Life Cycle Automation and Management of the Distributed SLAs for Cloud Services

Waheed Aslam Ghumman and Alexander Schill

Technische Universität Dresden, Germany
{Waheed-Aslam.Ghumman, Alexander.Schill}@tu-dresden.de

Abstract. Cloud service providers mostly offer service level agreements (SLAs) in descriptive format which is not directly consumable by a machine or a system. Manual management of SLAs with growing usage of cloud services can be a challenging, erroneous and tedious task especially for the cloud service users (CSUs) acquiring multiple cloud services. The necessity of automating the complete SLA life cycle (which includes SLA description in machine readable format, negotiation, monitoring and management) becomes imminent due to complex requirements for the precise measurement of quality of service (QoS) parameters. In this work, the complete SLA life cycle management is presented using an extended SLA specification to support multiple CSU locations. A time efficient SLA negotiation technique is integrated with the extended SLA specification for concurrently negotiating with multiple cloud service providers (CSPs). After successful negotiation process, next leading task in the SLA life cycle is to monitor the cloud services for ensuring the quality of service according to the agreed SLA. A distributed monitoring approach for the cloud SLAs is elaborated, in this work, which is suitable for services being used at single or multiple locations. The discussed monitoring approach reduces the number of communications of SLA violations to a monitoring coordinator by eliminating the unnecessary communications. The presented work on the complete SLA life cycle automation is evaluated and validated with the help of experiments and simulations.

1 Introduction

The quality of a cloud service is parameterized by a service level agreement (SLA) between a cloud service user (CSU) and a cloud service provider (CSP). An agreement (SLA) between a CSU and a CSP is finalized by following different steps, i.e., definition of business objectives, transforming the business objectives to service definitions, discovering the appropriate service providers and negotiating over the quality of service (QoS) parameters. Subsequently, monitoring and management of the final SLA are important phases of the SLA life cycle. The decision of choosing between a traditional solution and acquisition of a cloud service as a better replacement is influenced by budget constraints, financial benefits, performance expectations, expeditious availability and rapid elasticity of resources. These utilities of a cloud service over the traditional solution may

diminish due to inadequacy of the automation processes in different phases of the SLA life cycle (i.e., definition, negotiation, monitoring and management). For instance, manually negotiating with multiple cloud service providers may consume significant amount of time and may compel costly delays. Cloud service providers generally offer SLAs in descriptive/natural language format which is not directly consumable by a machine/system. A cloud service user is conventionally responsible itself to monitor and enforce a natural language based SLA by first manually transforming the SLA details into a suitable machine readable format. In this paper, the complete SLA life cycle management is presented which is based on different automation components that are joined collectively. Rest of the paper is organized as given in the following. Section 5 gives an overview of the SLA life cycle and its phases. Section 2 describes a specification for the cloud SLAs as a basic structure to describe the SLAs (currently available only in text/descriptive form) into a machine readable format. Specification for the distributed SLAs (deployed on more then one locations) is also presented in Section 2 which extends the SLA specification presented in [5]. A time-efficient and automated negotiation technique for multiple providers, along with a suitable negotiation protocol, is elaborated in Section 3. Section 4 describes a distributed monitoring approach for cloud SLAs. Results and analysis of different experiments are explained in Section 6. A comparison with related work is described in Section 7. In the end, conclusions and future directions are summarized in Section 8.

2 SLA Speficiation

In this Section, a brief description of the "Structural Specification for the SLAs in Cloud Computing (S3LACC)" [5] is given and also extended (as S3LACC+) to support distributed SLAs. The S3LACC is a basic structure to define cloud SLAs in a machine readable format.

S3LACC combines the SLA template parameters, negotiation, monitoring and management parameters in a single structure. S3LACC classifies the fundamental components (SLOs and metrics) of the cloud SLAs in a hierarchical way which makes the extension of basic structure an uncomplicated task. An SLA (along with its basic descriptive elements) is composed of one or more SLOs. An SLO contains one or more metrics. A priority level can be assigned to an SLO by changing its *Weight*. A metric can be a qualitative (e.g. service reliability) or a quantitative (e.g. service availability) metric and both types of metrics contain different types of values (descriptive and numeric respectively). A metric can have a *direct* or *inverse* ratio type, i.e. if by increasing the value of a metric also increase the utility level for its user then its ratio type is *direct* otherwise *inverse*. A metric also contains one or more monitoring schedule which defines the basic monitoring parameters, e.g. the location of the monitored data that is associated with that metric. A guarantee or an obligation has similar structure, i.e. a precondition (which works as a triggering event) and an action that is performed if the precondition is true. In practical scenarios, a CSU may acquire a cloud service for using it at multiple locations simultaneously. In such cases, monitoring

