Automated SLA Negotiation Framework for Cloud Computing

Linlin Wu, Saurabh Kumar Garg and Rajkumar Buyya
Cloud Computing and Distributed Systems Laboratory
Department of Computing and Information Systems
The University of Melbourne, Australia
{linwu, saurabh, raj} @csse.unimelb.edu.au

Chao Chen and Steve Versteeg
CA Technologies
Melbourne, Australia
{chao.chen28, steve.versteeg}@ca.com

***Abstract*—A Service Level Agreement (SLA) is a legal contract between parties to ensure the Quality of Service (QoS) are provided to the customers. A SLA negotiation between participants assists in defining the QoS requirements of critical service-based processes. However, the negotiation process for customers is a significant task particularly when there are multiple SaaS providers in the Cloud market, as service cost and quality are constantly changing and consumers have varying needs. Therefore, we propose 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. The automated negotiation framework facilitates intelligent bilateral bargaining of SLAs between a SaaS broker and multiple providers to achieve different objectives for different participants. To maximize profit and improve customer satisfaction levels for the broker, we propose the design of counter offer generation strategies and decision making heuristics that take into account time, market constraints and trade-off between QoS parameters. Our negotiation heuristics are evaluated by extensive experimental studies of our framework using data from a real Cloud provider.**

Keywords- SLA negotiation; Software-as-a-Service; market-oriented; Cloud computing; resource allocation;

I. INTRODUCTION

A service level agreement (SLA) is a legal contract between providers and consumers that defines the Quality of Service (QoS), which is achieved through a negotiation process [1]. Negotiation processes in Cloud are essential because participating parties are independent entities with different objectives and QoS requirements. Through negotiation, players in the Cloud marketplace [16] are given the opportunity to maximize their return-on-investment.

Currently, SLAs are defined by service providers without providing customers with sufficient negotiation opportunity. Moreover, current preliminary research work [15] on automated SLA negotiation frameworks in Cloud is minimal and generally does not consider, in combination, the following two factors: 1) the dynamic nature of the Cloud, as service cost and quality are constantly changing and consumers have varying needs, and 2) time and market oriented resource allocation, as any delay incurred in waiting for a resource assignment is perceived as an overhead [2]. These two factors make answering the following questions in design of a negotiation framework for Cloud a challenging task: 1) how to balance trade-off between multiple QoS parameters; 2) which provider offer to accept; and 3) how to make a decision to accept or how to generate a counter offer?

To address these questions, our proposed negotiation framework integrates the following: 1) multiple QoS parameters are balanced through prioritization, which is based on customer preferences, and 2) to choose the best provider, a SaaS broker is introduced on behalf of customers to negotiate with multiple providers simultaneously in order to select the best offer. The best offer is selected based on different objectives of the parties involved in the negotiation. Moreover, our decision making system considers the current Cloud market situation, time constraints and multiple QoS parameters. In the dynamic Cloud market, opportunities and competitions between providers can have a considerable impact on strategies and the decision making processes. For example, when the competition increases or the opportunity decreases, the counter offer generation strategy is to concede faster.

A. Motivation

Our work is motivated by: 1) the emergence of the SaaS broker model [12], and 2) the lack of automated negotiation frameworks along with decision making systems and strategies to maximize profit and improve CSL in Cloud.

The broker model has been used mainly in utility markets. Due to lack of detailed information about different providers and current market, customers prefer using brokers, which provide fast, economical solutions. Similarly, in Cloud, customers face the problem of identifying the best provider, when the number of providers is dramatically increasing. Therefore, the SaaS broker model in Cloud provides a one-stop-shop for guaranteed customer service.

Currently, in the Cloud market, brokers such as ViTLive [12] only provide a portal listing of different providers. However, they do not select or negotiate with providers to maximize profit and improve customer satisfaction. If negotiation is required, specialist knowledge is sourced to manage the process which incurs additional direct costs. In addition, the existing negotiation framework may not be automated [6], or suitable for Cloud specific negotiations [11].

We propose an automated Cloud negotiation framework, counter offer generation strategies, and decision making heuristics considering time and market factors to achieve various objectives for different parties. In this way, the broker can set up the parallel negotiation process to

maximize profit or the CSL. Our proposed negotiation framework can be extended for any layer (e.g. Platform-as-a-Service, and Infrastructure-as-a-Service) in both private and public Cloud. For public Cloud, SaaS providers can use brokers' strategies and Infrastructure providers can use providers' strategies. For private Cloud, the resource user could use broker's strategies and resource venders could use provider's strategies.

B. *Contributions*

The key contributions of this paper are: 1) a novel negotiation framework for Cloud along with decision making heuristics to achieve different objectives and strategies considering both time and market factors for counter offer generation, and 2) a prototype of our framework which is implemented proposed decision making heuristics and strategies, and compared with the latest best approach proposed by Zulkernine and Martin [9]. The experimental results demonstrate that our approach generates up to 50% increased profit and about a 60% customer satisfaction level (CSL) improvement for brokers over the base heuristic.

II. AUTOMATED NEGOTIATION FRAMEWORK

In order to design an automated negotiation framework in Cloud, it is important to define negotiation objectives, processes, and strategies.

A. *Framework Components*

The main components in our negotiation framework are: Customer Agent (CA), Broker Coordinator Agent (BCA), Provider Agent (PA), IaaS Provider, SLA Generator, Directory, Policy Database (PD), and Knowledge Base (KB).
Customer Agent: Represents a customer that submits requests for software services and registers their QoS requirements into PD.

