Contribution to Agents-Based Negotiation and Monitoring of Cloud Computing Service Level Agreement

Faisal Salem Alsrheed

A thesis submitted in partial fulfilment of the requirements of

Liverpool John Moores University for the degree of

Doctor of Philosophy

June 2014

Abstract

Cloud Computing environments are dynamic and open systems, where cloud providers and consumers frequently join and leave the cloud marketplaces. Due to the increasing number of cloud consumers and providers, it is becoming almost impossible to facilitate face to face meetings to negotiate and establish a Service Level Agreement (SLA); thus automated negotiation is needed to establish SLAs between service providers and consumers with no previous knowledge of each other. In this thesis, I propose, an Automated Cloud Service Level Agreement framework (ACSLA). ACSLA is made up of five stages, and the corresponding software agent components: Gathering, Filtering, Negotiation, SLA Agreement and Monitoring. In the Gathering stage all the information about the providers and what they can offer is gathered. In the Filtering stage the customer's agent will send the request to ACSLA, which will filter all the providers in order to recommend the best matched candidates. In Negotiation stage the customer's agent will negotiate separately with each candidate provider using different negotiation algorithms, which will be evaluated and for which recommendations and guidelines will be provided. The output of this stage is that the best outcome from the customer's perspective will be picked up, which will be the agreed value for each parameter in the SLA. In SLA Agreement stage the provider's agent and the customer's agent will be informed about the Agreement, which will be specified in measurable terms. The output of the SLA Agreement stage will be a list of metrics that can be monitored in the Monitoring stage. Customer's agent and provider's agent will also negotiate and agree about the penalties and actions will be taken in case the SLA has been violated and unfulfilled. There is a variety of actions that can be taken, like informing both sides, recommending solutions, self-healing and hot-swapping. ACSLA is evaluated using case studies which show its flexibility and effectiveness. ACSAL offers a novel approach to tackle many challenging issues in the current and likely future, cloud computing market. It is the first complete automated framework for cloud SLA. There are many automated negotiation algorithms and protocols, which have been developed over the years in other research areas; establishing functional solutions applicable to the

cloud-computing environment is not an easy task. Rubinstein's Alternating Offers Protocol, also known as the Rubinstein bargaining model, has been investigated for application in automated cloud SLA, and it offers a satisfactory technical solution for this challenging problem. The purpose of this research was also to apply the state of the art in negotiation automated algorithms/agents within a described Cloud Computing SLA framework, to develop new algorithms, and to evaluate and recommend the most appropriate negotiation approach based on many criteria.

Contents

List of tables

Table of Figures

List of Acronyms

ANAC: Annual Automated Negotiating Agents Competition.

ART : The Agent Reputation and Trust Testbed.

BOA: Bidding Opponent Acceptance

CSV: comma-separated values

GENIUS: General Environment for Negotiation with Intelligent multi-purpose Usage Simulation.

IaaS: Infrastructure as a Service.

OVF: Open Virtualization Format

PaaS: Platform as a service

QoS : Quality of Service

SaaS: Software as a Service

SLA: Service Level Agreements

SLO: Service Level Objective

SOA: Service-oriented architecture.

UDDI : Universal Description, Discovery and Integration.

VM: Virtual Machines.

XML: Extensible Markup Language.

Acknowledgement

Completing this PhD would not have been possible without the help and support of great and special people. First, and most importantly, I wish to thank my parents and my brothers and sisters. 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. Randles, Martin for his help. I would like to thank Tim Baarslag from the Interactive Intelligence group at Delft University of Technology for his help.

I will be forever thankful to Ahmad Alateeq for helping me to get this PhD scholarship. I am most grateful to my brother Ziad Abdulbaqi for supporting me at every step of this journey.

I would like to thank Carlos Paz for being a true friend and patient flatmate. During the course of this PhD Carlos has been a great inspiration and hard work personified.

My friends Anthony Caton and Abdulhamed Allari have both been positive and motivating forces and have been here during the good times and the hard times. I am very thankful and would like to express my appreciation to Tiute for the assistance in my third year of my PhD. For my friends back home in Saudi Arabia, I would like to thank them all for keeping in touch and motivating me throughout the course of this process. Also, I must thank my friend Faris Almoslamani for his advice.

My bother-in-law Mohammed Almeziny has been supportive and a great advisor especially during my first years in the UK. I will never forget all the great people that helped to get this stage of my life and I promise to do my best to pay them back one day and somehow.

Dedication

I would like to dedicate this thesis to my parents;

سالم خلف السرهيد و منيرة عبدالله النويصر

List of publications

1- Faisal Alsrheed, Abdennour El Rhalibi, Martin Randles, Madjid Merabti (2013) ***Rubinstein's Alternating Offers Protocol for Automated Cloud Computing Negotiation***, The 14th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2013) Liverpool, United Kingdom.

2- Faisal Alsrheed, Abdennour El Rhalibi, Martin Randles, Madjid Merabti (2014), ***Intelligent Agents for Automated Cloud Computing Negotiation,*** the 4th IEEE International Conference on Multimedia Computing and Systems (ICMCS'14), Marrakesh, Morocco. Proceedings ISBN :978-1-4799-3824-7

3- Faisal Alsrheed, Abdennour El Rhalibi, Martin Randles, Madjid Merabti (2014), ***Automated Service Level Agreement Negotiation for Cloud Robotics***, 7th Saudi Students Scientific Conference-UK, University of Edinburgh ,Scotland. Proceedings ISBN :978-0-9569-0452-2

Chapter 1

Introduction

In this chapter I will present the motivations behind this research and the challenges that have been faced during the development of the proposed solutions. Moreover, I will discuss the research aims and objectives, and the research scope. Afterwards, contribution to knowledge will be given in details. Finally, in the last section I establish the structure of the thesis.

1.1 Introduction:

Over the years, the way computing is provided has changed from mainframe and terminal to PC, and then networked PC, leading to clients and servers and, lately to cloud computing via the Internet. The difference between the early stages of computing and the more recent cloud computing is that computational resources used to be delivered as product (off-the-shelf hardware or software). Now computing is delivered as a service in a more customized way. In the past, the customers needed to adapt to what was available in the market and look for the seller that offered a product that fitted most closely to his/her requirements. Also, in the past the relationship between the seller and buyer ended just after the purchasing deal was made. Now, in the cloud market, the seller has become a provider of computing as a utility

and the customer has become a client. Also, the relationship and the negotiation between them will continue until the contract ends. This contract is called the SLA (Service Level Agreement), which is the legal contract between the provider and the client, in order to ensure the Quality of Service (QoS). In cloud computing the provider and client will negotiate before signing the contract. Moreover, in some cases they might need to re-negotiate. Furthermore, an additional negotiation might be needed if the client requests more resources or if they need to negotiate the penalties, compensation or the termination of the contract.

Today there are only a relatively small number of cloud providers in the cloud computer market and all of them offer solely "off-the-shelf" Service Level Agreements ("take it or leave it"). Introducing customized SLAs to the cloud market would offer customers and providers, added benefits. Customized SLAs can only be established by negotiation. The negotiation needs to be automated to handle the dynamic and complex environment of cloud computing.

I believe that SLA management for Cloud computing needs to be automated, including all the stages the SLA life cycle. In this work I proposed an automated SLA framework, which consists of five stages: gathering, filtering, negotiation, agreement and monitoring. In the gathering stage the providers' offers and the client's requests will be gathered. In the filtering stage the candidate providers will be filtered based on the client's requests and preferences. In the negotiation stage the providers and the client will be represented by the intelligent agents, which will negotiate on their behalf. In the agreement stage the client will compare all the outcomes of the negotiation sessions and will choose the best provider to sign SLA agreement with.

Providers and clients should have the freedom to make their own agents, which will represent them and will perform their negotiation strategy. Developing these intelligent agents is not an easy task, because every negotiation sessions can be different than the other. Also, the agent needs to be able to decide when to make a cooperative or a selfish offer as well as to decide when to accept the opponent's offer, in the same time to keep track of the remaining time in the negotiation session, if relevant, and to decide when to end the negotiation without agreement.

In addition, since the negotiation that I are considering in this work is closed in the sense that the opponent's preferences and strategies are not shared, hence for that the agents are expected to try to learn about the opponent's strategy from his consecutive bids. This is called *opponent modelling*, which is very important in order to improve the quality of the negotiation outcome by helping the agent to reach a better early agreement. The *opponent modelling* will help the agents to avoid ending the negotiation without agreements by proposing bids potentially converging to the *Pareto optimal* solution (Williams Colin R, 2012), which will maximize the *social welfare* (i.e. higher utility for both agents), that is, provides the best trade-off for both provider and consumer (Williams Colin R, 2012).

An intelligent agent is expected to be self-interested. The Agents that I will use in this work are *Utility-based* agents, which mean they use a *utility function* to make rational decisions. *Utility functions* are a way of representing agent's preferences with the aim to achieve a goal. The ultimate goal of each agent is to maximize its utility. When two utility-based rational agents try to maximize their utility in the negotiation process, there often occurs a conflict. Here is where *Game theory* (Myerson, 1991) might become useful. *Game theory* may be used at this point to analyse interactions between conflicting (competing) agents. *Game theory* is a mathematical theory that studies interactions among self-interested agents, in which

Negotiation can be seen as a game, where two agents try to come to an agreement. Each agent is assumed to have a fixed preference over all possible deals (bids). Both agents face the same problem, meaning that they are both trying to maximize their utility function. They also face the risk of a break-down in the negotiation process or even an expiration of a deadline if they do not cooperate enough to reach an agreement.

The agents will follow a negotiation protocol. The negotiation protocol is needed to determine the overall order of actions during a negotiation. In this work a protocol known as - *Rubinstein's Alternating Offers Protocol* (Rubinstein, 1982).] or simply Rubinstein bargaining model will be used. To encourage the cloud computing provider and the client to let the agents to negotiate on their behalf, they need to be able to pick an agent from the from a set of available agents or to create their own agent, which would follow their negotiation strategy based on the *offer-and-demand* situation in the market, and the specific requirements and constraints of the provider and consumer.

The client and the provider will face a dilemma in choosing the negation strategy. A very competitive strategy might lead to ending many negotiations with no agreement. On the other hand, choosing a very cooperative strategy might lead to an agreement with a low utility. So in this work I introduced a novel agent Wise H-T (Alsrheed, et al., 2014a), which has a mix of cooperative and competitive strategies.

Not only do the strategies affect the outcome of the negotiation, but also choosing a negotiation deadline or duration is important, as choosing short time might lead many negotiation sessions to end with no agreement. On the other hand, long deadlines might lead to a waste of time, without improvement of the agreement.

Hence, in this work I will compare the performance of different strategies from the literature, compare to our proposed technique, and make a set of recommendations about what the

deadline should be set at, taking into the account the size of the *domain*, which is the number of possible solution agreements.

This work is the first step toward the next generation of cloud computing, where cloud services (SaaS, PaaS and IaaS) will be discovered, negotiated about and monitored via intelligent agents with no human interaction needed.

1.2 Motivations

As I mentioned above, one of our motivations is to be the first work toward the next generation of cloud computing, where cloud services (SaaS, PaaS and IaaS) will be discovered, negotiated about and monitored via intelligent agents with no human interaction needed.

Presently there are a limited number of providers, but the number of the cloud brokers is increasing. Most of the time, the cloud brokers use the same cloud provider. Thus a future of the cloud computing market is expected to be similar to the mature mobile phone market, where most of the providers use the same infrastructure: The brokers provide the service to the clients. I believe in the future it will be easier to move from one provider to another in seconds, using techniques such as “hot-providers-swap” (Petcu, et al., 2013).This is made possible, as Most of the cloud providers support the Open Virtual Machine Format (OVF), which is an open-source standard for packaging and distributing virtual machines images (VM) (Petcu, et al., 2013). This standard will help cloud clients to easily move (import and export) virtual machine images between cloud providers.

Our second motivation is that in 2010, at the beginning of this project, there was a lack of an actual practical work in this field of cloud computing. The reason behind this was because the cloud computing was in the very early stage of its existence. Most of the practical work

offered solutions which were “borrowed” from other related technologies, for instance Grid computing and web services.

To the best of our knowledge, a complete automated negotiation by Agents-SLA life cycle (specially made for cloud computing) did not exist. Since cloud computing is a new model of providing computer resource that is why there is a need to create a new SLA life cycle which fits the dynamic notion and heterogeneity of cloud computing, taking into the account every single step of SLA life cycle in details.

To the best of our knowledge, there is some works which emphasized the need for negotiation in the cloud computing. However, none of them provides a practical solution taking into account that negotiation needs to support multi-issue negotiations with the possibility of giving preferences for each issue. Moreover, the negotiation needs to be automated, and run by a rational agent with different negotiation strategies. I do not believe that every negotiation should always end with an agreement. In other words, agents shall not be forced to reach agreement. So, each side has the right to end the negotiation without agreement.

Another motivation is to be the first work to study the effect of the deadline to the outcome of the negotiation. Most of the work that have been done before, studied the negotiation process with deadlines at 180 seconds. It is important to study shorter deadlines, so I can provide very fast and short negotiation sessions, for instance 10 seconds. In this work I have investigated how the agents perform with a short deadline, such as 10 seconds.

Another motivation is to investigate the effect of the size of the domain to the outcome, as big domains are challenging for the agent to explore and it is important to know which agents are able to work with a large domain.

Also, the relationship between the size of the domain and the deadline needs to be investigated, as no other work has studied this previously.

1.3 Challenges

There are many challenges I have faced before starting and during the development of this work. At the beginning of this work cloud computing was in the early stage of its development, hence during the development of this work many research works and commercial products were proposed to the market by some cloud providers. It was a challenge for us to keep up with such a fast change. However, I found this interesting. This challenge allowed us to propose a solution, which will fit into the state-of-the-art of cloud computing technology and even to contribute to the future trends of cloud computing.

The second challenge that I have faced was the limited availability of related works, which described agents similar to ours for benchmarking purpose, how they work and which learning techniques they are using for the opponent modelling.

Moreover, I have faced some technical challenges. One of the technical challenges I have faced was in using GENIUS (Lin, et al., 2012), which is the evaluation environment I have used for our agents development and negotiations. The previous public version of GENIUS suffered many technical errors, which made it difficult to trust or properly understand the outcomes. Fortunately, after the first half of 2012, a new version of GENIUS (Lin, et al., 2012) was released, where all the technical errors were fixed.

1.4 Research Aim and Objectives

In this section the aim and objectives of this thesis will be highlighted.

1.4.1 The aim.

This work's aim is to investigate the fundamental requirements for a novel automated SLA management for cloud computing including all the stages of SLA life cycle.

1.4.2 The Objectives.

This work's objectives are to:

1. Study and analyse existing solutions for SLA management from existing paradigms, such as Grid computing and web service.

This objective will help us to answer the following questions:

- Can I use the SLA life-cycle of Grid computing and web service for cloud computing?
- What are the stages (components) of the SLA for cloud computing?

2. Propose and develop a complete SLA framework to support negotiation, including all the SLA life cycle.

This objective will help us to answer the following questions:

- What are the inputs and outputs of each stage?
- How can I automate all the stages?
- Which negotiation protocols should be used in the negotiation stage?
- Which action each agent is allowed to take?

3. Implement our agent Wise H-T.

   This objective will help us to answer the following questions:

   - How to develop a novel agent that performs better than the state-of-the-art agents in term of performance, ability to learn from the opponent's offers, and negotiation outcomes?
   - How to implement our agent in the way that it can be improved easily by other researcher in the future?
   - Should I use the learning technique for opponent modelling and which one?

4. Finding and using the State-of-the-art-Agent.

   This objective will help us to answer the following questions:

   - Where can I find the state-of-the-art agents?
   - How to access information about them and how to understand the learning technique that each agent uses?

5. Evaluating our agent against the State-of-the-art-Agent.

   This objective will help us to answer the following questions:

   - How to evaluate all the agents in the environment that can be replicated easily by other researchers as a future work?
   - Which evaluation criteria will be used to evaluate the agents?
   - Is using an agent for negotiation will benefit the customer and the provider, comparing to not negotiating at all?
   - Which one is better: using a competitive strategy or using a cooperative strategy?
   - What are the risk and the benefit of using the competitive strategy or a cooperative strategy?

- What are the outcomes when the opponent plays similar strategy or different strategy?
- When the customer or provider should use a competitive strategy or a cooperative strategy?
- Is it better to use a combined strategy made of a competitive strategy and cooperative strategy rather than two separate strategies on their own?
- How our proposed agent Wise H-T will perform against the state-of-the-art agents?
- Is there a relation between the performance and fairness among all the agents?
- Could the relation between performance and fairness be improved while increasing and decreasing the deadline of the negotiation session?
- Would the results be affected if the deadline is switched between the round based protocol to a time based protocol?
- Is there a relationship between the size of the negotiation domain and the deadline of the negotiation session?
- If there is a relationship between the size of the negotiation domain and the deadline of the negotiation session, then would this affect the outcome of the negotiation, the performance of the agents and which agent would be affected the most?
- If there is a relationship between the size of the negotiation domain and the deadline of the negotiation session, then would the results be

affected if the deadline is switched between the round-based protocols to a time-based protocol?

1.5 Research Scope.

In this work, I will cover up and link a variety of 5 research areas (see Figure 1); cloud computing, intelligent agents, game theory, negotiation and machine learning. In the next figure, I illustrate our work scope and how the research areas are linked. After that, I will discuss each of 5 research areas separately.

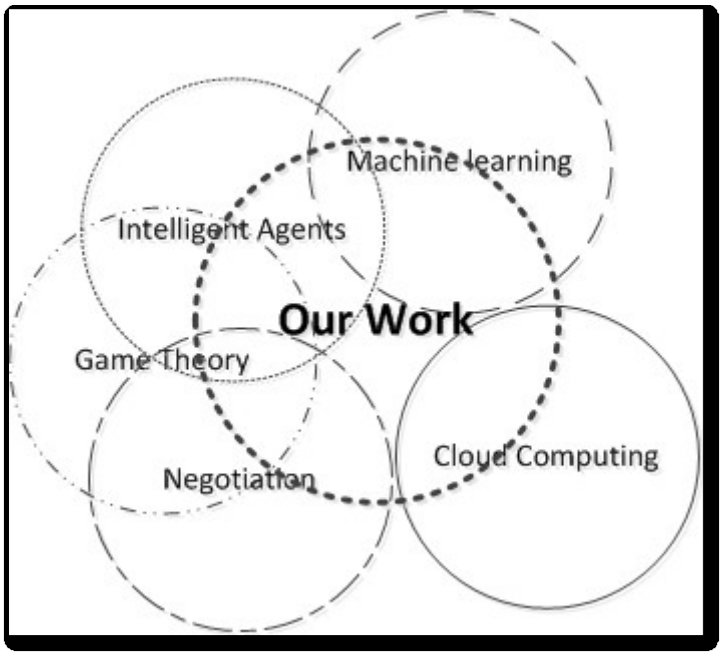


Figure 1 : Research Scope

Each circle in figure 1 represents a research area that we have coved in this work. However, the circle in the centre represents how this work linked the five research areas together.

1.5.1 Cloud computing.

Our framework will cover the SLA for all cloud computing layers, Infrastructure-as-a-Service (IaaS) also known as the Infrastructure services layer, Platform-as-a-Service (PaaS) also known as the Platform layer, Software-as-a-Service (SaaS) also known as the Application layer. This work focuses on the public cloud (Zhang, Cheng and Boutaba, 2010) which is one of cloud computing deployment models; this is because in the public cloud there is negotiation between provider and client.

1.5.2 Intelligent Agents.

In this work intelligent agents will be used to represent the provider and the client. The intelligent Agents, used in this work, are utility-based agents which mean they use the utility function. Utility functions are simply a way of representing an agent's preferences. The vital goal of each agent is to maximize its utility.

1.5.3 Game theory.

When two utility-based agents try to maximize their utility in the negotiation process, there often occurs a conflict. *Game theory* may be used at this point to analyse interactions between conflicting (competing) agents. *Nash equilibrium* (Nash, 1950) will be used to find a win-win agreement. From the game theory perspective, in this work I cover *Zero-Sum* game and *Win-win* game. The customer and the provider need to negotiate over the multiple *issues* with the possibility of giving preferences for each issue. More information about game theory will be explained later in this thesis.

1.5.4 Negotiation.

This research focuses on negotiation, which will only be between two agents at a time. The negotiation is closed in the sense that the strategies and preferences of each agent will be kept hidden and not revealed. Nonetheless agents can learn from the opponent offer using machine learning algorithms. In this work a Protocol called - Rubinstein's Alternating Offers Protocol (Rubinstein, 1982), also known as Rubinstein bargaining model, will be used. More information about Rubinstein's Alternating Offers Protocol will be introduced later in this thesis.

1.5.5 Machine Learning.

Agents are expected to try to learn about the opponent from his bid. This is called *opponent modelling*, which is important in order to improve the quality of the negotiation outcome. The Learning Methods that will be used for opponent modelling are Bayesian Learning, Non-linear Regression, Kernel Density Estimation, and Artificial Neural Networks (Dirkzwager, 2013) and (Dirkzwager, et al., 2012). Table 1 shows how and which learning methods are used for the opponent modelling of the agents that have been used to evaluate my proposed agent. Also, table 1 shows the Opposite Model which is the opponent model that is used in my proposed agent. In this work, a theoretical baseline opponent modeling technique is used.

Table 1: Opponent models state of the art (Baarslag, et al., 2013b)

Bayesian Model	
Bayesian Scalable Model	Estimates the issue and value weights separately using Bayesian learning. The opponent is assumed to concede a constant amount per round.
IAMhaggler Bayesain Model	A Bayesian Model in which the opponent is assumed to use a particular time-dependent strategy and only unique bids are used to update the model
Frequency Models	
HardHeaded Frequency Models	Learns the issue weights based on how often the value of an issue changes. The value weights are estimated based on the frequency they are offered.
Smith Frequency Model	Learn the value weights based on frequency they are offered. The issue weights are estimated based on the distribution of the values.
Agent X Frequency Model	A variant of the HardHeaded Frequency Model that takes the opponent's tendency to repeat bids into account.
N.A.S.H. Frequency Model	Learns the issue weights based on how often the best value for each issue is offered. The value weights are estimated based on their frequency.
Value Models	
Agent LG Value Model	Estimates the value weights based on the frequency they are offered.
CUHKAgent Value Model	Counts how often each value is offered. The utility of a bid is the sum of the score of its values divided by the best possible score. The model only uses the 1st 100 unique bids for its estimation.
Theoretical Baselines	
Opposite Model	Defines the opponent's utility as one minus the agent's utility

1.6 Research methodology.

1.6.1 Reviewing

The first step, to understand the research problems and to also find out what other researchers have proposed to resolve these problems, is a review of the state-of-the-art in cloud computing, SLA, automated negotiation, multi-agent system and game theory.

1.6.2 Analysing the requirements and designing the Framework.

In this step I identify all the stages of framework. Then, I identify all the users / agents requirements for each stage. I also identified all the inputs and outputs for each stage. After that, I will design all the stages of the framework including the detailed and high level design.

1.6.3 Implementing a Prototype.

In this step, I implemented the stages of the framework. I implemented the gathering and filtering stages by using Apache web server, PHP and MySQL. For the negotiation stage, I used GENIUS (General Environment for Negotiation with Intelligent multi-purpose Usage Simulation) (Lin, et al., 2012) and BOA Bidding Opponent Acceptance framework (Dirkzwager, et al., 2012), to create a novel negotiation rational agent dubbed Wise-H-T.

1.6.4 Evaluation

I evaluated our work against the state-of-the-art work, for example, the Agent that I have implemented (Wise-H-T) has been evaluated against the state-of-the-art automated negotiation agents by using GENIUS (Lin, et al., 2012).

1.7 Contribution to Knowledge

This work contributes to knowledge in many different ways. This research presents the contribution to cloud computing field and to the automated negotiation field:

1.7.1 Cloud computing field:

- This work proposes a novel framework for automated SLA life cycle management for cloud computing by using intelligent agents to perform the automated negotiation stage instead of humans (Alsrheed, et al., 2013).
- Making recommendations about the use of the negotiation strategy as well as when and how each strategy should be used in the cloud computing, taking into the account the supply and demand factors. (Alsrheed, et al., 2013).
- This work, the vision and the principles of autonomic computing by using autonomic control loop (Collect, Analyze, Decide and Act) are used to design and implement our solution for monitoring stage. I presented the implementation of our solution via a scenario of over-promising and under-delivering problem; I also demonstrated the idea of live virtual machine migration using cloud computing simulation.

1.7.2 Automated negotiation field

- This work proposes a novel agent, named Wise H-T, which combines and balances between cooperative and competitive behaviours, which lead to better negotiation outcomes from other agents (Alsrheed, et al., 2014a).
- Investigating the effect of increasing and decreasing the size of the domain (number of possible agreements) to the negotiation outcome. (Alsrheed, et al., 2014a).

- All the related works restrict their considerations to time–based protocols where the deadline is 180 seconds. I believe that 180 seconds is, to some extent, too long a time. Our vision for the next generation of cloud computing negotiations is one based on short timescales, so that the providers and customers can quickly negotiate with multiple-opponents. Thus, in this work, I investigated the effect of increasing and decreasing the deadline for the negotiation to the outcome. (Alsrheed, et al., 2014a) Investigating the effect of switching between the rounds-based protocol and time based deadlines protocol of the negotiation to the outcome. (Alsrheed, et al., 2014a)

1.8 Thesis Structure

In chapter 1, I covered the motivations behind this work and the challenges that I faced during the development of the proposed solutions. Moreover, the research aim and objectives were discussed as well as the research scope. Afterwards, contribution to knowledge has been given in details. Finally, the structure of the thesis has been highlighted. **In chapter 2**, I will give a detailed background review about cloud computing, which will include the definition of cloud computing, the cloud computing brief history as well as the related technologies and cloud computing layers. Afterwards, another three very important topics will be explained in details, which are related to service level of agreement (SLA), negotiation and game theory. Chapter 2 also will answer the questions; what is an agent, what are the capabilities of an intelligent agent. Then, the agent rationality, preferences and utility function will be explained. **In chapter 3,** the related works will be discussed. After that, State-of-the-art Negotiation Agents will be identified. **In chapter 4,** the requirements to achieve the negotiable SLA will be discussed. **In chapter 5,** a high level design of the framework architecture as well as a detailed design for gathering, filtering, negotiation and SLA agreement stages will be presented. Also, the overall framework closed loop will be

presented. **Chapter** 6 presents the implementation of our monitoring stage solution via a scenario of over-promising and under-delivering problem. Finally, I will demonstrate the idea of virtual machine migration using cloud computing simulation known as CloudSim [Calheiros., et al, 2011]. **Chapter 7** shows the implementation of gathering, filtering, negotiation and SLA agreement stages. **Chapter 8** covers the four experiments: Google and Amazon (Hardliner) experiment, The Negotiation deadlines experiment and the negotiation domain size experiment and price negotiation experiment. **In Chapter 9,** I conclude this thesis, and present our ideas for future work.

1.9 Conclusion

This chapter covered the motivations behind this work and the challenges that I faced during the development of this work. The main motivation was to propose the first work toward the next generation of cloud computing, where cloud services (SaaS, PaaS and IaaS) will be discovered, negotiated about and monitored via intelligent agents with no human interaction needed. This chapter highlighted the challenges that I have faced before starting and during the development of this work. The challenges include; limited availability of related works and some technical challenges with the evaluation environment. Moreover, the research aim and objectives were discussed as well as the research scope. This work's aim is to investigate the fundamental requirements for a novel automated SLA management for Cloud Computing including all the stages of SLA life cycle. This work's objectives are to; study and analyse existing solutions for SLA management from existing paradigms, such as Grid computing and web service, propose and develop a complete SLA framework to support negotiation, including all the SLA life cycle, implement my agent Wise H-T, find and use the State-of-the-art-Agent, evaluate my agent against the State-of-the-art-Agent. In terms of research scope, this work will cover up and link a variety of five research areas; Cloud Computing,

intelligent agents, game theory, negotiation and machine learning. Finally, the structure of the thesis has been highlighted.

The next chapter will give a detailed background about cloud computing, which will include the definition of what is cloud computing, its history as well as the related technologies and cloud computing layers. Afterwards, service level of agreements, negotiation and game theory will be discussed in detail.

Chapter 2

Background

In the last chapter, the aims and objectives, methodology, novelty contributions and thesis structure are discussed. In this chapter, a background and review of the existing literature on cloud computing will be performed and discussed to support the study undertaken in this thesis. The background and review of existing literature will cover in detail all the related topics: Cloud Computing, Negotiation and Service Level Agreement (SLA).

2.1 Cloud Computing.

In this section, the definition of cloud computing, the history of computing and related technologies; Grid computing, utility computing, Virtualization and Autonomic computing will be discussed.

2.1.1 Definition

In the literature, Cloud computing is yet to have a clear and complete definition. This is important as it is a very important task in the aim of determining areas of research and exploring the various uses of the Clouds through new application domains. The objective of

(Luis , et al., 2008) paper was to analyse in a formal manner all the features of Cloud computing so a definition that better incorporates them can be reached. They (Luis , et al., 2008) also aim to discuss the concept of the term to achieve a complete definition. An analysis of the main definitions of this term extracted from the literature has been done as a way to provide both essential and integrative Cloud definition in order to tackle the inconclusiveness of this problem. To reach a global definition as well as a minimum definition incorporating only the essential characteristics, more than 20 different sources have been studied in (Luis , et al., 2008) paper. In the end, (Luis , et al., 2008) defined cloud as :

"A large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services).These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs "

2.1.2 History of Cloud Computing.

In the 1960s John McCarthy introduced to society the concept of cloud computing, where his suggestion was that computing facilities could be provided to the general public as a public utility (Parkhill, 1966). Afterwards, in 1966 Douglas Parkhill explained the characteristics of cloud computing in his book "The Challenge of the Computer Utility". Moreover, in the 1990s the community started using the term "cloud" to describe large ATM networks and Virtual Private Network (VPN) services (Jadeja and Modi, 2012) (Zhang and Cheng, 2010).

2.1.3 Related technologies

There are some technologies, which shares certain aspects with cloud computing and which often leads to a confusion and comparison. These technologies are Grid computing, Utility computing, Virtualization and Autonomic computing. Each of these technologies will be described below.

2.1.3.1 Grid Computing

Grid computing is a distributed computing model that manages networked resources to accomplish a joint computational goal. The similarity between Cloud computing and Grid computing is that both of them uses distributed resources in order to achieve application-level objectives. However, the difference is that cloud computing takes one step ahead by taking advantage of virtualization technologies at multiple levels, including hardware (Zhang, Cheng and Boutaba, 2010).

2.1.3.2 Utility Computing

Utility computing is a model of providing on-demand resources and charging users based on "pay-as-you-go" model rather than upfront payment. The different between cloud computing and utility computing is that in cloud computing, virtualization is used. The benefit of providing on-demand resource is that the service provider can maximize resource utilization and minimize their operating costs (Zhang, Cheng and Boutaba, 2010).

2.1.3.3 Virtualization

Virtualization technology emulates physical computing resources, including desktop computers and servers, storage systems, networking, CPU and memory, as well as the

individual applications. Virtualization generates “virtual environments” by using a hypervisor that permits multiple guest virtual machines (VM) to run on one physical computer, as if each has its own private computer (Zhang, Cheng and Boutaba, 2010).

2.1.4 Cloud Computing Layers

The architecture of cloud computing environment can be separated into three layers which can be seen and delivered “as a Service”. All these layers will be explained with some examples.

2.1.4.1. Infrastructure as a Service (IaaS)

Infrastructure as a Service is a way of providing virtualized resources by using the virtualization technologies on the top of physical resource such as Xen (Xen, 2013), KVM (KVM, 2013) and VMware (VMware, 2013). Instead of purchasing infrastructure resources in a traditional way, this layer allows the customer to rent on-demand resources with low cost and “pay-as-you-go” model.

In this section, two of the leaders in the Cloud Computing market will be chosen to explain how the IaaS in the commercial Cloud Computing market is managed. The two companies/ services are Amazon EC2 and Google Compute Engine. Both companies offer customer IaaS as instances which are virtual machine (VM) in take-it-or-leave-it way. Google Compute Engine only offers 15 types of instances. Amazon EC2 only offers 23 types of instances. However, in this work the customer can customise their own unique instances.

Amazon EC2 gives the customer the ability to import and export VMs from and to the Amazon EC2 Infrastructure but this need to be done manually. Google Compute Engine does

not offer this service. However, this work proposes an automated way for importing and exporting VMs.

2.1.4.2 Platform as a Service (PaaS)

Platform as a Service is a way of providing on-demand, ready to use platform, such as Microsoft Azure, Google AppEngine, Amazon SimpleDB/S3. Compared to the traditional way of buying the whole packages of the environment to build and run software and websites, this layer promises to provide a scalable resource which is easy to access with less maintenance required (Zhang, Cheng and Boutaba, 2010).

2.1.4.3 Software as a Service (SaaS)

Software as a Service is a software distribution model where the software is hosted by the cloud provider and is made available to the customer over the internet. Normally customers pay for this through the subscription or "pay-as-you-go" model. However, there are some cases when the provider makes the service available for free, but with heavy advertising. SaaS example would be GoToMeeting or SalesForce (Zhang, Cheng and Boutaba, 2010).

2.2 Service Level Agreement

In this section, the definition of Service Level Agreement, SLA Components and SLA Lifecycle will be discussed.

2.2.1 SLA Definition

SLA has been used since 1980s in the different areas, where the definitions itself depend on the context and vary from one area to another. According to Dinesh, et al., (2004), the SLA

could be defined as “An explicit statement of expectations and obligations that exist in a business relationship between two organizations: the service provider and customer”.

2.2.2 SLA Components

Rick (2002) stated that SLA defines couple of things, which include the delivery ability of a provider, the scope of guaranteed availability, the performance target or consumers’ and customers’ requirement, and the measurement and reporting mechanisms.

Below are the complete descriptions of the SLA components, which were provided by Jin, et al., (2002):

- **Purpose**: SLA goals.
- **Restrictions**: Steps required to make sure that the requested level of services are provided.
- **Validity period:** How long does the SLA take time?
- **Scope**: Services that are and are not covered by the SLA
- **Parties**: Any involved organizations or individuals (e.g. provider and consumer) involved as well as their roles.
- **Service-level objectives (SLO):** Agreed level of service.
- **Penalties:** If the service which was delivered does not achieve SLOs or is below the performance measurement, some penalties will follow.
- **Optional services**: Services might be required although they are not compulsory.
- **Administration**: Processes that are used in order to guarantee the achievement of SLOs.

2.2.3 SLA Lifecycle

SLA life cycle has been defined by Ron, et al., (2001) in three phases; Creation phase, Operation phase and Removal phase. The 3 phases are:

- **Creation phase:** in this phase the customers find the service provider who matches their service requirements' the best.
- **Operation phase:** in this phase a consumer has only the permission to view the SLA, without any possibility to change it.
- **Removal phase:** in this phase SLA is already ended. Consequently, all the associated configuration information is being removed from the service systems.

Sun Microsystems Internet Data Center Group (2002) characterized even more detailed SLA life cycle, which included six steps; The 6 steps are:

- **Discover** – service providers: in the first step the service providers are located according to the requirements of the consumer.
- **Define** – SLA: in this step in order to reach a mutual agreement, it is possible to negotiate between parties. 'Define – SLA' step includes the definition of services, parties, penalty policies and QoS parameters.

- **Establish** – agreement: in this step SLA template is created and filled in by the specific agreement. Moreover, the commitment of the parties to the agreement occurs.
- **Monitor** – SLA violation: in the step of 'Monitor' the delivery performance of the provider is measured.
- **Terminate –** SLA: in this step SLA ends because of the timeout or because of any party's violation.

- **Enforce –** Penalties for SLA violation: in the last step the penalties are executed if any of the parties violated the contract terms.

2.2.4 IaaS attributes for SLA negotiation.

Table 3 show the subset of the IaaS attributes that will be use in this research. This subset is sufficient because it covers the essential parameters and attributes of IaaS that customers and providers concern about the most.

Table 2 : IaaS attributes.

Attributes	Description
Availability Zone (location)	Physical location of the server that hosting the VM.
CPU capacity	CPU speed for VM
Memory size	Cash memory size for VM
Storage	Storage size of data for short or long term of contract
Availability	Uptime of service in specific time
VM Operating System	Operating System inside the VM.
Platform	Platform can be 32-bit or 64-bit
Security	The level of security of the VM and physical server.

2.3 Negotiation

In this section, the definition of Negotiation, Human problems with negotiation and Negotiation Protocols will be discussed.

2.3.1 Definition

There are many areas studied the topic of negotiation, including economics (Martin, Osborne and Rubinstein, 1990), (Raiffa, 1982), e-commerce (Guttman, Moukas and Maes, 1998), (Lomuscio, Wooldridge and Jennings, 2003), artificial intelligence (Gerding, et al., 2000), (Jennings, et al., 2001), (Kraus, 1997), (Kraus, 2001), (Giampapa and Sycara, 2003), (Silaghi,

Serban and Litan, 2010), game theory (Binmore and Vulkan, 1999), (Gerding, et al., 2000), (Jennings, et al., 2001), (Giampapa and Sycara, 2003), (Martin, Osborne and Rubinstein, 1990), (Liang and Yuan, 2008), (Rubinstein, 1982), and social psychology (Rubin, Brown and Deutsch, 1975). However, the general definition for negotiation as Thompson (Thompson ,2012) defined it is: " a decision process in which two or more parties make individual decisions and interact with each other for mutual gain".

2.3.2 Human problems with negotiation

It is very important to study in detail the human problems and weakness with negotiation, as it will help to find out what kind of agent that will help and even surpass the human capabilities.

The most important problems from two perspectives (an outcome and a process) will be addressed in this section. According to (Thomson, 2005) the main outcome related pitfalls in negotiation are:

Leaving money on the table: when the negotiators fail to recognize and exploit win-win potential.

Settling for too little: the concessions that the negotiator makes may be too large thereby agreeing to a too-small share of the bargaining pie.

Rejecting a better offer than any other available option: this happens when the negotiator ends a negotiation process even though the opponent has provided a better offer than the other available options and when no agreement is reached.

Settling for terms worse than alternative options: this happens when negotiators feel obligated to agree to an offer that is worse than any other alternatives.

The pitfalls from the outcome perspective are caused by problems that occur during the negotiation process. There are some aspects that are recognized in the literature:

Lack of training (Thompson, 2005). Humans have the difficulty in arranging negotiation problems and thinking creatively about similar problems. Moreover, just negotiating in practice does not alleviate these problems due to incorrect feedback and self-reinforcing incompetence.

Lack of preparation (Thompson, 2005) (Harvard, 2003) (Filzmoser 2010). Preparation is incomplete when the negotiator is unaware of an important part of the bargaining pie as well as the circumstances and preferences of the parties involved, this might include himself.

Structural barriers to agreement (Harvard, 2003). This refers to such problems as: die-hard bargainers, differences of the culture and gender, confused or incommunicative people at the table, and a lack of information. This might be caused by insufficient preparation as well as by communication problems.

Mental errors (Harvard, 2003; Filzmoser 2010). Parties commit mental errors such as the escalation error, irrational expectations, overconfidence and biased perception. "The escalation error is the continuation of a previously selected course of action beyond the point where it continues to make sense" (Filzmoser 2010). Furthermore, "Biased perception is the problem of perceiving the world with a bias in your own favour" (Clancey, 1989; Harvard, 2003, Thompson, 2005).

Satisfying (Thompson, 2005) (Simon, 1955). Due to the uncertain future, the costs of obtaining the information, as well as the limitations of people's computational capacities made them bound only rationality by forcing them to make only satisfying decisions, not maximal.

This has led to an attention to automate negotiation (Beam and Segev, 1997), (Guttman, Moukas and Maes, 1998), (Jennings, et al., 2001), (Kraus, 2001), for example in the setting of e-commerce (Bosse and Jonker, 2005), (Kowalczyk, Ulieru and Unland, 2002), (Lomuscio, Wooldridge and Jennings, 2003). This interest is fuelled by the promise of computer agents being able to negotiate on behalf of human negotiators, or even outdoing them (Bosse and Jonker, 2005), (Jazayeriy, et al., 2011), (Lomuscio, Wooldridge and Jennings, 2003), (Oshrat, Lin and Kraus, 2009). In order to build and automate negotiation, the negotiation problem needs to be selected. The next section will discuss five e-negotiation protocols which will help to understand requirements of automatic negotiation and select the most suitable approach for our problem

2.3.3 Negotiation Protocols

The negotiation protocol is needed to determine the overall order of actions during a negotiation. The communication between negotiating parties is regulated by a negotiation protocol that describes the rules of how and when offers can be exchanged.

Many negotiation protocols have been developed over the years (Kexing, 2013), and to find the most useful one to cloud computing negotiation is not an easy task. In this section, the five commonly used e-negotiation protocols will be discussed. Moreover, State diagrams models (Rinderle and Benyoucef, 2005) for each protocol will be provided.

2.3.3.1 Fixed Price:

The first protocol is fixed price protocol. This protocol, which can also be called the “take-it-or-leave-it” protocol, is a special case of e-negotiation process where the exchange of offers and counter-offers does not occur. The “offer to sell” message creates the unique offer from the seller, which from this point can be either accepted by the buyer or be withdrawn by the seller himself in this way closing the negotiation process.

2.3.3.2 English Auction:

In the English auction protocol, An Updated message, which contains the bid submitted by the rival buyer, is sent to each possible buyer and, to which that each possible customer can respond with a counter-bid. After a particular time the auction is always closed.

2.3.3.3 Dutch Auction:

This type of auction often used to buy several items at the time. Moreover, a Dutch auction is a popular way to sell perishable and valuable goods, where the seller starts with the highest price and afterwards gradually decreases it (Kumar and Feldman, 1998). The seller keep making offers by decreasing the price until one of the buyers accept the offer. After that, the action will be closed.

2.3.3.3 Double Auction:

In the Double auction the buyers and the sellers are all bidding at the same time. A match between a seller's and buyer's bid implies a deal. When both bids are matched then the deal is made.

2.3.3.4 Bargaining and Rubinstein Model

In the Bargaining protocol, where both the seller and the buyer can make the offers, what makes this protocol a two party negotiation model. Typically, the initial and the first offer are made by the seller, but as it is seen in figure 8 the occasions where the buyer makes the initial offer are possible too.

In this work I will use Protocol called Rubinstein's Alternating Offers Protocol or as knowing as Rubinstein bargaining model as formalized in (Rubinstein,1982). Rubinstein's solution is one of the most influential findings in game theory. In Rubinstein's Alternating Offers Protocol, There's no delay in the transaction. A Rubinstein bargaining model has the following elements:

It supports two sides bargaining. It can support an unlimited number of offers (the negotiation keeps going until one side accepts an offer) if a deadline is not set. Alternating offers—the first side makes an offer, then the second side makes a counteroffer if he rejects the first party's offer, then the first side makes another counteroffer if he rejects, and so forth. Figure 9 shows how this protocol works.

