International Journal of Computer Applications (0975 - 8887)
Volume 175 - No.26, October 2020

Service-based Architecture for SLA Management in Cloud Computing

Hend S. Salem
Department of Computer Science
Suez Canal University
Ismailia, Egypt

Rania Hodhod
TSYS School of Computer Science
Columbus State University
Columbus, USA

Ghada S. El-Tawel
Department of Computer Science
Suez Canal University
Ismailia, Egypt

Hany F. ElYamany
Department of Computer Science
Suez Canal University
Ismailia, Egypt

ABSTRACT

Service Level Agreements (SLAs) have proven their added value in the modern distributed computing era. Well-established systems for managing SLAs represent a challenging aspect in cloud computing. Certainly, proper design of such systems can save a lot of costs that are spent in paying SLA violation penalties. A managed SLA can be accomplished through several stages (i.e. life cycle): Definition and Awareness, Negotiation, Design, Implementation, and Monitoring and Compliance. In this work, a reliable service-based architecture for managing the SLA life cycle in cloud computing is proposed. The introduced architecture is composed of five main layers: Infrastructure Layer, SLA Layer, Provider Layer, Consumer Layer, and Security and Privacy Layer; each layer in turn consists of multiple components that provide certain management functionalities (e.g. violation detection and recovery). This paper focuses on the design and implementation of the SLA layer and its components, in addition to a monitoring system that observes the SLA performance, detects any violations by determining a particular calculated compliance value, and diagnoses and recovers from such violations by taking the proper actions.

Keywords

Cloud Computing; Service Level Agreement(SLA); Service Level Objectives(SLO); Quality of Service(QoS); Engineering Process; Prioritization

1. INTRODUCTION

Cloud Service Level Agreement (SLA) is a formal agreement that ensures the level of quality of the cloud service(s). Recently, SLAs have become of great significance due to their added value in the computing world. However, these SLAs have also become a controversial element because cloud computing is a sensitive technology that has been pushed in the large-scale businesses. Consequently, there is a need to study the different aspects of SLAs in order to reach a gainful agreement. An SLA can be expressed as a well-structured contract that contains detailed information about the service terms and conditions, service backing, penalties and regulations, and the service level objectives (SLO) [1]. An SLO defines one or more measurable Quality of Service (QoS) constraints [2]. Generally, there are two kinds of the SLAs [3]: the *Negotiable SLA* that takes many negotiation rounds, and the *off-the-shelf SLA* or the *non-negotiable SLA* that does not require any negotiation rounds, however, it is just a template defined by the service provider and is offered to the consumer. In this work the negotiable SLAs that require many negotiation sessions are considered.

Generally, the SLA life cycle consists of five phases [4, 5]: *Definition and Awareness*, *Negotiation*, *Design*, *Implementation*, and *Monitoring and Compliance*. Typically, these phases should be well managed to extract a vital and effective SLA. In our previous work, an engineering process for the cloud SLA life cycle was introduced [4]. The main objective of this engineering process was to overcome some of the challenges that face the construction and management of cloud SLAs. One of these challenges was the gap between SLA definition and implementation phases where the design phase is mostly not considered in the full life cycle. Accordingly, the life cycle phases from the perspective of requirements engineering was discussed and a well-defined model for the design phase was presented.

The key objectives of the work presented in this paper are threefold: (1) provide a general architecture that harmonically manages SLAs in cloud computing. This architecture considers the full life cycle of extracting the SLA which is discussed in [4]. (2) generate an automatic SLA template. (3) implement a dynamic and adaptive monitoring system. The monitoring system is responsible for ensuring the validity of the SLA, detecting and recording any violation, and taking proper actions against these violations.

The proposed architecture is composed of five interactive layers with each layer consisting of several components. The overall tasks of the architecture are summarized as follows: (1) Generate an initial SLA template that contains all possible parameters. (2) Establish automatic SLA Negotiation through a negotiation system. (3) Design the SLA with the negotiated parameters. (4) Implement and configure the final SLA. (5) Monitor the resulting SLA, detect and diagnose violations, and take actions against these violations using the monitoring system.

The rest of this paper is organized as follows: related work is discussed in Section 2. The proposed services-based architecture for SLA management is presented in Section 3. The architecture implementation and simulation results are discussed in Section 4. Lastly, Section 5 concludes the paper and shows the ongoing work.

2. RELATED WORK

Currently, there are few works done in the area of managing complete SLA life cycle. A large body of work has considered one or two phases only of the entire life cycle. Accordingly, the related work is classified into (i) SLA lifecycle and data management, (ii) SLA Negotiation, and (iii) SLA Monitoring. In terms of SLA life cycle management, the work presented in [6] discussed the automation of the SLA life cycle in cloud computing. It suggested simple solutions for SLA's description languages, SLA attributes, and SLA constraints. It focused on some generic approaches towards these solutions without showing its practical impact.

There are many researches that focus on cloud SLA negotiation and monitoring [7–10]. The research in [7] outlined a process for establishing and monitoring the SLAs in complex service based systems. The authors proposed a framework for monitoring the SLA considering the evaluation of what the SLA offers based on historical data. This work focused on the high-level metrics without considering the lowlevel metrics and the matching between them. In other words, they did not consider the full life cycle for establishing the SLA.

A framework for SLA automated negotiation is presented in [8]. This framework considered many aspects related to the negotiation process along with decision making systems and strategies. However, the authors did not mention the various SLA parameters that would be negotiated. Additionally, they did not provide a clear scenario for parameters' prioritization process.

On the other hand, the work in [9] introduced an automatic negotiation mechanism for resource allocation where buyers and sellers were presented by agents that negotiate with each other and any agent can easily de-commit from the agreement at the cost of paying penalty. However, this mechanism neither considered the prioritization of the agreement elements nor the equilibrium strategies for allocating resources dynamically. Moreover, it did not focus on the negotiation objectives.

The authors in [10] proposed an architecture called (DeSVi) for detecting SLA objectives violations through resource monitoring. The provided architecture is able to predict the violation threats in order to take actions before actual violations happen. The violation detection process is based on a predefined service level objectives and knowledge databases to manipulate and prevent such violations. This work is designated to monitor a single data center, therefore it is not suitable for monitoring cloud environments which are mostly based on several and geographically dispersed data centers.