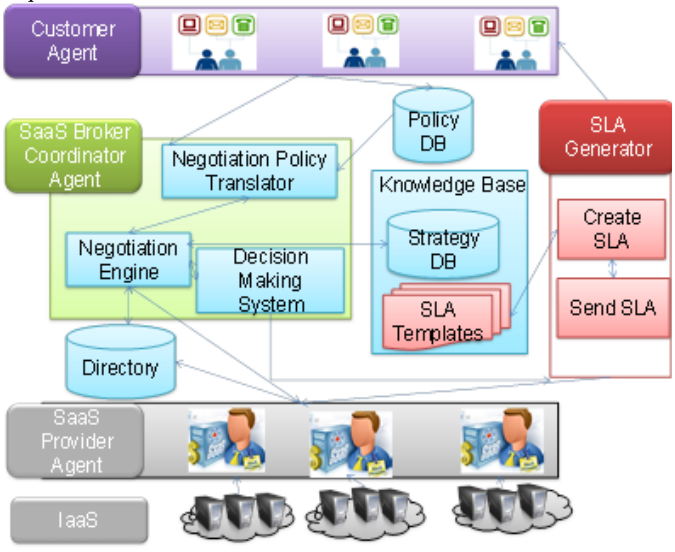


Figure 1: Negotiation Framework High Level Architecture

Broker Coordinator Agent: Represents the broker by receiving customer requests and negotiates with providers to achieve business objectives. It includes **Negotiation Policy Translator (NPT), Negotiation Engine (NE),** and **Decision Making System (DMS)**.
Negotiation Policy Translator: Maps customer's QoS parameters to provider level parameters.
Negotiation Engine: Includes workflows which use negotiation strategies during the negotiation process.
Decision Making System: Uses decision making heuristics to update the negotiation status.
Provider Agent: Represents the provider. PA could include the third party monitoring system to update the provider's dynamic information. Although out of the scope of this paper, systems and processes can be implemented to monitor and measure provider capabilities.
The SLA Generator: When the negotiation has been successfully completed, the SLA Generator creates an SLA between the customer and the provider using templates retrieved from the KB. The template includes specified Service Level Objectives (SLOs) according to the QoS (SLA excludes any general legal terms and conditions).
The Directory: The repository stores the providers' registered service information.
The Policy DB: The repository stores QoS terms that both providers and customers understand.
The Knowledge Base: The repository stores negotiation strategies and SLA templates.
This paper focus on two main components: the NE, by proposing strategies considering both time and market, and the DMS, by proposing heuristics for different objectives.

B. *System Scenario*

We consider three entities: consumers, SaaS brokers and SaaS providers. Each consumer c submits a service request to the SaaS broker, who leases software services from SaaS providers. The **customer** c requests services with the following attributes:

- Budget B_c: the maximum price a customer can afford.
- Software service set $SR_{b:}$ the service editions.
- The service start time t_{ss}: the latest service available time for a customer c.
- The contract length indicates the period of service usage *conLength*, so that customer c must be able to use software service within the contract term.
- The service refresh time t_r: time it takes a query operation to be executed in a software service.
- The service process time $t_{p:}$ the maximum time for a consumer c to wait for completing a transaction.
- The service availability *avai:* the minimum availability that the customer requires.
- The expected discount percentage for budget σ: the percentage a customer can save from their actual budget.
- The preference level of each QoS parameter γ: the absolute importance level which varies (0, 1].

The broker receives the customer request and calculates the expected budget, expected refresh time, process time, and availability. These expected values are the **best** values that the broker expects to provide to the customer and they will be proposed to providers in the quote request process. If providers cannot fulfil these expected values, the broker will adjust the expected value up to the customer requested value during the negotiation process. The broker always seeks to secure the expected value from provider.

Each **provider** offers the same or different types of services. The provider can host or lease infrastructure services from 3rd party IaaS providers.

C. Negotiation Objectives

In sophisticated markets, the negotiation objective is not only price but also other elements such as quality, reliability of supply, or the creation of long-term relationships. We consider multiple objectives including cost, refresh time, process time and availability. The main objectives for a customer, a SaaS broker and a provider are:

- **Customer**: minimize price and guaranteed QoS within expected timeline.
- **SaaS Broker**: maximize profit from the margin between the customer's budget and the providers' negotiated price.
- **SaaS Provider**: maximize profit by accepting as many requests as possible to enlarge market share.

1) Mathematical Models

a) SaaS Broker

The broker's actual budget $maxB_c$ for serving a customer c depends on the customer's budget B_c and the customer expected discount percentage σ for budget.

$$maxB_c = B_c \times (1-\sigma) \tag{1}$$

The initial budget proposed to all providers is the expected budget $expB_c$, which is based on the $maxB_c$ and the broker's expected margin $margin_c$:

$$expB_c = \max B_c \times (1-magin_c) \tag{2}$$

The profit of broker b gained from serving customer c depends on the B_c and the best provider's price $price_p$.

$$Prof_b = maxB_c - price_p \tag{3}$$

In the following sections, a QoS parameter shall also be referred to as an "**Issue**". The δ_i represents the expected improvement percentage for an issue. Therefore, the *CSL* is reflected by these Issues, which are service refresh time, process time and availability.

The expected refresh time $expT_r$ depends on the customer requested refresh time t_r and the improvement percentage for refresh time δ_r. The $expT_r$ changes during the negotiation process up to t_r.

$$expT_r = t_r \times (1-\delta_r) \tag{4}$$

The customer requested service process time t_p and the improvement percentage for process time δ_p impact the expected process time $expT_p$ and varies during the negotiation process up to the t_p.

$$expT_p = t_p \times (1-\delta_p) \tag{5}$$

The expected availability *expAvai* depends on the customer requested service availability *avai* and the improvement percentage of availability δ_a .

$$expAvai = avai \times (1+\delta_a) \tag{6}$$

The CSL of an individual Issue csl_i depends on the variation between the current proposed value from provider $currentV_i$ and the broker expected value $\exp V_i$. The parameter υ is a value to guarantee that csl_i lies in the interval [0, 1].

