2017

A concurrent interdependent service level agreement negotiation protocol in dynamic service-oriented computing environments

Lei Niu
University of Wollongong, ln982@uowmail.edu.au

Fenghui Ren
University of Wollongong, fren@uow.edu.au

Minjie Zhang
University of Wollongong, minjie@uow.edu.au

A concurrent interdependent service level agreement negotiation protocol in dynamic service-oriented computing environments

Abstract

Service Level Agreement (SLA) negotiations are capable of helping define the quality of service in order to meet the customer's service requirements. To date, a large number of negotiation protocols are proposed to handle single SLA negotiations, but little work can be found in handling multiple interdependent SLA negotiations in dynamic negotiation environments. This paper proposes an adaptive protocol for concurrently handling multiple interdependent SLA negotiations in dynamic environments. First, interdependencies between SLA negotiations are represented by a graph-based model. Then, an updating mechanism is proposed to handle the dynamism of multiple SLA negotiations. By applying the proposed updating mechanism, a protocol for concurrently processing SLA negotiations in dynamic environments with unexpected changes of service requests is presented. Experimental results show that the proposed approach can effectively handle unexpected changes of service requests from customers in dynamic environments, and successfully lead multiple SLA negotiations to agreements aligning with customers.

Publication Details

Niu, L., Ren, F. & Zhang, M. (2017). A concurrent interdependent service level agreement negotiation protocol in dynamic service-oriented computing environments. Lecture Notes in Computer Science, 10570 132-147. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)

A Concurrent Interdependent Service Level Agreement Negotiation Protocol in Dynamic Service-oriented Computing Environments

Lei Niu, Fenghui Ren, and Minjie Zhang

School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia
ln982@uowmail.edu.au, {fren,minjie}@uow.edu.au

Abstract. Service Level Agreement (SLA) negotiations are capable of helping define the quality of service in order to meet the customer's service requirements. To date, a large number of negotiation protocols are proposed to handle single SLA negotiations, but little work can be found in handling multiple interdependent SLA negotiations in dynamic negotiation environments. This paper proposes an adaptive protocol for concurrently handling multiple interdependent SLA negotiations in dynamic environments. First, interdependencies between SLA negotiations are represented by a graph-based model. Then, an updating mechanism is proposed to handle the dynamism of multiple SLA negotiations. By applying the proposed updating mechanism, a protocol for concurrently processing SLA negotiations in dynamic environments with unexpected changes of service requests is presented. Experimental results show that the proposed approach can effectively handle unexpected changes of service requests from customers in dynamic environments, and successfully lead multiple SLA negotiations to agreements aligning with customers.

Keywords: service level agreement, negotiation protocol, dynamic negotiation environment

1 Introduction

A Service Level Agreement (SLA) gives a definition of different aspects of web-based services between service providers and consumers. These aspects of the services, i.e., qualities of service properties (e.g., price, responding time, failure possibility, etc.), are important in cloud services [1]. SLA negotiations provide the procedure to reach service agreements between service providers and service customers in a cloud computing environment. To date, most of the previous work [2], [10], [13] deals nearly exclusively with single SLA negotiations involving independent negotiation goals. However, SLA negotiation problems in the real world contain complex interdependency relationships between multiple SLA negotiations as well as the dynamic changes of the negotiation environment. For instance, in service-oriented cloud computing, a customer might ask for a number of cloud services by processing multiple SLA negotiations. Interdependency

relationships exist between these SLA negotiations, where each service's process somehow impacts the process of other services. In order to maximize a customer's profit, concurrently processing these interdependent SLA negotiations is the optimal solution. Moreover, in the dynamic cloud computing environment, the customer may change their original requests during the negotiation, i.e., adding new service requests or cancelling ongoing service requests.

In order to solve the research problems in concurrently handling multiple SLA negotiations in dynamic negotiation environments, a sophisticated SLA negotiation protocol should be carefully defined by considering the interdependencies between SLA negotiations and the dynamism of the SLA negotiation environment. Three research challenges need to be addressed in the SLA negotiation protocol design. First, interdependency relationships between different cloud services impact the procedures and outcomes of other SLA negotiations. Therefore, the first challenging problem is how to model the interdependencies between multiple SLA negotiations by considering their procedures. Second, in order to concurrently process SLA negotiations, the second challenging problem is how to handle the concurrency of multiple SLA negotiations based on the customer's service requests by considering their interdependencies. Third, in dynamic service-oriented computing environments, the customer probably changes his/her requests of services (i.e., requesting new cloud services or cancelling existing service requests under negotiated) during the process of multiple SLA negotiations. With requests being changed, interdependencies between SLA negotiations will also be changed dynamically, and it will cause a chain-like impact on outcomes of other ongoing SLA negotiations and on whether the customer's overall goal will be achieved. Hence, the third challenging problem is how to handle dynamic changes of SLA negotiations in dynamic negotiation environments.

From present literature, Messina et al. [2] proposed a negotiation protocol for service level agreements. Dastjerdi and Buyya [13] proposed a solution to automate the negotiation process for cloud environments. These approaches both focus on single SLA negotiation, which has only one negotiation goal, and they do not work in the scenario of multiple SLA negotiations with interdependencies. In our previous work [5], a negotiation protocol is proposed to concurrently handle multiple negotiations in a static negotiation environment, where any change of negotiations is not allowed during the negotiation. The previous work is not suitable for dynamic environments, especially in service-oriented computing environments, where service requests from customers could change at any time during the negotiation. Therefore, little work from these present literature simultaneously addresses the challenges in SLA negotiation protocol designs.

