Collaborative SLA and reputation-based trust management in cloud federations

Article *in* Future Generation Computer Systems · May 2019

DOI: 10.1016/j.future.2019.05.030

6 authors, including:

Konstantinos Papadakis-Vlachopapadopoulos
National Technical University of Athens
3 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Ioannis Dimolitsas
National Technical University of Athens
5 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Dimitrios Dechouniotis
National Technical University of Athens
28 PUBLICATIONS **152** CITATIONS
SEE PROFILE

Ana Juan Ferrer
Atos S.A.
29 PUBLICATIONS **551** CITATIONS
SEE PROFILE

Collaborative SLA and Reputation-based Trust Management in Cloud Federations

Konstantinos Papadakis-Vlachopapadopoulos[a,*], Román Sosa González[b], Ioannis Dimolitsas[a], Dimitrios Dechouniotis[a], Ana Juan Ferrer[c], Symeon Papavassiliou[a]

[a]*School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece*
[b]*Research & Innovation, Atos, Tenerife, Spain*
[c]*Reserch & Innovation, Atos, Barcelona, Spain*

Abstract

Industry and academia shift from the single cloud provider paradigm to cloud federations and alternative models, which orchestrate heterogeneous resources, such as Mobile Edge Computing and Fog Computing. In such complex environments, incorrect selection of deployment platform can lead to underwhelming application performance. This dictates the necessity of Service Level Agreements (SLA) and trust management services in order to enforce performance guarantees and enable customers to simultaneously evaluate their application's performance and give performance indicators for future provider selection. In this paper, we propose a collaborative SLA and Reputation-based Trust Management (RTM) solution for federated cloud environment. The SLA service defines clearly the performance metrics and measures the actual performance of the deployed cloud applications. Based on the SLA, the RTM service of the collaborative solution leverages several technical and user's experience metrics to compute the reliability of the cloud providers and the credibility of the customers. A proof of concept of the collaborative solution in a realistic federated

*Corresponding Author
Email address: cpapad@netmode.ntua.gr (Konstantinos Papadakis-Vlachopapadopoulos),roman.sosa@atos.net (Román Sosa González), jdimol@netmode.ntua.gr (Ioannis Dimolitsas), ddechou@netmode.ntua.gr (Dimitrios Dechouniotis), ana.juanf@atos.net (Ana Juan Ferrer), papavass@mail.ntua.gr (Symeon Papavassiliou)

environment is provided and validated. The corresponding experimental results demonstrate that it objectively computes the cloud providers' reputation values under various scenarios.
Keywords: Cloud applications, SLA, Trust Management, Federation
2010 MSC: 90B18, 03B52

1. Introduction

Cloud Computing is the current dominant paradigm for application delivery. However, cloud applications shift from the typical web services to mobile smart applications, which require the orchestration of several heterogeneous resources. Nowadays, the Mobile Edge Computing and Fog Computing are the emerging architectures, which combine the powerful cloud resources with wireless communication in the proximity of the end mobile user, for delivering time or mission critical applications. These architectures aim to enable applications based on Internet of Things (IoT), social media and industrial vertical systems. Furthermore, many applications are geographically dispersed and multi-cloud architecture or cloud federation are utilized, which involve the interaction between private and public clouds controlled by different providers.

The application life-cycle in cloud environment has several phases including authentication, resource discovery, booking, provisioning and application deployment, monitoring, management and retirement [1]. The utilization of heterogeneous resources poses complex requirements at each stage of the life-cycle. Towards this direction, many research initiatives, such as FED4FIRE+ [**?**] and CloudLab [2], provide realistic testbeds to industrial players and academia for developing innovative cloud applications and architectures. In order to provide realistic conditions, these initiatives provide the essential software tools for the management of the cloud application life-cycle. Two of the most important management services, which are involved in various phases of cloud application's life-cycle, are the Service Level Agreement (SLA) and trust management.

In federated cloud scenarios, a cloud acquires and/or offers spare capacity to

a set of providers in an environment. While the federation enables scalability, fault tolerance and elasticity to the cloud environment, SLA is the fundamental mechanism that allows users to enforce guarantees about performance and conformance of single or federated Cloud services [3]. Service Level Agreements specifically establish the consensus on the characteristics of the service to be provided between the service provider and the cloud service user. SLA terms have to be defined in a unambiguous manner that permits both the cloud users and providers to have a common understanding of the provided service. Diverse works have explored different term definitions for Cloud computing. Commercial Cloud offerings limit application of the SLAs to service availability. More complex proposals from richer SLA terms have been proposed in [3] considering terms with regards to Access, Trust and Security.

The second life-cycle management service discussed in this paper is trust management. Gambetta defines the trust as the subjective belief of entity A, that entity B performs a given action [4]. On the contrary, reputation is public and created by a group of people or entities. Concise Oxford Dictionary defines reputation as "the general belief about a person's or thing's character or standing". Similarly, in the cloud application life-cycle, trust management facilitates the resource selection and the performance evaluation of the cloud application. In a federated environment, many providers offer similar resources and applications and the selection of the appropriate ones is not a trivial task. Based on well-defined Quality of Service (QoS) and Quality of Experience (QoE) metrics, a trust management framework quantifies the confidence on the cloud application's performance and enable users to utilize those that fulfill their requirements.

In this paper, we present a collaborative SLA and reputation-based trust management solution for applications, deployed in cloud federations, that aims to address jointly the following challenges:

- The SLA offerings are defined based on cloud application's key performance indicators (KPIs) and provide reports on SLA violations.

- The customers evaluate periodically the performance of the applications leveraging various QoS and QoE KPIs, and the Reputation-based Trust Management (RTM) service utilizes these ratings in order to produce a reputation score for the cloud provider, which reflect his reliability to provide a service/resource. Consequently, the reputation score facilitates the selection of the appropriate services and resources by future customer according to their needs.

- A credibility mechanism is developed to protect the output of the RTM service from biased evaluations. It measures the deviation of the customer's rating from objective SLA and monitoring values, and its output is considered by the RTM service.