$$csl_i = \frac{currentV_i - \exp V_i}{\exp V_i} \times \upsilon \tag{7}$$

The total customer satisfaction level CSL_c, where i represents the individual issue, I indicates all Issues, γ_i indicates the importance level of the Issue i, and the csl_i.

$$CSL_c = \sum_{i=0}^{I} \gamma_i \times csl_i \tag{8}$$

b) SaaS Provider

The provider's service price is based on the provider's cost $cost_p$ and expected margin $expMagin_p$. Different providers calculate price differently. The general equation for a provider to calculate price is proposed below.

$$price_p = \cos t_p + \exp Magin_p \tag{9}$$

The $cost_p$ depends on the base cost $baseCost_p$ (such as infrastructure cost, admin cost, software cost) and the relevant cost of satisfying each Issue i, *where* $i \in I$. Take availability as an example. To provide a higher availability than what currently exists, it may cost extra for the provider to buy another server as a mirror server. This extra cost is the relevant cost for satisfying availability.

$$cost_p = baseCost_p + \sum_{i=0}^{I} \cos t(i) \tag{10}$$

D. Negotiation Process

The negotiation process includes the *Negotiation Policy Specification* and the *Negotiation Protocols* used in our framework which are detailed in **Appendix A**. The negotiation policy specifications used to specify QoS parameters are briefly discussed below.

a) QoS Model: Various participants using different terms is one of the critical challenges in SLA negotiation [14]. In our framework, a QoS model defines a set of QoS dimensions, which represent specific quality aspects of a

service (e.g. availability is a QoS dimension). The QoS model is shared among service consumers and providers. Thus, they have a common understanding of the QoS attributes in relation to how these attributes are defined and measured. For existing service providers and consumers using different terms, transformations are necessary and can be challenging in practice due to overlapping semantics, which is out of the scope of this paper. In this paper, we consider the following QoS dimensions – price, refresh time, process time and availability. These dimensions are widely used and domain-independent. Our QoS model can be easily extended to include other QoS dimensions.

b) Negotiation Protocol: The negotiation protocol refers to a set of rules, steps or sequences during the negotiation process, aiming at SLA establishment. It covers the negotiation status (propose offer, accept/reject offer, and terminate negotiation), which can be updated during the negotiation process. It is common to characterize negotiations by their settings: bilateral, one-to-many, or many-to-many. In this paper we focus on the one-to-many bargaining scenario, where we consider three types of agents (customer agent, broker coordinator agent and provider agent). A broker agent negotiates with many provider agents in a bilateral fashion.

E. Decision Making System

In the negotiation process, the action that a participant performs is determined by a decision making system. In the decision making system, three main questions need to be answered: 1) how to evaluate the offer; 2) what actions to take: accept, reject or generate counter offer; and 3) how to generate counter offer? We design negotiation heuristics to answer them from the broker and provider's perspectives.

a) Broker

After **BCA** requests quotes from all PAs, each PA proposes an initial offer to the BCA, which selects the best offer and makes a decision. If the decision is to propose a counter offer, then the new counter offer will be proposed to all PAs. The best offer is selected based on different objectives. We consider cost-benefit objectives as follows:

- **Minimum cost**: selects the offer with the lowest price first and then the highest cumulative CSL for all QoS.
- **Maximize CSL**: selects the offer with the highest cumulative CSL for all QoS first and then the lowest price.

Table 2. The *Mincost* Heuristic

Conditions	Within BCA's expB	Exceed BCA's expB
All QoS parameters are satisfied	If deadline condition is urgent, agree. Otherwise decrease expB.	If expB is less than actual budget, then increase expB. Otherwise reject.
Not all QoS are satisfied	Satisfy all parameters and reduce expB.	Satisfy all parameters by negotiating on minimal (not desired) values.

Table 3. The *Maxcsl* Heuristic

Conditions	Within BCA's expB	Exceed BCA's expB
all QoS parameters are satisfied	If deadline condition is urgent, agree. Otherwise decreases the least preference parameter to decrease expB.	Decreases the value of parameters, which are better than expected to decrease price.
Not all QoS are satisfied	Satisfy all parameters and increases expB.	Increases expB.

After selecting the best offer, the broker needs to decide how to deal with the selected best offer. One of three actions can be adopted: accept, reject or generate counter offer according to negotiation heuristics. We design two broker negotiation heuristics (*mincost* heuristic and *maxcsl* heuristic) to decide which action to take according to different objectives.

In these two heuristics (Table 2, 3), cost and other Issue values are calculated using negotiation strategy functions, where the most desired and the minimal acceptable values for each Issue are considered for the broker.

In both decision making heuristics, two criteria is used to evaluate the offer: 1) weather offer is within BCA's expected budget: whether the service price offered by provider $price_p$ is less than the broker's expected budget *expB*, and 2) whether all QoS parameters are satisfied.

The above two criteria generate four combined conditions. For each condition, the decision making heuristics guide the broker to make different decisions on which Issue requires adjustment. There are two factors that require consideration when making adjustments. Firstly, trade-off between cost and QoS parameters depends on the objective. Secondly, when the broker must concede on QoS parameters, it always adjusts the least preferred parameter. After the broker decides which Issue to adjust, the new value of the Issue is calculated. The time complexity of these heuristics is *O(CPI)* depending on the number of customers (*C*), the number of providers (*P*) and the number of Issues (*I*).

b) Provider

The provider's objective is to maximize profit by accepting as many requests as possible. Therefore, the provider does not reject requests but continues to negotiate with each broker until negotiations have ended. Table 4 shows the provider's decision making heuristic.

Table 4. Provider's Decision Making Heuristic

Conditions	Within BCA's expB	Exceed BCA's expB
All QoS parameters are satisfied	If deadline condition is urgent, agree. Otherwise decrease the least preference parameter to decrease expB.	If expB is less than actual budget, increase expB. Otherwise decrease the QoS value.
Not all QoS are satisfied	Satisfy all parameters and increase price.	Increase price.

