Formulating and managing viable SLAs in Cloud Computing from a small to medium service provider's viewpoint: A state-of-the-art review

Walayat Hussain , Farookh Khadeer Hussain , Omar Hussain , Ernesto Damiani , Elizabeth Chang

PII: S0306-4379(17)30269-7
DOI: 10.1016/j.is.2017.08.007
Reference: IS 1244

To appear in: *Information Systems*

Received date: 27 April 2017
Accepted date: 11 August 2017

Highlights

- Presents a state of the art review on SLA management from service provider side in Cloud Computing.
- Presents a critical evaluation of the existing work and identifies the research gaps to be addressed for provider-side Cloud service management. A thorough literature review in the existing area is presented.
- Presents an overview of our proposed framework OPV-SLA framework to address the identified issues and validates the results on a cloud dataset.

Formulating and managing viable SLAs in Cloud Computing from a small to medium service provider's viewpoint: A state-of-the-art review

WALAYAT HUSSAIN, School of Systems, Management and Leadership, Faculty of Engineering and Information Technology, University of Technology Sydney, NSW, Australia
FAROOKH KHADEER HUSSAIN, Centre for Artificial Intelligence, School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, NSW, Australia
OMAR HUSSAIN, School of Business, University of New South Wales, Canberra, Australia
ERNESTO DAMIANI, Department of Computer Science, University of Milan, Via Bramante 65, 26013 Crema, Italy
ELIZABETH CHANG, School of Business, University of New South Wales, Canberra, Australia

Abstract

In today's competitive world, service providers need to be customer-focused and proactive in their marketing strategies to create consumer awareness of their services. Cloud computing provides an open and ubiquitous computing feature in which a large random number of consumers can interact with providers and request services. In such an environment, there is a need for intelligent and efficient methods that increase confidence in the successful achievement of business requirements. One such method is the Service Level Agreement (SLA), which is comprised of service objectives, business terms, service relations, obligations and the possible action to be taken in the case of SLA violation. Most of the emphasis in the literature has, until now, been on the formation of meaningful SLAs by service consumers, through which their requirements will be met. However, in an increasingly competitive market based on the cloud environment, service providers too need a framework that will form a viable SLA, predict possible SLA violations before they occur, and generate early warning alarms that flag a potential lack of resources. This is because when a provider and a consumer commit to an SLA, the service provider is bound to reserve the agreed amount of resources for the entire period of that agreement – whether the consumer uses them or not. It is therefore very important for cloud providers to accurately predict the likely resource usage for a particular consumer and to formulate an appropriate SLA before finalizing an agreement. This problem is more important for a small to medium cloud service provider which has limited resources that must be utilized in the best possible way to generate maximum revenue. A viable SLA in cloud computing is one that intelligently helps the service provider to determine the amount of resources to offer to a requesting consumer, and there are number of studies on SLA management in the literature. The aim of this paper is two-fold. First, it presents a comprehensive overview of existing state-of-the-art SLA management approaches in cloud computing, and their features and shortcomings in creating viable SLAs from the service provider's viewpoint. From a thorough analysis, we observe that the lack of a viable SLA management framework renders a service provider unable to make wise decisions in forming an SLA, which could lead to service violations and violation penalties. To fill this gap, our second contribution is the proposal of the Optimized Personalized Viable SLA (OPV-SLA) framework which assists a service provider to form a viable SLA and start managing SLA violation before an SLA is formed and executed. The framework also assists a service provider to make an optimal decision in service formation and allocate the appropriate amount of marginal resources. We demonstrate the applicability of our framework in forming viable SLAs through experiments. From the evaluative results, we observe that our framework helps a service provider to form viable SLAs and later to manage them to effectively minimize possible service violation and penalties.

Key Words: Cloud computing, SLA monitoring, SLA management, trusted relationship, service level objectives, viable SLA life cycle.

1. INTRODUCTION

There are a number of definitions of cloud computing. According to the National Institute of Standards and Technology [1], "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud computing is an emerging and popular new technology in parallel computing, due to the accessibility of resources to the user irrespective of their location, timing or platform [2]. According to Gartner Research, it is expected that US $677 billion will be spent on cloud computing from 2013 to 2016, including $310 billion on cloud advertising. Business cases collected by the Open group (www.opengroup.org) show that the cloud computing "per-per-use" cost model decreases investment in resource planning and reduces upfront expense. Indeed, cloud users feel liberated from the burden of managing IT resources and free of the fear of running out of them.

As cloud services are accessed remotely, however, cloud users are disappointed when a cloud-based computation does not scale as they expect [3]. Cloud service providers can be of two types. The first type is a large-scale enterprise level service provider such as Amazon or Azure, and the second is a small to medium (SME) service provider. Large-scale enterprise level service providers have abundant resources at their disposal but SME service providers do not, hence they need to manage their resources well to generate maximum revenue. Service providers manage their resources by using a Service Level Agreement (SLA), which is a contract between a service provider and a service user that defines the level of service expected from the former and the commitment of the latter. A typical SLA describes the relationship and roles of interacting parties, the agreed standards of service delivery (often called Service Level Objectives or SLOs), and the obligations and penalties imposed on violating parties [4]. In case of non-commitment to the formed expectations, SLAs describe the penalties that will be imposed on both signatories to the SLA. The seminal paper [5] describes the three means by which the analysis of non-commitment to SLAs is carried out: by an unbiased, mutually agreed third party; by trusted SLA management on the provider side; and by trusted SLA management on the consumer side. Irrespective of the approach used, most techniques for the detection of possible SLA violation initiate their detection process after the SLA has been established. For efficient SLA violation management, especially from the viewpoint of an SME service provider, we argue that the SLA management process should start at the time of SLA negotiation, not when the SLA is established. This is common practice in many business domains, such as finance, where only those service contracts (the counterparts of SLAs) which are likely to lead to a positive outcome are permitted to proceed. They are continuously monitored, even at the pre-establishment stage, and preventive actions are taken to ensure their successful outcome [6]. This concept is much rarer in IT contracts, where SLAs tend to involve limited negotiation by both parties. However, when being considered from the perspective of an SME cloud service provider, SLA management needs careful planning not only at the execution stage but also at the formation stage to protect cloud providers from:

(a) committing their limited marginal resources to service users who may not use them, as a result of which the provider will not receive a financial return. Marginal resources are those extra resources that are kept in reserve by the users and used in the case of an increase in business demands;
(b) defaulting on their obligations when many users ask to have their SLOs met at the same time.

While these problems may not affect a high scale cloud service provider such as Amazon or Azure, they have serious implications for an SME cloud service provider who has limited resources with which to generate and maximize its revenue. In this paper, we formalize this problem and present a survey of SLA management approaches in cloud computing from an SME cloud service provider's perspective, discussing the advantages and shortcomings of each approach. We then propose the Optimized Personalized Viable SLA (OPV-SLA) framework which helps SME providers to make optimal decisions in service formation and management.

The rest of the paper is organized as follows. Section 2 describes a viable SLA life cycle from the provider's perspective. Section 3 presents a classification of SLA management approaches along with their features, issues and operational methods. In Sections 4 to 6, we present our survey of SLA management approaches. Section 7 offers a critical evaluation of the existing literature and highlights the research gaps that need to be investigated further to form and manage viable SLAs from an SME provider's perspective. Section 8 presents a brief overview of our proposed OPV-SLA framework, which addresses the identified research gaps. In Section 9, we evaluate the OPV-SLA framework. Section 10 concludes the paper and discusses future directions for research.

2. SLA LIFECYCLE FOR SLA MANAGEMENT FROM THE SME CLOUD SERVICE PERSPECTIVE

The management of SLAs is an intricate process comprising many different activities that broadly form the SLA lifecycle. A basic SLA life cycle described in [7] consists of three phases, namely the creation phase, the operation phase and the removal phase. In the creation phase, the consumer subscribing to the services formally signs the contract with a provider. The provider grants access to the services and reserves resources as required. The second phase is the operation phase in which the consumer can access the read-only SLA but can also change certain parameters which may affect the charge for services. The third phase is the removal phase in which the consumer's configuration is removed following the completion of the services, and all reserved resources are released. Although the proposed SLA life cycle covers the three main functions of SLA management, it omits many factors, such as negotiation in the formation of the SLA, penalty enforcement, etc. A more thorough SLA life cycle is described in [8] which also comprises three phases: creation, operation and removal. In the creation phase, consumers first search for a suitable service provider that offers all the services they require. Consumer and provider define an SLA that contains service definitions, service objectives, SLA parameters and violation penalties. Once the SLA is agreed upon, the operation phase begins, in which real-time performance is monitored against agreed benchmarks. The SLA is terminated in the removal phase on completion of the service or in the event of violation; in the latter case, penalties are enforced. Building on this notion, [9] defines cloud SLA management as being composed of two phases, namely pre-interaction and post-interaction, as shown in Figure 1. Pre-interaction is the time phase from T-1 to T-m and includes all steps taken prior to establishing the SLA. The SLA is established at time T, when the post-interaction phase starts. The post-interaction phase from time T+1 to T+n includes all the steps taken after the SLA has been established, such as service monitoring, violation prediction and penalty enforcement for the management of the SLA.

Most studies in the literature focus on SLA management to detect possible violations in the post-interaction time phase once the SLA has been established, i.e. from time T in Figure 1. To take a proactive approach rather than a reactive one, as mentioned in Section 1, the process of SLA management should start even before the SLA is established, i.e. from time T-m. This extended timeframe enables the service provider to observe the past commitment and/or behavior of a cloud consumer and subsequently design by negotiation a viable SLA which has a high chance of success.

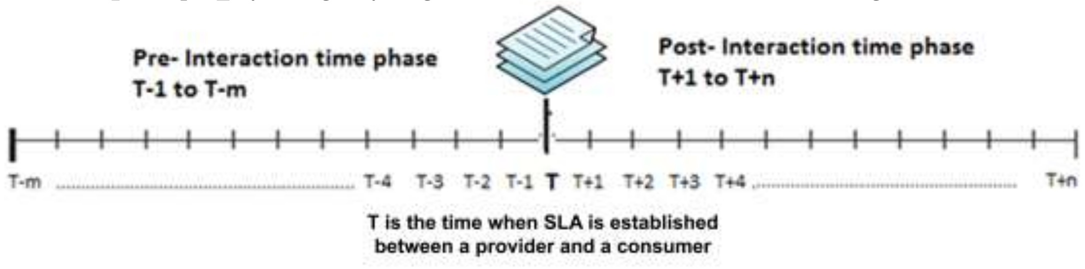

Pre- Interaction time phase
T-1 to T-m
Post- Interaction time phase
T+1 to T+n
T-m
T-4
T-3
T-2
T-1
T
T+1
T+2
T+3
T+4
T+n
T is the time when SLA is established
between a provider and a consumer

Figure 1: Pre-interaction and post-interaction time phases in SLA

A limited number of works in the literature, for example [10], have considered this problem from the provider's perspective prior to the establishment of an SLA, and have proposed a mechanism for the cloud provider to adaptively control SLA negotiation by taking the provider's resource status into consideration. Other researchers [11-16] have proposed different SLA negotiation models such as strategic SLA negotiation, automated SLA negotiation, multi-attribute negotiation, Markovian Arrival Process and Sandpiper, all of which assist cloud stakeholders to form SLAs [17]. However, none of these approaches consider the reliability of cloud users in committing to the SLAs formed during the negotiation phase, hence a stakeholder's decision can only be based on the ability of the cloud service provider to commit to the requested resources, which does not guarantee the reciprocal adherence of service users to the terms of the SLAs. As mentioned in Section 1, although these problems may not affect high scale cloud service providers, they have serious implications for SME cloud service providers who have limited marginal resources which they are required to manage properly to generate and maximize revenue. This requires an additional series of steps in the SLA lifecycle to be carried out in conjunction with those mentioned above and shown in Figure 2.

1. Resource/service request received from consumer: The consumer requests service and/or resource requirements in a formal manner. Parameters accompanying the request include type, quantity, duration and importance.

2. Determination of resource allocation criteria: When a provider receives a request from a consumer, the provider, unbeknownst to the consumer, uses intelligent algorithms to determine the trust value of the requesting consumer as well as the time for which the resources are requested. These criteria play a crucial role in determining whether partial or full resources should be allocated to the consumer during negotiation.

3. Analysis of request based on the resource allocation criteria determination: In this step, the provider compares the criteria established for the consumer against its defined threshold values.

4. Decision made by provider to accept, reject or negotiate: Depending on the determination of criteria, the provider may decide to:
   a. accept the request as is;
   b. provisionally accept the request but negotiate to formalize the amount of resources to be offered; or
   c. reject the service request, particularly if the resource allocation criteria indicate that the consumer is likely to violate the service agreement.

5. Formulation of SLA: Following the negotiation and re-negotiation steps, both parties come to a mutual agreement and an SLA is formed.

6. Threshold formation: Once the interaction with the provider and user has commenced, the service provider forms a customized threshold to warn of early possible service violations based on the agreed thresholds in the SLA,

7. Runtime Quality of Service (QoS) monitoring and QoS prediction: In this step, the QoS parameters for future intervals are predicted and monitored against the runtime QoS parameters. If there is a variation between the predicted QoS parameters and the observed QoS parameters, the risk management module is invoked to immediately take the necessary actions for SLA management.

8. Risk of SLA violation: Identifying the risk of possible SLA violation, estimating the severity of a risk, and calculating ways to mitigate such a risk.

The above-mentioned steps in the SLA lifecycle are explained in Table 1, which outlines and classifies the steps involved in each phase of SLA management from an SME provider's perspective. As large-scale cloud service providers such as Amazon and Azure have abundant resources at their disposal, the above series of steps for consumer vetting while SLA formation as mentioned in Table 1 have no relevance for them, thus they start from step five of Table 1, that is, *form SLAs*. However, this series of steps is extremely important from the viewpoint of an SME cloud service provider with limited resources in avoiding SLA violations. The term "SLA violation" refers to any failure to fulfill the service contract [17, 18]. As defined by [5], there are three types of SLA violation: "All or nothing provisioning" in which transactions are successful only when all SLOs are satisfied, "partial provisioning" in which transactions are successful when certain compulsory SLOs are satisfied, and "weighted partial provisioning" in which transactions are successful as a result of delivering those SLOs whose weight is greater than the threshold defined in the SLA.