The work in [11] introduced a structured analysis of the SLA components and data management. The authors presented an SLA digraph model for data handling and automated SLA that relies on the WSLA specification. The proposed digraph can be also used as a data model for managing the SLA information. Although this work discussed the SLA data management criteria, it did not address the different aspects related to the data management, such as prioritization of the SLA parameters and its importance in the management process. Also, it represented the SLA as a digraph, but it did not consider the negotiation and compliance phases in this digraph.

Another work [12] proposed a model for SLA management that focused on one phase of the SLA life cycle that is the Monitoring Phase only. The work done did not consider any other phases in the SLA life cycle.

3. THE PROPOSED SERVICE BASED SLA MANAGEMENT ARCHITECTURE

In order to build an automated cloud SLA management architecure, it is important to consider the different aspects that are related to cloud computing and, hence, the SLA various stages. In this paper, an architecture for managing the life cycle of creating SLAs in cloud computing is proposed. The architecture is built upon our previous work on SLA engineering model and requirements engineering [4]. This section provides a full description of the components of the proposed SLA management architecture.

3.1 Architecture Components

As depicted in Figure 1, the proposed architecture is composed of five basic layers: Infrastructure Layer, SLA Layer, Provider Layer, Consumer Layer, and Security and Privacy Layer. Each layer represents a set of services and functionalities that aid to accomplish the main task. A detailed description for each layer is provided in the following subsections.

3.1.1 Infrastructure Layer. This layer consists of three components:

—**Metrics Registry** is a data storage that records all possible cloud SLA metrics and their values (SLOs), for more detail check [4]. Indeed, a well-defined registry is a primary key success factor for an effective SLA management process. Wherefore, this registry should be fast, scalable, and easy to manage in order to be more flexible with any later customization.

—**Cloud Specifications** include data about the cloud architecture (e.g. public, private, community, governmental, or hybrid cloud computing), its characteristics, and the cloud service type (e.g. SaaS, PaaS, IaaS, or XaaS). Also, it records the cloud computing general rules and policies.

—**Service Directory** stores the required information about the providers' registered services.

3.1.2 Consumer Layer. This layer represents the consumer who requests a cloud service; can be either a regular person who uses any of the cloud services and this is in case of public cloud computing, a citizen in case of governmental cloud computing, or a specific person or organization in case of private cloud computing. Moreover, this layer describes the various interactions between the consumer and the other layers in the architecture.

3.1.3 Provider Layer. This layer represents the cloud service provider, i.e. the cloud service owner who provides the different cloud services, such as SaaS, PaaS, IaaS, or XaaS. Furthermore, it represents all the actions and reactions of the cloud provider that can be taken throughout the proposed architecture.

3.1.4 Security and Privacy Layer. This layer is responsible for securing the overall system; it provides various mechanisms for cloud security, data hiding, electronic signature to authenticate the identity, and data encryption. Many techniques are used for securing and insuring the privacy in cloud computing. For example, the work in [13] presented a metadata model to ensure the quality of security service for SOA. The authors provide different levels for describing the available variations of the authentication, authorization and privacy features. While the authors in [14] developed a generic privacy ontology in order to define the privacy preferences

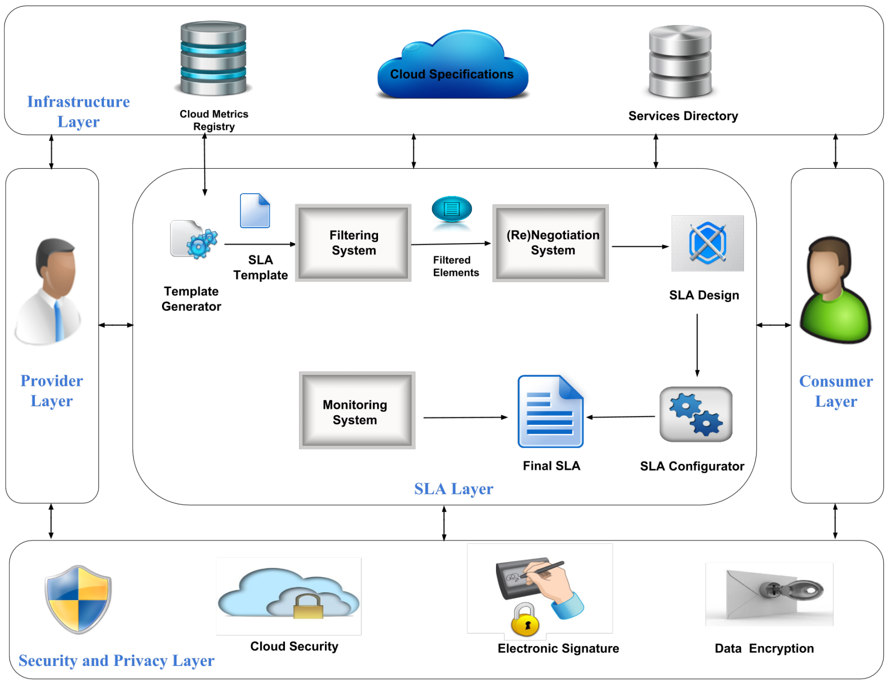


Fig. 1: SLA Management Architecture

within a Collaborative Working Environments(CWEs) focusing on the non-functional requirements of CWEs.
This layer is responsible for securing the overall system; it provides various mechanisms for cloud security, data hiding, electronic signature to authenticate the identity, and data encryption. Many techniques are used for securing and ensuring privacy in cloud computing. For example, the work in [13] presented a metadata model to ensure the quality of security service for SOA. The authors provide different levels for describing the available variations of the authentication, authorization, and privacy features. While the authors in [14] developed a generic privacy ontology in order to define the privacy preferences within collaborative working Environments (CWEs) focusing on the non-functional requirements of CWEs.