In order to address the three research challenges, this paper proposes an effective negotiation protocol for handling multiple SLA negotiations in dynamic negotiation environments. In the proposed SLA negotiation protocol, interdependencies between SLA negotiations are mathematically represented by a peer-to-peer graph model. Furthermore, through updating (1) the peer-to-peer graph model, (2) the utility representations of SLA negotiations and (3) the colored petri net representation by modifying the transitions and arcs, the dynamic

changes of service requests from customers can be effectively handled.

The organisation of this paper is as follows. Section 2 introduces a model for multiple SLA negotiations. Section 3 presents a colored petri net representation of multiple SLA negotiations. Section 4 proposes an updating mechanism for handling dynamism of SLA negotiations, and Section 5 proposes a negotiation protocol for handling multiple SLA negotiations in dynamic environments. Section 6 gives experimental results and the analysis. Section 7 and Section 8 present related work and conclusion.

2 A Model for Multiple SLA Negotiations

This section introduces a model for multiple SLA negotiations, which includes how to represent the interdependencies between multiple SLA negotiations, how to describe the overall goal of a customer on multiple SLA negotiations and how to calculate the overall utility from multiple interdependent SLA negotiations.

2.1 Definitions

Definition 1 (Graph Representation of SLA Negotiations) *Multiple SLA negotiations* $\mathbb{N} = \{A_0, \cdots, A_i, \cdots, A_{n-1}\}$ $(i \geq 0)$ *are represented by a directed graph* $G =< V, E >$*, where Set* V *indicates single SLA negotiations, and Set* E *indicates interdependencies between SLA negotiations, and* $e_{ij} = (A_i, A_j) \in E$ *indicates an interdependency between SLA negotiations* A_i *and* A_j*.*

The interdependency relationship in this paper is a kind of *issue interdependency*. For SLA negotiations $A_i, A_j \in \mathbb{N}$, the interdependency of "$A_i \propto A_j$" indicates that A_j depends on A_i, which means that the offer in A_j is dependent on the offer in A_i. Interdependency between SLA negotiations in this paper has "**unidirectionality**" and "**transitivity**" properties. The "**unidirectionality**" property indicates that the interdependency between SLA negotiations is a one-way relationship, and the "**transitivity**" property indicates that the interdependency between SLA negotiations can be transferred. Based on the interdependency between SLA negotiations, we define the connection between a series of interdependent SLA negotiations as a *SLA-negotiation thread* in Definition 2.

Definition 2 (SLA-Negotiation Thread (SLANT)) *A SLANT is defined as a set* $\mathbb{R}_i = \{r_{i,0}, \cdots, r_{i,j}, \cdots, r_{i,k_i-1}\}$*, where* $\mathbb{R}_i \neq \varnothing$*,* $\bigcup_{i=0}^{l-1} \mathbb{R}_i = \mathbb{N}$*,* $|\mathbb{R}_i| = k_i$ *and* l *indicates the total number of SLANTs in SLA negotiations. "* $r_{i,j-1}$ *" indicates the* j*th SLA negotiation in SLANT* $\mathbb{R}_i$*.*

In an SLANT $\mathbb{R}_i = \{r_{i,0}, \cdots, r_{i,j}, \cdots, r_{i,k_i-1}\}$, it satisfies $r_{i,0} \propto \cdots \propto r_{i,j} \propto \cdots \propto r_{i,k_i-1}$, and there is no such $r_{i,j}$ that for $\forall\, r_{i,j} \in \mathbb{R}_i$, $r_{i,j} \propto r_{i,0}$ and $r_{i,k_i-1} \propto r_{i,j}$ hold. That is to say, the relationship of SLA negotiations involved in an SLANT is chain-like interdependent, i.e., the $(j+1)$th SLA negotiation depends on the jth SLA negotiation in an SLANT. In an SLANT, if all involved SLA negotiations reach successful negotiation outcomes, this SLANT is successful. Then, the overall goal of a customer on SLA negotiations is defined as follows.

Definition 3 (SLA Multiple Negotiation Goal (SLA-MNG)) *The SLA-MNG of a customer on SLA negotiations is classified as follows based on the expected success on the number of SLANTs in SLA negotiations.*

- **Complete Success Goal***: A complete success goal will be achieved if all SLANTs are successful.*
- **Partial Success Goal***: A partial success goal will be achieved if not all but at least one SLANT is successful.*

Let $\Omega_{\mathbb{N}} \in [0,1]$ denote the value of an SLA-MNG, where $\Omega_{\mathbb{N}} = 1$ indicates a complete success goal, and $\Omega_{\mathbb{N}} = \frac{s}{q} \in (0,1)$ indicates a partial success goal.

$$\Omega_{\mathbb{N}} = \begin{cases} 1 & \text{if all SLANTs are successful,} \\ \frac{s}{q} & \text{if not all but at least one SLANT is successful,} \end{cases} \tag{1}$$

where s indicates the number of successful SLANTs and q indicates the number of all involved SLANTs in multiple SLA negotiations.