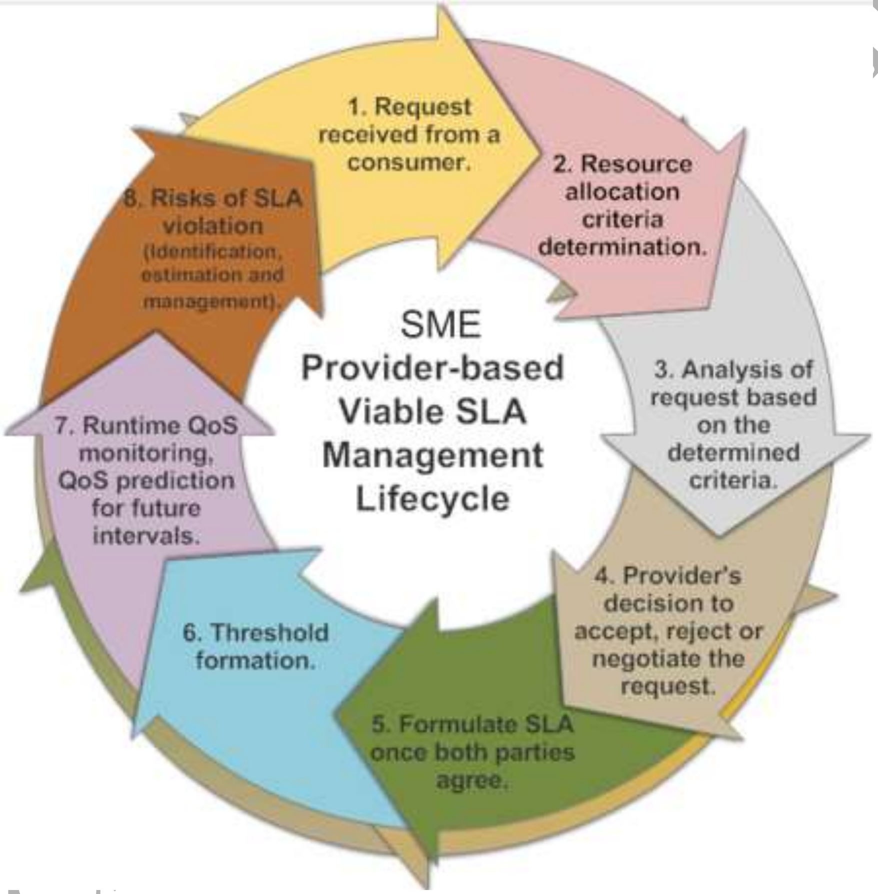

1. Request received from a consumer.
2. Resource allocation criteria determination.
3. Analysis of request based on the determined criteria.
4. Provider's decision to accept, reject or negotiate the request.
5. Formulate SLA once both parties agree.
6. Threshold formation.
7. Runtime QoS monitoring, QoS prediction for future intervals.
8. Risks of SLA violation (Identification, estimation and management).
SME Provider-based Viable SLA Management Lifecycle

Figure 2: Viable SLA life cycle from an SME provider's perspective

Table 1: Phases of SLA management from an SME provider's perspective

Pre-interaction phase	Post-interaction phase
1. Expression and classification of consumer's requirements in a formal manner: • Service/resource requirement • Duration of service/resource • Prioritization of components • Amount of resource/service required 2. Determination of resource allocation criteria by provider. 3. Provider determines to accept / reject / negotiate consumer's request. 4. Provider determines the capacity of resource/service offered to consumer. 5. Form SLAs.	6. Provider determines the threshold value for resource usage. 7. Real-time monitoring of consumer's behavior. 8. Comparison between real-time performance and expected performance. 9. Generation of an early warning to alert cloud provider to avoid possible service violation. 10. Identify, estimate and manage the risk of possible SLA violation by generating recommendation. 11. Update trust value of consumer on completion of the SLA agreement for formation of future SLAs.

The literature on SLA management proposes various approaches to detect possible SLA violations of these three types, which we discuss in the next section and critically evaluate from the perspective of SME cloud service providers.

3. CLASSIFICATION OF SLA MANAGEMENT APPROACHES TO DETECT POSSIBLE SLA VIOLATION

The management of SLAs involves many activities, of which monitoring is an essential element as it is a prerequisite for contract governance. Monitoring the difference between the agreed SLOs and the value delivered during runtime performance will lead to the detection of possible service violations. The literature presents various approaches for detecting possible SLA violations [19-22], however the subject of analysis in these approaches varies, thereby also varying their classification of SLA management analysis. For example, some approaches focus on the consumer for detecting possible SLA violation [20, 23], while others focus on the provider [22, 24]. Yet others, such as [25-27], consider the problem of SLA management as an optimization problem in which consumer satisfaction is increased by ensuring the provisioning of promised QoS and increasing the revenue of the provider. The authors in [28] categorize SLA violation management into two classes - SLA management for cloud and SLA management for cloud-hosted big data analytic applications. They mainly focus on monitoring the single layer while optimizing services by considering QoS parameters.

Our focus for SLA management in this paper is on the perspective of forming viable SLAs first and later managing them, and we therefore categorize the existing approaches according to the following three classes: *Self-manageable Case Based Reasoning (CBR) approach, Trust model-based approach,* and *Proactive SLA management approach,* as shown in Figure 3. These classifications are based on the functionality, working attributes and methodology employed to manage SLAs. After analyzing the existing approaches for SLA management from these categories, we identify the requirements for SLA management from the perspective of small and medium cloud providers.

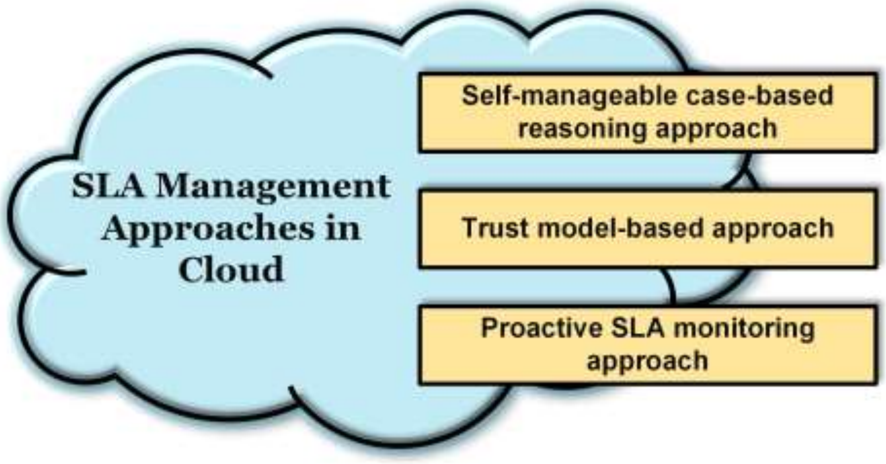

SLA Management Approaches in Cloud
Self-manageable case-based reasoning approach
Trust model-based approach
Proactive SLA monitoring approach

Figure 3: Classification of SLA management approaches in cloud computing

Below, we examine these classes in more detail:

1. Self-manageable Case Based Reasoning (CBR) approaches
   CBR approaches are techniques that use self-manageable case-based reasoning to manage SLAs when a variation between the agreed behavior and runtime behavior is detected [21, 24, 29-34]. These approaches use previous knowledge to find solutions for managing future SLA violations. Some of these approaches use a hierarchical self-healing approach [31] which detects violations and manages the SLA in a hierarchical way. The hierarchical system tries to prevent violations by reacting autonomously before notifying the end user. Approaches in this class are discussed in Section 4.
2. Trust model-based approaches
   This class incorporates techniques that use trust or reputation to manage SLAs. Reputation, or trustworthiness, is a key element in SLA management as it assists in the selection of a reliable service provider [14] [35]. The literature proposes techniques with which a consumer can score the reliability of a provider [14, 36-38] using approaches like the IP-based method [39], adaptive credibility model [40] and trust management model [36]. Approaches in this class are discussed in Section 5.
3. Proactive SLA management approaches
   These approaches to SLA management are techniques whereby the likelihood of SLA violation is predicted before violations occur, and the service provider is alerted to take all necessary actions to avoid such violations. Authors in this category use a variety of SLA monitoring approaches [15, 16, 23, 41-43] to predict likely SLO violations and to issue early warning to a cloud provider for remedial action. Approaches in this class are discussed in Section 6.

Using these classes, we present a comprehensive survey of the approaches for SLA management in relation to making viable SLAs. We discuss SLA management from the point of view of different stakeholders and present related concepts in an articulated manner that help to identify the individuality of the various approaches along with their features and shortcomings. In Sections 4-6, we discuss the techniques presented in the literature according to each type of approach for SLA management.

4. SELF-MANAGEABLE CASE-BASED REASONING APPROACH

The case-based reasoning (CBR) approach is a problem solving method in which new problems are solved based on the solution of previous similar problems [44]. This method has been widely used for decision making in a variety of dynamically changing complex and unstructured problems [45]. However, there are certain drawbacks to the CBR approach, such as adaptation, processing time and storage. The CBR approach produces reasonable solutions but it does not provide the optimal solution [46]. The CBR approach has been used in SLA management to identify the likelihood of SLA violation, and authors have used previous solutions to take early remedial action. Many techniques in the literature utilize this approach to quantify the degree of fulfillment of SLOs for SLA management. Some of these techniques are detailed below, and a summary of the various techniques is presented in Table 2.

4.1 LoM2HiS framework

Mapping low-level hardware resource metrics to high-level SLA parameters (LoM2HiS) is proposed in [24]. Mapping is either simple or complex, based on a pre-defined rule stored in a mapped metrics repository. In simple one-to-one mapping, low-level resource metrics are mapped directly to fulfill SLOs, with no further processing. In complex mapping, predefined rules are used to map resource metrics to SLOs. These rules define the thresholds for runtime SLA management and determine SLA violation. A run-time monitor accesses the repository and uses mapped metrics values to check service status. The values are compared against the corresponding thresholds; if a violation is detected, the enactor component is alerted to take preventive action. Although the system is capable of detecting SLA violation using this approach, there is no mechanism to show how an error can be rectified when a violation occurs. The authors define very few rules for converting low-level metrics to SLA parameters, and in the case of a violation it is a challenge for the system to find which low-level metrics need to be checked to address the violation.

4.2 Detecting SLA Violation infrastructure (DeSVi)

An automatic SLA violation detection infrastructure called Detecting SLA Violation infrastructure (DeSVi) is proposed in [21]. DeSVi manages and predicts SLA violations using resource management. The architecture is made up of three components: an automatic virtual machine deployer that is responsible for arranging all the required resources for a requested service and organizing its deployment on a virtual machine; an application deployer that is responsible for executing the requested resource; and the LoM2HiS framework to plot hardware-level metrics against SLA parameters. LoM2HiS performs SLA monitoring, which is comprised of the following modules: a run-time monitor, services, an agreed SLA repository, mapped metrics, a host monitor and infrastructure resources. The run-time monitoring module communicates with the consumer and the service provider. Monitoring starts when both parties agree on the SLA and the service provider establishes a mapping rule for LoM2HiS. The consumer requests services from the run-time monitoring module, which loads the corresponding SLAs from the SLA repository module. A monitoring agent is used to collect observables, compute the resource metrics and send them to the run-time monitoring module, which maps the low-level metrics and stores the results of the mapping in a repository module. The run-time monitoring module uses these mapped values to monitor service status and plot the degree of fulfillment of SLOs. In the case of SLA violation, the run-time monitoring module notifies a knowledge component to obtain an early remedy. Although LoM2HiS helps to detect a possible service violation, the system is unable to give a recommendation for its correction.

4.3 Hierarchical layered approach (LAYSI)

A hierarchical layered approach (LAYSI) to SLA management is proposed in [29]. The authors propose a bottom-up approach to the escalation of violations. There are two main components responsible for SLA violation escalation: the Knowledge Base and the SLA Manager. The knowledge base compares the violation threshold, which is generated according to a utility function, with the current system status and triggers a reactive action when it detects a violation threat. The reactive action is based on case-based reasoning and tries to solve the problem using past experiences. The system is multi-layered: when a particular layer is unable to suggest a reaction, the SLA manager is responsible for escalating SLA violation threats to the upper layers. The SLA manager receives violation notifications from the lower layer and accesses the current layer's knowledge base for an appropriate counter action. If no action is found, the upper layer is notified. Sometimes the violation escalation continues to the top layer, which informs the user by triggering the need for renegotiation or ending the service. The layered architecture assists in the better correction of errors; however further detail is needed that describes how each layer rectifies an error once a violation has occurred.

4.4 Holistic SLA validation framework

An holistic SLA validation framework is proposed in [30]. The framework combines three SLA management techniques: mapping low level resource metrics to a high level SLO (LoM2HiS) model [24], the hierarchical layer model (LAYSI) [29] and the rule-based SLA aggregation and validation model [47]. Once both parties have agreed on an SLA, the framework combines LoM2HiS with all the services that keep track of SLA violation threats. In the event of SLA violation, the framework follows the LAYSI model by trying to fix the problem at the current layer or by escalating it to an upper layer where corrective measures can be taken. When the framework detects a violation, it determines the reason and imposes a penalty on the service provider. Although the method imposes penalties on violating parties, there is no description of how the problem is rectified once it has occurred.

4.5 Cloud Application SLA Violation Detection (CASViD) framework

Cloud Application SLA Violation Detection architecture (CASViD) [31] manages SLAs at the application level. It comprises the provision of services, setting up services, monitoring services and detecting SLA violations. To detect violations, CASViD finds an effective measurement interval in which to identify the resource consumption of each application. Effective measurement is conducted by sampling time intervals and checking the applications at each interval. If the utility of the current time interval is greater than the previous interval, the current interval is set as an effective measurement interval. The process continues until the end of the interval. The monitoring agent in each node monitors the application and sends information to the SLA management module. The SLA management module accesses the database and retrieves the SLA with its violation threshold. The module then compares the current SLA with the predefined threat threshold to analyze future SLA violation threats. The threat threshold is defined manually by the provider; it indicates future SLA violations and the system reacts proactively to avoid these violations. Although the measurement interval helps in better managing an SLA, the system lacks a reaction based on previous records.