The rest of this paper is structured as follows; Section 2 discusses related work. Section 3 briefly highlights the cloud application life-cycle management and the collabotrative SLA-RTM architecture. Section 4 describes the details of the SLA service, while Section 5 presents the details of the introduced RTM service and the credibility mechanism. Section 6 contains a proof of concept of the designed SLA and RTM collaborative solution, deployed and tested under cloud federation and Mobile Edge Computing scenarios in a realistic federation of testbeds. Our conclusions and future work are drawn in Section 7.

2. Related Work

In this section, the most interesting approaches for trust and reputation management in cloud computing and web services, alongside with current SLA approaches, are presented.

Starting with the latter, the SLA current operational approaches in cloud providers are primarily limited to availability [5] (e.g. Amazon [6], Rackspace [7]). Beyond this, in research environments, SLAs and more broadly Service Level Management frameworks take often more complex forms in managing QoS in cloud distributed environments. They are mainly motivated by the fact

that the managed resources belong to different administrative domains. In these approaches, it is common that negotiation phase is implemented, since uttering expected QoS is not an acceptable possibility. Multiple projects [8], specifically addressing Cloud brokerage, have studied this problem from diverse perspectives. Cloud4SOA [9] project produced a framework facilitating dynamic SLA negotiation on multi-cloud Platform as a Service environments. Its SLA framework permits publication of offerings, as well as SLAs enactment for agreed QoS terms. Specifically, it considers business dynamics through business performance related SLA metrics. OPTIMIS [10] considered diverse deployment and run-time configuration scenarios. These included private, bursting, federated and multi-cloud deployments. OPTIMIS studied SLA negotiation and management in these scenarios. The QoS terms considered in OPTIMIS SLA's included operational Cloud capabilities. In addition to these, additional SLA terms were considered: service or provider's risk, trust, ecological or cost levels, as well as, legal requirements (related to personal data management) [11]. A significant number of these works are based on standardization efforts performed in Grid computing environments in WS-Agreement [12]. WS-Agreement is a full recommendation of the Open Grid Forum. WS-Agreement provides protocol and specific language in order to generate SLAs.

Other models besides WS-Agreement has been proposed. SLA@SOI[13] developed the SLA(T) model, that enables the description of both functional and non-functional characteristics of a service. The model allows providers to describe the offered services, and customers to describe their requirements and discover matching offers. Also, it allows multi-layered SLAs, which can be composed along functional and organizational domains. CONTRAIL[14] adopted the SLA(T) model and extended it to offer elastic PaaS services over a federation of IaaS clouds, while dealing with QoS and SLA management. The scenario considered in CONTRAIL is focused on cloud federations and automated generation of SLA offerings. In CONTRAIL, the user negotiates a SLA with the cloud federation, and the federation satisfies it by negotiating SLAs with the federated providers on behalf of the user. A different scenario was proposed in

MODAClouds [15], which considers that three actors take part in cloud SLAs; end users, Application Providers and Cloud Service Providers. Within this context, Application Providers lease resources from Cloud Service Providers to offer services to end users. Then, MODAClouds devises a two-level SLA system building up an aggregation of WS-Agreement SLAs. The first level describes the QoS to be offered by the Application Provider to the end users, incurring in penalties in case of SLA violations; this SLA only monitors for observable metrics by the end user (e.g., availability, response time). The second level describes the expected QoS from the Cloud Service Provider to the Application Provider for each of the resources, resulting in one agreement per VM; this second level is not an actual agreement, but just monitors the service offered by the Cloud Service Provider in order to be able to react and enforce the first level SLA.

With regards to security SLAs, SPECS[16] delivered an open source framework that provides SLA life-cycle, automatic negotiation and monitoring of security parameters specified in the SLAs. The proposed model is based on the WS-Agreement Standard. MUSA [17] presents a solution to SLA-based security assurance for multi-cloud applications, whose components are deployed in distributed cloud services. It enables the automatic creation of the offered Security SLA of the multi-cloud application, but it also enables to monitor at runtime the security service level objectives specified in the SLA. The proposed SLA composition adopts the SPECS cloud Security SLA model.

Several approaches were proposed for reputation and trust management in web services. In [18] a survey of trust and reputation systems for three types of web services is presented. These types are single, composite and communities web services. For single web services, a Bayesian network reputation and trust model is proposed in [19]. This model is based on user feedback, the recommendation of other users and the data that corresponds to the QoS, when a credibility mechanism evaluates the users' trustworthiness. In [20], a statistical approach was proposed to provide trust value to the parts of composite services. An online expectation maximization algorithm is used to assign trust to the individuals behind the service according to their contribution to the overall

performance.

Apropos of reputation and trust management in cloud computing, Yan et. al [21] proposed a data access control scheme based on individual trust and public reputation values. These values are used to apply attribute-based encryption and proxy re-encryption. Hatman [22] is a reputation-based trust management framework for Hadoop based clouds. It is based on EigenTrust algorithm [23], which assigned a global trust value to a user of a p2p network utilizing recursive method and their opinions, to improve the data integrity of distributed cloud computations. Zhu et. al [24] proposed an authenticated reputation and trust management system for integrated cloud and wireless sensor networks. This system focused on cloud and sensor providers' protection from malicious attacks, enabling users to select the proper providers based on their reputation and trust values. The trust value derived from the processing, data privacy and transmission capability of the cloud providers, while for sensor providers data collection, network lifetime, response time and data transmission metrics were used. The reputation score for both types of providers was computed leveraging SLA information about a service. Manuel [25] proposed a trust model for resource selection in heterogeneous cloud resources based on past credentials and present capabilities of a cloud resource provider focusing on four parameters. Those can be availability, reliability, turnaround efficiency and data integrity. The proposed solution also leveraged from combined usage of SLA and reputation. In [26] a new method for trust and reputation evaluation in the cloud environments leveraged the recommendations of opinion leaders' entities to remove the effect of troll entities'. The authors used a similar method with [25] for trust and reputation calculation using availability, reliability, data integrity, identity and capability as parameters but they used in addition three topological metrics. These included input-degree, output-degree and reputation to classify the evaluators as opinion-leaders and trolls. This social networks inspired approach, was used in order to remove the trolls opinion and duplicated the evaluation of the opinion leaders.