Definition 4 (Overall Utility from Multiple SLA Negotiations) *The overall utility from multiple SLA negotiations* $\mathbb{N}$ *is represented by* $U(\mathbb{N})$*, which can be calculated as follows.*

$$U(\mathbb{N}) = \frac{\sum\limits_{\forall \mathbb{R}_i \in \mathbb{N}} \Big(U(\mathbb{R}_i) \times V(\mathbb{R}_i) \Big)}{l}, \tag{2}$$

where $U(\mathbb{N}) \in [0,1]$*,* $i \geq 0$*,* l *indicates the total number of involved SLANTs in SLA negotiations* $\mathbb{N}$*,* $V(\mathbb{R}_i)$ *is the outcome of SLANT* $\mathbb{R}_i$ *which is calculated by Equation (3), and* $U(\mathbb{R}_i)$ *is the utility from SLANT* $\mathbb{R}_i$ *which is calculated by Equation (4).*

$$V(\mathbb{R}_i) = \prod\nolimits_{\forall A_j \in \mathbb{R}_i} V(A_j) \ (i \geq 0, j \geq 0) \tag{3}$$

$$U(\mathbb{R}_i) = \sum_{\forall A_j \in \mathbb{R}_i} \Big(\omega_j \times U(A_j) \times V(A_j) \Big) \ (i \geq 0, j \geq 0), \tag{4}$$

where $U(\mathbb{R}_i) \in [0,1]$, and $\omega_j \in [0,1]$ represents a customer's preference on SLA negotiation A_j, where $\omega = (\omega_0, \cdots \omega_j, \cdots \omega_{n-1})$, and $\sum_{j=0}^{n-1} \omega_j = 1$.

In Equations (3) and (4), $V(A_j)$ indicates the outcome of SLA negotiation A_j which is defined by Equation (5), and $U(A_j)$ indicates the utility from SLA negotiation A_j which can be calculated by existing methods of single negotiation utility calculation.

$$V(A_j) = \begin{cases} 0 & \text{if negotiation } A_j \text{ is failed,} \\ 1 & \text{if negotiation } A_j \text{ is successful.} \end{cases} \tag{5}$$

Additionally, $V(\mathbb{N})$ indicates the outcome of multiple SLA negotiations $\mathbb{N}$, which is calculated by Equation (6).

$$V(\mathbb{N}) = \frac{\sum_{\forall \mathbb{R}_i \in \mathbb{N}} V(\mathbb{R}_i)}{l} \ (i \geq 0) \tag{6}$$

where l is total number of SLANTs in multiple SLA negotiations $\mathbb{N}$.

3 A CPN Representation of Multiple SLA Negotiations

In order to handle concurrency of SLA negotiations, our protocol employs colored petri nets (CPNs) because CPN is a solid tool in processing systems with concurrency [4]. In the proposed CPN representation of multiple SLA negotiations, transitions represent SLA negotiations, and places represent states of SLA negotiations. The inputs and outputs of SLA negotiations are shown by arc directions, and Token $(A, m)(m \geq 1)$ indicates that one offer is received in the mth negotiation round in SLA negotiation A (i.e., enable transition t_A). In a CPN representation, the first place and last place of each SLANT are the initial place and the final place, respectively. Each initial place contains at least one token to fire following transitions. The SLANT $\mathbb{R}_i$ is the ith SLANT (refer to Definition 2) in multiple SLA negotiations $\mathbb{N}$.

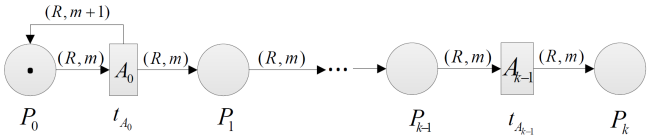

Fig. 1. The CPN representation of an SLA-negotiation thread

Fig. 1 shows the CPN representation of an SLANT $\mathbb{R}_i = \{A_0, A_1, \cdots A_j, \cdots A_{k-1}\}(0 \leq j \leq k-1)$, a token (R, m) in the initial place (i.e., Place P_0) of an SLANT $\mathbb{R}_i$ is used to activate the SLANT $\mathbb{R}_i$. If Transition t_{A_j} is activated by a token, it means that SLA negotiation A_j finishes a negotiation round. If SLA negotiation A_j is failed, Transition t_{A_j} will not be activated, and the conduct of SLANT $\mathbb{R}_i$ will be terminated. If SLA negotiation A_j is not failed, the token will fire the following transitions in SLANT $\mathbb{R}_i$ based on the arc directions. If all SLA transitions in SLANT $\mathbb{R}_i$ have been activated, we call $\mathbb{R}_i$ finishes an SLANT round. In order to handle the concurrency of SLA negotiations, a backward arc (i.e., from Transition t_{A_0} to Place P_0) is added to show that different SLA negotiations can be performed concurrently.

4 The Updating Mechanism for Handling Dynamism

This section proposes an updating mechanism for handling dynamic changes in multiple SLA negotiations, which includes a graph updating, a utility updating and a colored petri net (CPN) representation updating.

4.1 Graph Updating

(1) Adding SLA negotiations

Fig. 2(a) shows the graph updating of adding SLA negotiations A_3 and A_4.

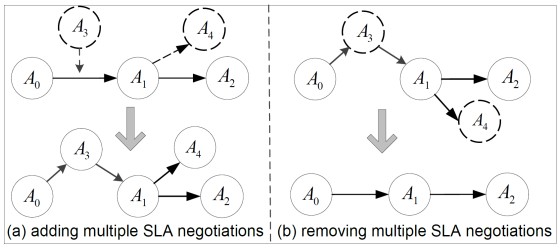

Fig. 2. A graph updating example of changing multiple SLA negotiations

In Fig. 2(a), SLA negotiations A_3 and A_4 are added, and Edge (A_0, A_1) is removed as the interdependency relationship of $A_0 \propto A_1$ satisfies the property of **Transitivity**. Thus, it is not necessary to keep Edge (A_0, A_1) after the creation of Edges (A_0, A_3) and (A_3, A_1).