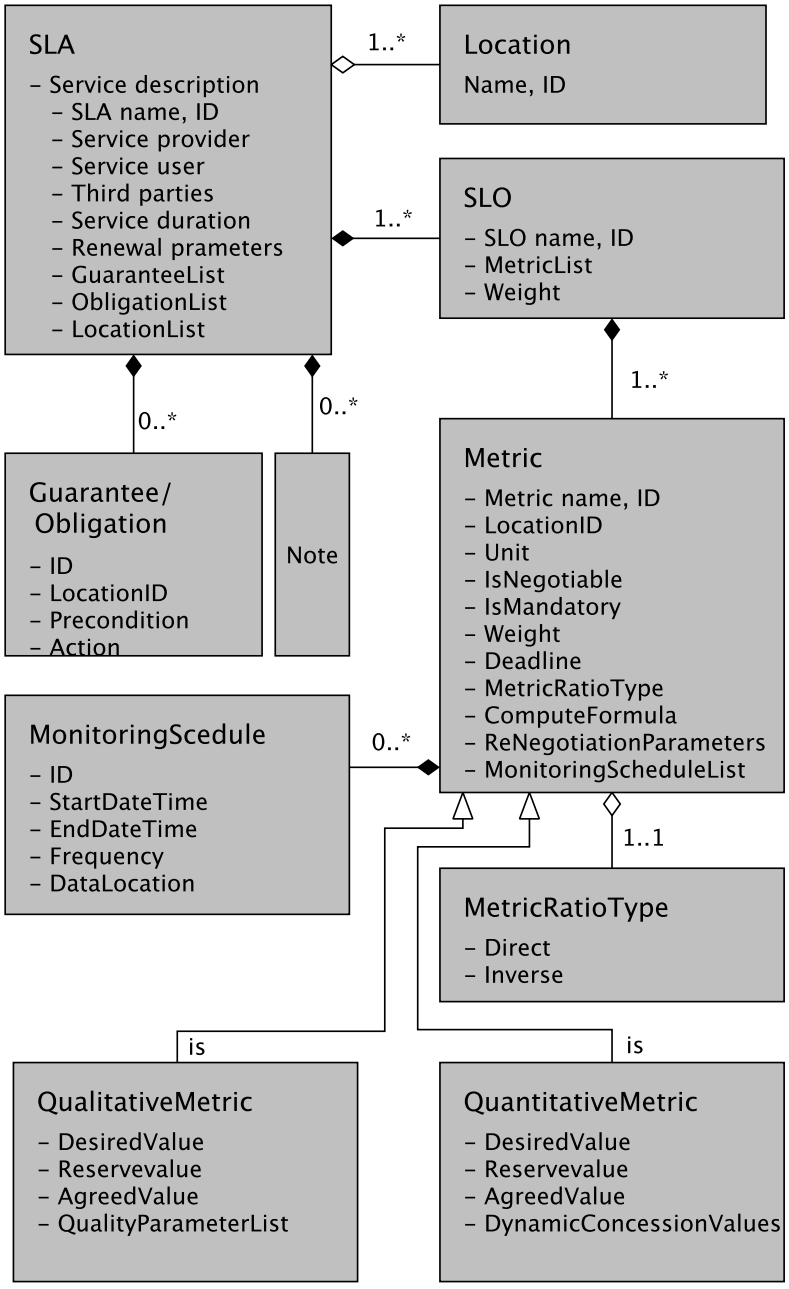

Fig. 1: UML representation of the S3LACC

the cloud service to detect SLA violations at different locations becomes a complex task, e.g. if S3LACC is used without any change then it is not possible to assign the location specific monitoring parameters in the same SLA. Moreover, one cloud service may offer different service levels for different global locations (a feature that may become part of the future cloud services, if not available in the present services), which leads to different negotiation parameters for different locations. An extension to the S3LACC is required to fulfill the needs for the distributed nature of the SLAs. So, in this paper, an extension to the S3LACC is presented which is termed as S3LACC+ and which enables the definition of the SLAs that are suitable for multiple locations. The S3LACC+ contains an additional class *Location* along with the basic S3LACC parameters as shown in Figure 1. By adding the *LocationID* to each metric, it becomes possible to have different negotiation parameters for different locations in the same SLA template. The monitoring parameters of only relevant metrics can be passed to each location, which helps to enforce privacy among different locations. A single SLA can possibly contain segregated metrics data for each location using S3LACC+. A formal and detailed description of basic SLA parameters under S3LACC is available in [5].