F. Negotiation Strategy

The negotiation strategy underpins the counter offer generation process using various strategy functions which guide to what degree the agent concedes or bargains considering time and market factors.

The strategy functions control whether an agent concedes on certain Issues, or in the alternative, negotiates very hard in each negotiation until the deadline is reached.

The new value $newv^i_{a\to\wedge a}$ proposed by agent *a* (e.g. broker) to opponent ^a (e.g. provider) for Issue *i* depends on the current value of Issue *i* proposed by the opponent agent $cv^i_{\wedge a}$, the best expected value $bestv^i_a$ and a strategy function.

$$newv^i_{a\to\wedge a} = cv^i_{\wedge a} + \alpha^i_a(\chi_1, \chi_2 \ldots \chi_n)(bestv^i_a - cv^i_{\wedge a}) \quad (11)$$

The strategy function $\alpha^i_a(\chi_1, \chi_2 \ldots \chi_n)$ guides the speed of adjustment, where χ_n indicates different factors (such as time, market related factors), which will be explained below.

Opportunity: At time *t*, the probability that an agent is ranked as the most preferred candidate is defined using the condition of opportunity $C_o(c_t, p_t)$. At time *t*, c_t indicates the number of competitors, and p_t indicates the number of partners[14], e.g. for a broker, competitors are other brokers and partners are providers.

$$C_o(c_t, p_t) = 1 - \left(\frac{c_t - 1}{c_t}\right)^{p_t} \quad (12)$$

Competition: At time *t*, the competition $C_c(c_t, p_t)$ in the market depends on the demand and supply ratio (equation 13). At time *t*, c_t indicates the number of customers, and p_t indicates the number of providers. The resource/market competition has the largest effect on the equilibrium price [14].

$$C_c(c_t, p_t) = \frac{c_t}{p_t} \quad (13)$$

Time: At time *t* the negotiation deadline condition $C_{dl}(t)$ of an agent depends on the deadline t_{nd} and negotiation start time t_{ns}.

$$C_{dl}(t) = \frac{t - t_{ns}}{t_{nd} - t_{ns}} \quad (14)$$

The negotiation period is the variation between negotiation start time t_{ns} and negotiation deadline t_{nd}. As deadline is a time-based condition, the well-adopted time-dependent functions, such as Linear (L), Boulware (B) or Conceder (C) are generally used to model how an agent varies its offer with time. These time-based functions are often used in negotiation systems because of their simplicity [10][11]. In this paper, we use a similar model and consider time, market (opportunity and competition) conditions to design new strategy functions for negotiation.

For the broker, we propose the strategy function for a particular issue by considering opportunity, competition and time constraints in equation 15:

$$\alpha(t, c_t, p_t, \gamma) = e^{(\gamma \times C_{dl}(t)^{\left(\frac{C_o(c_t, p_t)}{C_c(c_t, p_t)}\right)})^{\beta}} \times \ln k \quad (15)$$

For the provider, we propose strategy function for a particular issue by considering opportunity, competition and time constraints in equation 16:

$$\alpha(t, c_t, p_t) = e^{(C_{dl}(t)^{\left(\frac{C_o(c_t, p_t)}{C_c(c_t, p_t)}\right)})^{\beta}} \times \ln k \quad (16)$$

In equations 15 and 16, the function $\alpha(.)$ varies from 0 to 1 and guides the changes in the values of an Issue in the subsequent counter offers from its current value to the maximum allowable value within the negotiation deadline. The *k* is a constant value to make sure the value of $\alpha(.)$ varies from 0 to 1.

In equation 15, γ indicates the preference of the Issue considered by the customer. The degree of compensation depends on a parameter β and reflects the conceding nature of the broker. The higher value of β (>1) results in a steeper curve, i.e., faster increment in α with time indicating a more conceding attitude of the negotiating party. The lower value of β (<1) represents the restrictive attitude. The reason for us to design our strategy using exponential and not polynomial models is because the polynomial concedes faster at the beginning than the exponential one, even though both behave similarly on a whole level. For a small value of β the exponential waits longer than the polynomial model before it starts conceding. The objective of broker is to maximize profit by waiting as long as possible to start conceding.

III. PERFORMANCE EVALUATION

We present the performance results obtained from an extensive set of experiments by comparing our proposed heuristics with the most recently proposed heuristic (referred as *base*) [9]. The performance of each proposed heuristic *depends on three factors: time, cost and market constraints*. Therefore, to analyse how these heuristic can achieve customer, broker and provider's objectives, the following experimental scenarios are considered

- ***Impact of negotiation deadline (time factor)***: The impact of 4 sets of negotiation timeframes from the customer's perspective is observed; we use number 1 to 4 to represent the variation from 'very urgent' to 'very relaxed'.
- ***Impact of broker expected margin (cost factor)***: The impact of 4 sets of initial broker expected margins (varying from 20% to 50% over budget), are observed.
- ***Impact of market factor***: The impact of 4 sets of market factors (varying the ratio in relation to the number of providers and customers from less than 10%,

30%, 70%, and more than 90%), are observed. Numbers 1 to 4 are used to represent each set.

A. Experimental Methodology

We implemented a prototype of the framework considering both time and market factors using real data shared with us by cloud provider CA Technologies. CA Technologies offers a number of enterprise software solutions to customers delivered as SaaS. The data provided included the response, refresh and processing times of an enterprise solution hosted on VMs, as measured by the quality assurance team. As SaaS availability depends on the infrastructure availability, this information is collected from CloudHarmony benchmarking system [13], which provides real data from Cloud providers. These data are collected over 4 days including weekdays, weekends and Easter public holiday.