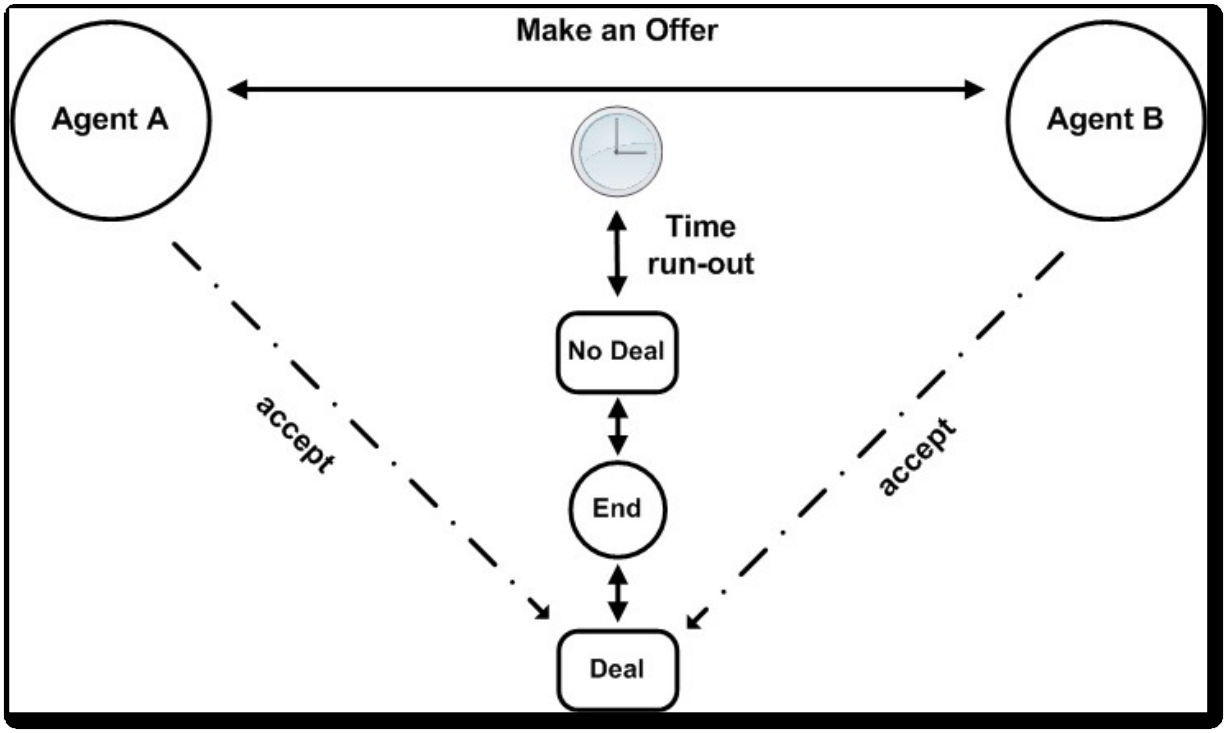


Figure 2 : Rubinstein's alternating offers protocol

This protocol is chosen due to its features;

- It supports bilateral negotiation.
- It is fast and there is no delay because the opponent can reject the offer by sending counteroffer, rather than first informing the opponent about the rejection then making counteroffer.
- It can represent how the cloud provider and cloud customer negotiate nowadays over the phone and in face to face negotiation.

Moreover, this protocol has been widely studied and used in the literature, both in game-theoretic and heuristic settings (Fatima et al., 2002) (Kraus, 2001) (Kraus et al., 1995) (Osborne & Rubinstein, 1990) (Osborne & Rubinstein, 1994) (Azzurra et al., 2007)(Nicola et al., 2009).

2.4. Game Theory Concepts

This section will clarify what game theory is and how negotiation can be seen as a game; also the negotiation dilemma will be described. Then, utility theory (Von Neumann and Oskar Morgenstern, 1944), utility maximization (Roger B. Myerson, 1991), Pareto frontier (Williams Colin R, 2012) and Nash equilibrium (Nash, 1950) will be explained.

2.4.1 Game theory

Game theory can be defined as "the study of the mathematical models of conflict and cooperation between intelligent rational decision-makers" (Roger B. Myerson, 1991) Game theory was created by Von Neumann and Oskar Morgenstern in their book *the theory of game and Economic Behavior*, which published in 1944 (Binmore, 1992). Game theory studies interactions among self-interested agents. In game theory, the outcome depends on choices of the players (Agents) and each player (Agent) has preferences for the entire possible outcome (Brams, S. J. ,2003). Game theory provides "general mathematical techniques for analysing situation in which two Agents make decisions that will influence on others welfare". (Roger B. Myerson, 1991). From the game theory perspective, in this work I cover Zero-sum game and Win-win game (Binmore, 1992)

2.4.2 Negotiation as a Game and negotiation Dilemma.

Negotiation can be seen as a game played by two *players* (Agents). There are some *Actions* each *players* can take. The actions based on Rubinstein's alternating offers protocol are;

offer, *Accept*, *End*. Before and during negotiation, players will face the *Dilemma of Negotiation* which is shown in table 3, highlighting how each agent choice will change the negotiation outcome.

Table 3 : Dilemma of Negotiation

	Agent B Cooperates	**Agent B Competes**
Agent A Cooperates	**Win-Win Agreement " Fair"** Agent A will gain average utility Agent B will gain average utility	**Agreement** Agent A will gain low utility Agent B will gain high utility
Agent A Competes	**Agreement** Agent A will gain high utility Agent B will gain low utility	**No agreement** because both Agents do not want to give in.

2.4.3 Utility Theory and Utility Maximization

Utility is a vital concept in economics and game theory, because it represents the satisfaction experienced by each agent of each possible agreement. The utility theory was established by Von Neumann and Oskar Morgenstern (Von Neumann and Oskar Morgenstern, 1944). They proposed the foundation of using utilities to represent preferences (Von Neumann and Oskar Morgenstern, 1944). Each agent will seek to maximize their own utility.

Utility maximization is defined as "The method of modelling choice by assuming that an individual's preferences can be represented by a utility function which they seek to maximize" (John B, Nigar H, and Gareth M, 2009).

2.4.4 Pareto Frontier

The negotiation outcome is considered Pareto efficient, or Pareto optimal if there is no other outcome that will make one agent better off without making the other agent worse off. Such

an outcome is efficient as no utility is wasted (Williams Colin R, 2012). In other words, if the negotiation outcome is not Pareto efficient, then there is another outcome that will make one agent happier (higher Utility) while keeping the other agent at least as happy (same utility). (Jennings, et al., 2001). Pareto frontier is a line that links the set of all of the possible outcomes that are Pareto optimal (Williams Colin R, 2012).

2.4.5 Nash Equilibrium

Nash equilibrium (Nash, 1950). Nash equilibrium is "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 B, Nigar H, and Gareth M, 2009). The different between Nash point and Pareto frontier, is that Nash point tell us which Pareto bids is the Nash equilibrium.

2.5 Intelligent Agents

2.5.1 What is an Agent and what are the Capabilities of Intelligent Agents?

The negotiation needs to be automated to handle the dynamic and complex environment cloud computing. The automated negotiation will be run by intelligent agents. The agent can be defined as "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" (Wooldridge and Jennings, 1995). Hence, when do I consider an agent to be an intelligent agent? This question is similar to the one "what is an intelligence?" itself, it is hard question to answer. However, researchers (Wooldridge, 2002), (Padgham and Winikoff, 2004), (Russell and

Norvig, 2010) suggested a way to answer this question, which is by studying the capabilities the intelligent agents have and what I may expect an intelligent agent to be:

- *Situated*. Exists in an environment.
- *Autonomous*. An Intelligent agent must be independent and not controlled externally.
- *Reactive*. Intelligent agents must respond in their timely manner to the changes that occur in it in order to satisfy their design objectives.
- *Proactive*. In order to satisfy the designed objectives of an Intelligent Agent, the agents are able to exhibit goal-directed behaviour by taking the initiative.
- *Flexible*. They have multiple ways to achieving their goals.
- *Robust*. Intelligent agents recover from failure.
- *Social*. They are capable of interacting with other agents as well as humans.

2.5.2 Agents for Cloud computing.

There are many advantages of using Agents for managing Cloud Computing environments; one of the most important advantages is that Agents are especially amenable for managing Cloud Computing environments because they can be mobile and hence fitting to the ‘migration’ attribute of the 'elastic ‘ Cloud environments. The second advantage is that Agent are interoperable, thus facilitating the communication with ‘other’ agents installed on top of a potentially heterogeneous Cloud infrastructures. This can be done by using the extensible messaging and presence protocol (XMPP) which will make the communications between the agents more suitable for the cloud computing environment because XMPP is based on decentralization. Using XMPP will be one of our future works. See section 9.5.4 for more information.

2.5.3 Rational Agents, Preferences and Utility function.

On the top of previously mentioned capabilities, these types of agents are expected to be rational. A part of being rational means to have fixed preferences and not acting 'dumb', meaning not committing to two courses of action that may conflict. I expect the intelligent agent to be self-interested. The term utility refers to the quality of being useful and it is a numeric value which measures the satisfaction of the state (the negotiation outcome). Utility functions are just the way of representing an agent's preferences. The ultimate goal of each agent is to maximize its utility. When two utility-based agents try to maximize their utility in the negotiation process, there often occurs a conflict. That is when the Game theory comes in handy. Game theory is a mathematical theory that studies interactions among self-interested agents (Binmore, 1992). Negotiation can be seen as a game, where two agents try to come to an agreement. Each agent is assumed to have a fixed preference over all possible deals. Both agents face the problem, meaning that they are both trying to maximize their utility function. They also face the risk of a break-down in negotiation process or even an expiration of a deadline.

The provider and customer will negotiate over a set of *issues*, and every issue has an associated range of alternatives or values. A negotiation outcome consists of a mapping of every issue to a value, and the set of all possible outcomes is called the negotiation *domain*. Both parties have privately-known preferences described by their utility functions. Both utility functions, map every possible outcome $\omega \in \Omega$ to a real-valued number in the range $[0, 1]$, where ω is the outcome and Ω is the domain. The overall utility consists of a weighted sum of the utility for each individual issue.

$$U(_{v1,\ldots\ldots vn}) = \sum_{i=1}^{N} w_i \frac{eval(v_i)}{Max(eval(v_i))}$$

A bid is a set of chosen values $_{v1,\ldots\ldots vn}$ for each of the n issues. Each of these values has been assigned an evaluation value $eval(v_i)$ in the utility space. The utility is the weighted sum of the normalized evaluation values. While the domain (i.e. the set of outcomes) is common knowledge, the utility function of each player is private information. This means that the players do not have access to the utility function of the opponent. However, the player can attempt to learn during the negotiation .The negotiators (provider and customer) will be represented by agents. Each agent has a different strategy of negotiating. The ideal agent needs to be rational to be able to;

- Learn about the opponent behaviour from its moves to predict the opponent's next moves.
- Decide when to make a cooperative offer or a selfish offer.
- Decide when to accept the opponent's offer.
- Keep track of the remaining time in the negotiation session.
- Decide when to end the negotiation without agreement.
- Estimate the Nash-Equilibrium point (Nash, 1950).

2.5.4 Agents communication.

The negotiation protocol is needed to determine the overall order of actions during a negotiation. In this work a Protocol called - Rubinstein's Alternating Offers Protocol also known as Rubinstein bargaining model will be used as formalized in (Rubinstein, 1982). Rubinstein's solution is one of the most influential findings in game theory. In Rubinstein's Alternating Offers Protocol, there's no delay in the transaction. Furthermore, this protocol is

chosen due to its simplicity; it is a protocol which is widely studied and used in the literature, both in a game-theoretic and heuristic setting (Filzmoser, 2010). This protocol is a one-to-one protocol (Agent-to-Agent): Agents negotiate over a series of rounds. At the first round, an agent makes an offer then the other agent either accepts or rejects it. If the offer is accepted, the deal is implemented (Agreement). If the offer is not accepted, then the negotiation keeps going until one agent accepts the other offer.

2.6. The Bidding Opponent Acceptance (BOA) Framework.

The Bidding Opponent Acceptance (BOA) framework is used to form our agent Wise H-T. (baarslag , et al., 2012a). The BOA negotiation agent architecture allows researchers to re-use existing components from other BOA agents, create new agents and compare them with BOA agents in the same environment. (Dirkzwager, 2013) (Dirkzwager, et al., 2012).

The BOA agent can be made of three different modules, one module that decides whether the opponent's bid is acceptable (*acceptance strategy*); one that decides which set of bids could be proposed next (*bidding strategy*); one that tries to guess the opponent's preferences (*opponent model*). The overall negotiation strategy is a result of the complex interaction between these components (baarslag, et al., 2012a). Table 4 shows the input and output of each component. (Dirkzwager, 2013) (Dirkzwager, et al., 2012).

Table 4 : BOA framework's components

Components	Input	Output
Bidding strategy	opponent utility of bids, negotiation trace	Provisional upcoming bid.
Opponent model	Set of possible bids, negotiation trace.	estimated opponent utility of a set of bids
Acceptance strategy.	Provisional upcoming bid, negotiation trace	Send accept, or send out the upcoming bid.

2.7 Conclusion

This chapter gave a detailed background about; Cloud Computing, Service Level of Agreements, negotiation, Game theory and intelligent agent. In terms of cloud computing, this chapter covered; the definition for Cloud Computing, Cloud Computing's history as well as the related technologies, Cloud Computing deployment models and Cloud Computing layers. In terms of Service Level of Agreements, this chapter covered; the definition of SLA, components and lifecycle. In terms of negotiation, this chapter covered; the definition of negotiation, human problems with negotiation and automated negotiation protocols. In terms of game theory, this chapter covered; definition of game theory, negotiator's dilemma, Nash bargaining solution and the concept of Pareto frontier. In terms of intelligent agents, this chapter answered the questions; what is an agent and what are the capabilities of an intelligent agent? Then, this chapter explained; the agent rationally, agent preferences and utility function, Agents communication and the Bidding Opponent Acceptance framework. The next chapter will cover the related works.

Chapter 3

Related Work

In the last chapter, the background is introduced. In this chapter, the related works will be discussed. The related work can be classified in the following categories: Frameworks testbeds, SLA, Negotiation Support Systems, Agent for grid computing and State-of-the-art Negotiation Agents.

3.1 Negotiation Frameworks

In this section, I will review the negotiation frameworks that have been proposed by others to discuss the limitations in each framework. In (Linlin, et al., 2013) , the authors proposed a novel automated negotiation framework where "a SaaS broker is utilized as the one-stop-shop for customers to achieve the required service efficiently when negotiating with multiple providers". (Linlin, et al., 2013). However, it only supports fixed and limited multi-issue negotiation for Software as a Service layer of cloud computing. Also, they assume that cloud customers always prefer using brokers to negotiate for them. The problem with using brokers is that brokers will try their best to end the negotiation with agreement, so they can obtain a commission from the deal/ agreement. In the work reported on, in this thesis, customers and providers negotiate directly with each other. Furthermore, Linlin et al. presume that in every

negotiation the only thing that customers care about the most is lowering price. They also presume that the only thing that providers care about the most is maximizing the profit. These are the limitations in Linlin et al work.

In our work, I give the customers and providers the complete control to rank and give their preferences in each negotiation issue.

In (Seokho and Sung, 2013) the authors designed a cloud SLA negotiation mechanism to support a flexible establishment of SLAs for both providers and consumers. The novelty of the SLA negotiation mechanism is that it can support multi-issue negotiation. But it only supports 2 issues which are time slot (Duration) and price negotiations. Also, in (Seokho and Sung, 2013) work providers and consumers have to end the negotiation with an agreement. While, in our work, the providers and consumers have the right of ending the negotiation without agreement. In our work I do not force both sides to end the negotiation with an agreement.

3.2 Service level Agreement (SLA)

3.2.1 Service level Agreement (SLA) and Negotiation

The most important specifications which are designed to explain the languages of SLA are: Web Service Agreement (WS-Agreement) (Thompson ,1998) and Web Service Level Agreement Language and framework. (WSLA) (Keller and Ludwig , 2003). However, neither of them investigates the Cloud Computing paradigm of outsourcing the services in a pay-as-you-go framework.

(Andrieux and Czajkowski. 2007) proposed an XML based language called WS-Agreement for specifying contract suggestions in the negotiation process, or specifying a contract if the negotiation process terminates successfully, with an agreement. Oldham et al., 2006 extended WS-Agreement to be semantic-enabled. However, in neither (Andrieux and Czajkowski. 2007) nor (Oldham et al.,2006) is it possible to develop negotiation protocols and strategies.

There are other works, however, which focus on SLAs and negotiation in related areas such as Grid computing (Al-Ali, et al., 2002) , (Lee, et al., 2008). (Al-Ali ,et al., 2002) extended the service abstraction in the Open Grid Services Architecture (OGSA) for Quality of Service (QoS). Meanwhile (Lee , et al., 2008) discuss integration of SLA-based resources with existing Grid systems.

3.2.2 SLA frameworks contents and issues

To the best of our knowledge, there is little research for cloud computing in this area (SLA framework). In (Alhamad , et al.,2010) , the authors present the main criteria which should be considered at the phase of designing the SLA in cloud computing including functional and non-functional requirements for cloud users. However, the proposed framework by Alhamad , et al.,2010 has not been designed and implemented.

3.3 Application of negotiation in related areas

In (Cao Mukun, 2010) the authors posit that making the automated negotiation system as a software service in line with the SOA (Service Oriented Architecture) is a practicable way for the practical application of the automated negotiation system. They discussed a technology roadmap for the development of automated negotiation systems using the software agent technology. They proposed two application architectures based on SOA and web services

technology, the first one is Architecture for SOA based automated negotiation service deployed in a third party e-commerce platform using public Universal Description, Discovery and Integration (UDDI) . The second is Architecture for SOA based automated negotiation service system with private UDDI. However, they do not take into account the dynamic nature of the cloud needs, and having multiple cloud providers.

3.4 Negotiation Support Systems

There are some existing negotiation frameworks and negotiation support systems that have already been developed: OPELIX (Hauswirth, et al., 2008) is a European project that enables a customer and provider complete fully automated bilateral negotiations. The OPELIX system implements all the important phases of a business operation including product offers and discovery, a negotiation process, payment activities, and the delivery of the product to the customer. Other related work has introduce two associated projects, Inspire (Kersten and Noronha, 1999) and Aspire (Kersten and Lo, 2003). Inspire (Kersten and Noronha, 1999) helps human operators in managing bilateral negotiations by organizing offers and counter-offers. Aspire (Kersten and Lo, 2003) improves upon Inspire by giving negotiation support using intelligent agents to make suggestions to the negotiators. Agents in Aspire do not completely run the negotiation process, but offer help in taking decisions. Though, they are fully aware of the status of the negotiation sessions. Another system, known as Kasbah (Chavez, et al., 1997) allows a customer and a provider to generate their own agents, assign them some strategic directions, and launch them at centralized marketplace for negotiations. Similarly, CAAT (Ncho and Aimeur, 2004) is a framework which can be used to design multi-agent systems for bilateral negotiations. The negotiation protocol allows valid series of interactions using messages.

The Negotiation frameworks and Negotiation Support Systems (NSS) presented above certainly make interesting advances towards automated negotiation; however they are not flexible enough to easily customize negotiations for individual application domains. In addition, there are a number of issues that the above-mentioned works largely ignore which will be discussed and addressed herein. These include:

- The dynamic nature and heterogeneity of cloud computing.
- Each participant (provider and customer) has different preferences. The above works assume that price is the only and most important issue for the customer.
- Each participant (provider and customer) can build their agent or select one of the agents provided. Each agent behaves differently according to their strategy.
- Supporting multi-issue negotiation with a large domain (hundreds of thousands possible outcomes).
- Negotiating with multi-providers.
- Open for new agents and strategies.
- Possibility of re-negotiating.
- Monitoring the SLA after the negotiating.

3.5 Agent for Grid Computing and Cloud Computing.

Work in (Gheorghe, Şerban and Cristian, 2012) suggested using Automated and intelligent negotiation solutions for reaching SLA for an open competitive computational grid. However, SLA negotiations in grid are completely different from cloud computing negotiations. The SLA negotiations in cloud are more complex. In grid computing negotiations will be between users that would like to use the same resource. On the other hand, in cloud there are many providers who are competing for a customer and at the same

time a customer is looking for better deal by negotiating with many providers. Also, the offer and demand in the cloud market play a big role in choosing a negotiation strategy. The proposed agent as they called it AgentFSEGA in (Gheorghe, Şerban and Cristian, 2012) will be evaluated and compared against our propose agent Wise H-T.

For cloud computing, Many works, including (Ariya, et al.,2012) (M.Abirami, 2013)(Talia, 2012) (Domenico, 2011) (Kwang, 2012), proposed the idea of using agent for automating cloud computing management. However, no practical solutions were given due to complexity of the problem.

3.7 State-of-the-art Negotiation Agents

In this section, I will explain how the State-of-the-art Negotiation Agents work and how they different from each other. These include HardHeaded, Tit-for-Tat, Hardliner, IAMhaggler and AgentFSEGA. Each agent has three different modules, one module that decides whether the opponent's bid is acceptable (*acceptance strategy);* one that decides which set of bids could be proposed next (*bidding strategy*); one that tries to guess the opponent's preferences (*opponent model*). The overall negotiation strategy is a result of the complex interaction between these components (Baarslag, et al., 2012a).

3.7.1 HardHeaded

This agent (Krimpen, Looije and Hajizadeh, 2013) starts each negotiation session by computing the utility for all possible bids. Then it stores them in a search tree (binary tree data structure) for fast recovery. This agent uses a learning module which called HardHeaded Frequency Models see section (2.6.1). The target of the learning module is to learn the utility value and the weights for the opposing agent (Krimpen, et al., 2013). To study the opposing agent, this agent makes two assumptions about the opponent; it first assumes that the opponent restricts the bids within a limited utility range. The second assumption is that the opponent does not prefer to be offered the same bid over and over again. HardHeaded's learning function is "a greedy reinforcement learning function" (Krimpen, et al., 2013). This learning function keeps updating the issue weights and value utilities of the preference profile immediately after each bid. At the same time this learning function will always try to identify the most valuable bid, and the least valuable bid for the opponent, so it can offer a bid which is most likely to be accepted when the time of the negotiating session is about to end (Krimpen , et al., 2013). The fact that this agent uses a simple learning module and also an optimal concession function with low computational complexity, enables it to offers bids very fast. Also, this is why this agent is able to concede rapidly when the time of the negotiating session is about to end, and at the same time can still carefully explore the bid space (Krimpen , et al., 2013).

3.7.2 Tit-for-Tat

This agent's strategy is based on the principle of Tit for Tat (tft) (Axelrod, 1984). In Tit for Tat strategy, the first move is always a cooperative move and then keeps mirroring whatever the other player did in the previous round (Axelrod, 1984).This agent plays a tit for tat strategy with respect to its own utility. In the beginning, this agent will cooperate, and then respond to the opponent's previous action, while aiming for the Nash point of the negotiation scenario (Baarslag, et al., 2013a). After every opponent's move, this agent will update its Bayesian opponent model to make sure it reacts with a beneficial move to a concession by the opponent (Baarslag, et al., 2013a). This opponent model will help the agent to measure the opponent's concession in terms of the agent's own utility function; Mirror this bid as described in the tft strategy above, giving up the same amount as is offered by the opponent; Make the offer as attractive as possible for the opponent using the Bayesian opponent model (Baarslag, et al., 2013a). In addition, the opponent model is used by this agent to make an estimate of the location of the Nash point of the negotiation scenario, and then aims for this outcome (Baarslag, et al., 2013a).

3.7.3 Hardliner

Hardliner (Baarslag , et al., 2011a) is a very selfish and stubborn agent. At the beginning, this agent selects an offer which got the higher utility for itself. Then it keeps repeating that offer, expecting the opponent will give up and accept the offer at the end. This agent does not learn from the opponent and have no accepting strategy

Its approach to negotiation is known as "take-it-or-leave-it" strategy. This strategy makes a bid of maximum utility for itself and never concedes. This is the most competitive strategy that can be used. It will give the opponent the full negotiation time to make concessions and

accept its offer. This agent is widely used, as it represents nowadays cloud providers approach to negotiation e.g. Amazon ES2, Google Compute Engine and Microsoft Azure, since they only propose take-it-or-leave-it cloud packages offers in the market.

3.7.4 IAMhaggler

The Agent involves three parts; the first part predicts the concession of the opponent by using a Gaussian process regression technique (Rasmussen and Williams, 2006). The second part sets the concession rate in such way that it optimizes the expected utility given that prediction. The third part generates a multi-issue offer according to the concession rate (Williams , et al., 2013). This agent first has to predict how the opponent will concede during the negotiation, only by using the information which can be observed (the opponent's offers and the utility of these offers according to the agent's utility function. This agent uses a Gaussian process regression technique (Rasmussen and Williams, 2006): first, to predict the opponent's future concession; second, to measure the level of confidence in that prediction (Williams , et al., 2013). This agent uses the strategy of the opponent's future concession prediction, to set its concession rate by optimizing the expected utility given to that prediction. After selecting a target utility, this strategy needs to make an offer which has a utility close to that target.

3.7.5 AgentFSEGA

This agent is a Bayesain learing (Baarslag , et al., 2012b) agent; they adapted the Bayesian learning negotiation strategy to cope with time constraints. Also, The Bayesian learning during the negotiation, will try to infer the utility function of the other player by executing

two actions; First, analysing the incoming opponent's proposal and update the opponent's profile. Second, selecting and proposing the next bid. Bayesian learning (Hindriks and Tykhonov, 2008) calculate a probability for each possible hypothesis about the opponent profile. The hypothesis can be the rankings of the preferences for the issues of the opponent. The Agent is a time-constrained agent which means that base on the remaining time of the negotiation session; this agent will act and be more flexible.

3.8 Automated Negotiation Testbeds

There are only few testbeds to design and test Agent for automated negotiation. For this work, I am looking for a testbed which provides a number of state-of-the-art negotiation agents, and allows us to create our own agent to compare with and improve upon them in the context of cloud SLA negotiation. The testbeds that I was considering using in this work are; (ART) The Agent Reputation and Trust Testbed (ART) (Fullam, et al., 2005) or (GENIUS) Generic Environment for Negotiation with Intelligent multi-purpose Usage Simulation (Lin, et al., 2012). The Agent Reputation and Trust is "a Testbed initiative has been launched with the aim of establishing a testbed for agent reputation- and trust- related technologies" (Fullam, et al., 2005). It provides researchers with easy access to a common experimentation environment and allows researchers to compete against one another to determine the most viable technology solutions. But, the ART testbed project is only limited to trust and reputations. On the other hand, GENIUS provides far more flexibility with designing negotiation strategies, creating the size of negotiation domain and determining negotiation preferences. GENIUS is maintained regularly and it is used in the annual Automated Negotiating Agents Competition (ANAC). The aims of ANAC competition (ANAC, 2010);

designing of more efficient negotiating agents, Testing bidding and acceptance strategies, Exploring learning strategies and opponent models; Collecting the state-of-the-art negotiating agents, negotiation domains, and preference profiles and making them available for the negotiation research community(ANAC, 2010).

3.9 Contributions and Novelty Compared to Related Work.

There are many open research areas which related works largely ignored.

A complete SLA life cycle – Cloud computing is a new model of providing computer resource, that is why there is a need to create a new SLA life cycle which fits the dynamic nation and heterogeneity of cloud computing, taking into account every single step of SLA life cycle in details. In order to create a novel SLA life cycle especially for cloud computing, I need to learn and improve the existing SLA life cycle for related technologies.

Negotiation – The negotiation for cloud computing has to be flexible in four ways: multi-issue negotiation, automated negotiation, negotiating with the multiple providers and re-negotiation:

- Multi-issue negotiation – The customer and the provider needs to negotiate over the multiple issues with the possibility of giving preferences for each issue.
- *Automated negotiation* – The flexibility which gives the providers and customers the ability to perform the negotiation process by using provided intelligent agents as well as the possibility to build their own agents.

- *Negotiating with the multiple providers* – The customer should be able to negotiate with the multiple providers and then compare the outcomes with the each ones in order to choose the best deal.
- *Re-negotiation* – The customer needs to have the possibility of re-bargaining with the current provider in case if there is a better deal in the market.

***Agents and negotiation experiments**:* There is a need for novel experiments that need to be taken. Each experiment will help us to answer very important unanswered questions. First experiment will help the provider and customer to pick the right strategy for them. The second experiment will help us to experiment our agent (Wise H-T) against state-of-the-art agents. The third experiment helps us to understand how the domain size and deadline is affecting the negotiation outcome. Fourth experiment focused on price negotiation which is a single issue negotiation.

Real-time monitoring – It is important to monitor in a real time the experience of the customer in order to repair the problem if they experience less quality than they have been promised on SLA. Moreover, real-time monitoring should help the provider to predict future problems and avoid them.

3.10 Conclusion

In this chapter, I have reviewed related works includes; negotiation frameworks, negotiation testbeds, SLA, negotiation support systems, Agent for grid computing and state-of-the-art negotiation Agents. After reviewing related works, I found out that there are many open research areas which related works largely ignored includes the need to create a new SLA life cycle which fits the dynamic nation and heterogeneity of cloud computing, taking into the

account every single step of SLA life cycle in details. Also, the negotiation for cloud computing has to be flexible in four ways: multi-issue negotiation, automated negotiation, negotiating with the multiple providers and re-negotiation.

Base on the limitations of above the related works, I propose a novel cloud SLA framework. This framework will be called ACSLA framework (Automated Cloud Service Level Agreement). ACSAL is made up of five stages: Gathering, Filtering, Negotiation, SLA Agreement and Monitoring. Each stage will be explained in the next chapter. In the Gathering stage all the information about the providers and what can they offer is gathered. In the Filtering stage the customer's agent will send the request to the ACSAL, which will filter all the providers in order to recommend the best matched candidates. In Negotiation stage the customer's agent will negotiate separately with each candidate provider using different negotiation algorithms, which will be evaluated and for which recommendations and guidelines will be provided. Then, the outcomes of each session of the negotiation will be compared. The output of this stage is that the best outcome from the customer's perspective will be picked up, which will be the agreed value for each parameter in the SLA. In SLA Agreement stage the provider's agent and the customer's agent will be informed about the Agreement, which will be specified in measurable terms. The output of the SLA Agreement stage will be a list of metrics that can be monitored in the Monitoring stage. Customer's agent and provider's agent will also negotiate and agree about the penalties and actions which will be taken in case the SLA has been violated and unfulfilled. There is a variety of actions that can be taken, like informing both sides, updating the review and the ranking of the provider, recommending solutions, self-healing and hot-swapping of providers.

Chapter 4

Analysis

After reviewing the related works in the previous chapter, it became clear that there is a need to create a framework that will fulfil the provider's and customer's requirements. In this chapter, the requirements to achieve the agent-based SLA negotiation for cloud computing framework will be discussed. Each stage of the framework will be explained, including the input and the output for each stage.

4.1. Users scenarios

Before defining the requirements here I would like to give an example of scenarios to explain what I am trying to achieve, from a user point of view. First, I will give scenario of cloud providers. After that, cloud customers.

4.1.3 Cloud Providers (CP)

I will give a scenario of cloud provider that able to offer to the customers Infrastructure as a Service (VM virtual machine). Table 5 shows what this provider can offer.

Table 5 : provider scenario virtual machine

Issue	Value
Availability Zone (location)	US-East US-West Europe Asia
Operating System	Linux Microsoft Win
Term (months)	[1-6] [7-12] More than 12
Memory GB	7 8 9 10
Compute Units /virtual core (CPU)	4 5 6 7
Storage GB	[251- 500] [501- 725]
Platform	32-bit 64-bit
Utilization	Light < 39% Medium <75%

There are 8 *issues* of virtual machine: Availability Zone, Operating System, Term, Memory, Compute Units, Storage GB, Platform, and Utilization. Each issue has options known as *Value*s. The availability zone issue has 4 values. The operating system issue has 2 values. The Term issue has 3 values. The memory issue has 4 values. The storage issue has 2 values. The platform issue has 2 values. The utilization issue 2 values. For that this provider, can offer 768 packages of virtual machines.

After that, provider needs to give the *Evaluation* and the *Weight* for each *Issue* and *Value* to calculate the *utility* as I explained early in section (2.5.2).

Table 6 : provider scenario virtual machine with weight

Issue	Value	provider Evaluati	Weight
Availability Zone (location)	US-East US-West Europe Asia	100 50 25 75	0.19
Operating System	Linux Microsoft Win	100 50	0.09
Term (months)	[1-6] [7-12] More than 12	25 50 100	0.32
Memory GB	7 8 9 10	25 50 75 100	0.13
Compute Units /virtual core (CPU)	4 5 6 7	25 100 75 50	0.05
Storage GB	[251- 500] [501- 725]	50 100	0.08
Platform	32-bit 64-bit	50 100	0.05
Utilization	Light < 39% Medium <75%	50 100	0.09

4.1.2 Cloud Customer (CC).

I will introduce a scenario of customer that is looking for Infrastructure as a Service (VM virtual machine). Table 7 shows what the customer preferences are (including, the *Evaluation* and the *Weight* for each *Issue* and *Value)*.

Table 7: customer preferences.

Issue	Value	Customer Evaluation	Weight
Availability Zone (location)	US-East US-West Europe Asia	25 50 100 25	0.36
Operating System	Linux Microsoft Win	100 50	0.19
Term (months)	[1-6] [7-12] More than 12	100 50 25	0.10
Memory GB	7 8 9 10	75 50 25 100	0.10
Compute Units /virtual core (CPU)	4 5 6 7	100 50 75 25	0.04
Storage GB	[251- 500] [501- 725]	100 50	0.04
Platform	32-bit 64-bit	100 50	0.08
Utilization	Light < 39% Medium <75%	50 100	0.09

In the next section I will show what happen after creating provider offer and customer request via user requirements.

4.2. User Requirements

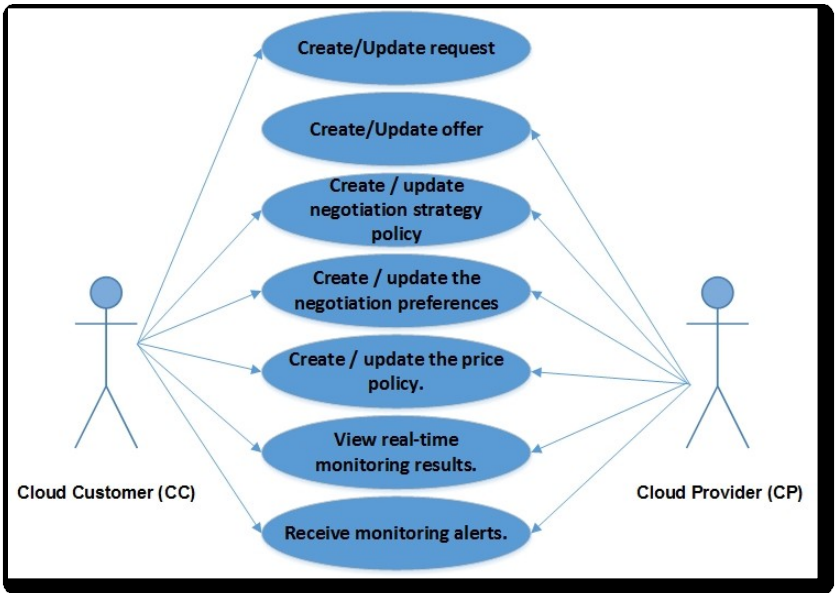


Figure 3 : Use Case Diagram

The user's requirements analysis is provided, which contains specification of user's functional requirements. Figure 11 is a use case diagram to illustrate user's requirements. Figure 11 shows that Cloud Customer (CC) and Cloud Provider (CP) will be able to:

- Create and update CC's request and CP's offer.
- Create and update the policy of negotiation's strategy.
- Create and update the negotiation's preferences.
- Create and update the price policy.
- Create and update the monitoring rules and policies.
- View the real-time monitoring results.
- Receive monitoring alerts.

Each of the above requirements, I will define; that will fit inside one of the stages of our proposed framework. The next section will detail each stage of the framework.

4.3 ACSAL Framework

To achieve an automated framework, which means less users' interaction, all the input from the users will be taken in advance in the earlier stages. The framework will consist of five stages: gathering, filtering, negotiation, SLA agreement and monitoring. The output of each stage is essential input for the next stage:

4.3.1 Stage 1: Gathering.

This stage is one of the most important stages, because all the input for the framework will be gathered together only in this stage. The inputs will be CC's request and CP's offer, the policy of negotiation's strategy, the negotiation's preferences, and the price policy. All the inputs will be saved in the accessible database. Moreover, this database can be updated by the users or automatically by the framework itself. Each user, whether it is the customer or the provider, will be able to create a profile about themselves.

4.3.1.1 Create and update CP's request and CC's offer (User requirement)

The users (Customers and Providers) need to be able to send and update the offers and requests via a variety of methods including: HTML form-based user interface, XML file, CSV file or Command Prompt.

Cloud Provider (CP) offer contains detailed criteria of all the services that can be provided, including what, where and for how long. Cloud Provider can update this information later, if needed. Cloud Customer (CC) request contains detailed criteria of the cloud service that a customer is looking for, including what, where and for how long a service is needed. The

CP’s request and CC’s offers will be kept in a database. This database needs to be online and easily accessible.

4.3.2 Stage 2: Filtering

Cloud computing marketplaces are dynamic and fast, where cloud providers and customer frequently join and leave. What's more, the cloud providers and customers’ preferences might change over the time. So, in this stage the customer will send the request to the framework, which will filter all the providers in order to recommend the best matched candidates. The customer request can include the detailed criteria of demanded service. At the same time, the customer can specify the preferred provider, for example a customer would like to choose the provider with the highest review, or based on the previous experiences. The output of this stage will be the candidate providers, with whom the customer will be negotiating separately.

In case of there is no matched candidate found then the customer will be in informed about the case. In case of the there is only one matched candidate found or more , customer and the provider will be in informed about the situation, So both of them can pick the best strategy of negotiation in this situation .

4.3.3 Stage 3: Negotiation.

In this stage the customer will negotiate separately with each candidate provider. Then, the outcomes of each session of the negotiation will be compared with each other.

4.3.3.1 Create and update the policy of negotiation’s strategy .

At the negotiation stage, the negotiators (cloud provider and cloud customer) will be represented by agents. Each agent has a different strategy of negotiation. Cloud provider and

cloud customer need to pick the right negotiation strategy based on supply and demand situation which can be told from the output of the filtering stage.

4.3.3.2 Create and update the negotiation preferences.

The cloud provider and cloud customer need to be able to setup the preferences for each issue in negotiation. Users need to be able to update the preferences of each issue in negotiation as the preferences might changes from time to time.

4.3.3.3 Create and update the price policy.

The negotiators (provider and customer) need to be able to set the price policy about the services. They need to setup a range of the price for each possible outcome of negotiation.

The provider and the customer need to update the framework and give the price for each value for each issue. Once the provider and the customer agreed about the package then they start to negotiate about the price for this package.

In order to calculate the *Negotiable Space* (See figure 12) the following steps will be required:

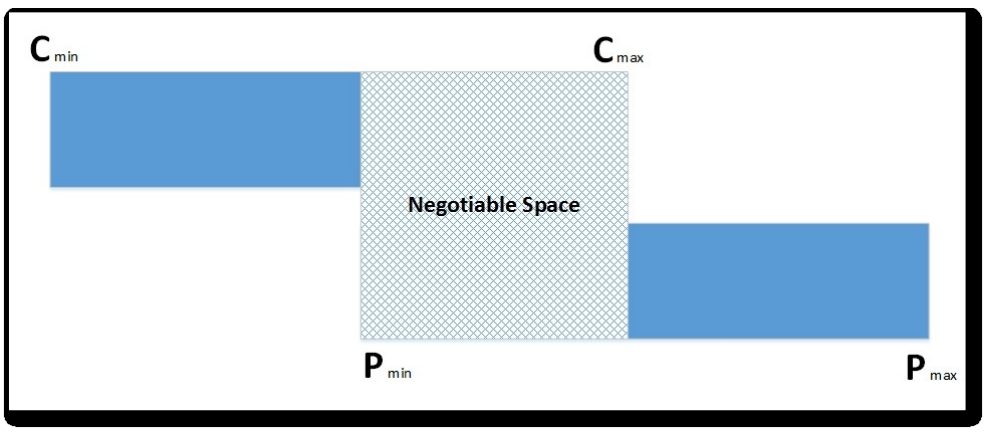


Figure 4 : negotiation Space

C_{min} Customer minimum price, which is the total minimum price, the customer is willing to pay. The C_{min} is calculated by adding all the minimum price of each value of an agreed package.

C_{max} Customer maximum price, which is the total maximum price that the customer is willing to pay. The C_{max} is calculated by adding all the maximum price of each value of an agreed package.

P_{min} Provider minimum price, which is the total minimum price, the provider would like to get for an agreed package. The P_{min} is calculated by adding all the minimum price of each value of an agreed package.

P_{max} Provider maximum price, which is the total maximum price, the provider would like to get for agreed package. The P_{max} is calculated by adding all the maximum price of each value of agreed package.

The following assumption has been made:

$$C_{min} < P_{min}$$
$$C_{max} < P_{max}$$

4.3.4 Stage 4: SLA Agreement.

In this stage the provider and the customer will be informed about the Agreement, which will be specified in measurable terms. The output of the SLA Agreement stage will be a list of metrics and the agreed package that are monitored by the following stage.

4.3.5 Stage 5: Monitoring.

This stage will use a monitoring client to gather the real-time data. Based on the monitoring rules and policies, the actions will be taken.

4.3.5.1 Create and update the monitoring rules and policies

The users need to able to setup monitoring rules and policies to generate alerts in the user dashboard. Monitoring rules and policies define the monitoring parameters as well as the alerting conditions. Monitoring policies is a setup of monitoring rules. In addition, the users need to setup rules and policies about the actions that need to be taken in case of SLA violations. The user will be able to choose what kind of action needs to be taken, whether it can be basic actions, like sending alerts; or advanced actions; self-healing or hot-swapping.

4.3.5.2 View the real-time monitoring results

The users (provider and customer) needs be able to view monitoring results in real-time using a monitoring dashboard. The monitoring dashboard needs to be online and easily accessible. The monitoring dashboard provides users with a quick overview and detail of the service level agreement status.

4.3.5.3 Receive monitoring alerts and violations

The users need to be able to setup when and how the alerts and notifications will be sent out to them like via SMS or Email.

4.4 Conclusion

This chapter explained the user functional requirements and the automated framework stages, including the Cloud Customer's and Cloud Provider's requests and offers; how the negotiation's strategy is being created and updated as well as how the negotiation preferences are being created and updated; the price policy; the monitoring rules and policy; also the real-time monitoring results along with receiving monitoring alerts and violations. In this chapter, I propose a novel cloud SLA framework. This framework called ACSLA framework (Automated Cloud Service Level Agreement). ACSAL is made up of five stages: Gathering, Filtering, Negotiation, SLA Agreement and Monitoring.

Each stage is explained in this chapter. In the Gathering stage all the information about the providers and what can they offer is gathered. In the Filtering stage the customer's agent will send the request to the ACSAL, which will filter all the providers in order to recommend the best matched candidates. In Negotiation stage the customer's agent will negotiate separately with each candidate provider using different negotiation algorithms, which will be evaluated and for which recommendations and guidelines will be provided. Then, the outcomes of each session of the negotiation will be compared. The output of this stage is that the best outcome from the customer's perspective will be picked up, which will be the agreed value for each parameter in the SLA. In SLA Agreement stage the provider's agent and the customer's agent will be informed about the Agreement, which will be specified in measurable terms. The output of the SLA Agreement stage will be a list of metrics that can be monitored in the Monitoring stage. Customer's agent and provider's agent will also negotiate and agree about the penalties and actions which will be taken in case the SLA has been violated and unfulfilled. There is a variety of actions that can be taken, like informing both sides, updating

the review and the ranking of the provider, recommending solutions, self-healing and hot-swapping of providers. In the next chapter, a high level design of the framework as well as a detailed design for each stage will be presented. Then, the overall framework closed loop will be presented.

Chapter 5

Design

In the previous chapter, provider's and customer's requirements and the framework stages have been defined. In this chapter, a high level design of the framework as well as a detailed design for each stage will be presented. Finally, the overall framework closed loop will be presented.

5.1. High level design

As mentioned in the last chapter, the main components of the proposed framework will be: gathering, filtering, negotiation, SLA agreement and monitoring. The following diagram (figure 13) shows high level design of framework's stages It shows the workflow of the framework where on the top is gathering stage getting providers offer and customer request. Then under the gathering stage is the filtering stage where only the selected candidates of the providers are passed to the next stage which is Negotiation stage. After completing the negotiation stage, the outcome of the negotiation will be passed to the next stage which is the Agreement stage. Then on the bottom is the monitoring stage.

