinderle and Benyoucef, 2005)

2.6.3. Dutch auction protocol

The state diagram of the Dutch auction protocol is shown in Figure 2-8:

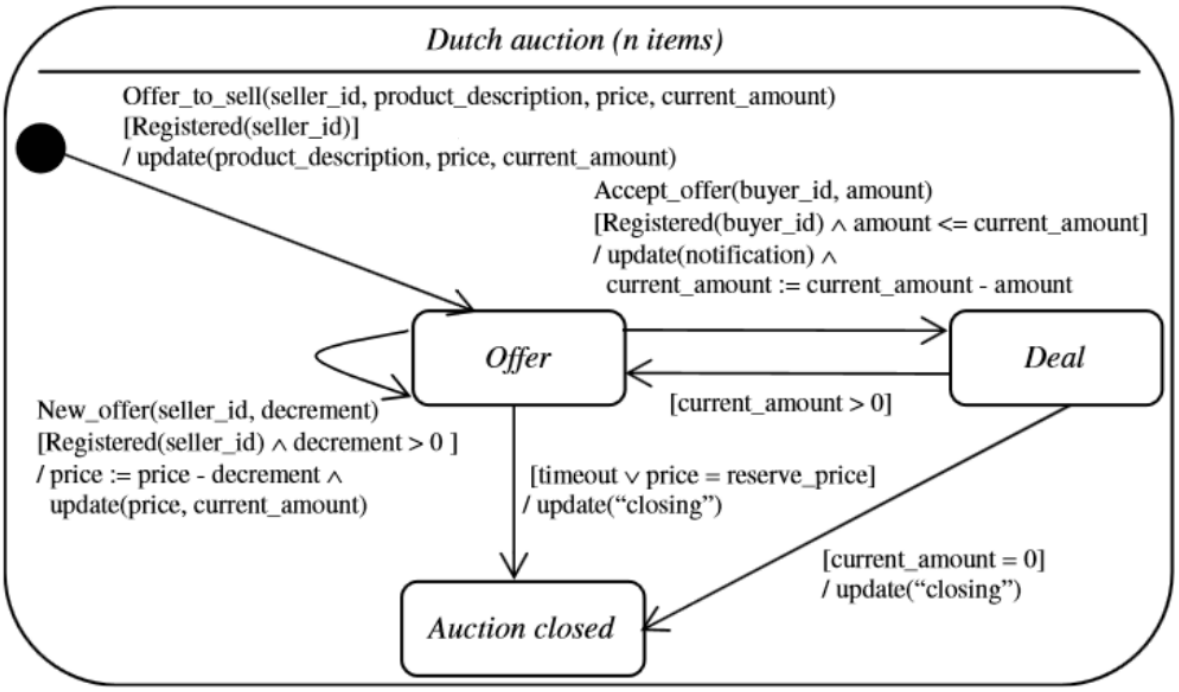


Figure 2-8: Dutch auction protocol (Rinderle and Benyoucef, 2005)

In the Dutch auction protocol, the seller creates the offer-to-sell message starting with a higher price for a specific number of items, and can gradually decrease the price for gradually increasing the number of items. If the buyer shows an interest in items, and wants to buy them in higher quantities than that specified, the seller may gradually decrease the price. In this protocol, the number of services/items matter in reducing the price. This type of auction protocol is more suitable to items such as vegetables in the grocery market and selling airplane seats.

2.6.4. Double auction

The state diagram for the double auction can be seen in Figure 2-9. In the double auction protocol, both parties submit their bids at the same time. A match between the buyer and seller decides whether the deal takes place between the parties involved. However, if offers submitted by seller and buyer do not match, there is no deal between the seller and the buyer. Matchmaking is shown in the clearing phase as indicated in Figure 2.9.

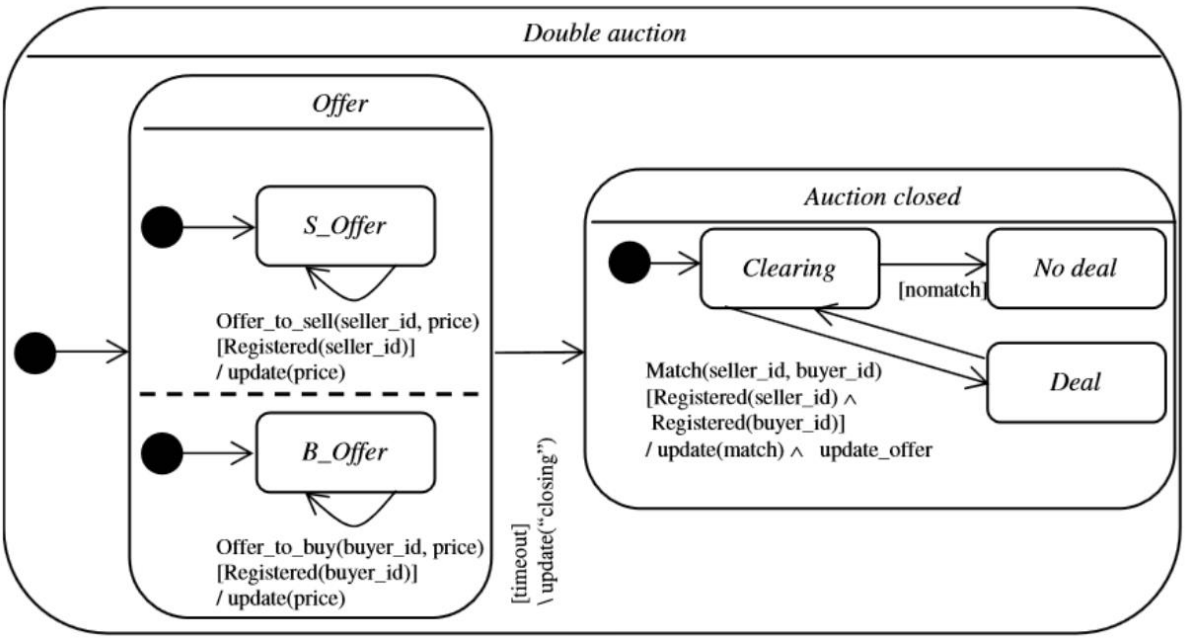


Figure 2-9: Double auction (Rinderle and Benyoucef, 2005)

2.6.5. Bargaining/Rubinstein's Alternating Offers Protocol

In the bargaining protocol, both parties have the right to make the initial offers, however, the seller initiates with the first offering, and the customer is given opportunity to offer his/her initial offerings at the later stages of the bargaining process. As this protocol involves both seller and buyer in the negotiation process, this is why it is termed as a bilateral or two-part negotiation protocol. Figure 2-10 shows the bargaining protocol:

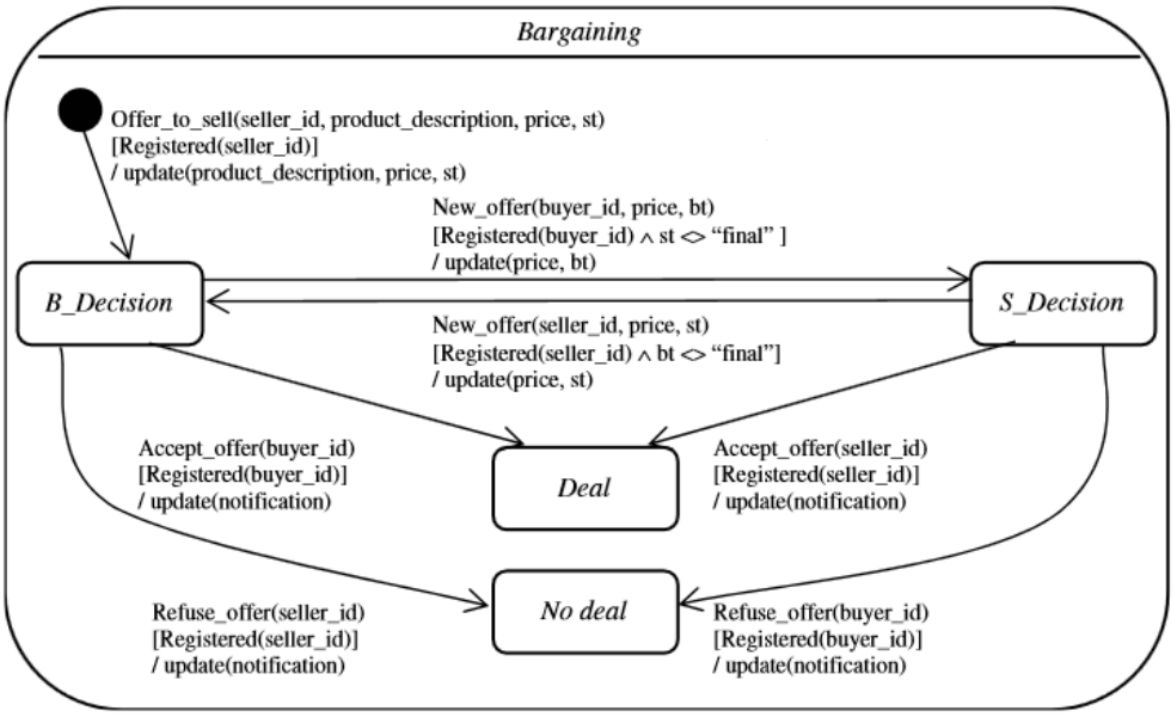


Figure 2-10: Bargaining (Rinderle and Benyoucef, 2005)

The most commonly used bargaining protocol used in the negotiation of web-services and cloud computing services is the 'Rubinstein alternating offers protocol' also termed as the 'Rubinstein bargaining model' which utilizes game theory. Game theory will be explained in the next section. In the Rubinstein alternating offers protocols, parties, cloud services provider and consumer, exchange their offers. If both provider and consumer agree to the offers, the agreement is done, otherwise, the process of the alternating offers from both sides

continues until agreement between the provider and the consumer is reached. In the worst-case scenario, if both parties are unable to agree to their offers, the negotiation process meets failure. The Rubinstein alternating offers protocol can be depicted in the self-explanatory Figure 2-11.

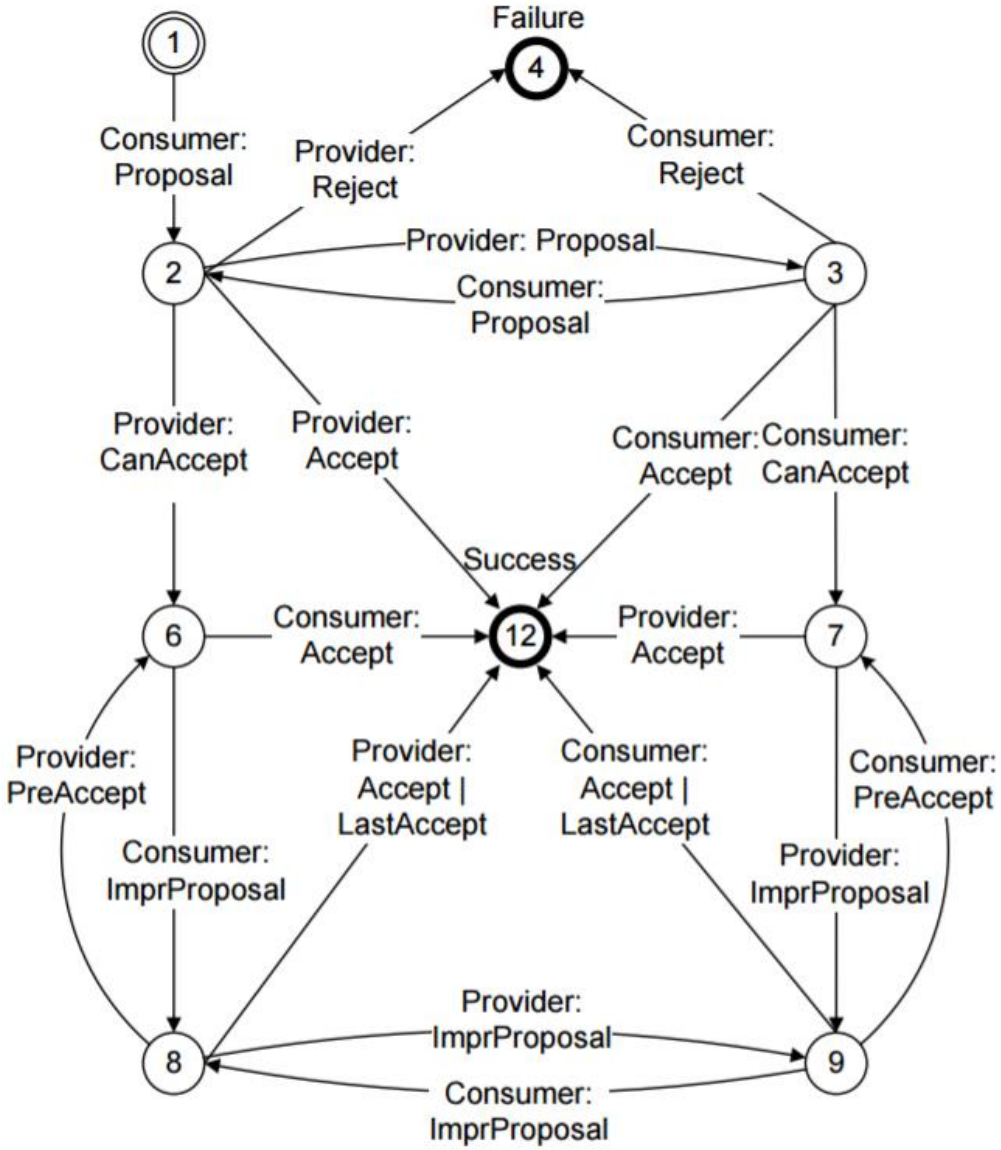


Figure 2-11: The Rubinstein alternating offers protocol (Holloway et al., 2015).

In this work, the Rubinstein alternating offers protocol is used because Rubinstein reported one of the critical findings in the bargaining process which can be simulated in the real-world. Furthermore, it does not incur any additional cost due to delays in transactions. The initiator/seller on the right side makes the initial offers, if the offer is rejected by the buyer on

the left side, then buyer on the left side makes an alternating offer to the seller. The buyer on the first side rejects the alternating offer of the buyer, then he/she makes another alternating offer and so on. The negotiation process simulates the game rules as argued by several scholars (Fatima et al., 2002; Kraus, 2001). The Rubinstein alternating offers protocol is widely used by other researchers in the game settings, so it is more suitable to be used in the negotiation process.

This protocol is also selected for this work due to its feature of making both players involved in the negotiation game infinitely patient and speeding up the process of the offers and counter-offers to reach the unique solution acceptable to both negotiating parties.

2.7. Negotiation and game theory

The process of negotiation takes place between two or more people. The people involved in the negotiation are called players or agents, and each player is allowed to perform certain actions by following certain rules as they do in any game (Brams, 2003). Hence, the process of negotiation is similar to the game, and rules of the game. Based on Rubinstein alternating offers protocol, each player can perform the following actions during the negotiation as a game: make an offer, accept the offer, reject the offer, and end the negotiation process (Binmore and Vulkan, 1999). In addition, the attitude of the players during negotiation plays a critical role in making the negotiation successful. For example, if both players cooperate, then both will gain high utility. In this case, the negotiation benefits both parties equally (Jennings et al., 2001). However, if one player shows an inclination to cooperate, while other player is in a mode of competing with his/her counterpart, in this situation negotiation will benefit the competing player with high utility, while the cooperating player will end up with low utility (Fatima et al., 2004). Nevertheless, if both parties are in the mode of competing

with each other, it means no player will give in. Therefore, the negotiation will not be able to make the agreement between two players (OBox, 2003).

2.8. Negotiation and Utility Theory

Utility theory was formulated by Von Neumann and Oskar Morgenstern (1944), which is considered a vital part of the game theory and negotiation, as the ultimate aim of the negotiation is to satisfy all parties involved in the process. However, the negotiation process fails, if the utility for each party is not maximized after the negotiation process. Utility is referred to as the 'preferences' of every player taking part in the negotiation. Thus, the agreement reached through negotiation should satisfy the principle of utility theory to maximize the utility for the agents/players involved in the negotiation. Quiggin (2012) defined the utility maximization as a main principle of the utility theory in this way: "the method of modelling choice by assuming that individuals' preferences can be represented by a utility function which they seek to maximize". Mathematically speaking, the utility actually represents the weighted sum of each individual evaluation value, and is calculated using the following formula (Alsrheed, 2014):

$$Utility\ (v1, \ldots., vn) = \sum_{i=0}^{N} wi \frac{Eval\ (vi)}{Max\ (Eval\ (vi))}$$

In the above equation, *v1,....,vn* represents the values of each issue presented in scenario, *wi* is the weight of chosen value of the issue, *N* refers to the total number of values of issues included in the scenario, *Eval (vi)* is the evaluation value assigned to an individual value of each issue.

2.9. Pareto efficiency/optimality

According to Lia et al (2008), the Pareto efficiency can be defined "as a property that an outcome cannot be further improved (i.e. no agent can get more utility) without sacrificing the other's utility". Thus, the pareto optimality or efficiency makes sure that all utilities are distributed evenly over both parties, and the end result appears in the form of satisfaction of both agents. The Pareto efficiency is feasible in the negotiation when both agents cooperate with each other and know each other's preferences (Robu et al., 2005). However, it is more difficult to achieve in the negotiation process which involves agents focussing on their self-interests and not intending to disclose their preferences (Lia et al., 2008). Pareto optimality is represented with different values on the pareto-optimality curve. All offers made by agents are shown on the curve. The counterpart will choose the offer satisfying his/her interests. The offers chosen in the pareto-optimal zone or near that area reflects the condition that both parties have gained maximum utility during the negotiation process, and no utility is wasted (Holloway et al., 2015). The Figure 2-12 shows an example of pareto-frontier showing all pareto-optimal solutions for both agents in the negotiation.

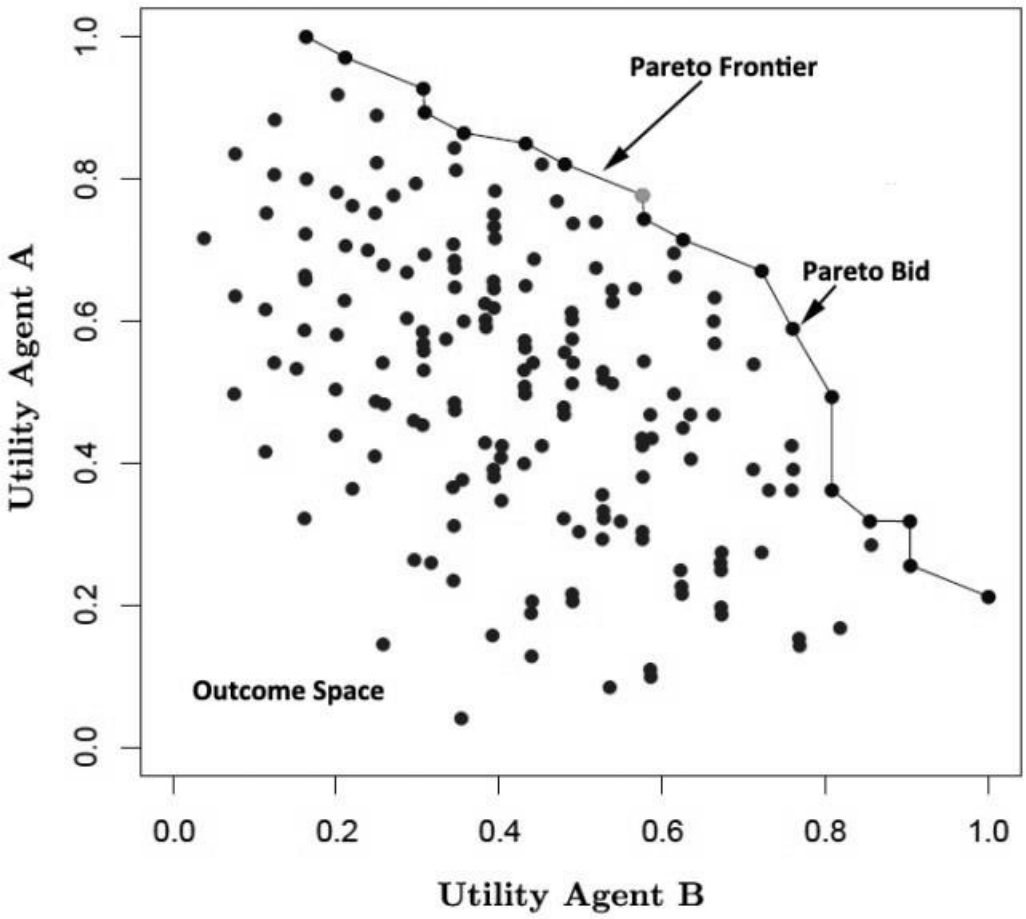


Figure 2-12: Pareto frontier curve with all possible pareto-optimal bids for agent A and agent B (Dirkzwager, 2013)

2.10. Nash Equilibrium

The 'Nash equilibrium concept' was introduced by Nash (1950), according to which 'a set of strategies (one for each player) constitutes Nash equilibrium if no player has an incentive to change their strategy given the strategies chosen by the other players' (John et al., 2009). Nash point refers to the best optimal solutions for any party on the Pareto frontier, where no party can gain more than the other party in the negotiation. Nash equilibrium can be gained at Nash point on the Pareto frontier.

The negotiating agents aim to optimise their utility/payoff (which may ideally lead to a Pareto-optimality) by selecting the most appropriate negotiation strategy (which may ideally use a Nash Equilibrium strategy). Hence the pareto-optimality and Nash equilibrium are the

key functions to determine the benefits gained by cloud providers and cloud customers out of negotiation process.

2.11. Conclusion

The background of cloud computing and SLAs is discussed in detail, which showed that cloud computing has emerged as a fundamental unit for offering services to the businesses and the individual customers in the form of infrastructure-related services, software-related services and platform-based services. The three-layered computing service models are described with emphasis on the core components in each layer. Therefore, the delivery of services to the clients from the cloud providers is regulated under the SLA in the constantly evolving cloud market. The core concepts related to the SLA such as core components of SLA, SLA lifecycle covering the creation, operation and removal phases are described. The negotiation is an important phase which occurs prior to reaching an agreement on SLA services beneficial for both parties. The negotiation process is controlled by the utility function which works to maximize the utility of the transacting parties in the negotiation and pareto function shows the line on which the offers are exchanged between the parties, while the Nash point on the pareto frontiers denotes the point where the parties can maximally benefit from the negotiation outcomes.

CHAPTER 3: RELATED WORK

In the previous chapter, background was introduced. This chapter presents the literature related to our research project. The chapter has been divided into five sections. The negotiations frameworks are presented in section one; while the issues relating to SLA and monitoring frameworks are discussed in section two. The overview of negotiation support systems and its relevance to the current research project is highlighted in section three. Section four describes the main autonomous intelligent negotiation agents. The gaps in the literature and contribution of our work are described in section five. The chapter is concluded in section six.

3.1. Negotiation Frameworks

Several researchers have attempted to develop the negotiation framework as an initial step for agreeing to the SLAs between the customers and service-providers. Mostly negotiation frameworks are efficiently developed and implemented in the web-service market, however, the research relating to solving the multiple issues encountered by the cloud market is still in its infancy. Yan et al (2007) developed a negotiation framework for agreement to the services and quality parameters related to the Web-Service Level Agreement (WSLA) which was aimed to resolve issues relating to the development of a flexible SLA specification and monitoring framework in the area of provision of web-services to clients. The negotiation mechanism was placed between the web-providers and customers, and autonomous negotiation agents were involved in competing on behalf of cloud provides and customers to achieve the desired ends. However, the proposed negotiation framework was valid only for the provision of web-services. Ludwig et al (2005) proposed a negotiation framework containing three-layer architecture of agent-based negotiation in order to solve issues relating to the web-services level agreements (WSLAs) in the service grids. However, the

implementation was limited to a simple scenario in the practical web market. Similarly, Mach and Shikuta et al (2012) developed the negotiation and re-negotiation framework for consumer-provider SLAs for web-services, however, it cannot be extended to the cloud computing market due to different modes of service delivery followed by the cloud-computing market.

In relation to the negotiation frameworks in cloud computing, Wu et al (2013) endeavoured to present the novel automated negotiation framework which made use of the SaaS broker as "the one-stop-shop for customers to achieve the required services efficiently when negotiating with multiple providers". Nevertheless, the limited issues were considered in the negotiation framework which was only limited to SaaS layer in the cloud computing. The authors argued that customers prefer to use the broker channels in order to negotiate the cloud services, which is not a viable and mutually beneficial means of settling the issues relating to cloud services. The brokers might be more interested in concluding the negotiation agreement with poor quality of services for customers in order to secure the commission.

The customers' requirement revolves around securing the low-priced deal with good quality of services, while the cloud providers intend to sell their cloud services to customers in order to maximize the profits. These opposing interests from cloud providers and customers make the negotiation cumbersome via the use of brokers. In our work, we provide customers and cloud providers an opportunity to interact with each other directly in order to negotiate over the issues pertaining to the cloud services. Alsrheed et al (2014) employed an intelligent-agents-based autonomous negotiation framework for resolving issues of services and negotiation of services between the customers and cloud providers. The authors provided strong empirical evidence of operationality of autonomous agents, but the issues considered were limited, and were not derived from the real-world vendors operating in the cloud-market. Therefore, our study goes one step further to consider issues in the real market and

develop the negotiation framework which can be scalable to figure out a negotiation mechanism for resolving real issues between cloud provider and customer.

3.2. SLA and monitoring frameworks

Although a significant level of work has been done to design and standardize the SLAs between the customers and service providers, most of the work has been performed over the standardization of SLAs in web-service, which are often termed as Web-Service Agreements. There are two types of specifications for the SLAs in the domain of web-services, which include language-based and rule-based approaches. WS-Agreement has been specified and promulgated by Open Grid Forum for dispensation of web-services to the potential customers in the web-market (Andrieux and Czajkowski et al., 2004; Andrieux et al., 2007). They developed WS-Agreement framework which specified the different phases in the life-cycle of the SLA which included creation, expiration and monitoring of SLA parameters. Another study conducted by Keller et al (2003) has developed the WSLA framework for defining the services, monitoring the terms and conditions of SLAs for delivery and management of web-services. They used the flexible and extensible language for describing the SLA monitoring services which are automatically configured for SLA implementation. Lamanna et al (2003) specified the language-based framework called SLAng in order to define SLAs which can fulfil the needs of customers and service-providers in the web-market. Another approach was developed by Paschke (2005), which was called RBSLA. The RBSLA is implemented in the form of rules, which implements SLA in web-services, which provides rules for interchanging, maintaining, managing and executing SLA rules in contract and contractual rule sets. There are many other studies which have developed SLAs for delivery of web-services (Dan et al., 2003; Sahai et al., 2002; Riamondi et al., 2008).

The issue with SLAs developed for dispensation of web-services cannot be applied to cloud computing due to differences in the dynamics of web-services and cloud-services markets. The cloud-services market follows the pay-as-you-go format which is not the case in the web-services, therefore, the web-services agreements are not portable to the cloud-computing environment. There are a handful of studies which have attempted to develop the SLAs in cloud computing, but their implementation and application in the cloud market is not widely acknowledged due to inherent issues remaining unaddressed in the SLA frameworks. For example, Patel et al (2009) based the proposed SLA mechanism on the concepts and language derived from the WSLA. The monitoring framework was also incorporated for enforcement of SLAs in the service-oriented architecture. However, the SLA framework proposed by Patel et al (2009) lacked scalability and suffered some issues with management of SLAs in the event of resource limitation. Joshi et al (2015) proposed the SLA incorporated with a monitoring framework for managing the legal perspectives of cloud-services delivery to the customers in the area of semantic web technologies such as RDF and OWL, The proposed SLA was effective in describing in detail the SLA ontology and presentation of prototype but this framework was limited to managing and complying with cloud-related legal documents and policies on behalf of cloud-service providers during the delivery of services, however, it was not intended to manage and deliver the multiple SLA parameters regarding the quality of services.

On the other hand, Torkashvan et al (2012) proposed an SLA framework in the area of services provision between cloud providers and customers. The proposed SLA framework was called cloud-based SLA management which defined and supported different phases of the cloud management during the life-cycle of services at the management level. The lifecycle of the proposed SLA framework was fully described by authors, which was supplemented with the monitoring framework in order to monitor the provision of services

according to the agreed terms and conditions in the SLA between cloud-provider and customers. However, this work only supported the management of the SLA in the controlled environment which could not mimic the conditions in the real cloud market. The specification of the third party for conducting monitoring was not fully operational at different stages of the SLA. The current work intends to develop the SLA framework which would be operational from the negotiation phase to the monitoring phase, thus covering the comprehensive demands of customers and patterns of service provisions on behalf of cloud-providers.

Another study conducted by Nie et al (2012) providing SLA frameworks was actually the refined version of cloud-based SLA management proposed by Torkashvan et al (2009). They combined two models: the management model and the coordination model to ensure the quality of services at the deployment phase of the SLA between the customers and cloud providers. The coordination model was applied at the stage of interaction of a customer with multiple cloud-providers and the management model was used at the deployment phase of services with integration of monitoring functions. However, the monitoring function in their work was not well-defined in terms of specification of third-parties, and reporting of violation events. The authors could not determine whether the use of third party in the monitoring mechanism can be extended to the development, deployment, assessment and management stages of the cloud services.

Binu and Gangadhar developed the SLA framework for building trust between the cloud provider and customers. The framework was effective in resolving issues related to the workload, however, it could not address the core issues of the changing nature of the cloud market. The dynamicity of the cloud market can be comprehended through the automation of the SLA market, which was absent in the SLA framework developed by Binu and Gangadhar.

Moreover, the SLA framework only addressed the issues relating to the infrastructure as a service (IaaS) layer of the computing.

Binu and Gangadhar (2014) also made an attempt to develop the monitoring framework involving the third party, and incorporated the monitoring framework in the main SLA framework in order to ensure the smooth flow of services and implementation of the SLA framework. The monitoring framework was tested on a case study in the real environment. The main issue with the monitoring framework was that it was not automated, and was only functional to resolve the issues related to workload. In addition, the monitoring framework was only applicable to the hardware services in cloud computing (IaaS).

El-Awadi and Abu-Rizka (2015) proposed an SLA framework for mapping the priorities and interests of customers and cloud-providers at the negotiation phase only, and they did not extend the SLA framework to the deployment phase of the cloud computing market. The real-time market situation was not simulated during the negotiation phase. The current work considers the offerings from the cloud provided as per the conditions and requirements of customers and cloud providers in the real market. In contrast to the previous works, the current work intends to integrate the monitoring framework in the automated environment in order to smooth the evaluation and measurement of SLA metrics at the deployment phase.

Taken together, the afore-cited data showed that there are multiple issues with existing SLA frameworks including the inefficient monitoring mechanism, the lack of automation, incompatibility with real-cloud market environment and the lack of sufficient evidence supporting the operational nature of the SLAs based on WSLA concepts and language in the context of cloud-services which have relatively different language and modus operandi to be dispensed to the end-customers. Therefore, there is a need to develop comprehensive solutions leading to SLA development with automated monitoring functions. This study

intends to use the autonomous intelligent agents to develop and maintain SLAs in order to ensure the quality of services for customers.

3.3. Negotiation support systems

Negotiation support systems (NSS) are the tools which "facilitate the various phases of the negotiation process such as understanding the negotiation case, assigning preference ratings for negotiable issues and options, and setting the reservation level before the negotiation begins" (Kersten and Lo., 2001). The NSS tools derive their decision-making capabilities by using the following strategies from the decision-making science methods, including decision-tables, decision-trees and multi-attribute theory, and game theory (Kersten and Lai, 2007). Lim (2003) argue game theory and decision analysis are key components of the NSS which play an important role in structuring and re-structuring negotiation events. There are some important NSSs which are developed with the purpose of negotiating the deals between the service providers and customers in the e-commerce and e-business domains. For example, the INSPIRE system has been employed for investigating cross-cultural negotiations over the web, and carries three modules: pre-negotiation, conduct of negotiation and post-settlement, and uses the Pareto-optimal functionality for reaching decisions. The utilities of both users are considered during the negotiation (Kersten and Lo, 2001).