4.6 SLA management using Sky framework

Falasi et al. [32] proposed an architecture capable of managing multilevel maintenance and monitoring SLAs in a federated cloud environment. Their proposed architecture, based on the Sky framework, consists of a sky broker, a socialization module and a federation module that together adaptively implement SLAs to manage changes in a federated cloud environment. A performance evaluation report for each dependent SLA in a federated cloud is used to ensure that the primary SLA is preserved and all relevant parties are updated when changes occur. The authors describe an SLA life cycle which lacks the pre-SLA negotiation stage, monitoring steps or renegotiation after SLA violation. Moreover, the system does not alarm a service provider in the event of violation to arrange for necessary actions for early remedy.

4.7 Hierarchical self-healing of SLA (HS-SLA)

Another hierarchical self-healing SLA management framework is proposed in [33]. Each SLA is connected to the related layer of the cloud. The service provider in each layer of the cloud has one or more copies of the SLA. Each upper layer is dependent on a lower layer. Two QoS parameters, response time and throughput, are considered to measure the efficiency of this system. The SLA is monitored by the monitoring function available in each SLA. When it detects a possible violation, the system tries to resolve the issue by switching to other resources in that layer, but if the layer is unable to resolve the problem, it informs the other SLAs in the upper layer. The system tries to prevent violation before it affects end users; however, the approach is reactive in SLA management.

Furthermore, due to vendor lock-in and lack of standards in a real scenario, most cloud consumers face problems when they migrate from one service provider to another.

4.8 Fault-tolerant actor system SLA management framework

An actor system was proposed by Lu et al [48] to manage the SLA life cycle. The agent creates a parent and child relationship that helps to escalate an error message upward for its resolution. The proposed actor system is responsible for managing the entire SLA process. If a violation occurs, a concerned actor tries to rectify the error. If it is unable to solve the problem, it informs its immediate supervisor actor in the hierarchy. However, it is not mentioned how SLA management would be controlled if the single actor system responsible for the whole process were to crash. Moreover, there is no violation prediction mechanism, and in the event of service violation, the concerned layer promotes the violation report to the upper layers for remedial action.

4.9 Multi-layer monitoring

A self-adaptive SLA management mechanism is proposed in [34] which monitors SLAs on the basis of monitoring time intervals and parameters. The proposed mechanism manages both application and infrastructure layers and triggers on-the-fly reconfiguration. The management mechanism comprises six components that are arranged according to the three layers of a typical cloud stack. The Platform as a Service (PaaS) layer includes a Monitoring Framework Service (MFS) and the Monitoring Central Index (MCI). The MFS is responsible for monitoring applications and performs corrective actions in the case of violations. The MCI is a repository that stores parameter values. The Infrastructure Monitoring Service (IMS), which resides in the Infrastructure as a Service (IaaS) layer, is responsible for collecting the parameter values and sending them to the MCI. The Software as a Service (SaaS) layer consists of three components, the Monitoring Service Instance (MSI), the Monitoring Index Service (MIS) and the Data Collector (DC), which completes the process with an additional data-like name, value and unit of measure and publishes them in the local repository. Self-adaptation allows both the data collector and the infrastructure monitoring service to adjust resources or monitor time intervals. The IaaS monitoring layer is based on low-level information and related metrics; however, the authors of [36] do not describe those metrics. Self-adaptation depends on customized policies with the help of experts, but these policy functionalities are integrated with the monitoring module which does not provide flexibility or a user friendly policy enforcement mechanism. The approach does not provide scalability.

Table 2: Self-manageable case-based reasoning approach

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
LoM2HiS framework [24]	Converts low-level metrics to high-level SLA parameters and compares them with the threat threshold to predict likely service violation.	Capable of detecting future SLA violation by comparing SLA objectives with the threat threshold values. When system detects SLA violation, it informs enactor components to take early remedial action. Automatic SLA management and enforcement.	When a violation occurs, the system does not describe the error correction method. Only two rules are discussed for converting resource metrics to SLA parameters. In case of service violation, it is very difficult to state which low-level metrics need to be checked to address the violation. The criteria for a threat threshold are not defined and the system is unable to prevent violation once it has started to occur.

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
DesVi framework [21]	Resource management by LoM2HiS, which is able to monitor and detect SLA violation. Effective SLA management interval.	Flexible and reliable management of SLA. Early detection of SLA violation using a threat threshold value. Reactive action using the case-based reasoning approach.	Only capable of managing a single cloud data center. There is a lack of reactive action based on the best measurement interval. Limitations of LoM2HiS discussed above. No mechanism is described to select an optimal measurement interval. The system is unable to prevent violation once it has started to occur.
LAYSI Approach [29]	SLA management by LoM2HiS. Bottom-up approach for propagation of an SLA violation threat.	Proactive alarming. Violation threat of SLA. Self-manageable. Propagation of violation threat to layer of concern.	Does not describe how a system rectifies an error once a violation has occurred. Uses the CBR approach, which has its own limitations. Lacks description of the basis on which the threat threshold for violation detection is defined. Limitations of LoM2HiS discussed above. Approach does not describe how a system reacts when a violation occurs.
Holistic SLA validation approach [30]	Combination of LoM2HiS, LAYSI and SLA aggregation and validation framework.	Features of LoM2HiS, LAYSI. Consistency check for SLA penalty enforcement on violating party.	Limitations of LoM2HiS and LAYSI approaches discussed above. Layered bottom up approach for violation propagation, but no mechanism for management by each layer. Limitations of CBR approach. Study focuses only on basic design and lacks detail on real implementation; system is unable to manage violation once it has started to occur.
CASViD [31]	Detects SLA violation based on threat threshold.	Monitors and detects SLA violation at the application layer.	Manually defined threat threshold by a provider.

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
	Manages and monitors the SLA at the application layer.	Determines an effective measurement interval.	Lack of reactive action based on previous knowledge. SLA management focuses on post-interaction phase only.
Sky framework model [32]	Handles multilevel SLA management in a federated cloud environment.	Management of a multilevel SLA. Dynamic SLA validation and deployment.	SLA life cycle lacks pre-SLA management. No procedure defined for SLA management once violation has started to occur. Lacks negotiation and renegotiation following SLA violation.
HS-SLA [33]	Manages SLA by monitoring violations in a hierarchical way and prevents them by migrating to another provider or propagating the violation to an upper layer.	Self-healing of SLA violation, which includes SLA monitoring, SLA violation detection and all necessary actions to rectify a violation. Follows a layered hierarchical pattern for violation propagation and prevention.	System reacts once a violation occurs, but there is no mechanism to predict violation in advance. Migrates to other service providers, which itself has many issues and is not a wise suggestion for a consumer. Does not describe actions to be taken when violation occurs.
Fault-tolerant actor system [48]	Actor system automatically manages the complete SLA life cycle, following a hierarchical structure for fault-tolerant, effective and efficient SLA management.	Actor system accomplishes better SLA management. Follows a layered hierarchical structure for fault-tolerant SLA management structure and in case of service violation, propagates the violation to an upper layer for a possible solution.	No prediction of SLA violation; remedial action is performed when a violation occurs. Complete SLA system depends on a single actor system, but there is no explanation of how SLA management works if problems arise with the actor system. Authors do not describe the necessary action to be taken by the system to avoid SLA violations.
Multilayer monitoring [34]	Aggregates the QoS from the application and infrastructure layer in the platform layer.	Measures QoS parameters at infrastructure and application levels. Self-configures the monitoring time interval and monitoring	No information about low-level metrics. Policy functionalities are integrated within the monitoring module and lack a convenient policy imposition system. It is necessary to enhance the scalability performance of the

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
		parameter. Runtime adaptability of resource provisioning, estimation and decision. SLA management is done at each layer.	system using a load balancing method.

5. TRUST MODEL-BASED APPROACH FOR SLA MANAGEMENT

Establishing and maintaining trust relationships in cloud computing is a great challenge because of the large number of service providers and consumers. Trust in the provider, trust representation, and the criteria for trust calculations are only three of the issues that concern consumers. Trust has a life cycle that includes establishment, maintenance and termination. Fachrunnisa and Hussain [49] proposed a proactive performance management mechanism for trust maintenance by introducing third party agents and trust-level metric recalibrations. The third party agent is responsible for SLA administration and performance management, and compares the actual behavior with the agreed behavior defined in the SLA. Both parties recalibrate their trust to calculate the final trust level. Trust can be calculated either in monetary form or in reputation form. Reputation is based on the reliability of a service provider, and consumers score providers on each successful or unsuccessful transaction. If consumers give good feedback on every successful transaction, this will result in a high reputation value for the provider; the converse will be true for unsuccessful transactions. This scoring method allows false, biased and unreliable feedback to skew the results, which can impact the reputation of the provider [36]. Wang et al. [39] speculated that transaction validity could be verified by analyzing the IP addresses of consumers. They proposed an iteration monitoring mechanism and IP monitoring mechanism to collect and record the IP addresses of service providers and consumers and make an analysis based on the IP region, the IP record and the transaction validity. The system cancels multiple feedback from the same IP region, thus differentiating between the biased feedback and the true consumer feedback. Nevertheless, the study does not describe how the system will function if multiple consumers from the same region and the same organization provide genuine feedback. In [50], the authors propose that trust between a provider and a consumer can be maintained by managing trust from the consumer-side and the provider-side. A trusted third party monitors communication between consumers and providers; however, it cannot determine the internal state of either consumer or provider. The provider-side trust model has access to the internal state of the provider and can take measures to avoid violations. Many techniques in the literature utilize trust as an approach for SLA management, some of which are detailed below and summarized in Table 3.

5.1 Adaptive credibility model

Noor and Sheng [40] proposed an adaptive credibility model offering trust as a service which differentiates between the credible and biased feedback of consumers by using consumer capability and majority consensus. It considers unanimous feedback and measures whether a specific score is close to the majority of the feedback. The proposed framework has two components – *a credibility module* and a *distributed trust feedback assessment and storage module*. The first module is responsible for distinguishing between true and biased feedback by considering the majority consensus feedback and the second module stores the feedback assessment in a distributive way. The proposed framework is comprised of three layers that use service oriented architecture to offer trust as a service. The authors attempted to address the issue of differentiating between true and biased feedback, but the model is only feasible for a service provider that has provided services for a long time and has enough previous feedback to satisfy the majority consensus. The authors did not describe how much feedback constitutes a majority consensus, nor how to manage the situation if a group of consumers deliberately give false feedback and target a specific service provider. The proposed model is suitable for consumers who have a previous history of service usage and feedback. The feedback of new consumers who have just started using a service has lower impact, whether or not these consumers give true feedback. There is no mechanism for bootstrapping new consumers.

5.2 Reliability-based trust management model

Fan and Perros [36] proposed a trust management model that filters feedback according to two factors, familiarity and consistency. These factors are calculated from the trust feedback value of the consumer and the duration of the services used. The two factors are multiplied together to calculate the trust feedback of consumers. The proposed trust management system is divided into two sections – provider and consumer. The provider section concerns the connection of a consumer with a provider domain, and the consumer section concerns the collection of cloud service information. The framework allows consumer and provider to establish a trusted relationship for service selection and classification. The authors proposed a trust value range from 1 to 5, and only consumers whose reliability factor exceeds a pre-defined threshold, determined by service usage history over a set period, are able to assess providers. The assessment of consumers who have not used services for a long time is not reliable, although the authors did not specify a timeframe. No bootstrapping mechanism is defined for consumers who have just subscribed to services or have no previous record, and no mechanism is defined for threshold formation and consumer comparison. The authors did not justify their selection of two parameters for decision-making, and there are many unconsidered parameters that could significantly improve results.

5.3 Trust mining model

Marudhadevi et al. [51] proposed a trust mining model that calculates the degree of trust based on the subjective and objective rating of consumers. The model calculates a trust value according to such attributes as the number of successful and unsuccessful responses, average response time, and the number of complaints from consumers. The system considers the consumer feedback for each service. The model helps consumers to select trustworthy cloud services and acts as a decision system for determining whether to continue with the same provider or to switch to another provider. The model also assists the service provider to monitor the services offered, which can help to sustain a trusted association with a consumer. Rough set and Bayesian inference are used to generate the prediction results. The proposed approach calculates trust at two levels. At the first level, the system uses existing data about a provider and calculates its trust value. Once a transaction has been completed, the consumer provides feedback, based on which the second level of trust is calculated using the Bayesian inference theorem. The consumer decides whether or not to continue based on the trust value determined at levels one and two. The approach is reactive and calculates a trust value when a provider violates its commitment; moreover, the authors do not describe how to alert the service provider to cases of service degradation. There is no procedure for SLA violation and no penalty enforcement is defined.

5.4 Dynamic trust calculation method using Markov Chains

Chandrasekar et al. [37] presented an effective SLA management and QoS monitoring technique to monitor trust in a provider. A provider's service profile should be based on both its present and past services, thus it is necessary to extract QoS information from previous SLAs. The authors proposed a dynamic trust calculation method based on Markov Chains. They assigned different weights to QoS parameters based on their importance and calculated their cost by multiplying the assigned weights by the difference between the actual value and the expected value. The cost is calculated regularly, as a result of which the Markov chain may be in one of three states: steady state, unsteady state, or failure state. The trust value is computed at regular intervals. When a provider reaches maximum trust, any extra trust is banked as a surplus to be used when there is a failure to maintain the maximum trust value. The authors proposed a 50% trust value for untrusted providers, but they were unable to describe how the system should handle a provider with poor trust value who starts a business with new details. Moreover, the authors focussed only on the bandwidth required to transmit QoS by analysis through the Markov Chain model, without comparing other models.

5.5 SLA-based trust model

Alhamad, Dillon and Chang [38] proposed an SLA-based trust model to assist cloud consumers to select the most suitable cloud provider based on trustworthiness value. Their proposed model comprises the components of the SLA agent, a cloud consumer module, a cloud service directory and a cloud provider module. Each of these components performs different functions which combine to form the trust model. The authors proposed a common directory in which all cloud providers register their details, which helps consumers to find a suitable provider. They defined a set of SLA metrics for each cloud layer and used them with the trust value to calculate the suitability of the provider; however, no method for choosing the parameters was described. There are a number of parameters which, if considered, could help a consumer to choose an appropriate service provider more effectively. The study lacks a description of the process of forming an SLA agreement and negotiation, which is very important for a cloud consumer. Additionally, the paper proposes a concept without describing the criteria and methodology, and lacks a method of evaluation and implementation for calculating a trust value.