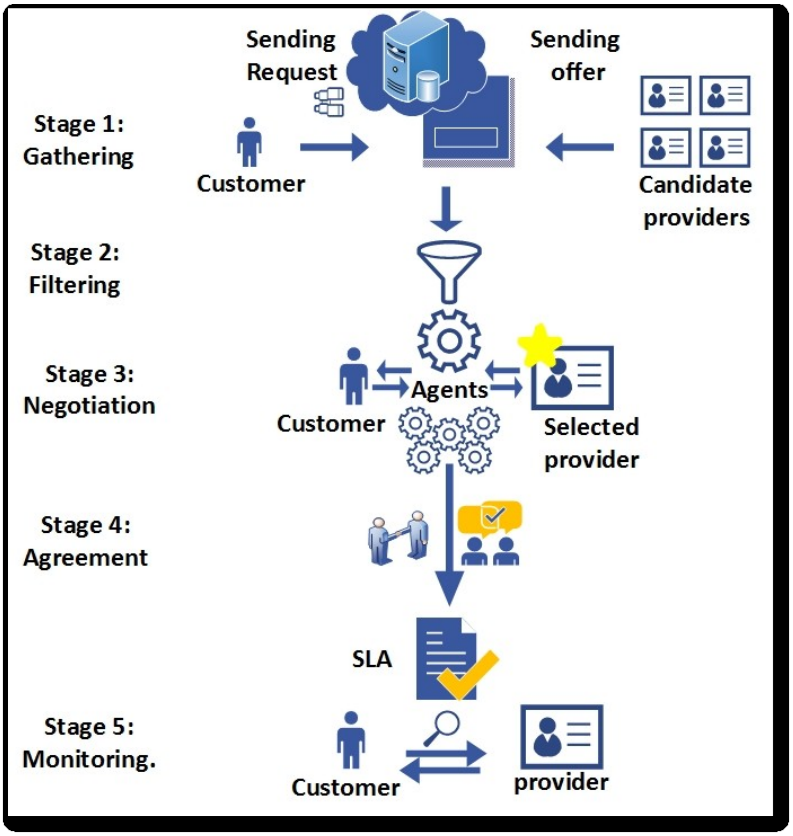


Figure 5 : high level design of framework's stages

Figure 13 shows only the workflow of the framework while figure 14 presents how stages operate and communicate with each another and in what order. First, our framework will receive providers' offers and save them in a database (knowledge). Then, the system will receive customer requests. After that, Filtering will occur to find the best match to the request to start the negotiation. After negotiation the SLA will be recommended. Finally, monitoring will start.

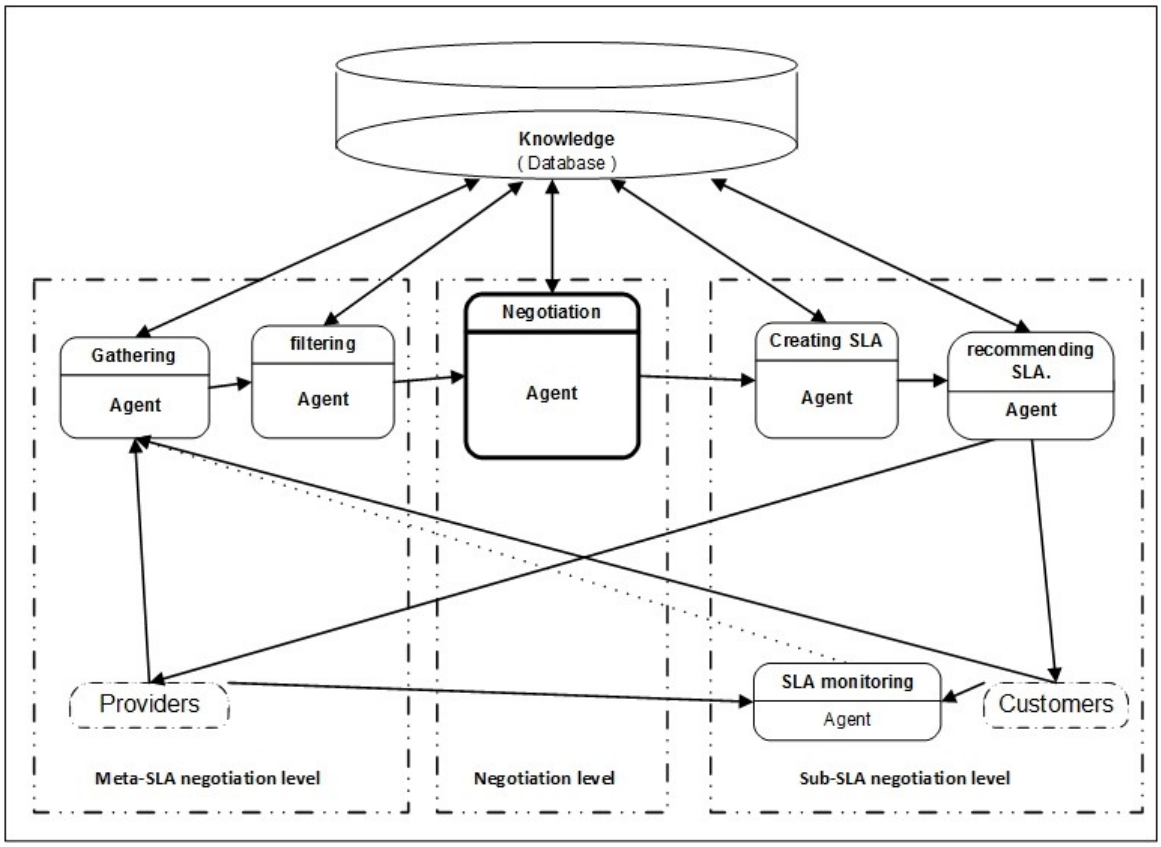


Figure 6 : high level design of the framework

5.2 Detailed design

This part of the chapter will demonstrate the detailed design for each stage.

5.2.1 Gathering and Filtering

Diagram 15 shows the gathering and filtering stage in detail. It shows each step that needs to be taken inside gathering and filtering stage. The first step is to receive the provider's offers then save the offer by sending them to the Database (knowledge). The same will be done for the customer's request; first they will be received. Then they will be saved by sending them to the Database (knowledge).

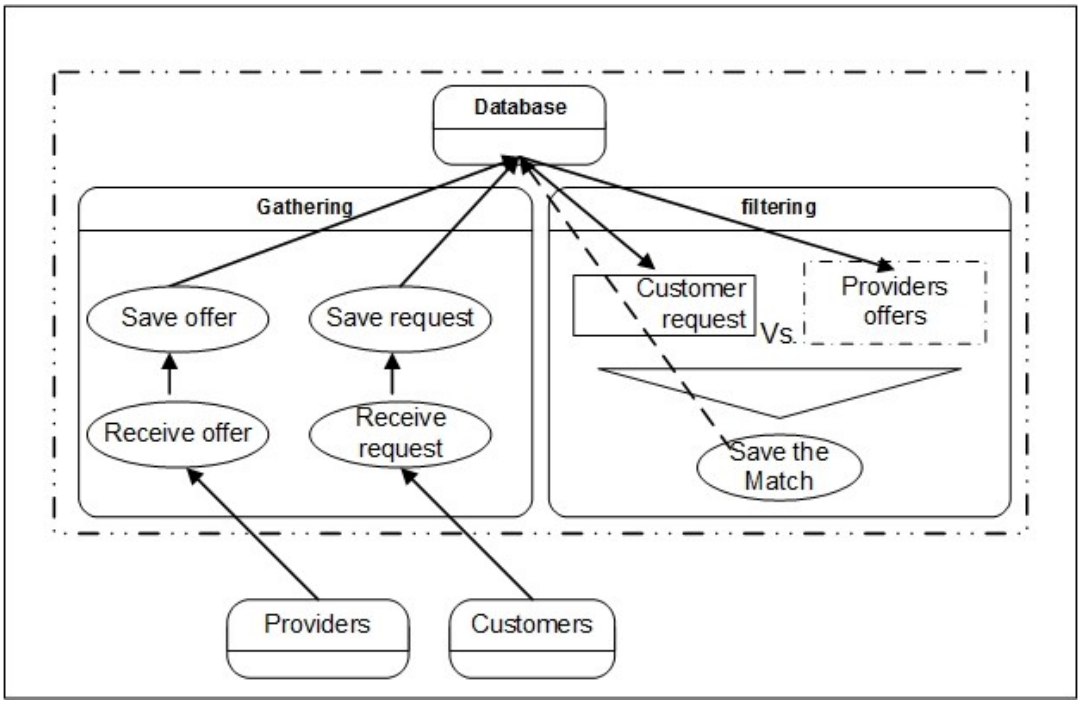


Figure 7 : detailed design of gathering and filtering stage

5.2.2 Negotiation stage

The following diagram shows the Negotiation stage in detail. It shows each step that needs to be taken inside Negotiation stage. The following diagram shows the candidate providers that have been sent by filtering stage. It also shows how the customer agent will be negotiating with each provider's agent separately. It also shows that the protocol of the negotiation session will be Rubinstein Alternating Offers Protocol. The outcome of each negotiation session will be saved in a database. The last step in this stage is to compare all negotiation session outcomes.

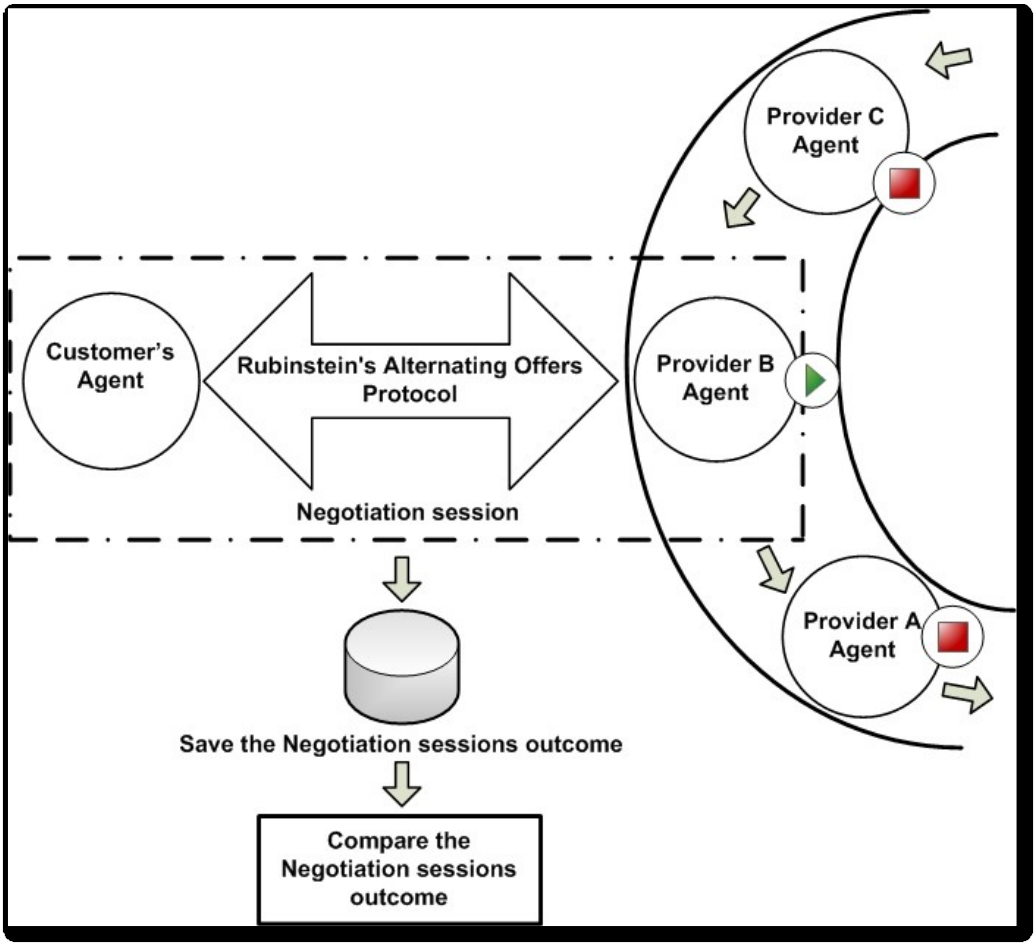


Figure 8 : detailed design of negotiation stage

5.2.2.1 Agents design

Diagram 17 shows the architecture of proposed agent. It shows the main components of our agent and how the main components communicate with each other.

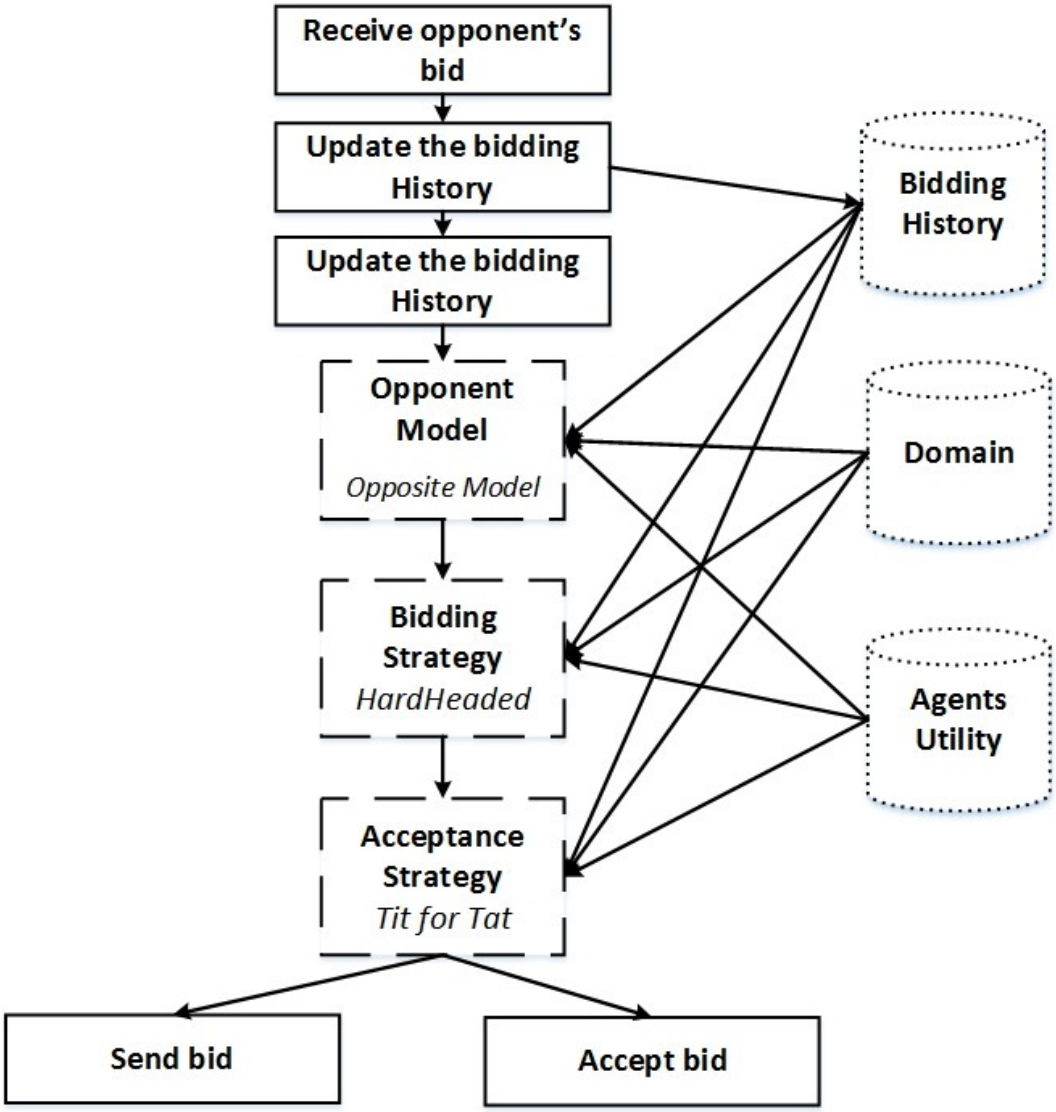


Figure 9 : Wise H-T diagram.

Once the opponent's bid is received, the bidding history and opponent model will be updated. Then, the bidding strategy defines the counter offer first, by generating a set of bids with alike preference for the agent. Second, the bidding strategy uses the opponent model to choose a bid from this set by taking the opponent's utility into account. Lastly, the acceptance strategy decides whether the opponent's bid should be accepted; if not, the bid generated by the bidding strategy is offered instead.

Our agent which I called Wise H-T is made of Bidding strategy from HardHeaded Bidding strategy and the Acceptance strategy from Tit for Tat Acceptance strategy. The Opponent model is opposite model.

5.2.2.2 Why the Bidding strategy from HardHeaded agent is being used?

At the beginning of each negotiation session, this agent will repeat the same selfish (with a high utility for its self) bids over and over again to give the impression that it is not willing to concede at all. Nevertheless, this agent is silently planning to start to be more flexible at the very end which is around the last 5% of remaining time of negotiation session. This agent does not depend on its Acceptance Strategy, instead it is hoping that the opponent will give up and accept one of its selfish offers at some point during the negation before HardHeaded start to be more flexible.

This negotiation strategy works well when a needy opponent is faced, but, On the other hand, there is high risk of ending the negotiation session with no agreement if the opponent is selfish too ,as in situation like this both sides will act competitively and not be willing to compromise to reach an agreement.

5.2.2.3 Why the Acceptance strategy from Tit for Tat is being used?

The Tit for Tat agent uses a certain type of acceptance condition called AC_{combi} , which Baarslag, Hindriks and Jonker, 2011 proved that this type of acceptance condition outperforms other acceptance conditions primarily by reaching agreements with higher utility.

The simple idea behind AC_{combi} is as follows: the opponent's offer will be accepted only if the bidding strategy plans to propose an offer that is worse (with lower unity) than the opponent's offer. However, if the time is running out, the agent will accept any offer that is not expected to improve in the remaining short time of negotiation session (Baarslag, Hindriks , Jonker, 2013) (Baarslag, Hindriks and Jonker, 2011).

Thus, by mixing-up the best part of each agent, this will lead to form an agent (Wise H-T) which will act as if it is a very hard-headed agent at the beginning of negotiation season. Hoping the opponent will give up and accept the offer. If not then Wise H-T will be more fixable and accept the opponent offer at the end. Therefore in both scenarios Wise H-T will end the negotiation will an agreement.

5.2.2.4 The Opponent model for Wise H-T.

The opponent model for this agent is the opposite model. The model simply assumes that the opponent has exactly opposite preferences to the agent itself. So for every bid x, if the agent's utility is u(x), then the estimate of the opponent utility is 1 - u(x).

5.2.3 SLA agreement.

Diagram 18 shows the SLA agreement stage in detail. It shows the each step that needs to be taken inside 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. Second step is to create the SLA then saving them to the Database (knowledge). A copy of SLA needs to be saved in the Database (knowledge) before recommending 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 SLA to provider and customer.

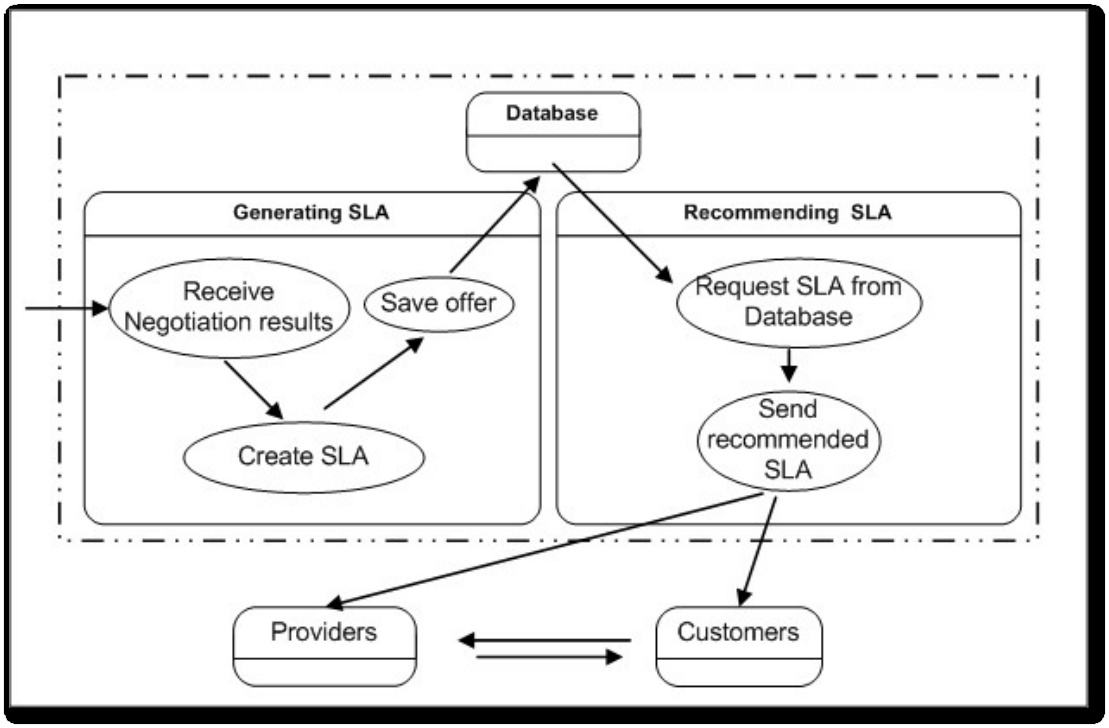


Figure 10 : detailed design of SLA agreement stage

5.3 Overall Framework Closed Loop.

Diagram 19 is the Activity diagram shows the activities inside the loop of framework.

At the beginning, the customer sends the request to look for providers. The customer will be asked to make a new request in case of no provider is found. In case of only one provider found then the customer and provider will negotiate with each other. If more than one provider is found, the customer will negotiate with each provider then the results of all the negotiation sessions will be compared. After that, if both sides agreed to make a deal, then a SLA will be made. At the monitoring stage, the real-live data will be gathered. If a breach is detected, actions will be taken.

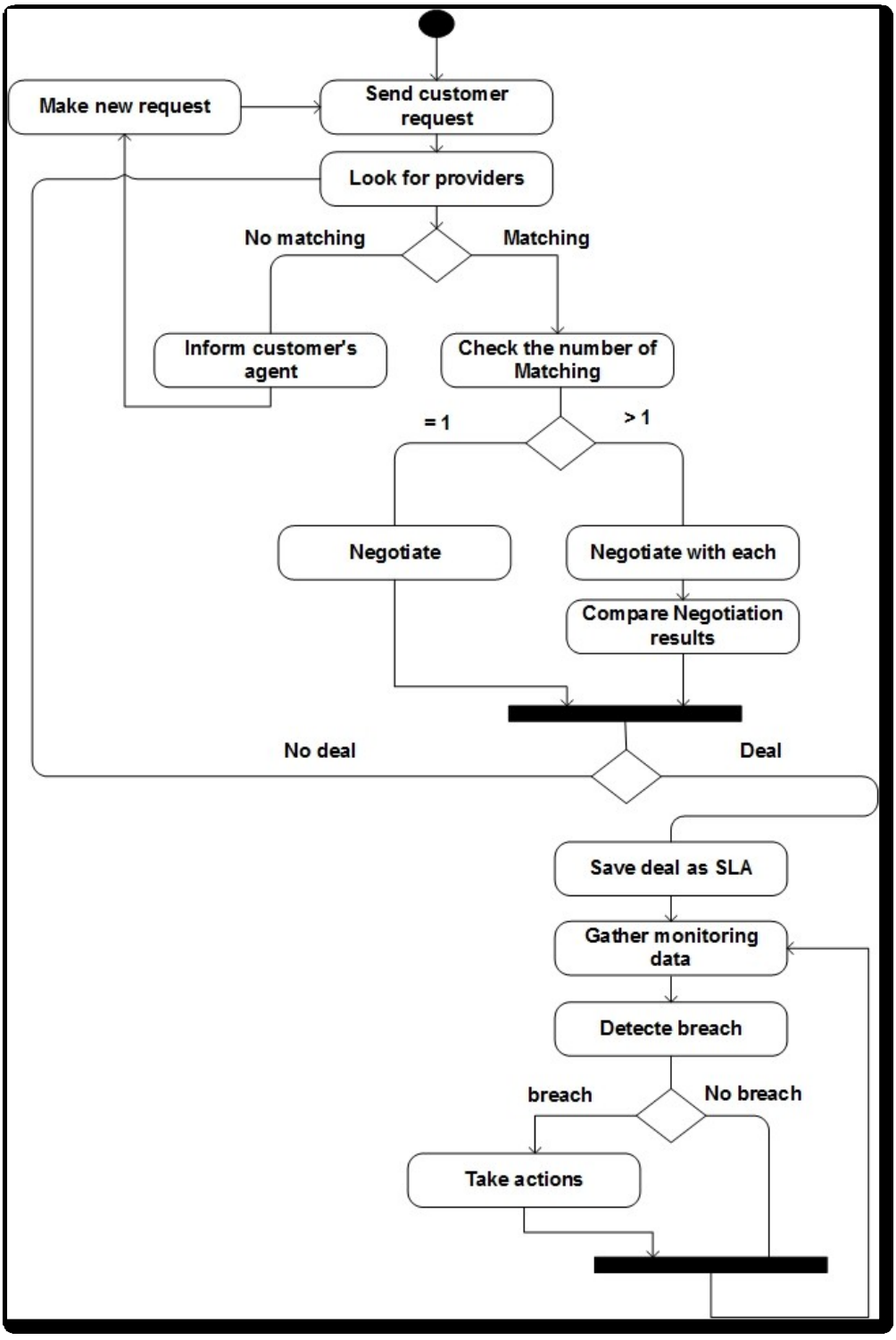


Figure 11 : Activity diagram of the framework

5.4 Conclusion

The main components of the proposed framework will be: gathering, filtering, negotiation, SLA agreement and monitoring. This chapter presented a high level design of the framework. The high level design shows the workflow of the framework where on the top gathering stage is getting providers offer and customer request. Then under the gathering stage is the filtering stage where only the selected candidates of the providers are passed to the next to stage which is Negotiation stage. After completing the negotiation stage, the outcome of the negotiation will be passed to the next stage which is the Agreement stage. Then on the bottom is the monitoring stage. Also, this chapter presented a detailed design for each stage. Finally, the overall framework closed loop is presented. In next chapter, first I will present the design of my solution for monitoring stage. After that, I will present the implementation of monitoring stage via a scenario of over-promising and under-delivering problem.

CHAPTER 6

Monitoring Stage

In last chapter, a high level design of the framework as well as a detailed design for gathering, filtering, negotiation and SLA agreement stages is presented. Finally, the overall framework closed loop is presented. In this chapter, first we will present the design of our solution for monitoring stage. After that, I will present the implementation of our monitoring stage solution via a scenario of over-promising and under-delivering problem. Finally, I will demonstrate the idea of virtual machine migration using cloud computing simulation known as CloudSim.

6.1 Our proposed solution:

First I will present the design of our solution. After that, I will present the implementation of our solution via a scenario of over-promising and under-delivering problem.

6.1.1 Monitoring Stage Design.

Monitoring stage created by following the vision and the principles of autonomic computing (Dobson, et al., 2010) which will give this stage the four self *- properties of autonomic computing: Self-Healing, Self-Configuration, Self-Optimization and Self-Protection. Each one can be defined as “Self-Configuration is the ability of the system to perform configurations according to pre-defined high level policies and seamlessly adapt to change

caused by automatic configurations. Self-Optimization is the ability of the system to continuously monitor and control resources to improve performance and efficiency. Self-Healing is the ability of the system to automatically detect, diagnose and repair faults. Self-Protection is the ability of the system to pro-actively identify and protect itself from malicious attacks or cascading failures that are not corrected by self-healing measures". (Khalid, et al., 2009). For Monitoring stage, the autonomic control loop has been followed to design our SLA monitoring loop.

Figure 20 shows how the autonomic control loop (Collect, Analyze, Decide and Act) (Dobson, et al., 2010) is used to design and implement our solution for monitoring stage.

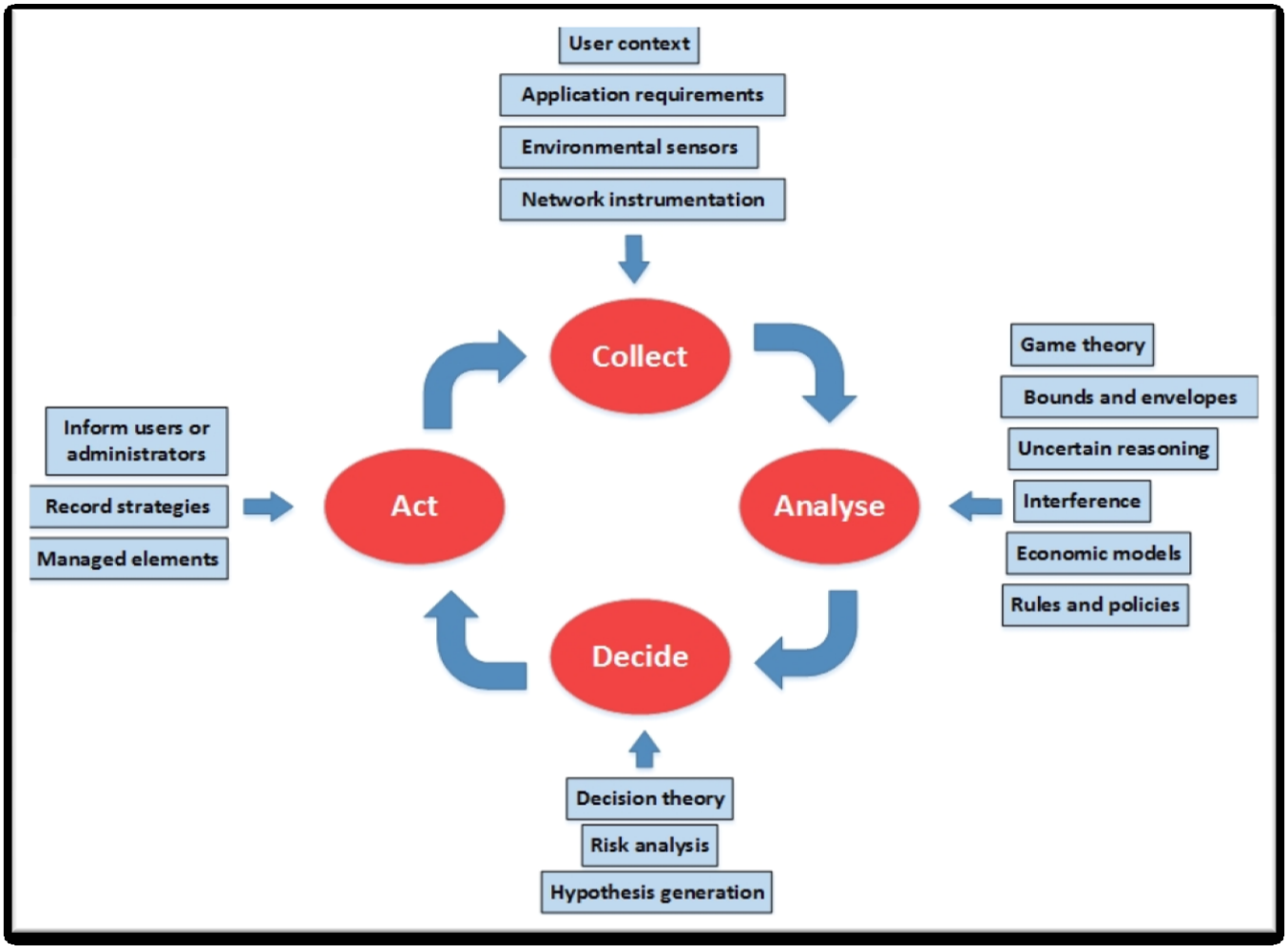


Figure 12 : The autonomic control loop (

The figure 21 illustrates detailed design of monitoring stage. It shows how the autonomic computing control loop (Collect, Analyze, Decide and Act) is used to design the Monitoring Stage.

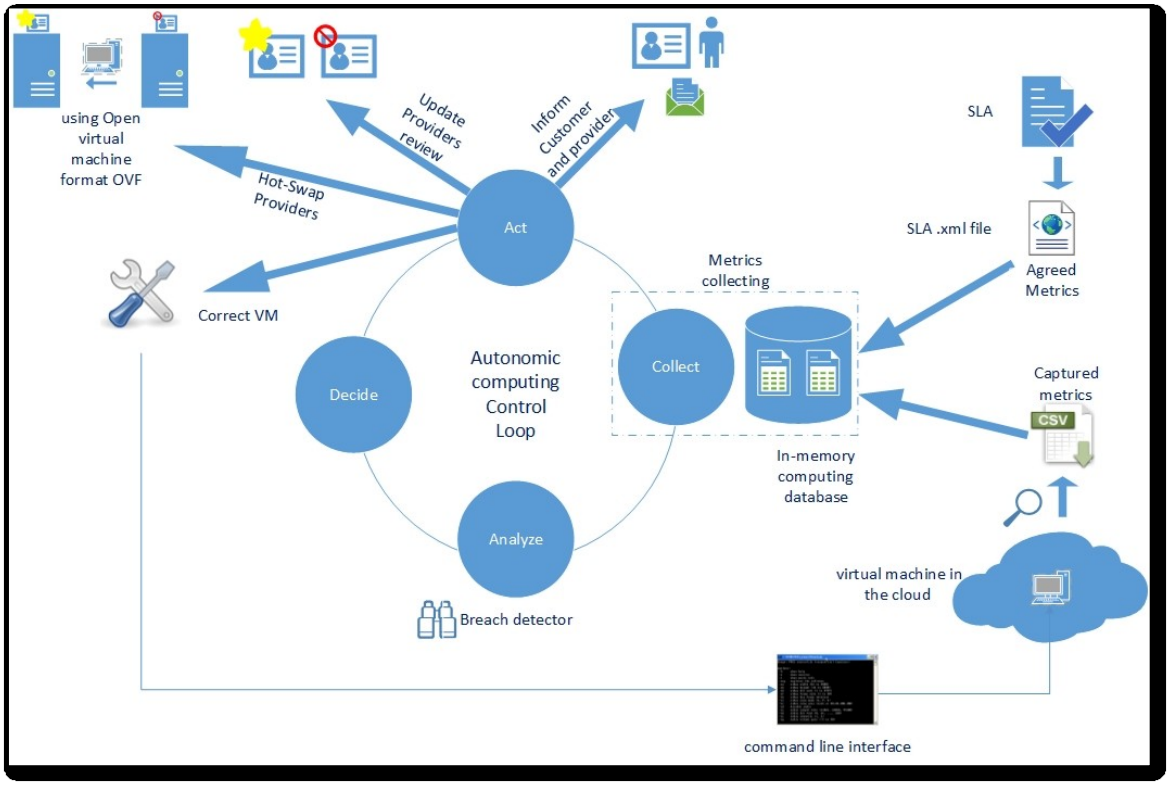


Figure 13: detailed design of Monitoring Stage.

From figure 21, it can be seen that in the first phase of autonomic monitoring control loop which is collect, the agreed metrics will be extracted from SLA document then sent to a database. At the same time, the gathered metrics from the monitoring will be sent to the same database. In the second phase of autonomic monitoring control loop which is analyse, the breaches and violations will be detected by comparing the gathered metrics against the agreed value in the SLA.

In the third phase of autonomic monitoring control loop which is Decide, recommendations of the suitable actions that can be taken will be passed to the next phase which is Act. The Act will be responsible of taking an action in case of SLA violation detected.

6.1.2 Monitoring Stage Implementation.

At the beginning of this section, the first stage of autonomic control loop which is *collect* will be discussed. After that, *Analyze*, *Decide* and *Act* will be discussed.

6.1.2.1 Collect

In this part, I would like to demonstrate the idea of using an agent to perform the *collect* phase. This agent will be running on the customer side in order to perform the following:

- To collect SLA metrics in real time. The metrics can be CPU, memory, I/O Bandwidth.
- To save the gathered metrics in CSV file.
- To upload the gathered metrics to the cloud (e.g. dropbox). Thus, in this way the gathered metrics file will be available for the monitoring server agent which will be explained next.

For a Real-time metrics collecting, a batch file is made to gather all information about the VM in real-time. This agent will keep running in the Client VM background.

The figure 22 shows the a batch file with a loop , it will run (systeminfo.exe) which is Command-line program that can "Displays detailed configuration information about a computer and its operating system, including operating system configuration, security information, product ID, and hardware properties, such as RAM, disk space, and network cards "

```
1 @echo off
2 cls
3 :start
4 Systeminfo.exe /FO csv /NH >> C:\xampp\htdocs\aa.csv
5 goto start
```

Figure 14 : Monitoring Client Agent

6.1.2.2 Analyse

In this part, I would like to demonstrate the idea of using an agent to perform the *Analyse* phase. This agent will be running on the server side, which can be independent third party between the customer and the provider. This agent will be responsible for the following:

- To read the gathered metrics file that made by collect phase.
- To detect the breach and violation by comparing the gathered metrics against the agreed value in the SLA.

Figure 23 and figure 24 shows how for demonstrating the agent, I implemented the following using a PhP programing language; this agent will do the following tasks:

1- To read the CSV file which made by the Client Agent.

2- To compare between the captured value and the agreed value in SLA.

3- To highlight in red the breach.

4- Calculate the number of SLA breaches. (In case it needs it)

5- Keep reloading every 1 second for real time monitoring.

```
<!DOCTYPE html>
<html>
<head>
<META http-equiv="refresh" content="1;URL=http://localhost/csv5.php">
</head>
<body>


<?php

$L = 1300;
$D = 0;

$L = intval($L);

echo "<h1> SLA is : $L </h1>";

$file_handle = fopen("aa.csv", "r");

while (!feof($file_handle) ) {
```

Figure 15 : Monitoring Server Agent code 1

```
21
22  $line_of_text = fgetcsv($file_handle, 1024);
23
24  $data_to_number = $line_of_text[23];
25  $data_to_number = str_replace(array(" ",",","MB"),"", $data_to_number);
26  $data_to_number = intval($data_to_number);
27
28  if ($data_to_number < $L )
29    {
30       echo "<font color=#ff0000> $data_to_number </font>" ;
31       $D = (($L - $data_to_number)*100)/ $L ;
32
33       echo str_repeat(".",1);
34       echo " percentage of breach is =" ;
35       echo number_format("$D",1);
36       echo "%";
37       echo "<br>";
38    }
39    else
40   {
41     echo "$data_to_number <BR>";
42   }
43  }
44  fclose($file_handle);
45  ?>
46  </body>
47  </html>
```

Figure 16 : Monitoring Server Agent code 2

6.1.2.3 Decide and Act.

In this part, I would like to demonstrate the idea of using an agent to perform the *Decide* and *Act* phase. This agent will responsible for the following:

A. Recommend suitable actions that shall be taking, in case of violation detected (*Decide).*

B. Perform the recommended action (*Act*).

Generally, in the SLA, the customer and provider agree of the penalties and actions in case the SLA will be violated and unfulfilled. There is a variety of actions, like informing both sides,

recommending solutions, elf-healing and negotiation. Each of these actions will be described below:

- **Informing customer and provider.** It is important to alert the customer and the provider about any violations, so that they can take an action themselves if they choose not to activate the automated correction. The customer and provider can be informed in different ways, like emails, SMS.
- **Recommending solutions**. Based on the scenario of violation, some recommendations will be given. Given recommendations are important in case of provider and customer choosing to receive only recommendations without giving the monitoring loop the permission to perform automated correction.
- **Self-healing.** If customer and provider gives the permission for the monitoring loop to perform the automated correction then it will be done automatically. There are different types of correction that can be taken like: "hot adding" or "auto-scaling" extra resource or "hot swapping" between providers.
- **Negotiation.** Customer and provider can agree to negotiate when violation detected. The negotiation can be about types of correction that shall be performing or kinds of compensation/ penalties.

Next I will demonstrate *Decide* and *Act* via a Scenario (Over-Promising and Under-Delivering Problem).

6.1 Scenario (Over-Promising and Under-Delivering Problem).

I will propose some solutions to solve one of the most complex problems that cloud providers usually face and cloud customers afraid of. Typically, the cloud providers try to increase their profit by promising all the clients to have "unlimited" resources which will be available on-

demand in seconds after requesting the resources. The cloud providers expect that it is unlikely that all clients will request extra resources in the same time. However, this might and often happen. So, what is the best solution if clients request extra resources more than the provider capacity? Here, I will try to propose some solutions for this complex problem

A: Satisfying some clients and compensating the others.

The first solution that I propose is that to satisfy some clients and compensate the others. This solution is not the best but it can protect the provider from the worst which is disappointing all the clients. For that, the provider needs to decide which client shall be satisfied. This is a tricky task. One of the ways to decide which client shall be pleased is to find out which client will cost less by unpleasing him/her. On the other world, the provider needs to find out which client that is less important to the provider for the long-run. There some factors that I propose to find how important the client is:

1: The contract length.

How long that client has been with this provider? And how long this client is going to be with this provider? The length of the contract can be use to tell how important is that client.

2: The penalties.

Before selecting which client that shall be given that extra resource, provider needs to assess penalties that need to be paid for each unhappy client.

3: Reputation.

The provider needs to assess how his/her reputation will be ruined. Then pick the client that will be ruining the reputation the less. This can be done by answering some questions like; is this client has been referred by existing client? Is this client work in media?

B: Requesting more resources from other provider:

The second solution that I am proposing is that the provider shall be ready for this kind of problem by being a client for another provider acting as broker. So the provider automatically request extra resources for other provider.

C: Migrations.

The third solution that I am proposing is live virtual machine migration [Violeta M and Juan Manuel, 2014]. I will demonstrate the idea of virtual machine migration using cloud computing simulation known as CloudSim. [Calheiros., et al, 2011]. I will demonstrate via 3 scenarios. In first scenario, there are 2 providers and 2 customers. In the second scenario, there are 2 providers and 15 customers. **.** In third scenario, there are 5 providers and 40 customers.

C.1. <u>First Scenario:</u> 2 hosts and 2 customers:

Figure 25 demonstrates the scenario. In this scenario, the assumptions are that there are 2 providers. Both of the 2 providers use the same datacenter, so the providers act as cloud

brokers. The 2 providers use cloud hosts/ hypervisor to host the customers' virtual machine. Host0 represent the first provider. Host1 represent the second provider. In this scenario will be 2 customers.

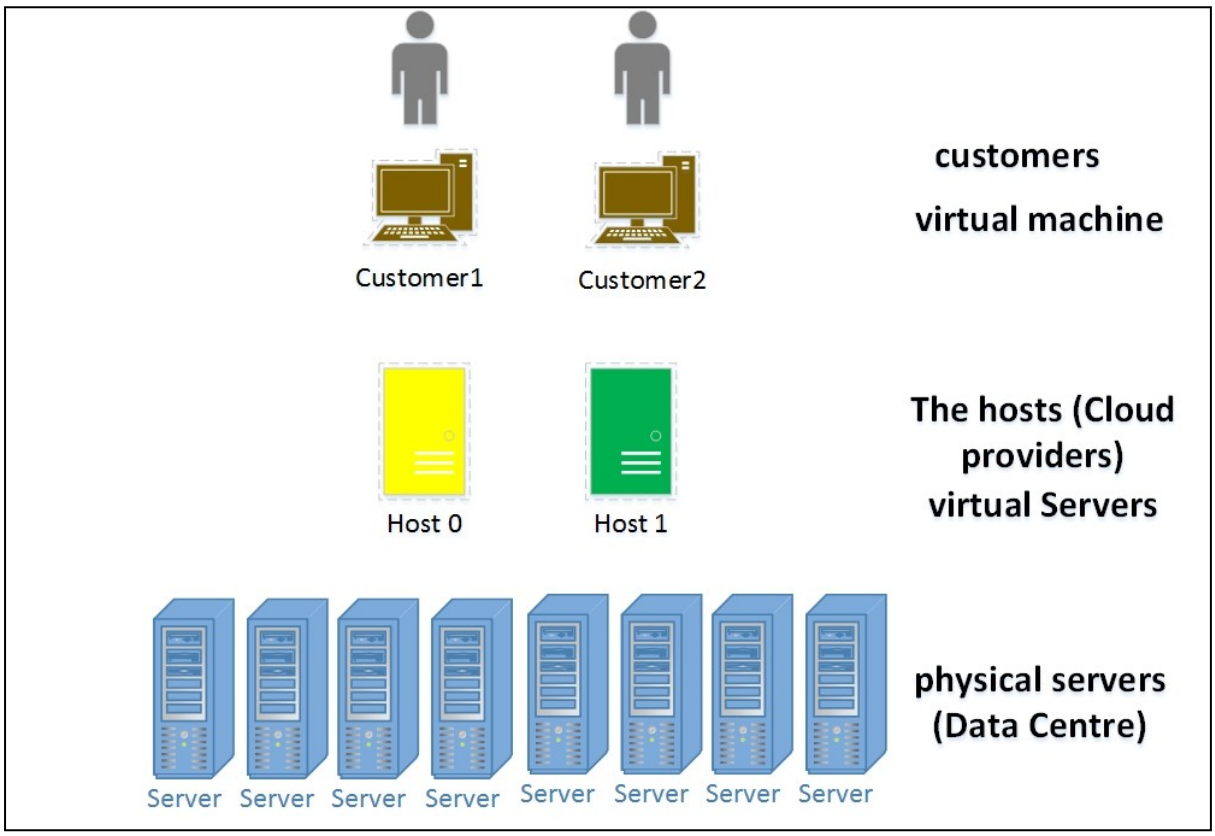


Figure 17 : First Scenario

From figure 26, it can be seen that after running the simulation, I have found out that one migration occurred; this is because the resources in host0 were not enough for customer1 need. Figure 26 shows that after 3mins customer1 migrated to host1.