The ASPIRE project was designed to improve the functionalities of INSPIRE, and aimed to automate the negotiation system using the autonomous intelligent agents which guide the users to perform a certain set of actions in order to improve decision-making ability of the customers and service-providers (Kersten et al., 2003). Of note, ASPIRE could not fully automate the negotiation process, and was restricted to providing some useful suggestions to customers and service-providers for finalizing the deals. The system is, however, fully aware

of the status of negotiation, implements the negotiation strategies based on objects and weights of issues assigned by the users.

Another NSS tool is OPELIX which allows users, customer and service provider, to conduct a fully automated bilateral negotiation (Hauswirth et al., 2008). The actions performed by the OPELIX tool are discovery of product offerings, negotiation between both users and delivery of products to customers. The main drawback of this system is that it is only useful for bilateral negotiations, and does not allow the users-specific sophisticated negotiation protocols to be implemented in its working environment. ASAPM is another multi-agent NSS tool which uses the FIPA Iterated Contract Net Protocol for negotiation between the users. Agents can conduct multiple rounds of negotiation while achieving the better outcomes out of the negotiation event, which ensures the quality of services (Chhetri et al., 2007). There are limited options of setting the users' preferences and utilities, and users cannot develop their own negotiation domains based on changing needs of businesses.

Kasbah is a sophisticated NSS which allows the buyer and seller to create their own agents which can send the bids to each other, and the opponent responds in the form of 'YES' or 'NO'. The buyer sends the bid, and seller responds it with an appropriate deal (Chavez et al., 1997). The E-Agora project was designed with a view to simulate the complex marketplace, which allows users to interact with each other through autonomous intelligent agents (Chen et al., 2004). The negotiation protocols are offered to users which increase their abilities to start bidding. The negotiation over E-Agora involves complex activities and phases and requires skills and understanding to handle the process of negotiation ingenuously.

Though the above-described NSSs are useful means of conducting the negotiation between the customer and service provider, however, they lack the element of flexibility which is the cornerstone for organizing the autonomous negotiation agents using the intelligent agents. The users are unable to customize their negotiation strategies and behaviours in the course of

negotiation. In addition, they are improved for implementation within the specific domain of the business such as web-based negotiation and auctions. Many scholars have argued that the practical utility of the above-described NSSs is limited, and required specialized knowledge on behalf of users before embarking upon the applications of these systems in the business environment.

3.4. Automated negotiation agents

In this section, the state-of-the-art automated negotiating agents are outlined and we will show the strategies used by various agents during the negotiation with their opponents. Overall, application of three components common to all of the intelligent agents including the acceptance strategy, bidding strategy and opponent modelling. In the acceptance strategy, the agents consider whether the offers made by the opponents are acceptable. In the opponent modelling strategy, the agent chooses the set of bids which can be proposed in response to the bids from the opponent, while the learning about the preferences and behaviour of the opponent is made via the opponent modelling (Baarslag et al., 2012).

3.4.1. Hard-headed agent

This agent starts the negotiation process through computation of all possible bids to maximize its utility, and these computations are stored in the repository using binary tree data structure from where the agent can easily recover the bids (Fukuta et al., 2016). The hard-headed agent makes use of the hard-headed learning module to learn about the opponents' preferences and utilities and weight of each utility. In order to do so, the hard-headed agent makes two important assumptions about the opponent (Fujita et al., 2017). The first assumption is that the opponent will not repeat the same bid again and again during

negotiation. The second assumption is that the opponent will throw the offers in a limited range, which means that opponent will not be flexible enough to accommodate the preferences of the hard-headed agent. This function of the hard-headed learning is also termed as "greedy reinforcement function" (Baarslag et al., 2013).

Through this function, the agent is updated about the utility values and their weights each bidding session. Moreover, the hard-headed agent computes the most valuable bid and identifies the least valuable bid for the opponent through this learning function (Fujita et al., 2017). Therefore, the hard-headed agent tries to throw the most valuable bid for the opponent at the end of negotiation session, so that it is more likely to be accepted by the opponent. This agent offers bids fast due to low computational complexity associated with simple learning module and optimal concession function (Fujita et al., 2017; Fujita et al., 2013).

3.4.2. Nice Tit-for-Tat Agent

This agent is known for using the tit-for-tat strategy to set its own utility. It uses a cooperative strategy in the beginning of the negotiation process, and changes his/her biding strategies depending on the response from the opponent, and tries to reach the Nash point in negotiation. The ultimate goal of this agent is to reach Nash Point by choosing the Pareto optimal value as quickly as possible. Moreover, this agent uses the Bayesian opponent model to learn about the utilities and weights of the opponent. This model helps calculate the estimated Nash point of the negotiation and tries to target these points. All the beneficial moves made by the opponent are updated to the agent who compares them with his/her utilities. It tends to give the same amount of concession as is given by the opponent, and tries to make the offers as attractive and lucrative for the opponent as possible (Baarslag et al., 2011)].

This agent concedes according to the level of concessions made by the opponent, and shows nice behaviour in the sense that it does not show the retaliatory approach towards its opponent. Whenever the opponent offers a bid with lower utility than the target utility of the nice tit-for-tat agent, then the agent thinks that the opponent made a mistake and wait for a better bid. In addition, if the negotiation domain is large or the discount factor is high or the negotiation end deadline is approaching, the nice tit-for-tat agent offers higher concessions to its opponent (Baarslag et al., 2013)

3.4.3. Hardliner Agent

This agent is termed as the one with a selfish and stubborn approach, and acts on the principle of serving his/her own interests (Baarslag et al., 2011). Therefore, this agent starts the negotiation process by offering the bid with highest utility for itself. It keeps repeating the same bid again and again assuming that the opponent will concede in the end of the negotiation. It follows the strategy "take-it-or-leave it" to maximize its utility, and does not allow any concession to the opponent. It gives the full time of negotiation to the opponent to think about the offer and accept it (Baarslag et al., 2012). Notably it does not follow any learning model or mechanism to learn about the utilities of the opponent (Ibid). The strategy of the hardliner agent is used nowadays by most of the cloud providers in the market such as Google, Amazon SE3 and Microsoft Azure because their approach of selling cloud services to consumers is "either take it or leave it" (Baarslag et al., 2013).

3.4.4. IAMHaggler Agent

IAMhaggler is implemented under the framework of SouthamptonAgent which is a set of methods used by the relevant agents to manage the negotiation time, proposing bids and

accepting or rejecting bids during negotiation (Williams et al., 2012). IAMhaggler uses a fully-fledged negotiation strategy which is based on Bayesian learning to model the opponent. The algorithm for this agent has been explained in different publications, and for the first time it was included in the ANAC 2010 competition (Baarslag et al., 2012). This agent starts with the offer carrying maximum utility for the opponent. With the passage of time, it adjusts the utility of its offers to the opponent based on the utility of the opponent's offers, time remaining from deadline of negotiation session, and profile of opponents such as hard-headedness, and hardliner (Williams et al., 2014).

When it receives the opponent's offer, it analyses the previous offers from the opponent in order to adapt its utility to the opponent's utility. Using the above model, this agent always seeks to hit the trade-offs acceptable to the opponent. Let 'u' be the utility of the offer made by the opponent during the negotiation process. Both agents accept the offer only in the satisfaction of one of the following three conditions: "1) when 'u' is at least 98% of the utility of the previous offer; 2) when' u' is at least 98% of the utility of its upcoming offer; 3) when 'u' is at least 98% of maximum aspiration constant" [default value is 0.9] (Baarslag et al., 2013). If the offer either from the opponent or IAMhaggler meets the above criteria, both agents accepts the offer, otherwise, the offer is rejected, and the process of counter-offers continues until and unless the three-point criteria are met.

3.4.5. Nozomi Agent

Nozomi follows the proposal strategy which is based on giving initial offers with maximum utility with the intention to gather knowledge about the preferences and utilities of the opponent. It calculates the difference between the utilities of the last offers, thereby creating a history of utilities, based on which it determines the gap in utilities and time left in

negotiation to calculate its utility values for the future offers (Barslaag et al., 2014; Baarslag et al., 2012).

For example, based on the time left and the gap in utilities, it may alter its utility for future bids, make compromises or continue with the existing utility values. Although it keeps track of the opponents' utilities, it does not model their utilities to infer their behavioural patterns. Hence it has no ability to predict the future moves of the opponent (Baarslag et al., 2012).

Nozomi usually tends to split the negotiation time into different negotiation segments which are located at three points in the negotiation time frame: 50%, 80%, and 90%. The acceptance model of Nozomi decides whether to accept or reject the offer based on the time left from the negotiation, and gaps in the utilities of parties involved in negotiation (Kawaguchi et al., 2011).

3.4.6. Yushu Agent

Yushu has its simple mechanism for setting its target utility which is further used to propose the future offers to the opponent in the negotiation process. For setting its target utility, Yushu tracks the last 10 best offers from the opponent, which are called the 'suggested proposals' (An and Lesser, 2012). Additionally, it counts the number of negotiation rounds left in order to decide about the utility value for the next offers. Hence, the target utility of Yushu is the function of the negotiation rounds left and the best moves/offers made by the opponent in the bidding history (Baarslag et al., 2012).

Moreover, Yushu also determines its acceptability rate, which means that it determines the minimum utility which can be accepted (Kawaguchi et al., 2011; An and Lesser, 2012). As a first step towards setting the acceptability-rate, Yushu seeks to explore the best possible options in terms of accepting the best offer with utility closer to its target utility, and sets the limit of accepting the offers with utility sharing 90% closeness to its target utility. However,

if the number of rounds left from the negotiation process are shorter, then it lowers the percentage to 92% in order to accept the utility (An and Lesser, 2012; Baarslag et al., 2012).

3.4.7. Meta-agent

Meta agent was designed by Ilany and Gal (2016); it uses the target utility setting mode for choosing the right target utility which it will pursue throughout the negotiation process. It collects information from the agents' previous bids, utilities repeated and preferred by opponents in order to model the opponent's behaviour.

Hence, it is very important for meta-agent to receive the first offer from the opponent for setting the target utility. If meta-agent is a proposer, then it creates an offer with intention to secure maximum utility. Nonetheless, if the meta-agent is a responder, it receives an offer from the opponent, and which will be used by it to calculate the first proposal features. The calculation of proposal features designed for the opponent during negotiation is dependent on the feature list for the negotiation domain.

Hence the feature list of the opponent, including utility and preferences as inferred from its bids, is employed to predict the future performance of the opponent, which is subsequently used by meta-agent to develop the acceptance strategy. The meta-agent does not accept an offer with utility lower than its target utility unless there are time restrictions. In the absence of the first offer from the opponent, it creates bids with maximum utility knowing that there is a maximum probability that the opponent will accept the proposed offer.

3.4.8. Fawkes Agent

Fawkes agent employs opponent modelling in order to compute its target utility. It takes into account different types of information about the opponent such as number of bids offered, time difference between different offers, and utilities attached to various offers at different

time points during negotiation. Based on the collected information, it models the utility which will be employed by the opponent in its future moves (Koeman et al., 2015).

The bidding strategy developed by fawkes is based on securing the maximal utility for its client. The weights and preferences assigned by the opponent to various issues in the negotiation domain are taken into consideration while calculating the target utility for Fawkes (Fujita et al., 2015). The target utility is calculated for each bid, which maintains the flexible and social behaviour of fawkes towards its client. The estimated utility is based on estimates of the bids likely to be made by the opponent, if the estimated utility exceeds the reserved utility, it is the optimistic scenario for fawkes (Chen et al., 2014).

If Fawkes receives all bids from the opponent with utilities below the estimated utility, it may consider giving a concession in order to reach an agreement with the opponent, however, the range of concessions has a narrow band, and most likely, the negotiation ends in failure (Koeman et al., 2015). Therefore, fawkes has some flaws in its acceptance strategy in the pessimistic scenario when utilities received by fawkes are not close to the target utility or estimated utility (Chen et al., 2014).

3.4.9. CUHK Agent

CUHK agent uses the non-exploitation strategy which aims to give concessions and benefits to the opponent based on its behaviour during the negotiation process. It is also called win-win strategy as both negotiating partners end up securing the beneficial negotiation outcomes at the end of the negotiation process (Marsa-Maestre et al., 2014). This strategy was presented in the 4th ANAC competition, and was winner of the competition in terms of increasing the utility, social utility and social welfare (Barslaag et al., 2016; Ilany, 2015).

CUHK agent uses the adaptive bilateral negotiation strategy which allows the agent to adapt to the dynamic conditions prevailing during the negotiation process. It determines its acceptance threshold based on Bayesian modelling of the behaviour of the opponent. It's

throws in the bid with highest utility at the beginning with the hope that the opponent will accept it, but gradually decreases its utility by giving discounts, though with little reductions, in the face of the hardliner approach of the opponent (Hao and Leung, 2014).

CUHK chooses its target utility based on the behaviour of the opponent. The hardliner opponent elicits the greater discounts, but simultaneously, the CUHK maintains its utility near the points of its target utility. As the time of negotiation approaches to the deadline, CUHK observes the behaviour of the opponent closely, and offers discounts if the latter throws offers/bids with concessions (Marsa-Maestre et al., 2014).

Hence, CUHK and the opponent are in a position to obtain the lucrative deals at the end of the negotiation. Near the deadline, CUHK offers all bids with utility closer to its target utility while considering the past behaviour of the opponent, and chooses the most appropriate utility from the opponent's bidding history to finalize the deal. It does not exploit the nice behaviour of the opponent, instead it rewards its opponent which is more interested in lowering its utility (Hao and Leung, 2014).

3.4.10. ValueModel Agent

Frieder and Miller (2013) developed ValueModel agent as an entrant into the ANAC2011 bilateral negotiation competition in order to participate in negotiation with other agents. This agent uses the 'a priori approximations' for arranging the opponents' bids which are used to create the opponent's preference profile. During negotiation with its opponent, ValueModel agent keeps its utility minimal and creates and offers bids with utilities above the threshold value. The maintaining of the threshold is dependent on the elapsed time during the negotiation process (Frieder and Miller, 2013; Baarslag et al., 2013).

In the beginning, the threshold is set to 0.98; as the time elapses to 80% of the negotiation, it uses a concession strategy leading to giving concessions to the opponent. If the opponent gives 50% concession determined from the average of the opponent's last 5 bids,

ValueModel agent lowers its utility. ValueModel agent also lowers its utility if the opponent stops lowering its threshold near the deadline (Frieder and Miller).

As the time to deadline approaches its last 10%-time segment, ValueModel agent lowers its threshold utility by 0.02, until the utility value reaches 0.7. After reaching 0.7 utility, ValueModel agent stops lowering its utility (Frieder and Miller). After elapsing of 90% of the time, ValueModel also employs 'scare tactics' to convince the opponent to accept the proposed bids. Another tactic used by ValueModel agent to win the best deal is to sleep three times without offering any bid after passing 50% of the negotiation time, followed by giving a bid to its opponent above its threshold. It may lower its threshold utility to 0.65, and if the opponent does not concede it further lowers the threshold utility to 0.6 depending on the behaviour of the opponent. If the best bid from opponent is above 0.55, it keeps proposing, however, if it is below 0.55, then it resumes bidding above the threshold (Frieder and Miller).

3.5. Metrics measuring performance of Negotiation agents

There are three key metrics which are used by various studies to determine the performance of autonomous agents during the negotiation,

3.5.1. Utility

Utility is the value assigned by the agent to bid, which is the total sum of utilities assigned by the agent to each issue with the negotiation domain. The higher the utility value, the higher the performance of the autonomous negotiation agent in terms of securing a better negotiation outcome for the client. The detailed discussion on the utility and calculation of utilities is done in Chapter 2, section 2.9.

3.5.2. Social utility

This is a fairness metric which shows the total beneficial outcome gained by the agents participating in the negotiation process over certain issues in the negotiation domain. This metric is usually applied to measure the benefits gained by the agents collectively through participation in the negotiation bilateral or multilateral tournaments. It is measured by taking

the sum of utilities gained by different agents and total social welfare achieved in the tournament (Yaqub et al., 2014). Hence, the social utility reflects the joint actions taken by participating agents, and their beliefs learnt during the negotiation process about each other (Gal and Pfeffer, 2007).

3.5.3. Social welfare

This is another fairness metric employed to measure the overall advantages and benefits gained by participating agents in the tournament, and is a reflection of the social performance of the participating agents. The tournament with higher social welfare indicates the beneficial outcomes for all participating agents, and vice versa. The social welfare is measured by taking the averaged sum of utilities of participating agents in a negotiation tournament (Yaqub et al., 2014). The social welfare contributes to the Pareto efficiency (Ito et al., 2007).

3.6. Automated Negotiation Testbeds

For the researchers working on the automated negotiations, there are limited options for creating and testing the automated negotiation agents. In this work, as we are not sure which behaviour of agents would be productive for successful negotiation, that means that we have to use the testbed which offers plenty of options to test the behaviour of different agents using the hit-and-trial approach in order to conduct a successful negotiation between the cloud provider and customers. There are two important testbeds developed for negotiation over the technologies, which are the Agent Reputation and Trust Testbed (Fullam et al., 2005) and Generic Environment for Negotiation with Intelligent multi-purpose Usage Simulation (Lin et al., 2005).

The Agent Reputation and Trust Testbed is widely used for negotiation over technologies in order to build trust on agents in trust-related technologies, as Fullan et al (2005) posited that it is a “testbed initiative which has been launched with the goal of establishing a testbed for agent reputation – and trust related technologies”. The ART testbed is considered to be

effective in offering the easily accessible with customizable format for researchers who can conduct different negotiation experiments in order to explore the most viable solutions for trust-related technologies. The major limitation of the ART platform is that it is only applicable for experimentation related to the trust and reputation of technologies. As the current project is intending to explore the most viable solutions for cloud-services to be dispensed based on the offerings of cloud services and requirements of the customers, therefore, ART testbed is not a suitable platform for the current research project.

On the other hand, GENIUS platform is the testbed initiative which is used for designing new agents which can compete with the agents designed by other researchers based on the needs of the customers and cloud providers. It has a user-friendly interface and design with customizable features. For example, researchers can specify their own domains based on the issues which need to be competed on in the negotiation. The flexibility in terms of design, the size of negotiation domains and determination of preferences are the main features which make it more attractive for conducting the negotiation on the evolving needs of customers and cloud providers. GENIUS is also regularly employed for negotiation competition using the automated negotiation agents in the well-known Automated Negotiating Agents Competition (ANAC) annually. The ANAC offers the opportunity to researchers to test the bidding and acceptance strategies via the design and test of novel negotiation agents, which makes it a repository of the plethora of agents using varied bidding and acceptance strategies.

In addition, the state-of-the-art negotiation agents with opponent modelling and learning strategies, negotiation domains and preference profiles are freely available for the researchers working in the negotiation of services area (ANAC, 2010). This provides opportunity for researchers to either design their own automated negotiation agents or use the existing negotiating agents in the repository to develop new strategies using the mix-and match approach. Based on the suitability of the GENIUS for achieving the research questions in this

study, we have used GENIUS to conduct negotiation experiments in the SLA development phase.

3.7. Research Gaps and contributions of the study

The following research gaps were found in the previous work, which need to be addressed by this study. Therefore, the current work will make important contributions to the existing knowledge on negotiation and monitoring strategies in the cloud computing market.

The cloud computing market is dynamic in terms of offerings in cloud services and evolving the needs of customers. The heterogeneity of the cloud computing market demands the review and improvement in the existing SLA lifecycles for the related technologies, and hitherto develop the novel automated SLA framework which can fit into the heterogeneous nature of cloud computing.

With the growing market of cloud computing globally, there are multiple issues which need to be considered during the negotiation between the cloud provider and customer. This suggests the requirement for organizing the multiple issues-based negotiation which can increase the ability of negotiating agents to negotiate over an array of preferences for each issue.

In addition, the negotiation process should not be in a controlled environment as it will not be able to mimic the ground realities in the business of cloud computing. The flexibility is a key requirement for any negotiation event to yield the purposeful end, which is not fully achieved in the controlled environment. The negotiation between the customers and cloud-providers can be made speedy and purposeful through the use of intelligent agents who can truly serve as representatives for winning the maximum benefits for customers and cloud providers.

With the growing number of customers and cloud providers, the negotiation needs to be organized between multiple cloud-providers and customers, so as the best deal for the

customer can be achieved within a short period of time. Moreover, the customers should have the choice of re-bargaining with the cloud-providers based on the increasing demand for services from the cloud-providers.

There is need for finding or developing novel negotiation intelligent agents and design the negotiation experiments in order to test the abilities to compete on behalf of customers and cloud providers in order to secure the best possible deals. The novel bidding and opponent modelling strategies can help the intelligent agents to understand the negotiating tactics or behaviour shown by the opponents in the course of negotiation events. The first experiment will be designed to learn the negotiating behaviour of intelligent agents; the second experiment will find the best bidding and opponent modelling strategies which can increase the capacity of intelligent agents to win the best outcomes for the customers or cloud-providers. The third experiment will be about testing the negotiating capability of the proposed intelligent agent in the real-time scenario of the cloud-market.

There is a need to conduct the monitoring of SLAs in the real time environment, so that any issues encountered by customers can be resolved quickly in order to comply with the terms and conditions outlined in the SLA, and for ensuring the quality of service to the customers.

3.8. Conclusion

Based on the literature review related to the negotiation, SLA, and monitoring frameworks, it was evident, there is a plenty of opportunity for researchers to resolve the issues in relation to contents of SLA, to design more generic and robust negotiation frameworks, so that process of SLA can be speeded up between the customers and cloud providers. Although several research endeavours have been made previously to address the issues in the context of negotiation, SLA and monitoring, they were mostly directed to the provision of web-services to the customers. However, the evolving nature of the cloud market demands the continuous

research to consider the emerging needs of customers and cloud providers for developing the robust solutions for negotiation, SLA development and monitoring the implementation of SLAs for ensuring the quality of services. The next chapter will provide an overview and stages of a novel SLA framework which is developed to address the issues which are not thoroughly considered by previous studies.

CHAPTER 4: DEVELOPMENT OF AUTONOMOUS SERVICE LEVEL AGREEMENT FRAMEWORK

In the previous chapter, the literature review was conducted in the area of issues relating to the cloud services and service level agreements (SLAs), and it was revealed that there is a need to develop the autonomous SLA framework which can come up to the demands of the evolving nature of the cloud computing market and customers' requirements. In this chapter, different stages designed as part of the autonomous SLA framework are presented. The inputs to, and outputs from, each phase in the proposed SLA framework are highlighted.

4.1. User scenario

The user scenarios, which can be either for cloud computer provider or customer, are designed to provide the overview of users' issues, goals and objectives which they intend to achieve through the negotiation process. In this section, an example of the user's scenario will be given.

4.1.1. Cloud provider's scenario

The cloud provider's offerings for the cloud customer are represented in the form of issues. The provider gives each issue evaluation and weight values in order to compute the utility. The evaluation value represents 'the preference' given by cloud provider to each value of an issue. For example, if the cloud provider gives '100' value to 'Windows OS' value, it means that he/she will give 100% preference to 'Windows OS'. The weight metric is assigned by the cloud provider to each issue in order to show the degree of importance of that issue. Of note, the sum of all weights assigned to different issues in the scenario is "1". The table 4-1 shows the cloud provider's scenario:

Table 4-1; The cloud provider's scenario

Issue	Value	Provider evaluation	Weight
Operating system	Windows Linux	25 75	0.18
Pricing plan	Hourly Monthly Annual	20 30 50	0.19
Delivery method	Amazon Machine Image CloudFormation Stack SaaS	30 50 20	0.18
Architecture	32-bit 64-bit	50 50	0.13
Availability Region	US East (Ohio) US West (Oregon) Asia Pacific EU (Frankfurt)	10 10 30 50	0.18
Memory (GB)	5 6 7 8	10 30 50 10	0.05
Storage (GB)	250-500 501-700	25 75	0.09

4.1.2. Customer's Scenario

The customer shows 'preference' for different values of issue by assigning the 'evaluation number'. The higher the evaluation number for specific issue value, the higher the preference for choosing that specific value of an issue. Customer also uses the weight metric to show the importance of an issue in order to gain maximum benefit from the cloud package. The sum of values of evaluation for an issue is 100, and the total of the weights for all issues is 1.

A customer is searching for the cloud services – infrastructure as a service. The preference of customers involving the issues, evaluation and weight assigned to each issue is shown in the table 4-2.

Table 4-2: The cloud customer's scenario

Issue	**Value**	**Customer evaluation**	**Weight**
Operating system	Windows Linux	75 25	0.19
Pricing plan	Hourly Monthly Annual	50 25 25	0.10
Delivery method	Amazon Machine Image CloudFormation Stack	50 20 30	0.10

	SaaS		
Architecture	32-bit 64-bit	75 25	0.11
Availability Region	US East (Ohio) US West (Oregon) Asia Pacific EU (Frankfurt)	10 30 50 10	0.36
Memory (GB)	5 6 7 8	20 20 10 50	0.10
Storage (BG)	250-500 501-700	75 25	0.04

Based on the evaluation and weight parameters of the scenario presented in the above table, the most suitable package for the customer will be [Operating system: Win, Pricing plan: hourly, Delivery method: Amazon Machine image, Architecture: 32-bit, Availability region: Asia Pacific, Memory: 8GB, Storage: 250-500GB,]). However, the less desirable package to the customer will be the following: [Operating system: Linux, Pricing plan: Annual, Delivery method: Cloudformation stack, Architecture: 64-bit, Availability region: EU (Frankurt)/US East (Ohio), Memory: 7GB, Storage: 501-700,]).

4.2. The Weight of Issue

The customer and the provider should give weight to each issue representing how important each issue is to them. The importance of each issue varies for the customer and the provider.

That's why we see different weight value for the same issue in the same scenario with the customer and the supplier. For example, based on the weight parameters of the last scenario presented in table 4-2, the importance of issues for the customer will be arranged by: (First: availability region, second: operating system, third: architecture, fourth: pricing plan, delivery method, and memory, finally: storage). The sum of the weights of all versions must be 1.

4.3 Analysis of Users' requirements

The analysis of the requirements of the customer and cloud provider will provide the specifications of their functional needs and preferences in the context of cloud computing services. Both users participating in the negotiation process should be able to generate and update their specifications, decide upon their negotiation strategies or rules of negotiation. In addition, users should be able to amend or update their negotiation preferences and price policies, establish and update the rules and regulations for monitoring, view the monitoring outcomes and finally receive the monitoring alerts and deciding upon the punishment policies in the event of violations. The interactions between the customer and cloud provider at different stages of service request and delivery of services in the cloud market are shown in the Figure 4-1.

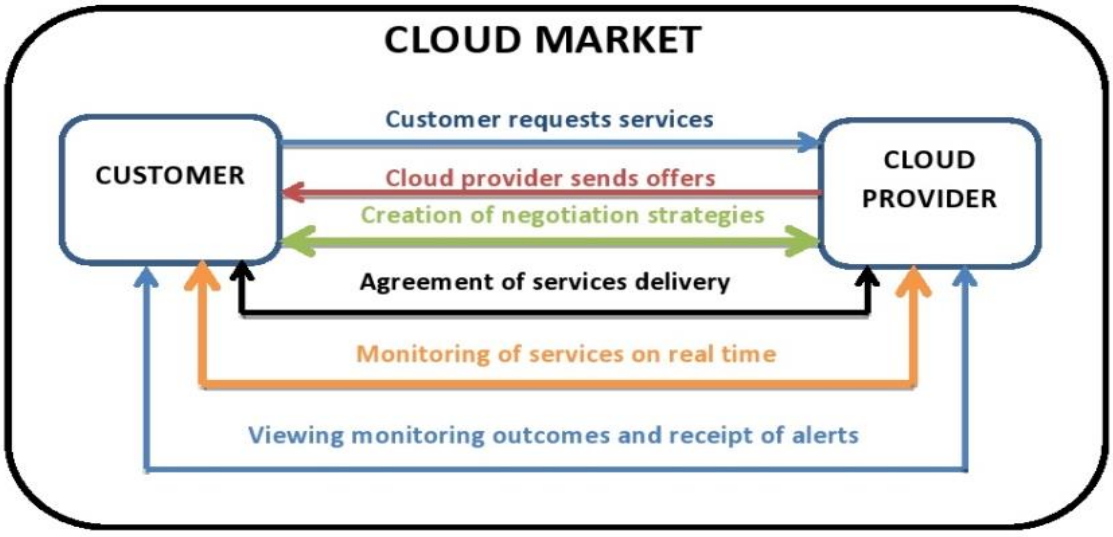


Figure 4-1: The complex communication patterns between the customer and cloud provider in the cloud market.

The requirements described above will be defined and fit into each stage of our proposed SLA negotiation framework. The details of each stage of the proposed framework will be explained in the next section.

4.4. Autonomous agent-based SLA framework

This research work will validate the automated framework which means that involvement of the customers and cloud providers will be less, and all the inputs from both will be retrieved in advance at the earliest stages of the process. The framework proposed for this research work will contain five fundamental stages to execute the SLA in an automated fashion, which are gathering, filtering, negotiation, agreement, and monitoring (Figure 4-2).

The output obtained in the earlier stage will determine the events in the next stage of the framework.

Figure 4-2: The five stage SLA model

4.4.1. Gathering Stage

In this stage, the priorities, preferences, requests and offers regarding the cloud services from both customers and cloud providers will be retrieved, and will constitute the input data. This data will be stored in the repository from where it can be retrieved and used easily for the negotiation. The framework will automatically update the input data stored in the database based on the changes in the customers and cloud providers negotiation preferences and strategies. Nevertheless, the data regarding the profile of users cannot be edited or changed by the users within this framework.

4.4.2. Filtering Stage

This is the second stage in the automated framework which filters the cloud providers based on the cloud customers' requests and preferences. This stage reflects the marketplace of cloud computing where the cloud providers may join or leave at any time, the preferences of the cloud customers and cloud providers can change any time during this stage. Therefore, this stage is highly volatile, dynamic and fast.

Initially, the customers register their requests through the proposed framework, and based on the criteria of the request, the framework will filter all possible cloud services providers matching with the requested criteria. The framework will also allow the customer to provide information about their preferred cloud provider, and some alternatives about their requested services. The output of the filtering stage is the provision of the cloud providers with whom the customers can interact to negotiate in the next stage.

4.4.3. Negotiation stage

At this stage of the framework, the customers receive the list of potential cloud providers and negotiate with each of them separately. The sessions of the negotiations will be conducted

between the customer and the cloud providers, and outcomes of these sessions will be compared with each other.

At this stage, the users (customer and the cloud provider) are represented by their respective agents. The agents carry out the negotiation on the behalf of their respective user. Because the users are different, each agent is characterised by a unique negotiation strategy depending on the supply and demand conditions of the market.

The cloud customer and provider choose the issues to be negotiated, and preferences for each issue to be established. With the passage of negotiation, the preferences might be changes for each user, therefore, the framework will be able to continuously update any change in the preference of each user involved in the negotiation process.

The cloud customer and provider set up a negotiation space which is located between the customer maximum price and the provider minimum price. This can be understood in terms of determining the minimum and maximum values such as price for a cloud service. The customer has the maximum price value which he/she can pay for a cloud package, which is represented by *Cmax*, and is the sum of values assigned to each item within a cloud package. Similarly, the customer has also set up a minimum price for a cloud package, which can be represented by *Cmin*. The *Cmin* is the sum of minimum prices values assigned by the customer to each item in a cloud package.

Likewise, the cloud provider also determines the maximum price and minimum price limits for the cloud package. The maximum price limit assigned by the cloud provider can be indicated by *Pmax*, and represents the sum of maximum price tags assigned to different items

in the cloud package. The minimum price limit attributed by the cloud provider to the cloud package is denoted by *Pmin*, and is equivalent to the sum of minimum prices reserved for each item in the cloud package.

As the cloud provider always thinks of taking the financial advantage from the customer, while the customer endeavour to search for cloud services which are cheaper, hence, it is assumed here that *Cmin* and *Cmax* reserved by the customer to the cloud package are smaller than the *Pmin* and Pmax set by the cloud provider. This assumption triggers the negotiation between the cloud provider and the customer within the negotiation space as indicated by Figure 4-3.