3 SLA Negotiation

In this Section, a dynamic and time-efficient SLA negotiation strategy (termed as flip-flop [4]) is described which uses S3LACC basically but the same negotiation strategy is applied to S3LACC+ to support multiple locations in an SLA. Negotiation protocol used in flip-flop strategy is based on Rubinstein's alternating offer protocol [17] with few alterations, i.e. negotiation process is limited by a deadline and offer acceptance is two step process (one party sends acceptance of an offer/counter offer and other party sends confirmation). The two step acceptance protocol is helpful to negotiate concurrently with multiple CSPs, i.e. if one CSP sends an acceptance then the CSU can evaluate the negotiations with other CSPs before making the final decision. The flip-flop negotiation strategy uses time duration as a deadline rather than fixed number of negotiation rounds. Cloud services generally require quick provisioning and having fixed number of negotiation rounds as a deadline may delay the negotiation time duration due to network delays. The flip-flop negotiation strategy works on a principle that opponent's expected final offer at the end of the negotiation process is predicted after each negotiation round (before preparing an offer) and the same expected final offer from the opponent is tried to be reached earlier in time using the flip-flop strategy. This strategy aims to reduce the negotiation process time which can be very advantageous in time-critical cloud services. The flip-flop negotiation strategy operates as given in the following:

- A CSU prepares its initial offers (until third negotiation round) according to the negotiation parameters as described in the SLA template using the S3LACC+. Initial offer contains the most suitable values for the CSU.

- From the fourth offer, the previous three counter offers (along with the time at which they were received) are used as input to derive a function ($\alpha_j(T_u)$ for each metric M_j at time T_u [4]) using the polynomial interpolation method, where:

$$\alpha_j(T_u) = \sum_{\substack{\iota=v \\ 0\leq v\leq q}}^{\iota+2} \left(\prod_{\substack{\kappa=v+1 \\ \kappa\neq\iota}}^{v+3} \frac{T_u - T(CO_\kappa^{b\longrightarrow a})}{T(CO_\iota^{b\longrightarrow a}) - T(CO_\kappa^{b\longrightarrow a})} \right) \zeta_{\iota,j}^{b\longrightarrow a} \tag{1}$$

  such that $T(CO_\kappa^{b\longrightarrow a})$ represents the point in time during the negotiation process when a κ-th counter offer from the CSP b is received to the CSU a and $\zeta_{\iota,j}^{b\longrightarrow a}$ is the ι-th concession that the opponent b has offered between two counter offers.
- The CSP's final offer is predicted with the help of function $\alpha_j(T_u)$ (using the polynomial extrapolation method).
- The CSU computes its concession to generate the next offer and adjusts the new offer with an extra amount of concession. The extra amount of concession is calculated using the decremented (by one) number of negotiation of rounds, i.e. if with CSU's normal concession value was reaching an agreement in n number of rounds then the extra concession value reaches the agreement in $n-1$ number of negotiation rounds. This step of increasing the CSU's concession is termed as *flip*. In formal terms, CSU's concession is set by using a partial function $\gamma_j(T_u)$ as given in the following (Equation 2):

$$\gamma_j(T_u) = \begin{cases} \dfrac{V_{j,q}^{b\longrightarrow a} - V_{j,k}^{a\longrightarrow b}}{NR_{rem}} & if\ V_{j,w} > V_{j,q}^{b\longrightarrow a} \\ \dfrac{V_{j,w} - V_{j,k}^{a\longrightarrow b}}{NR_{rem}} & if\ V_{j,w} \leq V_{j,q}^{b\longrightarrow a} \end{cases} \tag{2}$$

  where $V_{j,q}^{b\longrightarrow a}$ is the expected final offer value from the CSP b to the CSU a for the metric M_j, $V_{j,w}$ is the worst-possible/reserve value, $V_{j,k}^{a\longrightarrow b}$ is the offer value from the CSU a to the provider b with normal concession and NR_{rem} is the expected number of remaining negotiation rounds (considering the time consumed per negotiation round and the total amount of negotiation time).
- If the CSP responds with an equivalent or more percentage increase in its concession then the CSU continues the flip step. However, in case of a negative response (due to a greedy strategy by the CSP), the CSU adjusts its regular concession and decreases it by the same amount that it was increased in the previous flip step. This process of decreasing the CSU's concession is termed as *flop* step.
- This flip-flop process continues in every negotiation round until an agreement is reached or the negotiation process times-out.