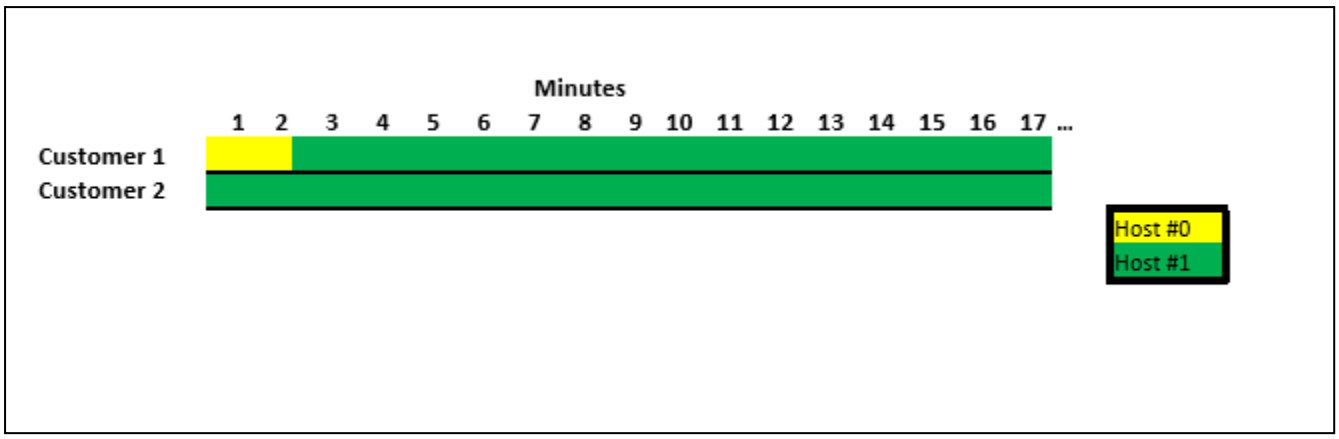


Figure 18 : Migration for the first scenario

The figure 27 shows how we set up the providers hosts. Host0 got only 10000MB. Host1 got 20000MB.

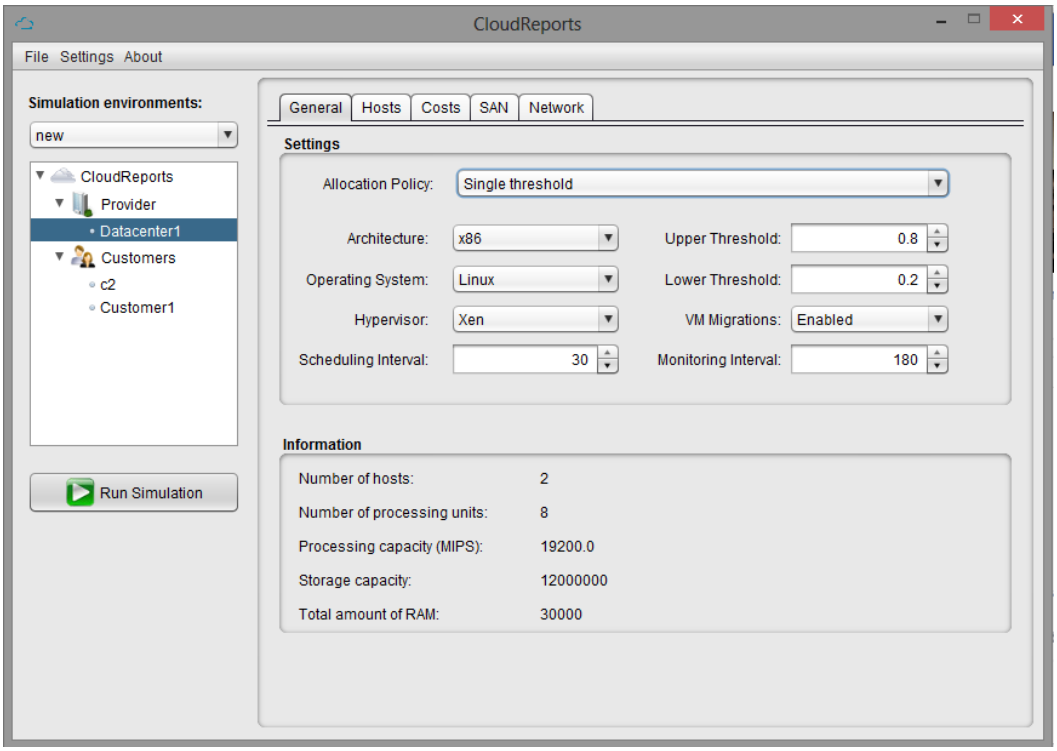


Figure 19 : Provider 2 cloud hosts

There are 2 customers each of them needs 5120 MB to run the virtual machine. Figure 28 shows more information about the virtual machine.

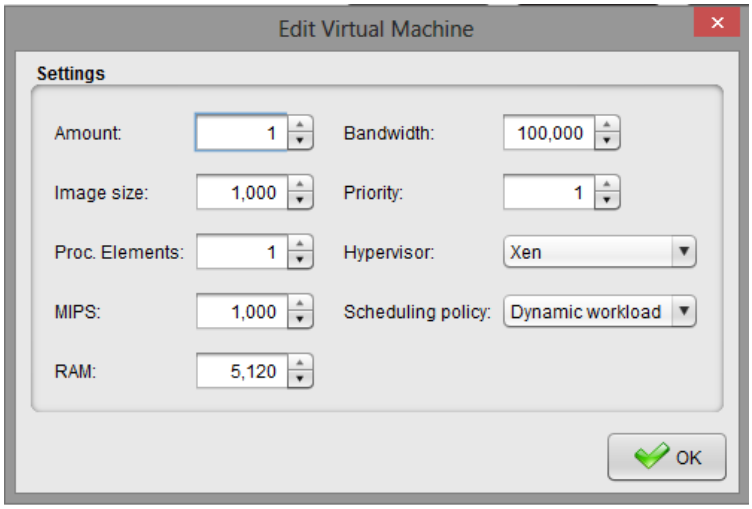


Figure 20 : Customers VM

Figure 29 shows the resource utilization for host#0 after running the simulation,

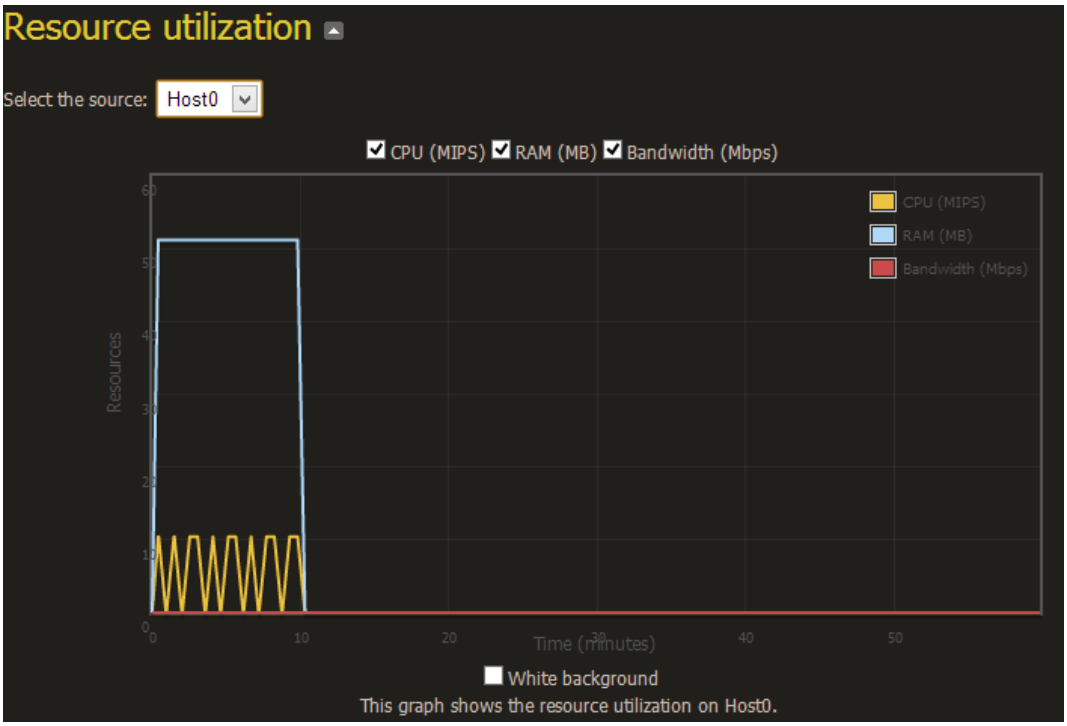


Figure 21 : Host #0 resource utilization

Figure 30 shows the resource utilization for host#1, after running the simulation,

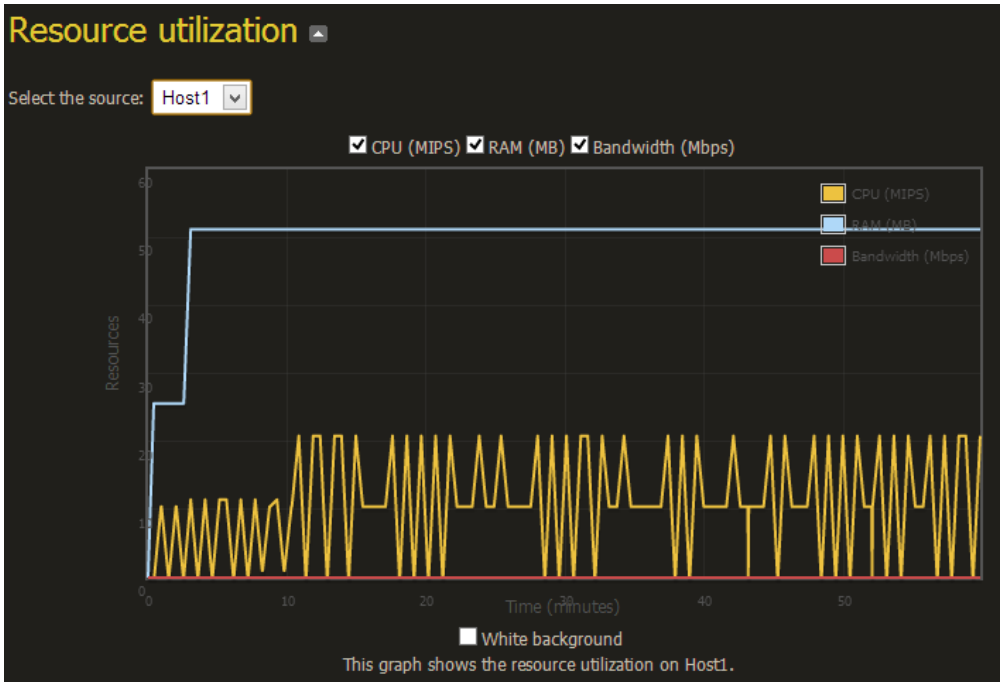


Figure 22 : Host #1 resource utilization

C.2. Second scenario is 2 hosts and 15 customers

Figure 31 demonstrates the scenario .Similar to the first scenario; there are 2 hosts/ hypervisors. But in this scenario, there are 15 customers /virtual machines. Host0 represent the first provider. Host1 represent the second provider. The memory in Host0 is only 5000MB. But, the memory in Host1 is 40000MB.

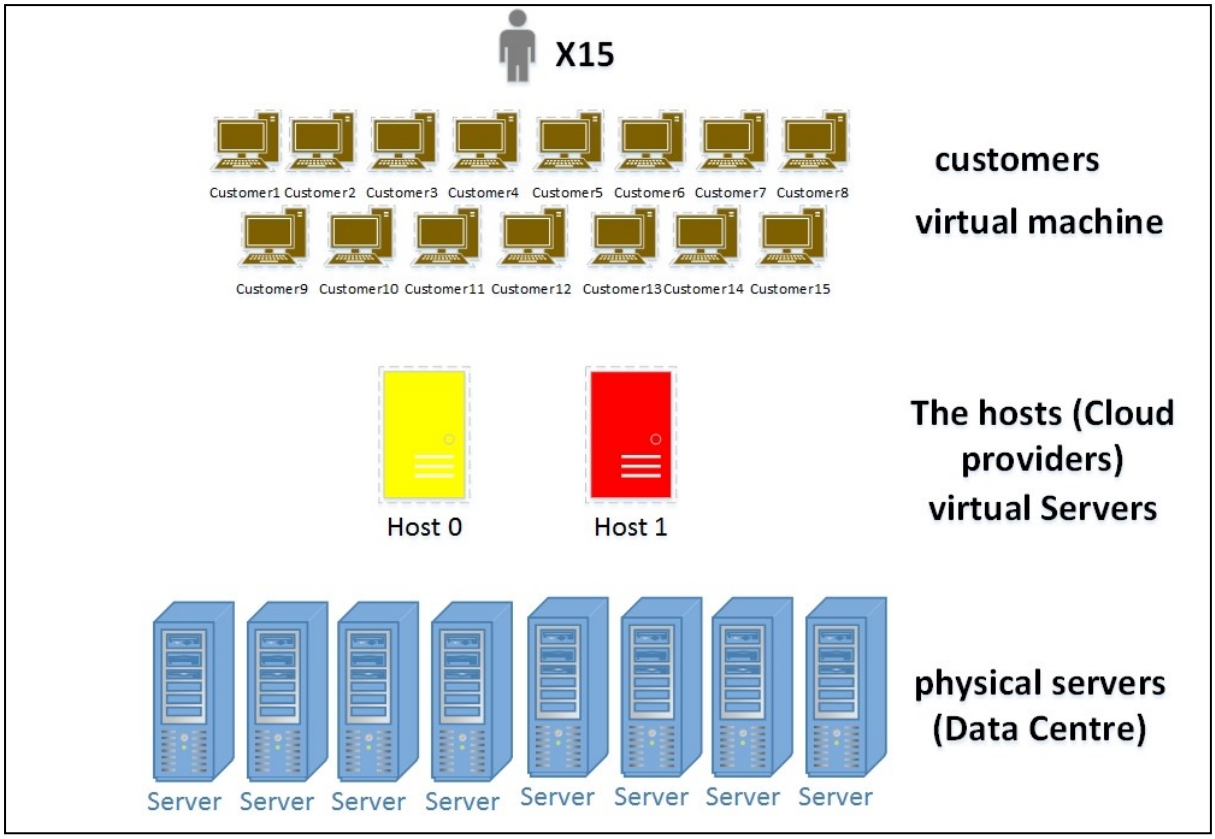


Figure 23 : the second Scenario

Figure 32 shows the 6 migrations that occurred after running the simulation. The migrations happen to customers' number 3, 4,5,6,8 and 9.

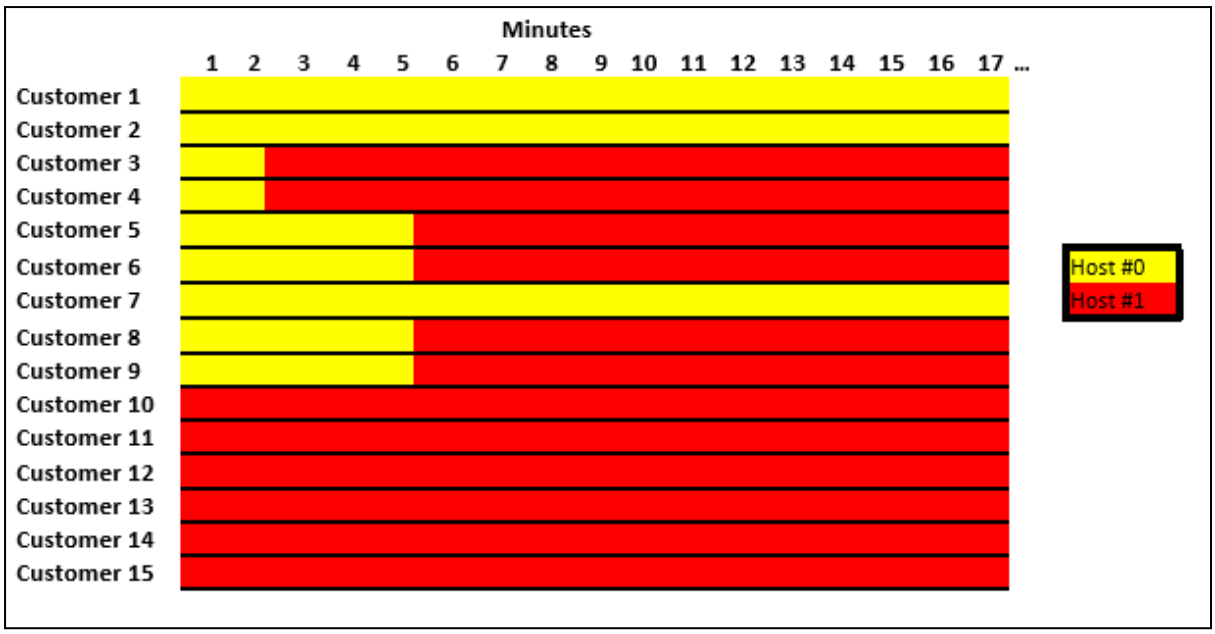


Figure 24 : the 6 migrations for the second scenario.

Figure 33 shows the scenario setup. I created 2 hosts and 15 customers.

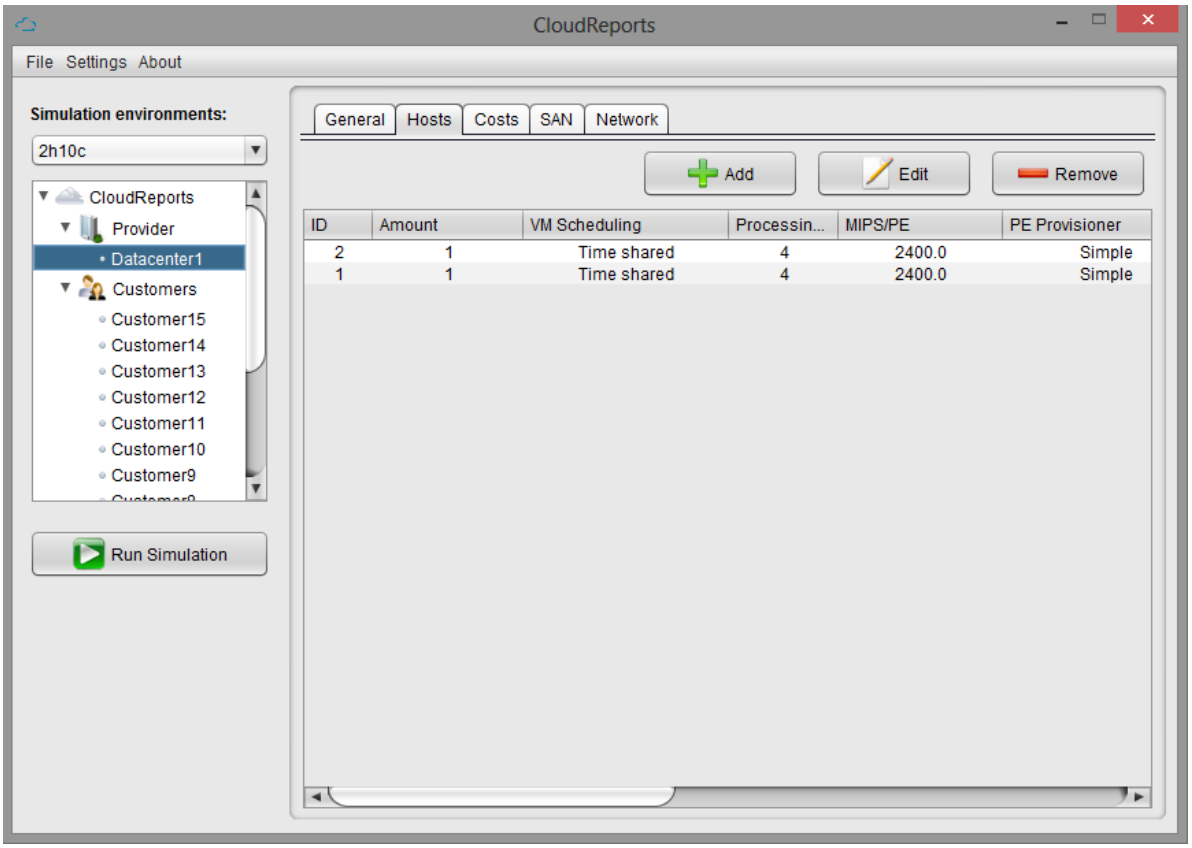


Figure 25: Scenario setup

The figure 34 shows the customers' virtual machines details.

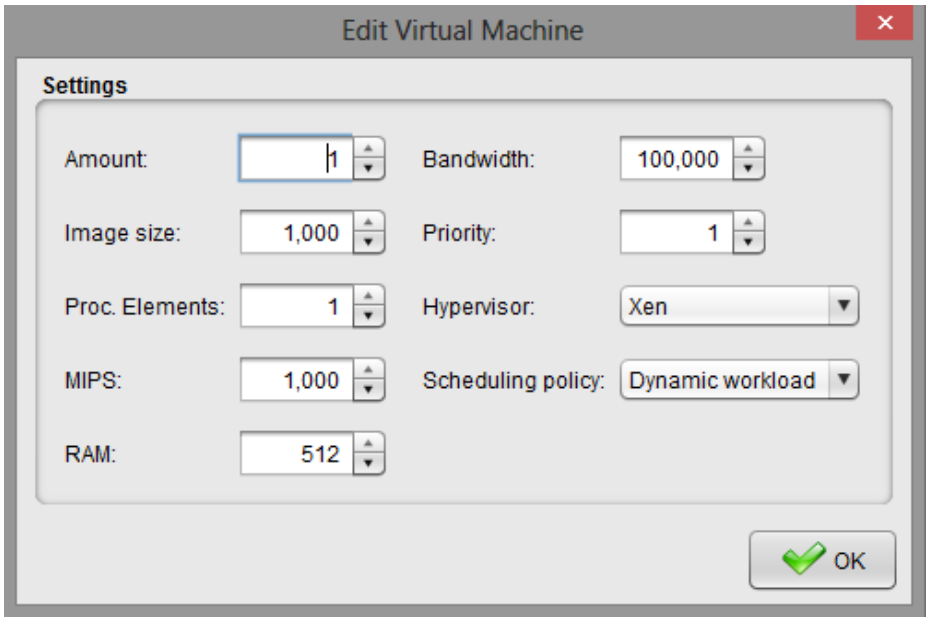


Figure 26: customers' virtual machines (Second scenario)

Figure 35 shows the host1 details.

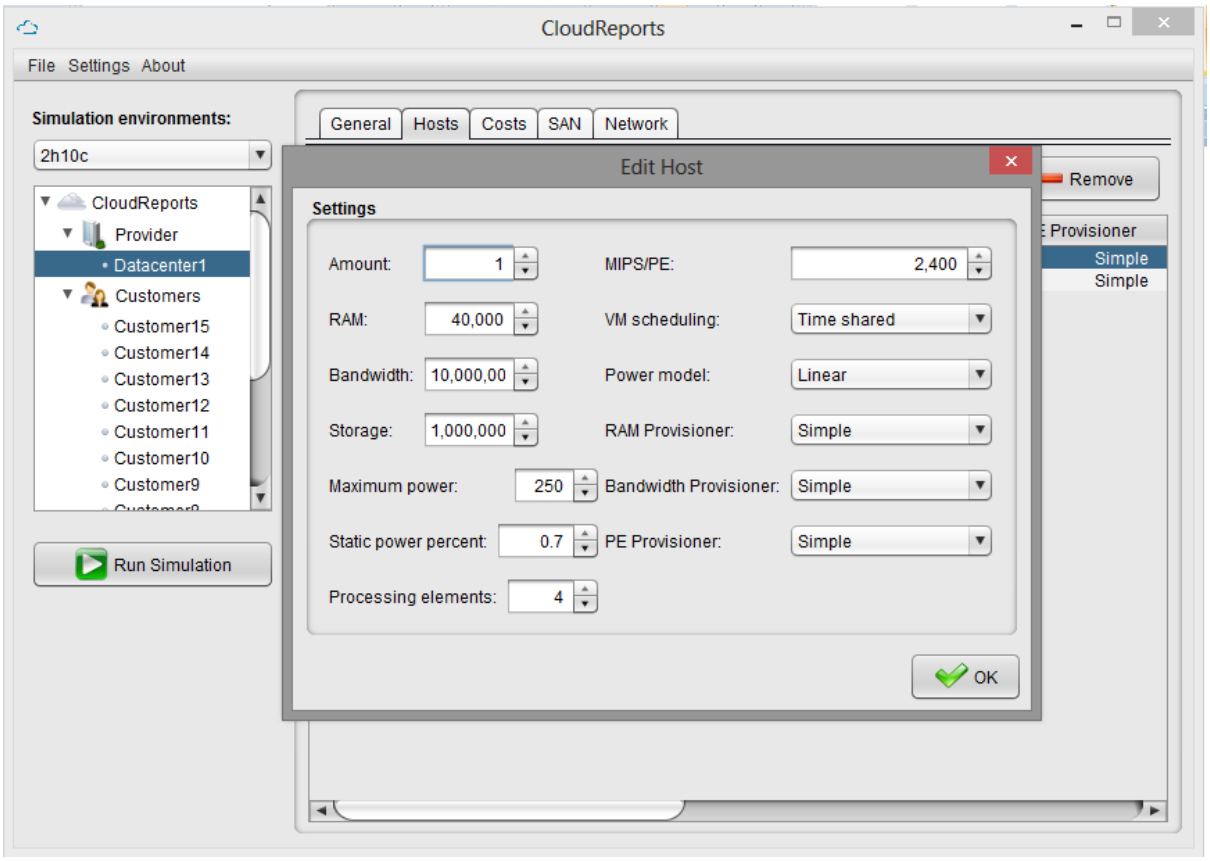


Figure 27 : second host setting (Second scenario)

Figure 36 shows the resource utilization for host#0, after running the simulation,

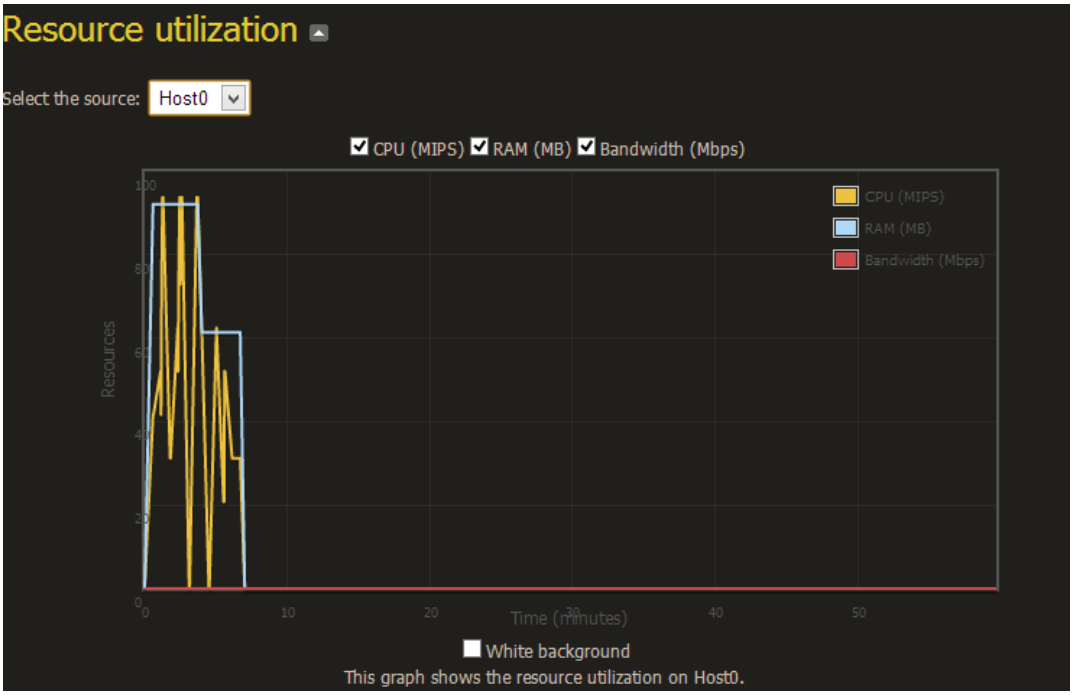


Figure 28: resource utilization for host0 (Second scenario)

Figure 37 shows the resource utilization for host#1, after running the simulation,

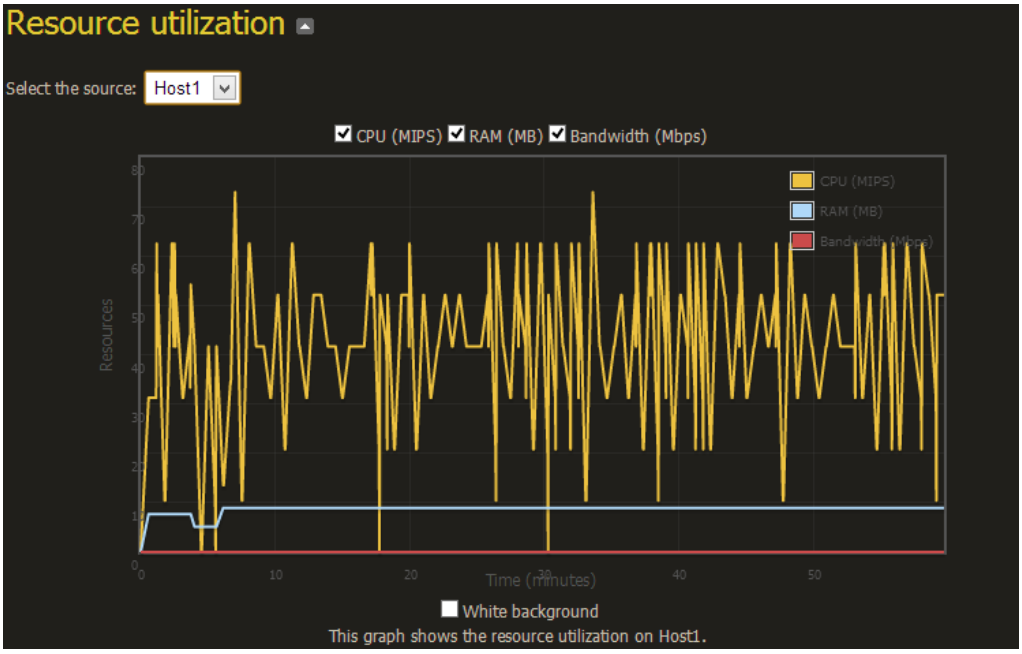


Figure 29 : resource utilization for host1 (Second scenario)

C.3. Third scenario is 5 hosts and 40 customers.

In this scenario, the assumptions are that there are 5 hosts/ hypervisors/providers and 40 customers. Figure 38 demonstrates the scenario.

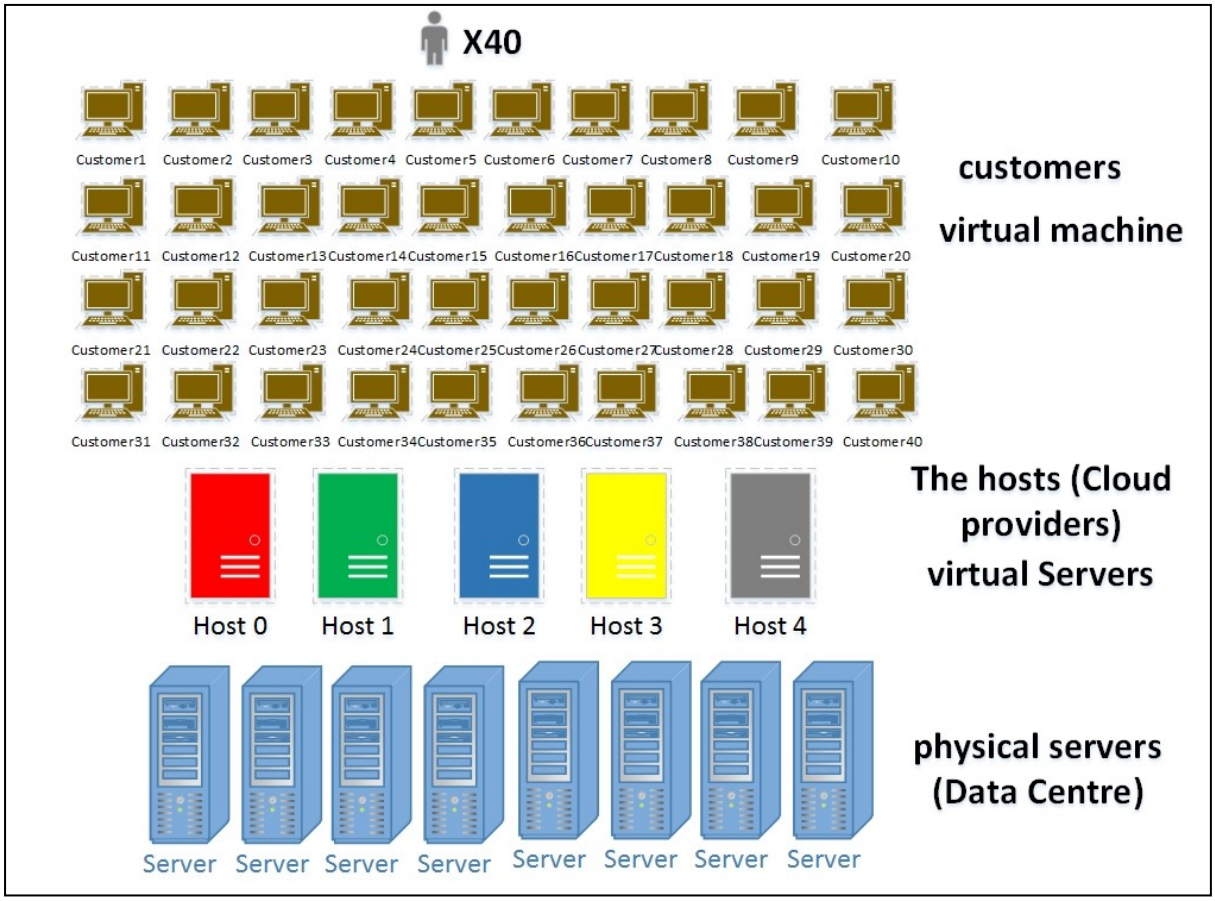


Figure 30 : Third scenario

Figure 39 shows the 11 migrations that occurred after running the simulation. The figure also show when each migrations took place.

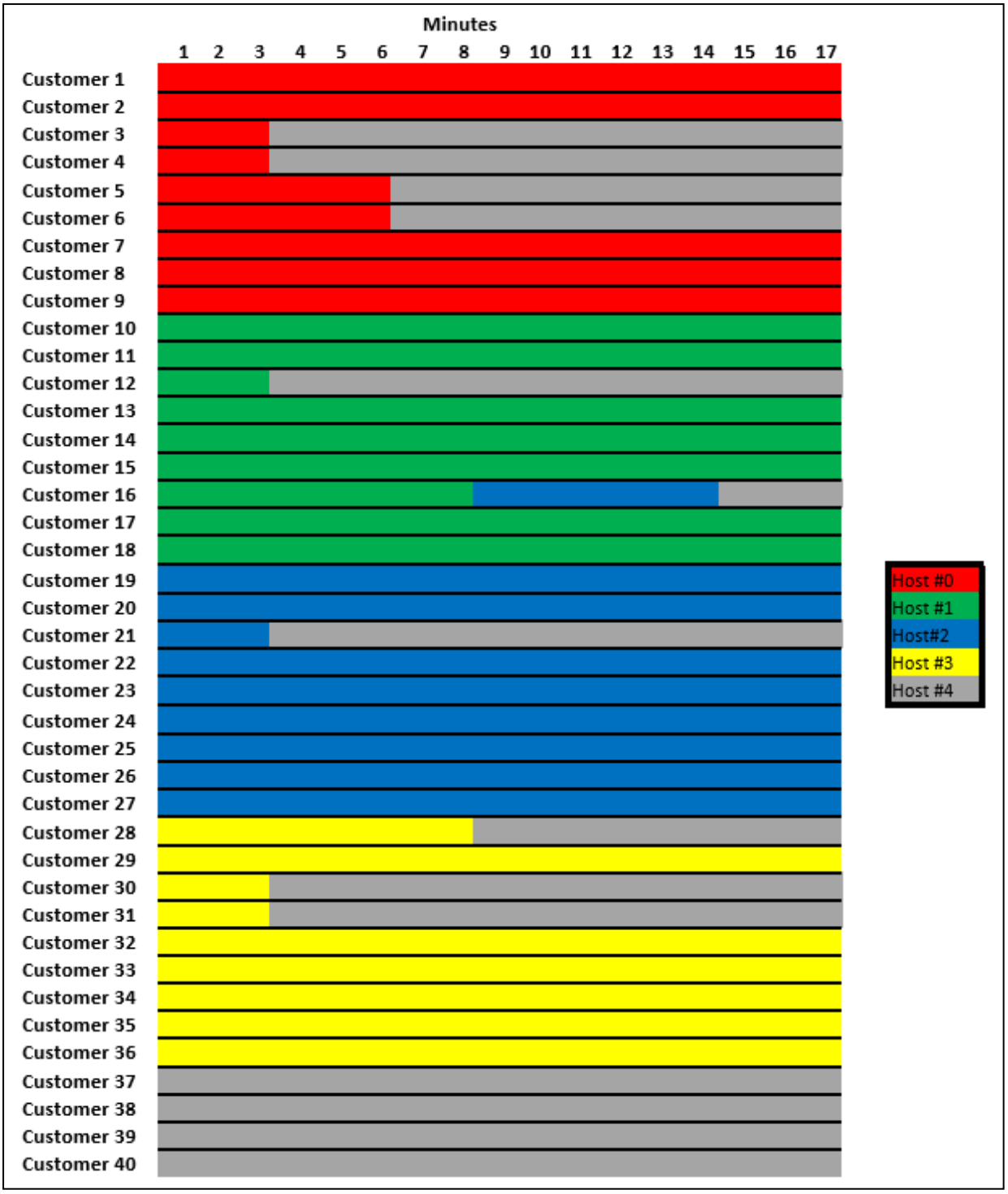


Figure 31: the 11 migrations for the third scenario

Figure 40 shows the hosts setting.

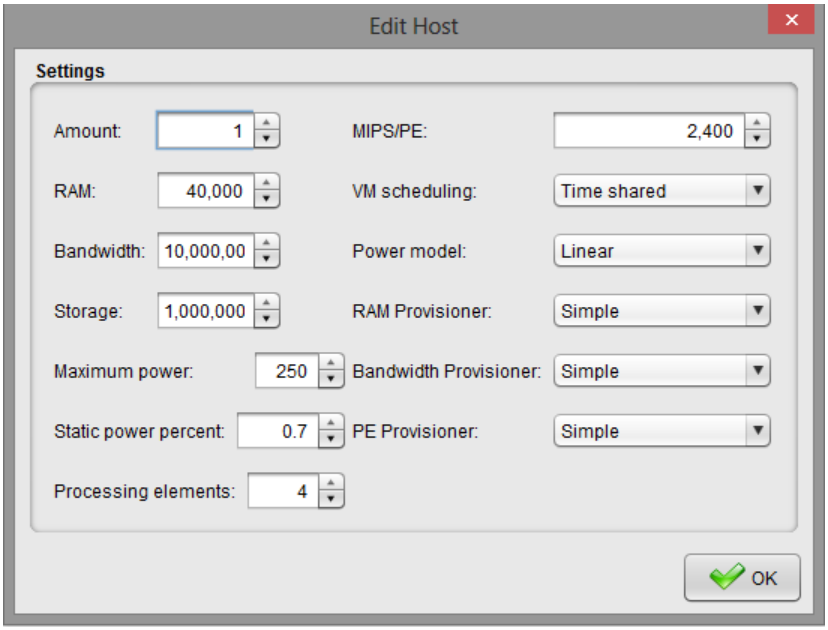


Figure 32 : Hosts setting (third scenario)

Figure 41 shows the customers VMs setting.

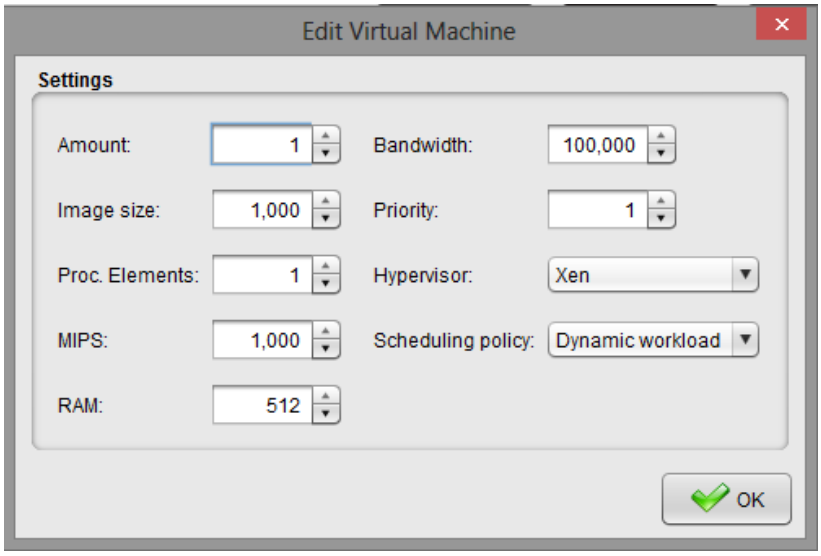


Figure 33 : customers VMs setting (third scenario)

After running the simulation, figure 42 shows the resource utilization for all providers.

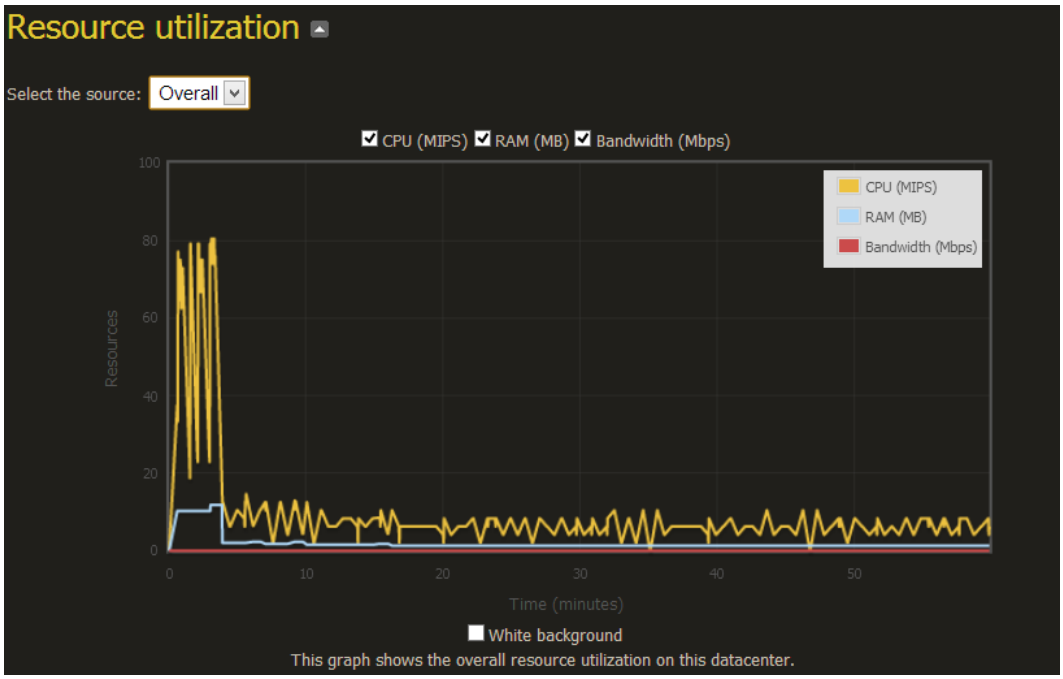


Figure 34 : resource utilization on this datacenter (third scenario)

Figure 43 shows the resource utilization for host#0.

