Rubinstein’s Alternating Offers Protocol for Automated Cloud Computing Negotiation

Faisal Alsheed, Abdennour El Rhalibi, Martin Randles, Madjid Merabti

School of Computing and Mathematical Sciences
Liverpool John Moores University
Byrom Street, L3 3AF, Liverpool, UK

Abstract—Cloud Computing environments are dynamic and open, 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. Moreover, traditional negotiations between humans are often fraught with difficulty. There are many automated negotiation algorithms and protocols, which have been developed over the years; 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, offers a satisfactory technical solution for this challenging problem. The purpose of this paper is to apply the state of the art in negotiation automated algorithms/agents within a described Cloud Computing SLA framework, and to evaluate the most appropriate negotiation approach based on many criteria. Also, the functional integration requirements are discussed to make the framework flexible and open to new algorithms/agents that will be developed in the future for a better negotiation outcome. In brief, this research output focuses on how Rubinstein’s Alternating Offers Protocol can be used for automated SLA negotiation in Cloud Computing. The agents/ algorithms performance, using different approaches, and the result are discussed.

Keywords - Negotiation, Automated Negotiation, Cloud computing, Service Level Agreements (SLA) management.

I. INTRODUCTION

Today, Cloud computing promises a new model of delivering computing with a lot of flexibility. The computing can be delivered as Software as a service (SaaS), Platform as a service (PaaS) or Infrastructure as a service (IaaS). Cloud computing is defined 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 for an optimum resource utilization. This pool of resources is typically exploited within a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs “[1]. These customized SLAs can only be established by negotiation. The negotiation needs to be automated to handle the dynamic and complex environment cloud computing. In this work, we propose a framework for achieving automated SLA negotiation between providers and consumers for cloud resources.

Thompson defined negotiation “as a decision process in which two or more parties make individual decisions and interact with each other for mutual gain” [2]. Furthermore, automated negotiation can be defined as “a kind of system, that applying information technology, communication technology and artificial intelligent into negotiation area, composed with game theory, operations research and decision theory” [3]. We will consider these two definitions in the development of our SLA negotiation framework. The bargaining development can be executed automated among intelligent agents instead of human, from the beginning to the end [3]. There are many areas studied in the topic of negotiation, including economics, e-commerce, artificial intelligence, game theory.

Traditionally, negotiation is necessary in everyday life, but it is a time consuming activity and might lead to unfair outcomes especially when the range of possible outcomes is large. This has led to a focus on automated negotiation, for example in the setting of e-commerce. This interest has increased by the promise of intelligent agents being able to negotiate on behalf of human negotiators, or even to outperform them. As Thompson pointed out, there are many problems with negotiation between humans. Firstly negotiation between humans is quite slow, and this is further complicated by issues of culture, ego and discrimination [2]. In addition, Cloud computing providers and consumers join and leave the cloud marketplaces frequently. In such an environment, which is dynamic and open, automated negotiation is needed to establish service level agreements between service providers and consumers with no previous knowledge of each other.

The motivation and challenges for this work are that a successful negotiation is needed to create a SLA, which will establish a strong working relationship between cloud consumers and the providers. However, due to the increased number of cloud consumers and providers, it becomes almost impossible to meet face to face to negotiate and create an SLA; thus automated negotiation is needed. The area of intelligent agent based automated negotiation is now being widely studied [4]. Beam [5] argued that it is impossible to comprehend automated negotiation completely by current technologies and theories of human behavior. However, many researchers are working hard on negotiation protocols and strategies from different points of view [4]. So, the purpose of the work is to apply the state of the art in negotiation automated
algorithms/agents to a Cloud computing SLA framework we define in this paper. This framework is flexible and open to new algorithms/agents that might be developed in future for a better negotiation outcome.

In this paper, a framework is proposed for achieving automated negotiation between providers and consumers in cloud environments. The rest of this paper is organized as follows: Section 2 introduces the related work; in Section 3 we discuss our Cloud computing SLA negotiation Framework. Section 4 discusses the Negotiation algorithms. Section 5 discusses negotiation scenario. Section 6 introduces our negotiation experiments. Section 7 introduces a discussion about the experiments and the agents' evaluation, and in Section 8 we summarize the main contributions and possible future development of this work as a conclusion.

II. RELATED WORK

There are some existing negotiation frameworks and negotiation support systems that have already been developed: OPELIX [6] is a European project that lets 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. Inspire [7] and Aspire [8] are associated projects. Inspire [7] helps human operators in managing bilateral negotiations by organizing offers and counter-offers. Aspire [8] improves upon Inspire by giving negotiation support using intelligent agents to make suggestions to 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. Kasbah [9] allows a customer and a provider to generate their own agents, assign them some strategic directions, and launch them at centralized marketplace for negotiations. CAAT [10] 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. Work in [17] 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 in the same time a customer is looking for a better deal by negotiating with many providers. Also, the offer and demand in the cloud market play a big role in choosing negotiations strategies.