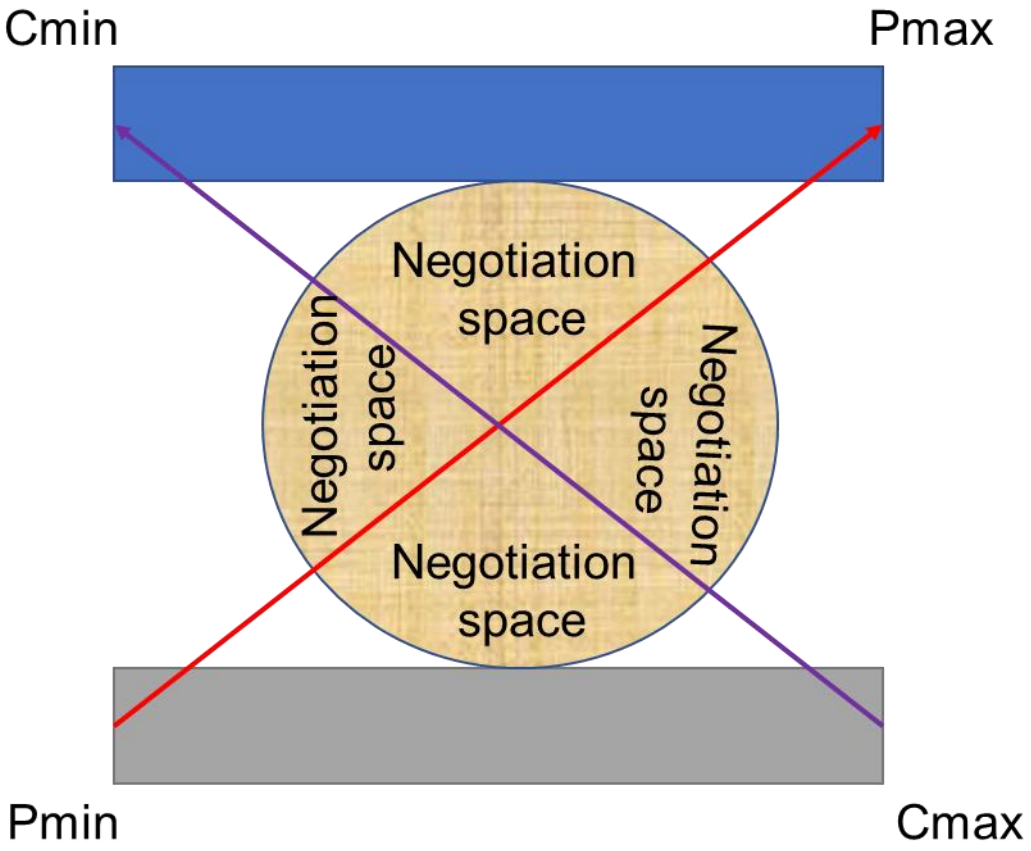


Figure 4-3: The negotiation space with display of potential actions of the customer and service

In the above figure, the customer tries to attain proximity to the minimum price value for a cloud package, while the cloud provider endeavours to become closer to the maximum price value during the negotiation between the cloud provider and the customer.

4.4.4. SLA agreement

At this stage, the customer and cloud provider will be notified of the SLA agreement along with measurable criteria governed by specific terms and conditions. The output of this stage will appear in the form of agreed services and metrics which need to be monitored in the next stage. The SLA metrics agreed at this stage will be response time, availability, memory, storage and bandwidth. The quality of services will be measured against foregoing metrics.

4.4.5. Monitoring

At this stage, the monitoring client will be used to monitor the agreement based on the agreed rules and regulations governing the SLA agreement. The breach of the agreement will invoke penalties for the provider. The users set up the rules for generating the alerts on the users' dashboard.

4.4.5.1. Setting and updating monitoring rules

The users are required to set up the monitoring rules in the beginning right after signing the end stage of the negotiation phase. The purpose of these rules will be to enable the monitoring client to issue the alerts at dashboard for users in relation to compliance with the promised service delivery associated parameters. The monitoring rules actually represent the SLA's policies covering the agreed parameters for assessment of compliance or detection of breaches on behalf of parties responsible for maintaining SLA in the workable condition. The

type of punishments will be determined by the users in the event of violation. The actions in response to violation may vary from the basic alert to the advanced action such as proactive adaptation.

4.4.5.2. Reception of monitoring outcomes

The monitoring outcomes will be viewable for both customers and cloud providers through the dashboard which will be made accessible by the monitoring agent to the relevant parties as mentioned in the SLA. The dashboard will show the alerts, updates, status of SLA, and description of the SLA.

The frequency of alerts and mode of delivery of alerts and notifications would be customizable for customers and cloud providers. The customer can opt for the delivery of alerts and notifications through either email or SMS on an hourly and daily basis. Similarly, the cloud provider will also be able to customize the frequency and delivery of the alerts for taking necessary actions in order to improve the quality of services.

The monitoring framework consists of three critical phases: In the first phase, agreed metrics from the SLA extracted to CSV (comma-separated values) which will be stored in the system's data centre. In the second phase, the cloud metrics data during use of cloud services will be collected and stored in the database. In the third phase, the CADA (Collect-Analyse-Decide-act) loop will be used to compare the data collected in real time use with the agreed metrics data extracted from the SLA. After analysis, if the violations are found, the Act component of the loop will impose penalties on the cloud provider, and provide instructions for correction of the issues.

4.5. Conclusion

This chapter has shown the various stages which are required for the automated SLA framework; these are explained with focus on the functional requirements. The gathering stage showed how the data from the customers' requirements and the services of the cloud-providers will be collected. The filtering stage can be operationalized through matching the customer requests with the potential (cloud providers) in the market. The negotiation between cloud provider and customer can be achieved in the negotiation phase in order to reach an SLA between the concerned parties. The quality of services and maintenance of the SLA will be monitored according to agreed rules and policies as per the SLA in the monitoring phase. The next chapter will present the design of an automated SLA framework with detailed description and functional requirements for each stage.

CHAPTER 5: DESIGN OF SLA FRAMEWORK

In the previous chapter, the customer's and cloud provider's requirements were outlined. Also, the different stages of the SLA framework were defined. In this chapter, detailed design of the SLA framework and its phases will be delineated along with diagrammatic illustration of the working mode of each stage of the proposed SLA framework. The working principle of the SLA framework will be illustrated in the form of closed functioning loop.

5.1. Illustration of Detailed Design Elements

In the previous chapter, it was described that the proposed SLA framework contained five main stages/phases: the gathering stage, filtering stage, negotiation phase, SLA agreement phase and monitoring stage. In this section, a detailed description of each phase of the SLA framework has been described. The figurative illustration of the SLA framework is shown in the Figure 5-1. The top part of the SLA framework represents the gathering phase one which is dependent upon the gathering of data from both customers and cloud providers about their preferences. The customers may have a variety of needs such as people from academia, research, and print media may have different preferences, interests and needs in terms of use of cloud services from the cloud providers in the cloud market. Similarly, cloud providers may differ in their offerings and quality of services desired by customers. The gathering stage determines the needs of customers and offerings of cloud services in the cloud market. After completion of gathering, the next stage is the filtering phase to match the customers' needs with cloud providers' services and offerings. Negotiation in the case of one or multiple cloud providers matching with customers' needs are arranged in the negotiation phase, while agreement is accomplished in phase IV. Once SLA agreement between cloud providers and customers is reached; monitoring of SLA parameters as agreed between cloud providers and

customers is conducted in monitoring phase. Figure 5-1 depicts the communication between different phases of the SLA framework, which are automatically connected with each other. The completion of the preceding phase is essential to have the subsequent phase initiated.

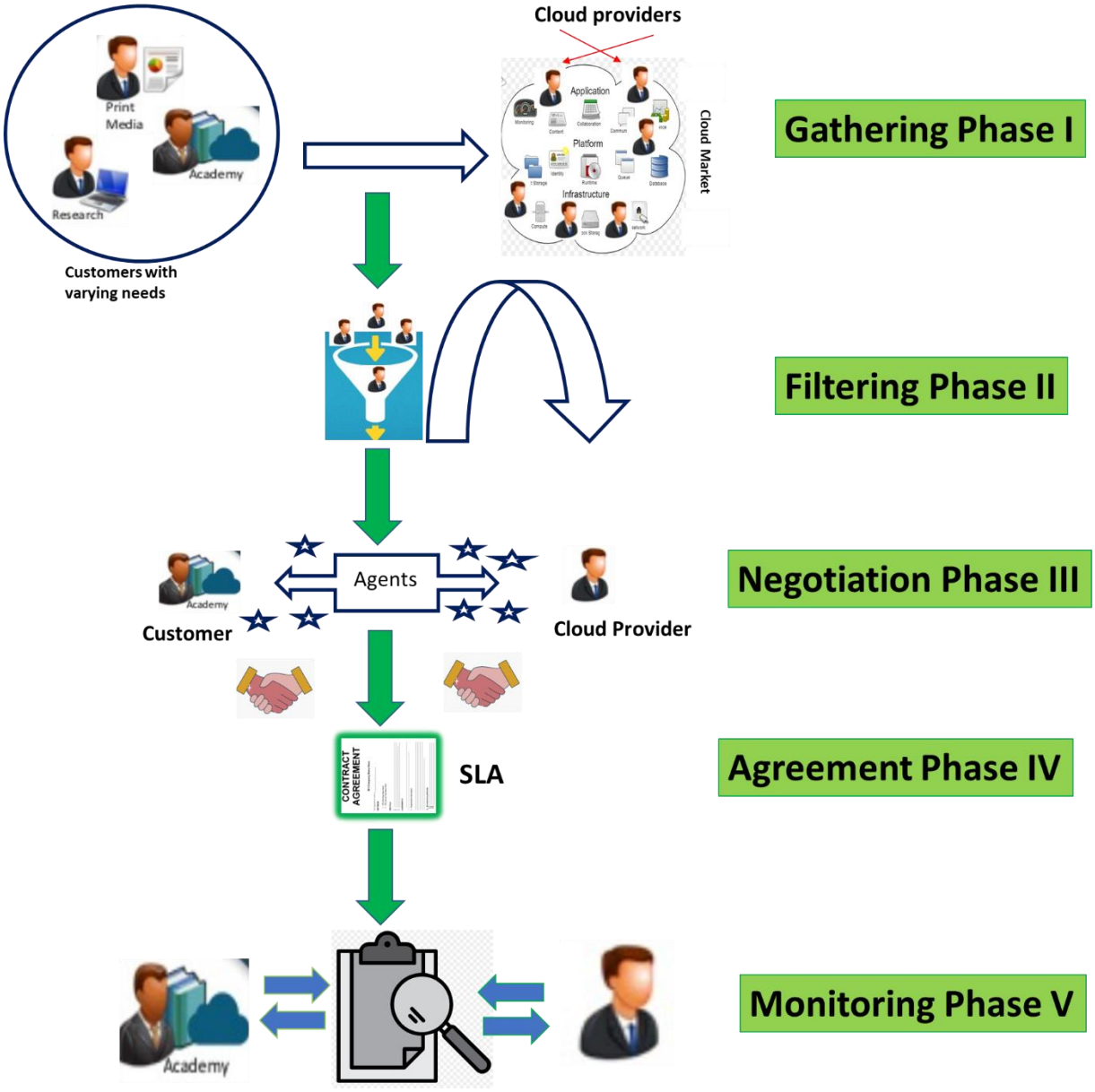


Figure 5-1: The design of SLA showing the communication between different phases of the autonomous SLA framework proposed in this study.

The detailed level of communication is shown in Figure 5-2. With help of arrows, the direction of communication between the customer and cloud provider in the cloud market is shown. The complexity of communication between the customer and cloud provider is managed through the proposed autonomous SLA framework, which can successfully receive process and monitor the SLA. The customer sends a request for services to the cloud provider in the cloud market. Both customers and cloud providers negotiate over the requested services to finalize the deal which is beneficial for both the customer and cloud provider. Following that, monitoring is introduced to ensure the quality of services. Monitoring modules send alerts for correction of services and improve the overall quality of services as agreed between the customer and the cloud provider.

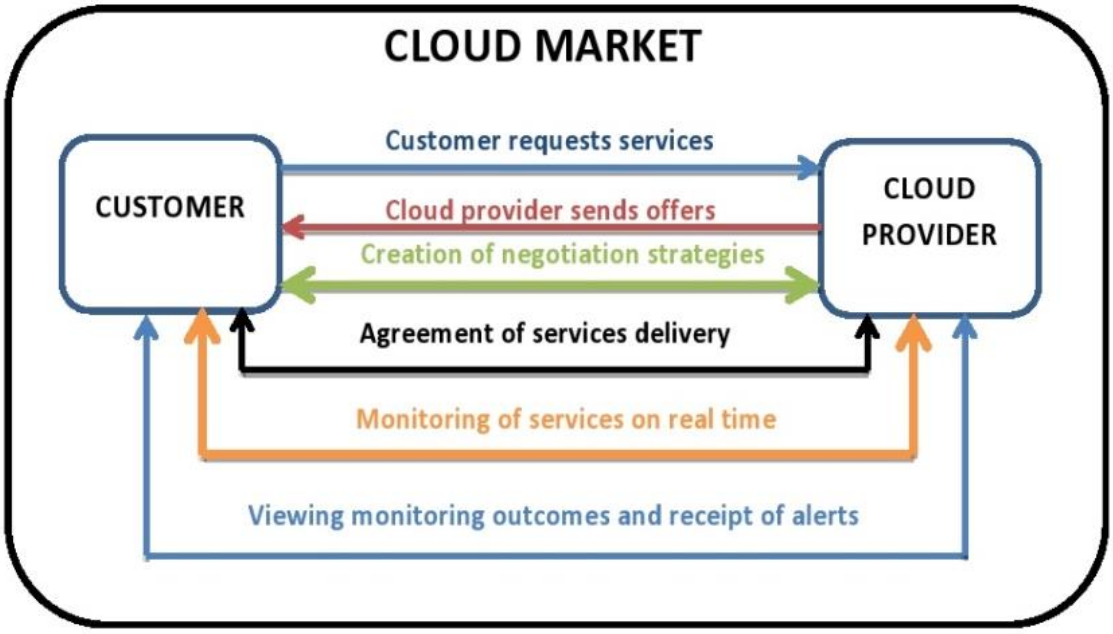


Figure 5-2: The complex communication patterns between the customer and cloud provider in the cloud market.

5.2. Illustration of designs of different phases of SLA framework

In this section, the design conceptualized for each phase is described with the aid of figures. First, the design of the gathering and filtering stage is discussed, followed by illustration of the negotiation phase of the SLA framework. The SLA will be explained with the help of a diagram, followed by description of the monitoring phase.

5.2.1 Gathering and Filtering

This phase starts with gathering data from different cloud providers such as Amazon AWS, AZURE, and Cisco. The cloud offerings displayed on the websites of the cloud providers are observed carefully and recorded. In addition, the offerings with high, medium and low popularity were identified on websites of the cloud providers based on the customers' traffic directed to each cloud offering. In similar fashion, customers' interest, needs and preferences were determined from data presented against each offering. Customers' requests and cloud services' offering were matched with each other in order to find the best match of cloud services with the customers' needs. This is done in the filtering stage. All potential matches are stored and saved in the knowledge database. The figurative illustration of the gathering and filtering stage is shown in Figure 5-3.

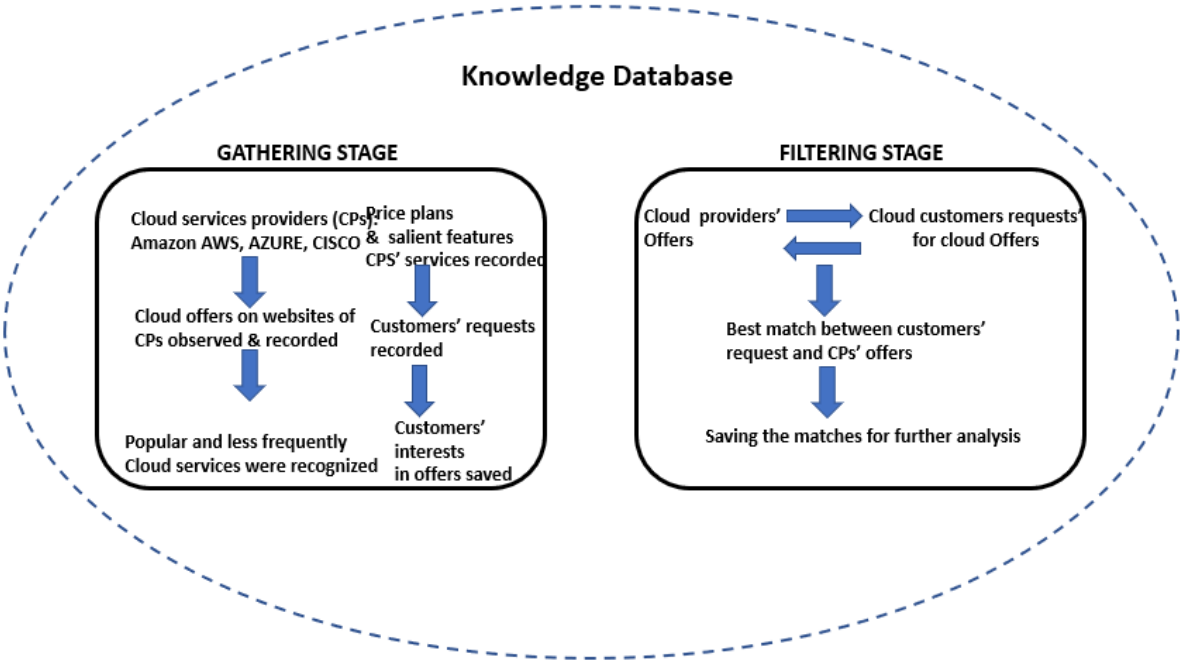


Figure 5-3: The design of gathering and filtering stages in the proposed autonomous SLA framework.

5.2.2 Negotiation Stage

Negotiation depends on finding matches against the customers' needs in the cloud market. If anyone of cloud providers fulfils the criteria specified by the cloud customers, then the negotiation phase will start between the customer and the provider. The negotiation outcomes are saved in the knowledge database for comparison and making decisions whether the SLA will go ahead between the cloud customer and cloud providers.

5.2.3 SLA Agreement Phase

The following diagram shows the SLA agreement stage in detail. It shows each step that needs to be taken inside the SLA agreement stage. The First phase inside the SLA agreement stage is the Generating SLA phase. The second phase inside the SLA agreement stage is the recommending SLA phase. The first step inside the Generating SLA phase is receiving the

negotiation results. The second step is to create the SLA then save it to the Database (knowledge). A copy of the SLA needs to be saved in the Database (knowledge) before recommending the SLA to the provider and the customer. The first step inside the recommending SLA phase is requesting the SLA from the Database (knowledge). The second step is to send a copy of the SLA to the provider and customer. The SLA agreement phase is shown in Figure 5-4.

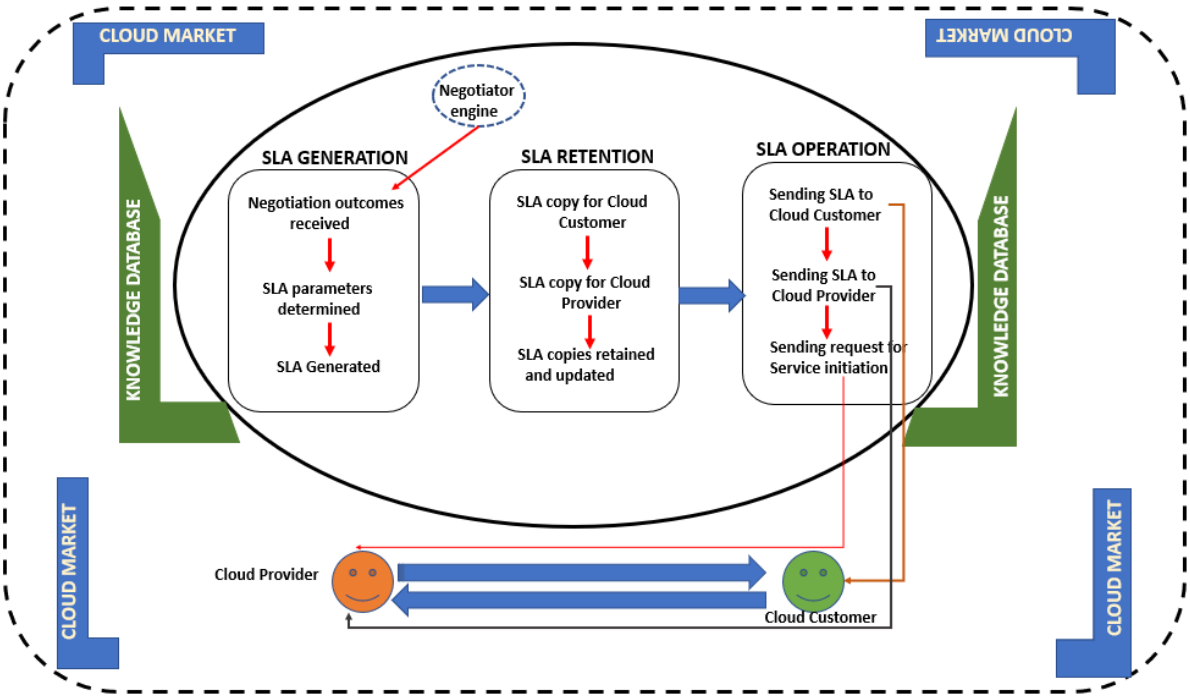


Figure 5-3: The design of the SLA agreement phase in the proposed SLA framework.

5.3 Overall SLA Framework's closed loop

The closed loop diagram showing the flow of activities executed as part of the implementation of the SLA framework is presented in Figure 5-5.

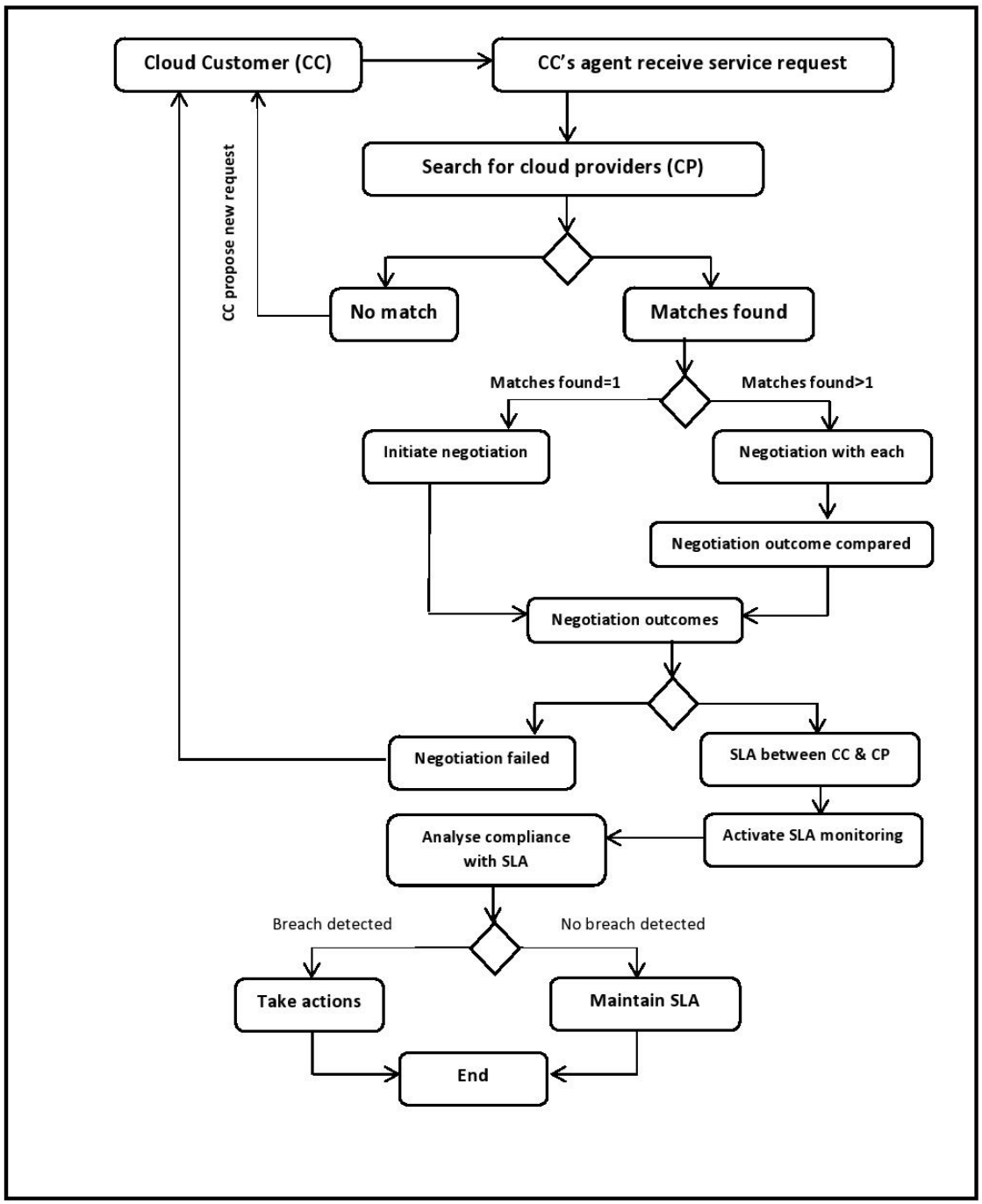


Figure 5-4: The closed loop design for the SLA framework proposed in this study.

5.4 Conclusion

This chapter presented a high-level design of the framework as well as a detailed design for each stage. The gathering and filtering stages are shown in one single diagram. Then, the

negotiation stage in one single diagram, after that, SLA agreement stage in one single diagram. Finally, the overall framework closed loop is presented. In the next chapter, first we will present the negotiation between the customer and cloud provider using game-theoretic negotiation strategies.

CHAPTER 6: IMPLEMENTATION OF GAME-THEORETIC NEGOTIATION AND AGREEMENT PHASES

In the last chapter, the detailed presentation of working elements of the SLA framework was given. The components of the negotiation phase and agreement stage were discussed as well. This chapter will illustrate the implementation of game-theoretic based negotiation and agreement stages. The chapter has been divided into five sections. Section one provides an overview of game-theoretic based negotiation leading to presentation of game-theoretic based agent algorithm. Section three provides the illustration of steps for implementing game-theoretic based negotiation. Section four presents the results, while discussion of outcomes of experiments is carried out in section five. The conclusion is given at the end of this chapter.

6.1. Game Theoretic-Based Negotiations

Game Theoretic based negotiation, uses multi-objective optimisation, which can be measured using Nash Equilibrium or Pareto-Optimality. The Nash equilibrium concept was introduced by John Nash (1950), whereby 'a set of strategies (one for each player) constitutes Nash equilibrium if no player has an incentive to change their strategy given the strategies chosen by the other players' (McNamara, 2013). The Nash point refers to the best optimal solutions for any party on the Pareto frontier – a set point shown graphically where the Pareto efficient allocations can be made, where no party can gain more than other party in the negotiation, the point on the Pareto-frontier which represents the point beyond which no further improvement in the negotiation can be made. The Nash equilibrium can be gained at the Nash point on the Pareto frontier.

The negotiating agents aim to optimise their utility/payoff (which may ideally lead to a Pareto-optimality) by selecting the most appropriate negotiation strategy (which may ideally use a Nash Equilibrium strategy).

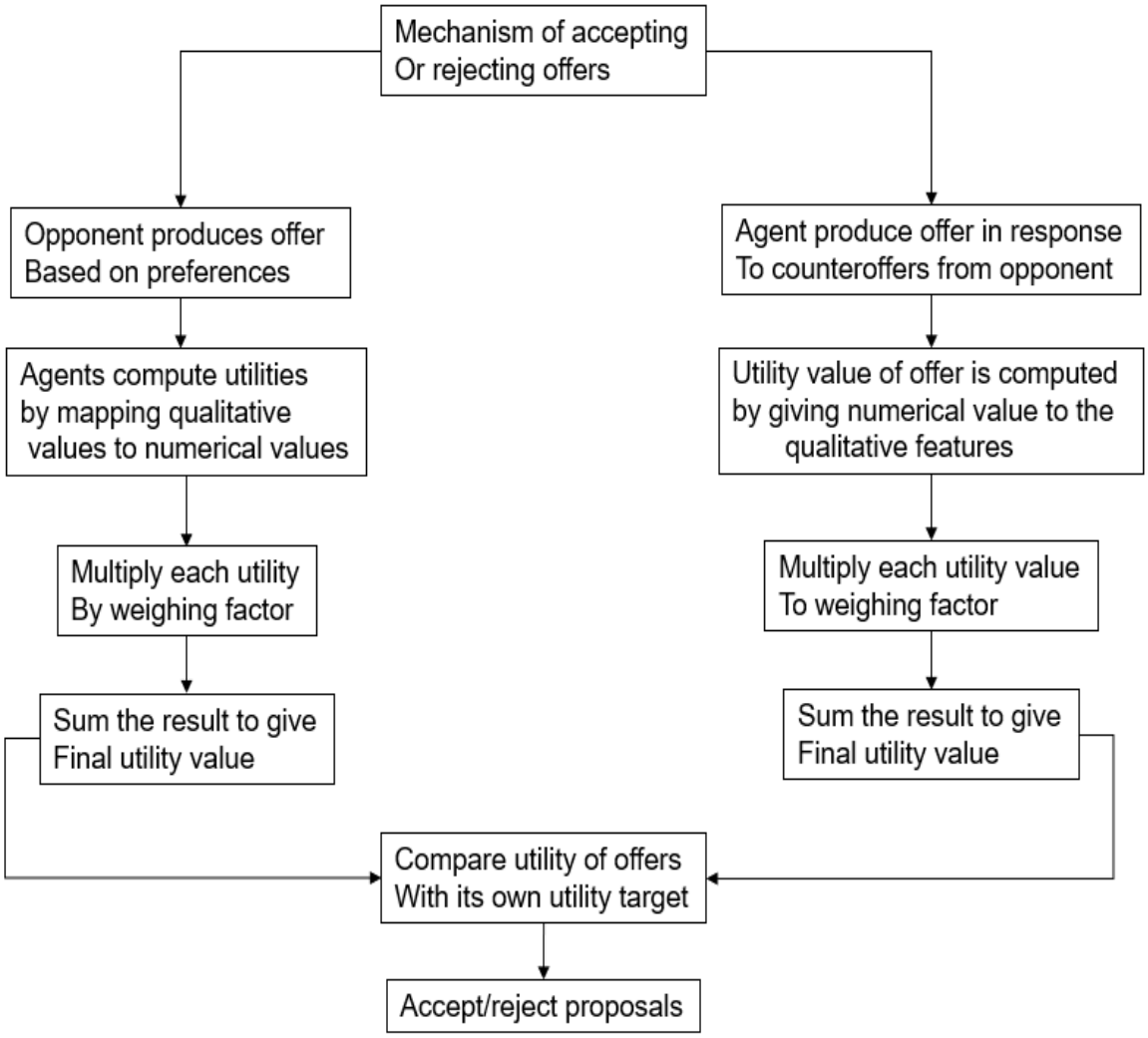


Figure 6-1: Game Theoretic-Based Agent Algorithms

The Game Theoretic Based Agent Negotiation is displayed in Figure 6-1. Several agents are discussed in the following, and used in later sections for cloud computing SLA negotiation.

6.2. Steps for implementing the game-theoretic based negotiation

We now discuss the application of these game-theoretic based negotiating agents for cloud computing SLA negotiation. The agents were chosen from the list of agents given in the GENIUS platform. They were tested for their efficiency in competing on behalf of cloud providers and customers who they represented during the negotiation competitions held in the GENIUS platform (see details in Chapter 3).

The main issues included in this study to be negotiated between cloud customers and cloud service providers were selected from three cloud resources: RackSpace, AWS Amazon, Azure which are platforms for offering a variety of cloud services to customers worldwide. These cloud computing resources were chosen due to negotiations between customers and cloud providers for negotiating multiple issues. Therefore, negotiation scenarios derived from cloud providers in the market are more likely to offer the real picture of negotiating agents in the real-world scenario. Hence it is argued that outcomes of this study will contribute to enhancing the importance of game theoretic-based negotiation in resolving the issues such as deadlines, negotiating failures, meeting the preferences of both customers and cloud providers.

After selection of issues, the four game-theoretic based agents: tit-for-tat, hardheaded, hardliner, and IAMhaggler2012/2011 were selected to represent customers and cloud providers during negotiation sessions. These agents were chosen for the following reasons:

- They have not been tested previously for their negotiation performance against each other in previous studies.
- They are known to compete effectively against their opponents in the ANAC competitions.
- Their algorithms detailed in chapter 3 predict their strengths for negotiating effectively and efficiently with opponents during negotiation. Nevertheless, there is

no empirical evidence regarding performance of these agents during the negotiations in the real market environment.