5.6 Cloud service registry and discovery model

Trust, privacy and security are three factors that hinder adaptation in cloud computing. Muchahari and Sinha [2] proposed a trust management architecture called the *cloud service registry and discovery* (CSRD) model which acts as a monitoring agent between the consumer and the provider. The framework is comprised of three modules: a registry module, trust calculating module, and dynamic trust monitoring module. The registry module registers service providers and service consumers, and lists them based on their trust values. The trust calculating module calculates the trust value of a provider by considering the feedback of credible consumers and credible cloud service providers. The feedback depends on the QoS parameters and SLA. Consumer credibility is the product of the total number of services consumed and their duration. Provider credibility is calculated based on service duration and the total number of services offered. The feedback of all consumers and providers that have a credibility value higher than the mean of the total calculated credibility value is considered to be reliable feedback, although the mean credibility value can be biased and the feedback of new consumers or providers is not considered at all. The approach calculates trust dynamically using standard deviation of duration, which is considered to be inversely proportional to trust. The proposed approach is suitable for systems that have existing records of consumers and providers, but in a real scenario, there are many factors that can influence the feedback of others for one provider. The criteria for credibility are not defined. The approach is based on a conceptual framework and has no validation or implementation.

5.7 Trust and risk assessment model

Hammadi and Hussain [35] proposed a risk and reputation assessment framework for third party cloud service providers that uses two inputs to help cloud consumers in the decision making process for the continuation or recomposition of services, namely, the trust of the provider and the risk of service level degradation. The proposed framework has three layers. A third party defines the time for QoS assessment and then divides the period into pre- and post-interaction phases. The selection of a provider is dependent on user recommendations. Credible users receive a reputation request from a third party and reply with a trust value based on the provider's previous record stored in the information repository. However, the authors do not describe how to calculate the credibility of consumers and how to deal with biased feedback and genuine consumer feedback. The authors proposed a fuzzy logic approach in which three inputs are considered: a credibility value, a time delay value, and a recommendation opinion. The third-party SLA monitoring component aggregates all the reputation values from all recommending users and calculates the final reputation value of the service provider. It also accesses the run-time SLA parameter and compares it with the threshold to identify the probability of failure. However, the literature does not describe the threshold parameters and the issues that might arise during migration to other cloud providers.

Table 3: Trust model-based approach

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
Adaptive credibility model [40]	Model offers trust as a service by considering consumer's capability and majority consensus to distinguish between true and biased feedback.	Offers trust as a service which helps service provider to distinguish between true and biased feedback. Proposed model offers distributed feedback management to avoid the hurdles of a centralized system. Feedback of an experienced consumer has higher value than feedback of other consumers.	The number of consumers that combine to form a majority is not described. If a certain number of consumers target a single provider, the system cannot handle problems. The authors consider only two factors for measuring reliability of feedback. Proposed framework is suitable for existing consumers, but there is no mechanism to bootstrap new consumers.
Reliability-based trust management model [36]	Trust management framework considers the familiarity and consistency of consumers and differentiates between true and biased feedback.	Differentiates between true consumer feedback and biased consumer feedback. System assists new consumers to select an appropriate service provider.	Timeframe for determining the recent past and distant past is not defined. No mechanism defined for bootstrapping new consumers. The comparison threshold is not defined.
Trust mining model [51]	Model assists cloud consumers to choose a reliable service provider during the negotiation phase. This approach uses Rough set and Bayesian inference to calculate trust.	Calculates trust at two levels, before an agreement is signed and during a transaction. Approach guarantees the services the consumer expects and allows the provider to monitor performance. Framework suggests whether a consumer should keep using services or switch to another provider.	Reactive SLA management approach. No mechanism to show how a provider can mitigate violation. Approach does not describe violation penalties and the procedure for their enforcement. Switching from one provider to another has many issues, such as data integrity and its compatibility.
Markov Chain model [37]	Trust in the service provider can be determined by	Effective monitoring technique using state monitoring and derived	Providers with low reputation can start business with different details and will be

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
	employing a state monitoring approach and establishing dynamic trust using the Markov Chain Model	monitoring techniques. Dynamic trust calculations based on deviation from the actual value. Trust is represented by numeric value, which is added or subtracted according to performance behavior.	able to obtain 50% trust value. Approach was tested using only Markov Chain without comparison with other methods. No method is proposed to handle situation if a system detects deviation from agreed and monitored QoS .
SLA-based trust model [38]	SLA-based trust model helps cloud consumers to select reliable cloud provider based on trustworthiness values.	Trust model helps consumer to select reliable cloud provider. Framework determines the responsible party in case of service violation and determines violation penalties. Credibility metrics define the trustworthiness of the provider.	Framework lacks the process of negotiation and SLA formation. Management of the service directory is not defined. Cannot differentiate between true and biased feedback to calculate provider's trust value.
CSRD model [2]	Framework calculates the trust value of each provider based on credible feedback from consumers and providers. It keeps track of the dynamic trust value with respect to time and transactions.	CSRD and dynamic trust overcomes security, privacy and trust problems for the adaptation of the cloud. Trust of provider and consumer is calculated and updated dynamically.	Model does not support third party providers, which are generally needed in many real-time applications. Considering provider's feedback for another provider has many issues. Approach is applicable for existing providers and consumers. Credibility criteria are not defined. Approach operates on an abstract level without implementation and evaluation.
Trust and risk assessment [35]	Selects reliable cloud providers based on their reputation value and monitors runtime performance as defined in the SLA.	Framework enables a consumer to select an appropriate cloud service provider. Framework provides real time assessment for SLA monitoring. QoS assessment in both pre-interaction and post-	No methodology defined for calculating credibility of consumer. No distinction between true and biased feedback. Parameters of the threshold are not defined. Issue of vendor lock-in when migrating to other cloud providers.

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
		interaction time phases.	

6. PROACTIVE SLA MANAGEMENT APPROACHES

A significant amount of research has been conducted on proactive and reactive SLA management in cloud computing [32, 37, 50]. In proactive SLA management, the system detects a possible SLA violation before it occurs and performs all necessary action to avoid actual violation. In reactive SLA management, the system monitors the SLA at runtime only. Predicting a possible SLA violation requires proactive SLA management to identify any discrepancies between parties and apply all necessary actions to achieve a possible remedy before the parties are affected. Proactive management approaches can be self-monitoring, self-healing, use a case-based reasoning approach, predict a violation based on QoS parameters, or use mathematical approaches. Several proactive SLA management approaches are described in the following sub-section and a summary is presented in Table 4.

6.1 SLA violation prediction by QoS prediction

An SLA, as discussed earlier, is composed of one of more than one SLO. Each SLO may consist of one or many QoS measurements. For example, throughput is one SLO defined in an SLA which is dependent on multiple components, each of which has a QoS throughput measurement. The prediction of QoS parameters plays a key role in avoiding SLA violation in the SLA management framework. A provider that predicts a difference between the agreed and actual QoS parameters takes all necessary actions to manage an SLA. Below are a number of approaches that use QoS prediction to manage SLAs in cloud.

6.1.1 QoS prediction by CloudPred

A neighborhood-based collaborative approach was proposed in [52]. The authors presented the idea of sharing local cloud component usage with all users to calculate global usage. Using QoS data from a nearest neighbor and applying both user-based and item-based collaborative filtering approaches, they were able to predict the QoS for a particular user. First, they collected QoS data using the concept of user-collaboration, in which all users send their previous web service QoS data to a central repository. Users with similar QoS data are then grouped using Pearson's Correlation Coefficient. The significance weighting of the top N users reduces the influence of less similar users. Once similar users have been identified, the QoS values are predicted, using user-based and item-based collaborative filtering methods. The approach is based on the assumption that consumers have used the same QoS parameters for the same services in the past, but in reality, this may not be so. Moreover, the study did not cover the criteria for monitoring and prediction intervals, both of which are very important in the decision-making process.

6.1.2 QoS monitoring as a service (QoS-MONaaS)

Offering QoS monitoring as a service to cloud consumers was proposed in [42, 53]. The proposed model monitors the performance of cloud providers using a stream processing framework for quick and timely responses. The framework operates on an SRT-15 platform [54] that has a two-tier architecture: a business logic tier and a data tier. The business logic tier is composed of two modules, QoS monitoring and QoS checking. The QoS monitoring module controls the monitoring process, manages the database scheme, and parses and adds a timestamp for the digital signature. The QoS checking module is responsible for executing the monitoring algorithm, which oversees all QoS parameters, and for notifying the QoS manager in the event of violation, so that the pool of QoS detectors, which uses a monitoring algorithm to parse the input with reference to defined ontologies, can be managed. The authors make no mention of how the prediction algorithm predicts the QoS parameters, and the prediction intervals are not defined. In addition, the authors do not describe the actions performed by the consumer or the provider. The approach works only if a cloud consumer and cloud provider are using the SRT-15 platform.

6.2 Workflow SLA Violation Detective Control Model (WSVDC)

The Workflow SLA Violation Detective Control Model (WSVDC) is proposed in [55]. The authors consider a utility function that measures the level of satisfaction and give control charts for each SLA. Performance is measured against four QoS parameters: response time, cost, reputation and reliability. The Western Electric rule is used to manage service behavior and detect service violation. In Statistical Process Control, the Western Electric Rules are decision rules for detecting "out-of-control" or non-random conditions on control charts. Observables that lie outside control limits (typically at ±3 standard deviations) attract the attention of the monitor to the service as they may predict future violations. The approach does not describe how the control charts and control rules are formed, nor how an optimal monitoring mechanism is guaranteed. The study only considers four SLA variables, whereas in reality there are many other attributes which cannot be ignored. Moreover, the study does not describe the reputation and reliability calculation mechanism.

6.3 RaaS-based Early Warning Framework

A Risk Assessment as a Service-based early warning framework was proposed in [23]. The framework detects future violations of SLAs based on SLO parameters or performance metrics. The authors' approach assists consumers to control deviations in performance and helps them to avoid violations before they occur. The framework comprises a number of modules. The early warning system monitors the difference between actual performance and predicts performance over a period of time. Depending on the difference between performance and the potential risk to the user, the system may suggest migrating the service. Autoregressive integrated moving average (ARIMA) and exponential smoothing methods are used to forecast the quality of cloud services. The QoS SLA violation detector determines the deviation between the QoS expected curve (QEC) and the QoS observed curve (QoC). To predict future violations, the previous QEC value is sampled into a different time interval, based on the assumption that future behavior can be predicted by observing a previous pattern of behavior. The risk propensity of all users is determined by three attitudes, i.e. risk averse, risk neutral and risk taking. Recommendations to discontinue a service are dependent on the output of the fuzzy inference system. The decision-making module checks the direction of a deviation between the observed QoS value and the expected QoS. An output value of 1 shows that the service is acceptable to the user, whereas a value of 0 shows that it is not. Although the approach covers both pre- and post-interaction phases, it lacks a mechanism for SLA negotiation and the formation of viable SLAs that guarantee the QoS parameters and appropriate actions for the avoidance and mitigation of violations. Migration to another service provider raises issues such as vendor lock-in and data compatibility.

6.4 SLA violation prevention by cross-layer adaptation

Schmieders et al. [56] proposed a cross-layer adaptation to manage SLA and prevent SLA violation of service-based applications. Service management is performed by a Service Level Agreement Monitor (SALMon) which compares the retrieved QoS with the expected QoS value in service-based applications. If a violation is detected, the SALMon sends a notification to the Specification and Assumption-based Detection (SPADE) module. This notification contains the assumption for the violation and the violating value. SPADE checks the requirements. If the requirements are not satisfied, the service-based application adapts to avoid delays. If the requirements are fulfilled, the Adaptation Strategy Engine (ASE) module is activated. Within the ASE module, each agent gathers information and negotiates the decision to adapt with the others. When the system detects a violation, a related Process Agent is activated to choose an adaptation strategy. These adaptations include service replacement, SLA re-negotiation or service infrastructure adaptation. The adaptation strategies are very limited and there is no guarantee that a suitable service replacement adaptation will be found every time.

6.5 Machine learning regression technique

Runtime SLA violation prediction, based on a regression model using existing data, was proposed in [57]. This approach predicts a likely violation before an actual violation takes place based on previous QoS data for each SLO. Prediction can be conducted at multiple checkpoints. At each checkpoint, there may be one of three types of information:

- fact data: data which are known at the checkpoint. These data are used as the input for determining unknown data.
- unknown data: data which are not known at the time of prediction. It is necessary to know all related data to achieve accurate prediction results.
- estimate data: estimate data are all those data which are not available at the time of prediction but which can be estimated. The checkpoint predictor module uses the fact data to approximate a numerical value for each SLO using a machine learning technique, such as regression. Approximation can be carried out on existing, known QoS and instance data. The prediction is then represented as a graphical user interface and the prediction manager manages the entire life cycle of prediction, i.e. its initialization, maintenance and termination. All predictions are stored in the database for future analysis.

The authors in [59] defined checkpoints for describing where the prediction should be carried out, based on an assumption which is triggered by their proposed component 'hook' and 'checkpoint predictor'. However, these checkpoints do not have a factual basis, and any approach for the prediction of SLA violation should make predictions early enough for a provider or consumer to take action to avoid actual violations. Prediction only works when data are available, and the authors used only a machine learning method for prediction and did not describe the dataset. There are other prediction methods which give optimal results. The approach did not describe the action to take if the system predicts a violation or when an actual violation occurs. Rather, the authors proposed a conceptual framework without any evaluation or implementation of their approach.

6.6 Prediction of violation by Workload Analyzer

Ciciani et al. [43] proposed the Workload Analyzer for the Cloud TM project which is able to anticipate future workload fluctuations, and hence predict SLA violations by monitoring resource data. The Workload Analyzer manages and classifies consumption data at both the infrastructure and platform layer. It combines the data from all nodes of the Cloud TM platform, then filters and correlates them. Once the data is gathered, it makes a complete workload outline of all applications, describing the current and future need for hardware and software resources. It uses various statistical functionalities to predict the future tendency of workload variations and generates a user alert to potential violations of their SLA. There is no mechanism when the violation is prediction then what remedial actions need to be taken to avoid actual violation. The proposed approach work only when a provider and a consumer are using cloud-TM project.