In context of S3LACC+, a custom negotiation strategy (e.g. linear, conceder or Boulware) can be dynamically integrated with the existing SLA structure by updating the *DynamicconcessionValues* parameter in the *QuantitativeMetric* class.

4 SLA Monitoring and Management

After the successful negotiation process with one of the CSPs, a final/agreed SLA is stored at a central location by a monitoring coordinator (MC). The monitoring coordinator distributes the monitoring parameters to each location that intends to use the related cloud service. A distributed monitoring strategy is needed to efficiently report SLA violations at each location. The number of communications from each location to the MC is important due the fact that excessive number of communications may consume the network bandwidth and lesser number of communications may result in missing an important event at a location. So, a precise method of distributed monitoring can decrease the unnecessary communications by ensuring the fact that important events are reported to the MC. The monitoring parameters defined in an SLA using the S3LACC+ work as a basis of the SLA monitoring process. The distributed SLA monitoring strategy using partial violations [6] is combined with S3LACC+ implementation to perform the SLA monitoring for multiple locations as described in the following:

- Each SLO is assigned the minimum number of violations V_{min} that must occur at a location before reporting to the MC.
- A partial violation value v (from the interval $[0, 1]$) is assigned to each metric at design time. This partial violation value is based on the type of violation, e.g. if violation is minor then a smaller v is assigned to a that violation and vice versa.
- When a violation occurs, its corresponding v value is calculated and added to the existing violation value total S. Whenever $S \geq 1$, then one violation is added to the the existing number of violations ($V_{current}$) for the SLO and if $V_{current} \geq V_{min}$ then they are reported to the MC along with the necessary information. After reporting, counters are set to zero for the next monitoring round.

As a location reports SLA violation(s) to the MC, the guarantees and obligations parameters in the SLA are checked for the related location and if a precondition in a guarantee or obligation class is fulfilled then the corresponding action is taken. An action in a guarantee/obligation may include different SLA management tasks, e.g. a claim to be sent to the CSP for service credits along with the SLA violation data that is received from a location, sending a message to a financial system for deductions from the monthly payments to the CSP, renegotiating with the CSP depending on the $ReNegotiationParameters$ in the SLA or adjusting the service usage if an obligation (on the CSU side) requires so. A custom management task can be embedded in the *action* function of a guarantee or obligation which makes this SLA specification very useful in context of the cloud services.