3.1.5 SLA Layer. This layer is considered to be the core layer that clarifies the proposed architecture. It consists of six interactive components that describe the full SLA life cycle. These components are: SLA Template Generator, Negotiation System, Design Service, SLA Configurator, and Monitoring System. A detailed explanation for each component is described below:

(1) **Template Generator**
This generator is accountable for the automatic generation of the initial SLA template from the SLA metrics registry. The initial SLA template contains all favorable metrics based on the cloud service(s) type. Generally, the generated template is designed to have open sections for further customization.

(2) **Filtering System**
In order to save cost and time, it is advantageous to prioritize QoS terms before negotiating them. Accordingly, a filtering system is designed in our architecture to accomplish this need. The filtering system is accountable for prioritizing the important SLA parameters to be negotiated based on the predefined negotiation objectives. When performing prioritization,

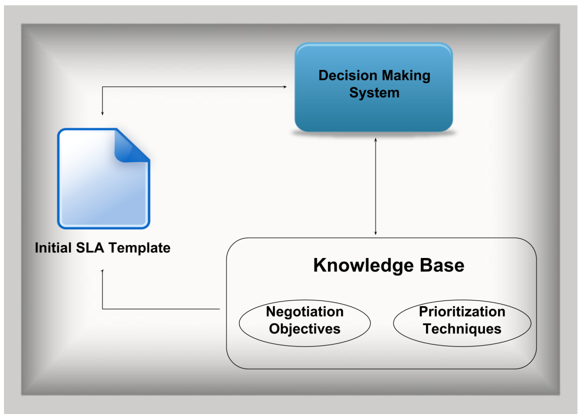


Fig. 2: Filtering System

many different aspects/attributes to prioritize the SLA parameters should be taken into account [15–17]. This can include:

—*Importance:* It is critical to prioritize which parameters are most important or critical to negotiate. Generally, importance is a relative aspect which depends on the perspective of the involved parties as well as the nature of the provided service, meaning that it has no standards.
—*Penalty:* It is important to prioritize parameters based on their penalties in case of violation, i.e. parameters that are given high penalties do have higher priority.

Many other aspects other than the above ones can be used in the prioritization process. In addition, either one aspect or a combination of aspects can be used.

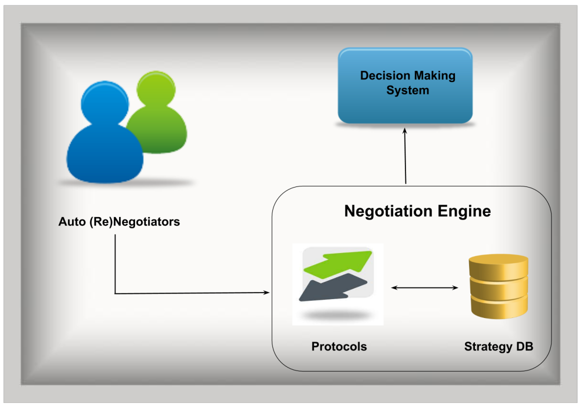


Fig. 3: (Re)Negotiation System

Mostly, a number of various techniques can be used to perform this prioritization process. Some of these techniques depend on assigning priority to each parameter, and some others consider grouping them under the priority level. Examples of the most common techniques are described as follows [17–21]:

—*Ranking:* This method gives each parameter a unique rank from 1 - n, where the most important parameter is ranked 1 and the least important parameter is ranked n.

—*Prioritization Groups (Grouping):* This technique groups the parameters into different priority groups. The number of groups is optional, however, the most common is to use three groups. For example, critical, standard, and optional or high, medium, and low.

—*Top-x Parameters:* Each party pitches its top-x parameter where x can be any defined number, such as 5, 10 or 20 based on the number of parameters they prefer to involve in the negotiation process. After then, the top-x parameters from each party are combined together and the extra similar ones are excluded. For example, assume there are involved parties P1 and P2, the top-five parameters for P1 are n1, n3, n2, n5, n10, and the top-five parameters for P2 are n7, n8, n2, n1, n6. Combining these two lists together gives us the final list of prioritized parameters: n1, n2, n3, n5, n6, n7, n8, n10.

As illustrated in Figure 2, the filtering system consists of two main components:

—Knowledge Base (KB): KB stores the negotiation objectives and the prioritization general techniques.

—Decision Making System (DMS): DMS decides on which parameters have a higher priority by matching the negotiation objectives with the initial SLA elements.

(3) **(Re)Negotiation System**

Certainly, (re)negotiation is a significant phase to create a vital SLA based on the negotiated parameters. It has many benefits for both cloud providers and consumers. In this architecture, (re)negotiation system is presented. This (re)negotiation system consists of many components which are demonstrated in Figure 3 as follows:

—**(Re)Negotiators Agents:** This includes intelligent and autonomous agents that represent the (re)negotiation participants [22]; the cloud provider, consumer, and service broker if exists.

—**(Re)Negotiation Engine:** This engine is responsible for the (re)negotiation process workflows. These workflows are underpinned by the (re)negotiation strategies and protocols. *Strategy* is the way used to make and exchange offers and counter offers by using various strategy functions [23]. *Protocol* is a set of communication rules or sequences that describe the behavior of the bargaining process. It covers how the (re)negotiation proceeds, how many rounds will it take, and the (re)negotiation states. The most common protocols are British auctions, contact Net protocol, take-it-or-leave-it, and Discrete-offer-Protocol [23,24]. Many states are defined within the (re)negotiation session including:

Propose/initiate: This state proposes an initial offer or counter offer by one of the involved parties.

Accept: This state accepts the case when the other party accords the offer or the counter offer.

Reject: This state rejects the case when the offer or the counter offer is undesirable.

Cancel/Terminate: This state cancels/terminates the case when a lack of agreement or a session timeout is occurred.

Failure: This state announces failure when any unexpected error is occurred such as a system failure.

End: This state ends the negotiation when the bargaining is normally finished.