(2) Removing SLA negotiations

Fig. 2(b) shows the graph updating of removing SLA negotiations A_3 and A_4. In Fig. 2(b), Edges (A_0, A_3) and (A_3, A_1) are removed, and Edge (A_0, A_1) is added. Based on the property of **Transitivity**, Edge (A_0, A_1) is added to indicate the existing interdependency between SLA negotiations A_0 and A_1.

4.2 Utility Updating

According to the calculation of the overall utility from SLA negotiations (refer to Definition 4), the overall utility can also be calculated as follows by combining Equations (2) (3) and (4).

$$U(\mathbb{N}) = \frac{\sum\limits_{\forall \mathbb{R}_i \in \mathbb{N}} \Big(\sum\limits_{\forall A_j \in \mathbb{R}_i} \Big(\omega_j \times U(A_j) \times V(A_j) \Big) \times \prod\limits_{\forall A_j \in \mathbb{R}_i} V(A_j) \Big)}{l}, \tag{7}$$

The values of $U(A_j)$ and $V(A_j)$ are updated based on the conduct of ongoing SLA negotiation A_j. Therefore, in order to get the updated overall utility from SLA negotiations, only the mechanism for updating preferences of SLA negotiations needs to be designed because the weight of preference distributions on unchanged SLA negotiations should be kept. The mechanism for updating preferences of SLA negotiations is as follows.

Let ω_i indicate the preference of SLA negotiation A_i. The set of modified SLA negotiations is $\mathbb{N}_0 = \{A_{n+p} | 1 \leq p \leq m\}$.

(1) Adding SLA negotiations

In SLA negotiations $\mathbb{N} = \{A_1, \cdots A_j, \cdots A_n\}$, preferences of the updated SLA negotiations $\mathbb{N}' = \{A_1, \cdots A_j, \cdots A_n, A_{n+1}, \cdots, A_{n+m}\}$ are calculated as follows.

$$\omega'_j = \frac{\omega_j}{1 + \sum_{p=1}^{m} \omega_{n+p}} \tag{8}$$

(2) Removing SLA negotiations

In SLA negotiations $\mathbb{N} = \{A_1, \cdots A_j, \cdots A_n, A_{n+1}, \cdots A_{n+m}\}$, preferences of the

updated SLA negotiations $\mathbb{N}' = \{A_1, \cdots A_j, \cdots A_n\}$ are calculated as follows.

$$\omega'_j = \frac{\omega_j}{1 - \sum_{p=1}^{m} \omega_{n+p}} \tag{9}$$

4.3 Colored Petri Net Representation Updating

The Colored Petri Net (CPN) representation of multiple SLA negotiations is transferred from its corresponding graph representation. Therefore, the mechanism of CPN representation updating is similar with graph updating mechanism. In the graph representation, an SLA negotiation is represented by a node, and the interdependency is represented by a directed edge. In the CPN representation, an SLA negotiation is represented by a transition, and the interdependency is represented by an arc. Therefore, updating a CPN representation obeys similar rules with updating its graph representation. The only difference is that changing transitions in a CPN representation accompanies with changing corresponding places. Due to page limitation, readers can find details in related literature [6],[7].

5 A Negotiation Protocol for SLA Negotiations in Dynamic Environments

In this section, a negotiation protocol for processing multiple SLA negotiations in dynamic environments is proposed.

The inputs of Algorithm 1 are the set of SLA negotiations $\mathbb{N}$ and the set of updated SLA negotiations $\mathbb{N}'$, and the outputs of the algorithm are the overall utility from SLA negotiations and the result (i.e. success or failure). At the beginning, the algorithm generates the graph based on the interdependencies between SLA negotiations, generates the corresponding CPN representation $C_{\mathbb{N}}$ and then starts executing CPN (Lines 1-3). If dynamic changes happen, the algorithm executes the proposed updating mechanism to get the updated CPN representation $C_{\mathbb{N}'}$. Then, the algorithm keeps executing the updated CPN $C_{\mathbb{N}'}$ (Lines 6-8). If every SLA negotiation finishes a new negotiation round, the algorithm computes $V(\mathbb{N}')$ and $U(\mathbb{N}')$ while considering the utility updating (Lines 9-10). If the value of $V(\mathbb{N}')$ is less than the value of SLA-MNG $\Omega_{\mathbb{N}'}$, the algorithm keeps executing the updated CPN. Otherwise, the algorithm terminates the updated CPN and quits. The algorithm shows that if the SLA-MNG $\Omega_{\mathbb{N}}$ (refer to Definition 3) is achieved, it returns the overall utility from SLA negotiations $U(\mathbb{N}')$ and "success" as the results (Lines 11-16). Because it is not necessary to execute other unfinished SLA negotiations, it can improve efficiency in some extent. If no SLA negotiations are required to be changed during the negotiation, the algorithm keeps executing CPN $C_{\mathbb{N}}$ and compares the values of $V(\mathbb{N})$ and $U(\mathbb{N})$ to decide whether to keep executing CPN or to terminate it (Lines 18-20). If the algorithm completely finishes executing CPN and the SLA-MNG is not achieved, the algorithm returns "failure" (Lines 22-24).