Trust and reputation management is a open research problem in feder-

ated clouds or testbeds. Regarding cloud federation, Hassan et. al [27] proposed a trust-based cooperative game-theory model for the most effective selection among federated cloud providers for data intensive applications aiming at maximum profits and minimum penalty cost's from SLA violations. In [28], a trust management framework for multi-cloud environments was presented where different and distributed Trust Service Providers, trusted from both Cloud Providers and Cloud Users, used data from the Service Level Agreements and the evaluations of the Cloud Service Users to differentiate trustworthy and untrustworthy Cloud Service Providers. The Trust Service Provider shared and obtained data about the Cloud Service Providers through a trust propagation network in the process.

Testbed federations provide heterogeneous resources for experimentation on several types of applications. In [29], authors proposed a reputation-based trust management framework that is based on various QoS and QoE performance metrics. Experimenters submitted their rating about QoS and QoE criteria and the reputation score of the involved testbeds was computed by a modification of fuzzy VIKOR algorithm [30].

Most of the previous approaches focused on a specific type of application deployed in the infrastructure of the single provider. However, cloud applications can be deployed in federated environment and involve heterogeneous resources. Compared with the above studies approaches, this paper proposes a complete collaborative SLA and Trust management framework, applicable particularly in cloud federations, which is involved in the most phases of the application's life-cycle management. The novelty of our approach is twofold. Regarding SLA management, this article proposes a WS-Agreement based framework, applied to a federation of service providers governed by a federated layer. This particular environment allows the providers not only to guarantee availability, but application-level metrics. In our understanding, we consider that this is very interesting due to the flexibility to the user they would provide, when applied to the real market. Secondly, the proposed reputation-based trust management facilitates the fair evaluation of the utilized services/resources from the customers

and the computation of the provider's reputation using multifaceted QoE and QoS KPIs.The computed reputation score and the SLA offerings reflect clearly the provider's reliability regarding the offered services/resources and facilitate the proper service/resource selection from future customers.

3. Cloud Application Life-Cycle and Collaborative SLA-RTM architecture

Cloud computing is composed of three service models; Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). With IaaS, a provider supplies the basic computing, storage and networking infrastructure along with the hyper-visor (the virtualization layer) and all the installation and configuration process depends on the users. With PaaS, a provider offers more of the application stack than IaaS providers, adding operating systems, middle-ware (such as databases) and other run-times into the cloud environment. With SaaS, a provider offers an entire application stack that the user simply consumes. In this paper, we focus mainly on PaaS, the cloud computing model in which a third-party provider delivers hardware and software tools, hosted by the provider on its own infrastructure and alleviates users from having to install in-house hardware and software to develop or run a new application. The biggest added value of PaaS is that developers are completely abstracted from the lower-level details of the environment, so they can fully focus on what there are really good at (rapid development and deployment) and not worry about things like scalability, security and more that are fully managed by PaaS.

3.1. Cloud Application Life-Cycle Management

The proposed collaborative SLA and RTM solution focuses on cloud federations. In such complex environments, the application's deployment is not a trivial task and needs careful resource selection, orchestration and performance monitoring. As it is shown in Figure 1, the complete life-cycle of cloud appli-

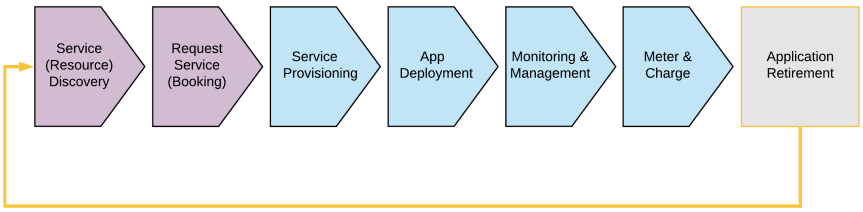


Figure 1: Cloud Application Life-Cycle in Federated Clouds

cations from the perspective of federated PaaS providers includes the following stages:

- **Service Discovery:** The cloud life-cycle starts with a user discovering the stack of the provided services in order to find the necessary services that satisfy his/her needs.
- **Request Service:** After the discovery phase, the user initiates the request for the corresponding services.
- **Service Provisioning:** Afterwards, the cloud provider allocates and assigns the resources to match user's usage demands.
- **Application Deployment:** Application is deployed by the user leveraging the necessary OS, applications and tools, which are provided.
- **Application Management:** Utilizing off-the-shelf tools, monitoring supervises the servers, the resources and the running software components. Another important management problem is the resource allocation of co-hosted cloud applications, which can be static or dynamic. In the latter case, this can be done by an automatic optimization tool or by a custom-made scheme.
- **Auditing and Billing:** Assessment tools report the resource usage (metering) and a periodic billing information is created.

- **Application Termination:** High utilization of resources is primary goal of cloud providers. This functionality enables the optimal resource reallocation after an application decommissioning, the pausing or the retirement of the application.

The SLA and RTM collaborative solution is involved in several of the above stages. Initially, both of them are required in service discovery. The SLA service advertises the offerings defined by the provider, while the RTM service provides a reputation value for every provider, enabling the proper selection of resources. Both RTM and SLA services are based on the same KPIs. During the application deployment, the SLA is activated and continuously evaluates the application performance (auditing stage). Also, the RTM service is involved in the auditing process, since it periodically prompts customers to submit their ratings about the application's status. Finally, at the final stage, the SLA terminates and the final rating is sent to RTM service.

3.2. Collaborative SLA and Reputation Architecture

In this subsection, we present the architecture of our collaborative SLA and RTM solution. Figure 2 demonstrates the high-level architecture of the SLA and RTM services in a cloud federation environment. The architecture is separated in two layers; namely the federation layer and the provider layer. Regarding the SLA service, the federation layer includes SLA Collector and Dashboard components, while the SLA Management Module lays in the provider layer.