—**Decision Making System:** This system determines the actions of the participants that can be performed during the (re)negotiation process. It decides when to accept, reject or exchange a counter offer, and how to evaluate offers and counter offers. Indeed, many decision-making techniques and heuristics are used to make the right choice. Before starting the bargaining, it is necessary to specify the cardinality of the negotiation participant: one-to-one (bilateral), one-to-many, or many-to-many. The work presented in [25] proposed a negotiation mechanism for flexible establishment of cloud SLAs. Also, the authors in [8] presented an automatic framework for negotiating SLAs in cloud computing.

(4) **SLA Design Service**

This is an embedded service responsible for designing the final SLA. It encapsulates the various activities that are used for designing the SLA template. The inputs for this service are: the negotiated SLA measurable parameters, the cloud specifications, service description, participants profiles, business policies and regulations, and conditions and penalties for exceeding expectations. The essential functionality of the design service is to portrait the overall structure of the SLA by merging the different outputs in a tidy manner. The design service also defines a user-friendly interface for users (providers or consumers) to facilitate the various interactions. Furthermore, it defines the structure of the SLA database. The corresponding outputs of the design service are the specifications of each of the SLA structure, SLA interface, and SLA database. Figure 4 depicts the ERD of the overall database. As shown in Figure 4, the database is a relational one and the schema consists of eight entities which are Service, Metric, Service_metric, Cloudspecification, Negotiation strategy, Monitoring, Violation, and Action. The Service and Metric entities are related to each other via the service_metric entity, where one service may have multiple metrics and one metric can be involved in many services. The cloud specification entity is related to the Service entity, where cloud specification should have one or more services. Monitoring and Violation entities are related to Metric entity where the monitoring and violation data is recorded and stored for a specific metric. The Action entity is related to the vio-

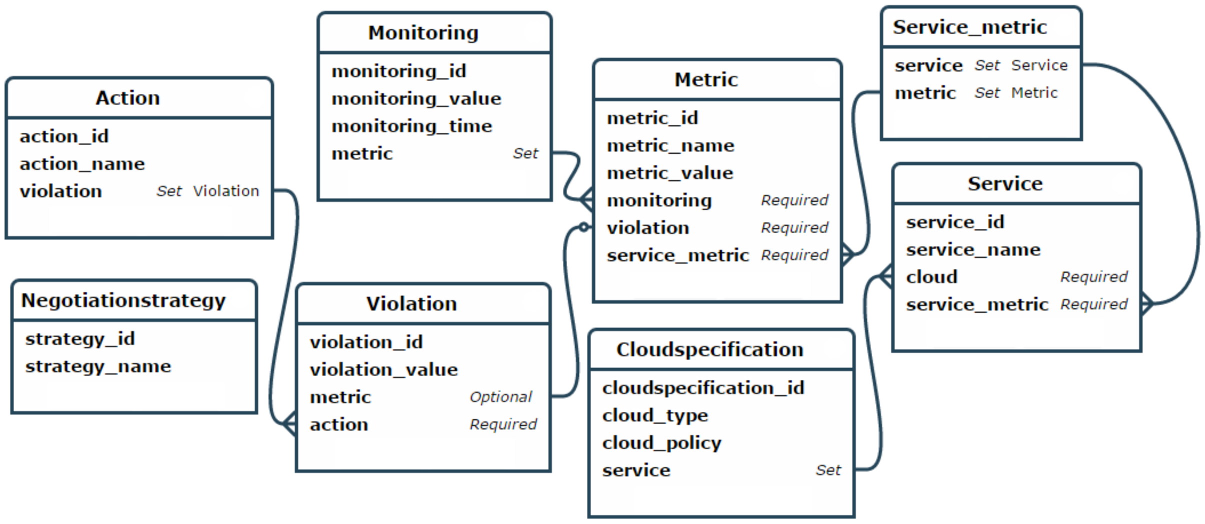


Fig. 4: SLA Database ERD

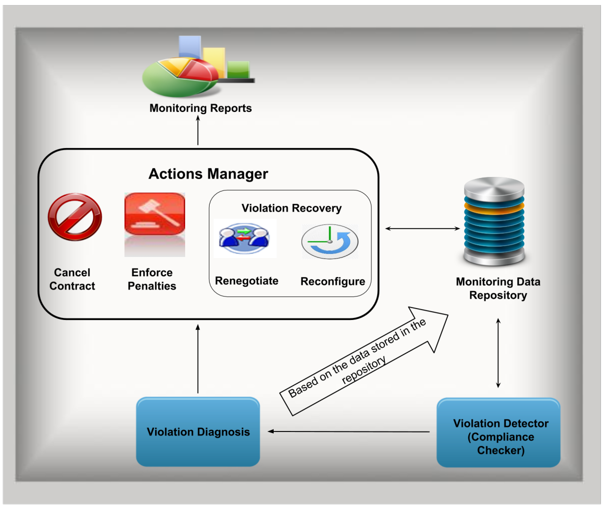


Fig. 5: Monitoring System

lation entity where each violation can be manipulated by one action while one action can be applied for multiple violations.

(5) **SLA Configurator**
The SLA configurator is accountable for configuring the final SLA template and putting it into action.

(6) **Monitoring system**
It plays a substantial role in proving the validity of the signed SLA. It is designed to be reliable and self-adaptable where it can detect violations and handle these violations by taking an appropriate action. As shown in Figure 5, the monitoring system is formed of four basic elements which are:

—**Monitoring Data Repository:** This repository stores all monitoring data samples for each QoS metric that are taken during the monitoring period, the recorded violation data, and the actions taken against these violations.