Algorithm 1 SLA negotiation protocol in dynamic environments

Input: the set of SLA negotiations $\mathbb{N}$, the set of updated SLA negotiations $\mathbb{N}'$
Output: the overall utility from multiple SLA negotiations, and success or failure.

Generate the graph based on the interdependency of SLA negotiations $\mathbb{N}$;
Generate the corresponding CPN representation $C_{\mathbb{N}}$;
Start executing CPN;
while CPN is not completed **do**
    Keep executing the CPN to concurrently process SLA negotiations;
    if negotiations are requested to be added or removed **then**
        Execute updating graph and CPN representation to get updated CPN representation $C_{\mathbb{N}'}$;
        Keep executing the updated CPN $C_{\mathbb{N}'}$;
        while every SLA negotiation finishes a new negotiation round **do**
            calculate $V(\mathbb{N}')$ and $U(\mathbb{N}')$ while considering the utility updating;
            if $V(\mathbb{N}') < \Omega_{\mathbb{N}'}$ **then**
                keep executing the updated CPN;
            else if $V(\mathbb{N}') \geq \Omega_{\mathbb{N}'}$ **then**
                terminate the updated CPN and quit;
                return $U(\mathbb{N}')$ and success;
            end if;
        end while;
    else if no SLA negotiations are requested to be changed **then**
        $C_{\mathbb{N}'} = C_{\mathbb{N}}$;
    end if;
end while;
if $V(\mathbb{N}) < \Omega_{\mathbb{N}}$ or $V(\mathbb{N}') < \Omega_{\mathbb{N}'}$ **then**
    return failure;
end if.

6 Experiment

In order to better simulate the performance of the proposed protocol, an experiment is conducted by employing CloudSim [9], which is a cloud computing simulator. In the experiment, the proposed protocol is tested by modifying the classes of "Datacenter" and "DatacenterBroker" in CloudSim. The detail experimental settings are described as follows.

6.1 Experimental Settings for Static SLA Negotiations

In the experimental settings, we assume that a cloud customer would like to apply a series of cloud services with interdependency relationships. Let $S = \{s_0, s_1, \cdots s_i, \cdots, s_n\}$ denote the cloud service set, where s_i indicates an individual cloud service. In order to conveniently test the proposed protocol in dynamic environments, interdependency relationships between services in static environments is given in Fig. 3 (a).

In Fig. 3, a customer applies four cloud services $S = \{s_0, s_1, s_2, s_3\}$ with

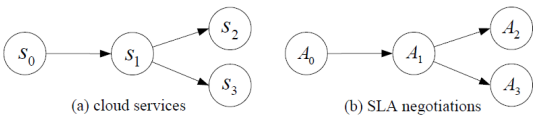

Fig. 3. The relationship between (a) cloud services and (b) SLA negotiations

the shown interdependency relationships. The cloud services are being negotiated through corresponding SLA negotiations $\mathbb{N} = \{A_0, A_1, A_2, A_3\}$. There are two SLANTs (refer to Definition 2) in Fig. 3, where $\mathbb{R}_0 = \{A_0, A_1, A_2\}$, $\mathbb{R}_1 = \{A_0, A_1, A_3\}$.

The main purpose of this experiment is to test the effectiveness of the proposed protocol in a dynamic negotiation environment. Therefore, a single-issue negotiation model [8] is employed to conduct each SLA negotiation, where the utility function for individual SLA negotiations is described by Equation (10).

$$U(\text{counter offer}) = \frac{\text{reserved offer} - \text{counter offer}}{\text{reserved offer} - \text{initial offer}} \tag{10}$$

The parameters for each SLA negotiation are described in Table 1. The preferences for SLA negotiations are selected randomly, and concession strategies for SLA negotiations are randomly picked up from Conceder, Linear and Boulware strategies [3].

Table 1. Parameters for Single SLA Negotiations

customer	initial offer	random from $[300k, 350k]$
	reserved offer	random from $[450k, 500k]$
	concession strategy	random from $\{(0, 1), 1, (1, 5]\}$
	deadline	random from $[10, 20]$
providers	initial offer	random from $[500k, 550k]$
	reserved offer	random from $[370k, 420k]$
	concession strategy	random from $\{(0, 1), 1, (1, 5]\}$
	deadline	random from $[10, 20]$

6.2 Experimental Settings for Dynamic SLA Negotiations

In order to get general results of the proposed protocol's performance, the mandatory overall goal is not specified for the customer. The SLA-MNG (refer to Definition 3) indicates the expected outcome of SLA negotiations. In the experimental settings, we classify a customer's goal into two intervals, i.e., SLA-MNG $= \{[1/2, 1), 1\}$. "SLA-MNG = 1" indicates that the expected outcomes of all SLANTs are successful, and "SLA-MNG $= [1/2, 1)$" indicates that at least 50% all involved SLANTs but not all SLANTs reach successful outcomes.

In order to better test the proposed protocol in a dynamic negotiation environment, all possible positions of adding and removing negotiations are shown in Fig. 4. The static SLA negotiations (i.e., from SLA negotiation A_0 to A_3) are shown by bold circles, and modified SLA negotiations (i.e., from SLA negotiation A_4 to A_{14}) are shown by dashed circles. For simplification, the dynamism of all cases in the following three scenarios happens in same negotiation round.