6.7 Resource management by heuristic policies

Cardellini et al. [58] proposed heuristic policies for Application Service Management to produce an optimal solution. The approach automatically manages resources at the application level while considering both QoS objectives and resource utilization. The authors proposed proactive and reactive heuristic policies that use a prediction algorithm based on the recursive least square algorithm to predict the workload for future time slots and evaluated their approach using only a stochastic workload model The policy is capable of detecting SLA violations but is unable to prevent them.

Table 4: Proactive SLA management approach

SLA management approach	Description of the approach	Features of the approach	Issues/limitations of the approach
CloudPred [52]	User-based and item-based collaborative filtering methods are applied to predict future QoS value and avoid service violation.	Time-aware personalized QoS prediction for different consumers. Predicts QoS value based on previous experience.	The basis for selecting monitoring and prediction intervals is not defined. The approach evaluates consumers using the same QoS parameters for the same services, however in real time this may vary.
QoS-MONaaS [52]	The framework offers monitoring as a service to all cloud consumers to monitor QoS parameters and detect service violation.	The QoS monitoring service allows consumers to monitor runtime services and predict violation in advance. The framework has a feature of complex event processing and content-based routing.	The approach only works when both provider and consumer are using the SRT-15 platform. Prediction intervals are not defined. No process is defined once the system detects QoS violation.
WSVDC [55]	The approach uses the SLA utility function and control charts to identify the difference in workflow composition and to improve the quality of cloud services and performance.	Proactively detects SLA violation based on runtime monitoring parameters. Helps an enterprise to detect faults in its system and adjusts workflow in the case of changing providers. Improves workflow reliability.	The formation of control charts and control rules is not defined. Does not guarantee an optimal monitoring mechanism. No procedure defined for reliability and reputation calculation. The study only considers four SLA variables but there may be other important variables which need to be examined. A detective model which considers multiple criteria is needed.
RaaS [23]	ARIMA and exponential smoothing are used to predict QoS, the result of which helps the consumer to decide whether to continue with the same provider or migrate to another service provider.	Generation of an early warning to alert consumer to likely service violation. The approach uses FIS by considering the risk attitude of the consumer and suggesting service continuation or migration.	Migration from one provider to another provider raises such issues as vendor lock-in and data compatibility. Approach lacks a methodology for suggesting appropriate actions once consumer detects violation. The pre-interaction phase lacks a negotiation process for QoS parameters and the formation of a viable SLA.
Cross-layer adaptation [56]	Discusses the proposed assumption in the context of service-based	Cross layer adoption and prevention of SLA violation. Both consumer and	There is no guarantee a suitable service replacement will be found. The adaptation strategies are very

	applications to check whether real-time data correlates with their proposed assumption. If there is a difference between the agreed and predicted SLA, the approach chooses an appropriate adaptation strategy to avoid violation.	provider benefit from service-based application. The adaptation strategies minimize the impact of service violation.	limited. The approach does not describe the process to follow when a violation has occurred. Migrating to other service providers has many issues.
Machine learning regression [57]	Runtime prediction of SLA violation for composite services using existing QoS data. Predicts SLA violation based on the prediction checkpoints.	Predicts SLA violation of composite services. Checkpoints describe the execution of composite services and define the input of the prediction. Consumer is alerted in the event of likely violation.	The selection of checkpoints for prediction is not justified. Prediction only works if data is available. The dataset is not defined. No procedure for avoiding actual violation when the system detects likely violation.
Workload analyzer [43]	Workload analyzer predicts future workload and demand for resources. Statistical data are gathered from different nodes to develop workload profile.	Anticipates future workload fluctuations for SLA violation prediction. Generates an alarm when violation is detected.	This approach works when consumer and provider are using a cloud-TM platform. No suggestion for appropriate remedial action when violation is detected.
Resource management by heuristic policies [58]	Manages resources on runtime using proactive and reactive heuristic policies to help the cloud provider to manage its resources to avoid violation.	Assists application service provider to manage resources. Improved workload prediction model.	No procedure defined for SLA management once violation has occurred. There are no criteria for monitoring intervals.

7. CRITICAL EVALUATION OF EXISTING SLA MANAGEMENT APPROACHES FOR FORMING VIABLE SLAS FROM AN SME SERVICE PROVIDER'S VIEWPOINT

In this section, we present a comparative analysis of SLA management approaches to forming viable SLAs from the viewpoint of the SME service provider, in order to proactively manage possible SLA violations. We compare the approaches according to the basic parameters required for SLA management. They are the focus of the SLA management process (whether in the pre- or post-interaction phase), their ability to predict future QoS to detect possible SLA violations, their approach to determining a process when a possible SLA violation is detected, and recommendations for possible action. The comparisons are presented in Table 5.

Table 5: Critical evaluation of existing SLA management approaches

Source	SLA management Process		Predict SLA / SLO /QoS	Procedure defined when violation threat detected	SLA violation recommendation
	Pre-interaction	Post-interaction			
Emeakaroha et al. [24]	✖	✔	✔	✖	✖
Emeakaroha et al.[21]	✖	✔	✔	✖	✖
Brandic et al. [29]	✖	✔	✔	✔	✖
Haq et al. [30]	✖	✔	✔	✔	✖
Emeakaroha et al. [31]	✖	✔	✔	✔	✖
Mosallanejad et al. [33]	✖	✔	✖	✔	✖
Katsaros et al. [34]	✖	✔	✖	✖	✖
Al Falasi et al. [32]	✖	✔	✖	✖	✖
Chandrasekar et al. [37]	✖	✔	✔	✖	✖
Alhamad et al. [38]	✖	✔	✖	✖	✖
Wang et al. [39]	✖	✔	✖	✖	✖
Hammadi and Hussain [35]	✖	✔	✖	✖	✖
Muchahari and Sinha [2]	✖	✔	✖	✖	✖
Cicotti et al. [52]	✖	✔	✔	✖	✖
Romano et al. [42]	✖	✔	✔	✖	✖
Sun et al. [55]	✖	✔	✔	✖	✖
Hussain et al. [23]	✔	✔	✔	✔	✔
Leitner et al. [57]	✖	✔	✔	✖	✖
Ciciani et al. [43]	✖	✔	✔	✖	✖
Cardellini et al. [58]	✖	✔	✔	✖	✖
Son et al. [10]	✔	✖	✖	✖	✖
Silaghi et al. [11]	✔	✖	✖	✖	✖
Badidi [14]	✔	✖	✖	✖	✖
Pacheco-Sanchez et al. [15]	✔	✖	✔	✖	✖
Wood et al. [16]	✖	✔	✖	✔	✖
Schmieders et al. [56]	✖	✔	✔	✔	✖
Noor and Sheng [40]	✖	✔	✖	✖	✖
Fan and Perros[36]	✔	✖	✖	✖	✖

Our comparative study of the literature demonstrates that there are a variety of approaches for SLA management, including mutually agreed third party, management at the provider or consumer side, and hierarchical self-monitoring and management of SLAs. In almost every approach, SLA management is conducted periodically. In the case of discrepancies, violations are recorded and relevant parties are informed. Works in the literature consider different methods of violation prediction, such as formula-based mapping between SLOs and resource metrics, defining threat thresholds, or applying different mathematical approaches, such as prediction, exponential smoothing, ARIMA, Markov Chain theory and Recursive Least Squares (RLS) to avoid possible violation of an SLA. However, the majority of approaches perform SLA management in the post-interaction time phase when both parties have formed the SLA. Although existing approaches try to avoid SLA violation and maintain a trusted relationship between both parties, they could be improved by first forming a viable SLA based on an intelligent determination of likely violations by the consumer before entering into an agreement. For an SME service provider, an SLA that reflects the consumer's requirements would not only assist with resource provisioning but would also help guide the provider in their decision to accept a customer. As evident from the literature [19, 59] the choice in selecting a consumer depends upon the cost/benefit ratio of the service provider. The service provider assesses a consumer based on certain parameters and decides to either accept or reject a new consumer request. To maximize their profit, an SME cloud provider desires to commit resources to consumers who will fully utilize them, therefore it is necessary to commence SLA management from the pre-interaction time phase. Once a provider is able to form a viable SLA, the discrepancies between parties are easily identified and are fixed by proactively managing each SLO, before either party is affected. In conclusion, we found the following shortcomings in the literature for efficient SLA management from the perspective of SME cloud service providers:

- Most SLA management approaches fail to guide a service provider in their decisions about service formation with a consumer. Providers thus form SLAs with users who may have flawed intent that will result in them failing to achieve the financial revenue they expect to generate in a given time period. In other words, most of the existing literature presents SLA management when the SLA is executed; none of the methods describe SLA management by considering the consumer's previous transaction history.
- The bulk of the literature focuses on SLA management from the consumer's perspective when an SLA is executed between a consumer and a provider. Some approaches, such as [10], help the cloud provider to adaptively control SLA negotiation parameters such as price, performance and timeslots, but they do not specify how these factors can be used by the provider to decide on marginal resources when SLAs are formed according to workload trends.
- Nothing in the literature determines the maximum amount of marginal resources to be offered to a consumer relative to information about the consumer's past performance, which is required to form a viable SLA.
- From a provider's perspective, the literature is unable to describe a complete SLA management framework that starts by forming a viable SLA (offering the optimal amount of marginal resources), monitoring the runtime behavior of the consumer, predicting the likely resource usage, and identifying and managing the risk of SLA violation.

8. OPTIMIZED PERSONALIZED VIABLE SLA (OPV-SLA) FRAMEWORK

Considering the shortcomings in the literature, we propose a novel optimized personalized viable SLA management framework (OPV-SLA) as shown in Figure 4. The proposed framework assists the service provider to form personalized and viable SLAs with the various consumers requesting resources [60]. This enables the process of possible SLA violation detection to start at the SLA negotiation phase and not after an SLA has been formed, as happens in most of the approaches in the literature. As shown in Figure 4, OPV-SLA performs computations over two different time phases, namely the pre-interaction time phase and post-interaction time phase [61]. The computations in the pre-interaction time phase, shown in Figure 4, are carried out by two modules: the Identity Manager Module (IMM) and Viable SLA module (VSLAM), whereas in the post-interaction time phase, these computations are run by four modules: the threshold formation module (TFM), the runtime QoS monitoring module (RQoSMM), the QoS prediction module (QoSPM) and the risk management module (RMM). A brief explanation of each phase in the pre- and post-interaction start time phases is discussed below.

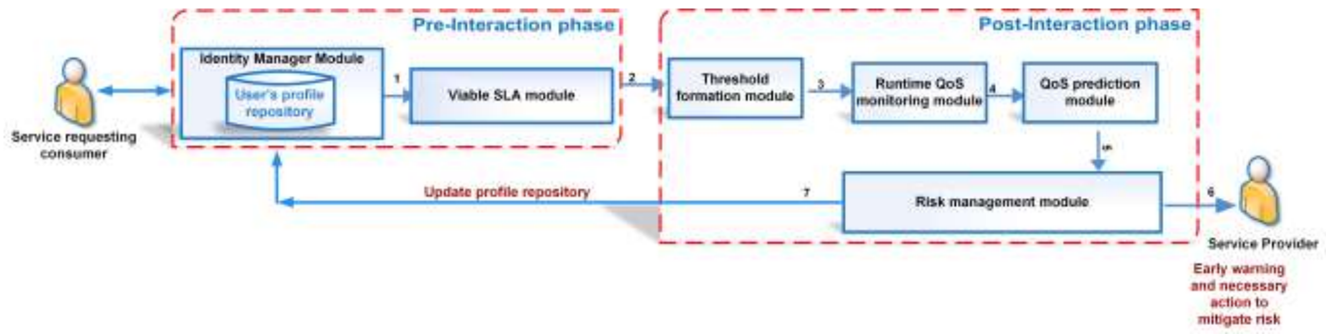

Pre-Interaction phase
Post-Interaction phase
Identity Manager Module
User's profile repository
1
Viable SLA module
2
Threshold formation module
3
Runtime QoS monitoring module
4
QoS prediction module
5
Risk management module
6
7
Update profile repository
Service requesting consumer
Service Provider
Early warning and necessary action to mitigate risk

Figure 4: Framework for formulating viable SLAs [61, 62]

8.1 Pre-interaction time phase

When a consumer requests resources or services, the service provider first validates the user using the Identity Management Module (IMM), with one of two possible outcomes: either the consumer is new or the consumer has a previous record of using services from the provider, as presented in Figure 5. Depending upon the category, the consumer is validated and the request is forwarded to the Viable SLA Module (VSLAM). As shown in Figure 6, VSLAM performs computations, based on which the decision is made to accept or reject the consumer's request. If the request is to be accepted, the amount of resources to be offered is determined. VSLAM comprises two sub-modules, namely the Consumer's request assessment module (CRAM) and the resource allocation determination module (RADM), which assist in making this decision. A brief explanation of each phase is given below.

8.1.1 *Consumer's request assessment module (CRAM)*

CRAM utilizes the trust value of a requesting consumer to determine whether or not to allocate resources. If the IMM determines that the consumer has a previous record, the concept of transaction trend (T_{trend}) is utilized. T_{trend} is the number of successful transactions or the successful commitment by the requesting consumer to the formed SLAs divided by the total number of transactions it has performed. For existing consumers, CRAM considers the consumer's previous profile and calculates their T_{trend} value. For a new consumer who does not have a previous profile, CRAM determines the top-K nearest neighbors that are similar to the requesting consumer's profile, and based on their T_{trend}, calculates the likely T_{trend} value of the requesting consumer. The requesting consumer's T_{trend} value is compared with the defined threshold value, which is the success ratio defined by a provider to classify a consumer as reliable or not. Based on the comparative result, the CRAM either accepts a request or it does not. If the request is accepted, the next decision making factor is to determine how much resource to offer to the consumer. This is done by the RADM module, which is explained in the next sub-section.