SLA Dashboard offers a web-based graphical user interface (GUI), which enables customers to discover available SLA templates and providers to create the agreements. Furthermore, the SLA Dashboard allows providers and customers to check the status of the existing agreements.

SLA Collector acts as the intermediate communication point between the SLA Dashboard and the SLA Management module of each cloud provider. Through a REST API, this component supports every SLA process from creation to termination of a cloud application. Additionally, it provides a subscrip-

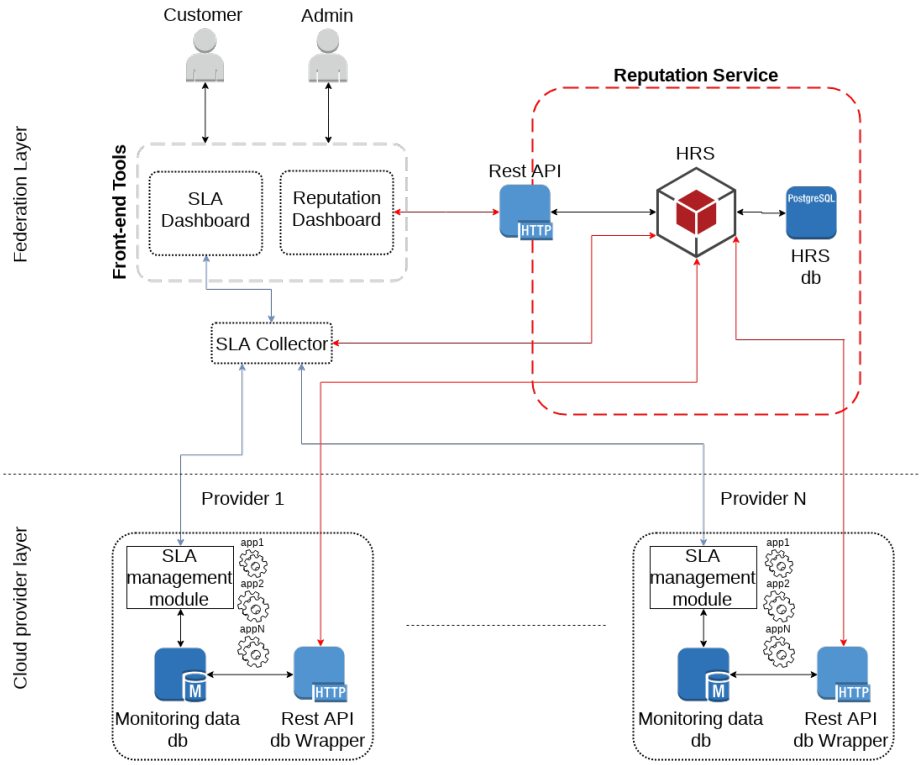


Figure 2: Architecture of SLA and RTM Service in cloud federation

tion mechanism, which allows other components to receive SLA events (e.g., SLA violations).

SLA Management Module, placed at the provider layer, is the core component for the SLA management process. It is responsible for storing all the SLA-related information and evaluating the active agreements, as described in 4.

As far as the RTM Service is concerned, the Reputation Dashboard offers a GUI to both customers and administrators. Through the Reputation Dashboard, the customers submit their rating and retrieve information about the reputation of each provider, while the administration of a cloud domain performs administrative tasks such as the definition of new KPIs for the computation of the reputation score. Through a REST API, the SLA Dashboard communicates

with the Hybrid Reputation System (HRS). HRS is the core component of the reputation service and is responsible for computing the reputation score of each provider. It retrieves SLA information and monitoring data through the SLA collector and a REST API respectively.

4. Service Level Agreement

Following the taxonomy of [3], we mainly focus on Access and Dependability aspects, which is the current practice for public cloud providers. Service (or Node) availability is defined as the degree of up-time for the service (or node) and expresses the Access perspective, while SLA penalties and violations of specific QoS metrics refer to Dependability notion. In this section, we describe the SLA life-cycle in a federated cloud environment.

4.1. SLA components

In this section we will present the components and functionality of the **SLA Management Module**, which is the core component for the SLA management process. It is responsible for storing all the SLA-related information and evaluating the active agreements. Figure 3 demonstrates the internal architecture of the SLA Management Module. Analytically, the most important components are:

- **Repository** is the database that stores any SLA entity, i.e., templates, agreements, violations and penalties.
- **REST service** is the REST interface of the SLA Management to external components. It provides CRUD (Create, Read, Update, Delete) operations to manage the SLA entities and change the state of an assessment.
- **Assessment** is the component responsible for the evaluation process of the SLA agreements. Utilizing monitoring data as input, it detects violations and generates penalties whenever a performance degradation occurs.

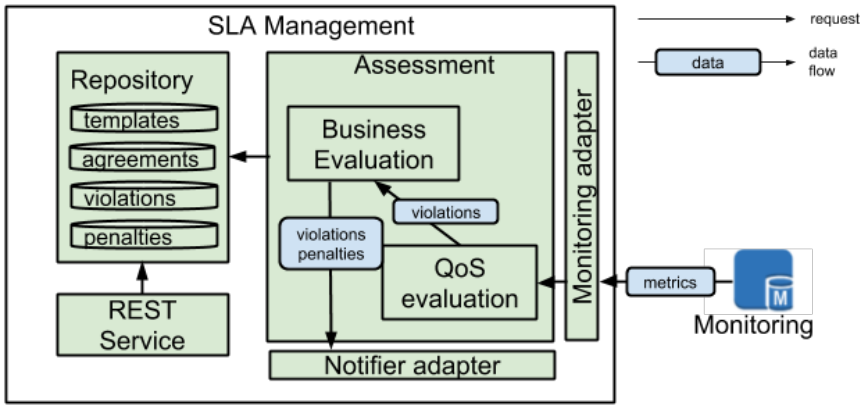


Figure 3: SLA Management Module

- **Monitoring adapter** makes queries to the monitoring database and parses the data for the the assessment of the active agreements.
- **Notifier adapter** is a customized component in charge of communicating events (e.g. violations and penalties) to interested entities through the SLA collector.