Following the selection of scenarios and agents, several negotiation experiments were performed in the GENIUS platform to allow the chosen agents to compete with each other during negotiation sessions. The issues in each scenario were given values and weight for both customers and cloud providers in domain specified for each scenario in the GENIUS platform (see Figure 6-2).

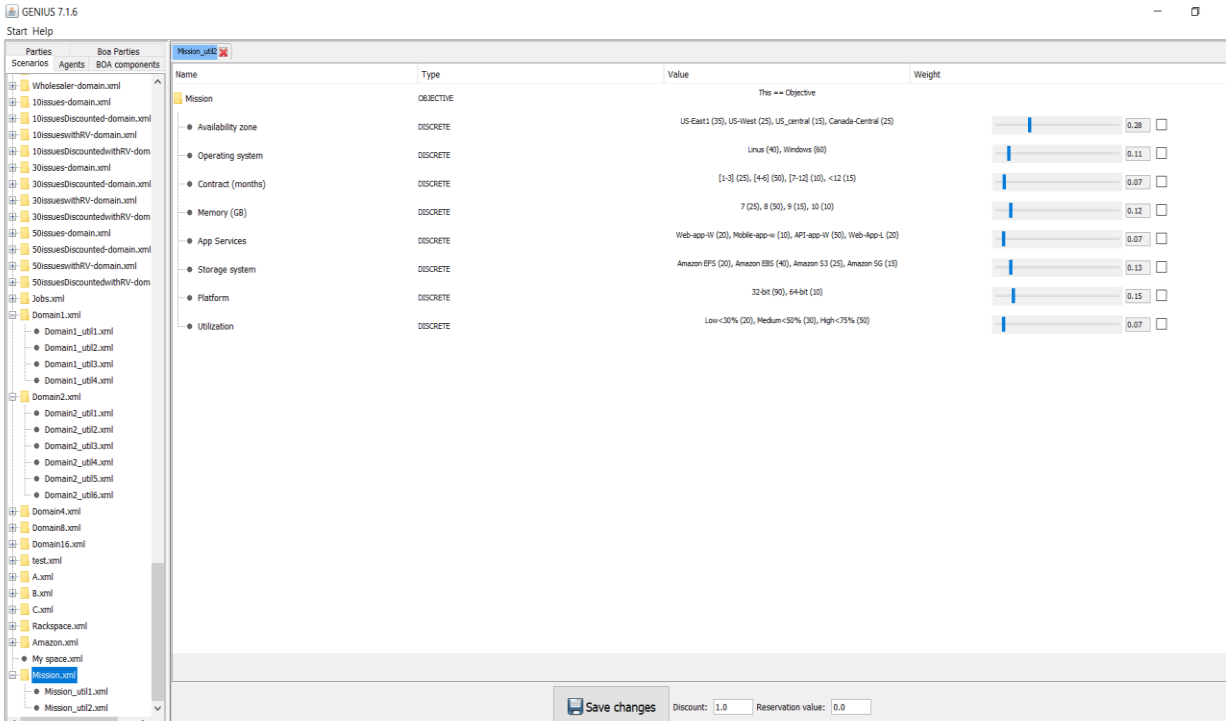


Figure 6-2: The screenshot of GENIUS window to show the issues, values and weights adjusted against all issues.

After adjusting the weights against all issues, the negotiation session was run by selecting the Rubenstein Alternating Offers Protocol (see chapter 3 for detailed discussion). The results were shown in the graphical formats. The screenshot of the GENIUS window to run the negotiation session is shown in Figure 6-3.

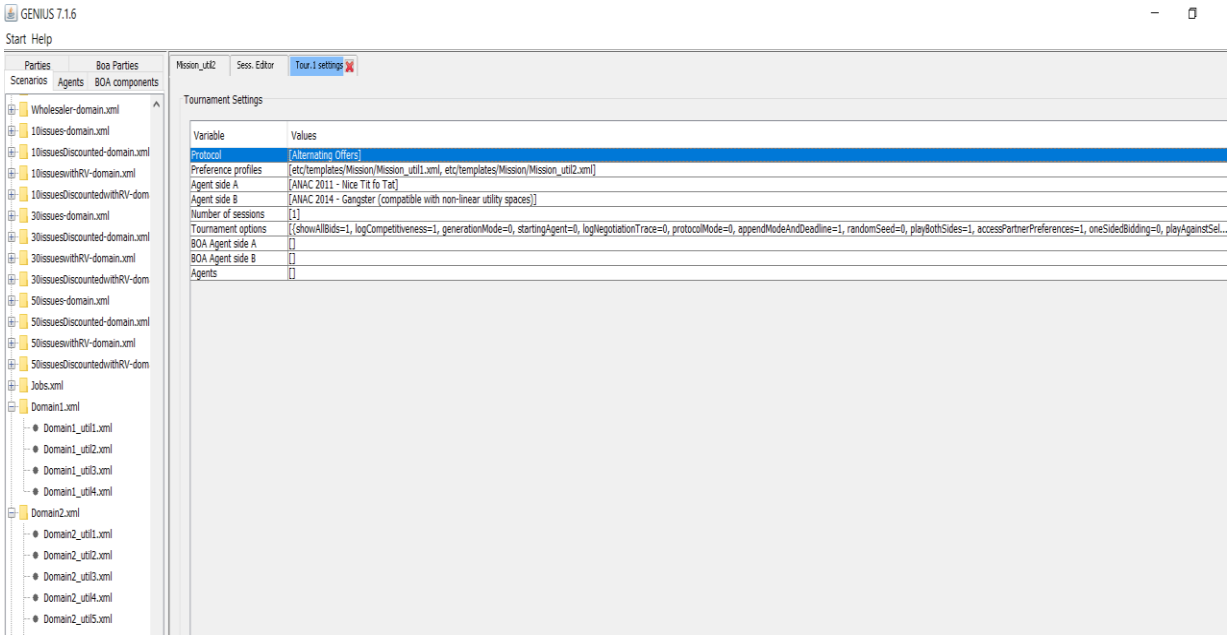


Figure 6-3: The screenshot of GENIUS window to run the negotiation session.

6.3. Results

This section provides outcomes derived from several negotiation experiments involving the game-theoretic based agents.

6.3.1. Experiment 1: Scenario and Negotiation session

6.3.1.1.Scenario 1

In this scenario (Table 6-1), 8 issues are taken, namely Availability Zone, Operating System; Term (months), Memory (GB), Compute units (CPU), Storage (GB), Platform and Utilization. These all make 3,072 possible bids. Customer/ Provider Evaluation value describes the relative preference of a given value. Weight describes the importance of the issue.

Table 6-1: Specifications and values for Scenario 1

Issue	Value	Customer Evaluation	Weight	Provider Evaluation	Weight
Availability Zone	US-East US-West Europe Asia	25 50 10 15	0.36	10 50 15 25	0.19
Operating System	Linux Windows	50 50	0.19	40 60	0.09
Term (months)	[1-6] [7-12] >12	10 50 40	0.10	40 50 10	0.32
Memory (GB)	7 8 9 10	15 50 25 10	0.10	25 50 15 10	0.13
Compute units (CPU)	4 5 6 7	10 50 20 20	0.04	20 10 50 20	0.05
Storage (GB)	[251-500] [501-725]	10 90	0.04	20 80	0.08
Platform	32-bit	80	0.08	90	0.05

	64-bit	20		10	
Utilization	Low <39% Med <75%	50 50	0.09	30 70	0.09

6.3.1.2. Negotiation session

Hard-headed agent maximizes its utility and at the same time estimates the opponents' preferences so that it can offer a bid that is more likely to be accepted. Tit for tat agent will first cooperate and then after each offer, update weights of the opponent model and then make offers aiming for the Nash point.

As the tit-for-tat agent prefers his opponent to initiate the negotiation session, therefore, Hard-headed agent (Cloud provider in this scenario) opens the negotiation session with the following offer: (Offer bit: Bid [Availability zone: US-west, operating system: Linux, Term: [7-12] months, Memory: 8GB, CPU: 8, Storage: [250-500]GB, Platform: 32-bit, Utilization: low <39%,]). This offer carries the utility of 1.0 for the Hard-headed agent and 0.35 for the nice tit-for-tat agent. Hard-headed's offer carried the maximum utility for itself and low concession rate for the opponent, which was consistent with its Boulware strategy (i.e. Take-it or Leave-it) in the beginning of the negotiation session.

The nice tit-for-tat agent (Cloud customer in this scenario) gives the following counter offer: (Offer bit: Bid [Availability zone: US-east, operating system: Window, Term: [1-6] months, Memory: 7GB, CPU: 8, Storage: [501-725]GB, Platform: 64-bit, Utilization: low <75%,]). This offer has utility of 0.47 for the hard-headed agent and 0.98 for the nice tit-for-tat agent. This offer was constructed using the tit-for-tat strategy, as the tit-for-tat agent gave as much concession rate to the hard-headed as the latter gave to the former in the first instance. During the first half of the negotiation session (90 seconds), the hardheaded continued with its

Boulware strategy (i.e. Take-it or Leave-it), and when the nice tit-for-tat agent did not accept, he slowly resorted to the conceding strategy through which it started offering the bids carrying the increasing concession rate to the nice tit-for-tat agent, and using the tit-for-tat strategy, the nice tit-for-tat began offering bids accordingly. The series of offers and counters are exchanged between the hard-headed and nice tit-for-tat agent before reaching agreement on the following offer given by the nice tit-for-tat agent: (Offer bit: Bid [Availability zone: US-east, Operating system: Linux, Term: [7-12] months, Memory: 7GB, CPU: 7, Storage: [501-725]GB, Platform: 64-bit, Utilization: low <75%,]). This offer carries the customer utility of 0.894 and provider utility of 0.679. They reached an agreement after a long negotiation of 7,282 rounds (Figure 6-4). Hardheaded agent accepts this offer due to the adoption of a conceding strategy at the end of negotiation session, even though it was lower than the Nash point.

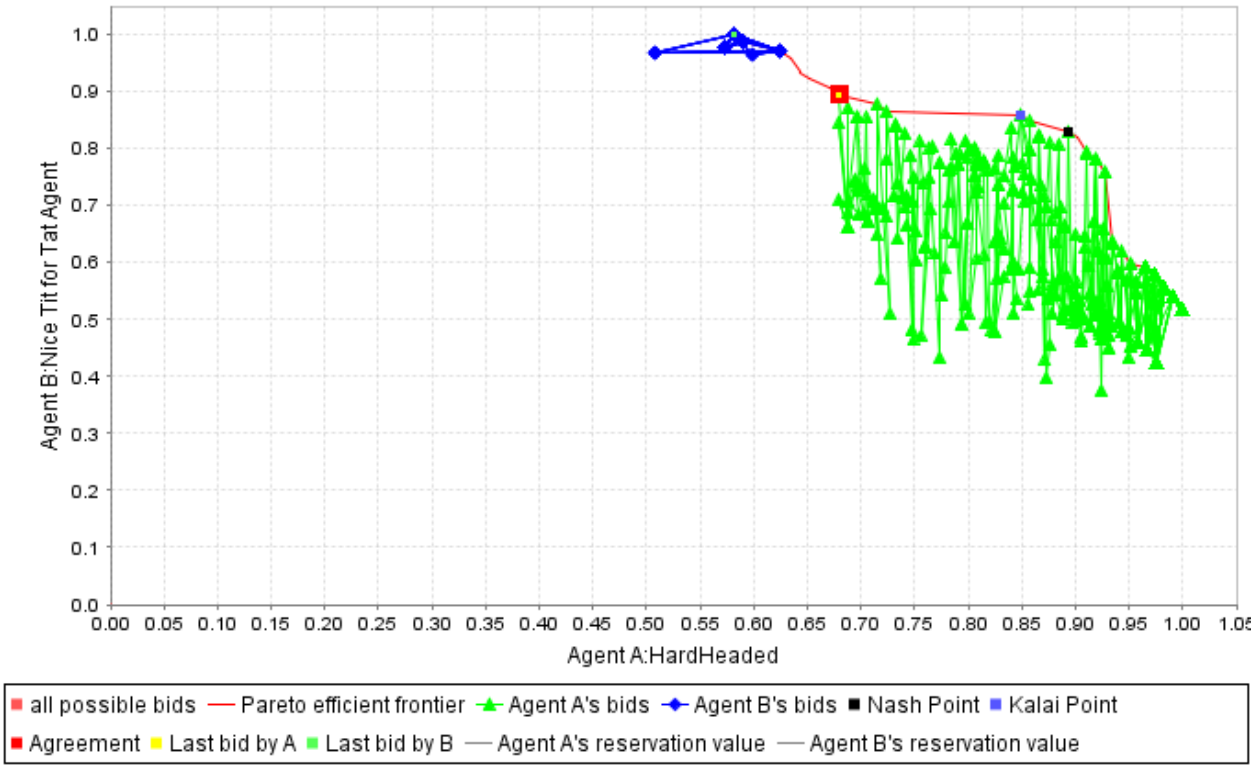


Figure 6-4: Simulation result for Scenario 1

6.3.2. Experiment 2: Scenario and negotiation session

6.3.2.1. Scenario 2

In this scenario (Table 6-2), 8 issues are taken namely Availability Zone, Operating System, Term (months), Memory (GB), Compute units (CPU), Storage (GB), Platform and Utilization. These all make 2,304 possible bids. Customer/ Provider Evaluation values describe the relative preference of a given value. Weight indicates the importance of the issue.

Table 6-2: Specifications and values for Scenario 2

Issue	Value	Costumer Evaluation	Weight	Provider Evaluation	Weight
Availability Zone	US-East US-West Europe	33 66 1	0.19	66 33 1	0.15
Operating System	RedHat Linux Ubuntu Oracle Linux	25 65 10	0.15	65 25 10	0.29
Term (months)	[1-6] [7-12]	50 50	0.01	40 60	0.03
Memory (GB)	7 8 9	25 50 10	0.18	50 15 15	0.02

	10	15		20	
Compute units (CPU)	4 5 6 7	25 25 10 40	0.07	30 10 25 35	0.11
Storage (GB)	[251-500] [501-725]	40 60	0.05	20 80	0.05
Platform	32-bit 64-bit	20 80	0.23	30 70	0.23
Utilization	Low <39% Med <75%	50 50	0.12	10 90	0.12

6.3.2.2. Negotiation session

In this experiment, the cloud provider is represented by the hardliner agent. This agent gives a bid with maximum utility for itself and doesn't accept an offer less than maximum utility for itself. It ends up at maximum utility or no agreement. It uses the 'Take it or leave it' approach. The customer is represented by hard headed. It tries its best to bring the provider to a compromise but the provider is hardliner so it won't cooperate. The negotiation session is started by the hard-headed agent with the following offer to the hardliner agent: (Offer bit: Bid [Availability zone: US-west, Operating system: Ubuntu Oracle, Term: [7-12] months, Memory: 8GB, CPU: 7, Storage: [501-725]GB, , Platform: 64-bit, Utilization: low <39%,]). The offer has a utility of 1.0 for hard-headed and 0.576 for the hardliner agent. The low concession rate for the opponent and highest utility for itself can be observed in the offer of hard-headed agent to the hardliner agent, which is in accordance with the Boulware strategy

(i.e. Take-it or Leave-it) used by the hard-headed in the first half of the negotiation session. The hardliner rejects the offer and presents the following counter offer to the hard-headed agent: (Offer bit: Bid [Availability zone: US-east, Operating system: RedHat Linux, Term: [1-6] months, Memory: 7GB, CPU: 7, Storage: [501-725]GB, Platform: 64-bit, Utilization: Med <75%,]). This offer has the utility of 1 for hardliner and 0.72 for the hard-headed agent. The hardliner continues to offer the same bid again and again during the negotiation session by enacting its 'take it or leave it' strategy, while the hard-headed agent implemented the Boulware strategy (i.e. Take-it or Leave-it) in the first-half of the session, and adopted a conceding strategy in the second half of the negotiation session. In this way, both agents continue to exchange the offers; the hard-headed offers shown in the green dots were presented in the higher proportions compared to hardliner which stuck to only a single offer due to its selfish and stubborn nature. The negotiation session finally came to an agreement at the following offer given by the hardliner to the hard-headed agent after a long negotiation of 17,023 rounds: (Offer bit: Bid [Availability zone: US-east, Operating system: RedHat Linux, Term: [1-6] months, Memory: 7GB, CPU: 7, Storage: [501-725]GB, Platform: 64-bit, Utilization: Med <75%,]) with provider utility of 0.72 for the hardheaded (Cloud provider) and 1.0 for hardliner (customer) (Figure 6-5). In this negotiation session, hard-headed accepted the offer due to its conceding strategy at the end of the negotiation, even though the offer accepted was lower than its optimal Nash point, while the hardliner gained maximum benefit in terms of its utility due to its stubborn and selfish approach.

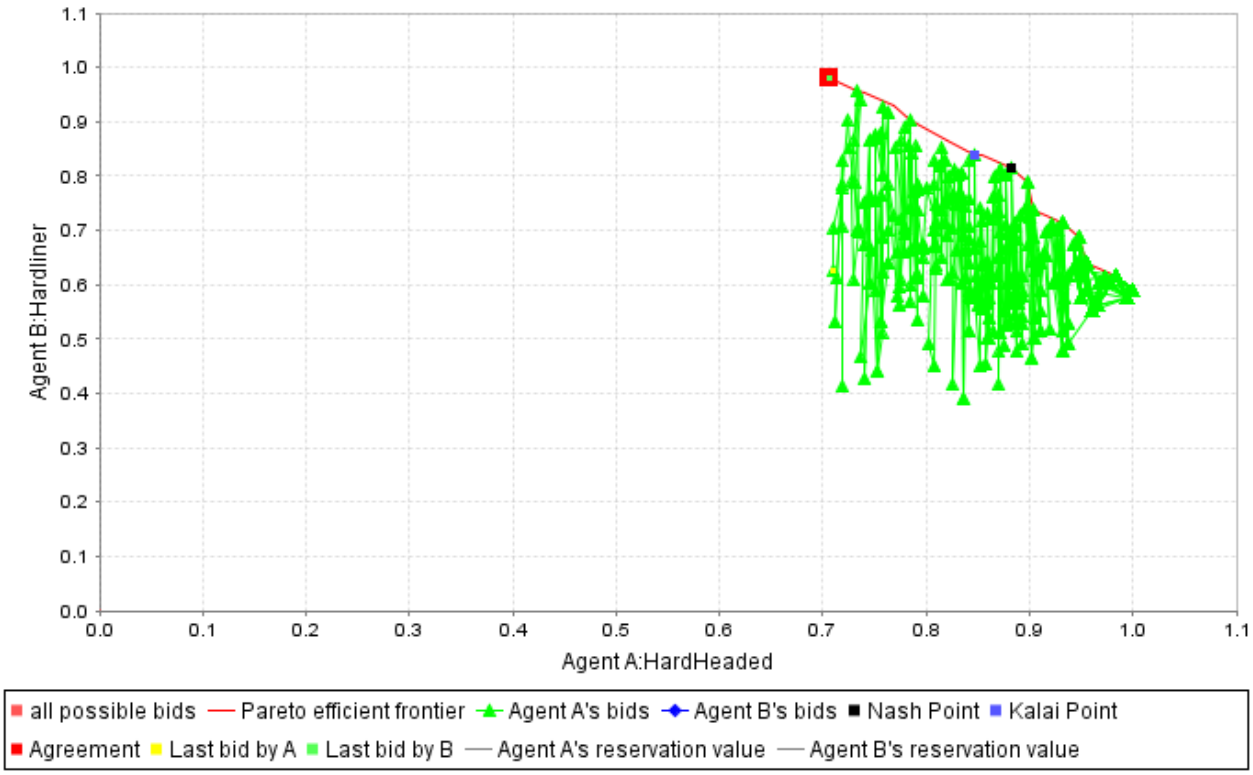


Figure 6-5: Simulation result for Scenario 2

6.3.3. Experiment 3: Scenario and negotiation session

6.3.3.1. Scenario 3

In this scenario, 7 issues are taken namely Operating System, Pricing plan, Delivery method, Architecture, Availability Region, Memory (GB), and Storage (GB). Costumer/ Provider Evaluation values and weights are provided against each issue in Table 6-3.

Table 6-2: Specifications and values for Scenario 3

Issues	**Value**	**Provider Evaluation**	**Weights**	**Customer Evaluation**	**Weights**
Operating system	Windows Linux	25 75	0.16	75 25	0.19

Pricing plan	Hourly Monthly Annual	20 30 50	0.12	50 30 20	0.1
Delivery method	Amazon Machine Image CloudFormation Stack SaaS	30 50 20	0.26	30 20 50	0.12
Architecture	32-bit 64-bit	25 75	0.13	50 50	0.07
Availability Region	US East (Ohio) US West (Oregon) Asia Pacific EU (Frankfurt)	15 5 30 50	0.19	40 40 10 10	0.36
Memory (GB)	5 6 7 8	10 30 50 10	0.05	10 10 30 50	0.09
Storage (BG)	250-500 501-700	25 75	0.09	50 50	0.07

6.3.3.2. Negotiation session

The negotiation session was run between the cloud provider represented by IAMhaggler2012 agent and the customer represented by the Hard-headed agent. The deadline was set at 180 seconds, and concession was kept 1.0.

The cloud providers' agent IAMhaggler2012 begins the negotiation session with this offer: (Offer bit: Bid [Operating system: Linus, Pricing plan: Annual, Delivery methods: Cloudformation stack: Memory: 8GB, Architecture: 64-bit, Availability: Europe (Farnkfurt), Memory: 8GB, Storage: 250-500GB]). As this agent gives offers carrying maximum utility for the opponent in the beginning, therefore, this package has 1.0 utility for IAMhaggler 2012 and 0.438 utility for hard-headed agent. However, the utility of bid offered by the IAMhaggler2012 was lower than the Nash point of the hard-headed, this offer was rejected. Hard-headed agent offered the following counter offer to IAMhaggler 2012 agent: (Offer bit: Bid [Operating system: Windows, Pricing plan: Hourly, Delivery methods: SaaS: Memory: 7GB, Architecture: 64-bit, Availability: US-East (Ohio), Memory: 7GB, Storage: 501-700GB,]). Using Boulware strategy (i.e. Take-it or Leave-it) with the highest utility for itself and low concession rate for the opponent in the beginning, the hard-headed offered the above bid with a utility of 0.39 for IAMhaggler 2012 and utility of 0.98 for hard-headed agent. IAMhaggler 2012 rejected this offer. Based on the previous offer of hard-headed agent, IAMhaggler determined the utility of the hard-headed, and adjusted the utilities of the upcoming offers accordingly, therefore, it kept increasing the concession rate for hard-headed to hit the trade-off acceptable to the opponent.

Therefore, both agents kept exchanging the offers and finally, agreed to this bid [Operating system: Windows, Pricing Plan: Hourly, Delivery Method: Image, Architecture: 64-bit, Availability Region: US East (Ohio), Memory (GB): 8, Storage (GB): 501-700,]). This offer was made by IAMhaggler (cloud provider's agent), and accepted by hard-headed (customer's

agent), with utility of 0.652 for the former and utility of 0.925 for the latter after a long negotiation session of 19,939 rounds. During this negotiation, IAMhaggler remains active in offering the relatively high proportions of bids (indicated as green dots in Figure 6-6) compared to the hard-headed.

In this negotiation session, the agreement was reached due to a more flexible approach adopted by IAMhaggler in terms of seeking to hit the Nashpoint of hard-headed at the end of the negotiation despite the package accepted being lower than its optimal Nashpoint. Thus, this package seems to give relatively higher utility to Hard-headed agent compared to IAMhaggler 2012 gained higher utility from the package due to its stubborn approach.

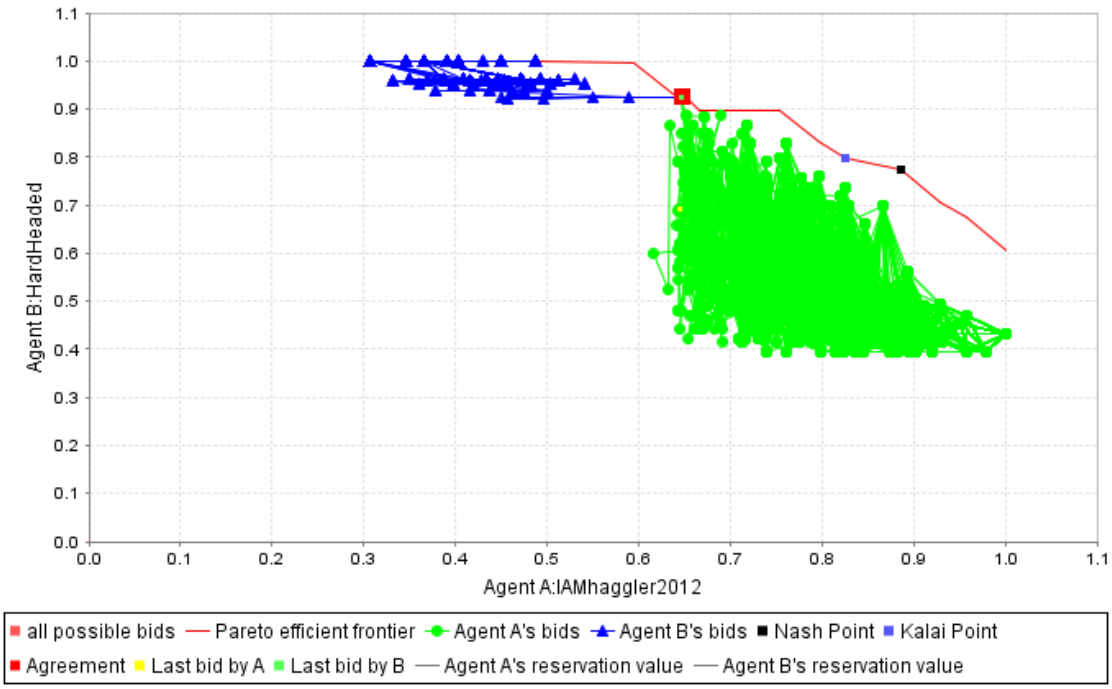


Figure 6-6: Simulation result for Scenario 2

6.3.4. Experiment 4: Scenario and negotiation session

6.3.4.1. Scenario 4

In this scenario, 5 issues are selected, which are Availability Zone, Term (months), Backup, Data In and Data Out. The cloud provider and customer's evaluation and weights are provided in table 6-4.

Table 6-3: Specifications and values for Scenario 4

Issues	**Values**	**Provider Evaluation**	**Weights**	**Customer Evaluation**	**Weights**
Availability	US East US West Europe Asia	10 10 5 75	0.29	40 20 20 20	0.07
Term	1-6 months 7-12 months 13-24 months More than 24 months	50 40 5 5	0.08	10 10 50 30	0.11
Back up	Every 12 hours 1 day 1 week 1 month	10 80 5 5	0.4	80 5 5 10	0.32

Data in	Up to 100 GB Up to 1 TB Up to 10 TB about 10 TB	50 20 10 10	0.08	5 90 3 2	0.38
Data out	Up to 100 GB Up to 1 TB Up to 10 TB about 10 TB	10 10 60 20	0.15	5 5 80 10	0.12

6.3.4.2. **Negotiation session**

In this experiment the provider is represented by IAMhaggler2012 agent, while the customer is represented by Nice Tit-for-Tat agent. IAMhaggler2012 agent gives bids based on maximum utility for its opponent and calculates its own utility based on future preferences. The customer is Nice Tit-for-Tat agent, and tries its best to bring the provider (IAMhaggler) to a compromise. Using the principle of 'Opponent begins first', Nice tit-for-tat allowed the cloud providers' agent IAMhaggler2012 to begin the negotiation session. IAMhaggler2012 came up with this offer: (Offer bit: Bid [Availability: US-East, Term: 7-12 months, Backup: 12 hours, Data In: 100GB, Data out: 1TB,]). Because IAMhaggler gives the maximum utility to the opponent in the beginning, therefore, this package has 0.97 utility for IAMhaggler 2012 and 0.488 utility for Nice Tit-for-Tat agent. However, it did not hit the Nashpoint for tit-for-tat, which is why it was rejected. Nice Tit-for-tat agent used the reciprocal approach in its upcoming approach, and offered the same amount of concession rate to the IAMhaggler 2012 as it was offered itself. Hence Nice Tit-for-Tat agent offered the following counter offer to IAMhaggler 2012 agent: (Offer bit: Bid [Availability: Asia, Term: 13-24 months, Backup:

24 hours, Data In: 1TBGB, Data out: 100GB]). This offer carries utility of 0.463 for IAMhaggler 2012 and utility of 0.98 for Nice Tit-for-Tat agent. IAMhaggler 2012 rejected this offer.

After recording the initial offers from Nice tit-for-tat, IAMhaggler adjusted the utility of its offers to the nice tit-for-tat. However, by keeping in mind the cooperative nature of the Nice tit-for-tat agent, IAMhagger continued with low increments in concession rate and utility for its upcoming offer, due to which the Nice-tit-for-tat kept rejecting in the 80% of the negotiation session. When 20% time of negotiation was left, IAMhaggler started giving more concession rate to the nice tit-for tat; and finally both agents reached the agreement on the following offer: (Offer bit: Bid [Availability: US-East, Term: 13-24 months, Backup: 24 hours, Data In: 1TBGB, Data out: 5TB,]) after 7,127 rounds of negotiation. This offer has utility of 0.866 for IAMhaggler2012 and utility of 0.961 for the Nice Tit-for-Tat agent (Figure 6-7). The almost equal utility of package gained by both agents through the negotiation session was the result of efforts made by IAMhaggler to reach the agreement, and cooperation resulting from tit-for-tat strategy adopted by Nice tit-for-tat agent. The offer carried almost equal utility for both agents because tit-for-tat agent always tries to make as much concession as its opponent does during negotiation.

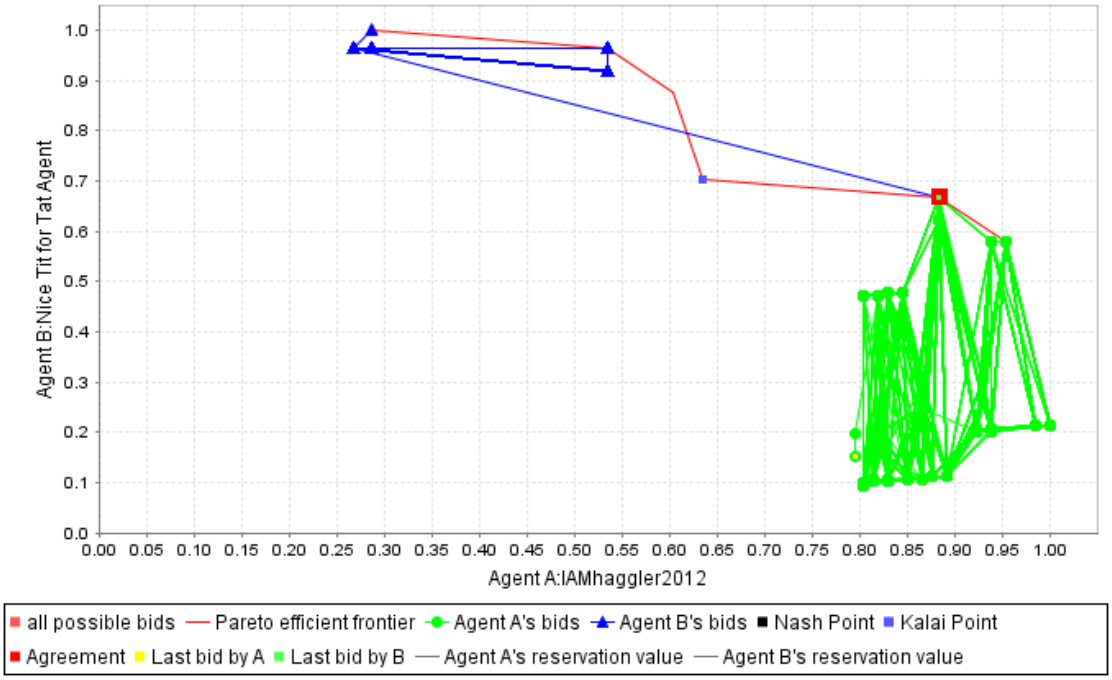


Figure 6-7: Simulation result for Scenario 4

6.3.5. Experiment 5: Scenario and negotiation session

6.3.5.1. Scenario 5

In this scenario, 8 issues are taken namely Availability Zone, Operating System, Term (months), Memory, Storage, SPU units, Platform and Utilization. The customer's and cloud provider's profiles against each issue are shown in Table 6-5.