Figure 35 : resource utilization on Host0. (third scenario).

Figure 44 shows the resource utilization for host#1.

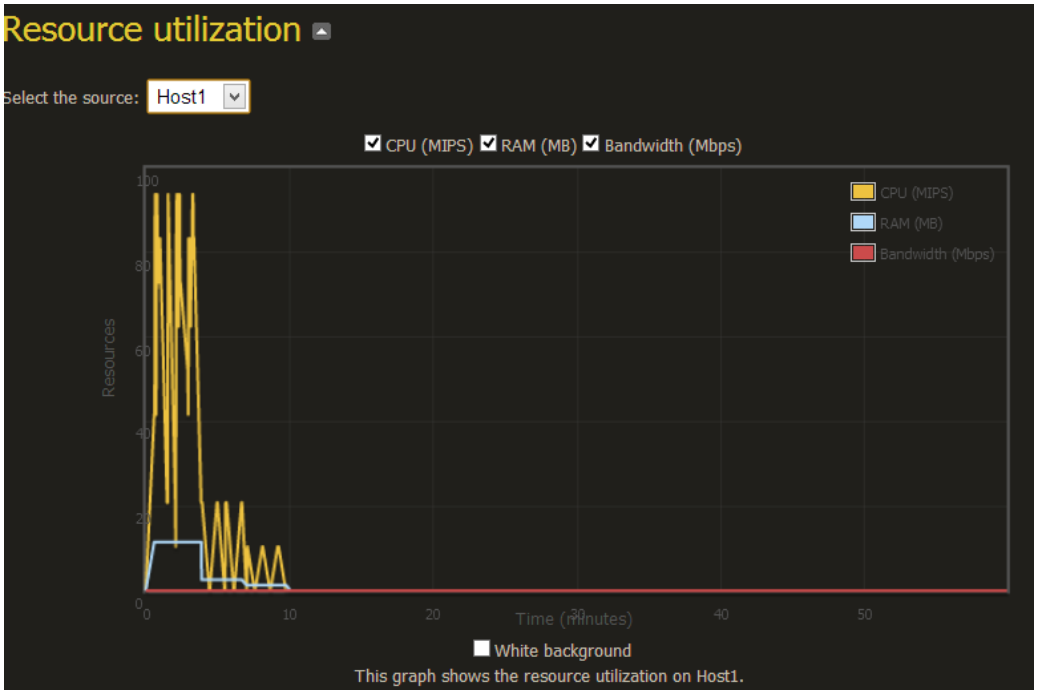


Figure 36 : the resource utilization on Host1 (third scenario).

Figure 45 shows the resource utilization for host#2.

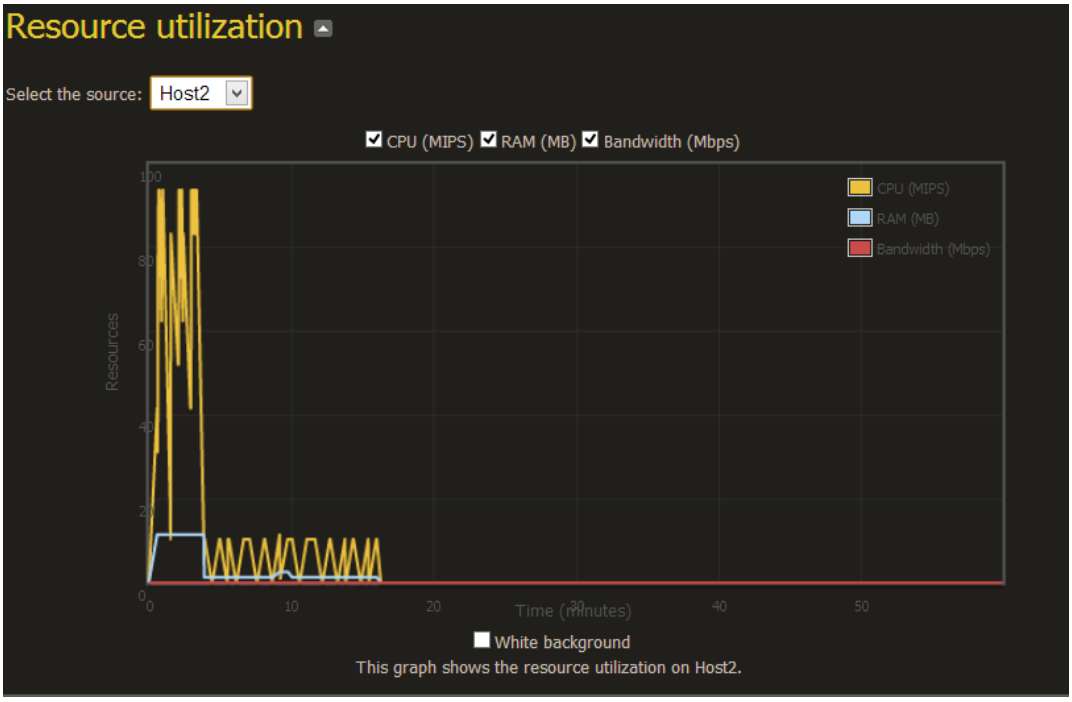


Figure 37 : the resource utilization on Host2 (third scenario).

Figure 46 shows the resource utilization for host#3.

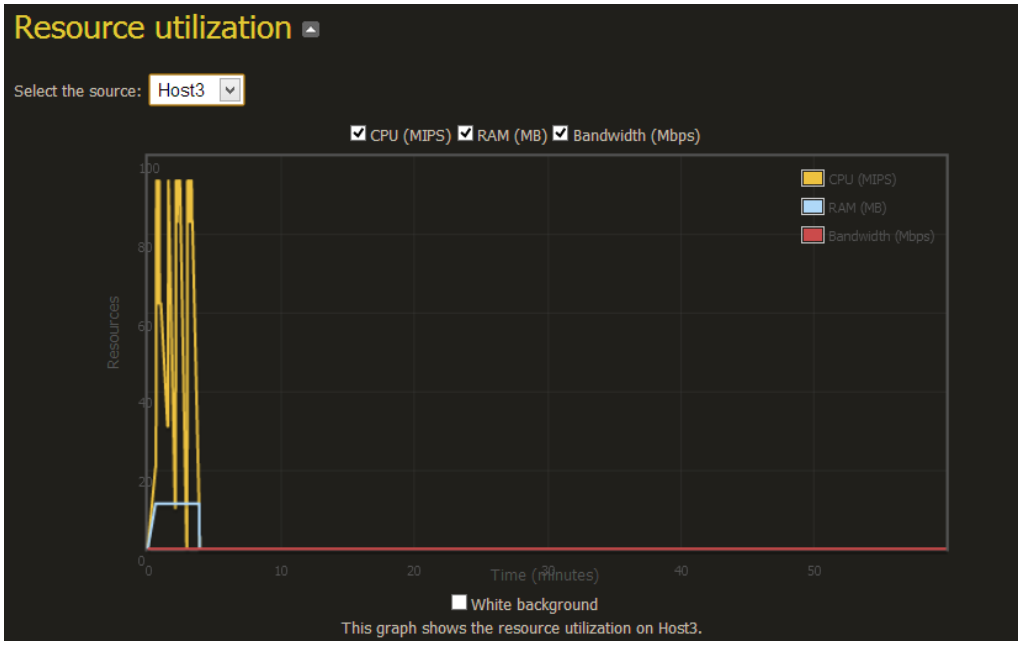


Figure 38 : the resource utilization on Host3 (third scenario).

Figure 47 shows the resource utilization for host#4.

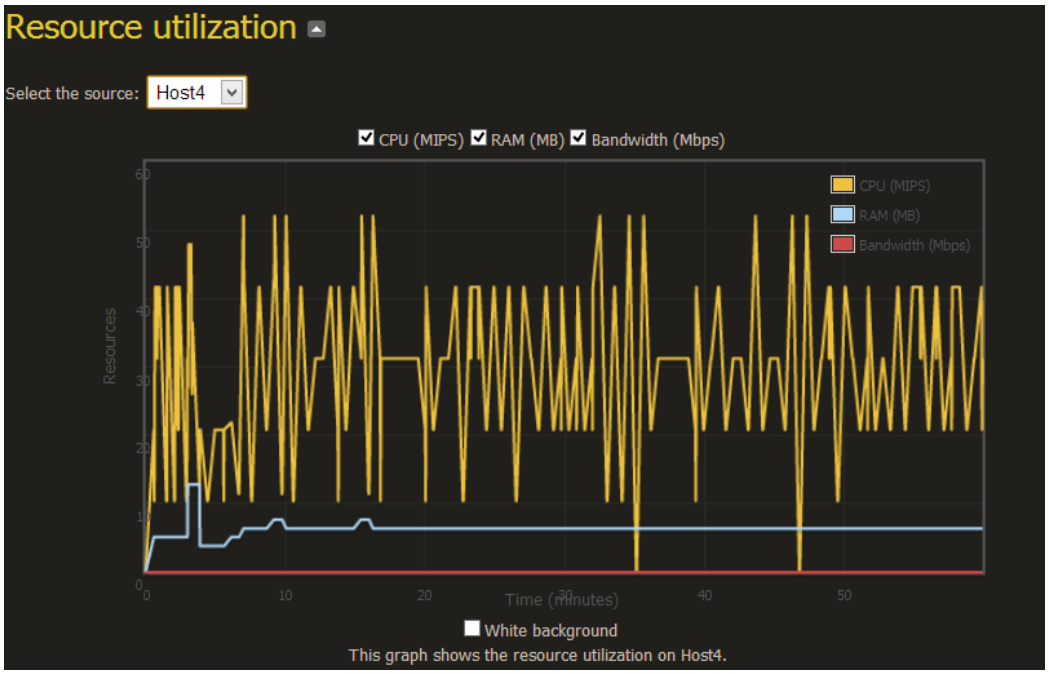


Figure 39 : resource utilization on Host4 (third scenario).

Conclusion:

In this chapter, first I presented the design of our solution for monitoring stage. After that, I presented the implementation of our solution via a scenario of over-promising and under-delivering problem. Finally, I demonstrated the idea of live virtual machine migration using cloud computing simulation known as CloudSim [Calheiros., et al, 2011]. I demonstrated virtual machine migration via 3 scenarios. In first scenario, there are 2 providers and 2 customers. In the second scenario, there are 2 providers and 15 customers. In third scenario, there are 5 providers and 40 customers. In the 3 scenarios, I show how in case of the SLA been violated and unfulfilled by one provider, our *Decide* and *Act* phase from our monitoring autonomic control loop can recommend and perform live virtual machine migration to another provider. For that, I recommend that some SLA parameters about migration (e.g. Down-time) need to be added and negotiate about, to make sure the virtual machine migration is *seamless* which means that the down-time of the virtual machine throughout the migration operation is unnoticeable by the customer. In next chapter, I will show the implementation of gathering, filtering, negotiation and agreement stage.

Chapter 7

Implementation of Gathering, Filtering, Negotiation and agreement stage.

In last chapter, first I presented the design of our solution for monitoring stage. After that, I presented the implementation of our solution via a scenario of over-promising and under-delivering problem. Finally, I demonstrated the idea of virtual machine migration using cloud computing simulation known as CloudSim. This chapter shows the implementation of gathering, filtering, negotiation and agreement stage.

7.1. Gathering and Filtering stage

The main purpose of gathering stage is to gather the customers' requests and providers' offers. The main purpose of filtering stage is filter providers in order to recommend the best matched candidates base on customer request.

First step, I installed and configured Apache *2.2.17* web server. I chose *Apache* over IIS Internet Information Services (IIS) from Microsoft, because it is open source and under

General Public License Moreover, it is Cross-platform and supported by more Platforms (Windows, Mac OS X, Linux, BSD, Solaris, eCS, OpenVMS, AIX, z/OS).

Second step, I installed and configured *PHP 5.3.5* server-side scripting language. One of the reasons why I chose PHP is because it supports command line scripting, this will let us to run or schedule-run PHP script without using browser. Command line scripting will let agent able to run the PHP without human interaction. PHP is also picked because it comes with SimpleXML. SimpleXML is a PHP's extension provides a fast way of getting data from an XML file.

Third step, I installed and configured *MySQL 5.5.8*. I selected MySQL database because is an open source system and under General Public License. Also, I chose MySQL because; it is Cross-Platform database system.

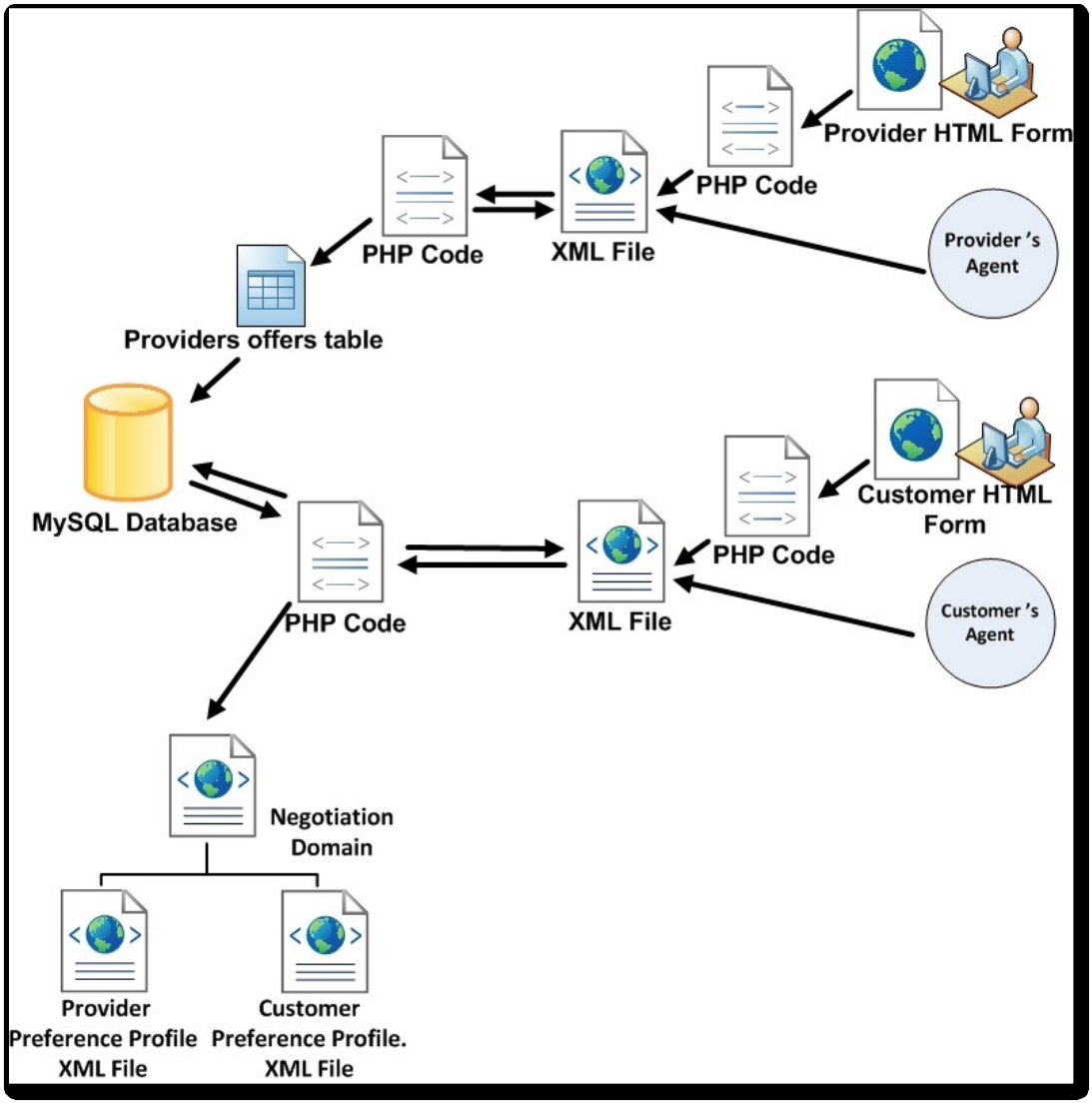


Figure 40 : Implementation of Gathering and Filtering stage

:

Figure 48 shows how gathering and filtering stage are implemented. Figure 48 also shows who I first implemented a user HTML form-based for the users (Customers and Providers) to create and update the framework with offers and requests. I implemented a PHP code to send what users enter in HTML form to a MySQL database. Also, by using SimpleXML, PHP code will easily read data from XML file then to save them to MySQL database.

Using XML file to update customers and provider's offers and requests will make taking human out of the loop for *Gathering* stage possible. User's agent will be also able to update offers and requests via a XML file. Once the XML file uploaded to the cloud by the users' agent, the PHP code will be able to access it and update the framework with customers and provider's offers and requests.

Figure 49 shows how I used *phpMyAdmin* tools to administer our MySQL database via web browser and over the web.

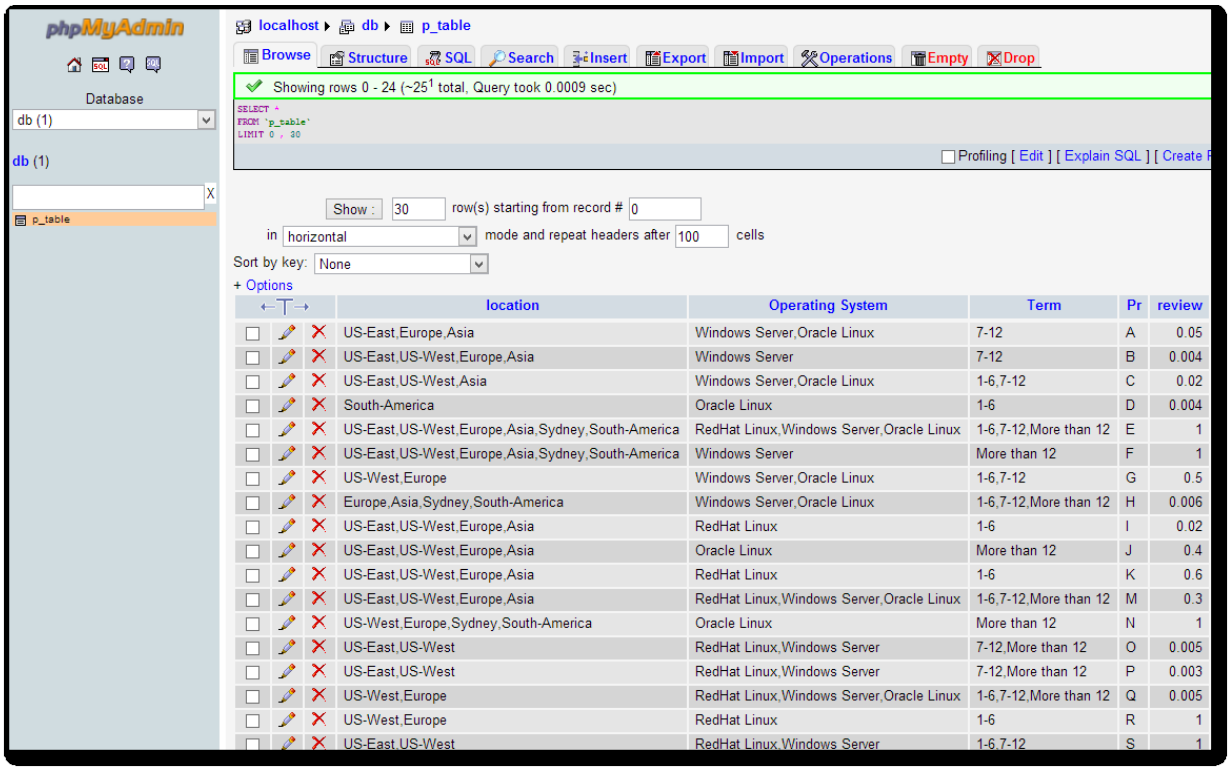
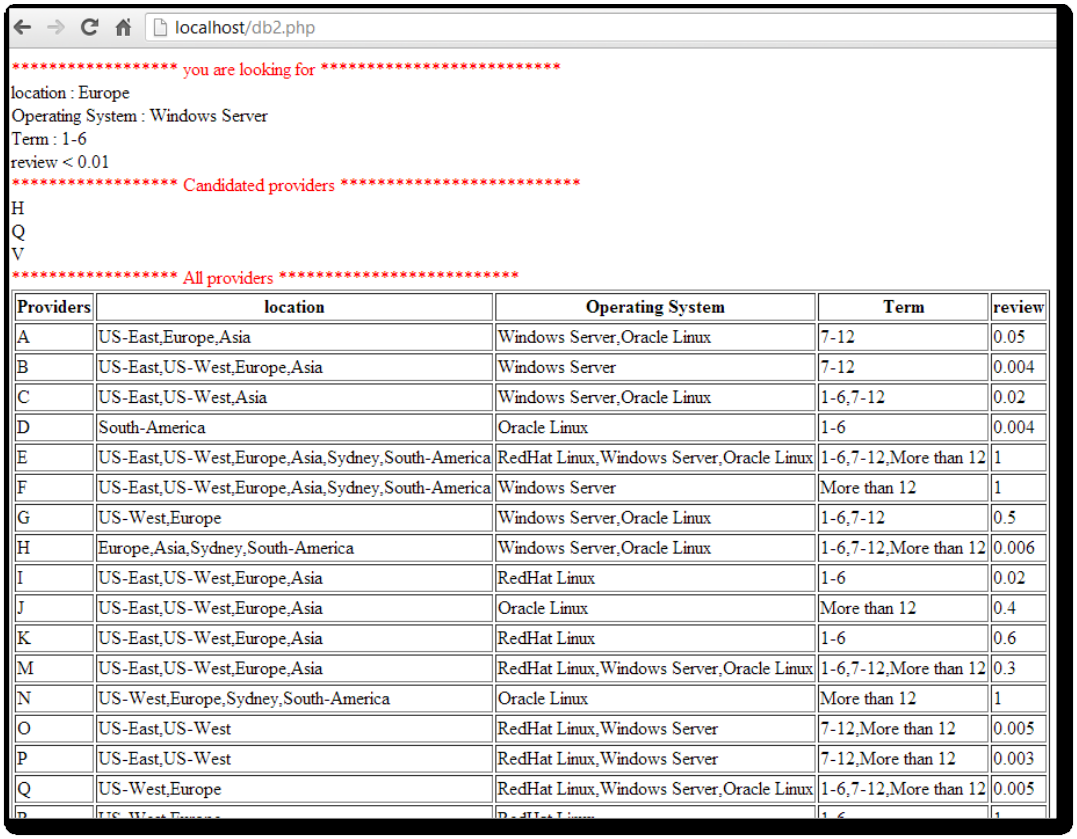


Figure 49 : Using phpMyAdmin to tools to administer our MySQL database



localhost/db2.php

********** you are looking for **********

location : Europe
Operating System : Windows Server
Term : 1-6
review < 0.01

********** Candidated providers **********

H
Q
V

********** All providers **********

Providers	location	Operating System	Term	review
A	US-East,Europe,Asia	Windows Server,Oracle Linux	7-12	0.05
B	US-East,US-West,Europe,Asia	Windows Server	7-12	0.004
C	US-East,US-West,Asia	Windows Server,Oracle Linux	1-6,7-12	0.02
D	South-America	Oracle Linux	1-6	0.004
E	US-East,US-West,Europe,Asia,Sydney,South-America	RedHat Linux,Windows Server,Oracle Linux	1-6,7-12,More than 12	1
F	US-East,US-West,Europe,Asia,Sydney,South-America	Windows Server	More than 12	1
G	US-West,Europe	Windows Server,Oracle Linux	1-6,7-12	0.5
H	Europe,Asia,Sydney,South-America	Windows Server,Oracle Linux	1-6,7-12,More than 12	0.006
I	US-East,US-West,Europe,Asia	RedHat Linux	1-6	0.02
J	US-East,US-West,Europe,Asia	Oracle Linux	More than 12	0.4
K	US-East,US-West,Europe,Asia	RedHat Linux	1-6	0.6
M	US-East,US-West,Europe,Asia	RedHat Linux,Windows Server,Oracle Linux	1-6,7-12,More than 12	0.3
N	US-West,Europe,Sydney,South-America	Oracle Linux	More than 12	1
O	US-East,US-West	RedHat Linux,Windows Server	7-12,More than 12	0.005
P	US-East,US-West	RedHat Linux,Windows Server	7-12,More than 12	0.003
Q	US-West,Europe	RedHat Linux,Windows Server,Oracle Linux	1-6,7-12,More than 12	0.005

Figure 50 : Finding the candidate providers (Filtering).

Figure 50 shows an example of the list of the candidate providers as known as Filtering stage. Once the candidate provider is found, the negotiation domain file and the preference profiles files will be created as XML files. The negotiation domain XML file and the preference profiles files will be used in the next stage which is negotiation.

7.2. Negotiation stage

The outputs of the last stage (Gathering and Filtering stage) will be the input for this stage. The output will be XML files (Negotiation domain file and the preference profiles). The Agent will be able to use the XML files in the negotiation environment and API I are using, GENIUS. The agents need to use the GENIUS Agent API which will allow the agents to sense the negotiation environment and to follow the negotiation protocol. GENIUS is a "negotiation environment that implements an open architecture for heterogeneous negotiating

agents" (Lin, et al., 2012). It allows researchers to form agent that can negotiate with other agents with no previous knowledge about the agents and agents' negotiation preferences. The only information that is shared is the negotiation domain file and negotiation session deadline. Negotiation session deadline can be Time-based or in Rounds-based. The negotiators need to agree on selecting the negotiation session deadline to be Round-based protocol or Time-based protocol. How is selection the negotiation session deadline will affect the negotiation outcome will be investigated later in this work.

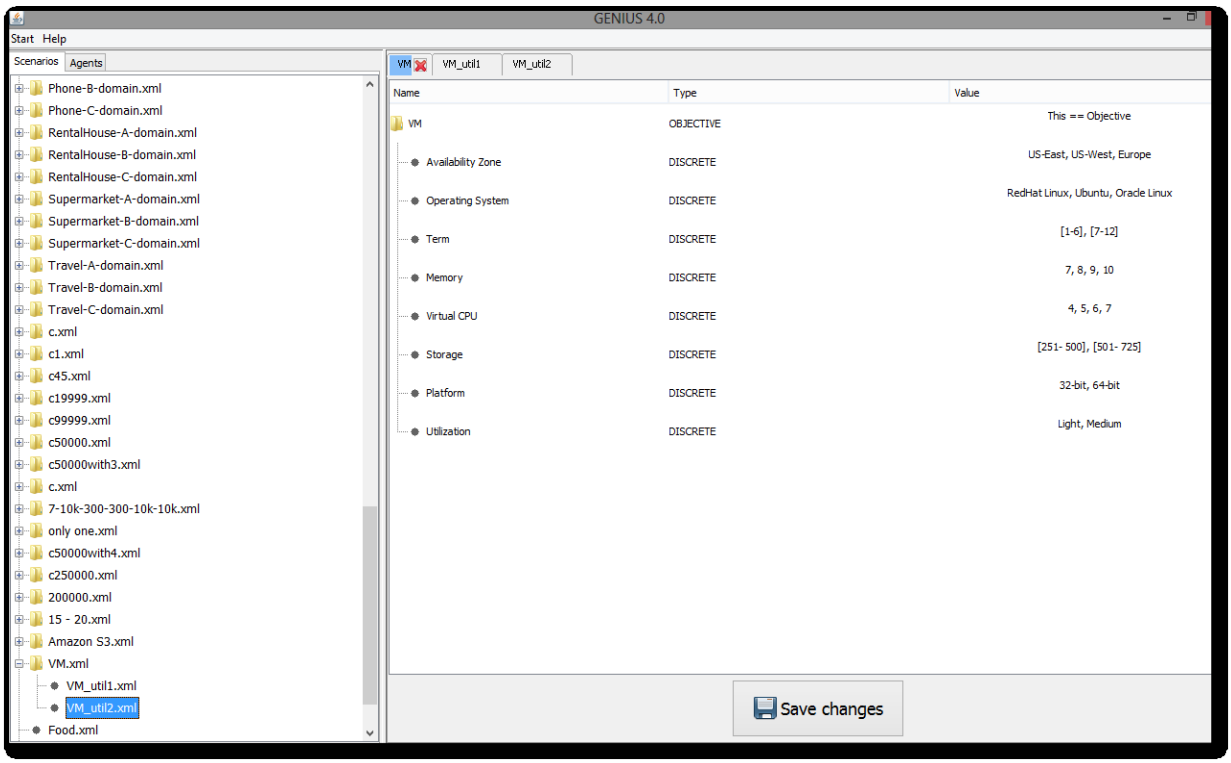


Figure 51 : Creating the Negotiation domain xml file via GUI

Negotiation domain is an XML file made of all the possible bids (list of *Issues and Values* that inside each Issue). Figure 51 shows an example of how to create negotiation domain via graphical user interface.

Negotiation preferences are extended files of Negotiation domain file. Negotiation preference is an XML file as well. Negotiation preferences hold private information about the agent preferences. Figure 52 show an example of how to create preference profiles xml file via GUL. So it includes the *Evaluation values* for each *value* inside each *Issue*. It also includes the *weight* of each *Issue*, showing the important of each *Issue*.

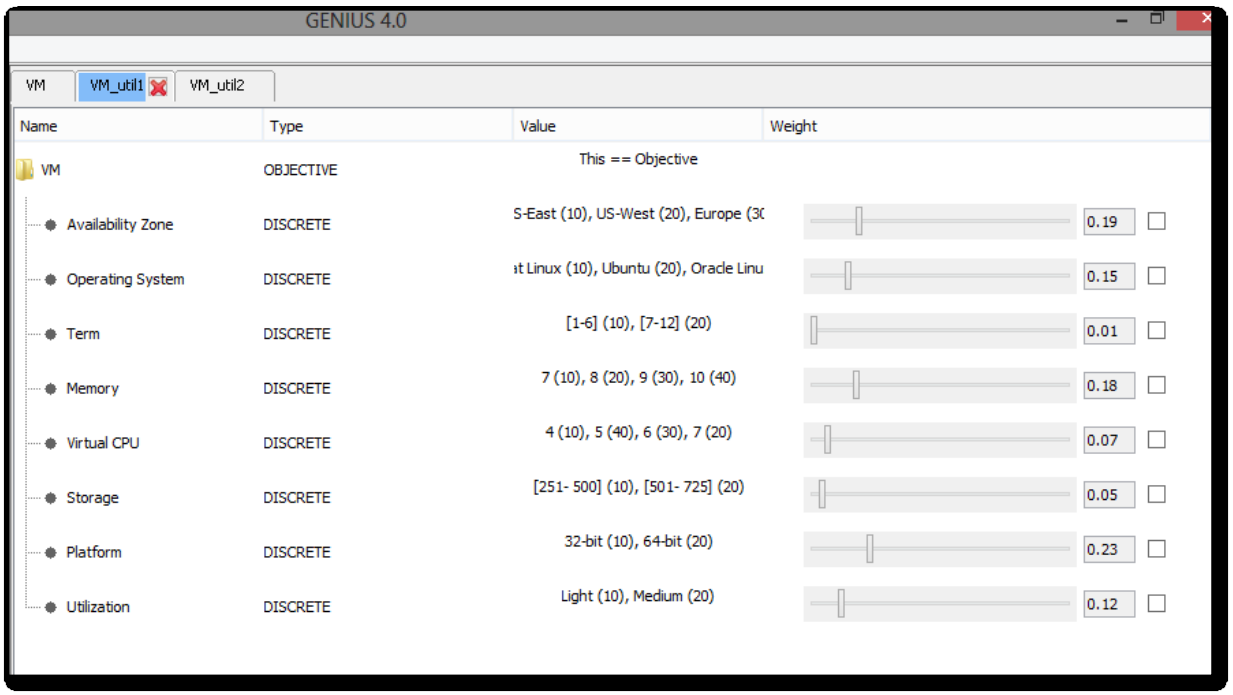


Figure 41 : Creating the preference profiles xml file via GUL

The outcome of this stage will be the agreed bid which will be put in XML file then made ready for the next stage (SLA Agreement Stage).

Figure 53 shows the class diagram for the main classes in GENIUS (Lin, et al., 2012).

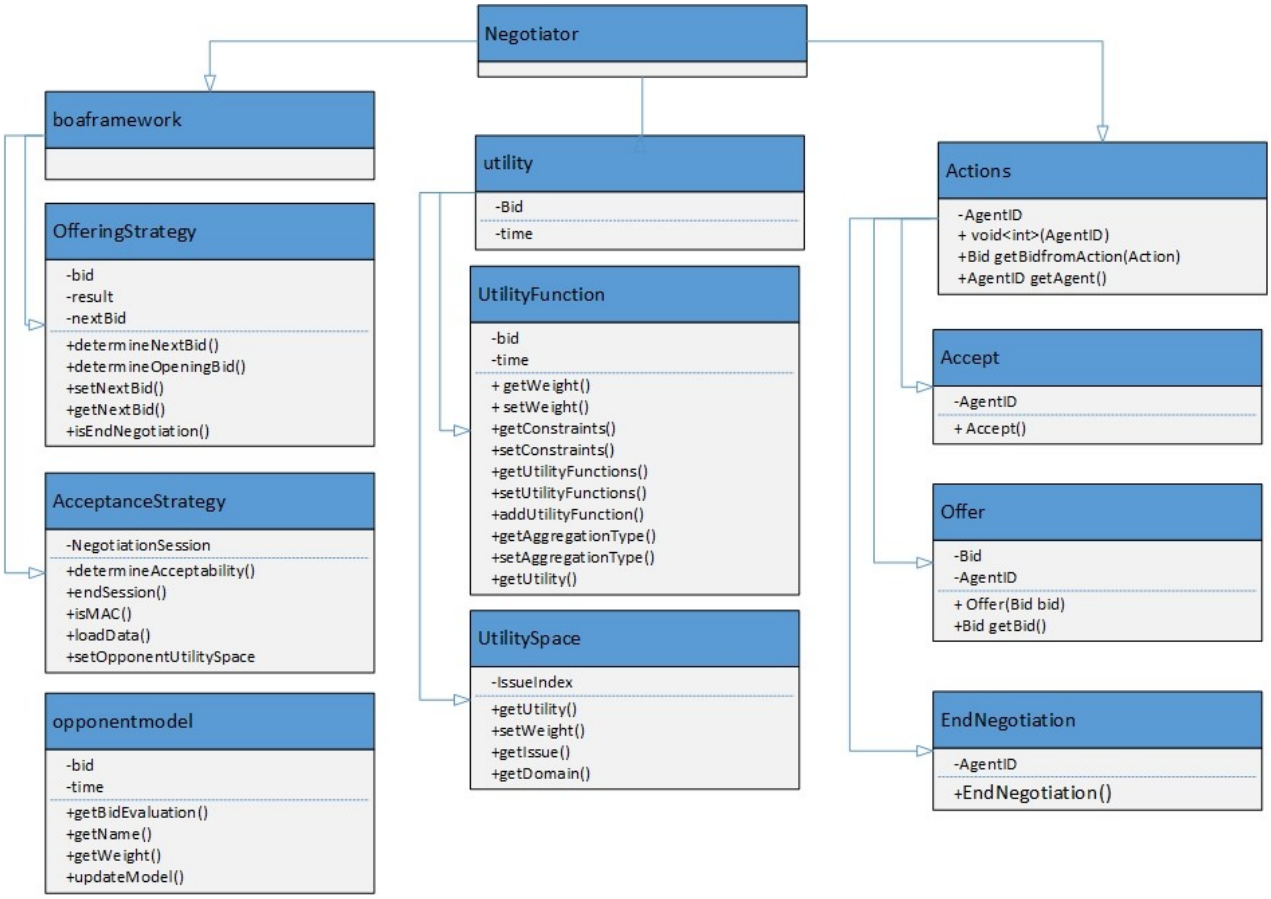


Figure 53: class diagram for GENIUS.

7.2.1 Designing and Implementing our agent Wise H-T.

As I mentioned above our agent Wise H-T is made of Bidding strategy from HardHeaded Bidding strategy and the Acceptance strategy from Tit for Tat Acceptance strategy. The Opponent model is opposite model. Figure 54 is the class diagram for Wise H-T.

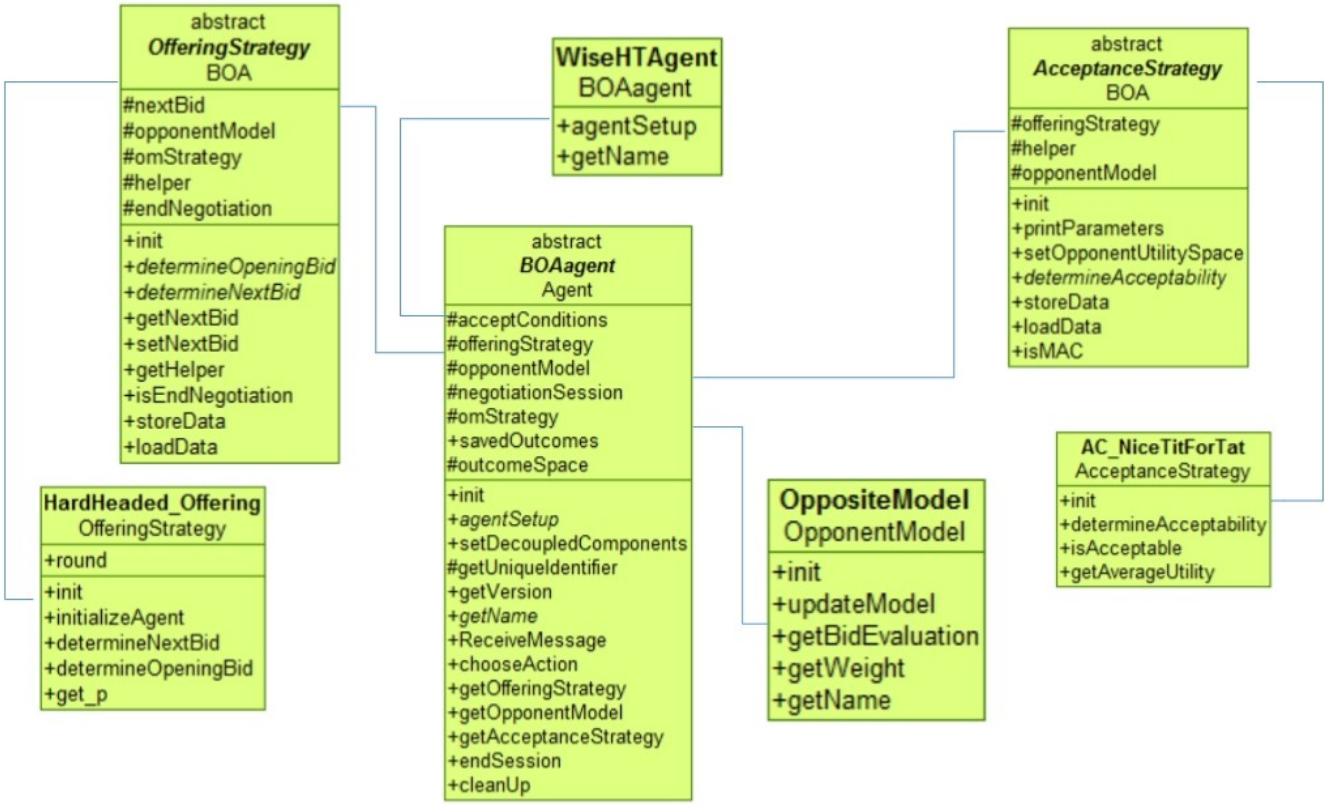


Figure 54: Class Diagram for Wise H-T

7.2.1.1Wise H-T Pseudo-Code for Accepting Strategy:

The next is the algorithm for Wise H-T accepting strategy.

INPUT: Opponent Offer (**OO**), Bidding Strategy Proposed Offer (**BSPO**), Remaining time of Negotiation (**RTN)** Domain size (**DZ).**
OUTPUT: Accepted Offer (**AO**).

While OO is not accepted
Calculate the OO utility (OO_U)

Calculate the BSPO utility (BSPO_U)
If BSPO_U < OO_U

```
    Accept the OO.

Endif

If RTN < %2 of the Negotiation time.
        If the DZ is > 10000
                Waite for 40 offers from the Opponent.
        Endif
        If the DZ is < 10000
                Waite for 10 offers from the Opponent
          Endif
Endif.
```

7.2.1.2 Wise H-T Pseudo-Code for Bidding Strategy.

Algorithm 1 : Wise H-T accepting strategy

The next is the algorithm for Wise H-T Bidding strategy.

INPUT: **P**ossible **A**greements (PA), Remaining time of Negotiation (**RTN)**.
OUTPUT: bids.

```
Calculate the utility for each PA
Sort all the PAs based on its utility.
WHILE RTN > %5 of the Negotiation time.
        Select the bid with the highest utility.
        Offer the selected bid.
ENDWHILE
If RTN < %5 of the Negotiation time.
        Select the bid with the 2nd highest utility.
        Offer the selected bid.
 Endif.
If RTN < %2.5 of the Negotiation time.
        Select the bid with the 3nd highest utility.
        Offer the selected bid.
 Endif.
If RTN < %1.5 of the Negotiation time.
        Select the bid with the 4nd highest utility.
        Offer the selected bid.
 Endif
```

Algorithm 2: Wise H-T Bidding strategy

7.2.1.3 Wise H-T Pseudo-Code of Opponent Model:

The next is the algorithm for Wise H-T Opponent Model.

INPUT: Opponent Offer (**OO**)
OUTPUT: Opponent Offer Assumed utility (**OOAU**).

Calculate Own utility of OO (OUOO)
Calculate OOAU = **1**-OUOO

Algorithm 3: Wise H-T Opponent Model

7.3. SLA agreement stage

The agreed values of the agreed bid will be kept in the database. In this stage the provider and the customer will be informed about the Agreement, which will be specified in measurable terms. The output of the SLA Agreement stage will be a list of metrics that are monitored by the following stage.

7.4 Conclusion

In this chapter the implementation of gathering, filtering, negotiation and agreement stages is explained. For the gathering and filtering stage, I installed and configured Apache 2.2.17 web server. I also installed and configured PHP 5.3.5 server-side scripting language. Then, I installed and configured MySQL 5.5.8. I implemented a user HTML form-based for the users (Customers and Providers) to create and update the framework with offers and requests. I implemented a PHP code to send what users enter in HTML form to a MySQL database.

The outputs of the Gathering and Filtering stage will be the input for the negotiation stage. The output will be XML files (Negotiation domain file and the preference profiles). In the Negotiation stage the agents need to use the GENIUS Agent API which will allow the agents to sense the negotiation environment and to follow the negotiation protocol. Then I explained

how I designed and implemented our agent Wise H-T using the BOA framework. In the next chapter, the evaluation methodology will be explained via a scenario first. Then, the four evaluation experiments will be performed.

Chapter 8

Negotiation Experiments and Agents Evaluation

In the last chapter, the implementation of gathering, filtering, negotiation and agreement stages explained. In this chapter, the evaluation methodology will be explained via a scenario first. Then, the four evaluation experiments will be performed. At the end of each experiment I will give recommendations based on the results to the customer, provider and the negotiation organizer.

8.1 Evaluation methodology.

In this section, I will explain the evaluation methodology via a scenario. In this scenario, a customer and a provider will negotiate over the specifications of a cloud virtual machine. The specifications has been taken from Amazon Elastic Compute Cloud (Amazon EC2)[Amazon EC2, 2014] to represent a real world problem data. There are 8 *issues* that they will negotiate about: Availability Zone, Operating System, Term, Memory, Compute Units, Storage GB, Platform, and Utilization. Each issue has options known as *Value*s. The availability zone issue has 4 values. The operating system issue has 2 values. The Term issue has 3 values. The memory issue has 4 values. The compute unit's issue has 4 values. The Storage issue has 2 values. The platform issue has 2 values. The utilization issue 2 values. For that in this

negotiation, there are 3072 possible agreements (packages) known as Domain. Table 8 shows the whole domain.