8.1.2 *Resource allocation determination module (RADM)*

A cloud provider offers static and marginal resources and, due to the dynamic nature of cloud, it is very important for providers to decide wisely how much of its marginal resources it wants to offer to a consumer in light of their trustworthiness value and the time they are requesting. In our framework, the provider is assisted in this decision-making by the RADM. RADM takes the reliability of the consumer (T_{trend} value), the contract duration and the risk propensity of the service provider as inputs and, using a multi-layered Fuzzy Inference System (FIS), decides the resource amount to offer the consumer. Based on the output from RADM, the provider informs the consumer how much resource it will offer. The consumer accepts, rejects or renegotiates the offer depending upon its circumstances, and when both parties have agreed, a formal SLA is signed and the provider is bound to reserve the committed resources for that consumer.

As a result of this series of steps, the SLA that is formed is a viable SLA in which the provider already knows the expected behavior of the requesting consumer through its transaction trend. Once the SLA has been formed, the management process shifts to the post-interaction time phase.

8.2 Post-interaction time phase

The second phase in the management process of OPV-SLA is the post-interaction time phase which we term the *provider-based Risk Management Framework for SLA violation abatement (RMF-SLA)*. The RMF-SLA framework enables SME cloud providers to manage the risk of SLA violations to avoid penalties. The proposed framework performs SLA monitoring in the post-interaction time phase, and detects and manages the risk of possible SLA violation by suggesting an appropriate action that the cloud provider should take. The run-time behaviour of consumers is constantly compared with the SLAs formed in the previous phase, based on which an early warning is generated in the event of possible SLA violation. This phase comprises the following five sub-modules, as shown in Figure 7: Threshold Formation Module (TFM), Runtime QoS Monitoring Module (RQoSMM), QoS Prediction Module (QoSPM), Risk Identification Module (RIM) and Risk Management Module (RMM). A brief explanation of each of these modules is given below.

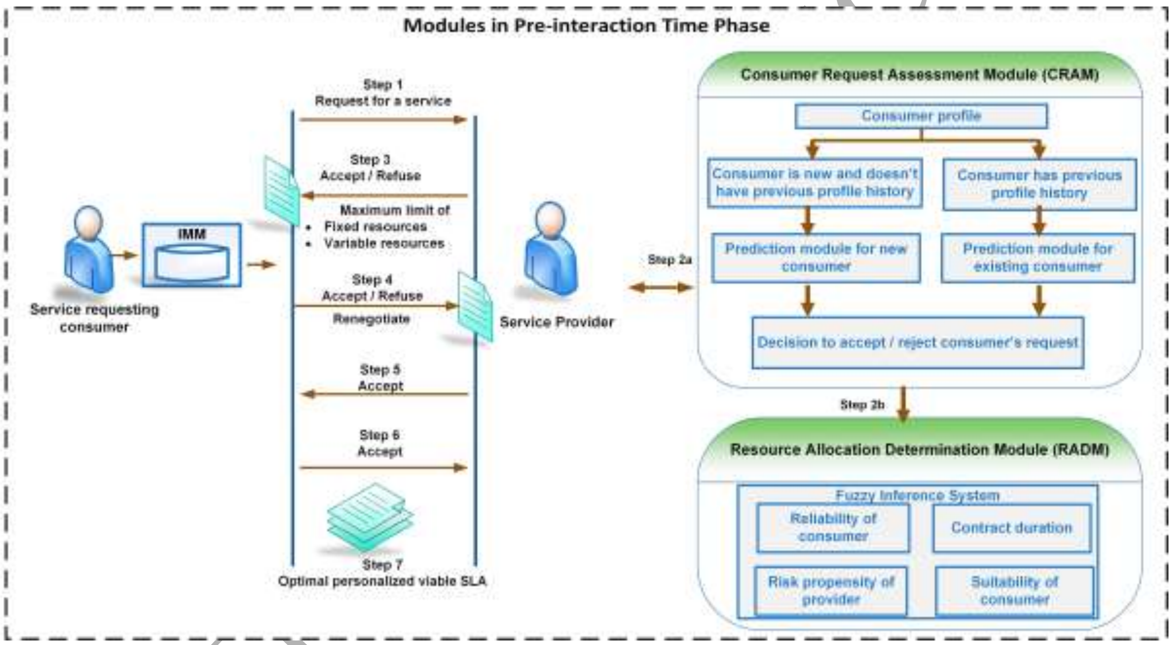

Modules in Pre-interaction Time Phase
Service requesting consumer
IMM
Step 1
Request for a service
Step 3
Accept / Refuse
Maximum limit of
Fixed resources
Variable resources
Step 4
Accept / Refuse
Renegotiate
Step 5
Accept
Step 6
Accept
Step 7
Optimal personalized viable SLA
Service Provider
Step 2a
Consumer Request Assessment Module (CRAM)
Consumer profile
Consumer is new and doesn't have previous profile history
Consumer has previous profile history
Prediction module for new consumer
Prediction module for existing consumer
Decision to accept / reject consumer's request
Step 2b
Resource Allocation Determination Module (RADM)
Fuzzy Inference System
Reliability of consumer
Contract duration
Risk propensity of provider
Suitability of consumer

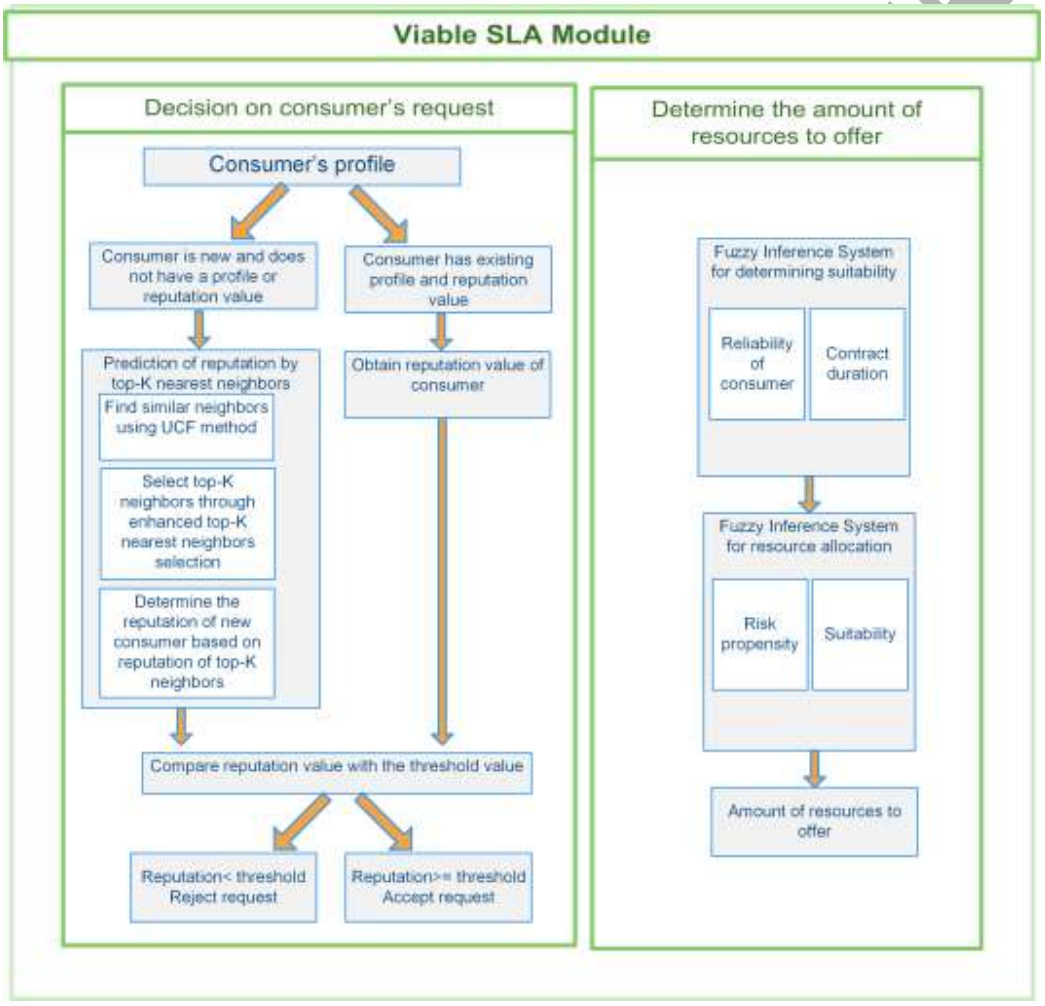
Figure 5: Modules in pre-interaction time phase

8.2.1 *Threshold formation module (TFM)*

This module is responsible for defining a threshold value for a provider upon which an early warning will be generated in the event of a violation occurring. In our framework, we defined two thresholds: the agreed threshold (T_a) and the safe threshold (T_s). T_a is the threshold that a provider and consumer have agreed in respect of each SLO and is defined in the SLA. T_s is a threat threshold that a provider forms for its own security. To explain the notion of Ts and Ta with an example, let us assume a provider and consumer agree on the provider giving 20TB of storage space to the consumer from 6 PM to 8 PM on 20/08/2016. The availability of the 20TB of storage space is the Ta value, agreed by both parties, which is also defined in the SLA. However, for service management and possible SLA violation abatement, a provider defines its customized threshold for the storage, say 22TB, from 6 PM to 8 PM on 20/08/2016, which is a Ts value for the provider. When the runtime availability of the memory space falls below Ts (22TB from 6 PM to 8 PM on 20/08/2016) the framework alerts the service provider and activates the risk management module to manage any risk of the provider violating the formed viable SLA.

8.2.2 *Runtime QoS monitoring module (RQoSMM) and QoS prediction module (QoSPM)*

RQoSMM is responsible for monitoring the runtime QoS parameters of each agreed SLO. Once the QoS parameters at the current point of time are observed, they are sent to the QoS prediction module (QoSPM) where they are used to recalibrate the QoS of SLOs in the near future. The QoSPM module of RMF-SLA is responsible for predicting the resource usage of consumers in terms of QoS parameters over the SLA time period to detect possible violations. The consumer's likely resource usage is predicted using the resource history and an optimal prediction algorithm. In our previous work [63], we observed that an optimal prediction result is obtained by considering small time intervals and using the Autoregressive Integrated Moving Average (ARIMA) method. The accuracy of a prediction result is enhanced by considering the value of the SLOs in the previous time intervals from the RQoSMM, thereby constantly updating it.


Viable SLA Module
Decision on consumer's request
Consumer's profile
Consumer is new and does not have a profile or reputation value
Consumer has existing profile and reputation value
Prediction of reputation by top-K nearest neighbors
Find similar neighbors using UCF method
Select top-K neighbors through enhanced top-K nearest neighbors selection
Determine the reputation of new consumer based on reputation of top-K neighbors
Obtain reputation value of consumer
Compare reputation value with the threshold value
Reputation< threshold Reject request
Reputation>= threshold Accept request
Determine the amount of resources to offer
Fuzzy Inference System for determining suitability
Reliability of consumer
Contract duration
Fuzzy Inference System for resource allocation
Risk propensity
Suitability
Amount of resources to offer

.

Figure 6: Sequence of steps in VSLA module of OPV-SLA [62]

8.2.3 *Risk identification module (RIM)*

The risk identification module (RIM) is responsible for comparing the value from QoSPM with the Ts value on a regular basis. If the value of the QoSPM reaches or exceeds the Ts value, it activates the risk management module (RMM) to manage the possible risk of SLA violation.

8.2.4 *Risk management module (RMM)*

This module is comprised of two sub modules: the risk estimation module (REM) and the risk mitigation module (RMtM). REM is activated with RIM to determine the possible occurrence of SLA violation and estimates the risk. Decisions on risk estimation depend on the risk attitude of the provider, the reputation of the consumer, and the transaction trend curve of future intervals. The risk attitude of a provider is the provider's capacity to deal with risk. A provider with a risk propensity of risk averse is more reluctant to take a risk than a provider with an attitude that is risk neutral or risk taking. The reputation of a consumer is its reliability or trust value, namely T_{trend} value, which is determined by CRAM in the pre-interaction phase. The third input is the predicted resource usage by the consumer, determined by the QoSPM in the post-interaction phase. The processed output of these input variables is determined by RMtM which gives the estimated risk of possible violation as either high risk, medium risk or low risk. Depending on the level of risk determined, the provider chooses an appropriate action to manage and mitigate possible violation of the formed SLA. When the risk of possible SLA violation is assessed as high, the module sends an alarm to the service provider for immediate action. When the risk is estimated as medium or low, the service provider decides whether to take delayed action or no action, depending on the input values. The provider arranges sufficient resources within a certain time period. When the risk is estimated as low, it has no significant effect on the provider. The provider accepts the risk and does not take any action.

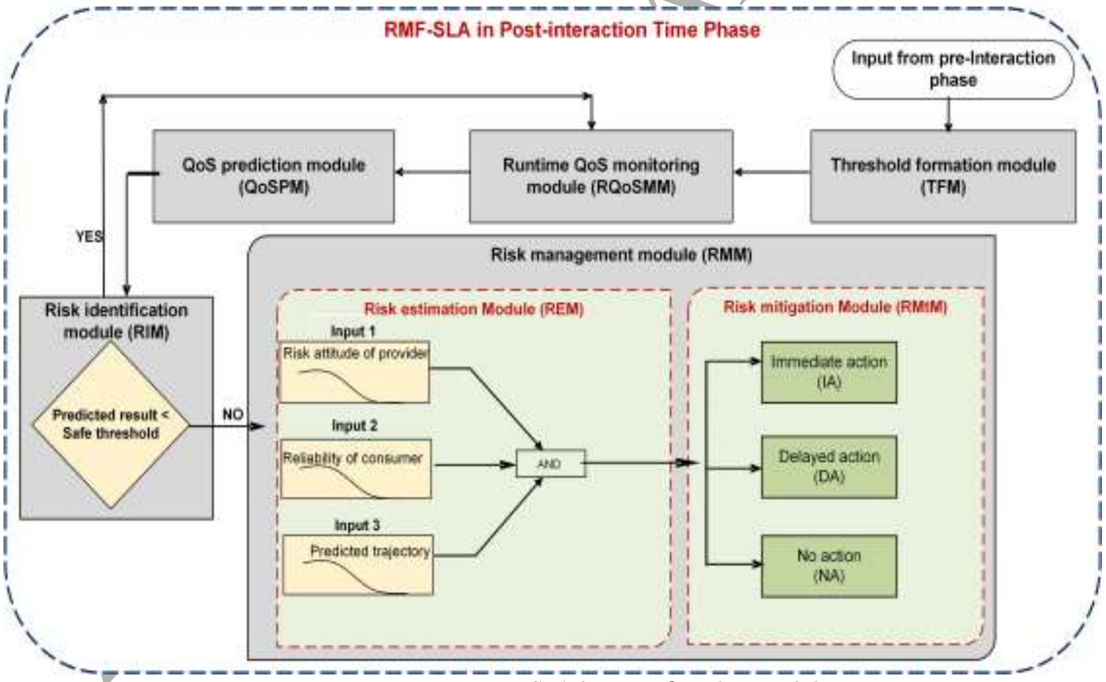

RMF-SLA in Post-interaction Time Phase
Input from pre-Interaction phase
QoS prediction module (QoSPM)
Runtime QoS monitoring module (RQoSMM)
Threshold formation module (TFM)
YES
Risk management module (RMM)
Risk identification module (RIM)
Predicted result < Safe threshold
NO
Risk estimation Module (REM)
Input 1
Risk attitude of provider
Input 2
Reliability of consumer
Input 3
Predicted trajectory
AND
Risk mitigation Module (RMtM)
Immediate action (IA)
Delayed action (DA)
No action (NA)

Figure 7: RMF-SLA framework and its modules

By using the proposed framework, a service provider is first able to form viable SLAs and then manage them in the best way. In the next section, we present the results of the validation of our approach in the pre-interaction start time phase and demonstrate the applicability of OPV-SLA in forming viable SLAs from the viewpoint of small to medium cloud service providers.