- ***Availability***: Varies from 98.654% (Colosseum) to 100% (Amazon EC2) as derived from Cloud Harmony.
- ***Process Time***: The mean 5.243 (± 2.043) s.
- ***Refresh Time***: The mean 1.581 (± 1.383) s.
- ***Cost***: Cost is considered similar to Windows VMs from 3rd party IaaS providers, which varies from $0.34 per hour (VCloud Express) to $0.46 per hour (Amazon EC2).

We conducted experiments considering 50 concurrent users based on the CA provided data, which is designed according to their customer historic data. The summary of customer data is:

- ***Availability***: uniformly distributed and varies from 99.95% to 100%.
- ***Process Time***: normally distributed mean 1.5 (±1) s.
- ***Refresh Time***: normally distributed mean 2 (± 1) s.

***Software service set*:** consists of 3 editions.

- ***The expected discount percentage***: normally distributed with mean value 30% (variation ± 20%).
- ***The preference level of each QoS parameter***: uniformly distributed between 0 and 1.
- ***Budget****: normally distributed with mean $40 (± $10).*

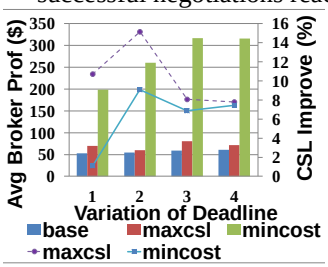
B. Reference Heuristic

For comparing our proposed heuristics, we used the most recent work related to our context on automated negotiation proposed by Zulkernine and Martin [9]. They developed a time-based Sigmond function in their negotiation process for generating counter offers. We however, consider both time and market functions in Clouds. To compare their negotiation strategy, we have implemented their heuristics and Sigmond function with the objective of cost minimization.

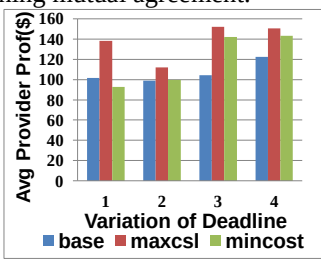
C. Result Analysis

The following performance metrics are considered for evaluation based on the objectives of the negotiating parties:

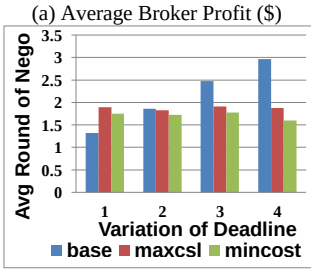
- ***Average broker's profit*:** The broker's average profit from accepted customers.
- ***CSL improvement***: The average CSL improvement over base.
- ***Average provider's profit***: The average provider's profit for accepting customers.
- ***Average round of negotiation:*** The average number of negotiations conducted during the negotiation process to reach mutual agreement.
- ***Number of successful negotiations:*** The number of successful negotiations reaching mutual agreement.



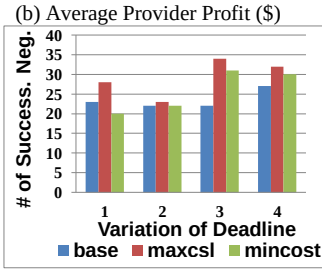
(a) Average Broker Profit ($)



(b) Average Provider Profit ($)



(c) Average Round of Neg.



(d) Number of Successful Neg.

Figure 4: Impact of Deadline Variation

a) ***Variation of negotiation deadline****:* The experiment is designed to evaluate *mincost* and *maxcsl* during negotiation deadline variations.

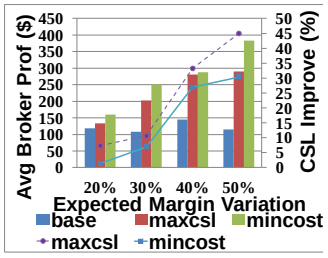
The bar chart in Fig. 4a represents average broker profit while the line chart represents the CSL improvement over *base* heuristic. For all the negotiation deadline variations, *mincost* generates the highest profit (up to 400%) for the broker over *maxcsl* and *base*. The reason for such a trend is that the broker concedes less or bargains harder for more profit. In terms of CSL improvement, *maxcsl* results in the highest improvement (up to 15%) over *base*, since it is designed to sacrifice profit for a higher CSL.

From the providers' perspective (Fig. 4b), on average *maxcsl* generates more profit for providers, because the *maxcsl* aims at satisfying all Issues within the broker's budget, which leaves more profit for providers.

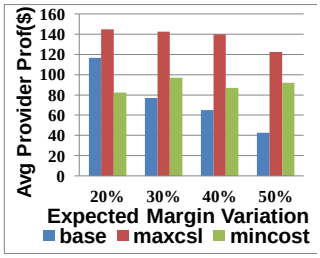
Fig. 4c shows the average negotiation round for *base* increases dramatically when deadlines are varied (as *base* is only time dependent), whereas our proposed heuristics increases slightly (less than 2 rounds), as market factors also impact on the negotiation process. In terms of the number of

successful negotiations (Fig. 4d), when the deadline becomes relaxed, our proposed heuristic performs better and increases in trend, as there is more bargening time.

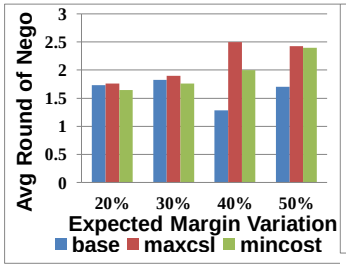
In summary, *mincost* generates more broker profit while *maxcsl* generates improved CSL and increased provider profit by increasing the number of successful negotiations with similar negotiation rounds.



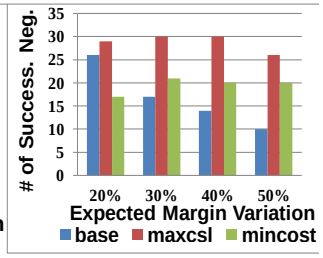
(a) Average Broker Profit ($)



(b) Average Provider Profit ($)



(c) Average Round of Neg.



(d) Number of Successful Neg.