Table 8 : evaluation methodology scenario

Issue	Value	provider Evaluation	Weight	Customer Evaluation	Weight
Availability Zone (location)	US-East US-West Europe Asia	100 50 25 75	0.19	25 50 100 25	0.36
Operating System	Linux Microsoft Win	100 50	0.09	100 50	0.19
Term (months)	[1-6] [7-12] More than 12	25 50 100	0.32	100 50 25	0.10
Memory GB	7 8 9 10	25 50 75 100	0.13	75 50 25 100	0.10
Compute Units /virtual core (CPU)	4 5 6 7	25 100 75 50	0.05	100 50 75 25	0.04
Storage GB	[251- 500] [501- 725]	50 100	0.08	100 50	0.04
Platform	32-bit 64-bit	50 100	0.05	100 50	0.08
Utilization	Light < 39% Medium <75%	50 100	0.09	50 100	0.09

A weight matrix is utilised to help the customer and a provider to give a weight for each issue. The customer and a provider need to give a weight for each issue to represent how important each issue for them. The weights of all issues must sum up to 1.The customer and a provider can adjust the relative weights of the issues by using the sliders next to that issue. See figure 55.

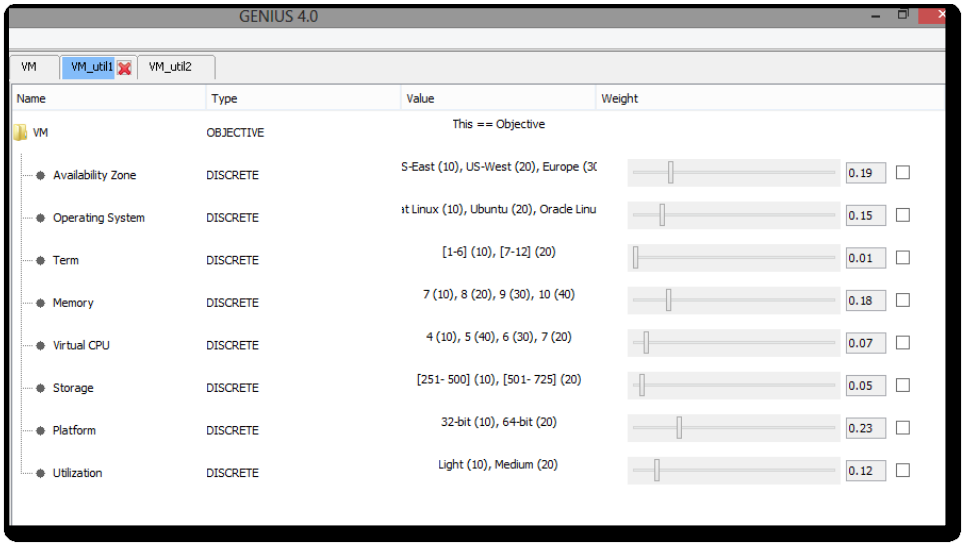


Figure 55: a weight matrix

When the customer and a provider move a slider, the weights of the other sliders are automatically updated such that the all weights still sum up to 1. If the customer and a provider do not want that the weight of another issue automatically changes, they can lock its weight by selecting the checkbox behind it.

Also, the customer and a provider need to specify the *evaluation* value of each value. During the negotiation the utility of a value is determined by dividing the value by the highest value for that particular issue.

The *Utility function* maps every possible outcome $\omega \in \Omega$ to a real-valued number in the range $[0, 1]$, where ω is the outcome and Ω is the domain. The overall utility consists of a weighted sum of the utility for each individual issue.

$$U(_{v1,\ldots\ldots vn}) = \sum_{i=1}^{N} w_i \frac{eval(v_i)}{Max(eval(v_i))}$$

An offer is a set of chosen values $_{v1,\ldots\ldots vn}$ for each of the n issues. Each of these values has been assigned an evaluation value $eval(v_i)$ in the utility space. The utility is the weighted sum of the normalized evaluation values.

Now, I will run the negotiation. Customer will be represented by Agent A which is Hardheaded. The provider will be represented by Agent B which is Tit for Tat. Figure 56 shows how the negotiation started. The 3072 red dots are the domain which is all possible bids.

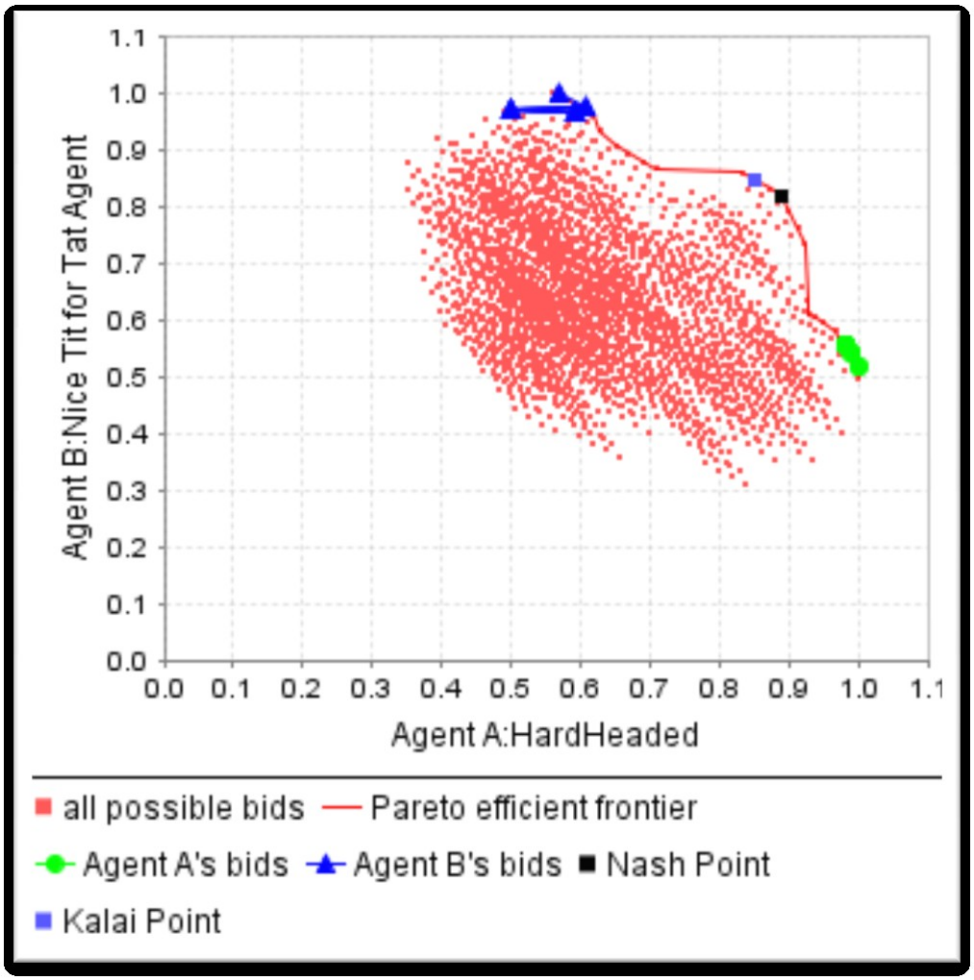


Figure 56 : Early offers HardHeaded vs Tit for Tat

The Customer's Agent A HardHeaded begins with this offer: (Offer: Bid[Availability Zone (location): Europe, Operating System: Linux , Term (months): [1-6], Memory GiB: 9, Compute : 4, Storage GB: [251- 500], Platform: 32-bit, Utilization: Medium <75%,]) This

offer has a utility of 1.0 for HardHeaded (A Customer) and in the same time this offer has a utility of 0.515 for Tit for Tat Agent B(Provider)

So the provider (Tit for Tat - Agent B) rejected the offer by sending the following counter-offer. (Offer: Bid[Availability Zone (location): US-East, Operating System: Linux , Term (months): More than 12, Memory GiB: 9, Compute : 6, Storage GB: [501- 725], Platform: 64-bit, Utilization: Medium <75%,]) This offer has a utility of 0. 569 for HardHeaded (Customer) and a utility of 1.0 for Tit for Tat Agent B (Provider).

Then, after that both sides kept sending offer and counter-offer. The deadline is set at 180 seconds. They reached the agreement after exchanging offers 3878 times. See figure 57 shows.

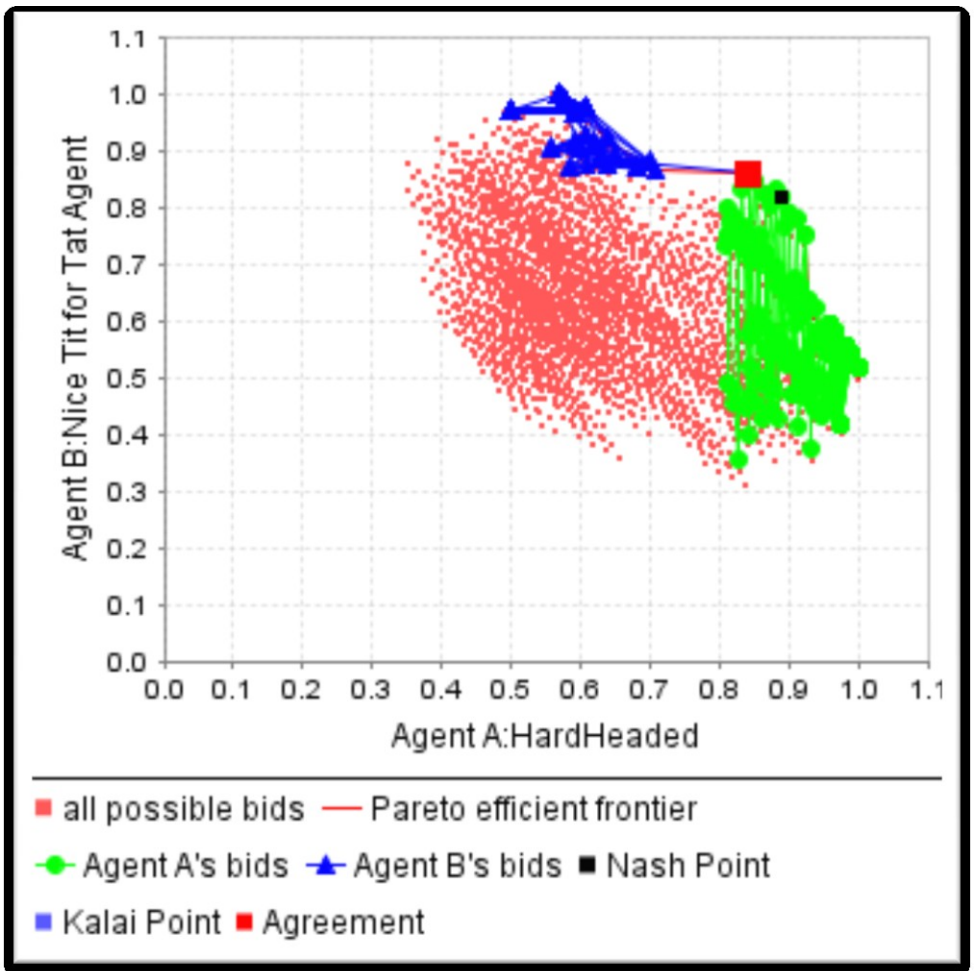


Figure 57: the Agreement HardHeaded vs Tit for Tat

The agreement was for this package: (Offer: Bid[Availability Zone (location): Europe, Operating System: Linux , Term (months): More than 12, Memory GB: 9, Compute : 6, Storage GB: [501- 725], Platform: 64-bit, Utilization: Medium <75%,]). The agreement utility is 0.842 for HardHeade (Agent A) and agreement utility is 0.857 Tit for Tat (AgentB).

Now after I showed an example how two agents negotiation works, in the next section I will use the same method to evaluate the agents, that is finding the *utility* that each agent get if the negotiation end with an agreement. If no agreement is reached then the agent will get a 0 *utility*. Also, there are two criteria that will be taken into account, in order to evaluate each agent when the agent negotiates with many agents as a tournament. First, is the *performance*

and the second one is the *fairness*. Performance is the sum of the all utilities the agent got while negotiating with the other agents. The agent with the highest number means that the agent has the best performance:

$$Performance = 0 \leq \frac{\sum_{k=1}^{n_a} u_a}{n_a} \leq 1$$

Where n_a is the number of the agents and $u_a$is the utility of the agent. The Performance is measured between 0 and 1.

Fairness is formally defined as:

$$Fairness = 0 \leq \frac{\sum_{k=1}^{n_a} u_a + u_{op}}{n_a * 2} \leq 1$$

Where n_a is the number of the agents and $u_a$is the utility of the agent and u_{op} is the utility of the opponent. The Fairness is measured between 0 and 1. The agent with the highest number means that the agent has the best fairness. The Fairness is measured between 0 and 1.

In this next section, I will perform four kinds of experiments. I ran 40 negotiation sessions to complete the first experiment. For the second experiment, I ran 210 negotiation sessions. To complete the third experiment, I ran 720 negotiation sessions. Also, 540 negotiation sessions had to be run to complete the fourth experiment. Therefore, in this work I ran total of 1510 negotiation sessions. More information about this number will be given later.

The order of the experiments in this research reflects the stages I haven't gone through this work evaluation. On the other words, first, I wanted to know if it is useful to use the start-of-the-art agents? Then I wanted to create a novel agent to improve the start-of-the-art agents. After that, I wanted to further investigate our agent and other agents based on the deadline and the size of the domain. In more details; the first experiments will try to answer important

questions to prove to customer and provider the benefit of using Agents to negotiate comparing to a "Take-it-or-leave-it" strategy. Also, the first experiments will be answering questions which are important to demonstrate to them the implications of using competitive strategy or using a cooperative strategy. Then, in the second experiments I are trying to answer questions about our Agent Wise H-T capability. Also, the second experiments will be help to answer questions about the benefit of using a combined strategies made of a competitive strategy and cooperative strategy. In the third experiments I wanted to found out if there are relation between the size of the negotiation domain and the deadline of the negotiation session and how this affects the agents' agreement outcome. In the fourth experiments I are trying to study how the agents will act in single issue negotiation, which will be the "Price Negotiation" in this work.

So at the end of all the experiments I will be able to give recommendations to the customer, provider and the negotiation organizer (which can be a third party who take care of negotiation environment and organise the negotiation sessions). For the provider and customer, some recommendations will be given about the risk and the benefit of using competitive agent or cooperative agent. Also, some recommendations will be given about which agent shall be used based on the size of the domain and deadline. For negotiation organizer, some recommendations will be given about how to improve the overall community utilities (OCU) which is the sum of all the agents' utilities at each experiment. Also some recommendations will be given about if increasing deadline will improve the overall community utilities (OCU). Also, I will give recommendations to the negotiation organizer about at what time is the negotiation deadline shall be set at, so there is not time wasted? In other words, I will advise the negotiation organizer about; when is increasing deadline will not change or improve the negotiation outcome? This is important for the negotiation organizer choosing the right deadline for each negotiation.

8.1.1 Negotiation Experiments:

In this chapter four kinds of experiments will be performed. Each section will try to answer some unanswered research questions.

The first experiments will try to answer the following questions:

1. Will using an agent for negotiation benefit the customer and the provider, comparing to not negotiating at all?
2. Which one is better: using a competitive strategy or using a cooperative strategy?
3. What are the risk and the benefit of using the competitive strategy or a cooperative strategy?
4. What are the outcomes when the opponent plays the similar strategy or the different strategy?
5. When the customer or provider should use a competitive strategy or a cooperative strategy?

The second experiments will try to answer the following questions:

1. Is it better to use a combined strategy made of a competitive strategy and cooperative strategy rather than two separate strategies on their own?
2. How our proposed agent Wise H-T will perform against the state-of-the-art agents?
3. Is there a relation between the performance and fairness among all the agents?
4. Could the relation between performance and fairness be improved while increasing and decreasing the deadline of the negotiation session?
5. Would the results be affected if the deadline is switched between the round based protocol to a time based protocol?

The third experiments will try to answer the following questions:

1. Is there a relationship between the size of the negotiation domain and the deadline of the negotiation session?
2. If there is a relationship between the size of the negotiation domain and the deadline of the negotiation session, then would this affect the outcome of the negotiation, the performance of the agents and which agent would be affected the most?
3. If there is a relationship between the size of the negotiation domain and the deadline of the negotiation session, then would the results be affected if the deadline is switched between the round-based protocol to a time-based protocol?

The fourth experiments will try to answer the following questions, taken into the account that since the "Price Negotiation" is the "Single Issue Negotiation" in the sense that the provider and customer will negotiate over the price of the previously agreed package:

1. Would all the agents be capable to perform this kind of negotiation?
2. Would performance of each agent be affected while increasing and decreasing the deadline, as well as increasing and decreasing the size of the domain?
3. What should the deadline be set up at that by increasing the deadline would not lead to any improvement to the outcome?

8.2 First experiment: Google and Amazon (Hardliner)

This part will demonstrate the benefit of negotiating to offer a negotiable customized SLA comparing to off-the-shelf SLA. Firstly, the agent *Hardliner* will be used to represent the cloud provider nowadays, for instance Google and Amazon. At the moment the cloud provider only offer off-the-shelf SLA, also known as "take-it-or-leave-it". In this experiment, I will show how current cloud providers miss a lot of agreement. Furthermore, I will run two scenarios: the first one is when the customer is not flexible. (HardHeaded will be used for this scenario) and the second scenario when the customer is more flexible and he needs to end the negotiation with an agreement (Tit for Tat will be used for this scenario).

Two methods will be used to investigate the agents' capability :

1. By negotiating with a Hardliner agent (see section 3.7.3).
2. By negotiating with itself. (using the same negotiation strategy).

The scenario assumption for this experiment is that a customer is looking for a provider who is capable of providing Infrastructure as a Service (IaaS). The customer is looking for virtual machine to be used as a Database Server with the criteria as shown in table 9. In table 9, the evaluation's value and the weight for each issue shows the preferences for the customer and the provider. In this scenario, there are 8 issues: Availability Zone, Operating System, Term, Memory, virtual CPU, Storage, Platform and Utilization.

In this scenario, Availability Zone issue has 3 options. Operating System has 3 options, Term has 2 options; Memory has 4 options. Virtual CPU has 4 options, Storage has 2 options and Platform has 2 options, Utilization has 2 options, so there are 2304 possible outcomes for this negotiation. The deadline for each negotiation is set to 3 minutes.

The second assumption is that the competitiveness level in this scenario is *Medium*. The competitiveness of a domain is defined as "the minimum Euclidean distance from a point in the utility space to the point which represents maximum satisfaction for both agents (that is, the point at which each agent achieves a utility of 1)" (Williams , et al., 2014). Therefore, in this negotiation scenario, there will be four kinds of outcomes/ packages; the first group of packages will satisfy the provider only, the second group of packages will satisfy the customer only, the third group of packages will not satisfy both sides, the fourth group of packages will fairly satisfy both sides.

Table 9 : virtual machine criteria

Issue	Value	Customer Evaluation	Weight	provider Evaluation	Weight
Availability Zone (location)	US-East US-West Europe	0.33 0.66 1.00	0.19	0.66 0.33 1.00	0.15
Operating System	RedHat Linux Ubuntu Oracle Linux	0.33 0.66 1.00	0.15	1.00 0.66 0.33	0.29
Term (months)	[1-6] [7-12]	0.50 1.00	0.01	0.50 1.00	0.03
Memory GB	7 8 9 10	0.25 0.50 0.75 1.00	0.18	0.75 0.50 0.25 1.00	0
Compute Units /virtual core (CPU)	4 5 6 7	0.25 1.00 0.75 0.50	0.07	1.00 0.50 0.75 0.25	0.11
Storage GB	[251- 500] [501- 725]	0.50 1.00	0.05	1.00 0.50	0.05
Platform	32-bit 64-bit	0.50 1.00	0.23	1.00 0.50	0.23
Utilization	Light < 39% Medium <75%	0.50 1.00	0.12	0.50 1.00	0.12

After finding providers who are willing to provide offers matching the above criteria, the customer will negotiate with them. However, each side (provider and customer) have different preferences. For example the provider would like a customer requiring long term facilities in one location with less Utilization. So now they need to negotiate. The negotiation will be closed in the sense that there is uncertainty about the opponent's preferences.

8.2.1 Negotiation experiments outcome:

Next the agent's performance will be investigated by negotiating against Hardliner agent and then against the same negotiation strategy.

8.2.1.1 HardHeaded vs Hardliner

The outcomes of all negotiation sessions between HardHeaded and Hardliner shows that the number of the rounds (offers exchanged) between the agents is high, the average of the rounds was 11200 rounds in each session (180 seconds). However, the results show that all the negotiation sessions ended with no agreements. The high number of round shows that HardHeaded is trying to reach an agreement. However, because it is a selfish and competitive agent, it will not compromise to an offer lower than Nash point to reach an agreement and this is why all negotiation sessions ended with no agreements. As figure 58 and table 10 show, agent HardHeaded was trying more than Hardliner by offering more compromising offers but not lower than 0.8 utility for itself.

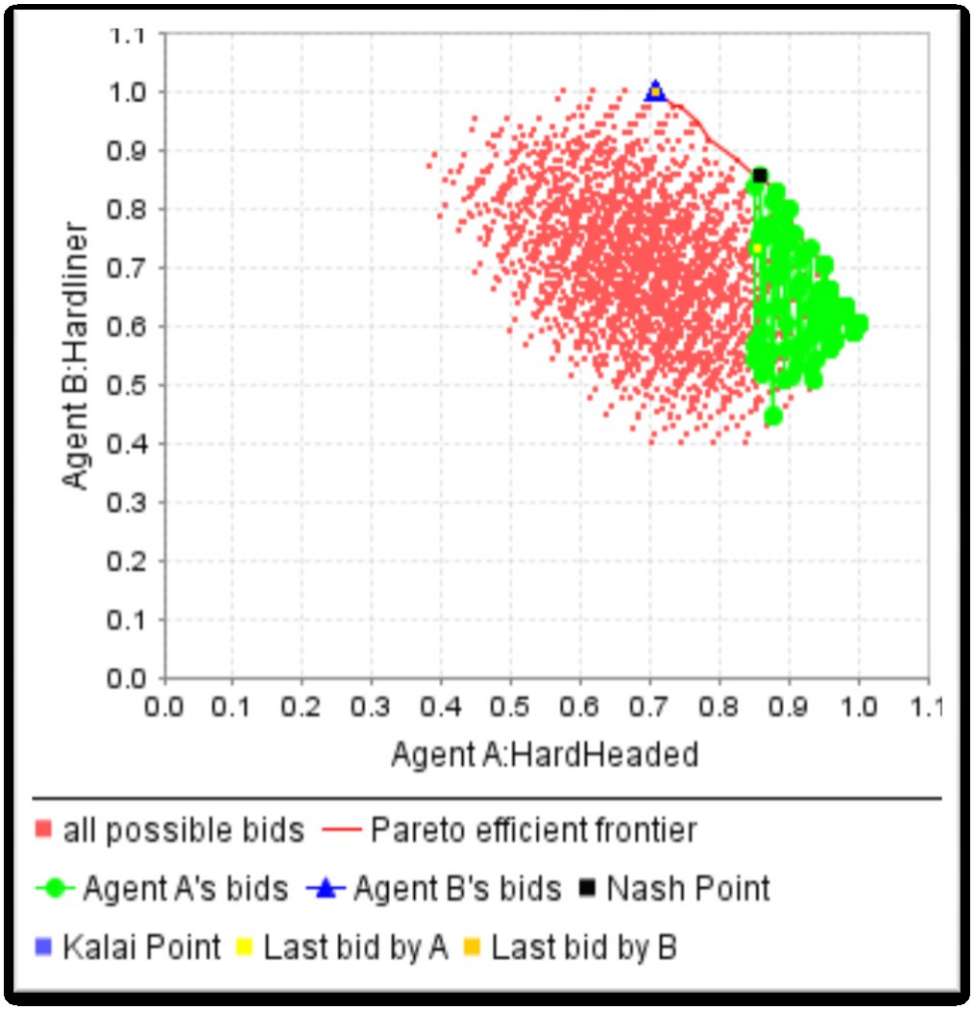


Figure 58 : HardHeaded vs Hardliner

The red dots are the domain. The green dots show that HardHeaded was trying to be more flexible. The green dots shows all the offers that hardheaded . The blue dots (many dots on the top of each other's) show that Hardliner kept offering the same bid which has high utility for itself.

Table 10 : HardHeaded vs Hardliner

Agent	**Utility of each Agent's first offer**	**Utility of Last offer for each agent.**
Hardliner	0.99	0.99
HardHeaded	0.99	0.85

8.2.1.2 Tit for Tat vs. Hardliner

The outcomes of all negotiation sessions between Tit for Tat and Hardliner shows that the number of the rounds (offers exchanged) between the agents is relatively low, the average of the number of rounds was 270 in each session. In addition, the results show that all the negotiation sessions ended with agreements, however all of them with very high utility for Hardliner agent.

The low number of round shows that Tit for Tat is willing to compromise easily to reaching an agreement. This is due to the fact that Tit for Tat is a cooperative agent. The outcome of the negotiation is that at the end agent Hardliner offer the same insisted offer and Tit for Tat agreed and accepted it even though it was lower than Nash point. The agreement was (Bid[Availability Zone: Europe, Operating System: RedHat Linux, Term: [7-12], Memory: 10, Virtual CPU: 4, Storage: [251- 500], Platform: 32-bit, Utilization: Medium,])

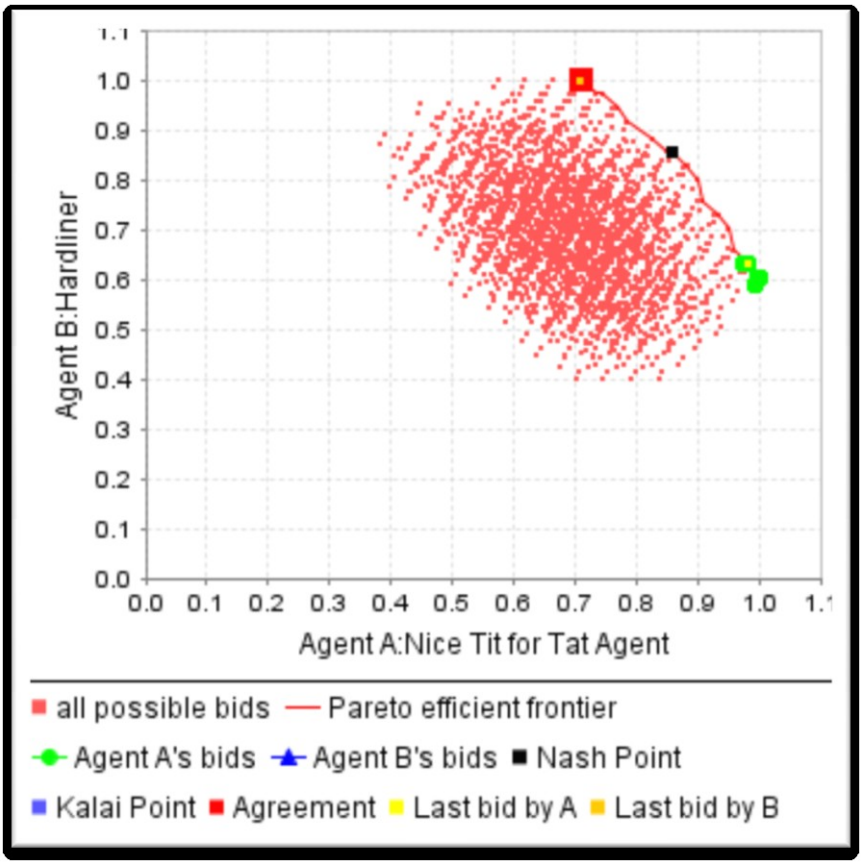


Figure 59 : Tit for Tat vs Hardliner

The red dots are the domain. The green dots are Tit for Tat offer. The red small square shows where the agreement is. There are blue dots under the agreement red square representing hardliner offer.

Table 11: Tit for Tat vs Hardliner

Agent	**Utility of each Agent's first offer**	**Utility of the Agreed bid**
Hardliner	0.99	0.99
Tit for Tat	0.99	0.70

7.2.1.3 By negotiating with itself (against the same negotiation strategy).

When Hardheaded negotiated with itself the negotiation sessions ended without agreements. This is because both sides are the same, selfish and competitive and not willing to compromise to reach an agreement. See the dilemma of negotiation in section 2.4.2.

When Tit for Tat Agent negotiated with itself all the negotiation sessions ended with agreements. Furthermore, most of the agreements were close to the Nash point. This is because Tit for Tat is a cooperative agent so when it faces a cooperative agent; the negotiation always ends with agreements with highest possible utility for both sides. See the dilemma of negotiation in section 2.4.2.

8.2.2 Discussion

After investigating the Hardheaded and Tit for Tat agents by negotiating with Hardliner agent and by negotiating with themselves, I can give the following recommendations;

8.2.2.1 Using take-it-or-leave-it strategy

Since the output of stage 2(Filtering) of our framework is the candidate providers with whom the customer will negotiate separately, the providers need to keep in mind that there is a competition. They need to be careful when they select or build the agent that represents them, selecting a very selfish agent is risky, as it might face a very selfish agent as well which will end the negotiation session to end with no agreement. In today's cloud market all of the providers 'use' a take-it-or-leave-it strategy by offering off-the-shelf SLA. This is because

there are not many cloud providers at present. However, in the near future, where there will be an increased number of cloud providers (or cloud brokers) so they will be more competition, I recommend the cloud providers to offer customized and negotiated SLA by using a cooperative agent like Tit for Tat.

8.2.2.2 Using a selfish agent e.g. HardHeaded is risky.

I recommend providers to use Hardheaded when demand is higher than supply in the market. There is risk of ending some negotiations without agreement. However when the negotiating ends with an agreement, this agent will get the higher utility.

8.2.2.3 Using compromising agent is safe but costly, e.g. TitforTat.

This agent can be recommended for providers who want to reach agreements effortlessly and attract new customers, e.g. new in the market providers or old providers to promote new products. The agent will do its best to reach the best possible agreement for itself, but with low utility when it faces selfish agent.

8.3 Second experiment: (Investigating Wise H-T and start-of-the-art agents)

In this Second experiment, I will try to answer the following questions:

1. Is it better to use a combined strategy made of a competitive strategy and cooperative strategy rather than two separate strategies on their own?
2. How our proposed agent Wise H-T will perform against the state-of-the-art agents?
3. Is there a relation between the performance and fairness among all the agents?
4. Could the relation between performance and fairness be improved while increasing and decreasing the deadline of the negotiation session?
5. Would the results be affected if the deadline is switched between the round based protocols to a time based protocol?

8.3.1 Scenario presentation

Each of the following agents; AgentFSEGA, Gahboninho, HardHeaded, Tit for Tat Agent, IAMhaggler, and Wise H-T will negotiate against a baseline that is made of the following agents (Hardliner, Gahboninho, HardHeaded, Tit for Tat Agent and IAMhaggler), which mean 30 negotiation sessions need to be ran as tournament , but because I are going to investigate the effect of changing the deadline for each tournament as well. The total will be 210 negotiation sessions in this experiment. I will investigate the effect of a deadline to the Negotiation outcomes, I ran the same scenario with the same agents three times, and then I

compared the outcomes: first one a very short time of 10 seconds; second one for 100 seconds and the last one for 1000 seconds.

The same has been done with the round-based protocol: first one a very short time of 10 rounds; second one for 100 rounds, third one for 1000 rounds and the last one for 10000 rounds.

The first scenario assumption is that a customer is looking for a provider who is capable of providing *Storage-a-a Service* to store thousands of high quality photos. The customer is looking for storage as a service with the criteria as shown in the table 12. In table 12, the evaluation's value and the weight for each issue shows the preferences for the customer and the provider. In this scenario, there are 5 issues: Availability Zone, Term, Back up, Data In, Data out. In this scenario, each issue has 4 options, so there are 1024 possible outcomes for this negotiation.

The second assumption is that the competitiveness level in this scenario is *Medium*. Therefore, in this negotiation scenario, there will be four kinds of outcomes/ packages; the first group of packages will satisfy the provider only, the second group of packages will satisfy the customer only, the third group of packages will not satisfy both sides, the fourth group of packages will fairly satisfy both sides.

Table 12 : storage as a service criteria

Issue	Value	Customer Evaluation	Weight	provider Evaluation	Weight
Availability Zone (location)	US-East US-West Europe Asia (Tokyo)	1.00 0.75 0.50 0.25	0.17	1.00 0.50 0.25 0.75	0.19
Term (months)	[1-6] [7-12] [12-24] >24	0.25 1.00 0.75 0.50	0.22	1.00 0.50 0.25 0.75	0.11
Backup	Every 12 hours 1 days 1 week 1 month	0.75 0.25 0.50 1.00	0	0.75 1.00 0.50 0.25	0.18
Data In (Terabyte)	Light (>100 GB) Medium (up to1TB) Heavy (up to 10TB) Very havey (<10TB)	0.25 0.75 1.00 0.50	0.07	1.00 0.25 0.75 0.50	0.22
Data Out (Terabyte)	Light (>100 GB) Medium (up to1TB) Heavy (up to 10TB) Very havey (<10TB)	1.00 0.75 0.50 0.25	0.54	1.00 0.50 0.75 0.25	0.30

After finding providers who are willing to provide offers matching the above criteria, the customer will negotiate with them. However, each side (provider and customer) have different preferences. The negotiation will be closed in the sense that there is uncertainty about the opponent's preferences.

It is essential to set up a deadline for the negotiation, as without a deadline the negotiation might go on forever. The effect of switching between time-based deadline and round-based deadline will be investigated. Also, the effect of the increase and the decrease of the deadline to the negotiation outcome will be investigated.

8.3.2 Negotiation experiments outcome:

In this section, the results will be shown. The following graphs and tables show the results of performance (**P**) and fairness (**F**) for AgentFSEGA (**FSEGA)** , Gahboninho (**Gab)** , HardHeaded (**HH)**, IMhaggler (**IMh)**, Tit for Tat (**TfT)**, Wise H-T (**W H-T)**.

8.3.2.1 Time-based deadline

The table 13 shows the results of *performance* (P) and *fairness* (F) for each agent when I set the deadline to time-based deadline (Seconds).

Table 13: Agents' Performance & Fairness for 10, 100 & 1000 Seconds

Agent	**10 Seconds**	**100 Seconds**	**1000 Seconds**
FSEGA - F	0.91513555	0.910674611	0.910674611
FSEGA - P	0.86859031	0.850717453	0.850717453
Gab – F	0.94257514	0.936155989	0.936155989
Gab – P	0.93785251	0.919180708	0.919180708
HH – F	0.84260277	0.935242133	0.935242133
HH – P	0.86712156	0.93545567	0.93545567
IMh – F	0.90832983	0.909788206	0.910674611
IMh – P	0.856947	0.856947	0.850717453
TfT – F	0.92256178	0.929562717	0.929562717
TfT – P	0.92141553	0.939282768	0.939282768
W H-T - F	0.94166128	0.935242133	0.935242133
W H-T - P	0.95412747	0.93545567	0.93545567

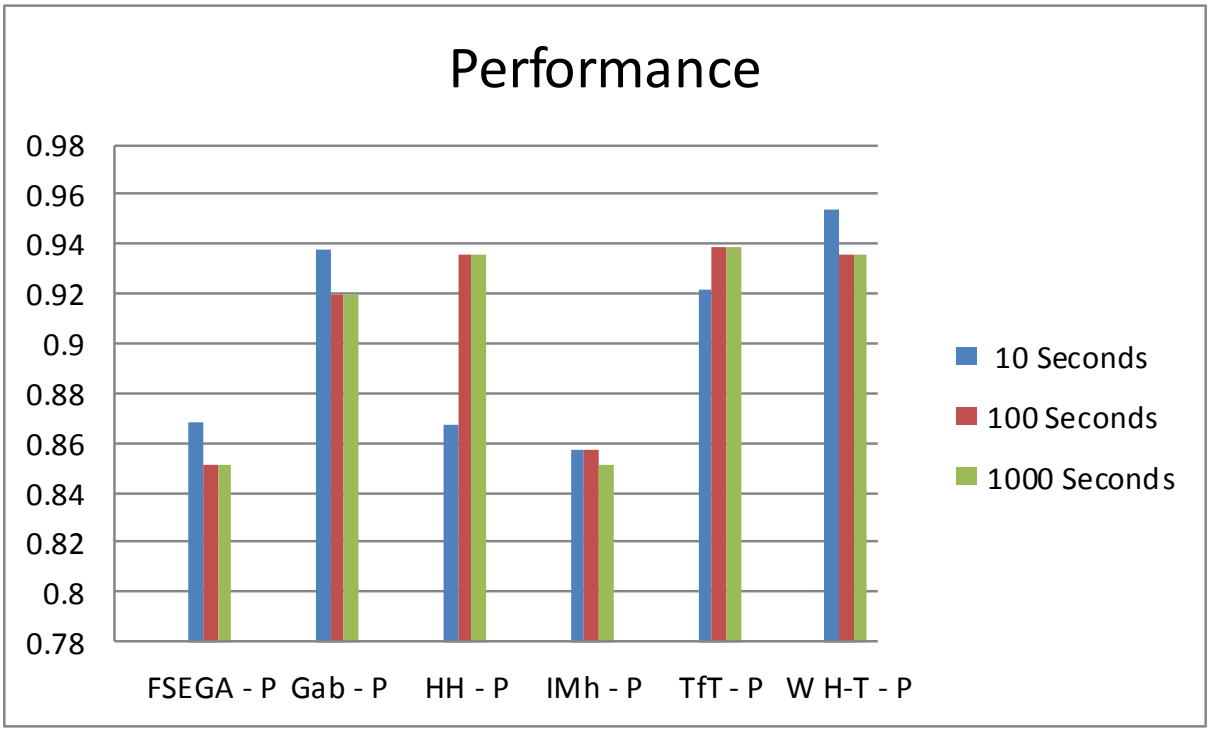


Figure 42: Agents' Performance for 10, 100 & 1000 Seconds

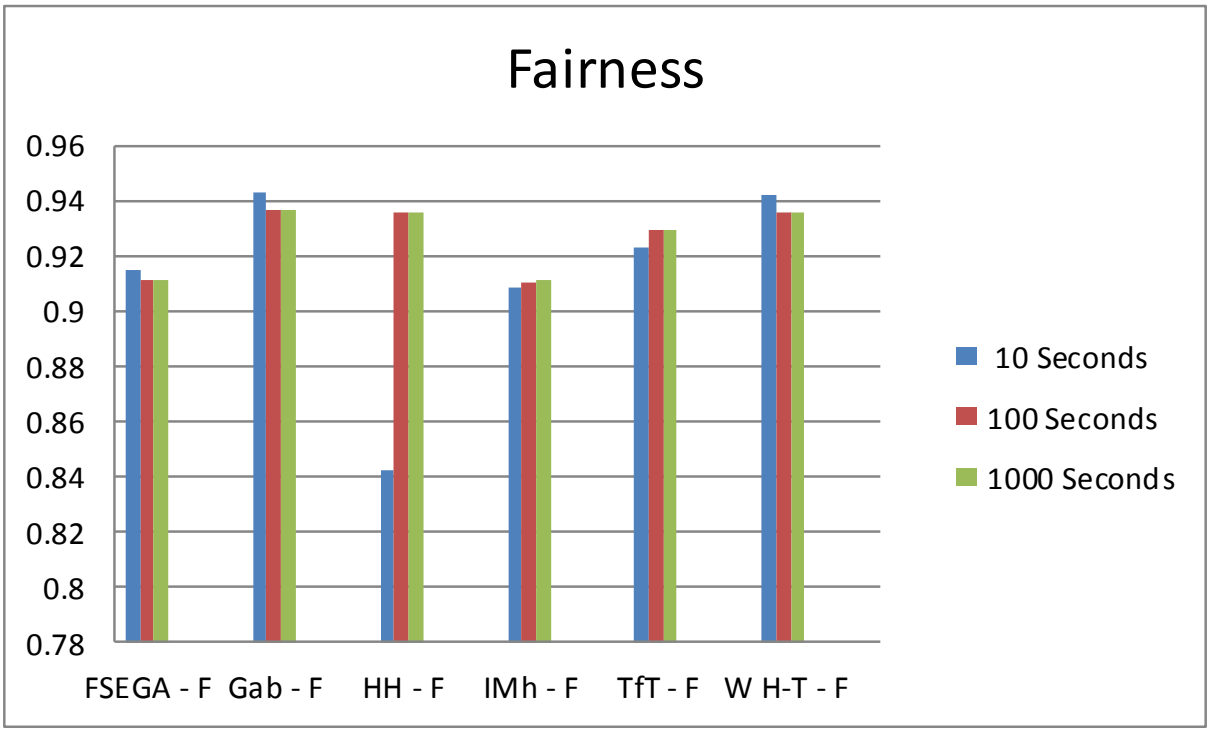


Figure 43 : Agents' Fairness for 10, 100 & 1000 Seconds

When the Deadline is 10 seconds, Wise H-T agent got the first place for Performance. Also, Wise H-T agent gets the second place for fairness. Gahboninho got the first place for the Fairness.

When the deadline has been changed to 100 seconds, Tit for Tat Agent got the first place for performance. Both Hardheaded and Wise H-T shared the second place. All the agents got the same results when the deadline is increased to 1000 seconds. Base on above results some discussions and recommendations will be given in the next section

8.3.2.1 Round-based deadline.

Table 14 shows the results of performance (P) and fairness (F) for each agent when I set the deadline to Round-based deadline.