Table 6-4: Specifications and values for Scenario 5

Issues	**Value**	**Provider Evaluation**	**Weight**	**Customer Evaluation**	**Weight**
Availability zone	Europe Asia Africa	30 50 20	0.11	50 30 20	0.28

Operating system	Linux Microsoft Win	80 20	0.27	30 70	0.01
Term	1 Year 2 Year 3 Year	10 30 60	0.06	50 20 30	0.01
Memory (GB)	6 8 10	10 80 10	0.14	70 10 20	0.01
Storage (BG)	150 200 250	10 10 80	0.19	20 60 20	0.51
CPU unit	5 7 9	10 5 85	0.3	60 20 20	0.01
Platform	32-bit 64-bit	50 50	0.16	80 20	0.02
Utilization	Light Medium Heavy	20 20 60	0.04	80 10 10	0.15

6.3.5.2. **Negotiation Session**

In this experiment, IAMhaggler2011 represents the cloud provider, while Nice Tit-for-Tat agent acts on behalf of the customer in the negotiation session. The agreement between two agents was reached after 11,909 rounds. The cloud providers' agent IAMhaggler2011 begins

the negotiation session with this offer: (Offer bit: Bid [Availability zone: Asia, Operating system: Linux, Term: 3 years, Memory: 8GB, Storage: 250GB, CPU: 9, Platform: 32-bit, Utilization: heavy]). This package has 1.0 utility for IAMhaggler2011 and 0.388 utility for Nice Tit-for-Tat agent. IAMhaggler2011 agent computes the preferences, utilities and concession rates of the opponent in order to determine the offers which can possibly lead to an agreement. Although the bid offers maximum utility to the nice tit-for-tat based on the preference profile of the agent, it did not reach the Nash point of nice tit-for-tat agent. Therefore, Nice Tit-for-Tat rejected this offer. Using the tit-for-tat strategy, Nice Tit-for-Tat agent offered the following counter offer to IAMhaggler2011 agent: (Offer bit: Bid [Availability zone: Europe, Operating system: Win, Term: 1 years, Memory: 6GB, Storage: 200GB, CPU: 5, Platform: 32-bit, Utilization: light,]). This offer carries utility of 0.363 for IAMhaggler2011 and utility of 0.999 for Nice Tit-for-Tat agent. IAMhaggler rejected this offer.

Based on the preference profile and utilities of offers exchanged by Nice tit-for-tat agent, IAMhaggler2011 continued to increase the concession rate for the opponent slowly but gradually, which lingered on the negotiation session up to 11,909 rounds. Tit for tat agent also continued to adjust its utilities and concession rate based on the concession rate and utilities of the previous offers of IAMhaggler2011 to target the Nash point calculated based on opponent's utilities and weights.

Both sides kept sending offers and counter-offers; and the deadline was set at 180 seconds. Finally they struck the agreement after exchanging offers 11,909 times at the following offer thrown by agent IAMhaggler2011: (Offer bit: Bid [Availability zone: Europe, Operating system: Linus, Term: 3 years, Memory: 8GB, Storage: 200GB, CPU units: 9, Platform: 64-bit, Utilization: light,]). This offer has utility of 0.776 for IAMhaggler2011 and utility of

0.951 for the Nice Tit-for-Tat agent. The green dots in Figure 6-8 show that IAMhaggler2011 was trying to be more flexible than nice tit-for-tat during the negotiation session.

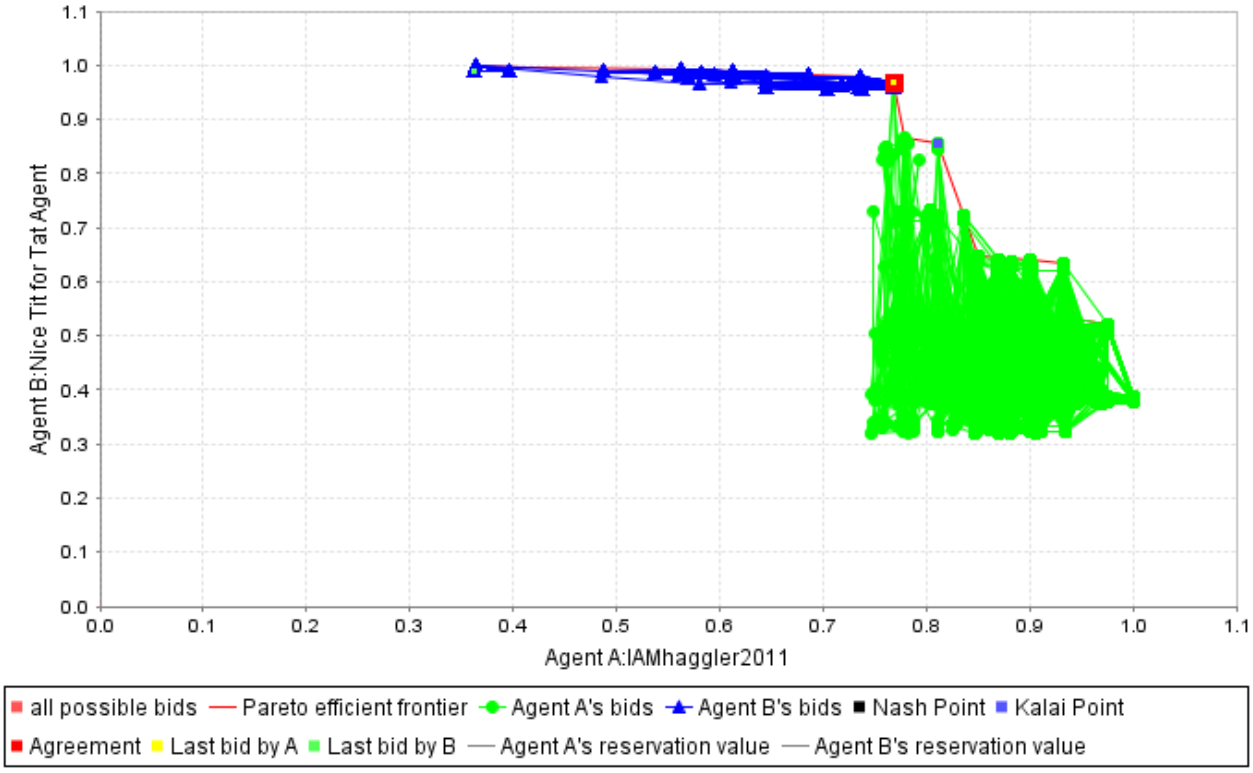


Figure 6-8: Simulation result for Scenario 5

The experiments and results discussed above show that IAMhaggler 2011/2012 and Nice tit-for-tat agents are good options to conduct negotiations when the customer or cloud provider have enough time and resources, and are willing to sell or buy the cloud computing services. We found that both agents are suitable to reach the Nash point and have enough strategies and tactics to reach the Nash point. However, both agents used a thousand bids in each negotiation session, which means that they used high computational resources and time in reaching the SLA with their opponent. In contrast, the hardliner and hardheaded were hesitant to exchange the deals, and they were more anxious about increasing their utility without considering the opponents' utility. These agents mostly exchanged the same offers during

negotiation which indicated their stubbornness. These agents were found suitable in the cases where the customer and cloud provider have limited options to change, and they only target a certain amount of utility during negotiation.

6.3.6. Summary of experiments

The aim of these experiments is to know how the different strategies of each e-agent work against the others, and to take a perception of the strengths and weaknesses of each e-agent. Here is the summary of all scenario s (experiments) that I have done.

In the first scenario, we aim to see how the Hard-headed strategy works with Nice Tit-for-tat strategy and witch one is better to get much utility. Hard-headed agent maximizes its utility. Then, he slowly resorted to the conceding strategy through which it started offering the bids carrying the increasing concession rate. Nice Tit-for-tat agent cooperated first and then starts to reject the offers and increase its utility. After a long negotiation of 7,282 rounds, they have reaching agreement with utility of 0.894 for Nice Tit-for-tat and utility of 0.679 for Hard-headed.

In the second scenario, we aim to see how the Hard-headed strategy works with hardliner strategy and can they get agreement if both agents try to get maximum utility. Also, both uses the 'Take it or leave it' approach. They have reaching agreement with a utility of 0.72 for hard-headed and 1.0 for the hardliner agent after 17,939 rounds. Hard-headed accepted the offer due to its conceding strategy at the end of the negotiation, even though the offer accepted was lower than its optimal point, while the hardliner gained maximum benefit in terms of its utility due to its stubborn and selfish approach.

In the third scenario, the negotiation was run between IAMhaggler2012 agent and Hard-headed agent. Both agents kept exchanging the offers with the highest utility for itself and low concession rate for the opponent at the beginning and that made a long negotiation session of 19,939 rounds. Finally, the agreement was done with utility 0.925 for IAMhaggler and 0.652 for hard-headed. IAMhaggler got a good utility because during the negotiation he remains active in offering the relatively high proportions of bids compared to the hard-headed.

In the fourth scenario, we aim to know how the negotiation going when the both agents are more flexible before the end of negotiations. The negotiation was between IAMhaggler2012 and Nice tit-for-tat. IAMhaggler2012 used to be a selfish at the beginning, but because all his offers have rejected, he getting more flexible at the end. Nice Tit-for-Tat started the negotiations by a compromise from the beginning kept changing the offers until the end. After 7,127 rounds of negotiation, both agents reached the agreement with utility of 0.866 for IAMhaggler2012 and utility of 0.961 for the Nice Tit-for-Tat agent.

In the fifth scenario, we aim to know how the negotiation going when the both agents are flexible with the opponent from the beginning. The negotiation was between IAMhaggler2011 and Nice tit-for-tat. Both of them continued to increase the concession rate for the opponent slowly but gradually. They have reached agreement after exchanging offers

11,909 times with utility of 0.776 for IAMhaggler2011 and utility of 0.951 for the Nice Tit-for-Tat agent. We found that IAMhaggler2011 was trying to be more flexible than nice tit-for-tat during the negotiation session.

The experiments and results discussed above show that IAMhaggler 2011/2012 and Nice tit-for-tat agents are good options to conduct negotiations when the customer or cloud provider have enough time and resources, and are willing to sell or buy the cloud computing services. We found that both agents are suitable to reach the agreement point and have enough strategies and tactics to reach the agreement. In contrast, the hardliner and hardheaded were hesitant to exchange the deals, and they were more anxious about increasing their utility without considering the opponents' utility. These agents mostly exchanged the same offers during negotiation which indicated their stubbornness. These agents were found suitable in the cases where the customer and cloud provider have limited options to change, and they only target a certain amount of utility during negotiation.

6.4. Discussion

In this negotiation session 1 and 2, hard-headed agent competed with Nice tit-for-tat agent; the results of negotiation showed that hard-headed accepted the offer due to the adoption of a conceding strategy at the end of the negotiation session, even though the offer was located lower than its actual Nash point. However, in both sessions, Nice- tit-tat agent was able to win the higher utility for its client compared to its opponent, who was mainly due to the efficiency of Nice tit-for-tat agent to learn the weights and utilities of the opponent's offers and make the offer attractive to the opponent.

Similarly, in negotiation session 3, hard-headed accepted the offer due to its conceding strategy at the end of the negotiation, even though the offer accepted was lower than its optimal Nash point, while the hardliner gained maximum benefit in terms of its utility due to its stubborn and selfish approach. Interestingly, it was observed that hard-headed agent could not yield the higher utility to its client compared to its opponent, which might be related to sudden change in strategy (conceding strategy). The sudden change in strategy might be a weakness on which the opponent fully capitalizes to maximize its utility.

In negotiation session 4 and 5, the IAMhaggler2011 and 2012 agents encountered the Nice tit-for-tat agent, and in both cases, Tit-for-tat agent was able to maximize utility for its client

though within a narrow margin compared to the opponent. In each case, IAMhaggler accepted an offer at a lower Nash point which was mainly due to accommodating nature of IAMhaggler 2011/2012 agents for their opponent. However, the IAMhaggler2011 yielded better than its client compared to IAMhaggler 2012, which showed that the former is more effective than the latter during negotiation with Nice tit-for tat agent. In all cases, Tit-for-tat agent was found to be effective for its client in terms of increasing utility for its client who is mainly attributed to its effective learning about the future moves of the opponent and to make the offers as attractive as possible for the opponent.

It is observed that Nice Tit-for-Tat is capable of performing well even if the deadline is short. Hard-Headed performed and the Nice Tit-for-Tat for a deadline of either 100 or 1,000 seconds. Hard-headed was the most efficient agent when time changed from 10 seconds to 100/1000 seconds. There is only a little difference in the performance of an IAMhaggler agent when the deadline was 100 seconds and 1,000 seconds, where the rest of the agents have achieved exactly the same performance and fairness whether the deadline was 100 or 1000 seconds (Table 6-2).

There is a relation between the deadline round-based protocol and the time-based protocol; it was found that when the deadline was set up to 10 seconds the number of rounds that each negotiation session took was between 2 and 2,000. When the deadline was set up to 100 seconds, each negotiation session took between 4,000 and 14,000 rounds. Consequently, investigation shows that increasing the deadline to more than 10,000 rounds or 100 seconds will not improve the negotiation outcomes. So, it is not recommended to increase the deadline to 100 seconds or more, as this will not make any difference to the outcomes of the negotiations.

After all, all the agents did better when deadlines were enhanced. When the rounds were increased up to 100, Tit-for-Tat had the best performance and Hard-Headed had the worst.

For 10,000 rounds, Hard-Headed, Tit-for-Tat got the same results as for when the deadline was 1,000 rounds. When the deadline was 100 rounds, IAMhaggler2011 did the best and shared the first place, the second placed agent was hardheaded and the final one was Tit-for-Tat. All agents did exactly the same as they did whether the deadline was 1,000 or 10,000. After these investigations, the following recommendations can be made (Table 6-6)

Table 6-5: Performance evaluation criteria for agents.

Agents	**Performance Evaluation Criteria**					
	Time-based deadline			Round-based deadline		
	10 seconds	100 seconds	1000 seconds	100 rounds	1000 rounds	10000 rounds
Tit-for-tat	Performed good	Performed better	Performed best	Performed good	Performed better	Performed better
Hardheaded	Performed bad	Performed better	Performed best	Performed better	Performed better	Performed best
Hardliner	Performed bad	Performed better	Performed best	Performed better	Performed better	Performed better
IAMhaggler	Performed good	Performed better	Performed best	Performed best	Performed better	Performed better

As it is apparent from data in Table 6.6, there is a high cost to setting the deadline to a lower number of rounds such as 10 rounds or fewer, as most of the negotiation sessions will end up without any agreement. The agents will do better if the deadline be increased to 100 rounds.

However, if the goal is to increase the overall performance, then it is recommended to increase the deadline to 1,000 rounds but not more than 1,000.

From the results presented above it is clear that different agents use different algorithms during the negotiation, and are always interested to increase the utilities for their own purposes depending on time-based and round-based deadlines. However, none of the above agents has a suitable set of strategies and tactics which could be used in different contexts within a negotiation session or tournament. Therefore, there is a need to propose a novel agent which can compete with their opponents by learning more about the preferences of the opponent, and can change its strategies to suit the opponent while maintaining the utility level of its own principal. Hence this study intends to combine the strategies of different agents in the light of findings of this study to develop a more robust and effective algorithm which would be flexible enough to overcome the obstacles in negotiation and help reach an SLA within the set time-limit.

6.5. Conclusion

A Game-Theoretic framework is proposed for achieving automated negotiation between providers and consumers in cloud environments. The main focus of work is that framework is made particularly for cloud computing by using automated negotiation algorithms or agents. At the same time, it is flexible to new algorithms or agents that might be developed in future. We discussed Negotiation Frameworks, Negotiation algorithms and Negotiation experiments output is analysed. The effect of increase and decrease of deadline to negotiation outcome is also investigated. The next chapter will involve developing a new negotiating algorithm combining the strength of the agents investigated in this chapter.

CHAPTER 7: DEVELOPMENT AND EVALUATION OF AUTONOMOUS AGENT-BASED NEGOTIATION STRATEGY

In the previous chapter, various negotiation scenarios were presented using the state-of-the-art negotiation agents. This chapter will focus on developing the negotiation strategy for the novel agent designed in this research work. The algorithm for the novel agent is proposed, and was tested against different autonomous negotiation agents. This chapter has been divided into four sections. Section one presents the negotiation scenario, while section two introduces the negotiation strategy for the proposed autonomous agent. Section three has evaluated the negotiation strategy. The conclusion is supplied in section four of this chapter.

7.1. Negotiation scenario

The negotiation scenario serves as an SLA template for the sake of service benchmarking and offers a large space of differing possibilities for both customers and cloud providers for negotiation. The negotiation scenario includes a total of 8 issues; the values, weight, customer evaluation and provider evaluation (Table 7-1).

Table 7-1: The negotiation scenario.

Issue	Value	Customer Evaluation	Weight	Provider Evaluation	Weight
Availability Zone	US-East 1 US-West US-central Canada-Central	25 50 10 15	0.24	35 25 15 25	0.28
Operating	Linux	70	0.16	40	0.11

System	Windows	30		60	
Contract (months)	[1-3] [4-6] [7-12] >12	5 30 40 15	0.07	25 50 10 15	0.07
Memory (GB)	7 8 9 10	15 50 25 10	0.11	25 50 15 10	0.12
App Services	Web App-W Mobile App-W API App-W Web App-L	10 50 30 10	0.15	20 10 50 20	0.07
Storage System	Amazon EFS Amazon EBS Amazon S3 Amazon SG	30 20 40 10	0.13	20 40 25 15	0.13
Platform	32-bit 64-bit	80 20	0.06	90 10	0.15
Utilization	Low <30% Med <50% High <75%	30 50 20	0.08	20 30 50	0.07

GENIUS platform was used to perform tournaments which involved the autonomous agents compiled by different researchers, which have the ability to negotiate autonomously over the given issues and exchange offers with each other using a bilateral variant called Rubinstein's alternating protocol. The utility (u) of the bid (b) is indicated by the equation 1 (Yaqub et al., 2011).

$$u\,(b) = \sum_{i=1}^{N} \omega_i V_i(x_i) \qquad (1)$$

Where u and b represent utility and bid, respectively, ω_i is the weight of *xi* issue; $\sum_{i=1}^{N} \omega_i = 1$ and $V_i(x_i) = eval\frac{(x_i)}{\max(eval(x_i))} \in [0,1]$ denotes the normalized values of i^{th} issue, which are shown in customer's evaluation and cloud provider's evaluation columns of Table 7.1. $eval$ is the evaluation function which determines the utility of each issue in the scenario. The cloud providers assign the evaluation function based on the scale of their resources, while the customers do so by considering the business demands.

The evaluation values of issues along with weights (priorities) form the preference profile of the customer and the cloud provider. The business objectives are defined based on utility functions, and agents try to maximize the utility function to which they represent without revealing any data regarding the utility values of opponents. The PaaS domain in the given scenario is reasonably large which might contain more than fifty thousand possible offers exchanged between the customer and the cloud provider. This task is very daunting for the human brain to handle; however, the intelligent agent can handle it efficiently within a reasonably short period of time.

Yaqub et al. (2014) showed that burden of the bidding on the human brain appeared in the form of reduced utility, while Chen et al. (2013) revealed that CHUCK agent was able to negotiate the deal within 2 minutes. Yaqub et al [2014] demonstrated that human brain was able to negotiate over only 83 rounds without achieving the deal due to the time factor taken by the human mind to process information and time required to make a decision, whereas the agent-to-agent negotiations could complete thousands of negotiation rounds without breaking off the negotiation process and simultaneously increasing the convergence rate. These data

clearly indicated that autonomous intelligent agents can negotiate without compromising the utility for their representatives compared to the human brain.

In this work we have considered the preferences of the customer and the cloud provider, which are often standing in conflict with each other, as is indicated by the evaluation indices. The customer assigns the highest value (70) to Linux as the operating system because of increasing utility for business, while the cloud provider assigns Linux the lowest value (40) due to increasing investment of the cloud provider in handling this resource. Therefore, the opposition between utility function of the cloud provider and the customer can be indicated by the following equation (Baarslag et al., 2011):

$$Opposition(\Omega) = \min dis\ \omega \in \Omega(\omega, \bar{\omega}) \qquad (2)$$

The $\omega \in \Omega$ denotes the utility space for all possible bids exchanged between two sides during the negotiation process. The *dis* is the Euclidean distance between $\omega\ and\ \bar{\omega}$, and $\bar{\omega}$ is the highest pay-off point (1,1) for all values of ω.

In this work, special attention is paid to the mutual benefit which assumes greater importance in any deal between transacting parties. The social welfare is another measure of fairness, which is the averaged sum of utilities of two agents in a negotiation tournament (Yaqub et al., 2011). The fairness is also measured through the social utility which is the sum of utility and social welfare in the negotiation tournament.

7.2. Negotiation strategy

Three main components were considered for modelling the negotiation strategy in this work. First, the biding function allowing the agent to generate offers and counter-offers during negotiation. Second, storage modelling is concerned with two main factors: time and storage. It stores offers made during negotiation and at the end of time without an agreement. Offers are reserved based on the highest utility. Third, the acceptance function enabling the agent to accept and conclude the negotiation session. Previous studies used complex methodologies to incorporate these functions into negotiation, which leads to an increased computational load on the negotiation platform. However, our objective in this study is to construct a negotiation strategy which can be adaptive in functions yet computationally inexpensive to carry out to produce cost-effective negotiations between the opponents. We label our negotiation strategy as "Aggressive Reaction" (AR) which is explained in the Algorithm in Table 7.2:

Table 7-2: Algorithm for description of aggressive reaction (AR) strategy.

Algorithm for acceptance function of AR showing generation of proposals and counter-proposals. The values of acceptable threshold are represented with $Reserved\ Utility = 0.8(U_r)$
Require: Current timeslot (t_i), Maximum timeslot (t_{max}), Selection bid by Customer (x_i), Utility for Customer ($U_{Customer}$), Utility for Provider ($U_{Provider}$), Total Utility (U_{Total}),Weight for customer (WC_i), Weight for Provider (WP_i), Value for Customer VC_i,Value for Provider VP_i,To Store Utility(U_i),Maximum Utility (U_{MAX}), Utility of offers (U_{List}), Evaluation of issue of Customer, Evaluation of issue of Provider.

$$U_{Customer} = \sum_{i=1}^{N} WC_i * VC_i \,/\, \max\left(eval(VC_i)\right) \quad (3)$$

$$U_{Provider} = \sum_{i=1}^{N} WP_i * VP_i \,/\, \max\left(eval(VP_i)\right) \quad (4)$$

$$U_{Total} = \frac{2 * (U_{Customer} * U_{Provider})}{(U_{Customer} + U_{Provider})} \quad (5)$$

1. **While** current timeslot (t_i) < the maximum timeslot (t_{max}) **Do**
2. $X(i) \leftarrow NewBid\ (Xi)$(Customer selects the options which has greater utility.)
3. $Calculete\ the\ value\ of\ (U_{Customer}, U_{Provider})$Eequation(3,4)
4. $U_{Total} = \frac{2*(U_{Customer}*U_{Provider})}{(U_{Customer}+U_{Provider})}$

   U_{List}=U_{List}+ U_i

   U_i =U_{Total} (Store utilities offered by opponents to show best utility (U_1, U_2, U_3... U_i)

   List= list U + U_i

   U_{MAX} = max (list U)
5. **If** U_{Total} > U_{MAX}

   U_{MAX} =U_{Total}(Store max utility)
6. **if** U_{Total} >U_r (Total Utility is higher than Reserved Utility)

   Print "Agreement"

   Return U_{Max}
7. **End While**

   Print "No Agreement. " (And Offer Best Utility)

   Return U_{Max}

The AR combines the strategies from tit-for-tat and hard-headedness in such a way that offers are proposed based on the both sides of the opponents. It can allow the agent to produce concessions if the opponent does so and vice versa because the utility of final negotiations is result of utility of the provider and the utility of the opponent together not only one of them and this why the AR is different.

The AR starts the process of negotiations by an initialisation phase where a reserved utility (Ur) is predefined as well as a maximum negotiation timer is configured. These parameters are internal ones and unknown to the opponent and their values are essential in shaping the negotiations' strategy of the proposed AR software.

Then the AR starts sending the first offer with the current bid and receiving an offer from an opponent with a different utility.

During the negotiation, the utility of a customer is calculated by dividing the multiplication of allocated Weight for a customer and its value for that customer by the highest evaluation value of the customer for the targeted individual issue as shown in Eq. 3. The same calculation is applied to a provider in calculating its utility as shown in Eq. 4. The AR takes into account that the total weight of issues is up to 1 for customer and provider and a provider and a customer will provide a new value of evaluation for every issue. By doing this, the AR maps all potential The function of utility for customer and provider maps all potential results to valued numbers in a range between 0 and 1. The utility includes a valued and a weighted sum of customer or provider utility for every issue.

$$U_{Customer} = \sum_{i=1}^{N} WC_i * VC_i \,/\, \max\big(eval(VC_i)\big) \quad (3)$$

$$U_{Provider} = \sum_{i=1}^{N} WP_i * VP_i \,/\, \max\left(eval(VP_i)\right) \quad (4)$$

One of the most important advantages of AR is that it cares about the utility for both the provider and the opponent, so AR tries to benefit both sides and to provide the highest utility as much as possible as shown in equation 5. In Eq. 5, the value of total utility is defined by dividing the two times of the multiplied utilities of customer and provider by the summation of their utilities.

$$U_{Total} = \frac{2 * (U_{Customer} * U_{Provider})}{(U_{Customer} + U_{Provider})} \quad (5)$$

During the negotiation, AR will keep a record of all the proposed offers sorted by their highest utility. This step is a key to facilitate a reference to decision-makers in selecting the best offer to send to a customer when the negotiation ends without an agreement. The AR software displays all the key negotiations details as well as stores them in an external log file. These details includes all the issues, the utility of the customer, the utility of the provider, the total utility, and the negotiation status at each time. As the output data is sorted by descending based on the offers' total utility values, the best offer with the highest value will be at the first reported offer in the log file. This will optimise the workflow of the analyst in deciding what to send to a customer when the negotiations are failed to reach an agreement with the customer. If the AR found a total utility that is bigger than the reserved utility, then it will end the negotiation process and automatically accept the offer and arrange the agreement. Otherwise, it will end the negotiations process with a rejection and will go back to start new negotiations and it is up the analyst or decision maker to make the call.

Finally, once the algorithm is executed, the deal is accepted and executed consecutively.

7.3. Experimental evaluations

In this section, the experimental evaluations of AR in bilateral and multi-lateral negotiations are presented.

7.3.1. Performance of AR in bilateral negotiations

The first part of the experimental evaluation shows the performance of AR as a cloud provider in bilateral negotiation against IAMhaggler, CUHK, Nice Tit-for-Tat (NTFT) and hardheaded. The negotiation sessions were run in GENIUS against the scenario shown in Table 7.1 in Section 7.1. The results are described below:

7.3.1.1. AR versus IAMhaggler

The agreement between two agents was reached after 15,909 rounds. The customer's agent IAMhaggler 2012 begins the negotiation session with this offer: (Offer bit: Bid [Availability zone: US-East 1, Operating system: Linux, Contract (months): [1-3] years, Memory: 8GB, App Services: Mobile App-W, Storage System: Amazon S3, Platform: 32-bit, Utilization: Med <50%,]). This package has 1.0 utility for IAMhaggler 2011 and 0.488 utility for AR agent. AR agent computes the preferences, utilities and concession rates of the opponent in order to determine the offers which can possibly lead to an agreement. The bid from IAMhaggler 2012 did not reach the Nash point of AR agent. Therefore, AR rejected this offer, and suggested the following counter offer to IAMhaggler 2012 agent: (Offer bit: Bid [Availability zone: US-East 1, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: High <75%,]). This offer carries utility of 0.353 for IAMhaggler 2011 and utility of 0.989 for AR agent. IAMhaggler rejected this offer.

Based on the preference profile and utilities of offers exchanged by AR agent, IAMhaggler2011 continues to increase the concession rate for the opponent slowly but gradually, which lingered on the negotiation session up to 15,909 rounds. AR agent also continued to adjust its utilities and concession rate based on the concession rate and utilities of the previous offers of IAMhaggler2011 to target the Nash point calculated based on the opponent's utilities and weights.

Both sides kept sending offers and counter-offers; and the deadline was set at 180 seconds. The agreement between AR and IAMhaggler 2012 was reached in the 15,909th round of negotiation, while negotiation time elapsed was 120.678 seconds. The bid agreed was: Bid [Availability zone: US-East 1, Operating system: Linux, Contract (months): [4-6] years, Memory: 8GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: Med <50%,]), which carried 0.81 utility for IAMhaggler and 0.85 for AR. Thus, both agents benefited from the bid by securing utilities closer to their target utilities (Figure 7-1).

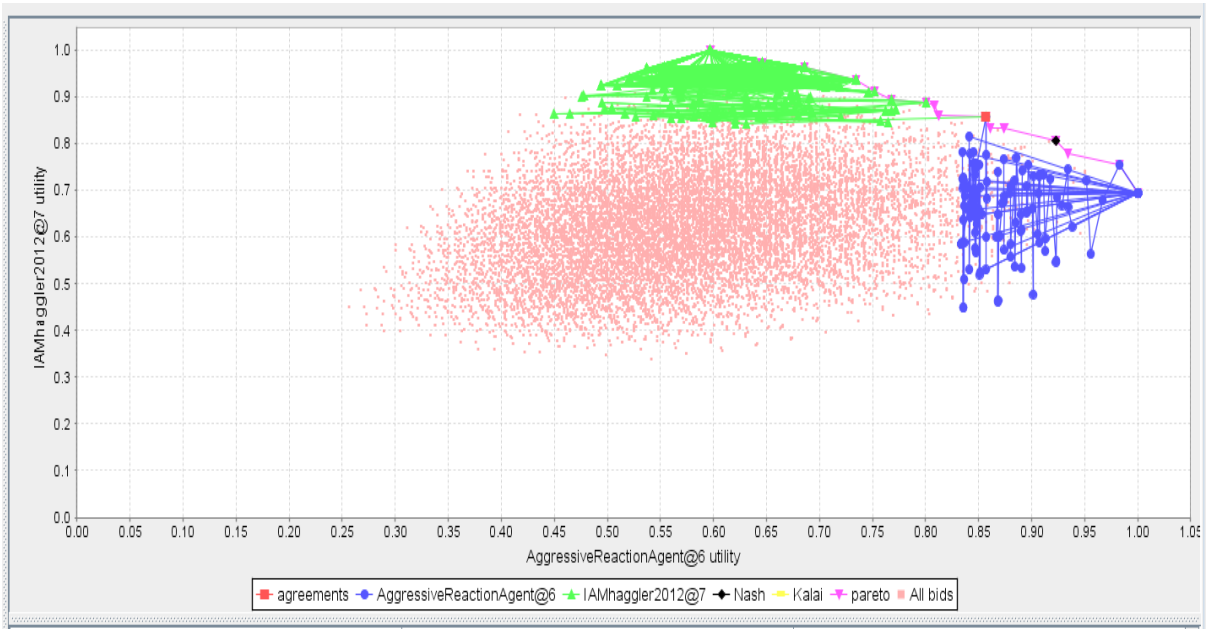


Figure 7-1: Simulation result for AR versus IAMhaggler

7.3.1.2. AR versus CUHK 2015

The negotiation session between AR and CUHK 2015 was run in GENIUS by setting the deadline 180 seconds. The agreement between AR and CUHK 2015 was reached at 13,680 rounds at 117.780 seconds. The CUHK 2015 represented the customer, while AR acted on behalf of the cloud provider in the negotiation session. CUHK 2015 opened the negotiation session with this offer: (Offer bit: Bid [Availability zone: US-Central, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon S3, Platform: 32-bit, Utilization: Med <50%,]). This package had 1.0 utility for IAMhaggler 2011 and 0.288 utility for AR agent. As the utility of the bid was far lower than the target utility of AR agent, therefore, it rejected the offer, and suggested the following counter offer to the CUHK 2015: (Offer bit: Bid [Availability zone: US-East 1, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: High <75%,]). This offer carried utility of 0.453 for IAMhaggler 2011 and utility of 1.0 for AR agent. IAMhaggler rejected this offer due to the difference between its target utility and the offered utility.