—**Violation Detector(Compliance Checker):** The checker is responsible for calculating the percentage of time that the QoS is being compliant within a predefined monitoring period (operating period). The operating period is the time in which the SLA is monitored and, hence, the compliance percentage is calculated (for instance, Sunday to Thursday form 08:00 AM - 17:00 PM). This period should be specified when defining the SLA and it can be specified on a daily, weekly, monthly basis as the contract parties decide, also, the different time zones are considered. The detection mechanism that is used to calculate the compliance percentage of the entire SLA is summarized as follows: the compliance percentage of each QoS metric is computed. Then, the resulted value is passed to the corresponding SLO to calculate its compliance percentage. After that, the compliance percentage of each SLO is passed to the SLA to calculate the total compliance percentage of the SLA.
Many calculation methods can be used to calculate the compliance value [4], such as average, median, sequential, weight, best, and worst. To calculate the compliance value, it is important to specify the operating period, the calculation method, and QoS values(thresholds). Threshold is the target value of the QoS metric that is defined in the SLA. The system begins by checking the collected sample data (stored in the monitoring repository) against the corresponding threshold to determine how many values are greater/less than this threshold. Then, a method for calculating the compliance percentage is applied. After calculating the compliance percentage of each QoS metric, the compliance percentage of the entire SLO is calculated. And finally, the compliance percentage of the total SLA is calculated from the compliance of the SLOs. For example, assume that an SLA with response time threshold of 5 msec has been created, the data samples taken for the response time parameter in the operating period specified from 08:00 AM - 18:00 PM are depicted in Table 1. As shown, there are three of ten samples that are greater than the threshold resulting in compliance percentage equals to 70% calculated using the average method.

—**Violation Diagnosis:** This component is responsible for understanding the reasons of violations. It determines which parameters are violated and what led to these violations in order to apply a recovery strategy. For example, assume that the violated parameter is the processing time, then the rea-

Table 1. : Monitoring Data samples Example

Hour	first	second	third	fourth	fifth	sixth	seventh	eighth	ninth	tenth
Samples	5 ms	3 ms	7 ms	10 ms	1 ms	2 ms	6 ms	5 ms	4 ms	2 ms

son that caused the violation might be the number of data centers is low or their configurations need to be enhanced.

—**Actions Manager:** This component is accountable for applying various actions in case of any violations based on the violation detector results. The actions manager can take any of the following three actions:

  —***Violation Recovery:*** Applying predefined recovery strategy. A number of strategies can be used for recovering SLA violations, two of these strategies are:

    —Renegotiate: This strategy renegotiates the SLA violated parameters. A new bargaining for renegotiation will start to overcome this violation by changing the old value.

    —Reconfigure: If the violation of a parameter happens due to the lack of some resources, then a reconfiguration process will start to troubleshoot this violation. Recall the example of the violated processing time. The recovery strategy here is to increase the number of data centers, or increase the specification of the hardware devices included in these data centers. This reconfiguration process can be accomplished by applying mechanisms for resource allocation. One of these mechanisms is proposed in [26]; it facilitates the resource allocation considering the workload and the geographical locations of distributed data centers. Another solution depends on resource allocation optimization by using optimization algorithms such as Ant Colony [27]. In [28], an approach for autonomic resource management was developed to master the problem of SLA violation resulting from the lack of resource management.

  —***Enforce Penalties:*** If there is no strategy for violation recovery or one of the existing recovery strategies is used but a violation has been repeated, then the stated penalties will be enforced by this action.

  —***End Contract:*** The end contract action triggered in the following cases: Firstly, the supply of the services is finished. Second, multiple and critical violations have occurred and there is no way to recover from these violations.

—**Monitoring Reports:** This action delivers reports to the contract parties that indicate the general performance of the SLA and illustrates the monitoring final results and what actions were taken.

Figure 6 illustrates the various activities and interactions between the components of the monitoring system. The monitoring system begins by initiating a monitoring instance for each SLA in a predefined operating period. Then stores the monitoring data in the data repository. After then, the stored monitoring data is checked against violation through the violation detector by computing the compliance value. When violations are detected, the violation data is stored in the repository. Then, the diagnosis unit starts diagnosing these violations and notifies the actions manager to take the right action against them. Thereafter, the actions manager is supposed to either recover these violations by setting a recovery plan and apply strategy (renegotiate or reconfigure), enforce penalties, or end the contract if these violations cannot be recovered. Then, the desired action for each violation is recorded in the data repository as well as the monitoring system got notified by these actions. Finally, the monitoring system generates monitoring reports that include the SLA monitoring results and the actions taken.

4. IMPLEMENTATION

This section discusses the implementation aspects of the proposed architecture. The goals of our evaluation are: Firstly, determine the efficiency of the proposed architecture in managing different SLAs. Second, detect SLA violations by the monitoring system. Third, show the different agreed actions that may be taken towards these violations. Section 4.1 describes the simulation environment setup and depicts the simulation results. This paper focuses on the automatic generation of the SLA and the implementation of the monitoring system.

4.1 Experimental Environment

The cloud environment was simulated using CloudSim [29] software. It is a toolkit for modeling and simulation of cloud computing environments. It supports the modeling and simulation of large scale cloud computing data centers and provides class description for virtual machines, users, applications, resources and policies [30]. Table 2 depicts our basic cloud experimental testbed. The table displays the resource specifications of the physical and virtual machines that were used.

Table 2. : Cloud Environment Resource Specifications

Machine Type	**OS**	**CPU**	**Cores**	**Memory**	**Storage**
Physical Machine	Linux	AMD	1	1 GB	30 GB
Virtual Machine	Linux	AMD	1	1 GB	10 GB

Table 3. : SLA Objectives Thresholds and the Monitoring Period

SLA parameter	**Threshold**
Data Center Processing Time	5 ms
User Base Response Time	126 ms
Operating Period (Daily)	7:00 AM - 16:00 PM

Herein, Xen virtualization technology is used, a fully open source solution for virtualization which is characterized by its high performance and high speed [31, 32]. The simulated environment has eleven user bases and five data centers geographically dispersed in different five locations. The number of virtual machines differs from one data center to another, the virtual cloud environment is created by using 13 virtual machines. For each user base, an SLA document is generated to guarantee the level of QoS for Platform-as-a-Service (PaaS) that is provided by a cloud provider (CP).