Table 2 : Agents' Performance & Fairness for 10 and 100,Rounds

Agent	10 Rounds	100 Rounds
FSEGA – F	0	0.631952079
FSEGA – P	0.222210992	0.74515229
Gab – F	0.196013489	0.905217863
Gab – P	0	0.650341172
HH – F	0.207806072	0.717728808
HH – P	0.222205378	0.550091265
IMh – F	0.210346072	0.922403834
IMh – P	0.195804294	0.850723068
TfT – F	0.210346072	0.521085592
TfT – P	0.222205378	0.898112076
W H-T – F	0.315519111	0.922403834
W H-T – P	0.333308067	0.898112076

Table 15 : Agents' Performance & Fairness for 1000 and 10 000 Rounds

Agent	1000 Rounds	10000 Rounds
FSEGA – F	0.942575139	0.936155991
FSEGA – P	0.850717453	0.856947
Gab – F	0.909788206	0.909788204
Gab – P	0.937852504	0.850717453
HH – F	0.910674611	0.910674613
HH – P	0.86089763	0.867121562
IMh – F	0.927327333	0.927327334
IMh – P	0.856947	0.919180708
TfT – F	0.838032428	0.842602772
TfT – P	0.933863394	0.933863395
W H-T – F	0.935242133	0.935242131
W H-T – P	0.93545567	0.93545567

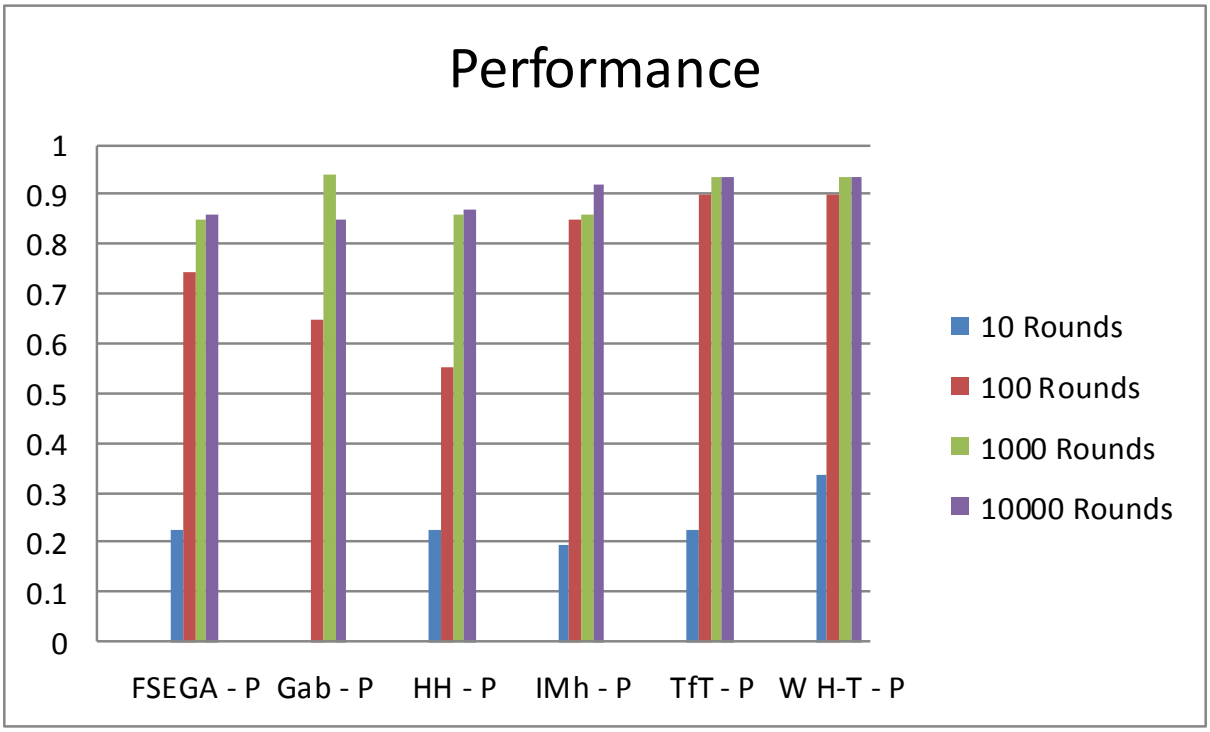


Figure 62: Agents' Performance for 10, 100, 1000 & 10 000 Rounds

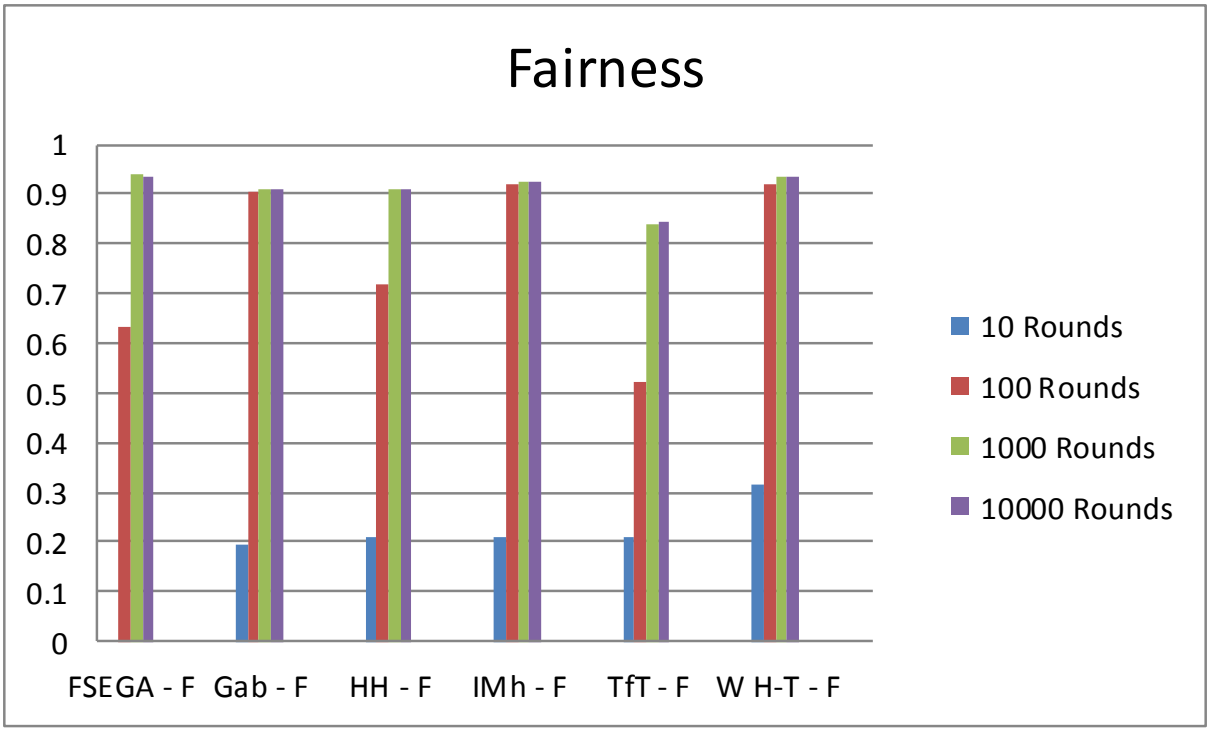


Figure 63 : Agents Fairness for 10, 100, 1000 & 10 000 Rounds

From the above results; I can see that when the deadline is 10 rounds, Wise H-T agent got the first place in terms of performance and fairness. When the deadline is 100 rounds, Wise H-T shared the first place with IMh in terms of fairness. For the performance, Wise H-T shared the first place with TfT. When the deadline is 1000 rounds, Gab got the first place and Wise H-T got the second place in terms of performance. When the deadline is 1000 rounds and 10000 rounds FSEGA got the first place and again Wise H-T the second place in terms of fairness. When the deadline is 10000 rounds wise H-T got the first place and TfT the second in terms of performance. Based on the above results some discussions and recommendations will be given in the next section.

8.3.3 Discussion

8.3.3.1 Time-Based Deadline.

The results show that Wise H-T is capable of doing well even if the deadline is short. HardHeaded did as well as the Wise H-T when the deadline was either 100 or 1000 seconds.

Hardheaded was the most affected agent when the time changed from 10 seconds to 100 or 1000 seconds. There is only a slight difference in the performance of a IMhaggler when the deadline was 100 seconds and 1000 seconds, where the rest of the agents have done exactly the same performance and fairness when the deadline was 100 or 1000 seconds

8.3.3.2 Round-Based-deadline.

Overall, all the agents did better when deadline was increased. When the rounds were increased up to 100, Tit for Tat and Wise H-T had the best performance and HardHeaded had the worst. Gahboninho had the best performance when the deadline was 1000 rounds, second place was the Wise H-T. When the deadline is 10000 rounds HardHeaded, Tit for Tat, AgentFSEGA and Wise H-T got exactly the same results as they got when the deadline was 1000 rounds. But, Wise H-T got the best results when the deadline was 10000 rounds.

The fairness, AgentFSEGA result was 0 and the best result was for Wise H-T agent. When the deadline was 100 rounds, IAMhaggler2011 and Wise H-T did the best and shared the first place, second agent was Gahboninho and the last one was Tit for Tat. All agents did exactly the same as they did when the deadline was 1000 or 10000. AgentFSEGA did slightly better at a 10000 rounds than the Wise H-T agent.

8.3.4 Recommendations:

At the end of this experiment, here are some recommendations that I would like to make.

8.3.4.1 Round-Based-deadline.

There is a big cost of setting the deadline to low number of rounds like 10 rounds or less, which is that most of negotiation sessions may end up with no agreement. The agents will do slightly better if the deadline is increased to 100 rounds. However, if the goal is to increase the overall outcomes of fairness and Performance then it is recommended to increase the deadline to 1000 rounds but no more than 1000 as it will not make any difference to the outcomes of the negotiations.

8.3.4.2 Time-based deadline.

There is a link between the two deadlines (round-based protocol and the time-based protocol); I found out that when the deadline was set up to 10 seconds the number of rounds that each negotiation session took between 2 and 2000 rounds. When the deadline was set up to 100 seconds, each negotiation session took between 4000 and 14000 rounds. However, when the deadline was increased to 1000 seconds, the number of rounds increased to be between 19000 and 23000.

This experiment shows that increasing the deadline to more than 10000 rounds or 100 seconds will not improve the negotiation outcomes.

8.4 Third experiment: (The Negotiation domain Experiment and The Negotiation deadline Experiment)

This experiment will try to answer the following questions:

Is there a relationship between the size of the negotiation domain and the deadline of the negotiation session?

If there is a relationship between the size of the negotiation domain and the deadline of the negotiation session, then would this affect the outcome of the negotiation, the performance of the agents and which agent would be affected the most?

If there is a relationship between the size of the negotiation domain and the deadline of the negotiation session, then would the results be affected if the deadline is switched between the round based protocol to a time based protocol?

8.4.1 Experiments setup.

In this experiment, each of the following agents; AgentFSEGA(**FSEGA)**, Gahboninho(**Gab)**, HardHeaded(**HH)**, Tit for Tat Agent (**TfT)**, IAMhaggler(**ImH)**, and Wise H-T(**W H-T)** will negotiate against each other including self-play which mean 36 negotiation sessions for each tournament. However, in this experiment will run 20 tournaments (4 different domain size multiply 5 different deadlines).Therefore, in this experiment I will run total of 720 negotiation sessions.

The following list shows the possible agreement ranges (as known as Domain) that have been investigated.

- 10 possible agreements.
- 100 possible agreements.
- 1000 possible agreements.
- 10000 possible agreements

Also, the deadline of the Negotiation session is investigated in order to study the effect of it to the negotiation outcome. The following list shows deadlines that have been instigated:

- 10 rounds.
- 100 rounds.
- 1000 rounds.
- 10 seconds.
- 60 seconds.

8.4.2 Round-based- Deadline.

8.4.2.1 The Deadline (10 rounds).

When the deadline is 10 rounds, all the negotiations end with no agreement. So I do not recommend setting the deadline to this low number of rounds. It seem that 10 rounds which is 5 rounds for each agents is not enough for the agents to end the negotiations with an agreement. This can be one of the future works to test more, why the agents are not able to perform.

8.4.2.2 The Deadline (100 rounds).

Here, I will highlight which agents are affected the most. Also, how the outcome is affected by increasing and decreasing the size of the domain known as all possible agreements (**pa**).

Table 16 : Agents' Performance 100 Rounds for different domain size.

100 rounds	10pa	100pa	1000pa	10000pa
FSEGA	0.233333	0.298333	0.278333	0.3
Gab	0.333333	0.333333	0.333333	0.3333333
HH	0.45	0.3	0.311667	0.3233333
TfT	0.3	0.35	0.3	0.2566667
Imh	0.283333	0.233333	0.228333	0.1366667
W H-T	0.383333	0.406667	0.356667	0.3333333

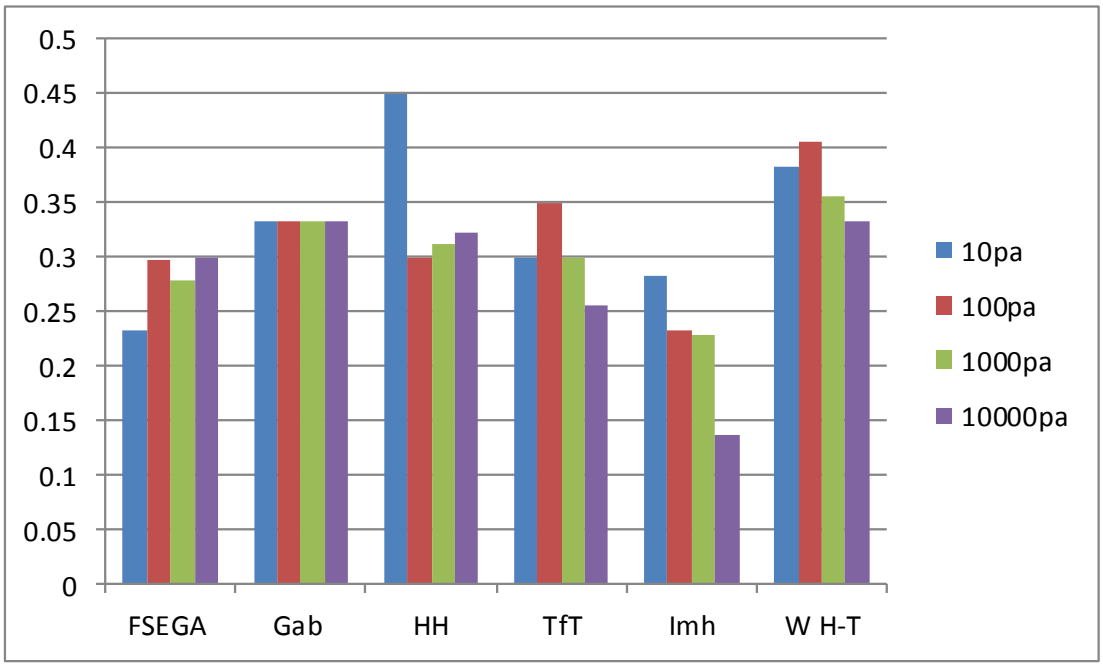


Figure 64: Agents' Performance 100 Rounds for different domain size.

The results show that when the domain size is 10pa, hardheaded (HH) got the best outcome then second place is W-HT. but , when the domain size is increased to 100pa W-HT came

first. Also, when the domain size increased to 1000pa W-HT come first when the domain size increased to 10000pa W-HT shared the first place with Gahboninho (Gab).

8.4.2.3 The Deadline (1000 rounds)

Overall the outcome is improved when the deadline is increased to 1000 rounds. Here I will highlight which agents are affected the most when the deadline is increased to 1000 rounds. Also, how the outcome is affected by increasing and decreasing the size of the domain.

Table 17: Agents' Performance 1000 Rounds for different domain size.

100 rounds	10pa	100pa	1000pa	10000pa
FSEGA	0.35	0.465	0.401667	0.5316667
Gab	0.55	0.606667	0.556667	0.5283333
HH	0.4	0.516667	0.58	0.445
TfT	0.383333	0.375	0.39	0.3416667
Imh	0.333333	0.416667	0.406667	0.3316667
W H-T	0.5	0.398333	0.485	0.4416667

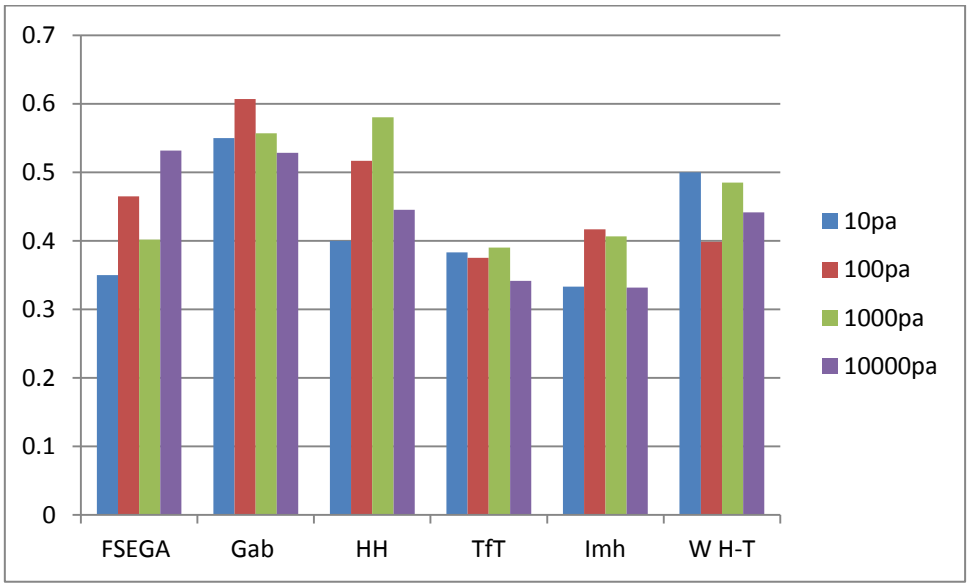


Figure 65: Agents' Performance 1000 Rounds for different domain size.

The results show that agent *Gab* got a first place when the size of the domain is 10 pa and 100 pa, when the domain increased to 1000 pa agent *HH* got the first place. When the domain size set to 10000 *FSEGA* got the First place.

8.4.2.4 Recommendation

At the end of this experiment, here some recommendation that can be made:

- I recommend using Wise H-T when the deadline is 100 rounds as the results .Wise H-T kept doing very well regardless to size of the domain.
- FSEGA shall be used when the deadline and the domain size is high and big. In the other hand, FSEGA shall not be used when domain is small and deadline is short. This agent needs time and space to show-off its capabilities.
- Increasing the size of the domain and decreasing the deadline (rounds) will decrease the overall Performance for all the agents.
- Increasing the size of the domain and increasing the deadline (rounds) will increase the overall Performance for all the agents.
- All the agents did they best when the domain size is 100pa and deadline is 1000 rounds , this is due to the enough rounds to explore all the possible agreement and to learn and predict the other side preferences.

8.4.3 Time-based- Deadline.

8.4.3.1 Deadline (10 seconds)

From table 18 and graph 66, it can be seen that agent hardheaded's (HH) performance improved when the domain size increased. FSEGA did poorly when domain size is 10 pa this is due to ending most the negotiation session with no agreement. But, when the domain size increased the performance improved. When the domain size is 10 pa agent Gab got the first place with high performance when compared to other agents. When the domain is large like 10000 pa, agent Imh gets the last place with a low performance. When the domain is 10pa, 100pa and 1000 pa, W H-T got the second place. But, when the domain size is 10000 pa W H-T got the third place.

Table 18: Agents' Performance 10 seconds for different domain size.

10 Seconds	10pa	100pa	1000pa	10000pa
FSEGA	0.116667	0.465	0.461667	0.516667
Gab	0.6	0.606667	0.625	0.62
HH	0.466667	0.49	0.546667	0.586667
TfT	0.45	0.45	0.395	0.158333
Imh	0.3	0.333333	0.351667	0.231667
W H-T	0.5	0.591667	0.608333	0.341667

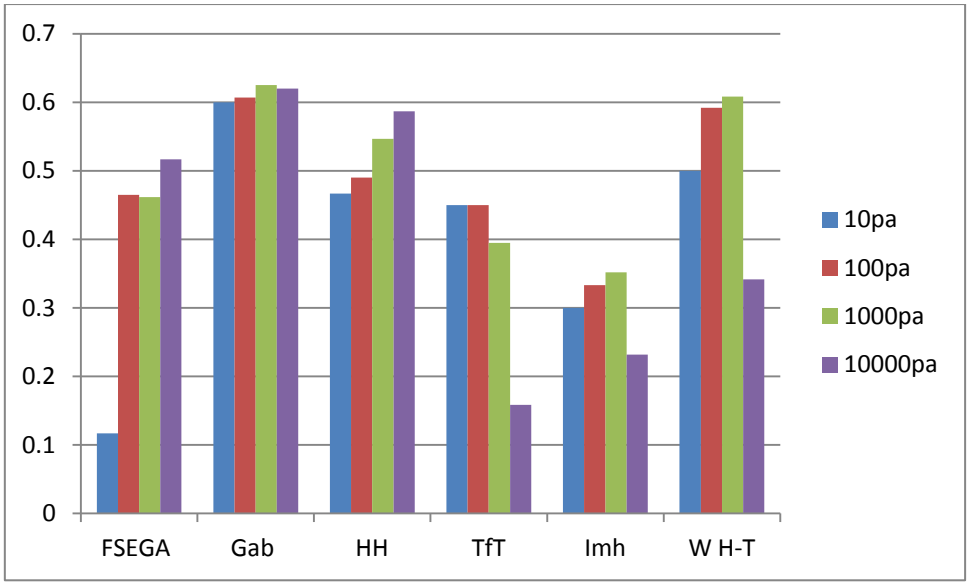


Figure 66 : . Agents' Performance 10 seconds for different domain size.

8.4.3.2 Deadline (60 seconds)

Form table 19 and the figure 67, it can be seen that agents W-H-T, HH and Ihm's performance did not change when the deadline is increased from 10pa to 100pa. The agent TfT is the agent that affected the most by increasing the domain size. The agent W-H-T is the only that kept the good performance regardless to domain size. Overall increasing the deadline increased the performance of some agents (FSEGA, Gah and HH) and decreased the performance of other agent (TfT, Imh and W H-T).

Table 19: Agents' Performance 60 seconds for different domain size.

60 Seconds	10pa	100pa	1000pa	10000pa
FSEGA	0.466667	0.373333	0.45	0.461667
Gab	0.533333	0.581667	0.5	0.59
HH	0.483333	0.483333	0.518333	0.533333
TfT	0.45	0.466667	0.491667	0.308333
Imh	0.416667	0.416667	0.44	0.416667
W H-T	0.55	0.548333	0.57	0.528333

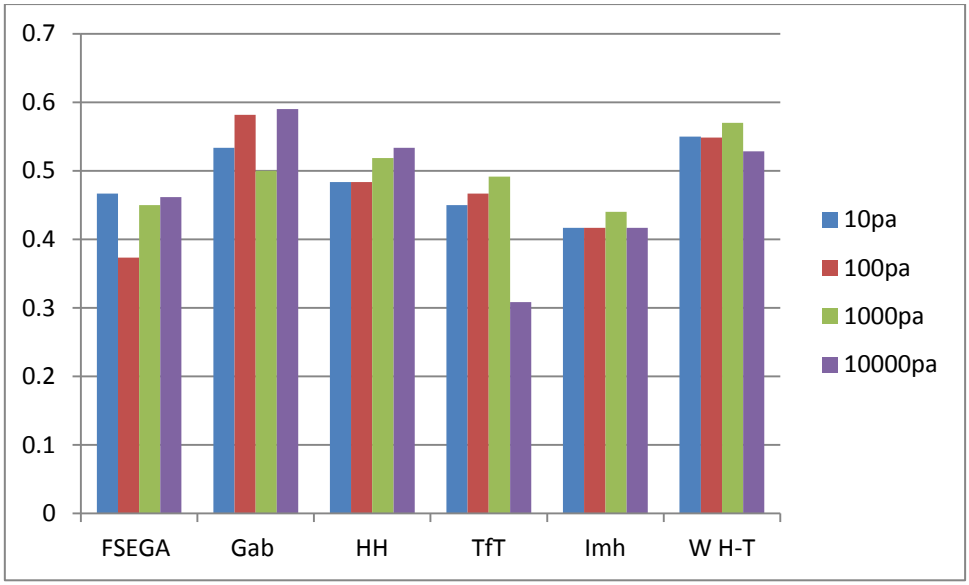


Figure 67: Agents' Performance 60 seconds for different domain size.

8.4.4 Improving the Overall Community Utilities:

Overall Community Utilities (OCU) is the sum of all the agents' utilities at each experiment .Here, I are trying to answer the following Questions:

- When all the agents will get higher utilities?
- Which deadlines will increase the Overall Community Utilities?
- Is increasing or decreasing the domain size will increasing Overall Community Utilities (OCU)?

First, I will study Round-based deadline then Time-based deadline.

8.4.4.1 Rounds-based-deadline (Overall Community Utilities):

Form table 20, it can be seen that the agents are not able to end the negotiations with agreement when the deadline is set to 10 rounds. But, when the deadline is increased to 100 rounds, the overall results improved due to ending the negotiations with agreement. Also,

when the deadline is increased to 1000 the overall results improved due to ending the negotiations with agreement and there were enough rounds for each agent to learn about each other preferences which led to agreement that satisfies both sides. But then again, when deadline is 1000 rounds and the domain size is 10000 possible agreements (pa) the resulted decreased, this due to that 1000 round is low number to explore a 10000 size domain.

Table 20 : Agents' overall OCU for 10, 100 and 1000 Rounds

	10pa	100pa	1000pa	10000pa
10 Rounds	0	0	0	0
100 Rounds	25.3	24.2	23.15	22
1000 Rounds	29.7	34	34.2	31.9

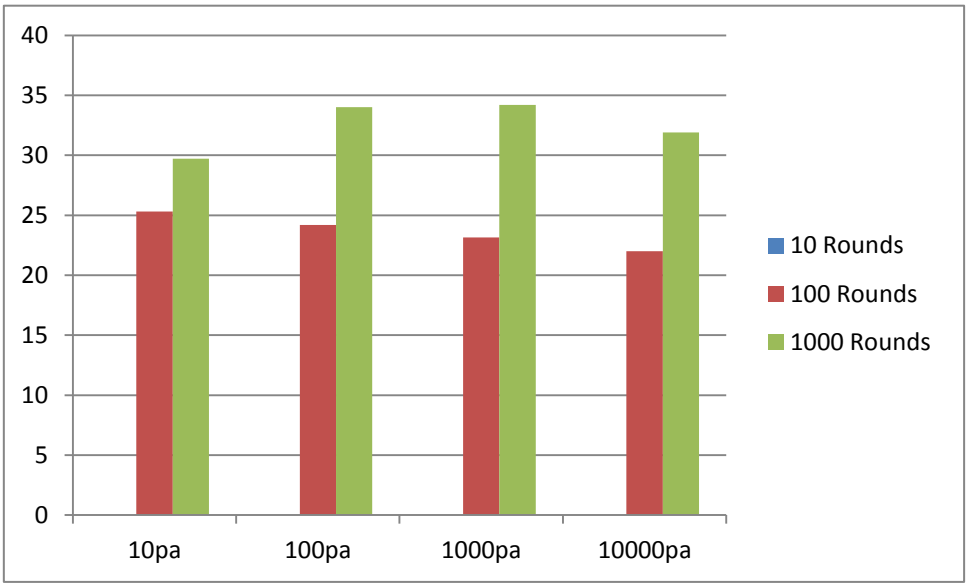


Figure 68: Agents' Overall Community Utilities for 10, 100 and 1000 Rounds

Form table 20 and figure 68, it can be seen that the agents are not able to end the negotiations with agreement when the deadline is set to 10 rounds so overall community utilities is 0. Then the overall community utilities improved when the deadline is increased.

8.4.4.2 Time-based- deadline (Overall Community Utilities)

In this part the effect of changing the deadline to time-based (seconds) will be studied. From table 21 and image 69, it can be seen that when the deadline increased from 10 seconds to 60 seconds only 10pa and 10000pa is effected positively. On the other hand, 100pa and 1000pa did not affected by increasing the deadline at all. I also can see that when the deadline is 60 seconds the outcome has not changed regardless to the domain size.

Table 21: Agents' Overall Community Utilities for 10 and 60 second

	10pa	100pa	1000pa	10000pa
10 seconds	30.8	35.1	35.3	29.69
60 seconds	35.2	35.1	35.3	35.1

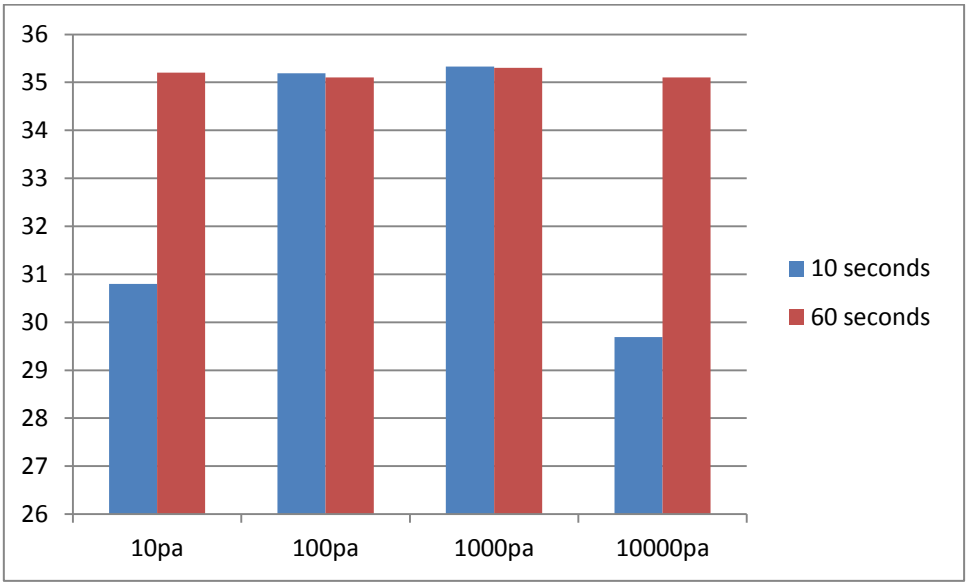


Figure 69: Agents' Overall Community Utilities for 10 and 60 second

8.4.5 Recommendations

At the end of this experiment, some recommendations can be made:

- Overall the performance for all the agents has increased when the deadline increased from 10 seconds to 60 seconds. So I recommend using 60 seconds than 10 seconds.
- W H-T is agent that kept the good performance regardless to domain size. So I recommend using W H-T when the domain size is unknown.
- When the deadline is 10s and the domain is 10000pa, some of the agents (HH, Gah and FSEGA) did better that other agents (Imh, TfT and W H-T).so . So I recommend using (HH, Gah and FSEGA) when the time is short and the domain is large.

8.5 **Fourth Experiment**: Price negotiation

Once the provider and the customer agreed about the package then they will start negotiating about the price for this package. For example let say they agreed about the next package:

[Availability Zone (location): Europe, Operating System: Linux , Term (months): [1-6], Memory GiB: 9, Compute : 4, Storage GB: [251- 500], Platform: 32-bit, Utilization: Medium <75%,])

The provider and the customer will be informed about the agreed package. After that, they will send back the range of the price as I discussed in (4.3.3.3 Create and update the price policy) .Then, the *Negotiable Space* will be found. See figure 70,

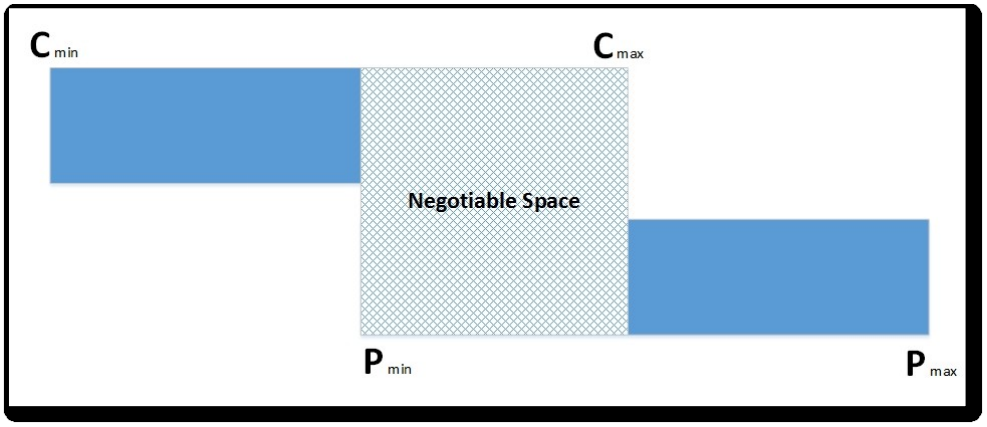


Figure 70 : negotiation Space.

For example, for the above package the *Negotiable Space* can be from 10 pounds to 20 pounds (10 possible agreements) or from 10 pounds to 50 pounds (40 possible agreements). For that, in this experiment, I want to study how each agent perform this kind of very competitive negotiation which is "Single Issue Negotiation". Figure 71 shows example of the (25 possible agreements) domain.

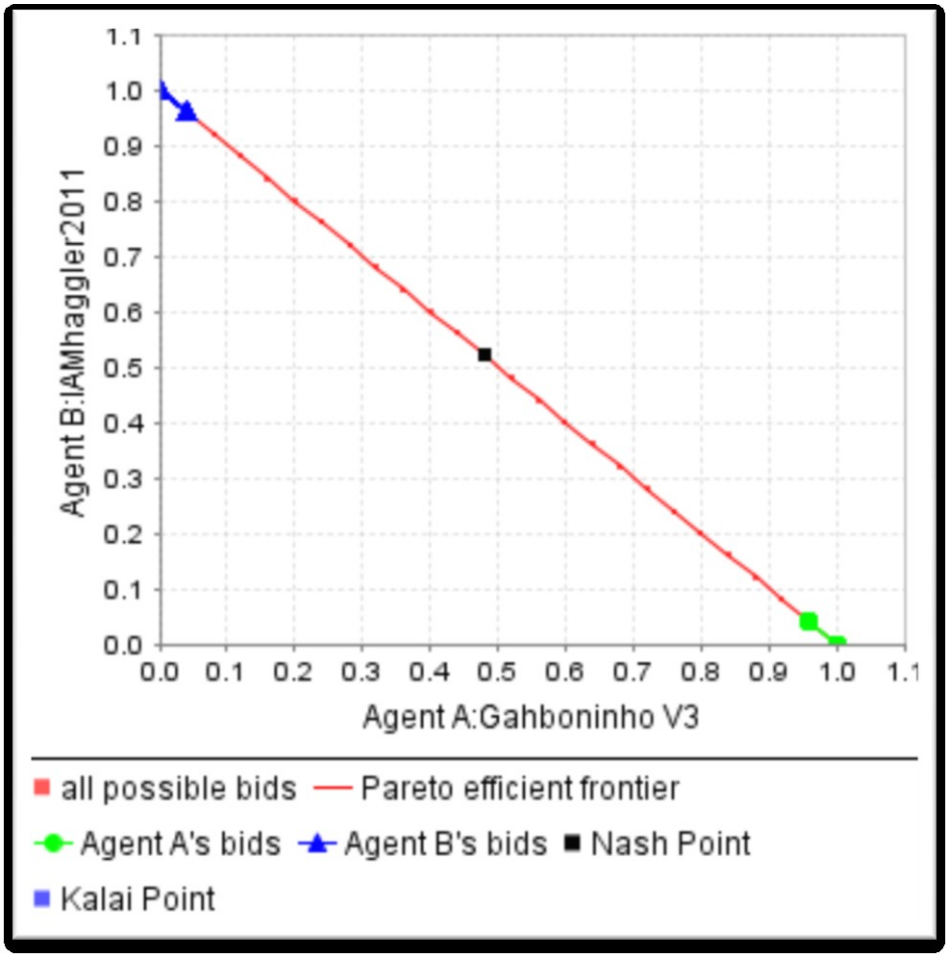


Figure 71 : Single Issue Negotiation

Figure 71 shows an example for 25 possible agreements domain.

So at the end of this experiment, I will be able to answer the following questions:

1. Would all the agents be capable to perform this kind of negotiation?
2. Would performance of each agent be affected while increasing and decreasing the deadline, as well as increasing and decreasing the size of the domain?
3. What should the deadline be set up at that by increasing the deadline would not lead to any improvement to the outcome?

8.5.2 Negotiable Space size and deadline experiment

The following experiment will investigate how the size of Domain (possible agreements) and the deadline will affect the performance of the agents in Single Issue Negotiation.

The following list shows the possible agreement ranges (as known as *Negotiable Space*/ Domain) that have been investigated.

- 10 possible agreements.
- 25 possible agreements.
- 50 possible agreements.
- 100 possible agreements.
- 200 possible agreements.

Also, the deadline of the Negotiation session is investigated in order to study the effect of it to the negotiation outcome.

The following list shows deadlines that have been instigated:

- 10 seconds.
- 60 seconds.
- 180 seconds.

The agents that are used to complete this experiment are the following: AgentFSEGA (FSEGA), Gahboninho (Gab), HardHeaded (HH), IMhaggler (IMh), Tit for Tat (TfT) and Wise H-T (W H-T).

Each agent negotiates with the other 5 agents. Hence there are 36 negotiation sessions in each experiment, meaning that in total there are 540 negotiation sessions (36*15= 540). The time that took to complete this experiment is 750 minutes (12.5 hours).

8.5.2.1 Agents' Performance with Single Issue Negotiation.

Here I will investigate the performance for each agent: first when the deadline is 10 seconds, then 60 seconds and lastly 180 seconds with the different sizes of the domains: 10, 25, 50, 100 and 200 possible agreements (See table 22 and figure 72). By running this experiment, I have noticed that the agent FSEGA and HardHeaded are not able to end this "Single Issue Negotiation" with an agreement.

8.5.2.1.1 Deadline (10 Seconds):

Table 22: Agents' Performance 10 seconds for different domain size.

10 seconds	10pa	25pa	50pa	100pa	200pa
Gahboninho V3	0.566667	0.566667	0.566667	0.416667	0.433333
Tit for Tat Agent	0.166667	0.166667	0.166667	0.166667	0.166667
IAMhaggler2011	0.366667	0.333333	0.366667	0.366667	0.365
WiseH-T	0	0	0	0	0

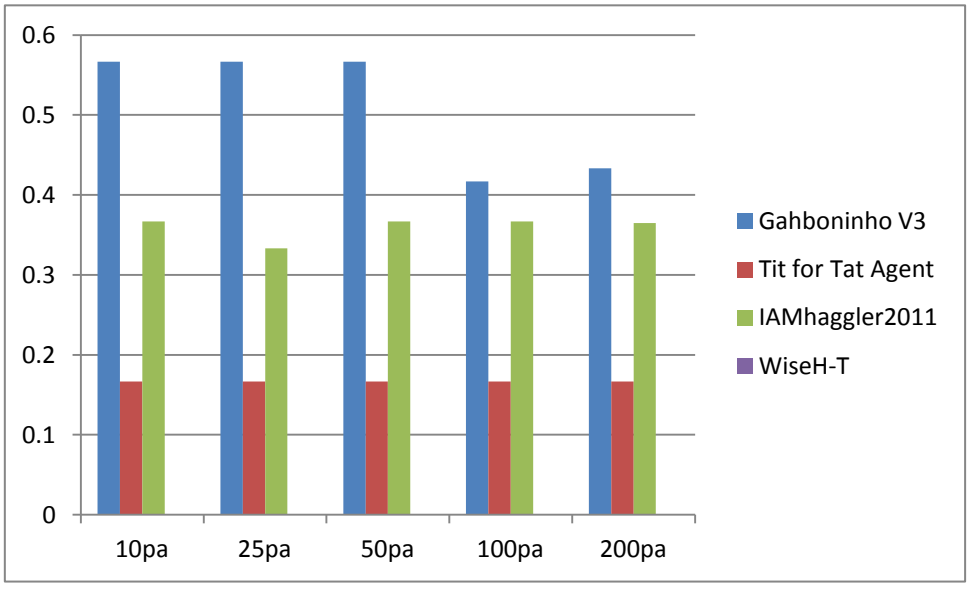


Figure 72: Agents' Performance 10 seconds for different domain size

After running this experiment I have noticed that the performance of Tit for Tat and Wise H-T did not change regardless to the size of the domain. I also noticed that the Gahboninho came first regardless to the domain size, however the performance decreased when the domain size got bigger (100 and 200 possible agreements). Even though Wise H-T received 0 results, it still managed to end negotiation with an agreement comparing to FSEGA and HardHeaded.

8.5.2.1.2 Deadline (60 Seconds):

Table 23: Agents' Performance 60 seconds for different domain size.

60 seconds	10pa	25pa	50pa	100pa	200pa
Gahboninho V3	0.533333	0.533333	0.533333	0.533333	0.533333
Tit for Tat Agent	0.166667	0.166667	0.166667	0.166667	0.166667
IAMhaggler2011	0.466667	0.466667	0.466667	0.466667	0.466667
WiseH-T	0	0	0	0	0

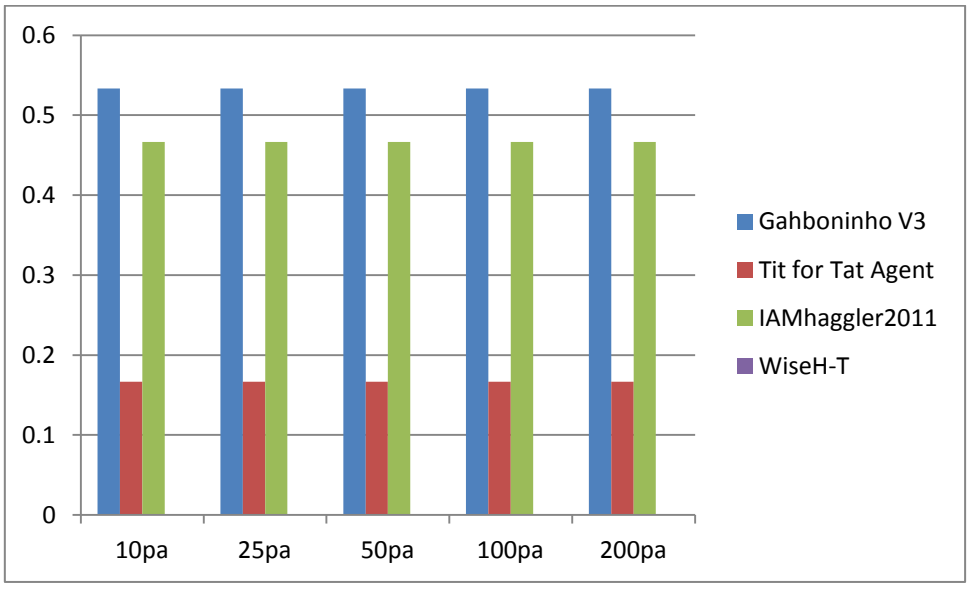


Figure 73: Agents' Performance 60 seconds for different domain size.

After running this experiment, it can be seen from table 23 and figure 73 that the agent Gahboninho received the first place; the IAMhaggler received seconds place; Tit for Tat receives third place and Wise H-T received the fourth place regardless to the size of the domain.

8.5.2.1.3 Deadline (180 Seconds):

Table 23: Agents' Performance 180 seconds for different domain size.

180 seonds	10pa	25pa	50pa	100pa	200pa
Gahboninho V3	0.533333	0.526667	0.53	0.53	0.528333
Tit for Tat Agent	0.166667	0.166667	0.166667	0.166667	0.166667
IAMhaggler2011	0.466667	0.466667	0.466667	0.466667	0.466667
WiseH-T	0	0	0	0	0

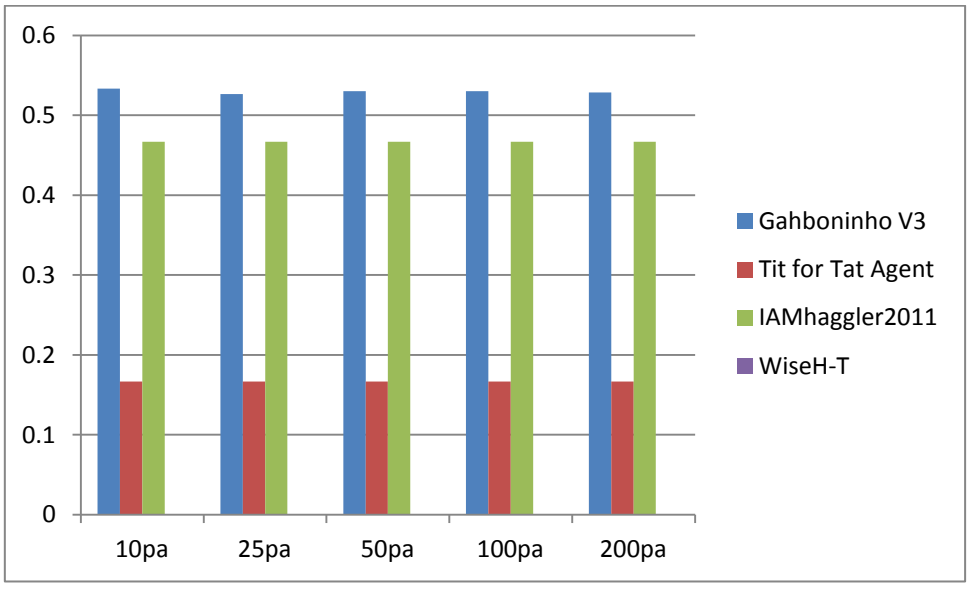


Figure 74: Agents' Performance 180 seconds for different domain size.

After running this experiment I have noticed that the result of this experiment is identical to the previous experiment of 60 seconds, therefore I can confirm that increasing the deadline more than 60 seconds will not improve the result.

8.5.3 Overall Community Utilities

Table 24 and figure 75 shows the OCU for each experiment at 10 seconds, at 60 seconds and at 180 seconds, where the sizes of the domains are 10, 25, 50, 100 and 200 possible agreements.