4.2. Overview of the SLA Life-cycle

Figure 4 illustrates an overview of the SLA life-cycle, which includes several phases. Initially, at the **service publishing** phase, the cloud providers publish their services by creating offerings for specific performance indicators, e.g. "99.99% of the deployed application's availability" or "the application response time is always lower than T ms". Furthermore, the penalties are described here. This information is formally described in an SLA template, a document structured as a WS-Agreement template [12]. The SLA templates are stored in the Service Registry. At the **service discovery** phase, the customers are able to discover the available offerings advertised in the Service Registry in a centralized way through the SLA Dashboard. The discovery can be as simple as a keywords

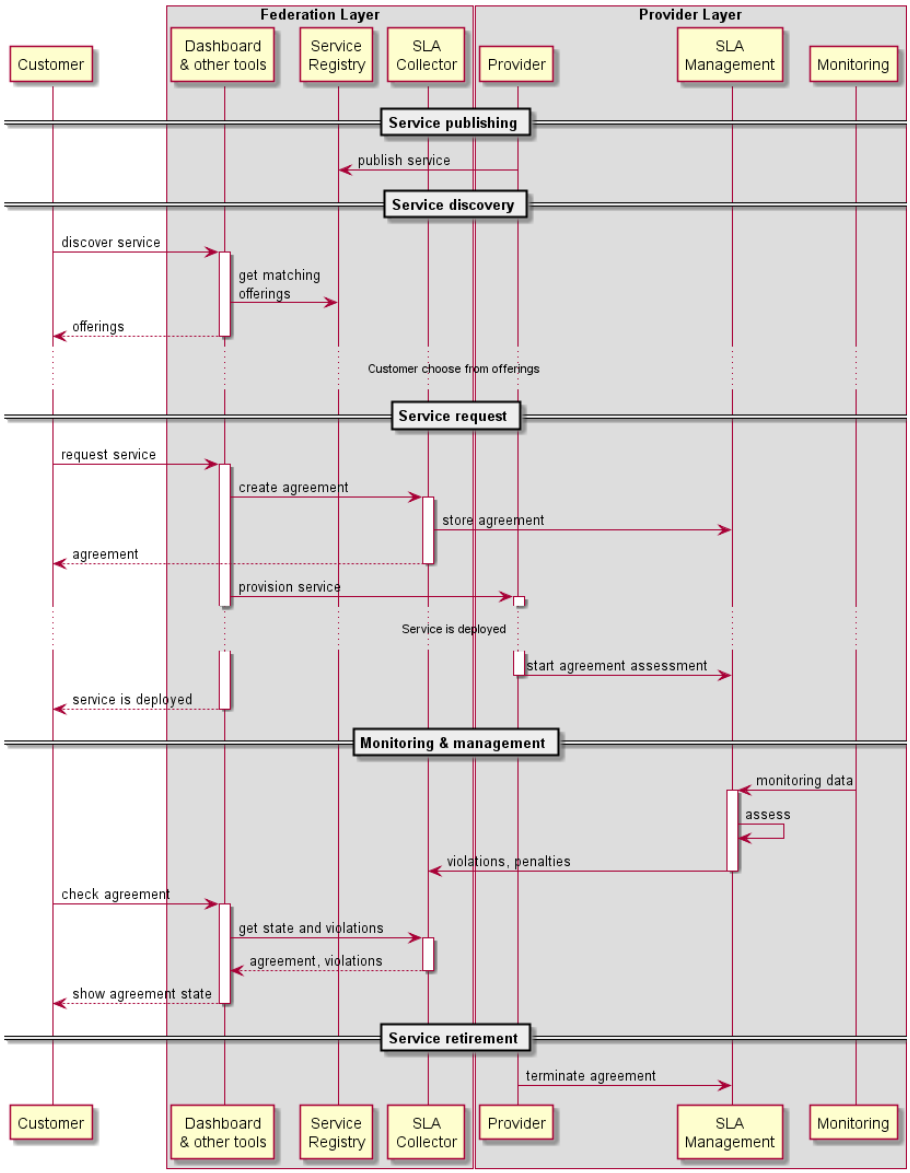


Figure 4: SLA Life-cycle sequence diagram

filtering, taking advantage of the extendable nature of the WS-Agreement standard to store the service keywords in the templates. Complex matchmaking mechanisms might be developed, but it is out of the article's scope. Next, at

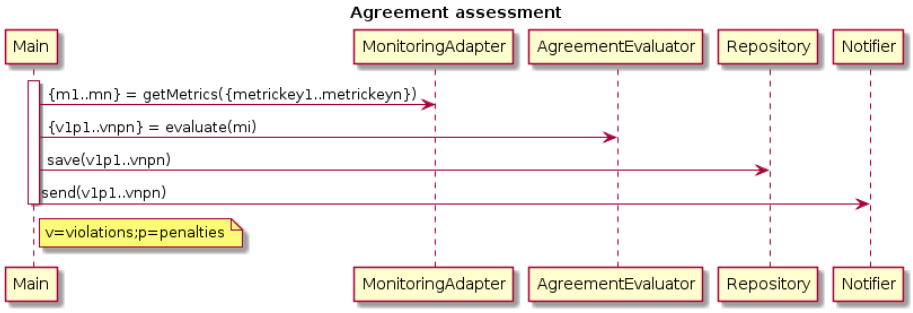


Figure 5: SLA assessment sequence diagram

the **service request** phase, the customer selects an offering and the corresponding agreement - based on the offering's SLA template - is created, containing the information of the customer, the provider, the performance indicators and the expected penalties. Sequentially, the SLA Collector forwards the agreement to the SLA Management module of the selected provider in order to be internally stored and evaluated until the cloud application's termination. After the application instantiation, the assessment of the agreement starts and the customer is notified. At the **monitoring and management phase**, the essential monitoring values for the agreement assessment are provided by the provider's monitoring service. The detected violations are stored internally and forwarded to the SLA Collector component, which includes a notification/subscription service to communicate violations to any interested entity, i.e. customer, or service such as the reputation service. Finally, the agreement is terminated with the service retirement.