9. VALIDATION AND APPLICABILITY OF OPV-SLA IN VIABLE SLA FORMATION AND MANAGEMENT

We validate the OPV-SLA framework in the pre- and post-interaction time phases. The objective of OPV-SLA in the pre-interaction time phase is to assist the service provider to form viable SLAs, and in the post-interaction time phase it is to manage or prevent SLA violations. We use two datasets to form a viable SLA. The first is the QoS dataset used by Zhang et al. [64], which comprises 142 users using 4532 web services for 64 time intervals, and the second dataset is from Amazon Elastic Compute Cloud (EC2) IaaS cloud services – EC2 US East, collected from cloudclimate [65] through the Paessler Router Traffic Grapher (PRTG) monitoring service [66]. In our experiments, we consider two Quality of Service (QoS) parameters, namely the throughput and response time in which SLAs are formed in the pre-interaction phase, and one QoS parameter, namely CPU usage, when managing SLAs in the post-interaction phase. The captured datasets are stored in Microsoft Visual Studio 2010 with Microsoft SQL Server Management Studio 2008 for the databases, and MATLAB is used to program the computations in the pre-interaction start time phases to form viable SLAs. Our objective in using these computations is to demonstrate how a provider can form viable SLAs by utilizing the previous record of a consumer's commitment to the formed SLAs, and can start the process of SLA management before the SLAs are formed, as opposed to approaches in which this is done after the formation of SLAs.

We assume that there are three possible consumers with the consumer IDs 122, 254 and 111 that are requesting services from a service provider. The implementation process in the OPV-SLA takes the following steps in the pre-interaction start time phase to form viable SLAs:

Step 1: When IMM receives a request with all the details of a consumer, it authenticates the consumer as either a returning customer or a new customer, as shown in Figure 4. To demonstrate the working of our framework, let us suppose the first consumer with consumer ID 122 is new, whereas the second and third consumers with consumer IDs 254 and ID 111 respectively have an existing record with the provider. The requests and all detail of the consumers as shown in Figure 4 are passed to VSLAM for a decision on the consumer's request and to decide the amount of marginal resources to be offered.

Step 2: The VSLAM, as shown in Figure 5, determines the T_{trend} value for all consumers. T_{trend} is the consumer's previous profile and is an important factor in determining possible future SLA violations [67, 68]. As consumer 122 is a new customer, the module selects its top-K nearest neighbors according to Pearson Correlation Coefficient (PCC) value to determine the T_{trend} value. The top-K nearest neighbors with their T_{trend} are presented in Table 6.

Table 6: Top-K NN with their T_{trend} value for consumer - ID 122 (reproduced from [62])

	Nearest neighbors Consumer #	**Level of nearness according to PCC value**	**No. of successful transactions**	**No. of violated transactions**	**Transaction trend**
1	131	0.9999889	6	4	60.00%
2	63	0.9999765	3	3	50.00%
3	11	0.9999503	3	19	13.63%
4	39	0.9999412	7	2	77.78%
5	87	0.9997249	1	20	4.76%
6	05	0.9992545	11	8	57.89%
7	110	0.9992148	11	4	73.33%
8	72	0.9992032	8	3	72.73%
9	15	0.9992014	0	1	0.00%

Let us consider that the provider has set a T_{trend} threshold of 40% to even consider forming resource provision requests from consumers. The T_{trend} obtained for consumer 122 from its top-KNN is T_{trend} = 45.55%. The T_{trend} for consumers 254 and 111 is determined by their past history and presented in Table 7.

Table 7: Transaction trends of consumers ID – 254 and 34 (reproduced from [62])

#	Consumer Number	No. of successful transactions	No. of violated transactions	Transaction trend $((T_s/T_n)*100$
1	254	6	5	54.55%
2	122	2	6	25. 00%

From Tables 6 and 7, we see that the T_{trend} value of consumers 122 and 254 is above the threshold value, but that the T_{trend} value of consumer 111 is below the threshold value, therefore the provider accepts only the request from consumers 122 and 254. This avoids forming a service-provisioning request with consumer 111, which may lead to possible SLA violation.

Step 3: In this step, VSLAM determines the amount of resources to offer to the customers whose requests it accepts, as shown in Figure 5. Upon determining the resource amount, the service provider and service consumer negotiate further and decide on the specific quantity of resources on which to form an SLA. To determine the amount of resources to offer, the VLSLAM utilizes fuzzy inference systems which combine many different variables to reach a decision on resource allocation and the amount of resources to offer to each requesting customer, as shown in Figure 6. A brief explanation of the different variables used is as follows:

<u>*Suitability value:*</u> In our proposal, the provider categorizes customers requesting resources according to four levels of suitability: none, low, medium and high. These levels are determined by fuzzy inference rules based on the reliability value of a customer and the duration for which they are requesting resources. The fuzzy rules are formed such that the provider gives high preference to requests from *reliable* customers who reserve resources for a short *time period*.

<u>*Decision on allocation and amount of resources to offer:*</u> The *suitability* value determined for each customer is combined with the *risk propensity* or risk appetite value of the service provider to ascertain whether an SLA should be formed with a consumer, and if so, the level at which its request should be accepted. The four fuzzy predicates over which the decision to allocate resources to a consumer are: none, marginal, partial and full. The fuzzy rules are formed to capture the risk attitude of the provider (risk averse, risk neutral and risk taking) with the suitability value to ascertain the level of acceptance of the consumer's request for resources.

Table 8 shows the results of performing the computations using the above processes, and details the amount of resources to offer to consumers with 122 and 254. The table presents the T_{trend} value for each customer along with the threshold set by the provider (from Step 2), the determined suitability value of each consumer (expressed as fuzzy variables over the range of low, medium and high) the risk attitude of the provider (expressed as fuzzy variables over the range of risk averse, risk neutral and risk taking) with the recommended decision on the consumer's request for service provisioning along with the level at which to accept the request.

Table 8: Request decision and amount of resources offered (reproduced from[62])

Consumer ID	T_{trend}	Threshol	Suitability value		Risk propensity		Decision on consumer request	Resource allocate
122	45.55%	40%	M=0.1	H=0.5	RN=1.0	RA=0.0	Accept	19.12%
254	54.55%	40%	M=0.3	H=0.7	RN=0.6	RA=0.4	Accept	65.68%
111	25.00%	40%	---	---	---	---	Reject	---

We see from Table 8 that the request by consumer 111 is rejected because it does not satisfy the required threshold, and the requests by consumers 122 and 254 are recommended for acceptance with provisioning at 19.12% and 65.68% of the requested marginal requests respectively. The provider is thus able to make the optimal decision regarding the consumers' requests and the

amount of resources to offer them. Negotiation, if needed, can be conducted between the provider and consumer following this stage. The approaches in the literature do not consider this approach for SLA management, but by using the OPV-SLA framework, the drawbacks mentioned at the end of Section 7 regarding the need to form viable SLAs can be addressed.
Once a viable SLA has been formed, the next step in SLA management is to anticipate possible SLA violations and take appropriate steps to manage them. As discussed earlier, we use the notion of risk in OPV-SLA to achieve this in the post-interaction time phase and manage the SLO CPU usage. We consider that, based on the recommendations from VSLAM as shown in Table 8, the provider forms an SLA and resource provisioning agreement with consumer 254 in the SLO - CPU usage. Let us consider that the *Ta* value of *290ms* is the defined level of commitment between the provider and the consumer in this SLO. As explained earlier, T_a is the threshold that a provider and consumer have agreed for each SLO and defined in the SLA. Our objective by using this value and using the RMF-SLA framework is to demonstrate how the provider constantly monitors the QoS parameters and ascertains in advance the likelihood of SLA violation occurring, along with taking appropriate action for violation management.

Step 4: Once the service resource provisioning between a consumer and provider has started, the provider defines a safe threshold T_s for SLA management that is stricter than the T_a agreed by both users at the SLA formation stage, as shown in Figure 6. Let us consider that the defined safe threshold level determined by the provider in the case of CPU SLO is *260ms*. Once the T_a and T_s values have been defined, the past data point values related to the resource usage of consumer 254 in terms of CPU usage over past SLAs or the available resources on the provider's side related to CPU SLO are captured and sent to the QoSPM module for prediction over the future period.

Step 5: Using the data collected from Amazon EC2 EU, the QoS values of the SLO CPU are predicted over the next 60 minutes in time intervals of 15 minutes each. Table 9 shows the predicted values for CPU SLO by the ARIMA method for the period 11:45 AM to 12:45 PM on 3/2/2016.

Table 9: Predicted values for CPU SLO by ARIMA method

Interval	Predicted result	T_s	T_a
3/2/2016 11:45:00 AM - 12:00:00 PM	249.028569ms	260ms	290ms
3/2/2016 12:00:00 PM - 12:15:00 PM	258.0277777ms	260ms	290ms
3/2/2016 12:15:00 PM - 12:30:00 PM	266.8571404ms	260ms	290ms
3/2/2016 12:30:00 PM - 12:45:00 PM	272.818178ms	260ms	290ms

Figure 9 graphically represents the predicted QoS value of CPU along with the agreed threshold and defined safe threshold. The blue line represents the prediction result using the ARIMA method for the time interval, the red line represents the T_s value, and the green line represents the T_a value. The objective in this representation is to ascertain when the CPU QoS value is expected to intersect or exceed T_s in order to activate the RMM module of RMF-SLA. The process of ascertaining whether the predicted SLO value at a given point in the future is more or less than the safe threshold is performed by the RIM module of RMF-SLA. Figure 7 shows that the provider, through RIM, notes that the predicted QoS value of SLO CPU at the first interval of 11:45:00 AM to 12:00:00 PM is less than the T_s value, but it starts to increase from 12:00:00PM and exceeds the T_s value for the next two intervals, 12:15:00 PM and 12:30:00 PM. When the predicted result exceeds the T_s value, the RMM is activated to predict the possibility of a violation occurring and to manage it accordingly [69].

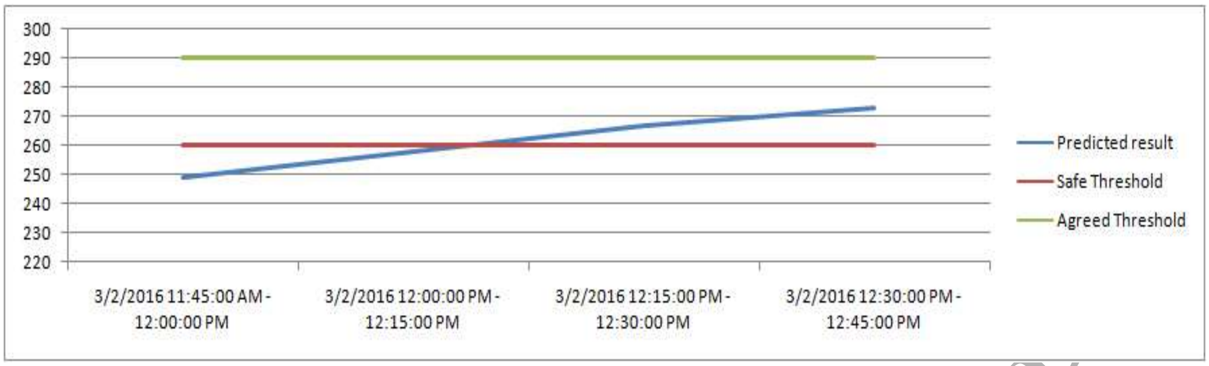


Figure 9: Representation of predicted CPU SLO value along with defined T_a and T_s

Step 6: In this step, RMM ascertains the risk of a possible SLA violation occurring by considering three decision-making variables. The first variable is the risk attitude of the service provider, classified on the levels of risk averse, risk neutral and risk taking. The second input variable is the reliability of the consumer, classified on the levels of bronze, silver and gold according to their commitment to previous SLAs. The premise here is that the more reliable the service consumer is, the more responsive the service provider will be to managing and avoiding the possible risk of non-service provisioning to these users. The third decision making variable is the direction of the predicted trajectory, which is either towards T_a or away from T_a once it crosses T_s. When the predicted trajectory is defined as 'Towards', it means that the trajectory has reached the T_s and is moving towards the T_a. When the predicted trajectory is defined as 'Away', it means that the trajectory has exceeded the T_s value and is moving back towards the T_s. These inputs are important considerations in ascertaining the possibilitiy of SLA violation and are used to manage the risk.