Table 24: Agents' OCU at 10, 60 and 180 seconds, for different domain size

	10pa	25pa	50pa	100pa	200pa
10second	15	14.5	14.9	13.9	13.9
60second	16	16	16	16	16
180second	16	16	16	16	16

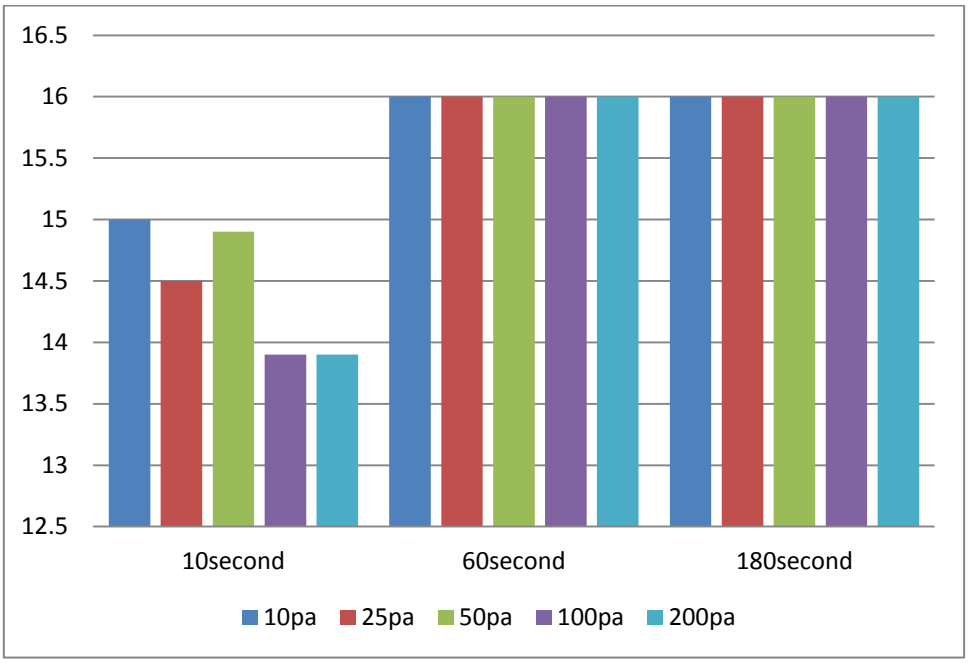


Figure 75: Agents' OCU all the 10 and 60 seconds and the domain size

It can be seen from table 24 and figure 75 that the OCU is improved when the deadline is increased from 10 seconds to 60 seconds, therefore when the deadline has been increased from 60 seconds to 180 seconds, the result did not change.

When the deadline was 10 seconds and the domain was very small then all the agents got the best results. On the other hand, when the deadline was 10 seconds and the domain was large (100 or 200 possible agreements) the OCU have decreased. Hence, I can recommend to set the deadline at 60 seconds to gain a better OCU regardless to the size of the domain as decreasing the deadline will affect the OCU negatively; consequently increasing the deadline more than 60 seconds will not increase the OCU.

8.5.3 Discussion:

After this experiment now I am able to answer the fourth experiment question.

1. *Would all the agents be capable to perform this kind of negotiation?*

No. After running this experiment, I can confirm that FSEGA and HardHeaded cannot perform this kind of experiment and end the negotiation with an agreement. This is due to a very stubborn approach of the negotiation that these agents use.

2. *Would performance of each agent be affected while increasing and decreasing the deadline, as well as increasing and decreasing the size of the domain?*

Yes. Regarding to the OCU, it can be improved when the deadline increased from 10 seconds to 60 seconds, after that increasing the deadline would not improve the situation. On the other hand, regarding to the performance, some agents (like IAMhaggler and Gahboninho) were affected by increasing and decreasing the deadline and the domain sizes; while other agents (like Tit for Tat and Wise H-T) were not affected.

3. *What should the deadline be set up at that by increasing the deadline would not lead to any improvement to the outcome?*

60 seconds regarding to the fairness and the performance.

Conclusion:

In this chapter four novel experiments are taken. Each experiment helped us to answer very important unanswered questions. First experiment will help the provider and customer to pick the right strategy for them. The second experiment will help us to experiment our agent (Wise H-T) against state-of-the-art agents. The third experiment helps us to understand how the domain size and deadline is affecting the negotiation outcome. Fourth experiment focused on price negotiation which is single Issue negotiation

At the end of the first experiment, I gave the following recommendations;

- The providers need to keep in mind that there is a competition. They need to be careful when they select or build the agent that represents them, selecting a very selfish agent is risky, as it might face a very selfish agent as well which will end the negotiation session to end with no agreement.
- I recommend providers to use Hardheaded (competitive strategy) when demand is higher than supply in the market. There is risk of ending some negotiations without agreement. However when the negotiating ends with an agreement, this agent will get the higher utility.
- Using cooperative agent can be recommended for providers who want to reach agreements effortlessly and attract new customers, e.g. new in the market providers or

old providers to promote new products. The agent will do its best to reach the best possible agreement for itself, but with low utility when it faces selfish agent.

At the end of the second experiment, I gave the following recommendations:

- There is a big cost of setting the deadline to low number of rounds like 10 rounds or less, which is that most of negotiation sessions may end up with no agreement. The agents will do slightly better if the deadline is increased to 100 rounds. However, if the goal is to increase the overall outcomes of fairness and Performance then it is recommended to increase the deadline to 1000 rounds but not more than 1000 as it will not make any difference to the outcomes of the negotiations.
- There is a link between the two deadlines (round-based protocol and the time-based protocol); I found out that when the deadline was set up to 10 seconds the number of rounds that each negotiation session took between 2 and 2000 rounds. When the deadline was set up to 100 seconds, each negotiation session took between 4000 and 14000 rounds. However, when the deadline was increased to 1000 seconds, the number of rounds increased to be between 19000 and 23000. This experiment shows that increasing the deadline to more than 10000 rounds or 100 seconds will not improve the negotiation outcomes.

At the end of the third experiment, I gave the following recommendations:

- I recommend using Wise H-T when the deadline is 100 rounds as the results .Wise H-T kept doing very well regardless to size of the domain.
- FSEGA shall be used when the deadline and the domain size is high and big. In the other hand, FSEGA shall not be used when domain is small and deadline is short. This agent needs time and space to show-off its capabilities.

- Increasing the size of the domain and decreasing the deadline (rounds) will decrease the overall Performance for all the agents.
- Increasing the size of the domain and increasing the deadline (rounds) will increase the overall Performance for all the agents.
- All the agents did they best when the domain size is 100pa and deadline is 1000 rounds , this is due to the enough rounds to explore all the possible agreement and to learn and predict the other side preferences.
- Overall the performance for all the agents has increased when the deadline increased from 10 seconds to 60 seconds. So I recommend using 60 seconds than 10 seconds.
- W H-T is agent that kept the good performance regardless to domain size. So I recommend using W H-T when the domain size is unknown.
- When the deadline is 10s and the domain is 10000pa, some of the agents (HH, Gah and FSEGA) did better that other agents (Imh, TfT and W H-T). So I recommend using (HH, Gah and FSEGA) when the time is short and the domain is large.

At the end of the fourth experiment, I gave the following recommendations:

- After running this experiment, I can confirm that FSEGA and HardHeaded cannot perform this kind of experiment and end the negotiation with an agreement. This is due to a very stubborn approach of the negotiation that these agents use.
- Regarding to the Overall Community Utilities (OCU), it can be improved when the deadline increased from 10 seconds to 60 seconds, after that increasing the deadline would not improve the situation. On the other hand, regarding to the performance, some agents (like IAMhaggler and Gahboninho) were affected by increasing and decreasing the deadline and the domain sizes; while other agents (like Tit for Tat and Wise H-T) were not affected.

CHAPTER 9

CONCLUSION

This chapter will summarize this thesis. First of all, the conclusion will summarize the motivations of this work. Afterwards, the contribution to the knowledge will be highlighted. And lastly, the future work will be proposed for any further research projects.

9.1 Motivations Summary

There were many different kind of motivations behind doing this project. At the very beginning of this project there was a lack of an actual practical work in this field. The reason behind this was because cloud computing was in the very early stage of its existence. Another motivation was that the solutions (that were offered as the practical works) were "borrowed" from other technologies related to the cloud computing, for example Grid Computing. Furthermore, to the best of our knowledge, a complete SLA life cycle, which would be made especially for cloud computing, did not exist and there was a need to create a new SLA life cycle which would fit the dynamic notion and heterogeneity of cloud computing, taking into account every single step of SLA life cycle in details. The second motivation is that there were some works which emphasized the need for negotiation in the Cloud Computing. However, none of them provided a practical solution that considers the negotiation that

needs to support multi-issue negotiation with the possibility of giving preferences for each issue and that the negotiation needs to be automated, meaning it has to be run by an intelligent agent with different negotiation strategies.

During the development of this research, I have faced different challenges which will be discussed in this section. There is a lack of standardization in the cloud computing, meaning that there is no agreed pattern to follow, no agreed rules. Another challenge that I have faced that was the lack of cloud computing simulation tools. Finally, in the beginning of this work cloud computing was in the early stage of its development, hence during the development of this work many research works and commercial products were proposed to the market. For that, it was essential to keep the research up-to-date by reviewing the state-of-the-art technologies.

9.2 Contribution to knowledge

This work contributed to knowledge both in the Cloud Computing field and in automated negotiation field:

Cloud computing field:

- This work proposed a novel framework for total automated SLA life cycle management especially for could computing by using rational agents to perform the automated negotiation stage instead of humans (Alsrheed, et al., 2013).
- The flexibility of the framework in the sense that any agents can be improved or a new agent could be created in the future will fit into this framework and compete against the existing agents.
- Making recommendations about the use of the negotiation strategy as well as when and how each strategy should be used in the cloud computing, taking into account the supply and demand factors (Alsrheed, et al., 2013).
- In this work, the vision and the principles of autonomic computing by using autonomic control loop (Collect, Analyze, Decide and Act) are used to design and implement our solution for monitoring stage. I presented the implementation of our solution via a scenario of over-promising and under-delivering problem; I also demonstrated the idea of live virtual machine migration using cloud computing simulation.

Automated negotiation field

- This work proposed a novel agent, named Wise H-T, which combines and balances between cooperative and competitive behaviours, which leads to better negotiation outcomes from other agents(Alsrheed, et al., 2014a).

- This work investigated the effect of increasing and decreasing the size of the domain (number of possible agreements) to the negotiation outcome. (Alsrheed, et al., 2014a).
- The related works only study time–based protocol where the deadline is 180 seconds. However, in this work I investigated when the deadline is less or more than 180 seconds. Also, in this work I investigated the effect of increasing and decreasing the deadline of the negotiation to the outcome. (Alsrheed, et al., 2014a). I also investigated the effect of switching between the rounds-based protocol and time-based deadlines protocol of the negotiation to the outcome. (Alsrheed, et al., 2014a).

9.3 Limitations

The current research work completed for this thesis has the following limitations:

9.3.1 Independent issues:

In this work, the issues are independent from each other. on other words the customer need to pick one value from each value. See table 25.

Table25: Issues and Value for VM

Issue	Value
Availability Zone (location)	US-East US-West Europe
Operating System	RedHat Linux Ubuntu Oracle Linux
Term (months)	[1-6] [7-12]
Memory GiB	7 8 9 10
Compute Units /virtual core (CPU)	4 5 6 7
Storage GB	[251- 500] [501- 725]
Platform	32-bit 64-bit
Utilization	Light < 39% Medium <75%

So, what about if there are different values dependent on the previous Issue. For example, if the customer pick the value “Europe” from “Availability Zone” Issue , the Operating System’s Values will change which means that the provider can offer different”Operating System”dependent on the “Availability Zone”.

9.4 Proposed Future Works / Trends

9.4.1 Negotiation without a deadline

Negotiation without deadline might lead to never end negotiation, but Dose this applies to all the Agents? There are some unanswered questions about the effect of not having deadline to the outcome of negotiation;

1. Which agent that can reach a better outcome with open-end negotiation?
2. Is the overall Utilities for both Agents will be improved without a deadline?

9.4.2 Improving Wise H-T.

There are 11 opponent models (Baarslag, et al., 2013b), as I mentioned in section (2.6.1). So, more work need to be done to improve Wise H-T in terms of opponent modelling. There are some unanswered questions about the effect of changing the opponent models to agent *Performance* and *Fairness*:

- Is changing the opponent modelling will improve the agent *performance* and *Fairness* and when, in terms of domain size and deadline?

9.4.3 Cloud Robotics.

Cloud robots are the next generation robots which are not limited by on-board computation and memory. The concept of "*remote-brain*" is first introduced by Inaba from the University of Tokyo in 1996 (Inaba, 1996). Then again, in 2010, James Kuffner at Google introduced the term "*Cloud Robotics*" (Goldberg and Kehoe 2013). Cloud robots will be able of offloading computationally intensive tasks to the cloud for fast performance.

I believe that our work in (Alsrheed , et al., 2013) can be very useful for SLA Negotiation Cloud robots, Since Cloud robots need to negotiate to use the cloud provider's resources. Cloud robot can be customer looking for providers for big task offloading, or Cloud robot can be robot-a-service (RaaS) acting as "provider", in both cases a negotiation is needed. The negotiation is automated and executed by intelligent agents.

9.4.4 Using discovery, migration and communications protocols.

Using discovery and migration protocols and extensible messaging and presence protocol (XMPP) is one of the future trends of this work. The discovery and emigrate protocols will be useful to find the cloud providers and cloud customers and to migrate the cloud customers from one cloud to another without difficulty. Using the extensible messaging and presence protocol (XMPP) will make the communications between the agents more suitable for the cloud computing environment because XMPP is based on decentralization. XMPP is open standards and extensible which mean that Rubinstein's alternating offer protocol can be built on top of it. In this way, the Agents' makers (cloud providers and cloud customers) will be focusing on improving the agents' algorithms rather than how the agent will communicate with others.

References

Alessio R. Lomuscio, Michael Wooldridge, and Nicholas R. Jennings (2003) “A classification scheme for negotiation in electronic commerce”. Group Decision and Negotiation, Volume 12, pages 31-56.

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

Alhamad, M.; Dillon, T.; Chang, E.(2001) "Service Level Agreement for Distributed Services: A Review," Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference, pages.1051,1054.

Alsrheed faisal , Abdennour El Rhalibi, Martin Randles, Madjid Merabti (2013) “Rubinstein's Alternating Offers Protocol for Automated Cloud Computing Negotiation”, The 14th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2013) Liverpool, United Kingdom.

Alsrheed Faisal, Abdennour El Rhalibi, Martin Randles, Madjid Merabti (2014a), “Intelligent Agents for Automated Cloud Computing Negotiation, the 4th International Conference on Multimedia Computing and Systems (ICMCS'14), Marrakesh, Morocco.

Alsrheed Faisal, Abdennour El Rhalibi, Martin Randles, Madjid Merabti (2014b), “Automated Service Level Agreement Negotiation for Cloud Robotics, 7th Saudi Students Scientific Conference-UK, University of Edinburgh,United Kingdom .

Amazon Elastic Compute Cloud (Amazon EC2),(2014) [online] Available at:< https://aws.amazon.com/ec2/> [Accessed 15-10-2014].

Ariya, T. K., Paul, C., & Karthik, S. (2012). “Cloud Service Negotiation Techniques computer, Volume 1, Issue 8.

Automated Negotiating Agents Competition (ANAC), (2010) [online] Available at:<http://mmi.tudelft.nl/negotiation/index.php/Automated_Negotiating_Agents_Competition_(ANAC> [Accessed 15-10-2014].

Axelrod, Robert (1984), “The Evolution of Cooperation”, Basic Books, ISBN 0-465-02122-0

Baarslag, T., Fujita, K., Gerding, E. H., Hindriks, K., Ito, T., Jennings, N. R., ... & Williams, C. R. (2013). “Evaluating practical negotiating agents: Results and analysis of the 2011 international competition”. Artificial Intelligence, Volume 198, pages 73-103.

Baarslag, T., Hendrikx, M., Hindriks, K., & Jonker, C. (2013b). “Predicting the performance of opponent models in automated negotiation”. InWeb Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on Volume 2, Pages 59–66.

Baarslag, T., Hindriks, K., & Jonker, C. (2011b). Acceptance conditions in automated negotiation. In Complex Automated Negotiations: Theories, Models, and Software Competitions ,Pages 95-111.

Baarslag, T., Hindriks, K., Hendrikx, M., Dirkzwager, A., & Jonker, C. (2012). Decoupling negotiating agents to explore the space of negotiation strategies. InNovel Insights in Agent-based Complex Automated Negotiation pages. 61-83.

Baarslag, Tim, Koen Hindriks, and Catholijn Jonker (2013a). "A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiations." In Complex Automated Negotiations: Theories, Models, and Software Competitions, pages. 229-233.

Beam, C &Segev, A (1997) "Automated Negotiations: A Survey of the State of the Art".Wirtschaftsinformatik, Volume 39, Issue 3, Pages 263–268.

Binmore K (1992) Fun & Games: a text on Game Theory, D.C. Heath and Company, Lexington, MA

Binmore, K., & Vulkan, N. (1999)."Applying game theory to automated negotiation". Netnomics, Volume 1, Issue 1, Pages 1–9.

Bosse, T &Jonker, C.M (2005) "Human vs. Computer Behavior in Multi-issue Negotiation". In Rational, Robust, and Secure Negotiation Mechanisms in Multi-Agent Systems, pages 11–24.

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

C. E. Rasmussen & C. K. I. Williams (2006), "Gaussian Processes for Machine Learning, the MIT Press.

C. Li, J. Giampapa, and K. Sycara (2003). "A review of research literature on bilateral negotiations", Technical report, Carnegie Mellon University Robotics Institute, Pittsburgh, PA.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". Software: Practice and Experience, Volume 41, Issue 1, Pages 23–50.

Chavez, A., Dreilinger, D., Guttman, R., & Maes, P. (1997). "A real-life experiment in creating an agent marketplace". In Software Agents and Soft Computing Towards Enhancing Machine Intelligence, Pages 160-179.

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.

Chhetri, M. B., Mueller, I., Goh, S. K., & Kowalczyk, R. (2007). "ASAPM-An Agent-Based Framework for Adaptive Management of Composite Service Lifecycle", In Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops, Pages. 444-448.

Clancey, W. J (1989) “The Frame of Reference Problem in Cognitive Modeling”.In: Proceedings of The Eleventh Annual Conference of the Cognitive Science Society, Pages. 107-114.

Colin R. Williams, Valentin Robu, Enrico H. Gerding, Nicholas R. Jennings (2013) “IAMhaggler2011: A Gaussian Process Regression Based Negotiation Agent, Complex Automated Negotiations”: Theories, Models, and Software Competitions Studies in Computational Intelligence,Volume 435, Pages 209-212.

Colin R. Williams, Valentin Robu, Enrico H. Gerding, Nicholas R. Jennings: (2014) “An Overview of the Results and Insights from the Third Automated Negotiating Agents Competition (ANAC2012)”. Novel Insights in Agent-based Complex Automated Negotiation Pages 151-162.

Dana Petcu, Georgiana Macariu, Silviu Panica, Ciprian Crăciun,(2013) Portable Cloud applications—From theory to practice, Future Generation Computer Systems, Volume 29, Issue 6, , Pages 1417-1430.

Dinesh, V. (2004). “Supporting Service Level Agreements on IP Networks”. In Proceedings of IEEE/IFIP Network Operations and Management Symposium, Volume 92, Issue 9, , Pages 1382-1388.

Dirkzwager, A.S.Y(2013) Towards Understanding Negotiation Strategies: Analyzing the Dynamics of Strategy Components, Delft University of Technology, Electrical Engineering, Mathematics and Computer Science, Master thesis

Dobson, S and Sterritt, R and Nixon, P and Hinchey, (2010). M, Fulfilling the vision of autonomic computing, Computer, Volume 43, Issue 1, Pages 35-41.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2002). Multi-issue negotiation under time constraints. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems. pages 143-150.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2002). “Optimal negotiation strategies for agents with incomplete information”. In Intelligent Agents ,pages 377-392.

Fatima, S.S., Wooldridge, M., Jennings, N.R. (2002) Optimal Negotiation Strategies for Agents with Incomplete Information. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), volume 2333, Pages 377–392.

Fatima,S.S, Wooldridge, M & Jennings, N.R (2002) Multi-issue negotiation under time constraints. In Proc. of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pages 143–150.

Filzmoser M. (2010): Automated vs. human negotiation, International Journal of Artificial Intelligence,Volume 4, Issue 10, Pages 64–77.

Fisher, R., and Ury, W.L., and Patton, B. (2003). “Getting to Yes: Negotiating Agreement Without Giving In”. Penguin Books.

Fullam, Karen K., Tomas B. Klos, Guillaume Muller, Jordi Sabater-Mir, Zvi Topol, K. Suzanne Barber, Jeffrey Rosenschein, and Laurent Vercouter.(2005)"The Agent Reputation

and Trust (ART) Testbed Architecture." In Proceedings of the 2005 conference on Artificial Intelligence Research and Development, Pages 389-396.

G.C. Silaghi, L.D. Serban, and C.M. Litan (2010) “A framework for building intelligent sla negotiation strategies under time constraints”. In Proceedings of Economics of Grids, Clouds, Systems, and Services: 7th International Workshop, volume 6296, page 48.

Gatti, N. (2009). Extending the alternating-offers protocol in the presence of competition: models and theoretical analysis. Annals of Mathematics and Artificial Intelligence Volume 55, Issue 3, pages 189-236

Gerding, E. H., van Bragt, D. D. B., & La Poutré, J. A. (2000). “Scientific approaches and techniques for negotiation: a game theoretic and artificial intelligence perspective”. Centrum Wiskunde & Informatica.

Gerding, E.H, Van Bragt, D.D.B, La Poutre, J.A & Centrum voor Wiskunde en Informatica (2000) Scientific Approaches and Techniques for Negotiation: A Game Theoretic and Artificial Intelligence Perspective.Citeseer

Gheorghe Cosmin Silaghi, Liviu Dan Şerban, Cristian Marius Litan,(2012) A time-constrained SLA negotiation strategy in competitive computational grids, Future Generation Computer Systems, Volume 28, Issue 8, Pages 1303-1315.

Gheorghe CS,Liviu D Şerban, Cristian M L (2012), A time- constrained SLA negotiation strategy in competitive computational grids, Future Generation Computer Systems, Volume 28, Issue 8

Goldberg, K., & Kehoe, B. (2013). “Cloud robotics and automation: A survey of related work”. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5.

Guttman, R. H., Moukas, A. G., & Maes, P. (1998). “Agent-mediated electronic commerce: a survey”. The Knowledge Engineering Review, Volume 13, Issue 2, pages 147-159.

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

Hauswirth, M., Jazayeri, M., Miklós, Z., Podnar, I., Di Nitto, E., & Wombacher, A. (2001). “An architecture for information commerce systems”. In: Proceedings of the Sixth International Conference on Telecommunications (ConTEL)

Hendrikx, M. J. C. (2012). “Evaluating the Quality of Opponent Models in Automated Bilateral Negotiations” , PhD thesis , Delft University of Technology.

Hindriks, K &Tykhonov, D (2008) “Opponent Modeling in Automated Multi-issue Negotiation Using Bayesian learning”. In Proc. of the Seventh International Joint Conference on Autonomous Agents and Multi-agent Systems, pages 331–338

Hindriks,K,Jonker, C.M &Tykhonov, D (2009) The Benefits of Opponent Models in Negotiation. In Proc. of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Volume 2, pages 439–444

Jadeja, Y & Modi, K (2012) “Cloud Computing - Concepts, Architecture and Challenges”, International Conference on Computing, Electronics and Electrical Technologies [ICCEET], India

Jazayeriy, H, Azmi-Murad, M, Sulaiman, M.N and N.I. Udzir (2001) “A Review on Soft Computing Techniques in Automated Negotiation”. Scientific Research and Essays, Volume 6, Issue 24, Pages 5100–5106.

Jennings, N.R, Faratin, P, Lomuscio, A.R, Parsons, S, Wooldridge, M.J & Sierra, C (2001) “AutomatedNegotiation: Prospects, Methods and Challenges. Group Decision and Negotiation”, Volume 10, Issue 2, Pages 199–215.

Jin, L. J., & Machiraju, V. A. (2002). “Analysis on Service Level Agreement of Web Services”. Technical Report HPL-2002-180, Software Technology Laboratories, HP Laboratories.

John B, Nigar H, and Gareth Myles (2009), “A Dictionary of Economics” (3 ed.) Oxford University Press.

K. Hindriks and D. Tykhonov (2008) “Opponent modelling in automated multi-issue negotiation using bayesian learning”. In Proc. of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, pages 331–338.

K. Hindriks, C. M. Jonker, and D. Tykhonov (2009) The benefits of opponent models in negotiation. In Proc. of the 2009IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, volume 2, pages 439–444.

Kahneman, D and Tversky, A (1979) “Prospect Theory: An Analysis of Decision under Risk” Econometrica volume. 47, Issue.2, pages 263-292.

Karaenke, P., & Kirn, S. (2010).”Towards model checking & simulation of a multi-tier negotiation protocol for service chains”. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 pages 1559-1560.

Ken binmore (1992) “Fun and games a text on game theory”, Houghton Mifflin.

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, volume. 1, Issue.3, pages 293-315

Kersten, G. E., & Noronha, S. J. (1999). WWW-based negotiation support: design, implementation, and use. Decision Support Systems. volume. 25, Issue.2, pages 135-154.

Khalid, A., Haye, M. A., Khan, M. J., & Shamail, S. (2009). “Survey of frameworks, architectures and techniques in autonomic computing. In Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on pages 220-225.

Kowalczyk, R., Ulieru, M., & Unland, R. (2002). Integrating mobile and intelligent agents in advanced E-commerce: a survey. In Agent Technologies, Infrastructures, Tools, and Applications for E-Services pages 295-313.

Kraus, S (2001) Strategic Negotiation in Multi-agent Environments.MIT Press, Cambridge, MA.

Kraus, S, Wilkenfeld, J &Zlotkin, G (1995) Multi-agent negotiation under time constraints. Artificial Intelligence, volume 75, Issue 2, pages 297-345.

Kraus, S. (1997). Negotiation and cooperation in multi-agent environments.Artificial Intelligence volume. 94, Issue.1, pages 79-97.

Kraus, S. (2001) "Automated negotiation and decision making in multiagent environments." Multi-agent systems and applications. Springer Berlin Heidelberg, pages 150-172.

Kraus, Sarit (1997) "Negotiation and cooperation in multi-agent environments."Artificial Intelligence, pages 79-97.

Kumar, M & Feldman, S (1998) Internet auctions. Technical report, IBM Research

KVM, 2013. Kernel-based Virtual Machine. [Online] Available at: <http://www.linux-kvm.org > [Accessed 16 October 2014].

Kwang Mong Sim (2012) "Agent-Based Cloud Computing," Services Computing, IEEE Transactions volume.5, Issue.4, pages.564,577.

Liang, Y. Q., & Yuan, Y. (2008). Co-evolutionary stability in the alternating-offer negotiation. In Cybernetics and Intelligent Systems, 2008 IEEE Conference on pages 1176-1180.

Liang, Y. Q., & Yuan, Y. (2008). Co-evolutionary stability in the alternating-offer negotiation. In Cybernetics and Intelligent Systems, 2008 IEEE Conference on ,pages. 1176-1180.

Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K. and Jonker, C. M. (2012), “Genius: An Integrated Environment For Supporting The Design Of Generic Automated Negotiators”. Computational Intelligence. Pages 1467-8640.

Linlin Wu and Buyya R. (2010). Service Level Agreement (SLA) in Utility Computing Systems. Tech. Rep.

Linlin Wu; Garg, S.K.; Buyya, R.; Chao Chen; Versteeg, S., (2013) "Automated SLA Negotiation Framework for Cloud Computing," Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on Pages.235,244.

Liu Kexing,(2011) "A Survey of Agent Based Automated Negotiation," International Conference on Network Computing and Information Security.volume.2,pages.24-27.

Liu, K. (2013) “A Model of Protocol for Automated Negotiation System”. In Proceedings of the 2nd International Conference on Green Communications and Networks 2012 (GCN 2012), Volume 3, pages 407-412.

Lomuscio, A.R, Wooldridge, M & Jennings, N.R (2003) “A Classification Scheme for Negotiation in Electronic Commerce”. Group Decision and Negotiation, Volume 12, pages 31–56.

M. Inaba (1996) “Remote-brained humanoid project”, Advanced robotics, Volume 11, Issue.6, pages.605-620.

M. J. Osborne and A. Rubinstein (1990) “Bargaining and Markets”. Academic Press, San Diego, CA, 1990.

M. J. Osborne and A. Rubinstein(1994) “A Course in Game Theory”. MIT Press, Cambridge, MA.

M.Abirami (2013)” Negotiation Mechanisms for Cloud Services”,International Journal of Emerging Trends in Engineering and Development ,Volume 2, Issue 3.

Mai Ben Adar, Nadav Sofy, Avshalom Elimelech (2013) “Gahboninho: Strategy for Balancing Pressure and Compromise in Automated Negotiation”.Complex Automated Negotiations: Theories, Models, and Software Competitions, Studies in Computational Intelligence Volume 435, pages 205-208.

Martin J. Osborne and Ariel Rubinstein(1990) “Bargaining and markets”. Academic Press.

Mastenbroek, W.F.G (1989) “Negotiate”. Basil Blackwell Inc.

Mastenbroek, W.F.G (1999) “Negotiating as Emotion Management. In: Theory, Culture & Society” volume.16, Issue.4, pages.49-73.

Medina, V., & García, J. M. (2014). A survey of migration mechanisms of virtual machines. ACM Computing Surveys (CSUR), Volume 46, Issue 3,Pages 30-33.

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

Nash, Jr. J.F (1950) the Bargaining Problem. Journal of the Econometric Society, Volume.18, Issue.2

Ncho, A., & Aimeur, E. (2004). Building a multi-agent system for automatic negotiation in web service applications. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems.Volume 3 ,pages 1466-1467.

Neumann, D., Stoesser, J., Anandasivam, A., & Borissov, N. (2007). SORMA–building an open grid market for grid resource allocation. In Grid Economics and Business Models pages 194-200.

Osborne M. J & Rubinstein, A (1990) “Bargaining and Markets”. Academic Press

Osborne, M.J & Rubinstein, A (1994) “A Course in Game Theory”.MIT Press, Cambridge, MA

Oshrat, Y, Lin, R & Kraus, S (2009) “Facing the Challenge of Human-agent Negotiations via EffectiveGeneral Opponent Modeling”. In Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, volume 1, pages 377–384.

Parkhill D (1966) “The challenge of the computer utility”. Addison-Wesley, Reading

R. Kowalczyk, M. Ulieru, and R. Unland (2002) “Integrating mobile and intelligent agents in advanced e-commerce: A survey. In Agent Technologies, Infrastructures, Tools, and Applications for E-Services”, pages 295–313.

Ragone, A., Di Noia, T., Di Sciascio, E., & Donini, F. M. (2007). “Alternating-offers protocol for multi-issue bilateral negotiation in semantic-enabled marketplaces” (pp. 395-408).

Raiffa, H (1982) “The Art and Science of Negotiation, How to Resolve Conflicts and Get the Best out of Bargaining”. Cambridge, Mass., Belknap Press of Harvard University Press

Raiffa, H, Richardson, J & Metcalfe, D (2002) Negotiation Analysis: The Science and Art of Collaborative Decision Making, Cambridge, MA: Belknap Press of Harvard UniversityPress

Raiffa, H. (1982). “The art and science of negotiation”. Harvard University Press.

Rasmussen, C.E., Williams, C.K.I (2006) “Gaussian Processes for Machine Learning”. The MIT Press

Rick, L. (2002). “IT Services Management a Description of Service Level Agreements”. RL Consulting.

Rinderle, S., & Benyoucef, M. (2005). “Towards the automation of e-negotiation processes based on web services a modeling approach” Springer Berlin Heidelberg, pages 443-453.

Roger B. Myerson (1991) “Game theory: Analysis of conflict”. Harvard Univ. Press, Cambridge, MA.

Ron, S. & Aliko, P. (2001). Service level agreements. Internet NG. Internet NG project.

Rubin, J. Z., & Brown, B. R. (1975). The social psychology of bargaining and negotiation, Academic press.

Rubinstein, A (1982) “Perfect Equilibrium in a Bargaining Model”. Econometrica: Journal of theEconometric Society, Volume 50, pages 97-109.

Russell S & Norvig P (2010) “Artificial Intelligence: a modern approach”, New Jersey, Pearson

S. Kraus (2001) “Strategic Negotiation in Multiagent Environments”. MIT Press, Cambridge, MA.

S. Kraus, J. Wilkenfeld, and G. Zlotkin. “Multiagent negotiation under time constraints(1995)”. Artificial Intelligence Volume.75, Issue.2, pages 297–345.

Sanchez-Anguix, V., Julian, V., Botti, V., & García-Fornes, A. (2013). Tasks for agent-based negotiation teams: Analysis, review, and challenges. Engineering Applications of Artificial Intelligence, Volume 26, Issue 10, Pages 2480-2494.

Sarit .K (2001). Automated negotiation and decision making in multiagent environments. In Multi-agent systems and applications ,pages 150-172.

Shell, G. R. (2006). Bargaining for advantage: Negotiation strategies for reasonable people. Penguin Group press.

Silaghi, G. C., Şerban, L. D., & Litan, C. M. (2010). A framework for building intelligent SLA negotiation strategies under time constraints. In Economics of Grids, Clouds, Systems, and Services pages. 48-61.

Simon, H. A. (1955). A behavioral model of rational choice. The quarterly journal of economics, pages 99-118.

Son, S., & Jun, S. C. (2013). Negotiation-based Flexible SLA Establishment with SLA-driven Resource Allocation in Cloud Computing. InCluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on pages. 168-171.

Talia, D. (2011). “Cloud Computing and Software Agents: Towards Cloud Intelligent Services” Volume. 11, pages. 2-6.

Talia, D. (2012) "Clouds Meet Agents: Toward Intelligent Cloud Services," Internet Computing, IEEE Volume.16, Issue.2, pages 78–81.

Thijs van Krimpen, Daphne Looije, Siamak Hajizadeh (2013) "HardHeaded", Complex Automated Negotiations: Theories, Models, and Software Competitions Studies in Computational Intelligence Volume 435 , pages 223-227

Thompson, L. L. (2012). The mind and heart of the negotiator, pages. 18-19. Upper Saddle River, NJ: Prentice Hall.

Thompson, L.L (2005) The Heart and Mind of the Negotiator, Upper Saddle River, New Jersey Pearson Prentice Hall

Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). “A break in the clouds: towards a cloud definition”. ACM SIGCOMM Computer Communication Review, volume.39, Issue.1, pages.50-55.

VMware, 2013. VMware. [online] Available at: <http://www.vmware.com> [Accessed 10 October 2014].

Von Neumann and Oskar Morgenstern (1944) “Theory of Games and Economic Behavior” Princeton University Press.

Weiss, G (2000) Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press

Williams, C. R., Robu, V., Gerding, E. H., & Jennings, N. R. (July). Using gaussian processes to optimise concession in complex negotiations against unknown opponents. In IJCAI Proceedings-International Joint Conference on Artificial Intelligence ,Volume 22, Issue 1, Pages 432–435.

Williams, Colin R. (2012) “Practical Strategies for Agent-Based Negotiation in Complex Environments”. University of Southampton, School of Electronics and Computer Science, PhD Thesis.

Wooldridge M & Jennings NR (1995) ”Intelligent Agents: theory and practice”. The Knowledge Engineering Review Volume 10, Issue 2, Pages 115-152.

Wooldridge, M. (2002). An introduction to multiagent systems. John Wiley and Sons.

Wurman, P. R., Wellman, M. P., & Walsh, W. E. (1998). The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In Proceedings of the second international conference on Autonomous agents, Pages 301-308.

Xen, 2013. Xenserver. [online] Available at: <http://www.xenserver.org/> [Accessed 10-10-2014].

Zeng, D., Sycara, K. (1998): “Bayesian learning in negotiation”. International Journal of Human Computer Systems Volume 48, Pages 125-141.

Zhang, Q., Cheng, L., & Boutaba, R. (2010). “Cloud computing: state-of-the-art and research challenges”. Journal of internet services and applications, Volume 1, Issue 1, Pages 7-18.

Appendix

APPENDIX A: Gathering and Filtering stage

This appendix contains the MySQL and XML code from the Gathering and Filtering stage. I implemented a PHP code to send what users enter in HTML form to a MySQL database. Also, by using SimpleXML, PHP code will easily read data from XML file then to save them to MySQL database. Figure 76 shows how to export and Import provider's offers to/ from MySQL database via XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- - phpMyAdmin XML Dump - version 3.3.9 - http://www.phpmyadmin.net - - Host: localhost - Generation Time: Dec 17, 2013 at 04:00 AM - Server version: 5.5.8 - PHP Version:
5.3.5 -->
- <pma_xml_export xmlns:pma="http://www.phpmyadmin.net/some_doc_url/" version="1.0">
      <!-- - Structure schemas -->
  - <pma:structure_schemas>
    - <pma:database charset="latin1" collation="latin1_swedish_ci" name="db">
        <pma:table name="p_table"> CREATE TABLE `p_table` ( `location` set('US-East','US-West','Europe','Asia','Sydney','South-America') NOT NULL,
          `Operating System` set('RedHat Linux','Windows Server','Oracle Linux') NOT NULL, `Term` set('1-6','7-12','More than 12') NOT NULL, `Pr` varchar
          (255) NOT NULL, `review` double NOT NULL, PRIMARY KEY (`Pr`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1; </pma:table>
      </pma:database>
    </pma:structure_schemas>
      <!-- - Database: 'db' -->
  - <database name="db">
      <!-- Table p_table -->
    - <table name="p_table">
        <column name="location">US-East,Europe,Asia</column>
        <column name="Operating_System">Windows Server,Oracle Linux</column>
        <column name="Term">7-12</column>
        <column name="Pr">A</column>
        <column name="review">0.05</column>
      </table>
    - <table name="p_table">
        <column name="location">US-East,US-West,Europe,Asia</column>
        <column name="Operating_System">Windows Server</column>
        <column name="Term">7-12</column>
        <column name="Pr">B</column>
        <column name="review">0.004</column>
      </table>
    - <table name="p_table">
        <column name="location">US-East,US-West,Asia</column>
        <column name="Operating_System">Windows Server,Oracle Linux</column>
        <column name="Term">1-6,7-12</column>
        <column name="Pr">C</column>
        <column name="review">0.02</column>
      </table>
    - <table name="p_table">
        <column name="location">South-America</column>
        <column name="Operating_System">Oracle Linux</column>
        <column name="Term">1-6</column>
```

Figure 76: Exporting and Import provider's offers to/ from MySQL database

Using XML file to update customers and provider's offers and requests will make taking human out of the loop for *Gathering* stage possible. User's agent will be also able to update offers and requests via a XML file. Once the XML file uploaded to the cloud by the users' agent, the PHP code will be able to access it and update the framework with customers and provider's offers and requests. After that, the offer is saved into the database as illustrated in figure 77.

```
12
13  // Create connection
14  $con=mysqli_connect("localhost","root","","db");
15  // Check connection
16  if (mysqli_connect_errno($con))
17    {
18    echo "Failed to connect to MySQL: " . mysqli_connect_error();
19    }
20
21    $result2 = mysqli_query($con,"SELECT * FROM `p_table` LIMIT 0, 30 ");
22
23    $result = mysqli_query($con,"
24
25  SELECT *
26  FROM  `p_table`
27  WHERE  `location` LIKE  '%$location1%'
28  AND  `Operating System` LIKE  '%$OperatingSystem1%'
29  AND  `Term` LIKE  '%$Term1%'
30  AND  `review` < $review1
31  LIMIT 0 , 30
32
```

Figure 77: Connecting to the MySQL server

After that, the offer is saved into the database as illustrated in figure 78.

```
12
13  // Create connection
14  $con=mysqli_connect("localhost","root","","db");
15  // Check connection
16  if (mysqli_connect_errno($con))
17    {
18    echo "Failed to connect to MySQL: " . mysqli_connect_error();
19    }
20
21    $result2 = mysqli_query($con,"SELECT * FROM `p_table` LIMIT 0, 30 ");
22
23    $result = mysqli_query($con,"
24
25  SELECT *
26  FROM  `p_table`
27  WHERE  `location` LIKE  '%$location1%'
28  AND  `Operating System` LIKE  '%$OperatingSystem1%'
29  AND  `Term` LIKE  '%$Term1%'
30  AND  `review` < $review1
31  LIMIT 0 , 30
32
```

Figure 78: Connecting to the MySQL server

APPENDIX B: Negotiation stage.

This appendix contains the MySQL and XML code from the Negotiation stage. Negotiation domain is an XML file made of all the possible bids (list of *Issues and Values* that inside each Issue). See figure 80 and figure 80 Negotiation preferences are extended files of Negotiation domain file. Negotiation preference is an XML file as well. Negotiation preferences hold private information about the agent preferences. So it includes the *Evaluation values* for each *value* inside each *Issue*. It also includes the *weight* of each *Issue*, showing the important of each *Issue*.

```
<?xml version="1.0"?>
- <negotiation_template>
  - <utility_space number_of_issues="8">
    - <objective etype="objective" type="objective" name="VM" description="" index="0">
      - <issue etype="discrete" type="discrete" name="Availability Zone" index="1" vtype="discrete">
          <item index="1" value="US-East"> </item>
          <item index="2" value="US-West"> </item>
          <item index="3" value="Europe"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Operating System" index="2" vtype="discrete">
          <item index="1" value="RedHat Linux"> </item>
          <item index="2" value="Ubuntu"> </item>
          <item index="3" value="Oracle Linux"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Term" index="3" vtype="discrete">
          <item index="1" value="[1-6]"> </item>
          <item index="2" value="[7-12]"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Memory" index="4" vtype="discrete">
          <item index="1" value="7"> </item>
          <item index="2" value="8"> </item>
          <item index="3" value="9"> </item>
          <item index="4" value="10"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Virtual CPU" index="5" vtype="discrete">
          <item index="1" value="4"> </item>
          <item index="2" value="5"> </item>
          <item index="3" value="6"> </item>
          <item index="4" value="7"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Storage" index="6" vtype="discrete">
          <item index="1" value="[251- 500]"> </item>
          <item index="2" value="[501- 725]"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Platform" index="7" vtype="discrete">
          <item index="1" value="32-bit"> </item>
          <item index="2" value="64-bit"> </item>
        </issue>
      - <issue etype="discrete" type="discrete" name="Utilization" index="8" vtype="discrete">
          <item index="1" value="Light"> </item>
          <item index="2" value="Medium"> </item>
        </issue>
      </objective>
    </utility_space>
</negotiation_template>
```

Figure 79: Negotiation domain as XML file

```
<?xml version="1.0"?>
- <utility_space>
  - <objective etype="objective" type="objective" name="VM" description="" index="0">
    - <issue etype="discrete" type="discrete" name="Availability Zone" index="1" vtype="discrete">
        <item index="1" evaluation="10" value="US-East"> </item>
        <item index="2" evaluation="20" value="US-West"> </item>
        <item index="3" evaluation="30" value="Europe"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Operating System" index="2" vtype="discrete">
        <item index="1" evaluation="10" value="RedHat Linux"> </item>
        <item index="2" evaluation="20" value="Ubuntu"> </item>
        <item index="3" evaluation="30" value="Oracle Linux"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Term" index="3" vtype="discrete">
        <item index="1" evaluation="10" value="[1-6]"> </item>
        <item index="2" evaluation="20" value="[7-12]"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Memory" index="4" vtype="discrete">
        <item index="1" evaluation="10" value="7"> </item>
        <item index="2" evaluation="20" value="8"> </item>
        <item index="3" evaluation="30" value="9"> </item>
        <item index="4" evaluation="40" value="10"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Virtual CPU" index="5" vtype="discrete">
        <item index="1" evaluation="10" value="4"> </item>
        <item index="2" evaluation="40" value="5"> </item>
        <item index="3" evaluation="30" value="6"> </item>
        <item index="4" evaluation="20" value="7"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Storage" index="6" vtype="discrete">
        <item index="1" evaluation="10" value="[251- 500]"> </item>
        <item index="2" evaluation="20" value="[501- 725]"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Platform" index="7" vtype="discrete">
        <item index="1" evaluation="10" value="32-bit"> </item>
        <item index="2" evaluation="20" value="64-bit"> </item>
      </issue>
    - <issue etype="discrete" type="discrete" name="Utilization" index="8" vtype="discrete">
        <item index="1" evaluation="10" value="Light"> </item>
        <item index="2" evaluation="20" value="Medium"> </item>
      </issue>
      <weight index="1" value="0.187592252955141"> </weight>
      <weight index="2" value="0.1494128546436554"> </weight>
      <weight index="3" value="0.014110001933379959"> </weight>
      <weight index="4" value="0.18036901153144957"> </weight>
      <weight index="5" value="0.06814081644643968"> </weight>
      <weight index="6" value="0.049097268672403514"> </weight>
      <weight index="7" value="0.23022557432735322"> </weight>
      <weight index="8" value="0.12105221949017769"> </weight>
    </objective>
    <discount_factor value="0.5"> </discount_factor>
    <reservation value="0.1"> </reservation>
</utility_space>
```

Figure: 80 The preference profiles xml file