4.3. SLA Assessment

The importance of SLA assessment is twofold. First, it provides a clear view on the cloud application's performance, while the produced violations are used to compute the provider's reputation score. A high-level sequence diagram of the SLA assessment is shown on Figure 5. Its main process is executed periodically (e.g., every minute), getting in first place the necessary metric values to evaluate each guarantee term from the MonitoringAdapter, which is in charge

of making the requests to the actual Monitoring component (not shown in the diagram) and transforming the data from the Monitoring domain to the SLA Management module domain, hence allowing the use of various monitoring tools in a transparent way. Then, the AgreementEvaluator checks if the retrieved values of each guarantee term satisfy the guarantee term constraint. In the case of an unsatisfied constraint, a violation is produced. Also, a penalty is produced only if it was defined by the agreement. The output of the AgreementEvaluator is the collection of generated violations and penalties, which are then stored in the internal Repository and sent to the Notifier, a plugin component intended to push notifications of the agreements status to interested observers. Depending on the nature of the cloud application, there is flexibility on the assessment's configuration and execution. The evaluation of a guarantee term can be chosen to be performed periodically (e.g per hour, per day, etc) or at the termination of a service, lease, etc. Additionally, the evaluation can be done using single values (e.g. response time for a particular timestamp) or by aggregated values (e.g. average availability of the service).

5. Hybrid Reputation System

In this section, the steps of the HRS for cloud application rating, evaluation and selection is presented in detail. The trustworthiness of a user is considered in the computation of reputation score and it is determined by a credibility mechanism, described in detail below. The proposed framework is actually a scalable multi-criteria decision-making system with hierarchical structure. In order to respond to the conditions of a federated cloud environment, this hybrid framework is able to process various types of data, which correspond to technical QoS and non-technical QoE KPIs of each provider. The technical KPIs refer to objective performance metrics, e.g, network latency and CPU utilization, while the non-technical KPIs correspond to subjective user's experience metrics such as Support Satisfaction and Usability. In a nutshell, this framework enables the use of numeric, binary and linguistic values, giving the customer the opportunity

to express his or her subjective opinion in the best possible and effective way. Each customer has unique needs and different criteria about service selection and overall performance. To accommodate this, the users through HRS can evaluate both cloud infrastructure and application performance considering different criteria, preferences and priorities. For this reason, in our approach, the customer himself/herself is capable of assigning different weights to the criteria according to his/her personalized service deployment goal.

The proposed reputation system is based on the principles of Fuzzy Analytic Hierarchical Process (FAHP) [31]. FAHP is widely used in various cases, such as product design and operational research [32]. FAHP is a ranking method based on numeric QoS and fuzzy QoE KPIs, such as node availability and support satisfaction respectively. However, in order to compute the reputation score of cloud applications, several modifications and extensions are required. There are three key differences with respect to FAHP between our HRS and the provider selection use cases, such as in [33]. First, our approach allows the customers to assign their weights on the criteria according to their application's performance criteria. Secondly, we compare the application's evaluation with an ideal rating of a virtual user, which contains the best values of all criteria. Finally, the computation of the reputation score takes into account the customer's credibility, as it is described in Section 5.2, in order to ensure the fair judgment of each cloud provider. In the following, the phases of the proposed HRS mechanism are described in detail.

5.1. Modified Fuzzy Analytical Hierarchical Process

5.1.1. Phase 1 - Selection of service KPIs

The cloud provider determines the technical (QoS) and the user experience (QoE) KPIs and attributes that are used in the computation of the reputation score of the provided application. Figure 6 shows an example of KPIs and attributes in a hierarchical structure and highlights which of them are provided by the cloud providers. The difference between KPIs and attributes is that a KPI measures a specific technical or experience metric, while an attribute

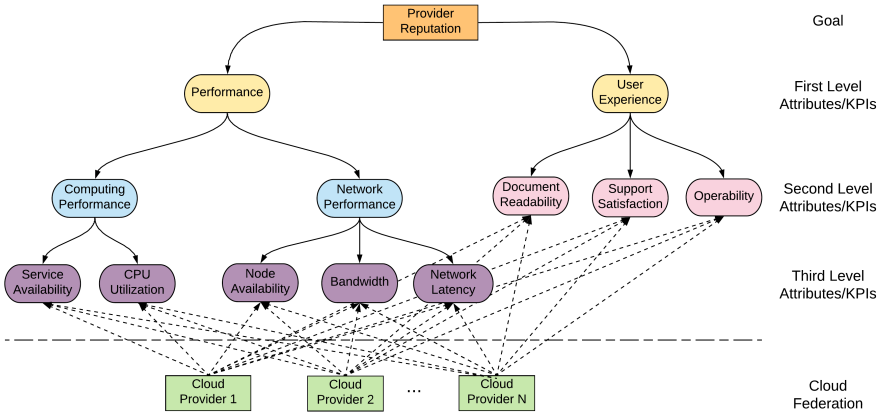


Figure 6: HRS Model for Federated Clouds

summarizes several KPIs of relevant metrics. At any level, the attributes can be further decomposed into the sibling attributes or KPIs of the lower level, while the KPIs cannot be decomposed further. Adopting SMICloud approach [34], numerical KPIs and attributes are represented by numeric, boolean, unordered sets and range values. On the contrary, the QoE KPIs are represented by fuzzy numbers. In Figure 6, pink (right) and purple (left) colored KPIs refer to fuzzy and numerical attributes respectively. In this study, we utilize triangular fuzzy numbers of the form $A = \{l, m, u\}$ and the membership function is defined as $\mu_A(x)$. Furthermore, the arithmetic operations on fuzzy numbers are defined as follows

$$A \oplus B = (l_A + l_B, m_A + m_B, u_A + u_B), \tag{1a}$$

$$A \ominus B = (l_A - u_B, m_A - m_B, u_A - l_B), \tag{1b}$$

$$A \otimes B = (l_A * l_B, m_A * m_B, u_A * u_B), \tag{1c}$$

$$A \oslash B = (l_A / u_B, m_A / m_B, u_A * l_B). \tag{1d}$$

Table 1 contains the linguistic terms and the membership functions of the fuzzy numbers used for QoE attributes.