Table 10: Output of RMM showing the action to be taken to avoid possible SLA violation

Consumer ID	T_{trend}	Risk propensity value of provider		Predicted trajectory	Decision recommendation	
254	4.54%	RN=1.0	RA=0.0	towards	Immediate action = 67%	Delayed action = 33%
254	20%	RN=1.0	RT=0.0	away	Immediate action = 33%	Delayed action = 67%

Continuing with the discussion of consumer 254, the output from the fuzzy inference rules after considering consumer reliability, the service provider's risk attitude (from Table 8) and the projected trajectory of T_s (from Figure 9), is shown in Table 10. The first row of the table shows the RMM output after defuzzification as 67% towards immediate action and 33% towards delayed action. This is determined by considering the customer's reliability, the service provider's risk attitude and the project trajectory, and utilizing this information to categorize the risk as high risk with a very high possibility of SLA violation. The recommendation is for the provider to take immediate action to remove the risk of SLA violation at the earliest possible time. The second row of Table 10 shows the RMM output if the input details were to be changed as shown. It can be seen from the output that, depending on the scenario, RMM will recommend the most appropriate action for the provider to take to avoid possible SLA violation. These two phases of OPV-SLA, when combined, assist the provider to first form viable SLAs and then to manage them appropriately to prevent SLA violations.

10. CONCLUSION AND FUTURE WORK

The rise of cloud computing promises to eliminate the need for managing complex and expensive computing resources. The elastic nature of cloud computing allows cloud providers to maximize profits if they can ensure the provision of an optimal level of resources to meet consumer needs. This is very important if the provider is an SME that has limited resources from which to generate and maximize its revenue. To achieve a satisfactory outcome, SME cloud providers need to intelligently determine the likely resource usage of prospective consumers and form a viable SLA that allows them to meet those needs. In this work, we have described existing SLA management approaches, highlighting their limitations in addressing this problem. We have discussed our proposed Optimized Personalized Viable SLA (OPV-SLA) framework and demonstrated the working of each of its phases. In contrast to the approaches in the literature, our proposed approach focuses on the SME cloud service provider and assists with the formation and management of optimal and viable SLAs with consumers. In our future work, we will look at applying the framework in a real world SME cloud provider setting.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology,* vol. 53, p. 50, 2009.

[2] M. K. Muchahari and S. K. Sinha, "A new trust management architecture for cloud computing environment," in *Cloud and Services Computing (ISCOS), 2012 International Symposium on*, 2012, pp. 136-140.

[3] C. Ardagna, E. Damiani, F. Frati, G. Montalbano, D. Rebeccani, and M. Ughetti, "A Competitive Scalability Approach for Cloud Architectures," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, 2014, pp. 610-617.

[4] Z. Liu, M. S. Squillante, and J. L. Wolf, "On maximizing service-level-agreement profits," in *Proceedings of the 3rd ACM conference on Electronic Commerce*, 2001, pp. 213-223.

[5] O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu, "Managing violations in service level agreements," in *Grid Middleware and Services,* ed: Springer, 2008, pp. 349-358.

[6] H. H. Saunders, "We need a larger theory of negotiation: The importance of pre-negotiating phases," *Negotiation journal,* vol. 1, pp. 249-262, 1985.

[7] S. Ron and P. Aliko, "Service level agreements," *Internet NG project,* 2001.

[8] E. Wustenhoff and S. BluePrints, "Service level agreement in the data center," *Sun BluePrints,* 2002.

[9] O. K. Hussain, T. Dillon, F. K. Hussain, and E. Chang, *Risk assessment and management in the networked economy* vol. 412: Springer, 2012.

[10] S. Son, D.-J. Kang, S. P. Huh, W.-Y. Kim, and W. Choi, "Adaptive trade-off strategy for bargaining-based multi-objective SLA establishment under varying cloud workload," *The Journal of Supercomputing,* vol. 72, pp. 1597-1622, 2016.

[11] G. C. Silaghi, L. D. ŞErban, and C. M. Litan, "A time-constrained SLA negotiation strategy in competitive computational grids," *Future Generation Computer Systems,* vol. 28, pp. 1303-1315, 2012.

[12] J. Gwak and K. M. Sim, "A novel method for coevolving PS-optimizing negotiation strategies using improved diversity controlling EDAs," *Applied intelligence,* vol. 38, pp. 384-417, 2013.

[13] K. M. Sim, "Grid resource negotiation: survey and new directions," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 40, pp. 245-257, 2010.

[14] E. Badidi, "A cloud service broker for SLA-based SaaS provisioning," in *Information Society (i-Society), 2013 International Conference on*, 2013, pp. 61-66.

[15] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for qos prediction in the cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 147-154.

[16] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks,* vol. 53, pp. 2923-2938, 2009.

[17] W. Hussain, F. K. Hussain, and O. K. Hussain, "Maintaining trust in cloud computing through sla monitoring," in *International Conference on Neural Information Processing*, 2014, pp. 690-697.

[18] L. Wu and R. Buyya, "Service level agreement (sla) in utility computing systems," *IGI Global,* 2012.

[19] L. Wu, S. K. Garg, and R. Buyya, "SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 195-204.

[20] S. Sakr and A. Liu, "SLA-based and consumer-centric dynamic provisioning for cloud databases," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 2012, pp. 360-367.

[21] V. C. Emeakaroha, M. A. Netto, R. N. Calheiros, I. Brandic, R. Buyya, and C. A. De Rose, "Towards autonomic detection of SLA violations in Cloud infrastructures," *Future Generation Computer Systems,* vol. 28, pp. 1017-1029, 2012.

[22] V. C. Emeakaroha, R. N. Calheiros, M. A. Netto, I. Brandic, and C. A. De Rose, "DeSVi: an architecture for detecting SLA violations in cloud computing infrastructures," in *Proceedings of the 2nd International ICST conference on Cloud computing (CloudComp'10)*, 2010.

[23] O. K. Hussain, F. K. Hussain, J. Singh, N. K. Janjua, and E. Chang, "A User-Based Early Warning Service Management Framework in Cloud Computing," *The Computer Journal,* p. bxu064, 2014.

[24] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, 2010, pp. 48-54.

[25] H. Liu, D. Xu, and H. K. Miao, "Ant colony optimization based service flow scheduling with various QoS requirements in cloud computing," in *Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on*, 2011, pp. 53-58.

[26] S. Yun, "Guaranteed QoS Resource Scheduling Scheme Based on Improved Electromagnetism-Like Mechanism Algorithm in Cloud Environment," in *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, 2013, pp. 353-357.

[27] A.-F. Antonescu, P. Robinson, and T. Braun, "Dynamic sla management with forecasting using multi-objective optimization," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 2013, pp. 457-463.

[28] R. Sahal, M. H. Khafagy, and 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,* vol. 9, pp. 107-118, 2016.

[29] I. Brandic, V. C. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertesz*, et al.*, "Laysi: A layered approach for sla-violation propagation in self-manageable cloud infrastructures," in *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, 2010, pp. 365-370.

[30] I. U. Haq, I. Brandic, and E. Schikuta, "Sla validation in layered cloud infrastructures," in *Economics of Grids, Clouds, Systems, and Services*, ed: Springer, 2010, pp. 153-164.

[31] V. C. Emeakaroha, T. C. Ferreto, M. A. Netto, I. Brandic, and C. A. De Rose, "Casvid: Application level monitoring for sla violation detection in clouds," in *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, 2012, pp. 499-508.

[32] A. Al Falasi, M. A. Serhani, and R. Dssouli, "A Model for Multi-levels SLA Monitoring in Federated Cloud Environment," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, 2013, pp. 363-370.

[33] A. Mosallanejad and R. Atan, "HA-SLA: A Hierarchical Autonomic SLA Model for SLA Monitoring in Cloud Computing," *Journal of Software Engineering and Applications,* vol. 6, p. 114, 2013.

[34] G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, A. Menychtas, and T. Varvarigou, "A Self-adaptive hierarchical monitoring mechanism for Clouds," *Journal of Systems and Software,* vol. 85, pp. 1029-1041, 2012.

[35] A. M. Hammadi and O. Hussain, "A framework for SLA assurance in cloud computing," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 393-398.

[36] W. Fan and H. Perros, "A reliability-based trust management mechanism for cloud services," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, 2013, pp. 1581-1586.

[37] A. Chandrasekar, K. Chandrasekar, M. Mahadevan, and P. Varalakshmi, "QoS monitoring and dynamic trust establishment in the cloud," in *Advances in Grid and Pervasive Computing*, ed: Springer, 2012, pp. 289-301.

[38] M. Alhamad, T. Dillon, and E. Chang, "Sla-based trust model for cloud computing," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, 2010, pp. 321-324.

[39] M. Wang, X. Wu, W. Zhang, F. Ding, J. Zhou, and G. Pei, "A conceptual platform of SLA in cloud computing," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, 2011, pp. 1131-1135.

[40] T. H. Noor and Q. Z. Sheng, "Trust as a service: a framework for trust management in cloud environments," in *Web Information System Engineering–WISE 2011*, ed: Springer, 2011, pp. 314-321.

[41] M. Egea, K. Mahbub, G. Spanoudakis, and M. R. Vieira, "A Certification Framework for Cloud Security Properties: The Monitoring Path," in *Accountability and Security in the Cloud*, ed: Springer, 2015, pp. 63-77.

[42] L. Romano, D. De Mari, Z. Jerzak, and C. Fetzer, "A novel approach to QoS monitoring in the cloud," in *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*, 2011, pp. 45-51.

[43] B. Ciciani, D. Didona, P. Di Sanzo, R. Palmieri, S. Peluso, F. Quaglia*, et al.*, "Automated workload characterization in cloud-based transactional data grids," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, 2012, pp. 1525-1533.

[44] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI communications,* vol. 7, pp. 39-59, 1994.

[45] D. Chua, D. Li, and W. Chan, "Case-based reasoning approach in bid decision making," *Journal of construction engineering and management,* vol. 127, pp. 35-45, 2001.

[46] W. Cheetham, A. Varma, and K. Goebel, "Case-Based Reasoning at General Electric," in *FLAIRS Conference*, 2001, pp. 93-97.

[47] I. U. Haq, A. Paschke, E. Schikuta, and H. Boley, "Rule-based workflow validation of hierarchical service level agreements," in *Grid and Pervasive Computing Conference, 2009. GPC'09. Workshops at the*, 2009, pp. 96-103.

[48] K. Lu, R. Yahyapour, P. Wieder, E. Yaqub, M. Abdullah, B. Schloer*, et al.*, "Fault-tolerant Service Level Agreement lifecycle management in clouds using actor system," *Future Generation Computer Systems,* 2015.

[49] O. Fachrunnisa and F. K. Hussain, "A methodology for maintaining trust in industrial digital ecosystems," *Industrial Electronics, IEEE Transactions on,* vol. 60, pp. 1042-1058, 2013.

[50] T. B. Quillinan, K. P. Clark, M. Warnier, F. M. Brazier, and O. Rana, "Negotiation and monitoring of service level agreements," in *Grids and Service-Oriented Architectures for Service Level Agreements*, ed: Springer, 2010, pp. 167-176.

[51] D. Marudhadevi, V. N. Dhatchayani, and V. S. Sriram, "A Trust Evaluation Model for Cloud Computing Using Service Level Agreement," *The Computer Journal,* p. bxu129, 2014.

[52] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS prediction in cloud computing," in *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*, 2011, pp. 1-10.

[53] G. Cicotti, L. Coppolino, S. D'Antonio, and L. Romano, "How to monitor QoS in cloud infrastructures: the QoSMONaaS approach," *International Journal of Computational Science and Engineering,* vol. 11, pp. 29-45, 2015.

[54] G. Cicotti, L. Coppolino, R. Cristaldi, S. D’Antonio, and L. Romano, "QoS monitoring in a cloud services environment: the SRT-15 approach," in *European Conference on Parallel Processing*, 2011, pp. 15-24.

[55] Y. Sun, W. Tan, L. Li, G. Lu, and A. Tang, "SLA detective control model for workflow composition of cloud services," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, 2013, pp. 165-171.

[56] E. Schmieders, A. Micsik, M. Oriol, K. Mahbub, and R. Kazhamiakin, "Combining SLA prediction and cross layer adaptation for preventing SLA violations," 2011.

[57] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann, "Runtime prediction of service level agreement violations for composite services," in *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, 2010, pp. 176-186.

[58] V. Cardellini, E. Casalicchio, F. Lo Presti, and L. Silvestri, "Sla-aware resource management for application service providers in the cloud," in *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, 2011, pp. 20-27.

[59] L. Wu, S. K. Garg, and R. Buyya, "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments," *Journal of Computer and System Sciences*, vol. 78, pp. 1280-1299, 2012.

[60] W. Hussain, F. K. Hussain, and O. K. Hussain, "SLA Management Framework to Avoid Violation in Cloud," in *International Conference on Neural Information Processing*, 2016, pp. 309-316.

[61] W. Hussain, F. K. Hussain, O. Hussain, and E. Chang, "Profile-based viable Service Level Agreement (SLA) Violation Prediction Model in the Cloud," presented at the 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, Poland, 2015.

[62] W. Hussain, F. K. Hussain, O. K. Hussain, and E. Chang, "Provider-Based Optimized Personalized Viable SLA (OPV-SLA) Framework to Prevent SLA Violation," *The Computer Journal*, vol. 59, pp. 1760-1783, 2016.

[63] W. Hussain, F. Hussain, and O. Hussain, "QoS prediction methods to avoid SLA violation in post-interaction time phase," in *Industrial Electronics and Applications (ICIEA), 2016 IEEE 11th Conference on*, 2016, pp. 32-37.

[64] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized QoS prediction framework for Web services," in *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on*, 2011, pp. 210-219.

[65] CloudClimate. *Watching the Cloud*. Available: http://www.cloudclimate.com/

[66] P. N. Monitor. Available: https://prtg.paessler.com/

[67] W. Hussain, F. K. Hussain, and O. Hussain, "Comparative analysis of consumer profile-based methods to predict SLA violation," presented at the FUZZ-IEEE, Istanbul Turkey, 2015.

[68] W. Hussain, F. K. Hussain, and O. Hussain, "Allocating Optimized Resources in the Cloud by a Viable SLA Model," presented at the Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on. IEEE, 2016, Vancouver, Canada 2016.

[69] W. Hussain, F. K. Hussain, and O. K. Hussain, "Risk Management Framework to Avoid SLA Violation in Cloud from a Provider’s Perspective," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2016, pp. 233-241.