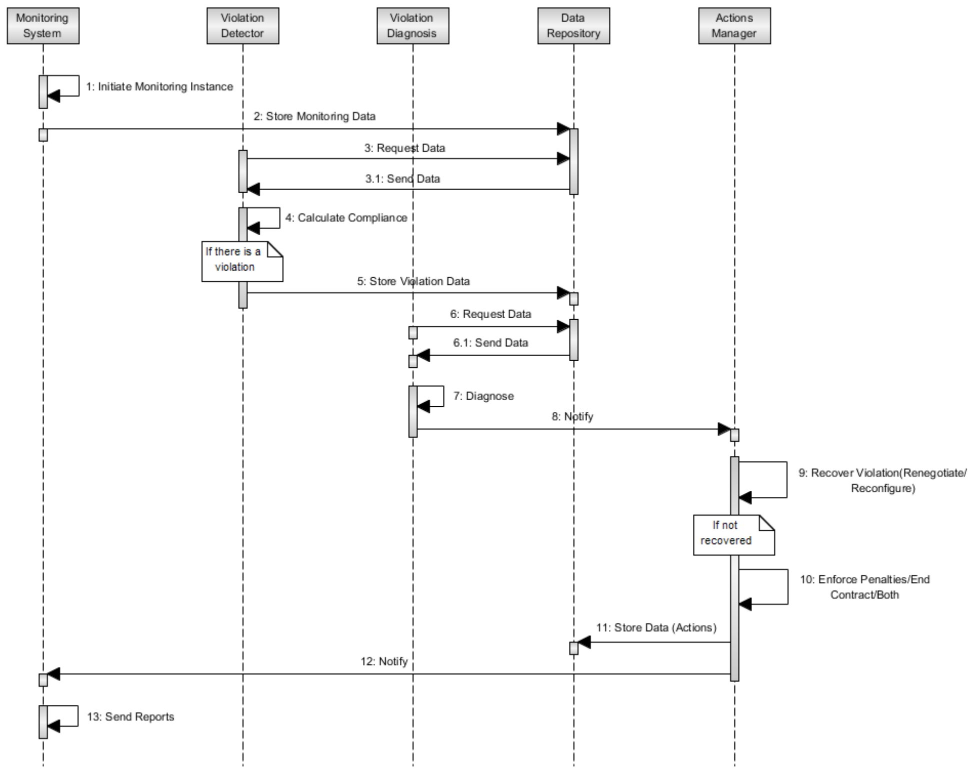


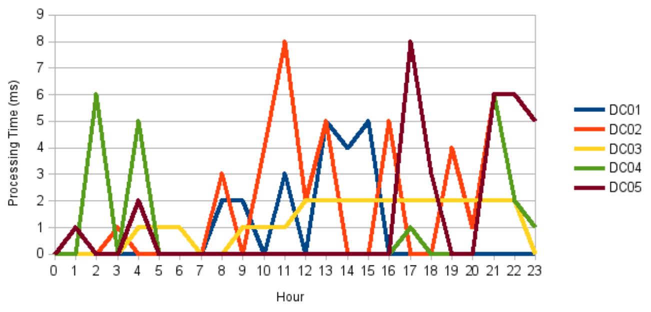
Fig. 6: Monitoring System: Components Interactions

4.2 Simulation Results Analysis

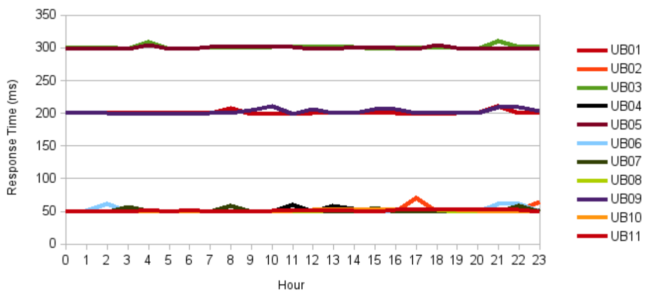
This section presents the simulation scenarios that were used to test the monitoring system. Table 3 shows the thresholds for the SLA objectives for the execution and response time parameters and the predefined operating period in which these two parameters are monitored. Based on these predefined objectives' thresholds, the SLA for each user base is monitored to detect SLA violations. Figure 7 depicts the simulation results for monitoring the SLA parameters which were created for eleven user bases and five data centers. Figure 7a shows the monitoring results of the processing time of the five data centers, and Figure 7b shows the response times monitoring results for all user bases. The average of the processing time and response time for the data centers and userbases is illustrated in Figures 7c and 7d, respectively. The red line represents the threshold value; it is obvious that no parameter exceeds the threshold values (5 ms for processing time, 126 for response time) in average.
Another simulation scenario was used and its results are demonstrated in Figure 8. On average, a violation is detected for the two parameters: processing time and response time. Figure 8a shows that two sets of processing time violations are recorded within the operating period in two different times (10:00 AM - 13:00 PM and 15:00 PM - 16:00 PM) for five data centers in average. The figure also shows sample breaches in the time period (18:00 PM - 22:00 PM), but these breaches are not considered as violations because they occurred outside the defined operating time period. User response time violations are shown in Figure 8b, these violations are recorded in the period (11:00 AM - 13:00 PM) only, and the other breaches cannot be taken into considerations because they were outside the operating time period. Accordingly, the monitoring system invoked the reconfiguration trigger to notify the provider entry of this violation. As a result, the provider increased the number of data centers in an attempt to overcome this violation. By operating the monitoring simulation again after this reconfiguration, no violations were detected as shown in Figure 9. Thus, there was no need for the provider and consumers to begin a renegotiation bargaining, and the provider avoided the cost of paying penalties.