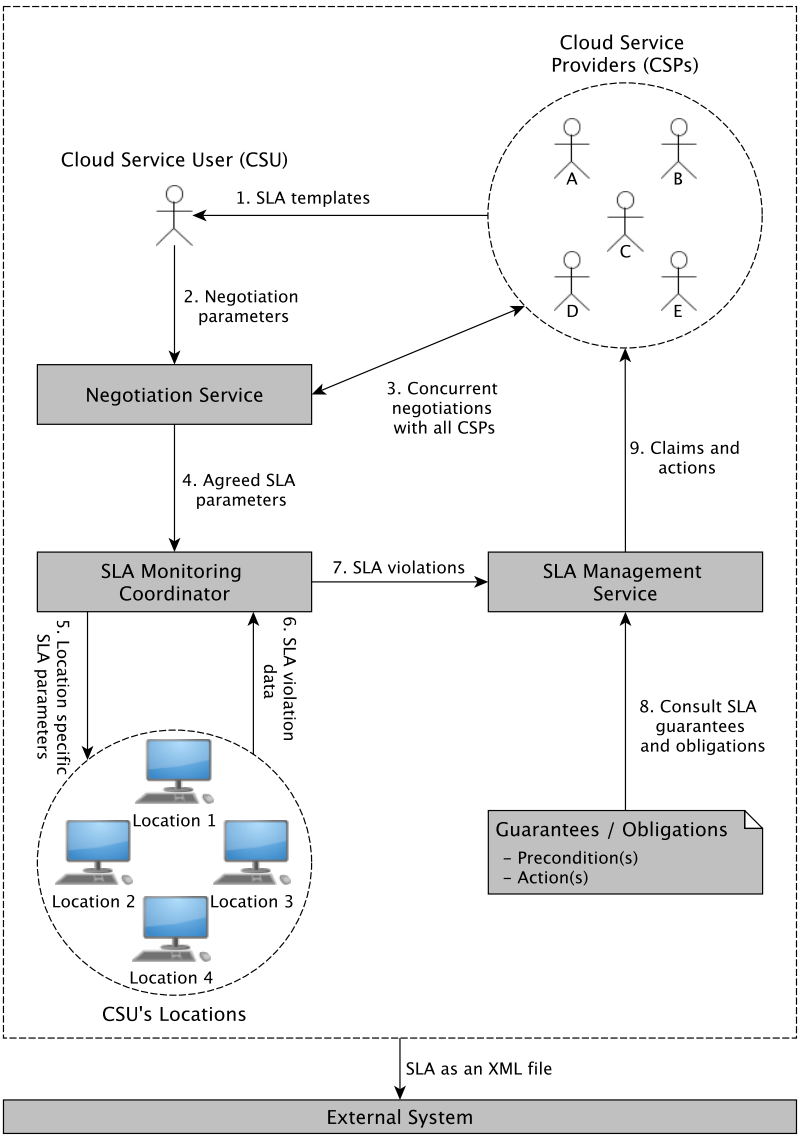

Fig. 2: An overview of the complete SLA life cycle management using S3LACC+

5 S3LACC+ Framework Overview

In this Section, a collective functioning of the different phases of the SLA life cycle is described using S3LACC+. Generally, an SLA template and a final/agreed SLA are two different documents, whereas S3LACC+ combines them in a single document and each CSU and a CSP maintain their personal copy of the SLA (which includes template parameters, negotiation, monitoring and management parameters). It is assumed that all CSPs share the same SLA structure using the S3LACC+ implementation. The complete SLA life cycle for the S3LACC+ framework is described in the following:

- A CSU requests the SLA templates from all of the CSPS. These SLA templates contain the basic information, e.g. CSP name, maximum negotiation time or names of possible negotiable SLOs and their respective metrics. Each CSP may set its own deadline (different from other CSPs) for the negotiation process in the SLA template. A CSU may also set the deadline for the negotiation process with few constraints, i.e. the CSU can not set the deadline that is greater than the minimum of all CSPs' deadlines and the minimum deadline among all the CSPs and the CSU is shared with all CSPs included in the concurrent negotiation process.
- The CSU prepares its SLA template (based on the SLA templates acquired from the CSPs) and adds the negotiation parameters according to its business objectives.
- The concurrent negotiation service starts the negotiation process with the each CSP according to the SLA negotiation described in the Section 3.
- After the successful negotiation process, with one of the CSPs, the CSU adds the monitoring parameters for each location and communicates them to the respective locations.
- Each location starts using the cloud service and marked SLA parameters are monitored according to the monitoring approach described in the Section 4.
- SLA violations are reported to the MC according to the rules defined in the monitoring parameters. The MC sends the SLA violations to the SLA management service which consults the guarantees and obligations sections of the SLAs to take the appropriate action against each SLA violation.
- As the SLA built using the S3LACC+ is based on an object oriented approach, so it can be serialized to an XML file for interaction(s) with external system(s).

The S3LACC+ framework is useful for continuous change management and for repeating the whole SLA life cycle by muting the already negotiated parameters (that require no further change) and by only marking a few metrics as negotiable based on the changes in business objectives.

6 Experiments and Validation

S3LACC+ framework is implemented in Java and multiple experiments are performed to evaluate the complete SLA life cycle. The sample experiments are

conducted with different number of SLOs and metrics including quantitative and qualitative metrics. The negotiation process is tested by comparing the result (overall agreement utility) achieved with and without using flip-flop negotiation strategy in a concurrent setup. A CSP is allowed to adopt a greedy strategy with 33% chances for all of the experiments. For instance, Table 1 shows the experiment results for the negotiation process which is concurrently completed with 10 CSPs. It can be noted that, in most of the cases, flip-flop negotiation strategy