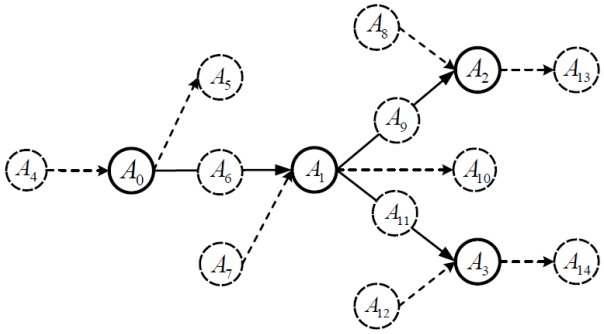

Fig. 4. Possible positions of changing negotiations

(1) ***Adding SLA negotiations***

Let a negotiation set N_{add} indicate all cases of adding SLA negotiations, where $N_{add} = \{N'_a | N'_a \subseteq N_a, N'_a \neq \varnothing\}$ and $N_a = \{A_i | i \in [4, 14]\}$.

(2) ***Removing SLA negotiations***

Let a negotiation set N_{remove} indicate all cases of removing SLA negotiations, where $N_{remove} = \{N'_b | N'_b \subseteq N_b, N'_b \neq \varnothing\}$ and $N_b = \{A_i | i \in [1, 3]\}$. There are two special cases in this scenario. The first one is removing A_0. All other SLA negotiations will be removed if removing A_0 due to the interdependencies between SLA negotiations. The second case is simultaneously removing A_1, A_2, A_3. There will be only A_0 left in this case, and it can be treated as a single SLA negotiation. Therefore, these two special cases are not considered.

(3) ***Simultaneously adding and removing SLA negotiations***

Let a negotiation set N_{mix} indicate all cases of simultaneously adding and removing SLA negotiations, where $N_{mix} = N_{add} \times N_{remove}$. Some cases in this scenario do not exist due to the experimental settings and the graph-based structure in Fig. 4. The details of these special cases are explained as follows.

If the number of removed SLA negotiations is 1, the maximum number of added SLA negotiations is 9; If the number of removed SLA negotiations is 2, the maximum number of added SLA negotiations is 7; If the number of removed SLA negotiations is 3, the maximum number of added SLA negotiations is 3.

6.3 Experimental Results

Based on experimental settings, *average percentages of achieving a customer's SLA-MNGs* and *overall utilities from SLA negotiations* are tested in three dynamic scenarios, respectively. Here, the average percentage is taken as a result since there are many cases of randomly selecting adding/removing SLA negotiations. Moreover, some special cases are conducted 100 times (i.e., static SLA negotiations, adding eleven SLA negotiations, simultaneously adding three and removing three SLA negotiations). The reason is that the selection of adding or removing SLA negotiations in each special case is unique. Black vertical lines

are introduced to indicate the deviations of utilities from SLA negotiations.

(1) ***Adding SLA negotiations***

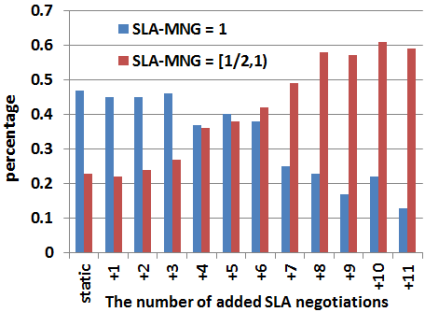

Fig. 5. Percentage of a customer's SLA-MNGs in the scenario of adding SLA negotiations

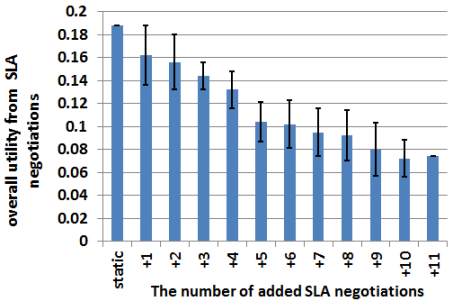

Fig. 6. Overall utility from SLA negotiations in the scenario of adding SLA negotiations

Fig. 5 shows the average percentages of achieving a customer's SLA-MNGs, where SLA-MNG $= \{[1/2, 1), 1\}$. In Fig. 5, the x-axis shows all cases of adding SLA negotiations (i.e., "+3" indicates adding three SLA negotiations), and the y-axis indicates average percentages of achieving SLA-MNGs. The average percentage of achieving "SLA-MNG $= [1/2, 1)$" goes up with adding more SLA negotiations. For "SLA-MNG $= 1$", the average percentage of achieving it goes down with adding more SLA negotiations. The reason is that adding more SLA negotiations would increase the possibility of failed SLA negotiations.

Fig. 6 shows the overall utilities from SLA negotiations in all cases of adding SLA negotiations. The x-axis indicates all cases of adding SLA negotiations, and the y-axis indicates the overall utilities from SLA negotiations. Fig. 6 shows that, with adding more SLA negotiations, overall utilities from SLA negotiations decrease. The reason is that more SLA negotiations would make it hard to get a higher overall utility from SLA negotiations. However, overall utilities from SLA negotiations tend to be relatively steady when adding more than four SLA negotiations, which indicates that the proposed protocol well handles the scenario of adding SLA negotiations.