5. CONCLUSION AND FUTURE WORK

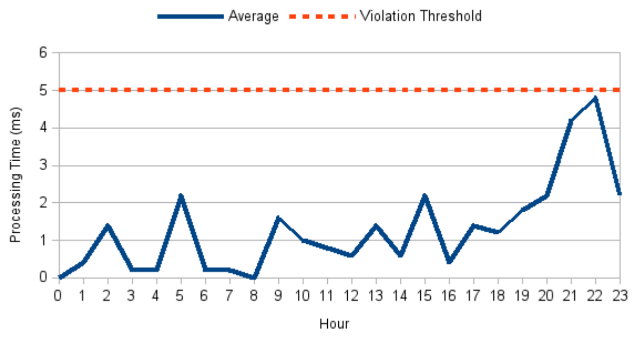
In this paper, an architecture for managing SLAs in cloud computing is proposed. The architecture is composed of five layers: Infrastructure Layer, SLA Layer, Provider Layer, Consumer Layer, and Security and Privacy Layer. Each layer encompasses several components and services for handling the diverse SLA management functionalities including SLA generation, monitoring, and evaluation and recovery. Also, the evaluation of the embedded monitoring system inside the proposed architecture is discussed. An experiment with several users was conducted through specifying and observing some highlighted SLA parameters (i.e. execution time and response time) for a provided PaaS in a cloud environment. The simulation outcomes are discussed in both violations and non-violations scenarios. The results from testing the monitoring system show that the system is able to detect and diagnose whether or not a violation has occurred and compute the compliance value. In addition, it can trigger proper actions to address violations based on the causes of these violations.
In the future, an intelligent component will be added to the monitoring system that can predict violations and provide the necessary actions to avoid and mitigate them. Moreover, this monitoring sys-



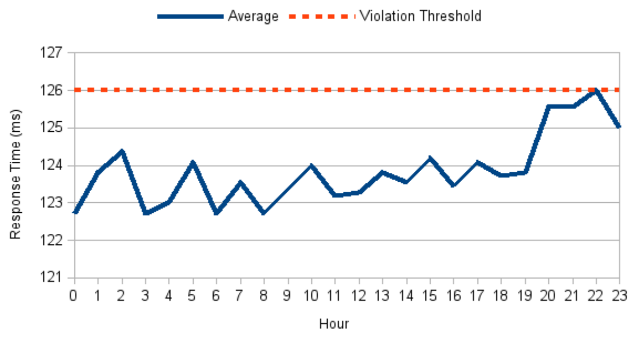
(a) Data Centers Processing Time



(b) User Bases Response Time

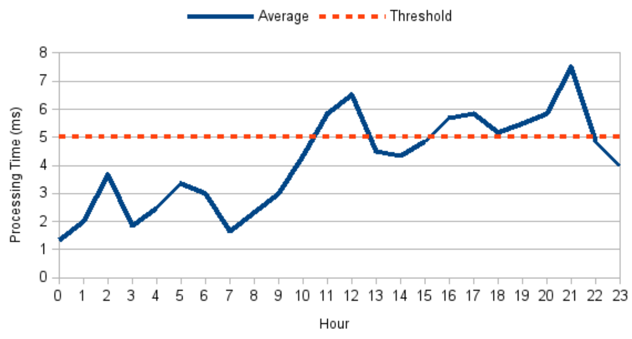


(c) Average Data Centers processing time with threshold value

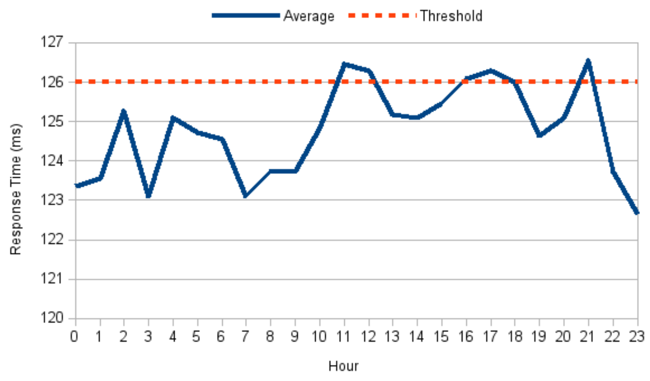


(d) Average User Bases response time with threshold value

Fig. 7: Experimentation Results



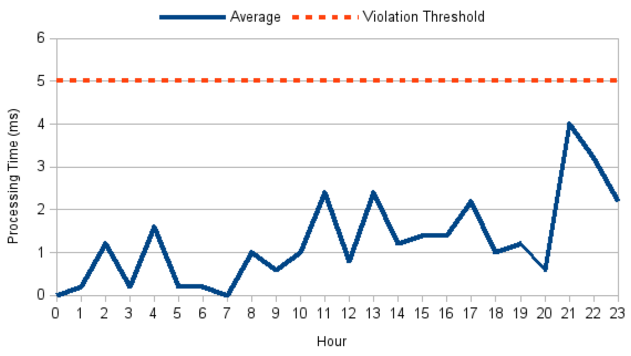
(a) Average Data Centers processing time with violation



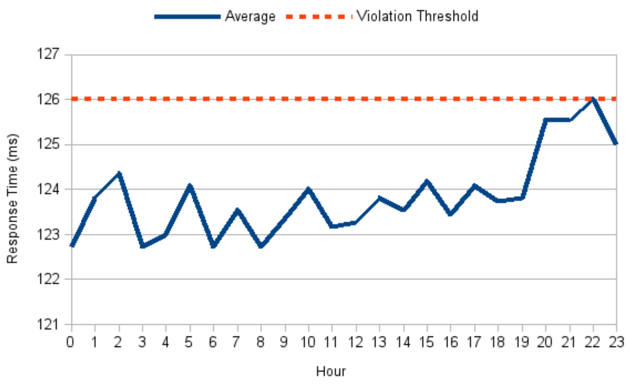
(b) Average User Bases response time with violation

Fig. 8: Objectives Violation Scenarios

tem will be extended to be more effective and efficient through the consideration of other complicated sets of parameters including the trust and security items. Furthermore, a smart prioritization system will be developed to filter the SLA parameters to be negotiated and monitored with respect to the providers and consumers' . Finally, a full solution for the SLA management as-a-service including service composition scenarios will be proposed.



(a) Average Data Centers processing time after recovery



(b) Average User Bases response time after recovery

Fig. 9: SLA Parameters after Recovery

6. REFERENCES

[1] D. Serrano, S. Bouchenak, Y. Kouki, F. A. de Oliveira Jr, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, P. Sens, Sla guarantees for cloud services, Future Generation Computer Systems 54 (2016) 233–246.

[2] W. Hussain, F. K. Hussain, M. Saberi, O. K. Hussain, E. Chang, Comparing time series with machine learning-based prediction approaches for violation management in cloud slas, Future Generation Computer Systems 89 (2018) 464–477.