Figure 5. Impact of Variation in Expected Margin

b) ***Variation of initial expected margin:*** As increase in expected margin leads to reduced initial broker budget (cost), the experiment is designed to evaluate *mincost* and *maxcsl* heuristics during the varition of broker costs. The expected margin varies from 20% to 50%, since after 50% the observed trend is similar.

Fig. 5a bar chart depicts that the *mincost* generates the highest profit for the broker, which is up to 200% more than the *base*. The line chart shows that the *maxcsl* has improved CSL by up to 15% over the *mincost.* Fig. 5b shows that the *maxcsl* generates a higher profit for providers when the broker negotiates for higher levels of CSL.

Generally, the average round of negotiations increases for all heuristics when the expected margin increases (Fig. 5c), because when time and market factors are constant, the broker is required to negotiate more rounds with less budget to achieve the objectives and reach agreement.

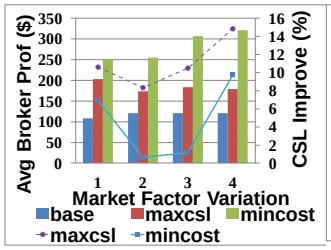
In summary, during expected margin variations, the *mincost* generates more profit for the broker, whereas *maxcls* achieves more profit for the provider as the broker sacrifices cost for securing improved CSL.

c) ***Variation of the market factor*****:** The experiment is conducted to evaluate the proposed heuristics during the variation of market factors. When market factors vary from 1 to 4, which represents an increase in market competition, the *mincost* generates up to twice the profit than the base (Fig. 6a bar chart) and the *maxcsl* improves up to 4 times more CSL compare to *mincost* (Fig.6a line chart). The broker's profit generated by *base* only changes slightly during market factor variations, as *base* does not consider market conditions. When the market factor is 2, the profit and CSL are less due to less rounds of negotiation.

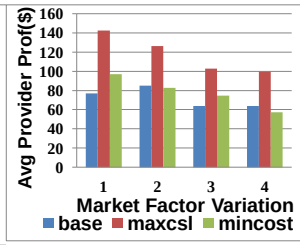
Fig. 6b illustrates that the provider's profit decreases due to an increase in market competition. The *maxcsl* generates more profit for providers than *mincost* and *base*, as *maxcsl* considers the CSL as the highest priority, which leaves more profit for providers.

When competition increases, more negotiation rounds are required to reach agreement (Fig. 6c), as participants bargain harder and the number of opportunities to reach agreement increases (Fig. 6d).

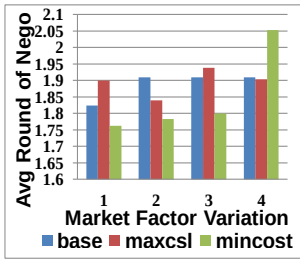
To conclude, the experiment demostrates that *mincost* produces more profit while the *maxcsl* achives better CSL for the broker and more profit for providers.



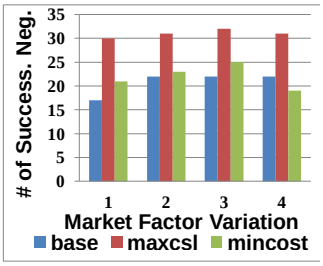
(a) Average Broker Profit ($)



(b) Average Provider Profit ($)



(c) Average Round of Neg.



(d) Number of Successful Neg.

Figure 6. Impact of Market Factor Variation

IV. RELATED WORKS

With the advancement of web technology, various approaches of resource allocation have been developed for distributed systems [17]. Current literature indicates that research focusing on resource allocation is rapidly growing. However questions remain as to whether multi-agent systems can be adopted in the domain of resource allocation. In this context several multi-agent approaches were developed to leverage the wide applicability and efficient adoption of multi-agent systems for the heterogeneous domain [18]. However, these approaches have some limitations when applied to Cloud. For example, most popular strategies such as Game theory [19], Reinforcement Learning [20] and Markov Decision Process (MDP) [21] require either expensive storage of each status or that every agent is required to expose tactics to opponents. Therefore, these approaches are not applicable for Cloud where private information such as the number of utilized resources is not

advertised. Other approaches such as fuzzy similarity and adaptive fuzzy logic [22], lead to inaccurate negotiation, and thus, result in failed negotiations.

Faratin et al. presented a formal model of negotiation between autonomous agents in service-oriented environments [3]. Chhetri, et al. proposed an agent-based negotiation architecture for coordinated negotiation in service composition [4]. Comuzzi and Pernici proposed a negotiation broker framework to support semi-automated or fully automated negotiation of QoS for service selection [10]. Similarly, Zulkernine et al. proposed a policy based negotiation broker framework for automated negotiation of SLA's [9]. Dastjerdi and Buyya proposed negotiation strategies for infrastructure layer in Cloud which depends on provider resource capabilities [24]. These approaches have not considered elements such as CSL objectives, broker's profit, and market factors in their algorithms.

V. Conclusions and Future work

In Clouds, SLA is a legal contract between the consumer and provider to guarantee the QoS. Negotiation is essential for both participants to feel comfortable about meeting their objectives prior to SLA finalization. In this paper, we proposed a novel negotiation framework which included strategies and decision making heuristics by considering factors such as time, market constraints and trade-offs.

Our two proposed algorithms have been evaluated by using real data from a cloud-hosted enterprise software solution provided by CA Technologies. Results show that our proposed heuristics minimize cost or maximize CSL in comparison to the most recently proposed base heuristic.

In the future, we plan to evaluate additional issues in the context of SaaS in Cloud by considering trade-off between cost and CSL. Moreover, the penalty for negotiation failure from the consumer's perspective (e.g. no service offered for a consumer request) will be considered. We will also investigate the renegotiations by considering the dynamic changes of customer needs.