Both sides kept exchanging offer and counter-offer until the agreement between AR and CUHK 2015 was reached in the 13,680th round of negotiation, while negotiation time elapsed was 120.678 seconds. The bid agreed was: Bid [Availability zone: US-West, Operating system: Windows, Contract (months): [4-6] years, Memory: 8GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: High <75%,]), which carried 0.82 utility for IAMhaggler and 0.87 for AR. Thus, both agents benefited from the bid by securing utilities closer to their target utilities (Figure 7-2).

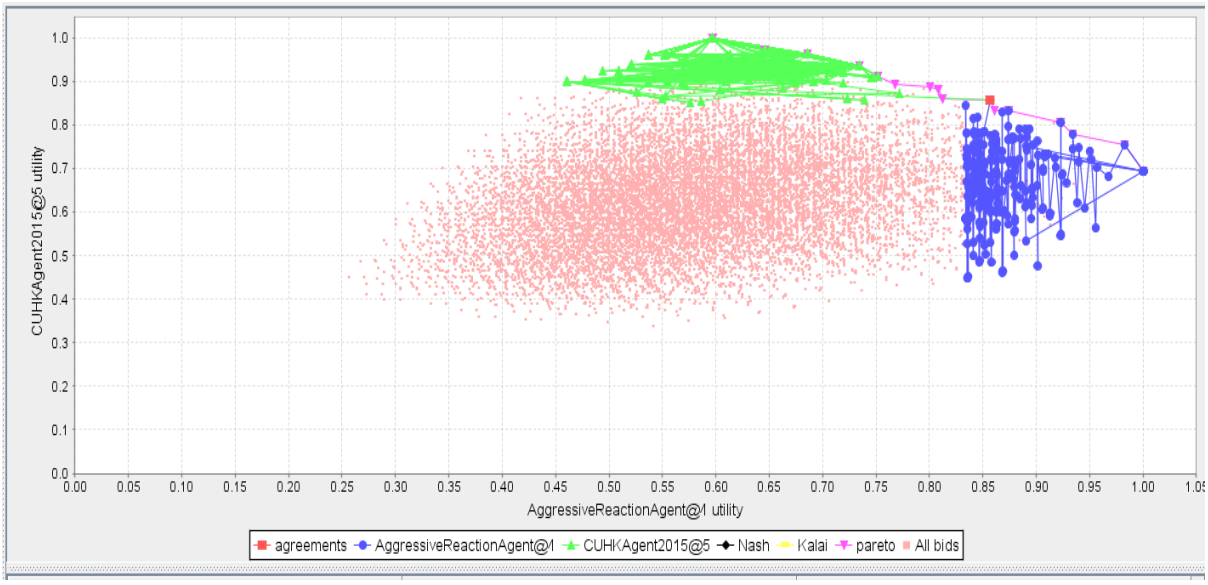


Figure 7-2: Simulation result for AR versus CUHK 2015 agent

7.3.1.3. AR versus Nice Tit-for-Tat

In this experiment, AR competes with NTFT. NTFT opens the negotiation session by giving the following bid to AR: (Offer bit: Bid [Availability zone: US-Central, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon S3, Platform: 32-bit, Utilization: Med <50%,]). This package has 1.0 utility for NTFT and 0.411 utility for AR agent. IAMhaggler 2011 agent computes the preferences, utilities and concession rates of the opponent in order to determine the offers which can possibly lead to an agreement. Although the bid offers the bid with good utility to AR based on the preference profile of the agent, it did not reach the Nash point of the AR agent. Therefore, AR rejected this offer. The AR agent offered the following counter offer to NTFT agent: (Offer bit: Bid [Availability zone: US-East 1, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: High <75%,]). This offer carries utility of 0.363 for NTFT and utility of 0.999 for AR agent. NTFT rejected this offer.

Both sides kept sending offers and counter-offers; and deadline was set at 180 seconds. Finally NTFT and AR agreed to the following bid at the 16,009th round of the negotiation: (offer bit: Bid [Availability zone: US-East , Operating system: Windows, Contract (months): [4-6] years, Memory: 8GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: Med <50%,]), This offer has utility of 0.746 for NTFT and utility of 0.895 for the AR agent (Figure 7-3).

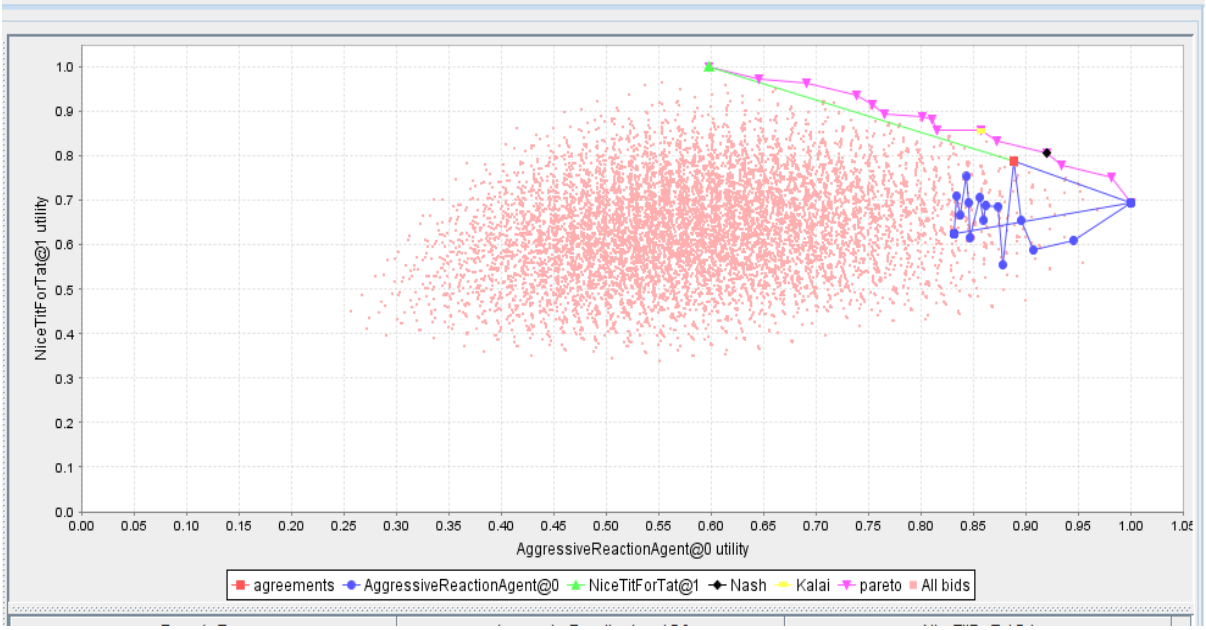


Figure 07-3: Simulation result for AR versus NTFT agent

7.3.1.4. AR versus hard-headed

In this experiment, AR competes with hard-headed (KLH) agent in a bilateral negotiation session with deadline 180 seconds on GENIUS platform. Hard-headed opened the negotiation session with the following bid: (Offer bit: Bid [Availability zone: US-Central, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon S3, Platform: 32-bit, Utilization: Med <50%,]). This package had

1.0 utility for hard-headed and 0.323 utility for AR agent. As the utility of the offered bid was not compatible with the target utility of AR agent, therefore, it rejected the offer, and suggested the following counter offer to the hard-headed agent: (Offer bit: Bid [Availability zone: US-East 1, Operating system: Linux, Contract (months): [1-3] years, Memory: 7GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: High <75%,]). This offer carried utility of 0.433 and 1.0 for hard-headed agent and AR agent, respectively. The hard-headed agent rejected this offer due to a large difference between its target utility and the offered utility.

Both sides kept exchanging offers and counter-offers until the agreement between AR and hard-headed agent was reached at the 13,680th round after 120.710 seconds. The bid agreed was: Bid [Availability zone: US-West, Operating system: Windows, Contract (months): [4-6] years, Memory: 8GB, App Services: Mobile App-W, Storage System: Amazon EBS, Platform: 32-bit, Utilization: High <75%,]), which carried 0.82 utility for IAMhaggler and 0.87 for AR (Figure 7-4).

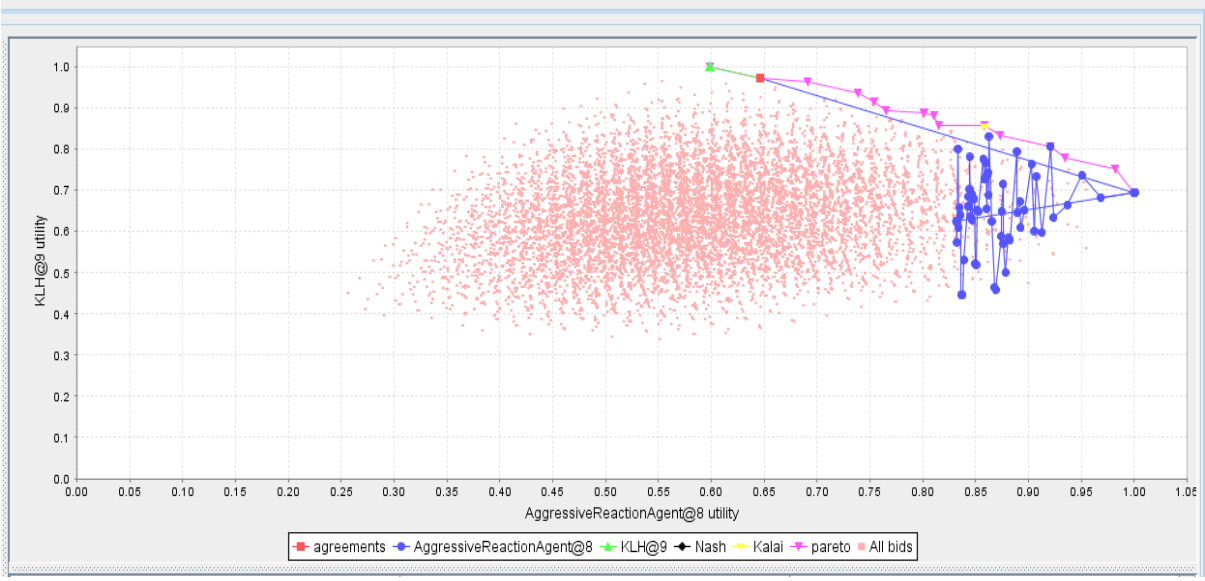


Figure 07-4: Simulation result for AR versus KLH (hard-headed) agent

For most of bids, hard-headed did not give much discount to the AR and hovered around the utility 1.0-0.98, while approaching the deadline, AR gave the discount to hard-headed while compromising its utility. However, the AR could not achieve the higher utility level and remained adhered to its reserved utility (0.65), but it showed more friendly and social behaviour to the opponent compared to the opponent.

7.4. Utilities and social welfare of AR

The performance of the AR agent is evaluated against the state-of-the-art opponents with both learning functions. The aggressive reaction agent was simply called 'AggressiveReactor' for the sake of convenience, and it was depicted in graphs used in this section. Hence AR denotes 'Aggressive Reaction Agent' or 'AggressiveReactor'. The first tournament was run by involving three agents: AR, IAMhaggler 2012, and Value Model

Agent. The second tournament was run by selecting the agents which were top-performers in the previous ANAC competitions; they involved the AR, CUHK, the MetaAgent, Fawkes. The third tournament involved the agents: AR, hard-headed, Nozomi, and Yushu. We have chosen all agents participating in Tournament # 1, #2 and # 3 for running the tournament 4.

The social utility (fairness) and individual utilities (optimality) were used to score the top-performers in these tournaments. Furthermore, the combining all agents in one tournament in this study showed how variations exist in the marketplace due to differing strategies of the agents. Because this study intends to support the procurement of cloud services, we have decreased the deadline from 3 minutes (180 seconds) as used in ANAC competitions to 2 minutes (120 seconds).

7.4.1. Results from Tournament 1

AR, IAMhaggler 2012 and Value Model Agent are included in this tournament, which use different learning methods to model the behaviour of opponents. Figure 7-5 shows the pairwise negotiation outcomes of the tournament. The social utility and individual utility of the AR were higher than its opponents. This indicated that AR was a clear winner of the tournament 1, while the IAMhalgger2012 won the second position. The AR showed Boulware-like behaviour towards opponents without giving concessions, and from beginning to the end of the negotiation, the AR continued to retain its highest utility value, which was in conflict with the standard tit-for-tat strategy.

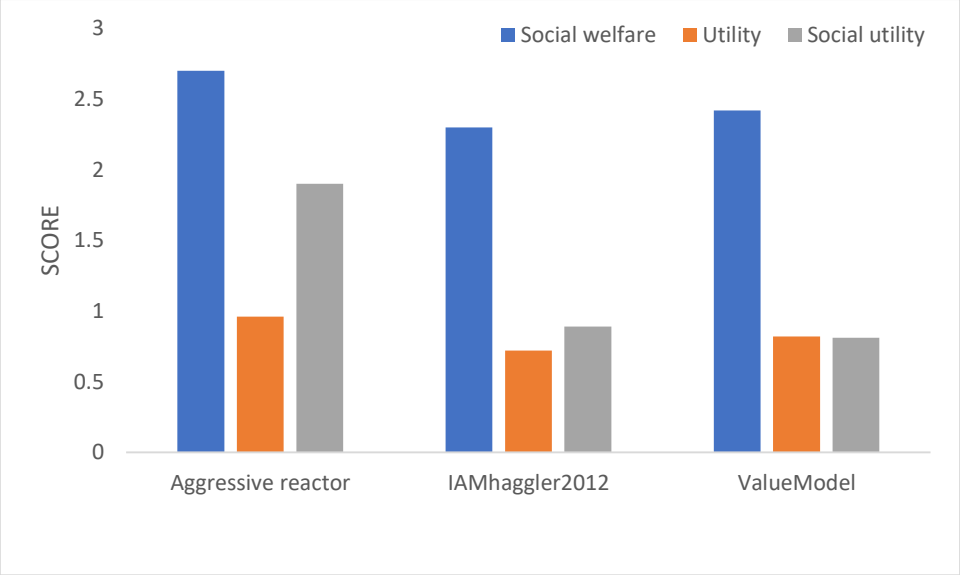


Figure 7-5: Experimental results for Tournament 1

7.4.2. Results from Tournament 2

This tournament includes agents which are already nominated as top 10 performers in previous ANAC competitions. AR, CUHK (winner of ANAC 2012), the MetaAgent (ANAC 2013), Fawkes (winner of ANAC 2013). These agents are reported to be very competitive due to having ability to use complex learning process and prediction models to model the opponents' behaviour. Although NTFT acted with a Boulware-like strategy in the beginning, after elapsing of 80% of the negotiation times, it yielded to CUHK and Fawkes which extracted more concessions from the AR, and caused significant reduction from 0.98 to 0.73 in the AR's utility.

The results of the tournament are presented in the Figures 7.6, 7.7, and 7.8. Based on the outcomes of tournament. AR and CUHK achieved the first position because both agents yielded the same social welfare and individual utility to their clients. MetaAgent scored the second position (Figure 7-6).

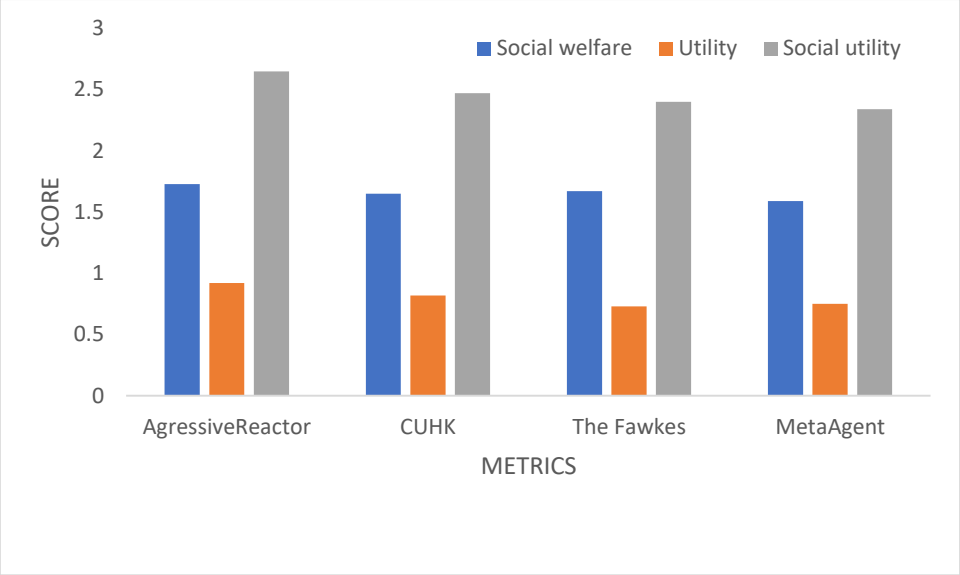


Figure 7-6: Experimental results for Tournament 2.

7.4.3. Results from Tournament 3

This tournament was run with the four agents: AR, hard-headed, Nozomi, and Yushu. AR negotiates as cloud customer with its opponents (cloud providers). AR received concessions in its 92% of moves, while it had given concessions to its opponents in 8% of moves. Even the concessions yielded by AR to its opponents was in the range of 0.1-0.5. The results from the negotiation are shown in Figure 7-7.

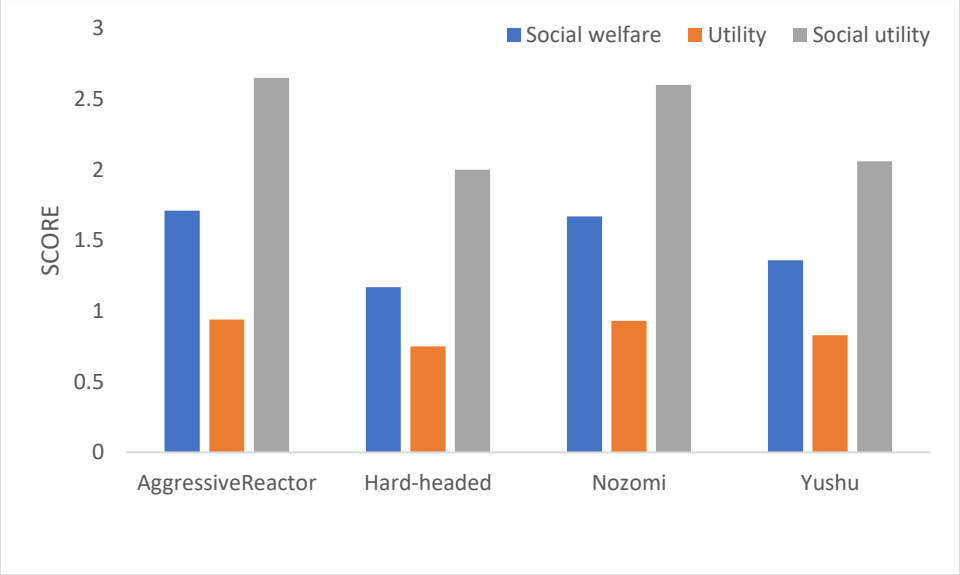


Figure 7-7: Experimental results for Tournament 3.

Nozomi and Yushu are demanding strategies which split the negotiation time into two intervals, and they use sub-strategy for each interval. In the first interval, they made offers with highest utility, and in the second interval, they lower or increase their utilities depending on the opponents' behaviour. AR obtained high concessions from Yushu and Nozomi due to the more aggressive nature of AR with making bids carrying the highest utility in the beginning. After passage of 60% of the negotiation time, Nozomi and Yushu carried out with making bids with high utility, which made AR yield to them. The demanding behaviour of Nozomi and Yushu in the end caused a decrease in AR's utility compared to the opponents. Competing with hard-headed agent, AR used its aggressive exploitation strategy to benefit from the moves of opponents with lower utility to increase its utility. Thus, AR increased its pay-offs against hard-headed agent. However, AR scored the highest social welfare and social utility, which made it the winner of the tournament, while Nozomi obtained the second position in the tournament.

7.4.4. Results from Tournament 4

The tournament 4 was run with all agents participating in the previous tournaments in order to show the strongest and weakest negotiation strategies. The final outcomes of the tournament are displayed in the Figure 7-8.

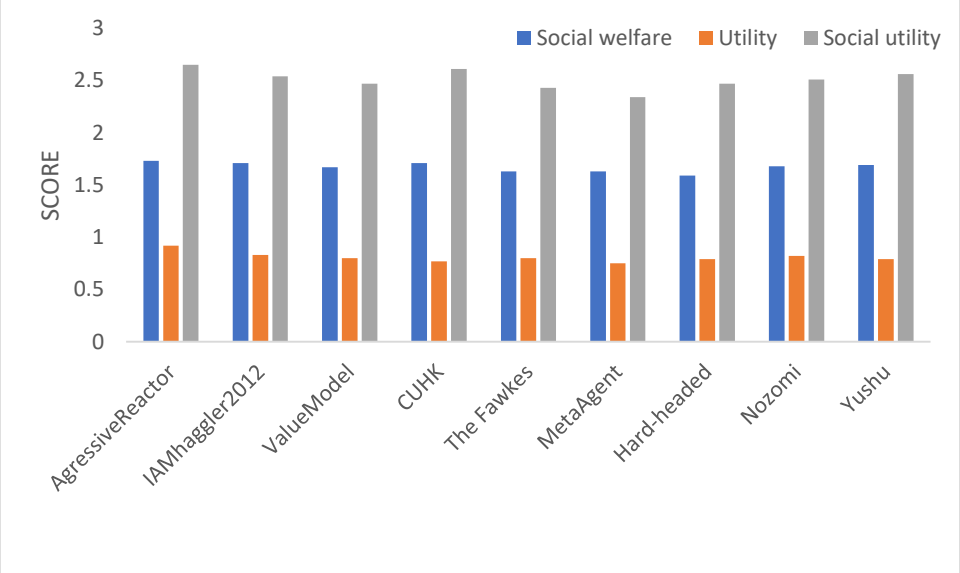


Figure 7-8: Experimental results for Tournament 4.

AR scored the number 1 position due to winning the highest social welfare and social utility, followed by The Fawkes, CUHK and Yushu.

7.5. Discussion

During bilateral negotiations, AR showed high performance compared to its opponents in terms of making the agreement beneficial to both parties involved in the negotiation process. In all cases, except negotiation with hard-headed, AR secured a utility of more than 8 and the opponent also obtained good utility though lower than AR. There was not a big difference between the utilities of AR and the opponent, which fulfilled the criterion of very good deal

between AR and the opponent. The negotiation is termed as lucrative and beneficial if both parties stand at a win-win situation. Negotiation agents with ability to mediate the win-win situation are more socially acceptable to boost the businesses in the cloud market (Yaqub et al., 2014)

The salient feature of the AR was that it did not exploit its opponents due to the R function introduced in aggressive reaction strategy used by AR, which suppresses the selfish behaviour of AR like other agents such as hard-headed, and allows AR to reach an agreement with the opponent, which is beneficial to both parties participating in the negotiation. Another key and distinguishing feature of AR was that it started with giving discounts and concessions to opponents for the first 20% of the negotiation, which was an indication of generosity in the beginning from AR in order to motivate the opponent to continue with the negotiation process.

Learning from the behaviour of opponents, AR showed hard-headedness after the elapse of 20% of negotiation time. The acceptance module become activated in the last 10% of negotiation time with tendency of AR to select and accept the best offer from the bidding history of the opponent. The proposed aggressive reaction strategy implemented by aggressive reaction agent thus proved to be useful in yielding the beneficial negotiation outcomes for most of agents tested in this study. However, it could not yield the higher utility against hard-headed agent, which was due to the hard-headedness shown by both AR and hard-headed agent during most of the negotiation time. Despite the persistent hard-headed behaviour from the hard-headed agent, the acceptance module of the AR was able to select and agree to the bid which was lower than its target utility but higher than its reserved utility. Hence AR favoured the execution of negotiation rather than breaking it down.

During the multilateral negotiation experiments, it was discovered that the utility gaps in the tournaments 1, 2 ,3 and 4 were 36%, 25% and 28%, and 43% respectively, which showed that as learning increases in the market, the market competitiveness also increases. In addition, the mean market utility was created for tournaments 1, 2, 3 and 4, which were found to be 0.75, 0.73, 0.71 and 0.80 respectively. This showed that overall trends of market competition and gains for both customers and cloud providers were stable, even though some weaker strategies participated in this study. Despite the participation of the weaker agents, no agent crashed or quit the tournament. In all tournaments, the social welfare values were maintained between 171-173 with marginal gap, which indicated that the mutual benefits were received by the participating parties. This is also indicative of the fact that participation of the different strategies in the market is most likely to increase the joint gains received by markets, rather than using the homogenous negotiation strategies. This argument is further reinforced by the mean social utility values which stood at 2.41, 2.33, 250 and 2.45 in tournaments 1, 2, 3, and 4, respectively.

AR made use of the simple strategy based on the Boulware-like strategy and perceived hard-headedness to frame its bidding function, which helped it to avoid the freefalling of its concessions to the opponent. It uses the learning and modelling of its opponents' utility to continue with its high utility during the negotiation process, which was evident from most of AR's bids falling on the pareto-frontier.

The outcomes of the tournaments demonstrated that AR was able to secure the high utility margins against its weaker opponents, but it neither compromised its utility to a damaging level against the stronger opposition nor broke off the any negotiation. The matches in whom AR had lost in terms of securing lower utility, the opponents won only with 16% utility margin. However, the matches in which AR won in the sense of individual utility values, it

had secured 38% utility margins against the opponents. AR won all matches against hard-headed agent, IAMhaggler2012, The ValueModel Agent, Yushu and MetaAgent.

However, it has lost some matches to CUHK, but it has lost all matches against The Fawkes and Nozomi. Taken together, these data indicate that AR is the viable negotiation strategy to achieve the advantage and dominance over different metrics used in this study. AR showed a strong performance in areas of social utility, social welfare, and all agreements reached by AR with its opponents showed fair distribution, indicating the satisfaction of the opponents. This has implication for the cloud providers who can benefit from this strategy against the weaker and stronger opposition in the marketplace. The cloud providers using AR are in a better position to make a fair distribution of utilities for the customers (opponents), thereby entailing the satisfaction of the clients.

7.6. Conclusion

We have presented an AR negotiation strategy which can be used by the cloud providers in the cloud services-based marketplaces to dynamically negotiate with the state-of-the-art opponents to produce a fair and optimal SLAs. We have employed a PaaS domain with high-conflicting issues to propose a robust AR strategy for cloud-based service providers.

CHAPTER 8: MONITORING FRAMEWORK

In the previous chapter, results relating to the negotiations between the cloud providers and cloud customers were presented. The different parameters within the negotiation scenarios were considered and negotiated over by the negotiating parties. After successful negotiation, the SLA was drafted, and the subsequent task was to ensure that cloud providers comply with the terms and conditions within the SLA framework. For this purpose, an SLA monitoring framework has been presented in this chapter. The first section of this chapter presents the simulation architecture developed for monitoring the compliance of cloud providers with the metrics specified in the SLA framework. The second section presents the algorithm employed for monitoring in the monitoring engine used in this study. The third section evaluates the proposed monitoring framework using different scenarios. The chapter has been concluded in the fourth section.

8.1. Monitoring SLA

The monitoring of SLA is important for ensuring the quality of services provided to the customers, and plays a key role in promoting the cloud services across various sectors. In the previous chapter, negotiation sessions were conducted using different issues in domains of negotiation with a view to reaching an agreement between the cloud provider and the customer. The metrics agreed upon between the cloud provider and customers as part of the SLA in the previous chapter need to be monitored using a robust monitoring mechanism, so that compliance of the cloud providers with the quality of services can be executed in the cloud market. This study included three metrics: storage, response time, and virtual machines for testing the monitoring framework proposed in this study. The data relating to these metrics were gathered from the real time cloud market environment, and were negotiated between the customer and the cloud provider in Chapter 6.

8.2. Monitoring Framework

This section presents the monitoring framework which is proposed to detect the violations against the agreed metrics in the SLA. As a result of negotiation, the SLA was drafted stating the agreed metrics, and agreement on the metrics to be monitored continuously in order to ensure the best quality of services from the cloud providers in the cloud market. The data provided by the SLA documents are stored in the Cloud database. SLA document specifies the agreed metrics and metrics agreed to be monitored, as all metrics might not be worth monitoring. Moreover, data from the SLA document is also shared with the monitoring engine for referencing and comparison of services provided to customers during execution of the monitoring function.

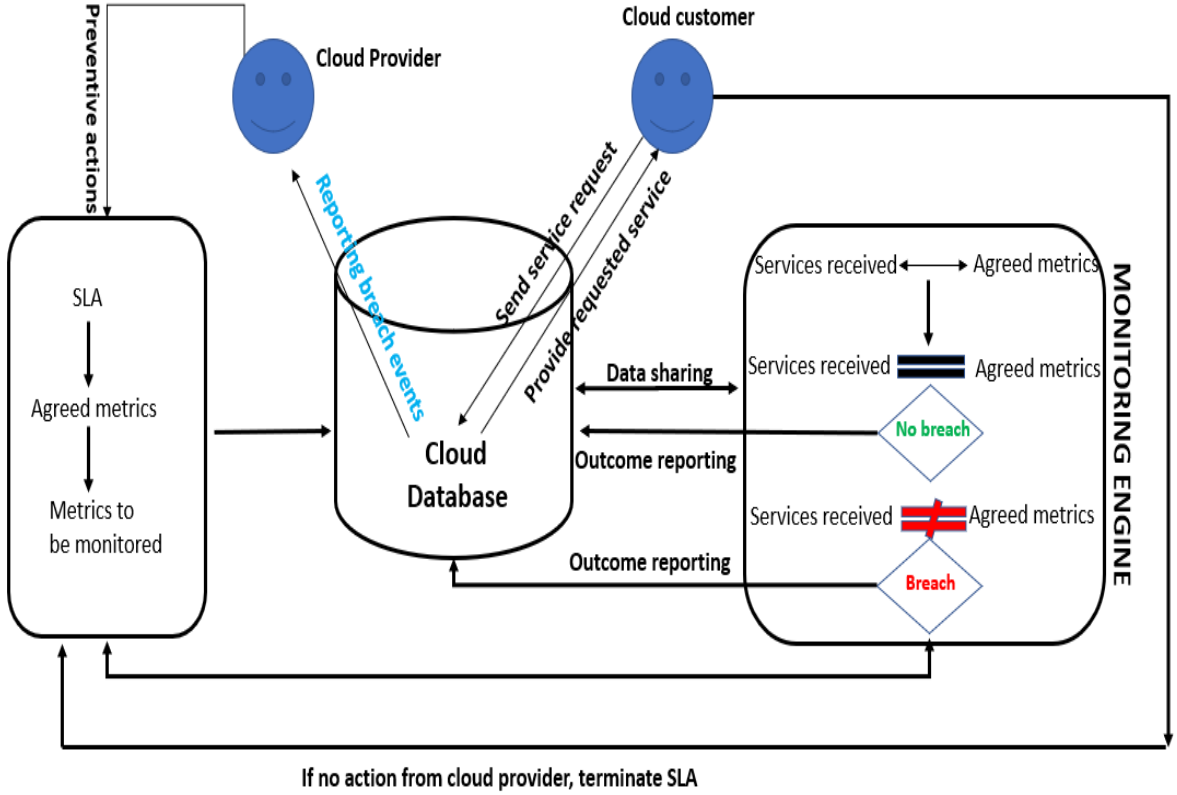


Figure 8-1: The detailed description of the monitoring framework developed to monitor the metrics agreed upon by the parties signing SLA.

In Figure 8-1, the customer sends the service request to the cloud database which provided the requested services. The data regarding the requests from customers and services provided are shared with the monitoring engine, which compares the provided services with the agreed metrics. If the services received by the customers are in agreement with the agreed limits and within limits, and cloud database provides the services requested, the monitoring mechanism detects that there is no violation of the agreed metrics in the SLA.

However, if the service received by the customers violates the limits of the agreed metrics, the monitoring function detects it as violation of the SLA. The outcomes of 'No violation' and 'SLA violations' are communicated with the cloud database which issues notifications to the cloud providers in the event of detection of SLA violations. The cloud provider takes

preventive measures to stop the occurrences of SLA violation to ensure the continuity of the SLA. Nevertheless, if no appropriate action is taken by the cloud provider, the cloud customer takes action to terminate the SLA.