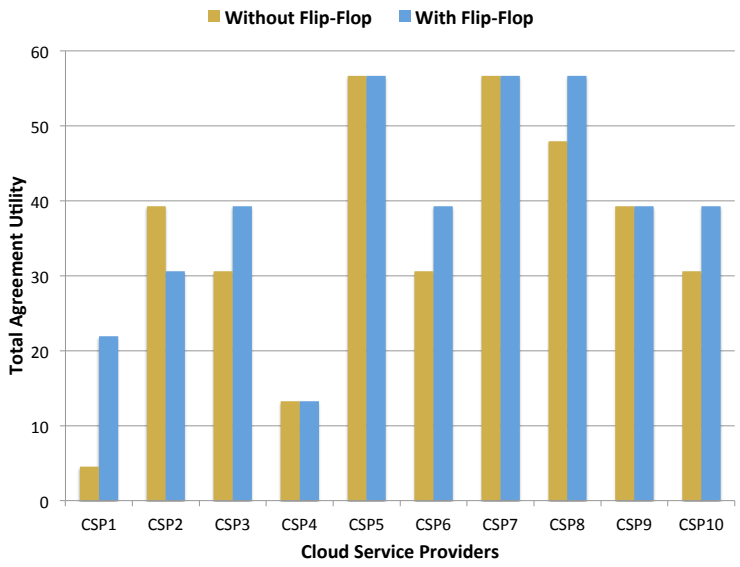

Fig. 3: Comparison of overall agreement utility achieved with and without flip-flop negotiation strategy

performs better than normal concession strategy. This automated negotiation strategy eliminates the human-intervention during the complex negotiation scenarios and modifies its concession amount depending on the response from the opponent, i.e. if opponent responds positively then this strategy seeks to reach the same agreement (that is expected at the end of negotiation process) in a lesser amount of time. If an opponent is responding with greedy approach to benefit from the increased concession (during the flip step) from the CSU, then the flop step aims to recover the loss made during the previous step by reducing the CSU's concession. A graphical representation of the experiment results of Table 1 are shown in Figure 3.

A simulation service (also used in the previous related work [6]) is implemented which induces the SLA violations for different number of SLOs to evaluate the effect on the total number of communications made to the monitoring coordina-

Table 1: Experimental results for overall agreement utility achieved with and without using the flip-flop negotiation strategy

	Agreement Utility	
	Without flip-flop	**With flip-flop**
CSP1	4.55	21.91
CSP2	39.27	30.59
CSP3	30.59	39.27
CSP4	13.23	13.23
CSP5	56.63	56.63
CSP6	30.59	39.27
CSP7	56.63	56.63
CSP8	47.95	56.63
CSP9	39.27	39.27
CSP10	30.59	39.27

tor. Table 2 shows results of one experimental simulation for 10 SLOs. Figure 4 chart gives the graphical representation of Table 2.

The monitoring simulation data for an experiment (shown in Table 2) includes

Table 2: Experiment data and results with 10 SLOs

		Number of partial violations per interval					
SLOs	Total partial violations	[0,.2[	[.2,.4[	[.4,.6[	[.6,.8[	[.8,1]	Number of communi-cations
10	20	8	0	1	7	4	0
10	40	5	9	8	7	11	2
10	60	11	15	10	7	17	4
10	80	13	17	18	15	17	8
10	100	18	16	20	26	20	7
10	120	19	22	26	25	28	12
10	140	32	33	20	28	27	12
10	160	34	33	36	27	30	17
10	180	46	32	32	39	31	14
10	200	35	37	47	37	44	18

the increasing number of induced partial violations in second column. The column number 3 to column number 7 (in Table 2) classify the number of values that fall under the interval (mentioned in the second row of respective columns). The last column (in Table 2) represents the resulted number of communications to the monitoring coordinator. The partial violation limits for each metric of an SLO is set randomly. Multiple experiments with different number of SLOs show the similar behavior in total number of communications which validates the consistency of the monitoring approach used for the S3LACC+ framework.