(2) ***Removing SLA negotiations***

Fig. 7 shows average percentages of achieving SLA-MNGs in the scenario of removing SLA negotiations. The x-axis and the y-axis indicate all cases in this scenario (i.e., "-1" indicates removing one SLA negotiation) and average percentages of achieving SLA-MNGs, respectively. Fig. 7 shows that with removing more SLA negotiations, the average percentages of achieving "SLA-MNG $= 1$" and "SLA-MNG $= [1/2, 1)$" go up and down, respectively. The reason is that removing SLA negotiations would decrease the number of failed SLA negotiations, and having less failed SLA negotiations could obviously increase the

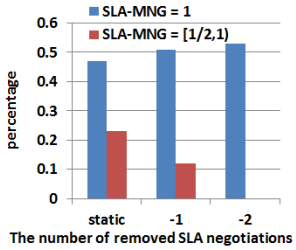

Fig. 7. Percentage of a customer's SLA-MNGs in the scenario of removing SLA negotiations

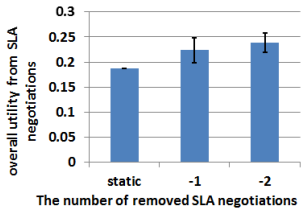

Fig. 8. Overall utility from SLA negotiations in the scenario of removing SLA negotiations

possibility of achieving "SLA-MNG = 1". The average percentage of achieving "SLA-MNG = $[1/2, 1)$" is 0 in the case of removing two SLA negotiations because there is only one SLANT left in this case.

Fig. 8 shows the overall utilities from SLA negotiations in the scenario of removing SLA negotiations. The x-axis indicates all cases of removing SLA negotiations, and the y-axis indicates the overall utility from SLA negotiations. It can be seen that the overall utilities from SLA negotiations slightly increase with removing more SLA negotiations. The reason is that it will be easier to make an agreement with fewer negotiations. The results also show that the proposed protocol works well in the scenario of removing SLA negotiations.

(3) ***Simultaneously adding and removing SLA negotiations***

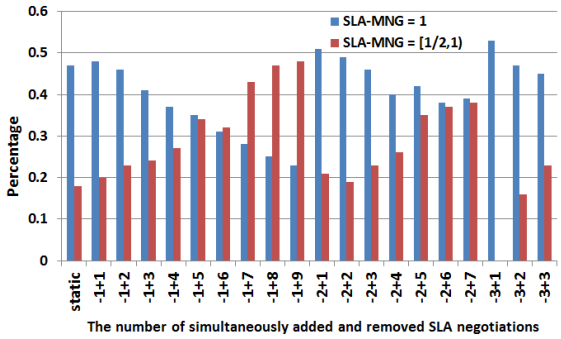

Fig. 9. Percentage of a customer's SLA-MNGs in the scenario of simultaneously adding and removing SLA negotiations

Fig. 9 shows average percentages of achieving SLA-MNGs in the scenario of simultaneously adding and removing SLA negotiations. The x-axis indicates all cases in this scenario (i.e., "-2+3" indicates simultaneously adding three and

removing two SLA negotiations), and the y-axis indicates the average percentage of achieving SLA-MNGs. From Fig. 9, we can see that if the number of removed SLA negotiations is fixed, adding more SLA negotiations can decrease the average percentage of achieving "SLA-MNG = 1" and increase the average percentage of achieving "SLA-MNG = $[1/2, 1)$". If dividing all data bars in Fig. 9 into three parts based on the number of removed SLA negotiations, it can be seen that removing more SLA negotiations can make the average percentage of achieving "SLA-MNG = 1" slightly go up and make the average percentage of achieving "SLA-MNG = $[1/2, 1)$" go down, respectively. These results have a good match with the results in both previous scenarios of adding and removing SLA negotiations. The average percentage of achieving "SLA-MNG = $[1/2, 1)$" is 0 in the special case of "-3+1" since there is only one SLANT left.

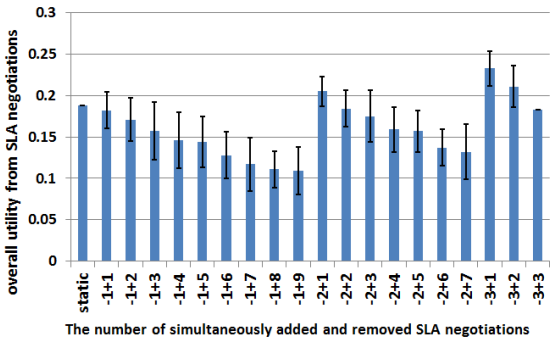

Fig. 10. Overall utility from SLA negotiations in the scenario of simultaneously adding and removing SLA negotiations

Fig. 10 shows the overall utilities from SLA negotiations in the scenario of simultaneously adding and removing SLA negotiations. The x-axis and y-axis indicate all cases in this scenario and the overall utility from SLA negotiations, respectively. From Fig. 10, we can see that if the number of removed SLA negotiations is fixed, the overall utility from SLA negotiations can decline with adding more SLA negotiations. If dividing all data bars in Fig. 10 into three parts based on the number of removed SLA negotiations, the overall utility from SLA negotiations would slightly go up with removing more SLA negotiations. These results match the results of the overall utilities from SLA negotiations in both scenarios of adding and removing SLA negotiations, and the results show that the proposed protocol well handles the dynamism when adding negotiations and removing negotiations happen simultaneously.

In summary, the experimental results show that: (1) the proposed protocol is effective while considering all possible changes in three dynamic scenarios, (2) when dynamic changes happen, the proposed protocol is able to handle the concurrency of multiple SLA negotiations as well as their interdependencies.