8.3. Simulation architecture of monitoring SLA framework

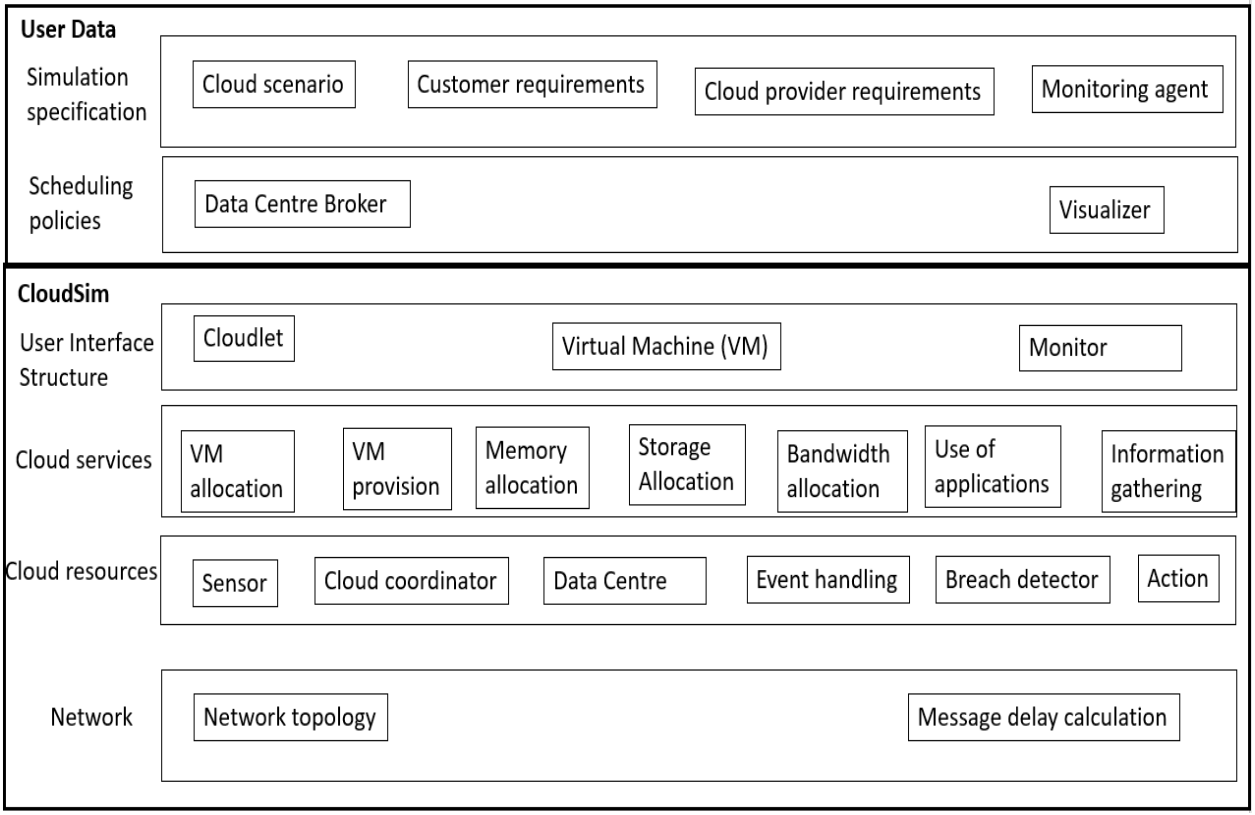


Figure 8-2: Simulation architecture of monitoring SLA framework

In Figure 8-2, the CloudSim simulation architecture for autonomous monitoring of the SLA framework is presented. The monitoring agent is a component which will analyse the request and services in accordance with the agreed SLAs and will generate warnings and alerts if the services provided by the cloud providers are not in accordance with the agreed terms and conditions in the SLA between the customer and cloud provider. Visualizer is a component

which applies various algorithms in order to adjust the VMS to different workloads so that customers' requirements can be fulfilled. Monitor module in CloudSim monitors the users' request and services of cloud providers according to the agreed SLA. The Info Gathering module collects all data from the modules interacting with each in the architecture.

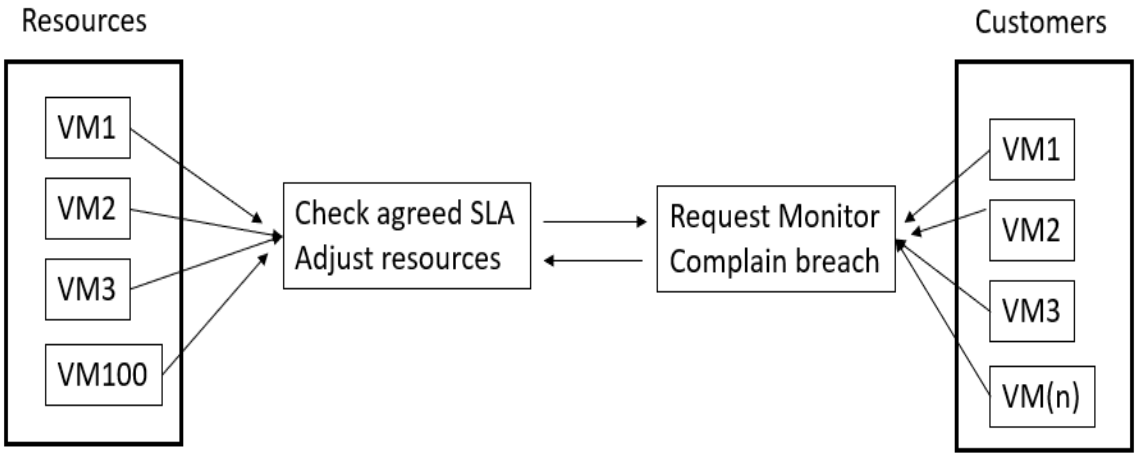


Figure 8-3: Simulation of autonomous monitoring SLA

In Figure 8-3, a pool of resources was created, which contained 100 VMs on behalf of cloud providers. The 'n' number of VMs was generated, which sends a request to monitor the flow of services from the cloud provider or request from the customer. The 'Monitor' module obtains the metrics of resources in the form of XML files, which are forwarded to the communication channel. In order to check the monitoring performance of the simulation architecture, a large number of resources' VMs and customers' VMs are generated to generate the real environment, so that the performance of the simulation architecture can be checked in the real environment. In order to test the functioning and performance of the proposed monitoring framework, the following parameters will be tested: 1) Response time, 2) storage, 3) virtual machines.

The monitoring module of the CloudSim works in accordance with the algorithm developed for monitoring the aforementioned specific parameters of SLA in this study. The proposed algorithm employed by the monitoring module is given in the next section.

8.3.1. The proposed algorithm

This section presents the algorithm which was implemented in the CloudSim's monitoring module to carry out the monitoring function.

Initalize the cloudSim package(libaray)

Identify the number of cloud customers

Create datacenter with 100GB storage and 5GB RAM

Create Broker

Create Virtual machine list

Add Virtual Machine with 4GB RAM, 97GB Storage,1GB bandwidth, 1 CPU and cloudletSchedularTimeShared object

Add Virtual Machine to Virtual Machine List

Submitted Virtual Machine list to broker

Create 'n' cloudlets submitting with following output file size

- *1st 22 cloudlets with 2GB file size*
- *Next 7 cloudlets with 4.5GB file size*
- *Remaining 5 cloudlets with 6.5 GB file size*

Add cloudlets to cloudlet List

Submitted cloudlet List to broker

11. Mapped CloudSim entities to BRITE entities PowerDatacenter corresponded to BRITE node 0

12. Broker corresponded through Brite Node3 for network topology

13. Cloudsim simulation started

14. Got Cloudlet Received List

Broker check:

If all cloudlets executed

Then Print "output simulation"

Else

Print "error happen"

End if

Broker receive amount of storage requested by cloudlet

Broker allocate the amount of storage requested to the cloudlet

Begin if

If cloudlet requested > = cloudlet provided

Output "no SLA violation"

Else

Output "SLA violation"

Finished the CloudSIm simulation

Printed the Cloudlet received list

The above algorithm is presented in the form of a flowchart to sequence the various actions in order to detect the violations of the terms and conditions specified in the SLA.

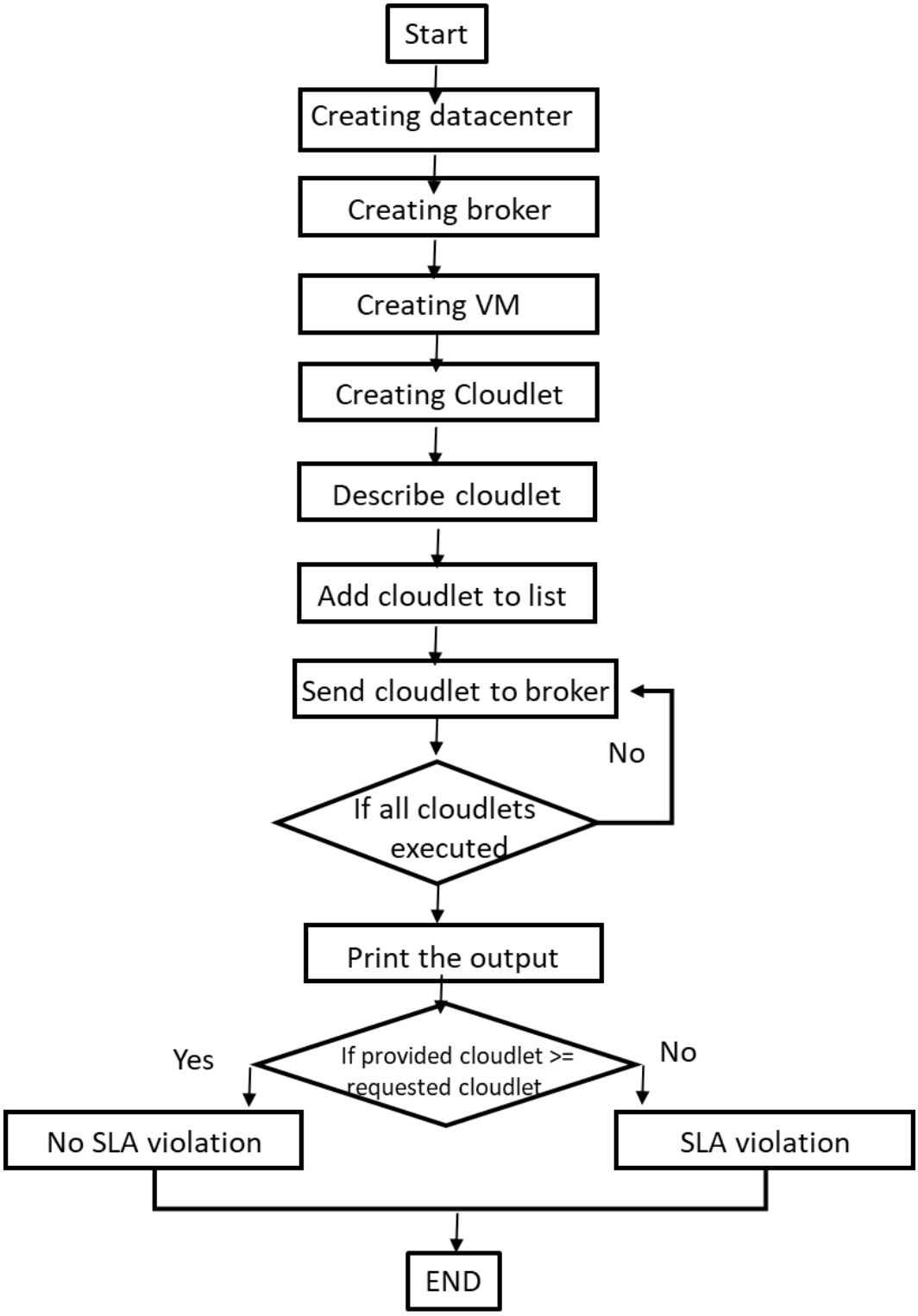


Figure 8-4: The flow of activities while implementing the monitoring framework within CloudSim

8.4. Monitoring Scenario: Details of SLA agreement with Service Level Objectives (SLO)

The following metrics as shown in Table 8-1 will be part of the SLA agreement which will be monitored using the CloudSim.

Table 8-1: The SLA metrics for evaluation of monitoring function of the proposed monitoring framework.

SLA parameters	SLO values	Threshold values
Response Time	≤10,000 milliseconds (ms)	≥10,100 milliseconds (ms)
Storage	4 GB	3.98 GB
Virtual Machines	14	≤14

8.4.1. Rules for Monitoring

- Monitor SLA metrics against the set SLOs for 50 seconds, and repeat the Monitoring after 10 seconds for an initial 10 rounds, and if there is no violation detected, then extend the repetition time after 30 seconds, and if no violation found for 10 rounds, then extend the monitoring repetition intervals for 50 seconds.
- The Cloudsim tests the monitoring of one metric at a time in order to check the compliance of the cloud service providers with terms and conditions set out in the SLA framework.
- Threshold values will be set as flag values for detection of violations in SLA agreement
- Report the level of compliance of cloud provider to the SLOs.
- Algorithm for violation detection and poor performance of SLA will be developed

- Algorithm for cloud provider for efficient resource allocation and improvement of services will be developed to inform resources management policies. The triggers for resource allocation (memory, number of virtual machines, applications etc.) will be derived from the monitored SLA.
- Report the performance of SLA for the following number of tasks (workload): 2,000, 4,000, 6,000, 8,000, 10,000.
- Report the number of violations/predict the number of violations within 24 hours
- Check the performance of SLA against three policies: Resource utilization policy, Maximize Throughput Provision Policy, Maximize Utilization Provision Policy and Minimize response Time Provisions.

8.4.1. Monitoring of Storage

The metric of storage agreed between the cloud provider and customer was the maximum storage Limit = 4GB, and threshold Limit = 3.98 GB. If the cloud provider provides storage up to 4GB, then there is no violation, otherwise there will be SLA violation.

The algorithm described earlier was implemented in CloudSim in order to monitor whether the storage as agreed in the SLA is available to the cloud customers. Three scenarios in relation to storage were found during monitoring.

The outputs generated by CloudSim, followed by exportation of data into notepad and Word for presentation purposes. The data produced by CloudSim contained different columns including Cloudlet ID assigned by the user, status showing the execution completed by the simulator, data centre ID showing which data centre is processing the cloudlet in the simulator, VM ID representing the virtual machine assigned by the data centre to process the cloudlet, time representing the execution of cloudlet by the simulator, ‘start time’ and ‘finish

time' denoting the start and end of the simulation task for a particular cloudlet, output file size in bytes, and result showing the outcome of the monitoring in the form of SLA violation or 'SLA no violation'. The screenshot of output file from CloudSim exhibiting all aforementioned columns is shown in Figure 8-5.

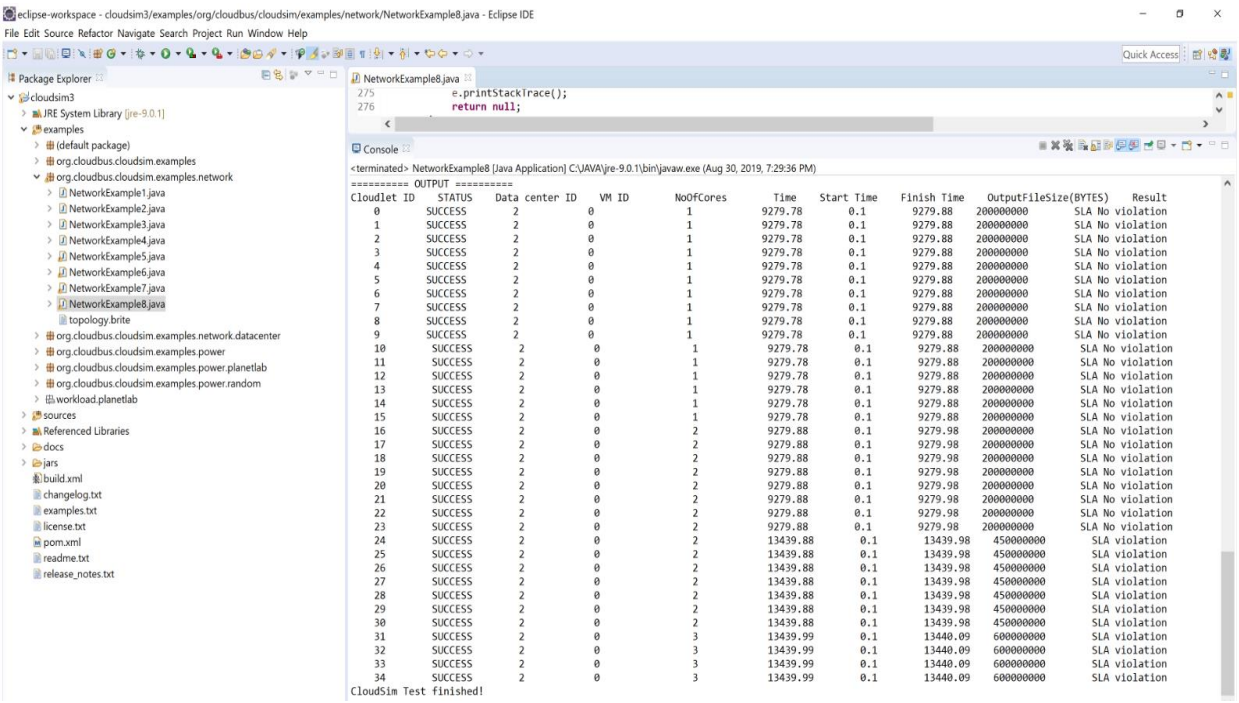


Figure 8-5: The screenshot showing the output of the proposed monitoring framework within CloudSim.

8.4.1.1. First scenario

The storage allocated to the cloudlet was greater than the storage requested by the cloudlet, which means that there was no SLA violation (Table 8-2, Figure 8-6).

Table 8-2: First Scenario

Storage allocated	Storage requested	Status

3GB	2.5GB	No SLA violation

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	OutputFileSize(BYTES)	RESULT
0	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
1	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
2	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
3	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
4	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
5	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
6	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
7	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
8	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
9	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
10	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
11	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation

CloudSim Test finished!

Figure 8-6: The outcomes of monitoring of storage allocated to the cloud customer

As shown in Table 8-2 and Figure 8-5, the monitoring mechanism detected that broker has allocated 3GB storage space for storing files to users, while the cloud customer requested 2.5GB storage space. Therefore, the monitoring framework detected that there was no violation against the SLA.

8.4.1.2. Second Scenario

The storage space allocated to the cloud customer was less than that requested by the cloud customer, which led the monitoring framework to signal the violation against the SLA (Table 8-3, Figure 8-7).

Table 8-3: Second Scenario

Storage allocated	Storage requested	Status
3.2 GB	3.5 GB	SLA violation

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	OutputFileSize(BYTES)	RESULT
1	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
2	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
3	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
4	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
5	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
6	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
7	SUCCESS	2	0	7359.88	0.1	7359.98	450000000	SLA violation
8	SUCCESS	2	0	7359.88	0.1	7359.98	600000000	SLA violation
9	SUCCESS	2	0	7359.88	0.1	7359.98	600000000	SLA violation
10	SUCCESS	2	0	7359.88	0.1	7359.98	600000000	SLA violation
11	SUCCESS	2	0	7359.88	0.1	7359.98	600000000	SLA violation

CloudSim Test finished!

Figure 8-7: The outcomes of monitoring of storage allocated to the cloud customer.

From the data presented in Table 8-3 and Figure 8-7, it was clear that the broker could not provide the storage space to the customer as per the request launched by the customer. The cloud customer requested the storage equal to 3-5, while the broker managed to provide the 3.2 GB of storage space. This was less than the cloud customer had requested, which was signalled as a breach of the SLA by the proposed monitoring framework.

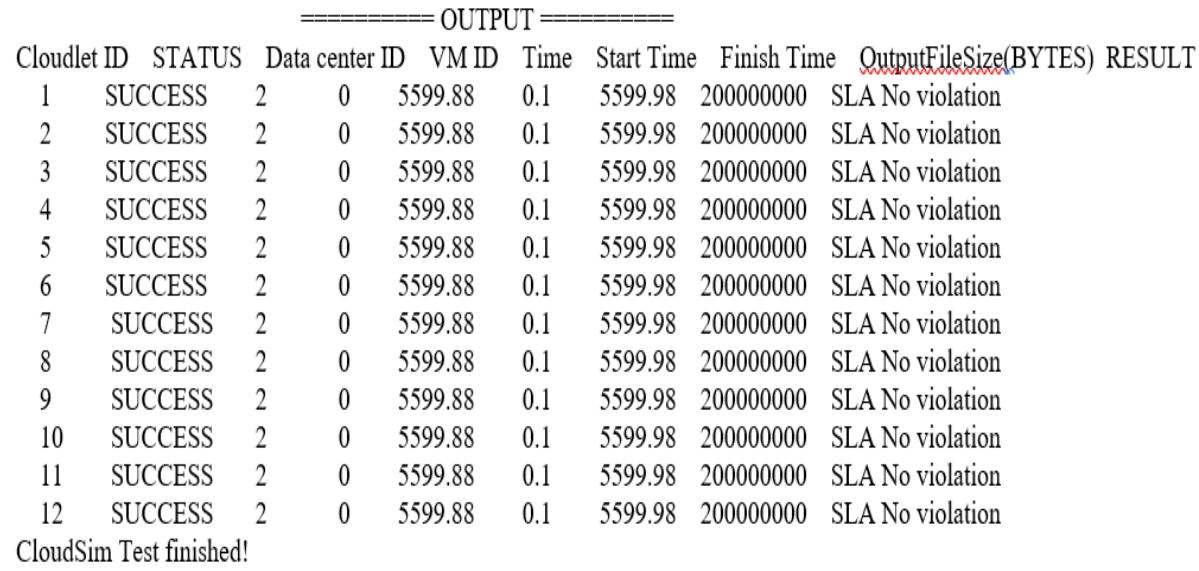
8.4.1.3. Third Scenario

The storage space provided by the cloud provider was equal to what was originally requested by the cloud customer. In this case, there is no SLA violation (Table 8-4, Figure 8-8).

Table 8-4: Second Scenario

Table 8-4: Third Scenario

Storage allocated	**Storage requested**	**Status**
2.9 GB	2.9 GB	No SLA violation



========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	OutputFileSize(BYTES)	RESULT
1	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
2	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
3	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
4	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
5	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
6	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
7	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
8	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
9	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
10	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
11	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation
12	SUCCESS	2	0	5599.88	0.1	5599.98	200000000	SLA No violation

CloudSim Test finished!

Figure 8-8: The data showing the results of monitoring of the stoarge allocation by the proposed monitoring framework.

In the Table 8-4 and Figure 8-8, data showed that the monitoring framework successfully detected the ‘No SLA violation’ events in the case of allocation of stoarge space on behalf of broker as per requested by the cloud customer. The customer requested the 2.9 GB stoarge space, and the broker provided the 2.9 GB storage space. Therefore, monitoring framework detecetd that there was no SLA violation.

8.4.2. Monitoring of Response Time

According to terms and conditions agreed upon in the SLA, the cloud provider is required to execute the tasks in 10,000 milliseconds. The threshold set for the execution of tasks was 10,100 milliseconds. If the response time is greater than or equal to 10,100 milliseconds, it will be considered as an "SLA violation". We monitored the execution time while transferring heavy files.

8.4.2.1. First scenario

The task executed by the cloud provider took less time than it was expected or requested by the cloud customer, which means that there was no SLA violation (Table 8-5, Figure 8-9).

Table 8-5: First Scenario

Response Time allocated	**Response Time requested**	**Status**
6000 milliseconds	6999 milliseconds	No SLA violation

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	NoOfCores	Time	Start Time	Finish Time	OutputFileSize(BYTES)	Result
0	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
1	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
2	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
3	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
4	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
5	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
6	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
7	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
8	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
9	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
10	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
11	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
12	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation

CloudSim Test finished!

Figure 8-9: Data showing the monitoring outcomes for response time.

In the Table 8-5 and Figure 8-9, data displayed that the monitoring framework was able to detect successfully the 'No SLA violation' events in the case of allocation of task execution time taken by the cloud provider as requested by the cloud customer. The customer requested that the task should be executed in 6,999 milliseconds, while the broker allocated the VM which was able to get the task executed within 6,000 milliseconds. Therefore, the monitoring framework detected that there was no SLA violation.

8.4.2.2. Second scenario

The task executed by the cloud provider was done in the time as requested or expected by the cloud customer, which means that there was no SLA violation (Table 8-6, Figure 8-10).

Table 8-6: Second Scenario

Response Time	**Response Time**	**Status**

allocated	requested	
5000 milliseconds	5000 milliseconds	No SLA violation

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	NoOfCores	Time	Start Time	Finish Time	OutputFileSize(BYTES)	Result
1	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
2	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
3	SUCCESS	2	0	1	9279.78	0.1	9279.88	200000000	SLA No violation
4	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
5	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
6	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
7	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
8	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
9	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
10	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation
11	SUCCESS	2	0	2	9279.88	0.1	9279.98	200000000	SLA No violation

CloudSim Test finished!

Figure 8-10: Data showing the monitoring outcomes for response time.

In the Table 8-6 and Figure 8-10, data exhibited that if execution time for the task is allocated in accordance with the requested time for execution on behalf of the cloud customers, then the monitoring framework detected it as the 'No SLA violation'. The cloud customer requested that task should be executed in 5,000 milliseconds, and the broker allocated the VM which executed the task within 5,000 milliseconds. Therefore, the monitoring framework detecetd that there was no SLA violation.

8.4.2.3. Third scenario

The task executed by the cloud provider was done in the time greater than what was requested or expected by the cloud customer, which means that there was an SLA violation (Table 8-7, Figure 8-11).

Table 8-7: Third Scenario

Response Time allocated	**Response Time requested**	**Status**
8000 milliseconds	7000 milliseconds	SLA violation

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	NoOfCores	Time	Start Time	Finish Time	OutputFileSize(BYTES)	Result
1	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
2	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
3	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
4	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
5	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
6	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
7	SUCCESS	2	0	2	13439.88	0.1	13439.98	450000000	SLA violation
8	SUCCESS	2	0	3	13439.99	0.1	13440.09	600000000	SLA violation
9	SUCCESS	2	0	3	13439.99	0.1	13440.09	600000000	SLA violation
10	SUCCESS	2	0	3	13439.99	0.1	13440.09	600000000	SLA violation
11	SUCCESS	2	0	3	13439.99	0.1	13440.09	600000000	SLA violation

CloudSim Test finished!

Figure 8-11: Data showing the monitoring outcomes for response time.

In the Table 8-7 and Figure 8-11, data exhibited that if execution time for the task is exceeded what is requested or expected by the cloud customers, then the monitoring framework detected it as 'SLA violation'. The cloud customer requested that the task should be executed in 7,000 milliseconds, while the broker allocated the VM which executed the task within 8,000 milliseconds. Therefore, the monitoring framework detected that there was SLA violation.

8.4.3. Monitoring of Virtual Machines

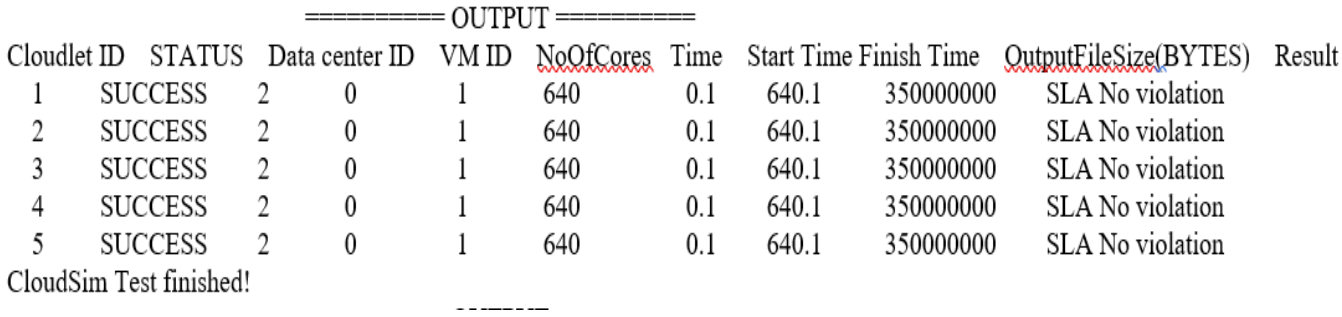
According to terms and conditions agreed upon in the SLA, the cloud provider is required to provide 14 virtual machines (VMs) execute the tasks in 10,000 milliseconds. The threshold set for the execution of tasks was 13 VMs. If the VMs are greater or equal to 14 milliseconds, it will be considered as a "No SLA violation".

8.4.3.1. First scenario

The VMs allocated to the cloudlet was greater than the storage requested by the cloudlet, which means that there was no SLA violation (Table 8.8, Figure 8.12).

Table 8-8: First Scenario

VMs allocated	VMs requested	Status
10GB	8 VM	No SLA violation



========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	NoOfCores	Time	Start Time	Finish Time	OutputFileSize(BYTES)	Result
1	SUCCESS	2	0	1	640	0.1	640.1	350000000	SLA No violation
2	SUCCESS	2	0	1	640	0.1	640.1	350000000	SLA No violation
3	SUCCESS	2	0	1	640	0.1	640.1	350000000	SLA No violation
4	SUCCESS	2	0	1	640	0.1	640.1	350000000	SLA No violation
5	SUCCESS	2	0	1	640	0.1	640.1	350000000	SLA No violation

CloudSim Test finished!

Figure 8-12: Data showing the results from monitoring of VM allocation to the cloud customer.

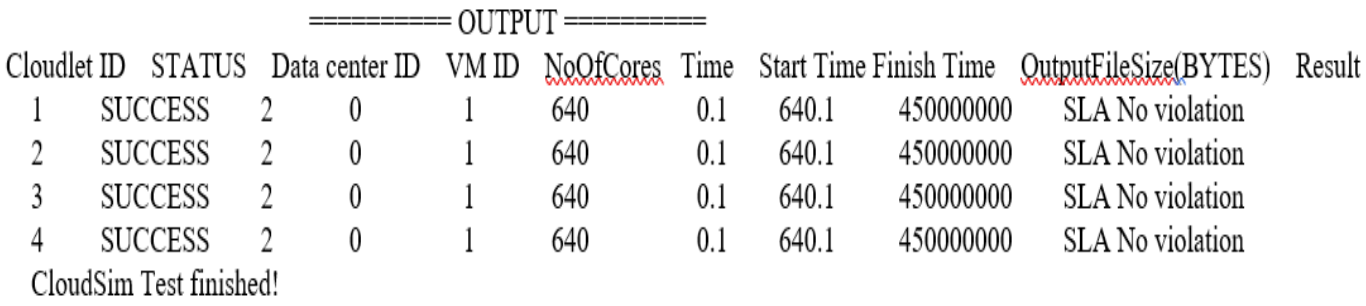
As shown in Table 8-8 and Figure 8-12, the monitoring mechanism detected that broker has allocated 10 VMs, while the cloud customer requested 8 VMs. Therefore, the monitoring framework detected that there was no violation against the SLA.

8.4.3.2. Second scenario

The VMs allocated to the cloudlet were equal to what was requested by the cloudlet, which means that there was no SLA violation (Table 8-9, Figure 8-13).

Table 8-9: Second Scenario

VMs allocated	VMs requested	Status
12GB	12 VM	No SLA violation



========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	NoOfCores	Time	Start Time	Finish Time	OutputFileSize(BYTES)	Result
1	SUCCESS	2	0	1	640	0.1	640.1	450000000	SLA No violation
2	SUCCESS	2	0	1	640	0.1	640.1	450000000	SLA No violation
3	SUCCESS	2	0	1	640	0.1	640.1	450000000	SLA No violation
4	SUCCESS	2	0	1	640	0.1	640.1	450000000	SLA No violation

CloudSim Test finished!

Figure 8-13: Data showing the results from monitoring of VM allocation to the cloud customer.