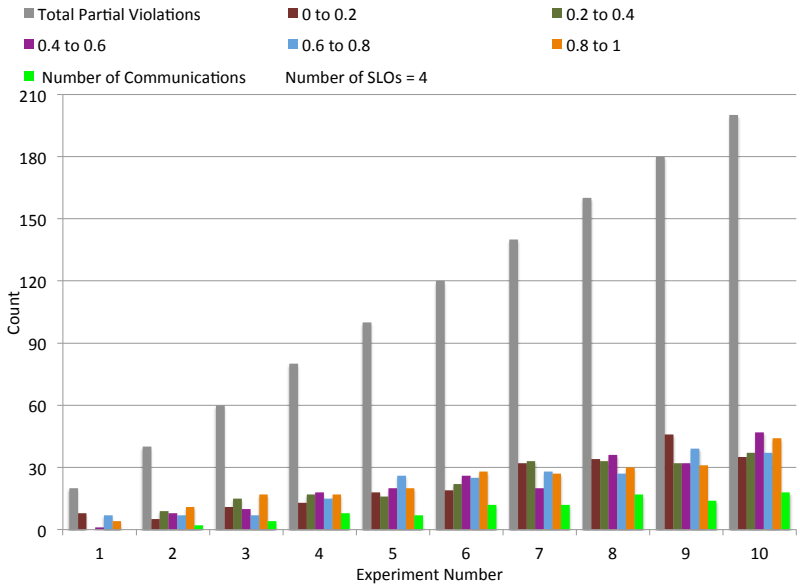

Fig. 4: Experiment results using the monitoring simulation for 10 SLOs

7 Related Work and Analysis

In this Section, a comparison of existing approaches for SLA specification, monitoring and management is described. First, an overall analysis is given that compares different SLA specification languages with S3LACC+ and its features with respect to capabilities of SLA negotiation and monitoring/management. Qualitative metrics play an important role in cloud SLAs specifically e.g. reliability is a major factor while selecting a cloud service which is a qualitative metric in general, requiring a different method of specification and negotiation than quantitative metric. Most of the specification languages compared in Table 3 contain no method for processing the qualitative metrics whereas S3LACC+ provides a comprehensive support for the qualitative metrics. Table 3 shows a brief feature based comparison of WSLA [9], WS-Agreement [1], SLAng [12], SLA* [8], SLALOM [2], Stamou *et al.* [18], Joshi *et al.* [7], CSLA [11], Kotsokalis *et al.* [10] and S3LACC+.

In second column of the Table 3, target domain (original domain for which the specification was given) is mentioned, next columns show if the SLA negotiation, monitoring and management are supported by the specification or not. The word *Partial* in the negotiation column represents that either negotiation parameters are partially definable or negotiation strategy is not integrated within the specification. S3LACC+ enables complete integration of the static and dynamic negotiation parameters. Also, S3LACC+ enables a user to include any custom negotiation strategy within the SLA template. Another feature of

Table 3: Comparative analysis of S3LACC+ framework with other approaches

Source	**Original domain**	**Negotiation**	**Monitoring/ Management**
WSLA	Web services	Yes (static)	Yes
WS-Agreement	Web services	Yes (static)	Partial
SLAng	Internet/web services	No	Only monitoring
SLA*	Domain independent	Partial	No
SLALOM	IT services	No	No
Stamou *et al.*	Cloud data services	No	No
Joshi *et al.*	Cloud services	Partial	Yes
CSLA	Cloud services	No	Yes
SLAC	Cloud services	Partial	Yes
Kotsokalis *et al.*	IT services	Yes	Yes
S3LACC	Cloud services	Yes	Yes

S3LACC+ adds the capability of merging the SLA template and the final SLA as a single document. *Partial* in monitoring/management column represents that either full SLA monitoring is not supported by the specification or a customizable SLA monitoring technique is not possible to integrate using the specification.
SLA management of cloud services includes tasks such as preparing claims in case of service violations, updating SLA parameters if requirements change or performing an action triggered due to a monitoring event. Shu *et al.* [19] present an approach for life cycle based SLA management for web services. An SLA management platform is presented in [19] to define SLAs for web services, registration of SLAs, monitoring and mapping of provider supplied parameters to service user's QoS parameters. In a most recent survey, Faniyi *et al.* [3] present an overview of SLA management for cloud services in which it is argued that cloud SLAs have still not standardized enough to be automatically deployed. It is also concluded in [3] (based on detailed analysis), majority of approaches related to SLAs have considered between one to three SLA parameters. Rak *et al.* [15] base their work for SLA monitoring on the mOSAIC API [14] (which offers development of inter-operable, portable and provider independent cloud applications). In [16], mOSAIC API is used as basis for user-centric SLA management. Maarouf *et al.* [13] present a model for the SLA life cycle management in a more recent paper where different phases of the SLA life cycle are discussed and modelled using UML (unified modeling language) diagrams. However, this work does not includes any SLA specification itself.