Table 1: Linguistic Terms and Membership Functions of Fuzzy Numbers

Linguistic Term	Fuzzy Numbers
Very Poor (VP)	$(1, 2, 3)$
Poor (P)	$(3, 4, 5)$
Medium (M)	$(4, 5, 6)$
Good (G)	$(5, 6, 7)$
Very Good (VG)	$(6, 7, 8)$
Excellent (E)	$(7, 8, 9)$

5.1.2. Phase 2 - Computation of relative attribute importance

Periodically or after the termination of a cloud application, the customer submits his rating for the QoS and QoE KPIs of the selected providers. The ratings of QoS KPIs are modified by the credibility mechanism, as it is analyzed later. The modified customer's ratings are compared against the ideal rating of a virtual user. The ideal rating is used to measure the distance between the actual performance of a cloud application and its perfect performance according to the customer's preferences. This is achieved by computing the Relative Attribute Comparison Matrix (RACM) for each KPI of the hierarchical model. Given the ideal rating A_v and the modified customer's rating $\tilde{A}_u$ for the X KPI, $RACM_X$ is defined as follows,

$$RACM_X = \begin{bmatrix} 1 & \tilde{A}_u/A_v \\ A_v/\tilde{A}_u & 1 \end{bmatrix} \tag{2}$$

In the case of fuzzy KPIs, the division of $RACM_X$ elements corresponds to the fuzzy division of (1d). For numerical KPIs, the division follows the cases of Table 2, leveraging the values of the corresponding KPIs v_i, v_j and the size of the set v_r.

Table 2: Relative ranking model for the four types of numerical values [34]

Numeric KPI: $A_i/A_j = \begin{cases} v_i/v_j & \text{if higher is better} \\ v_j/v_i & \text{if lower is better} \\ w_q & \text{if } \nexists v_i \\ 1/w_q & \text{if } \nexists v_j \end{cases}$	Boolean KPI: $A_i/A_j = \begin{cases} 1 & \text{if } v_i = v_j \\ w_q & \text{if } v_i = 1 \wedge \\ & \quad \wedge v_j = 0 \\ 1/w_q & \text{if } v_i = 0 \wedge \\ & \quad \wedge v_j = 2 \end{cases}$
Unordered Set KPI: For essential attributes $A_i/A_j = \frac{size(v_i)}{size(v_j)}$ For non-essential attributes $A_i/A_j = \begin{cases} \frac{len(v_i \cap v_r)}{len(v_i \cap v_r)} & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \\ 1 & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ w_q & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ 1/w_q & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \end{cases}$	Range KPI: For essential attributes $A_i/A_j = \frac{len(v_i \cap v_r)}{len(v_i \cap v_r)}$ For non-essential attributes $A_i/A_j = \begin{cases} \frac{len(v_i \cap v_r)}{len(v_i \cap v_r)} & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \\ 1 & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ w_q & \text{if } v_i \cap v_r \neq \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \equiv \emptyset \\ 1/w_q & \text{if } v_i \cap v_r \equiv \emptyset \wedge \\ & \quad \wedge v_j \cap v_r \neq \emptyset \end{cases}$

5.1.3. Phase 3 - Computation and update of reputation score

In the case of numerical KPIs and attributes, the extended AHP approach is applied as described in SMICloud [34]. For the fuzzy KPIs, the extended analysis on FAHP is adopted according to Chang's approach [31]. The combination of these methodologies uses the RACM of each KPI and attribute at any level of the hierarchical model in order to calculate the score vector of all intermediate attributes and the top level reputation attribute. For the fuzzy RACMs, the

following steps of extent analysis on FAHP [31] are applied. Let the N-dimension fuzzy $RACM_A = [a_{ij}]\,,\ i,j = 1,\ldots,N$, the fuzzy synthetic extent of each row i of $RACM$ is defined by,

$$D_i = (D_{il}, D_{im}, D_{iu}) = \sum_{j=1}^{N} a_{ij} \otimes \left(\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \right)^{-1} \tag{3}$$

where the first term of the fuzzy multiplication is the sum of the elements of the the i^{th} row and the second terms is the fuzzy inverse of the sum of the RACM's elements. The fuzzy multiplication is defined by (1c), while the fuzzy inverse is defined using the fuzzy division of (1d). We find the attribute with the higher fuzzy synthetic degree by computing the degree of possibility for a fuzzy number to be greater than another one,

$$\begin{aligned} V\left(D_i \geq D_j\right) &= hgt\left(D_i \cap D_j\right) = \mu_{D_i}(d) \\ &= \begin{cases} 1 & \text{if } D_{im} \geq D_{jm} \\ \frac{D_{jl} - D_{iu}}{(D_{im} - D_{iu}) - (D_{jm} - D_{jl})} & \text{if } D_{im} \leq D_{jm} \text{ and } D_{jl} \leq D_{iu} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

The degree of possibility is a comparison method between two convex fuzzy numbers. It is defined by the ordinate d of the highest intersection point D, as it is shown in Figure 7. The degree of possibility that a fuzzy synthetic extent D_i is greater than the rest synthetic fuzzy extents of the fuzzy RACM is,

$$d_i = V\left(D_i \geq D_k, \forall k = 1,\ldots,N, k \neq i\right) = \min V\left(D_i \geq D_j\right) \tag{5}$$

Finally the normalized comparison vector is obtained,

$$c = [c_1 \ldots c_N]^\top \text{ where } c_i = \frac{d_i}{\sum_{k=1}^{N} d_k} \tag{6}$$