Acknowledgments

This work is supported by the Australian Research Council (ARC) via Discovery/Linkage Project grants. We thank our colleagues Rodrigo Calheiros and Bevan Mailman for their comments on improving the paper, Shanshan Wu for implementing and testing part of the system.

References

[1] K. Chao, R. Anane, J. H. Chen, and R. Gatward, (2002). Negotiating Agents in a Market-Oriented Grid. In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002), Berlin, Germany.

[2] K. M. Sim, (2006). A Survey of Bargaining Models for Grid Resource Allocation. ACM SIGECOM:E-Commerce Exchange, 5(5), pp. 22–32.

[3] P. Faratin, C. Sierra, and N. R. Jennings, (1998). Negotiation Decision Functions for Autonomous Agents. Rototics and Autonomous System 24 (3-4), pp. 159-182.

[4] M. Chhetri, et. al., (2006). A Coordinated Architecture for the Agent-based Service Level Agreement Negotiation of Web Service Composition. In Proceedings of Australian Software Engineering Conference. (ASWEC), Sydney, Australia.

[5] M. Comuzzi, and B. Pernici, (2005). An Architecture for Flexible Web Service QoS Negotiation. In Proceeding of the 9th IEEE International Enterprise Computing Conference, The Netherlands.

[6] F. Zulkernine, et al., (2009). In A Policy-Based Middleware for Web Services SLA Negotiation. IEEE International Conference on Web Serivce (ICWS), pp. 1043-1050.

[7] J. Akhani, S. Chaudhary, and G. Somani, (2011). Negotiation for Resource Allocation in IaaS Cloud, In Proceedings of the 4th Annual ACM Bangalore Conference, Bangalore, India.

[8] J. Brzostowski and R. Kowalczy, (2006). Adaptive Negotiation with On-Line Prediction of Opponent Behaviour in Agent-Based Negotiations. In Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology, HongKong, China.

[9] F. Zukernine and P. Martin, (2011). An adaptive and intelligent SLA negotiation system for web services. IEEE Transactions of Service Computing, vol. 4, no. 1, pp. 31-43.

[10] M. Shell, M. Comuzzi, and B. Pernici, (2007). An Architecture for Flexible Web Service QoS Negotiation. In Proceedings of the 1st IEEE International Enterprise Distributed Object Computing (EDOC) Conference, Maryland, USA.

[11] H. Li, S. Su, and H. Lam, (2006). On Automated e-Business Negotiations: Goal, Policy, Strategy and Plans of Decision and Action, Journal of Organizational Computing and Electronic Commerce, vol. 13, no. 1, pp. 1-29.

[12] http://vitlive.com/, accessed on 10th April 2012.

[13] http://www.cloudharmony.com/, accessed on 6th April 2012.

[14] M. Comuzzi, and B. Pernici, (2009). A Framework for the QoS-Based Web Service Contracting, ACM Transaction on the Web, vol. 3, no. 3.

[15] http://sites.google.com/site/gistcloudresearchgroup/automated-sla-negotiation, accessed on 10th April 2012.

[16] S. K. Garg, C. Vecchiola, and R. Buyya, (2012). Mandi: A Market Exchange for Trading Utility and Cloud Computing Services. The Journal of Supercomputing, DOI: 10.1007/s11227-011-0568-6.

[17] K. Czajkowski, I. Foster, and C. Kesselman, (1999). Resource Co-Allocation in Computational Grids. In Proc. 8th IEEE Symp. On High Performance Distributed Computing.

[18] J. W. Cao, D. P. Spooner, and G. R. Nudd, (2002). Agent-based Resource Management for Grid Computing. In Proceedings of 2nd Intl. Symposium on Cluster Computing and the Grid, Germany.

[19] K. Binmore, and N. Vulkan, (1997). Applying Game Theory to Automated Negotiation. Paper prepared for DIMACS Workshop on Economics, Game Theory and the Internet.

[20] S. Arai, K. Sycara, and T. Payne, (2000). Experience-Learning in based Reinforcement Learning to Acquire Multi-Agent Domain. The Sixth Pacific Rim International Conference on Artificial. Intelligence, Springer-Verlag.

[21] F. Teuteberg, and K. Kurbel, (2002). Anticipating Agents' Negotiation Strategies in an E-marketplace Using Belief Models. In Proceedings of 5th Intl. Conference on Business Information System, Poland.

[22] P. Faratin, et. al., (2000). Using Similarity Criteria to Make Negotiation Trade-Offs. In Proceedings of 4th International Conference on Multi-Agent Systems, pp.d119-126, Boston, USA.

[23] M. Comuzzi and B. Pernici, (2009). A Framework for the QoS-Based Web Service Contracting, ACM Transaction on the Web, 3(3) pp.1-10,

[24] A. V. Dastjerdi and R. Buyya, (2012). An Autonomous Reliability-Aware Negotiation Strategy for Cloud Computing Environments. In Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid), Ottawa, Canada.

[25] http://www.cordys.com/cordys-for-cloud-brokers, access on 10th April 2012.

Appendix A

1.1. Negotiation Policy Specification

The negotiation policy specifications are used to specify QoS parameters, which are to be negotiated and the acceptable range of them to reach the mutual agreement. In this section, we propose the QoS model and policy specification.

A. QoS Model

Different participants' using different terms is one of the critical challenges in SLA negotiation [23]. In our framework, a QoS model is used to provide shared knowledge about QoS attributes among negotiating participants. A QoS model defines a set of QoS dimensions. Each QoS dimension represents a specific quality aspect of a service, such as refresh time, availability, and price. In our QoS model, a quality dimension is defined using: a title, a category, a name, a description, and a metric. The QoS model is shared among service consumers and service providers. Thus, they have a common understanding on the QoS attributes about how they are defined, how they are measured, and so on. In this paper, we consider the following QoS dimensions – price, refresh time, process time and availability. These dimensions are the ones that are mostly used and they are domain-independent. Our QoS model can be easily extended to include other QoS dimensions.