[3] N. Ghosh, S. K. Ghosh, An approach to identify and monitor sla parameters for storage-as-a-service cloud delivery model, in: Globecom Workshops (GC Wkshps), 2012 IEEE, IEEE, 2012, pp. 724–729.

[4] H. S. Salem, H. F. El Yamany, G. S. El-Tawel, Towards service level agreements engineering process in cloud computing, Accepted in International Journal of Internet Manufacturing and Services (IJIMS), Inderscience Publishers, 2014.

[5] Ieee guide–adoption of iso/iec tr 24748-2:2011 systems and software engineering– life cycle management– part 2: Guide to the application of iso/iec 15288 (system life cycle processes), IEEE Std 24748-2-2012 (2012) 1–96`doi:10.1109/IEEESTD.2012.6187665`.

[6] W. A. Ghumman, Automation of the sla life cycle in cloud computing, in: Service-Oriented Computing–ICSOC 2013 Workshops, Springer, 2014, pp. 557–562.

[7] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, R. Yahyapour, Establishing and monitoring slas in complex service based systems, in: Web Services, 2009. ICWS 2009. IEEE International Conference on, IEEE, 2009, pp. 783–790.

[8] L. Wu, S. K. Garg, R. Buyya, C. Chen, S. Versteeg, Automated sla negotiation framework for cloud computing, in: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, IEEE, 2013, pp. 235–244.

[9] B. An, V. Lesser, D. Irwin, M. Zink, Automated negotiation with decommitment for dynamic resource allocation in cloud computing, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 981–988.

[10] V. C. Emeakaroha, M. A. Netto, R. N. Calheiros, I. Brandic, R. Buyya, C. A. De Rose, Towards autonomic detection of sla violations in cloud infrastructures, Future Generation Computer Systems 28 (7) (2012) 1017–1029.

[11] K. Stamou, V. Kantere, J.-H. Morin, Sla data management criteria, in: Big Data, 2013 IEEE International Conference on, IEEE, 2013, pp. 34–42.

[12] M. Kajko-Mattsson, C. Makridis, Outline of an sla management model, in: IEEE Conference on Software Reengineering and Maintenance, European Conference on Software Maintenance and Reengineering, 2008, pp. 308–310.

[13] H. F. El Yamany, M. A. Capretz, D. S. Allison, Quality of security service for web services within soa, in: Services-I, 2009 World Conference on, IEEE, 2009, pp. 653–660.

[14] D. S. Allison, A. Kamoun, M. A. Capretz, S. Tazi, K. Drira, H. F. El Yamany, An ontology driven privacy framework for collaborative working environments, Accepted in International Journal Autonomous and Adaptive Communications Systems, InderScience Publishers, 2014.

[15] IEEE Computer Society, Software Engineering Body of Knowledge (SWEBOK V3), EUA, 2014.
URL `http://www.swebok.org/`

[16] M. I. Babar, M. Ramzan, S. Ghayyur, Challenges and future trends in software requirements prioritization, in: Computer Networks and Information Technology (ICCNIT), 2011 International Conference on, IEEE, 2011, pp. 319–324.

[17] S. Baskaran, A survey on prioritization methodologies to prioritize non functional requirements, International Journal of Computer Science and Business Informatics 12 (1) (2014) 32–44.

[18] M. Ramzan, M. A. Jaffar, A. A. Shahid, Value based intelligent requirement prioritization (virp): expert driven fuzzy logic based prioritization technique, International Journal Of Innovative Computing, Information And Control (ICIC) 7 (3) (2011) 1017–1038.

[19] R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum, Prioritization of quality

requirements: State of practice in eleven companies, in: Requirements Engineering Conference (RE), 2011 19th IEEE International, IEEE, 2011, pp. 69–78.

[20] J. H. Allen, S. Barnum, R. Ellison, G. McGraw, N. Mead, Software Security Engineering, Addison-Wesley Professional, 2009.

[21] A. Herrmann, M. Daneva, Requirements prioritization based on benefit and cost prediction: an agenda for future research, in: International Requirements Engineering, 2008. RE'08. 16th IEEE, IEEE, 2008, pp. 125–134.

[22] R. L. Gomes, E. Madeira, An automatic sla negotiation protocol for a future internet, in: Communications (LATINCOM), 2011 IEEE Latin-American Conference on, IEEE, 2011, pp. 1–6.

[23] E. Yaqub, P. Wieder, C. Kotsokalis, V. Mazza, L. Pasquale, J. L. Rueda, S. G. Gómez, A. E. Chimeno, A generic platform for conducting sla negotiations, in: Service Level Agreements for Cloud Computing, Springer, 2011, pp. 187–206.

[24] P. Hasselmeyer, H. Mersch, B. Koller, H. Quyen, L. Schubert, P. Wieder, Implementing an sla negotiation framework, in: Proceedings of the eChallenges Conference (e-2007), Vol. 4, 2007, pp. 154–161.

[25] S. Son, S. C. Jun, Negotiation-based flexible sla establishment with sla-driven resource allocation in cloud computing, in: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, IEEE, 2013, pp. 168–171.

[26] S. Son, G. Jung, S. C. Jun, An sla-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider, The Journal of Supercomputing 64 (2) (2013) 606–637.

[27] R. Sahal, M. H. Khafagy, F. A. Omara, A survey on sla management for cloud computing and cloud-hosted big data analytic applications, International Journal of Database Theory and Application 9 (4) (2016) 107–118.

[28] S. S. Gill, I. Chana, M. Singh, R. Buyya, Chopper: an intelligent qos-aware autonomic resource management approach for cloud computing, Cluster Computing 21 (2) (2018) 1203–1241.

[29] Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services, accessed: 2013-02-16 (2014).
URL `http://www.cloudbus.org/cloudsim/`

[30] R. Kumar, G. Sahoo, Cloud computing simulation using cloudsim, International Journal of Engineering Trends and Technology (IJETT) 8 (2) (2014) 82–86.

[31] Xenserver: Open source virtualization, accessed: 2014-06-12 (2014).
URL `http://www.xenserver.org`

[32] The xen project, accessed: 2014-06-12 (2014).
URL `http://www.xenproject.org/`