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 ignored 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 strategy.
- Supporting multi-issue negotiation with a large domain (hundreds of thousands possible outcomes).
- Negotiating with multi-providers.
- Open for new agents.
- Possibility of re-negotiating.
- Monitoring the SLA after the negotiating.

This approach proposes a more comprehensive novel framework for automated cloud computing negotiation including all the stages involved in Cloud computing negotiations in addition to an autonomic monitoring stage. The framework is introduced in the following section.

III. FRAMEWORK

![Framework Workflow](image)

The framework is made up of 5 stages; the output of each stage is an essential input for the next stage:
Stage 1: Gathering
This stage is one of the most important stages, because all the input for the framework will be gathered together only at this point. The inputs will be customer’s request and provider’s offer, the policy of negotiation’s strategy, the negotiation’s preferences, the price policy, the monitoring rules and policies, the real-time monitoring results and the monitoring alerts. All the inputs will be saved in the accessible database. The users (Customers and Providers) will 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. 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 personal profile. However, the users will not have permission to edit some of that profile’s information, for example: the review/reputation data.

Stage 2: Filtering
In this stage, the customer’s request that has been sent in the gathering stage will be used to filter all the providers in order to recommend the best matched candidates. The customer’s 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 previous experiences with the provider. The output of this stage will be the candidate providers, with whom the customer will be negotiating separately.

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. 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.

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 that can be monitored in the following stage.

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.
In this paper it will be demonstrated how stage 3 works; the next section will introduce the negotiation protocol and the algorithms.

IV. NEGOTIATION PROTOCOL AND ALGORITHMS
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 [11]. 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 [3]. 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.

Now, after determining the protocol of negotiation, it is necessary to discuss the negotiators. 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 \( \omega \) is the outcome and \( \Omega \) is the domain. The overall utility consists of a weighted sum of the utility for each individual issue.

\[
U(\nu_1,\ldots,\nu_n) = \sum_{i=1}^{N} \frac{\text{eval}(\nu_i)}{\text{max}([\text{eval}(\nu_1),\ldots,\text{eval}(\nu_n)])}
\]

A bid is a set of chosen values \( \nu_1,\ldots,\nu_n \) for each of the \( n \) issues. Each of these values has been assigned an evaluation value \( \text{eval}(\nu_i) \) 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 artificially intelligent to be able to;

- Learn about the opponent behaviour from its moves.
- 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 [12].

Each agent follows a completely different approach to perform each of these attributes. In this work, we will demonstrate how two agents will follow different kinds of strategies to perform each of above attribute. The agents are HardHeaded [13] and Tit-for-Tat [13].
A: HardHeaded

This agent [13] 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. The target of the learning module is to learn the utility value and the weights for the opposing agent [13]. To study the opposing agent, this agent makes two assumptions about the opponent; it first assumes that the opponent restricts the bids with 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”. This learning function keeps updating the issue weights and value utilities of the preference profile immediately after each bid. 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 [13]. The fact that this agent uses a simple learning module and also an optimal concession function with low computational complexity; this 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 bids space [13].

B: Tit-for-Tat

This agent’s strategy is based on the principle of Tit for Tat (tft) [14]. 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 [14]. 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 [15]. 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 [15]. 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 [15]. 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 [15].

C: Hardliner agent

Hardliner is a very selfish and stubborn agent that keeps repeating the same offer which is only good for itself, expecting the opponent will give up and accept the offer at the end. 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 conceeds. This is the most competitive strategy that can be used. This agent is deterministic. It will give the opponent the full negotiation time to make concessions and accept its offer. This agent is used, as it represents the 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.

IV. Scenario Representation

After discussing the theory behind each agent and the agent’s negotiation strategies, now the agents Hardliner and tft will be analyzed in action using [16]. Two methods will be used to investigate the agents’ performance:

1- By negotiating with a Hardliner agent.
2- By negotiating with itself.