Before negotiation, both participants specify the rule of QoS parameter in a policy specification. The policy usually refers to a high-level description of goals to be achieved and actions to be taken in different situations.

B. Policy Specification

Our policy specification is inspired by WS-Policy and XACML. WS-Policy is a XML-based specification, in which assertions are basic blocks [25]. Each assertion defines domain specific constrains, capabilities, and requirements. However, the WS-Policy framework does not provide any assertion, and therefore users of this framework need to develop their own assertions. XACML is a XML-based language which is standardized by OASIS and has been successfully used widely as access-control policy languages [23]. With XACML, the QoS parameter constraints can be domain-independent, because XACML is based on generic data type. However, both of them are only machine-readable but not human-readable, especially for non-IT background users. Therefore, based on the concept of constraints and goals in WS-Policy and XACML, we design our domain-independent policy in a both human-readable and machine-readable manner by providing web user interface to register constraints (rules) and goals.

The main concepts of our policy specification are rules and goals:

- The rules: are used to specify the QoS parameters and the acceptable range of these parameters (Fig. 2).
- The goals: are non-negotiable rules.

Moreover, in order to take care of different policy rules from different agents we provide a rule register to extend policy flexibly.

Figure 2: Negotiation Rule Register Web Form.

In Fig. 2, the rule names are QoS parameters. The lower value and upper value fields are lower and upper bounds of the rule value. If a rule does not exist, there is another interface to register new rule names. Any policy and rule registered by providers are stored in Policy DB component of the framework. The NPT component matches these policies with customer QoS parameters during the negotiation.

1.2. Negotiation Protocol

The negotiation protocol refers to a set of rules, steps or sequences during the negotiation process, aiming at SLA establishment. It covers the negotiation states (e.g. propose offer, accept/reject offer, and terminate negotiation). It is common to characterize negotiations by their settings: bilateral, one-to-many, or many-to-many. In this paper we focus on the one-to-many bargaining setting, where we consider three types of agents (CA, BCA and PA). A BCA negotiates with many PAs in a bilateral fashion.

During the negotiation process, the negotiation status is updated using negotiation states described in Table1.

Table 1. The Negotiation States and Description Summary

States	Description
PROPOSE	The agent propose initial or counter offer to the opponent agent.
REJECT	The agent does not accept the offer proposed by the opponent agent.
ACCEPT	The agent accepts the offer proposed by the opponent agent.
FAILURE	System failure, trigger renegotiation.
TERMINATE	Negotiation is terminated due to timeout or no mutual agreement.

In our framework, the sequential negotiation process is described as follows and depicted in Fig. 3:

Phase 1: *CA submits requests*: CA requests services on behalf of the customer to the Broker.

Phase 2: The BCA requests initial proposals from all providers, who are registered in the Directory. The values sent from BCA to PAs are expected values.

Phase 3: *PAs propose initial offer:* All PAs propose initial offers based on their current capabilities and availability to fulfil BCA's requirements.

Phase 4: *Negotiation Process with PAs:*

a). If there are providers who can fulfil all requirements, then the BCA selects the best vendor.

b). If there is no provider that can fulfil all requirements, then the BCA starts the negotiation process with PAs.

*Step 1***:** BCA selects the best initial offer from all offers that are proposed by all providers according to the objective.

*Step 2***:** BCA adjusts its initial offer according to the offer selected in **Step 1** to generate new counter offer and propose it to all providers.

*Step 3***:** A PA evaluates BCA's counter proposal.

Step 4: If the counter offer proposed by BCA cannot be accepted, PA proposes a counter offer.

*Step 5***:** Terminate negotiation. There are three termination conditions: First, when negotiation deadline expires. Second, when the offer is mutual agreed by both the CA and the PA. Third, when BCA is not able to accept any counter offer proposed by all providers within the negotiation deadline.

Phase 5: *SLA Generation:* Initiate SLA creator to generate SLA for customer and provider respectively using SLA templates stored in KB.

Phase 6: *Send SLA to all participants:* The generated SLA will be sent to the customer and provider respectively by the SLA creator.

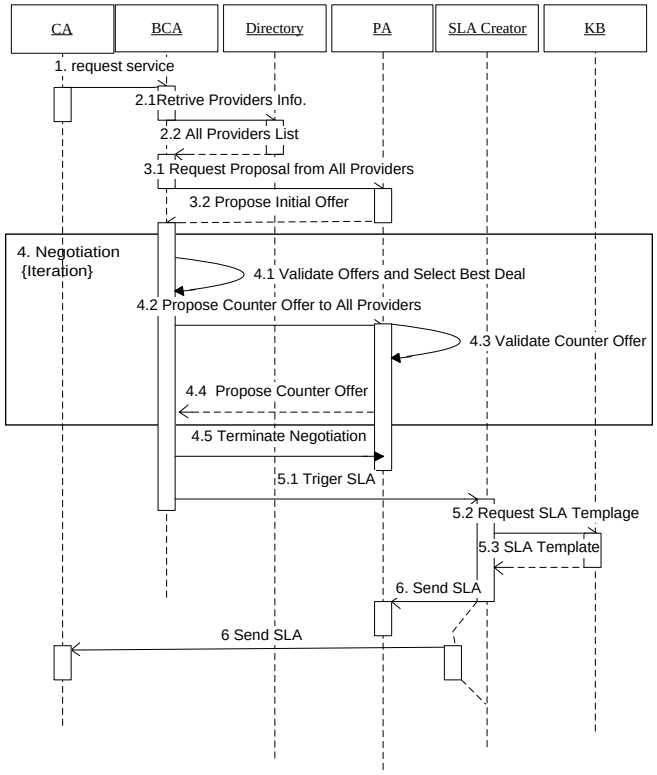


Figure 3. The Interaction between Components During Negotiation Process**.**