8 Conclusions and Future Work

In this paper, automation and management of the complete SLA life cycle is presented which extends an existing SLA specification (S3LACC) for multiple locations. S3LACC consists a core structure (with most suitable relationship among SLA elements) which is easily extensible to meet the customer specific requirements and also it can be easily modified for future changes, i.e. an extension (S3LACC+) is presented in this work to support SLAs for multiple locations based cloud services. The extended S3LACC+ specification targets the complete SLA life cycle whereas most of the specifications lack one or other critical phase of SLA life cycle. An SLA specification is not very beneficial if one of the SLA life cycle phase is not supported or complete features of the cloud service specific SLAs are not supported. The negotiation strategy used in this work (*flip-flop* negotiation) enables a CSU and a CSP to conclude the negotiation process in lesser time, hence efficient use of cloud resources is ensured which is an essence of cloud computing. The *flip-flop* negotiation strategy can be easily integrated in the SLA using S3LACC+ and a CSU can make use of this efficient negotiation strategy without making any changes to the SLA template. Similarly, the monitoring strategy used in this work enables distributed and continuous monitoring which can be joined with the S3LACC+ easily as well. The used monitoring approach decreases the number of communications made from different service locations towards the monitoring coordinator. Also, this monitoring approach helps a monitoring coordinator to define different monitoring parameters for different locations rather than a global monitoring strategy. The future directions of this work include the extension of S3LACC+ for a CSP perspective and to design a negotiation strategy that enables offline negotiations to reduce the number of round trips between a CSU and a CSP during the negotiation process. These extensions require special considerations with respect to security and privacy issues as well.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (Ws-Agreement). In: Open Grid Forum. vol. 128, p. 216 (2007)
2. Correia, A., Amaral, V., et al.: Slalom: a language for SLA specification and monitoring. arXiv preprint arXiv:1109.6740 (2011)
3. Faniyi, F., Bahsoon, R.: A systematic review of service level management in the cloud. ACM Comput. Surveys 48(3), 43:1–43:27 (Dec 2015)
4. Ghumman, W.A., Schill, A., Lässig, J.: The flip-flop SLA negotiation strategy using concession extrapolation and 3D utility function. In: IEEE 2nd International Conference on Collaboration and Internet Computing. pp. 159–168 (Nov 2016)
5. Ghumman, W.A., Schill, A.: Structural specification for the SLAs in cloud computing (S3LACC). In: 13th International Conference on the Economics of Grids, Clouds, Systems, and Services (Sep 2016)

6. Ghumman, W.A., Schill, A.: Continuous and distributed monitoring of cloud SLAs using S3LACC implementation. In: The 11th IEEE International Symposium on Service-Oriented System Engineering (Apr 2017)
7. Joshi, K.P., Pearce, C.: Automating cloud service level agreements using semantic technologies. In: Cloud Engineering (IC2E), 2015 IEEE International Conference on. pp. 416–421. IEEE (2015)
8. Kearney, K.T., Torelli, F., Kotsokalis, C.: SLA*: An abstract syntax for Service Level Agreements. In: 11th IEEE/ACM International Conference on Grid Computing. pp. 217–224 (Oct 2010)
9. Keller, A., Ludwig, H.: The WSLA framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management 11(1), 57–81 (Mar 2003)
10. Kotsokalis, C., Yahyapour, R., Rojas Gonzalez, M.A.: Modeling service level agreements with binary decision diagrams. In: Service-Oriented Computing: 7th International Joint Conference, ICSOC-ServiceWave Proceedings. pp. 190–204 (Nov 2009)
11. Kouki, Y., Ledoux, T.: Csla: a language for improving cloud sla management. In: International Conference on Cloud Computing and Services Science, CLOSER 2012. pp. 586–591 (2012)
12. Lamanna, D.D., Skene, J., Emmerich, W.: Specification language for service level agreements. EU IST 34069 (2003)
13. Maarouf, A., Marzouk, A., Haqiq, A.: Practical modeling of the sla life cycle in cloud computing. In: 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA). pp. 52–58 (Dec 2015)
14. Moscato, F., Aversa, R., Martino, B.D., Forti, T.F., Munteanu, V.: An analysis of mOSAIC ontology for cloud resources annotation. In: Federated Conference on Computer Science and Information Systems. pp. 973–980 (Sept 2011)
15. Rak, M., Venticinque, S., Máhr, T., Echevarria, G., Esnal, G.: Cloud application monitoring: The mOSAIC approach. In: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom). pp. 758–763 (Nov 2011)
16. Rak, M., Aversa, R., Venticinque, S., Di Martino, B.: User Centric Service Level Management in mOSAIC Applications, pp. 106–115 (2012)
17. Rubinstein, A.: Perfect equilibrium in a bargaining model. Econometrica 50(1), 97–109 (Jan 1982)
18. Stamou, K., Kantere, V., Morin, J.H., Georgiou, M.: A SLA graph model for data services. In: Proceedings of the Fifth International Workshop on Cloud Data Management. pp. 27–34 (Oct 2013)
19. Zhang, S., Song, M.: An architecture design of life cycle based SLA management. In: Proceedings of the 12th International Conference on Advanced Communication Technology. pp. 1351–1355 (Feb 2010)