The scenario assumption 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 with the criteria as shown in table 1:

<table>
<thead>
<tr>
<th>Issue</th>
<th>Value</th>
<th>Customer Evaluation</th>
<th>Weight</th>
<th>Provider Evaluation</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability Zone</td>
<td>US-East</td>
<td>0.33</td>
<td>0.19</td>
<td>0.66</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>US-West</td>
<td>0.66</td>
<td>0.33</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Operating System</td>
<td>RedHat Linux</td>
<td>0.33</td>
<td>0.15</td>
<td>1.00</td>
<td>0.29</td>
</tr>
<tr>
<td></td>
<td>Ubuntu Linux</td>
<td>0.66</td>
<td>0.66</td>
<td>1.00</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td>Oracle Linux</td>
<td>1.00</td>
<td>0.01</td>
<td>0.50</td>
<td>0.03</td>
</tr>
<tr>
<td>Term (months)</td>
<td>[1-6]</td>
<td>0.50</td>
<td>0.50</td>
<td>0.75</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>[7-12]</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Memory GB</td>
<td>7</td>
<td>0.25</td>
<td>0.18</td>
<td>0.75</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.50</td>
<td>0.01</td>
<td>0.50</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.75</td>
<td>0.11</td>
<td>0.25</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.11</td>
</tr>
<tr>
<td>Compute Units</td>
<td>4</td>
<td>0.25</td>
<td>0.07</td>
<td>1.00</td>
<td>0.11</td>
</tr>
<tr>
<td>/virtual core (CPU)</td>
<td>5</td>
<td>1.00</td>
<td>0.50</td>
<td>0.75</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.75</td>
<td>0.50</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.50</td>
<td>0.25</td>
<td>1.00</td>
<td>0.25</td>
</tr>
<tr>
<td>Storage GB</td>
<td>[256 - 500]</td>
<td>0.50</td>
<td>0.05</td>
<td>1.00</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>[501 - 725]</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.23</td>
</tr>
<tr>
<td>Platform</td>
<td>32-bit</td>
<td>0.50</td>
<td>0.23</td>
<td>1.00</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>64-bit</td>
<td>1.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.12</td>
</tr>
<tr>
<td>Utilization</td>
<td>Light &lt; 39%</td>
<td>0.50</td>
<td>0.12</td>
<td>1.00</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>Medium &lt; 75%</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

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.

In this scenario, a customer and a provider, negotiate over the specifications of a virtual machine. 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. After agreement about the outcome, both sides will negotiate over the price. The deadline of the negotiation is fixed and is set to 3 minutes in total for these experiments. Without a deadline, the negotiation might go on forever. However in future work, it is planned to investigate the effects of the negotiation deadline on the outcome of negotiation.

V. NEGOTIATION EXPERIMENTS OUTCOME:
Next the agent’s performance will be investigated by negotiating with a Hardliner agent.

A: 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 the next figure and table show, agent HardHeaded was trying more than Hardliner.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Utility of each Agent’s first offer</th>
<th>Utility of Last offer for each agent.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardliner</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>HardHeaded</td>
<td>0.99</td>
<td>0.85</td>
</tr>
</tbody>
</table>

B: 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 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, ])

C: By negotiating with itself.
When Hardheaded negotiated with itself 90% of 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. When Tit for Tat Agent negotiated with itself all the negotiation sessions ended with agreements. However, 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.

VI. DISCUSSION

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

A: Using take-it-or-leave-it strategy e.g. Hardliner, most of the times lead to no agreement unless facing a desperate customer. Since the output of stage 2(Filtering) of our framework is the candidate providers who the customer will separately negotiate with, 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 tend 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 they will be an increased number of cloud providers with more competition, we recommend the cloud providers to offer customized and negotiated SLA by using a cooperative agent like Tit for Tat.

B: Using a selfish agent e.g. HardHeaded is risky. We recommend providers to use Hardheaded when demand is higher than supply in the market. There is risk of ending some of the negotiations without agreement. However when the negotiating ends with an agreement, this agent will get the higher utility.

C: 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 agreement but with low utility when it faces selfish agent.

VII. CONCLUSION

This work focuses on how Rubinstein’s Alternating Offers Protocol can be used for automated negotiation for Cloud Computing. A framework is proposed for achieving automated negotiation between providers and consumers in cloud environments. The main novelty of the work is that the framework is made especially for cloud computing by using state-of-the-art automated negotiation algorithms/agents. Also, at the same time it is flexible and open to new algorithms/agents that might be developed in the future. The related work has been classified into the following categories; Negotiation Evolution, Negotiation frameworks and Negotiation Support Systems (NSS), Negotiation algorithms are discussed and, the Negotiation experiments outcome is analyzed. In future work, it is planned to investigate the effects of deadlines on the outcome of the negotiations; and also to investigate more negotiation agents/ algorithms.

ACKNOWLEDGMENT

The authors are grateful to Interactive Intelligence Group (Department of Intelligent Systems) at Delft University of Technology for making Genius [16] publicly available.

REFERENCES