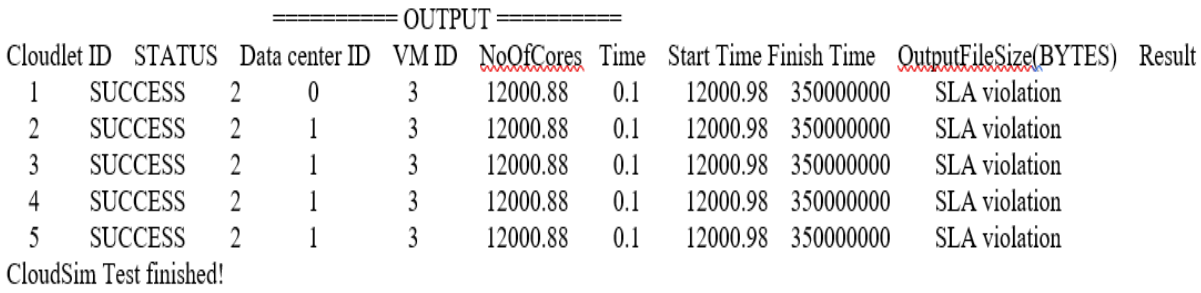
As shown in Table 8-9 and Figure 8-13, the monitoring mechanism detected that the broker has allocated 12 VMs which were equal to 12 VMs requested by the cloud customer. Therefore, the monitoring framework detected that there was no violation against the SLA.

8.4.3.3. Third scenario

The VMs allocated to the cloudlet were equal to what was requested by the cloudlet, which means that there was no SLA violation (Table 8.10, Figure 8.14).

Table 8-10: Third Scenario

VMs allocated	VMs requested	Status
9 VMs	11 VMs	SLA violation



========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	NoOfCores	Time	Start Time	Finish Time	OutputFileSize(BYTES)	Result
1	SUCCESS	2	0	3	12000.88	0.1	12000.98	350000000	SLA violation
2	SUCCESS	2	1	3	12000.88	0.1	12000.98	350000000	SLA violation
3	SUCCESS	2	1	3	12000.88	0.1	12000.98	350000000	SLA violation
4	SUCCESS	2	1	3	12000.88	0.1	12000.98	350000000	SLA violation
5	SUCCESS	2	1	3	12000.88	0.1	12000.98	350000000	SLA violation

CloudSim Test finished!

Figure 8-14: Data showing the results from monitoring of VM allocation to the cloud customer.

As shown in Table 8-10 and Figure 8-14, the monitoring mechanism detected that the broker has allocated 9 VMs, while the cloud customer requested 11 VMs. Therefore, the monitoring framework detected that there was violation against the SLA.

8.5. Conclusion

This chapter has presented the proposed monitoring framework, and evaluated using three scenarios including VM, storage and response time. The proposed monitoring framework was able to successfully detect the SLA violations, and sent the alerts to the cloud provider on detection of violations. If violations of SLAs are stopped and preventive measures are taken on behalf of the cloud provider, the SLA will continue, otherwise, the cloud customer terminates the SLA. The proposed monitoring framework can help improve the trust between the cloud customers and cloud providers. In addition, the monitoring mechanism developed

and tested in this chapter can improve the quality of service as well, which is important for large-scale proliferation of cloud services in the organizations.

CHAPTER 9: CONCLUSION

This chapter presents the conclusions drawn from this study. The first section of this chapter gives the motivations which triggered this research work, while the second section describes the contributions of this study to the knowledge and existing literature. The summary of the thesis is given in section three; the limitations of this study are provided in section four. The directions for future research are presented in the section five of this chapter.

9.1. Motivation for this research work

This research work was conducted in response to the need to solve the negotiation and monitoring issues faced by the cloud providers and customers. The ever-increasing base and diversity of cloud services in the market triggered the complex relationships between the service providers and cloud customers, which demanded the research into providing the simplest and easiest-to-use solutions for surmounting the issues of negotiating the services, and facilitating the negotiations which can be beneficial to both parties. Hence, it was very important to conduct the research into exploring and optimizing the negotiations options available in the market

Previously most of the research in the field of negotiation and monitoring of SLA were focussed on solving the complexities of web-service provision to the customers. However, cloud computing services were comparatively new and emerging concepts which required the consideration of different service dynamics, for example, remotely controlled services, security issues, and quality issues and compliance with metrics in SLA are the unique problems encountered by customers in the cloud market.

The afore-mentioned issues need the attention of researchers to provide the negotiation and monitoring solutions to the customers and cloud providers, which will further help proliferate the cloud services on large scale across different businesses. The need to develop and validate the negotiation solutions by building an effective and efficient SLA framework containing the monitoring element motivated this research work to be carried out.

The cloud computing' market harbours a variety of cloud services and cloud customers with different objectives, and carrying an independent entity with varying quality of service requirements. Against this background, there is a need to develop autonomous and efficient negotiation and monitoring mechanisms incorporated into the SLA framework. The intelligent SLA negotiation in the cloud computing domain may offer the viable solutions to create consensus between the service level parameters and quality of service requirements by placing stringent quality checks through the monitoring mechanism. Therefore, the necessity for developing and testing the autonomous SLA framework with a monitoring component for enhancement of trust and durability of the SLA for flow of cloud services from the cloud providers to cloud customers was another motivation behind the conducting of this research work.

9.2. Contribution to the knowledge

The data presented in this thesis contributed to both cloud computing literature and the domain relating to exploring the intelligent solutions for negotiating the beneficial SLA agreements in the business-oriented environment.

In this thesis, a novel autonomous SLA framework was presented, which involved the applications of intelligent agents in order to perform negotiation and the state-of-the-art

platform for monitoring the terms and conditions specified and agreed in the SLA drafted following the negotiation (Alqarni et al., 2019).

Humans may take plenty of time in order to negotiate over the complex matters, while the intelligent agents representing the relevant parties within the negotiation framework can offer the viable negotiation solutions within the seconds. Hence the autonomous SLA framework including the agents-based negotiation mechanism is an addition to the existing efforts leading to proliferation of cloud computing services across different organizational environments.

The autonomous SLA framework proposed in this study carries a great deal of flexibility in terms of improving the performance of negotiating agents, adding new agents for testing their negotiation performance, and even allowing mixing and matching the negotiation strategies of different agents to produce novel agents within the boundaries of the proposed SLA framework in this study.

The findings from this study also contribute to the application of varying negotiation strategies which can be used to boost the proliferation of cloud services, especially in the area of SaaS, in the cloud market based on the supply-and-demand mechanism.

This study contributed to the development of a novel negotiation agent based on mixing and matching negotiation strategies employed by other intelligent agents, which fulfilled the purpose of creating and optimizing the negotiation outcomes for the client to yield a beneficial negotiation outcome serving the trade objectives of all clients involved in the negotiation competition.

The negotiation performance of the proposed intelligent agent was tested in the real time cloud market environment, so that its application in the real cloud market with fluctuating requirements of customers can be promoted. This is in contrast to previous agents designed

which were tested using the mock data, and their performance in the real cloud market carried a question mark.

The negotiation strategy of the proposed intelligent agent was tested with a different number of domain sizes, and it satisfied the client requirements to win the negotiation with favourable outcomes. Most importantly, the evolving nature of the cloud market triggers the changes in the domains of services required or discarded by the cloud consumers. Hence, the strategy developed for the proposed intelligent agent suits the requirements of the consumers, and can be employed to negotiate over the issues carrying different weight for the consumers in the market.

Another contribution of this study is to develop and test the monitoring mechanism as part of the proposed SLA framework using the CloudSim. The proposed monitoring framework successfully detected the violations against or compliance with the terms and conditions specified and agreed between the consumer and cloud provider within the SLA document. The proposed monitoring mechanism is an addition to the existing literature focusing on increasing the quality of the cloud services, durability of the SLA and trust between the consumers and cloud providers.

9.3. Limitations of the study

This study, like any other study, has some limitations which should be kept in mind while interpreting and applying the outcomes of this study to improve the negotiation and monitoring of the cloud services offered by the cloud providers to the cloud customers in the cloud market. This study only benefited from the data from Azure and Amazon, and not from other cloud providers in the market. The other cloud providers may have different cloud

services and offerings, which might affect the applicability of outcomes to other cloud providers in the market.

The impact of fluctuations in demand-and-supply, changing trust levels between customers and cloud providers, emerging and novel cloud offerings and variations in cloud services across various regions is some of factors which may affect the negotiation outcomes between the customers and cloud providers. This study did not take into account the afore-mentioned factors while experimenting and interpreting results. Therefore, the negotiation strategies employed during this study should be optimized while incorporating the foregoing factors into the negotiation framework consisting of intelligent agents.

9.4. Future research directions

The future studies should apply the proposed SLA framework to other cloud providers in the cloud market. The different offerings and cloud products may be included as issues in development of domains to be tested through the negotiation strategies presented in this study.

The proposed negotiation framework involving the use of the state-of-the-art negotiation agents can come in handy in cloud robotics which is an emerging field, as cloud robotic is described to be an effective tool in reducing the computational loads while executing the tasks in the cloud environment. The future research may test the performance of the proposed negotiation strategy to negotiate over the services provided under the robotics-as-a-service (RaaS). The future studies can consider the generalizability of the proposed strategy for evaluation of low-conflict domains for the services less vulnerable to risks. In addition, some more metrics such as altruism, selfishness to evaluate the fairness of the proposed AR can be included in the future research work.

Also, in the future there is a very important point we can work in. The ability to collect data automatically, it will be a qualitative contribution because it will save a lot of time and effort, and will reduce the possibility of errors in the data collection process by a very large percentage.

Also, a model for calculating and evaluating the weights of issues in the offers. Because in its current situation calculated by estimating the importance of each issue for the customer and the service provider. If there is a model concerned with calculating weights, this will increase the accuracy of offers and increase the utility for the providers and customers.

The future studies should apply the inclusion of penalties because they were not covered in this research and we only checked whether there is a violation of the agreements in providing the service to the customer or not. However, it is possible to work on agreeing on the types of potential violations, specifying a time to fix this problem, and specifying the type of penalties in advance within the agreements. And then monitoring the service and providing it to the customer with the same agreed-upon, and then monitoring the violations, determining their type, and taking the appropriate action against them.

REFERENCES

Alexander, N., Howieson, J., & Fox, K. H. (2015). Negotiation: Strategy style skills.

Alhamad, M., Dillon, T., & Chang, E. (2010, April). Conceptual SLA framework for cloud computing. In *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on* (pp. 606-610). IEEE.

Aljoumah, E., Al-Mousawi, F., Ahmad, I., Al-Shammri, M., & Al-Jady, Z. (2015). SLA in cloud computing architectures: A comprehensive study. *International Journal of Grid and Distributed Computing*, *8*(5), 7-32.

Alsrheed, F., El Rhalibi, A., Randles, M., & Merabti, M. (2014, April). Intelligent agents for automated cloud computing negotiation. In *2014 international conference on Multimedia Computing and Systems (ICMCS)* (pp. 1169-1174). IEEE.

An, B., & Lesser, V. (2012). Yushu: a heuristic-based agent for automated negotiating competition. In *New Trends in Agent-Based Complex Automated Negotiations* (pp. 145-149). Springer, Berlin, Heidelberg.

An, B., Lesser, V. R., Irwin, D. E., & Zink, M. (2010, May). Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *AAMAS* (Vol. 10, pp. 981-988).

Andrieux, A., & Czajkowski, K. (2004). A. t. Dan, Web Services Agreement Specification (WS-Agreement). In *World-Wide-Web Consortium (W3C), http://www. w3. org/XML, http://www. gridforum. org/Meetings/GGF11/Documents/draft-ggfgraap-agreement. pdf*.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., ... & Xu, M. (2007, March). Web services agreement specification (WS-Agreement). In *Open grid forum* (Vol. 128, No. 1, p. 216).

Anithakumari, S., & Chandrasekaran, K. (2015, September). Monitoring and management of service level agreements in cloud computing. In *2015 International Conference on Cloud and Autonomic Computing* (pp. 204-207). IEEE.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50-58.

Arora, P., Wadhawan, R. C., & Ahuja, E. S. P. (2012). Cloud computing security issues in infrastructure as a service. *International journal of advanced research in computer science and software engineering*, *2*(1).

Automated Negotiating Agents Competition (ANAC), 2010 [online] Available at:<http://mmi.tudelft.nl/negotiation/index.php/Automated_Negotiating_Agents_Competition_(ANAC> [Accessed 2 June 2017].

Baarslag, T., Hendrikx, M. J., Hindriks, K. V., & Jonker, C. M. (2016). Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, *30*(5), 849-898.

Baarslag, T., Hendrikx, M. J., Hindriks, K. V., & Jonker, C. M. (2016). Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, *30*(5), 849-898.

Baarslag, T., Hendrikx, M., Hindriks, K., & Jonker, C. (2013, November). Predicting the performance of opponent models in automated negotiation. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)* (Vol. 2, pp. 59-66). IEEE.

Baarslag, T., Hindriks, K. V., Jonker, C. M., Kraus, S., & Lin, R. (2012). The First Automated Negotiating Agents Competition (ANAC 2010). *New Trends in agent-based complex automated negotiations*, *383*, 113-135.

Baarslag, T., Hindriks, K., & Jonker, C. (2013). Acceptance conditions in automated negotiation. In *Complex Automated Negotiations: Theories, Models, and Software Competitions* (pp. 95-111). Springer, Berlin, Heidelberg.

Baarslag, T., Hindriks, K., Hendrikx, M., Dirkzwager, A., & Jonker, C. (2014). Decoupling negotiating agents to explore the space of negotiation strategies. In *Novel Insights in Agent-based Complex Automated Negotiation* (pp. 61-83). Springer Japan.

Baarslag, T., Hindriks, K., Jonker, C., Kraus, S., & Lin, R. (2012). The first automated negotiating agents competition (ANAC 2010). In *New Trends in agent-based complex automated negotiations* (pp. 113-135). Springer, Berlin, Heidelberg.

Baarslag, T., Pasman, W, Hindriks, K, Tykhonov, D, Visser, W, Hendrikx, M., Feirstein, D. (2017). Negotiation User Guide.

Badidi, E. (2013, June). A cloud service broker for SLA-based SaaS provisioning. In *International Conference on Information Society (i-Society 2013)* (pp. 61-66). IEEE.

Bhardwaj, S., Jain, L., & Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, *2*(1), 60-63.

Bianco, P., Lewis, G. A., & Merson, P. (2008). *Service level agreements in service-oriented architecture environments* (No. CMU/SEI-2008-TN-021). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

Binmore, K., & Vulkan, N. (1999). Applying game theory to automated negotiation. *Netnomics*, *1*(1), 1-9.

Binu, V., & Gangadhar, N. D. (2014, October). A cloud computing service level agreement framework with negotiation and secure monitoring. In *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)* (pp. 1-8). IEEE.

Blythe, J., Deelman, E., & Gil, Y. (2004). Automatically Composed Workflows for Grid Environments. IEEE Intelligent Systems, (pp. 16-23).

Brams, S. J. (2003). *Negotiation Games: Applying game theory to bargaining and arbitration* (Vol. 2). Psychology Press.

Brams, S. J. (2003). *Negotiation Games: Applying game theory to bargaining and arbitration* (Vol. 2). Psychology Press.

Buyya, R., Garg, S. K., & Calheiros, R. N. (2011, December). SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *Cloud and Service Computing (CSC), 2011 International Conference on* (pp. 1-10). IEEE.

Buyya, R., Yeo, C. S., & Venugopal, S. (2008, September). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on* (pp. 5-13). Ieee.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems, 25*(6), 599-616.

Cai, H., Wang, N., & Zhou, M. J. (2010, July). A transparent approach of enabling SaaS multi-tenancy in the cloud. In *Services (services-1), 2010 6th world congress on* (pp. 40-47). IEEE.

Chen, E., Kersten, G. E., & Vahidov, R. (2004). An e-marketplace for agent-supported commerce negotiations. In *5th World Congress on the Management of eBusiness*.

Chen, S., Hao, J., Weiss, G., Tuyls, K., & Leung, H. F. (2014, September). Evaluating practical automated negotiation based on spatial evolutionary game theory. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)* (pp. 147-158). Springer, Cham.

Chhetri, M. B., Mueller, I., Goh, S. K., & Kowalczyk, R. (2007, November). ASAPM-an agent-based framework for adaptive management of composite service lifecycle. In *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops* (pp. 444-448). IEEE.

Dastjerdi, A. V., Tabatabaei, S. G. H., & Buyya, R. (2012). A dependency-aware ontology-based approach for deploying service level agreement monitoring services in Cloud. *Software: Practice and Experience*, *42*(4), 501-518.

De Donno, M., Tange, K., & Dragoni, N. (2019). Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *Ieee Access*, *7*, 150936-150948.

Durkee, D. (2010). Why cloud computing will never be free. *Queue*, *8*(4), 20.

El-Awadi, R., & Abu-Rizka, M. (2015). A framework for negotiating service level agreement of cloud-based services. *Procedia Computer Science*, *65*, 940-949.

Emeakaroha, V. C., Netto, M. A., Calheiros, R. N., Brandic, I., Buyya, R., & De Rose, C. A. (2012). Towards autonomic detection of SLA violations in Cloud infrastructures. *Future Generation Computer Systems*, *28*(7), 1017-1029.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2004). An agenda-based framework for multi-issue negotiation. *Artificial Intelligence*, *152*(1), 1-45.

Frieder, A., & Miller, G. (2013). Value model agent: A novel preference profiler for negotiation with agents. In *Complex Automated Negotiations: Theories, Models, and Software Competitions* (pp. 199-203). Springer, Berlin, Heidelberg.

Fujita, K., Ito, T., Baarslag, T., Hindriks, K., Jonker, C., Kraus, S., & Lin, R. (2013). The second automated negotiating agents competition (ANAC2011). In *Complex Automated Negotiations: Theories, Models, and Software Competitions* (pp. 183-197). Springer Berlin Heidelberg.

Fujita, K., Ito, T., Zhang, M., & Robu, V. (Eds.). (2015). *Next Frontier in Agent-based Complex Automated Negotiation* (Vol. 596). Springer.

Fullam, Karen K., Tomas B. Klos, Guillaume Muller, Jordi Sabater-Mir, Zvi Topol, K. Suzanne Barber, Jeffrey Rosenschein, and Laurent Vercouter. "The Agent Reputation and Trust (ART) Testbed Architecture." In Proceedings of the 2005 conference on Artificial Intelligence Research and Development, pp. 389-396. IOS Press, 2005.

Furht, B. (2010). Cloud computing fundamentals. In *Handbook of cloud computing* (pp. 3-19). Springer US.

Gal, Y. A., & Pfeffer, A. (2007, January). Modeling reciprocal behavior in human bilateral negotiation. In *Proceedings of the national conference on artificial intelligence* (Vol. 22, No. 1, p. 815). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Gelfand, M. J., Fulmer, C. A., & Severance, L. (2011). The psychology of negotiation and mediation.

Hao, J., & Leung, H.-F. (2014). CUHK agent: An adaptive negotiation strategy for bilateral negotiations over multiple items. In I. Marsa-Maestre, M. A. Lopez-Carmona, T. Ito, M.

Zhang, Q. Bai, & K. Fujita (Eds.), Novel insights in agent-based complex automated negotiation. Studies in computational intelligence (Vol. 535, pp. 171–179). Japan: Springer

Hashmi, K., Alhosban, A., Malik, Z., Medjahed, B., & Benbernou, S. (2014). Automated negotiation among web services. In *Web Services Foundations* (pp. 451-482). Springer, New York, NY.

Hindriks, K., Jonker, C. M., Kraus, S., Lin, R., & Tykhonov, D. (2009, May). Genius: negotiation environment for heterogeneous agents. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (pp. 1397-1398). International Foundation for Autonomous Agents and Multiagent Systems.

Holloway, M., Schuller, D., & Steinmetz, R. (2015, December). Customized Cloud Service Quality: Approaching Pareto-Efficient Outcomes in Concurrent Multiple-Issue Negotiations. In *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on* (pp. 256-260). IEEE.

Ilany, L. (2015). The fourth automated negotiation competition. In *Next Frontier in Agent-Based Complex Automated Negotiation* (pp. 129-136). Springer, Tokyo.

Ilany, L., & Gal, Y. A. (2016). Algorithm selection in bilateral negotiation. *Autonomous Agents and Multi-Agent Systems*, *30*(4), 697-723.

Ito, T., Hattori, H., & Klein, M. (2007, January). Multi-issue Negotiation Protocol for Agents: Exploring Nonlinear Utility Spaces. In *IJCAI* (Vol. 7, pp. 1347-1352).

Jadeja, Y., & Modi, K. (2012, March). Cloud computing-concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on* (pp. 877-880). IEEE.

Jamakovic, A., Bohnert, T. M., & Karagiannis, G. (2013, May). Mobile cloud networking: mobile network, compute, and storage as one service on-demand. In The Future Internet Assembly (pp. 356-358). Springer, Berlin, Heidelberg.

Jazayeriy, H., Azmi-Murad, M., Sulaiman, M. N., & Udzir, N. I. (2011). A review on soft computing techniques in automated negotiation. *Scientific Research and Essays*, *6*(24), 5100-5106.

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M. J., & Sierra, C. (2001). Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, *10*(2), 199-215.

Joshi, K. P., & Pearce, C. (2015, March). Automating cloud service level agreements using semantic technologies. In *2015 IEEE International Conference on Cloud Engineering* (pp. 416-421). IEEE.

Joshi, K. P., Yesha, Y., & Finin, T. (2014). Automating cloud services life cycle through semantic technologies. *IEEE Transactions on Services Computing*, *7*(1), 109-122.

Kawaguchi, S., Fujita, K., & Ito, T. (2011, June). Compromising strategy based on estimated maximum utility for automated negotiation agents competition (ANAC-10). In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (pp. 501-510). Springer, Berlin, Heidelberg.

Keller, A., & Ludwig, H. (2003). The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, *11*(1), 57-81.

Keller, A., & Ludwig, H. (2003). The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, *11*(1), 57-81.

Keller, A., & Ludwig, H. (2003). The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, *11*(1), 57-81.

Kersten, G. E. (1998). Negotiation support systems and negotiating agents. *Modèles et Systèmes Multi-Agents pour la Gestion de l'Environement et des Territoires*, 307-316.

Kersten, G. E., & Lai, H. (2007). Negotiation support and e-negotiation systems: An overview. *Group Decision and Negotiation*, *16*(6), 553-586.

Kersten, G. E., & Lo, G. (2001). Negotiation support systems and software agents in e-business negotiations. *Sprott School of Business*, *1*, 1-9.

Kersten, G. E., & Lo, G. (2003). Aspire: An integrated negotiation support system and software agents for e-business negotiation. *International Journal of Internet and Enterprise Management*, *1*(3), 293-315.

Kersten, G. E., Strecker, S. E., & Law, K. P. (2004, January). Protocols for electronic negotiation systems: Theoretical foundations and design issues. In *EC-Web* (Vol. 3182, pp. 106-115).

Koeman, V. J., Boon, K., van den Oever, J. Z., Dumitru-Guzu, M., & Stanculescu, L. C. (2015). The Fawkes Agent—the ANAC 2013 Negotiation Contest Winner. In *Next Frontier in Agent-Based Complex Automated Negotiation* (pp. 143-151). Springer, Tokyo.

Lai, G., Sycara, K., & Li, C. (2008). A decentralized model for automated multi-attribute negotiations with incomplete information and general utility functions. *Multiagent and Grid Systems*, *4*(1), 45-65.

Lamanna, D. D., Skene, J., & Emmerich, W. (2003, January). Slang: A language for defining service level agreements. In *NINTH IEEE WORKSHOP ON FUTURE TRENDS OF*

DISTRIBUTED COMPUTING SYSTEMS, PROCEEDINGS (pp. 100-106). IEEE COMPUTER SOC.

Lim, J. (2003). A conceptual framework on the adoption of negotiation support systems. *Information and software technology*, *45*(8), 469-477.

Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K., & Jonker, C. M. (2014). Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, *30*(1), 48-70.

Linlin, W., Kumar, G. S., Chao, C., & Steven, V. Automated SLA negotiation framework for cloud computing. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST cloud computing reference architecture. *NIST special publication*, *500*(2011), 292.

Long, W., Yuqing, L., & Qingxin, X. (2013, December). Using cloudsim to model and simulate cloud computing environment. In Computational Intelligence and Security (CIS), 2013 9th International Conference on (pp. 323-328). IEEE.

Luo, J. Z., Jin, J. H., Song, A. B., & Dong, F. (2011). Cloud computing: architecture and key technologies. *Journal of China Institute of Communications*, *32*(7), 3-21.

Mach, W., & Schikuta, E. (2012, December). A generic negotiation and re-negotiation framework for consumer-provider contracting of web services. In *Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services* (pp. 348-351).

Maes, P., Guttman, R. H., & Moukas, A. G. (1999). Agents that buy and sell. *Communications of the ACM*, *42*(3), 81-ff.

Marsa-Maestre, I., Lopez-Carmona, M. A., Ito, T., Zhang, M., Bai, Q., & Fujita, K. (Eds.). (2014). *Novel insights in agent-based complex automated negotiation* (Vol. 535). Springer.

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision support systems*, *51*(1), 176-189.

Maurer, M., Emeakaroha, V. C., Brandic, I., & Altmann, J. (2012). Cost–benefit analysis of an SLA mapping approach for defining standardized Cloud computing goods. *Future Generation Computer Systems*, *28*(1), 39-47.

McNamara, J. M. (2013). Towards a richer evolutionary game theory. Journal of the Royal Society Interface, 10(88), 20130544.

Modi, C., Patel, D., Borisaniya, B., Patel, A., & Rajarajan, M. (2013). A survey on security issues and solutions at different layers of Cloud computing. *The Journal of Supercomputing*, *63*(2), 561-592.

Moghaddam, M., & Davis, J. G. (2014). Service selection in web service composition: A comparative review of existing approaches. In *Web Services Foundations* (pp. 321-346). Springer, New York, NY.

Nie, G., Xueni, E., & Chen, D. (2012). Research on service level agreement in cloud computing. In *Advances in Electric and Electronics* (pp. 39-43). Springer, Berlin, Heidelberg.

OBox, P. (2003). Family-Winner: integrating game theory and heuristics to provide negotiation support. In *Legal Knowledge and Information Systems: JURIX 2003: the Sixteenth Annual Conference* (p. 21). IOS Press.

Ograph, T, B., & Morgens, Y. R. (2008). Cloud computing. *Communications of the ACM*, *51*(7), 9-11.

Pallis, G. (2010). Cloud computing: the new frontier of internet computing. *IEEE internet computing*, *14*(5), 70-73.

Paschke, A., & Schnappinger-Gerull, E. (2006). A Categorization Scheme for SLA Metrics. *Service Oriented Electronic Commerce*, *80*(25-40), 14.

Patel, P., Ranabahu, A. H., & Sheth, A. P. (2009). Service level agreement in cloud computing.

Patel, P., Ranabahu, A. H., & Sheth, A. P. (2009). Service level agreement in cloud computing.

Patel, P., Ranabahu, A. H., & Sheth, A. P. (2009). Service level agreement in cloud computing.

Patidar, S., Rane, D., & Jain, P. (2012, January). A survey paper on cloud computing. In *Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on* (pp. 394-398). IEEE.

Qian, L., Luo, Z., Du, Y., & Guo, L. (2009, December). Cloud computing: An overview. In *IEEE International Conference on Cloud Computing* (pp. 626-631). Springer Berlin Heidelberg.

Quiggin, J. (2012). *Generalized expected utility theory: The rank-dependent model*. Springer Science & Business Media.

Reaidy, J., Massotte, P., & Diep, D. (2006). Comparison of negotiation protocols in dynamic agent-based manufacturing systems. *International journal of production economics*, *99*(1), 117-130.

Rimal, B. P., Choi, E., & Lumb, I. (2009). A taxonomy and survey of cloud computing systems. *INC, IMS and IDC*, 44-51.

Rinderle, S., & Benyoucef, M. (2005, November). Towards the automation of e-negotiation processes based on web services–a modeling approach. In *International Conference on Web Information Systems Engineering* (pp. 443-453). Springer, Berlin, Heidelberg.

Robu, V., Somefun, D. J. A., & La Poutré, J. A. (2005, July). Modeling complex multi-issue negotiations using utility graphs. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (pp. 280-287). ACM.

Rosenschein, J. S., & Zlotkin, G. (1994). *Rules of encounter: designing conventions for automated negotiation among computers*. MIT press.

Shi, W., & Dustdar, S. (2016). The promise of edge computing. *Computer*, *49*(5), 78-81.

Silaghi, G. C., Şerban, L. D., & Litan, C. M. (2010, August). A framework for building intelligent SLA negotiation strategies under time constraints. In *International Workshop on Grid Economics and Business Models* (pp. 48-61). Springer Berlin Heidelberg.

Sim, K. M. (2006). A survey of bargaining models for grid resource allocation. *ACM SIGecom Exchanges*, *5*(5), 22-32.

Sim, K. M. (2011). Agent-based cloud computing. *IEEE transactions on services computing*, *5*(4), 564-577.

Spring, J. (2011). Monitoring cloud computing by layer, part 1. *IEEE Security & Privacy*, *9*(2), 66-68.

Stanoevska-Slabeva, K., & Wozniak, T. (2010). Cloud basics–an introduction to cloud computing. *Grid and cloud computing*, 47-61.

Takabi, H., Joshi, J. B., & Ahn, G. J. (2010). Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, *8*(6), 24-31.

Torkashvan, M., & Haghighi, H. (2012, November). CSLAM: A framework for cloud service level agreement management based on WSLA. In *6th International Symposium on Telecommunications (IST)* (pp. 577-585). IEEE.

Tsai, W. T., Sun, X., & Balasooriya, J. (2010, April). Service-oriented cloud computing architecture. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on* (pp. 684-689). IEEE.

Venticinque, S., Aversa, R., Di Martino, B., Rak, M., & Petcu, D. (2010, August). A cloud agency for SLA negotiation and management. In *European Conference on Parallel Processing* (pp. 587-594). Springer, Berlin, Heidelberg.

Villegas, D., Bobroff, N., Rodero, I., Delgado, J., Liu, Y., Devarakonda, A., ... & Parashar, M. (2012). Cloud federation in a layered service model. *Journal of Computer and System Sciences*, *78*(5), 1330-1344.

Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W., & Stößer, J. (2009). Cloud computing–a classification, business models, and research directions. *Business & Information Systems Engineering*, *1*(5), 391-399.

Wieder, P., Butler, J. M., Theilmann, W., & Yahyapour, R. (Eds.). (2011). *Service level agreements for cloud computing*. Springer Science & Business Media.

Wilkes, J. (2008). Utility functions, prices, and negotiation. *Market Oriented Grid and Utility Computing. Wiley Series on Parallel and Distributed Computing*, 67-88.

Williams, C. R., Robu, V., Gerding, E. H., & Jennings, N. R. (2012). Iamhaggler: A negotiation agent for complex environments. In *New Trends in Agent-based Complex Automated Negotiations* (pp. 151-158). Springer Berlin Heidelberg.

Wu, L., & Buyya, R. (2012). Service level agreement (SLA) in utility computing systems. In *Performance and dependability in service computing: Concepts, techniques and research directions* (pp. 1-25). IGI Global.

Wu, L., & Buyya, R. (2012). Service level agreement (sla) in utility computing systems. *IGI Global*, *15*.

Wu, L., & Buyya, R. (2012). Service level agreement (sla) in utility computing systems. *IGI Global*, *15*.

Wu, L., Garg, S. K., Buyya, R., Chen, C., & Versteeg, S. (2013, May). Automated SLA negotiation framework for cloud computing. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing* (pp. 235-244). IEEE.

Wu, L., Garg, S. K., Buyya, R., Chen, C., & Versteeg, S. (2013, May). Automated SLA negotiation framework for cloud computing. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (pp. 235-244). IEEE.

Xu, X. (2012). From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing*, *28*(1), 75-86.

Xu, X. (2012). From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing*, *28*(1), 75-86.

Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., & Zhang, J. (2007). Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems*, *23*(6), 748-759.

Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., & Zhang, J. (2007). Autonomous service level agreement negotiation for service composition provision. *Future Generation Computer Systems*, *23*(6), 748-759.

Youseff, L., Butrico, M., & Da Silva, D. (2008, November). Toward a unified ontology of cloud computing. In *Grid Computing Environments Workshop, 2008. GCE'08* (pp. 1-10). IEEE.

Zheng, X., Martin, P., & Brohman, K. (2012, May). Cloud service negotiation: Concession vs. tradeoff approaches. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)* (pp. 515-522). IEEE.