7 Related Work

In the real world, negotiation protocols for handling multiple SLA negotiations are indispensable. However, the achievements of protocols in handling multiple SLA negotiations while simultaneously considering concurrency, interdependency between SLA negotiations and the dynamic negotiation environment are few.

To date, achievements on SLA negotiations vary from different aspects. Dastjerdi et al. [13] focused on the SLA negotiation strategy, which is able to dynamically adapt to increase profits for cloud providers. Yaqub et al. [14] proposed a negotiation strategy for agents to efficiently create near-optimal SLAs under time constraints. Copil et al. [10] proposed an SLA negotiation protocol in order to obtain a balance between the energy consumed and performance offered in the cloud. However, the approaches above did not consider interdependency relationships between SLA negotiations. In an open cloud computing environment, interdependency relationships between SLA negotiations do exist and impact on the procedures and outcomes of SLA negotiations. The strategies and protocols in these approaches without the consideration of interdependency relationships are powerless. By contrast, interdependencies between SLA negotiations are well considered in this paper. Hence, the proposed protocol is more effective in handling multiple SLA negotiations with interdependency relationships.

Zan et al. [11] proposed a policy-based framework to support dynamic SLA negotiations for web services. Their approach focused on the bilateral negotiation, where negotiation agents are dynamically created to perform SLA negotiations. Zulkemine et al. [12] presented an SLA negotiation system for web services and proposed a negotiation broker framework to conduct bilateral SLA negotiations based on each party's requirements. These approaches concentrate on the bilateral or multilateral SLA negotiation where only one negotiation goal is involved. However, in an open and complicated cloud computing environment, a customer may have an overall goal for applying multiple cloud services, where each service corresponds to one individual goal. Therefore, the approaches which can only handle bilateral or multilateral SLA negotiation with one negotiation goal are not applicative in open cloud computing environments. However, in this paper, a protocol is proposed for concurrently handling multiple interdependent SLA negotiations in dynamic negotiation environments, where requesting new services and cancelling existing service requests under negotiated are allowed.

In summary, this paper well addresses challenging problems in the design of SLA negotiation protocols and proposes an effective negotiation protocol for handling multiple interdependent SLA negotiations in dynamic service-oriented computing environments.

8 Conclusion

Concurrently handle multiple SLA negotiations in dynamic service-oriented computing environments is a challenging research topic. In this paper, a negotiation protocol for concurrently handling multiple interdependent SLA negotiations

in dynamic environments is proposed. Experimental results show the proposed protocol is effective in concurrently handling multiple interdependent SLA negotiations and dynamic changes of multiple SLA negotiations.

Acknowledgments. This work is supported by a DECRA Project (DP140100007) from Australia Research Council (ARC) and a UPA and an IPTA scholarships from University of Wollongong, Australia.

References

1. Zheng, Z., Zhang, Y., Lyu, M., R.: Investigating QoS of Real-world Web Services. IEEE Transactions on Services Computing. 7(1), 32–39 (2014)
2. Messina, F., Pappalardo, G., Santoro, C., Rosaci, D., Sarn, G., M.: An Agent Based Negotiation Protocol for Cloud Service Level Agreements. In: 23rd IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, 161–166 (2014)
3. Faratin, P., Sierra, C., Jennings, N., R.: Negotiation Decision Functions for Autonomous Agents. Robotics and Autonomous Systems. 24(3), 159–182 (1998)
4. Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. 1, Springer (2013)
5. Niu, L., Ren, F., Zhang, M.: A Concurrent Multiple Negotiation Protocol Based on Colored Petri Nets. IEEE Transactions on Cybernetics, doi: 10.1109/TCYB.2016.2577635.
6. Jensen, K., Rozenberg, G.: High-level Petri Nets: Theory and Application. Springer (2012)
7. Jensen, K., Kristensen, L., M., Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer. 9(3-4), 213–254 (2007)
8. Fatima, S., S., Wooldridge, M., Jennings, N.,R.: Multi-issue negotiation under time constraints. In: 1st International Joint Conference on Autonomous Agents and Multiagent Systems, pp.143–150. ACM (2002)
9. Calheiros, R., N., Ranjan, R., Beloglazov, A., De Rose, C., A., F., Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience. 41(1), 23–50 (2011)
10. Copil, G., Moldovan, D., Salomie, I., Cioara, T., Anghel, I., Borza, D.: Cloud SLA Negotiation for Energy Saving − A Particle Swarm Optimization Approach. In: 8th International Conference on Intelligent Computer Communication and Processing (ICCP), 289–296 (2012)
11. Xiao, Z., Cao, D., You, C., Mei, H.: A Policy-based Framework for Automated Service Level Agreement Negotiation. In: 9th IEEE International Conference on Web Services, 682–689 (2011)
12. Zulkemine, F., H., Martin, P.: An Adaptive and Intelligent SLA Negotiation System for Web Services. IEEE Transactions on Services Computing. 4(1), 31–43 (2011)
13. Dastjerdi, A., V., Buyya, R.: An Autonomous Time-dependent SLA Negotiation Strategy for Cloud Computing. The Computer Journal. 58(11), 3202–3216 (2015)
14. Yaqub, E., Yahyapour, R., Wieder, P., Kotsokalis, C., Lu, K., and Jehangiri, A., I.: Optimal Negotiation of Service Level Agreements for Cloud-based Services Through Autonomous Agents. In: 11th IEEE International Conference on Services Computing (SCC), 59–66 (2014)