At any level of the cloud provider's hierarchical model, we calculate the comparison vector for each attribute with the following bottom-up procedure. Given the weights of Phase 2, the ratings of the customer and the ideal rating, we start from the level where KPIs exist, and compute the comparison vector of

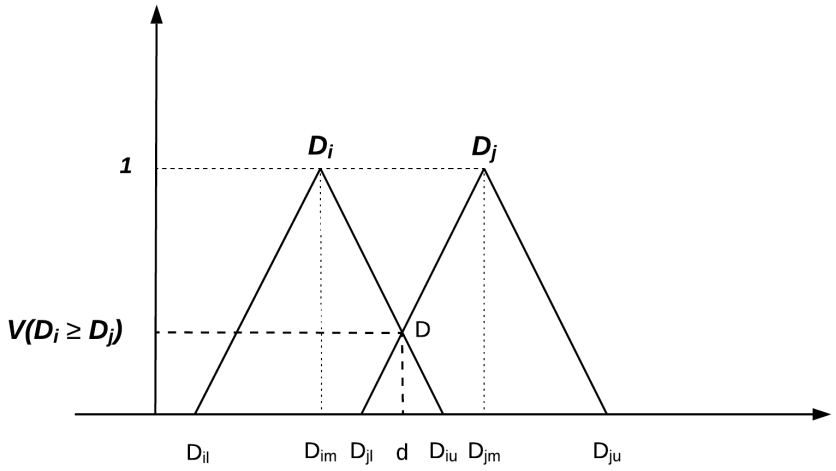


Figure 7: Graphical Presentation of the degree of possibility.

the parent attribute by the comparison vectors of the sibling KPIs or attributes. Assuming a parent attribute with M sub-attributes and the weight vector with M elements, the comparison vector of the parent attribute is defined,

$$c_{par} = \begin{bmatrix} c^{\tilde{u}}_{sub1} & \cdots & c^{\tilde{u}}_{subM} \\ c^{v}_{sub1} & \cdots & c^{v}_{subM} \end{bmatrix} \begin{bmatrix} w_{sub1} \\ \vdots \\ w_{subM} \end{bmatrix} = \begin{bmatrix} c^{\tilde{u}}_{par} \\ c^{v}_{par} \end{bmatrix} \tag{7}$$

Reaching the top level of the hierarchical model, the normalized comparison vector for the provider's Reputation attribute is computed, $c_{rep} = \left[c^{\tilde{u}}_{rep} c^{v}_{rep}\right]^{\top}$. The first element of this vector refers to the service evaluation, while the second corresponds to the best possible rating of the virtual user. The difference between the two elements indicates the distance between the actual performance as interpreted by the customer, and the perfect performance of the cloud service. Thus, for the n^{th} submitted rating, the cloud provider's reputation score

is computed by,

$$R_{exp}^{T} = \frac{c_{rep}^{\tilde{u}}}{c_{rep}^{v}} 100\% \tag{8}$$

After n customers' evaluations, the provider's reputation value is updated,

$$R_{n}^{T} = \frac{(n-1)R_{n-1}^{T} + R_{exp}^{T}}{n} \tag{9}$$

5.2. Credibility Mechanism

In our approach, the notion of credibility express the user's ability to provide objective evaluation for QoS KPIs. With this capacity, the credibility mechanism aims at reducing the impact of malicious users in the computation of reputation score. It takes into account the QoS KPIs and the respective SLA values. Essentially, the customer's subjective opinion, the predefined SLA and the monitoring values are compared in order to check the divergence between the user's rating and the cloud's actual performance. In this process the non-technical QoE KPIs are excluded due to their subjective nature.

Algorithm 1 shows the steps that define the user's credibility calculation process. The represented process concerns the use of one cloud provider. Nevertheless, in an real life scenario, customers have the opportunity to use more than one cloud providers. In that case the customer's credibility value is calculated utilizing the customer's ratings for all QoS KPIs of all providers. Considering the customer's opinion (ratings) for every QoS KPI, as the vector $UO = [UO_i]^{\top},\ i = 1, \ldots, k$, the monitoring data vector, $MD = [MD_i]^{\top},\ i = 1, \ldots, k$ and the SLA data vector, $SD = [SD_i]^{\top},\ i = 1, \ldots, k$, the algorithm updates the credibility value for the specific customer and a vector with the updated ratings for the QoS KPIs as those modified by the credibility mechanism, $\widetilde{UO} = [\widetilde{UO}_i]^{\top},\ i = 1, \ldots, k$ respectively. For each KPI of the cloud provider, the threshold and correction vectors are initialized (lines 4-5). The elements of the threshold vector express the tolerance against an opinion and is based on the deviation of the monitoring data from the SLA reference value (lines 6-13). The elements of the correction vector are actually the credibility values for each

Algorithm 1 Credibility Mechanism

Inputs: $UO,\ SD,\ MD$
Outputs: $CR,\ \widetilde{UO}$
for $\forall UO_i \in UO$ **do**
  $E = [e_i]^\top, e_i = 0.1,\ i = 1, \ldots, k$, Threshold Vector
  $C = [c_i]^\top,\ i = 1, \ldots, k$, Correction Vector
  if $|MD_i - SD_i| \geq e_i$ **then**
    $e_i = |MD_i - SD_i|$
  end if
  if $|MD_i - UO_i| \leq e_i$ **then**
    $c_i = 1$
  else
    $c_i = \frac{e_i}{|MD_i - UO_i|}$
  end if
end for
$\hat{c} = avg(c_i)$
$CR_n = \frac{(n-1)CR_{n-1} + \hat{c}}{n}$
for $\forall UO_i \in UO$ **do**
  if $|MD_i - UO_i| \geq e_i$ **then**
    if $UO_i < MD_i$ **then**
      $\widetilde{UO}_i = MD_i - e_i CR_n$
    else
      $\widetilde{UO}_i = MD_i + e_i CR_n$
    end if
  else
    $\widetilde{UO}_i = UO_i$
  end if
end for

KPI. The customer's credibility for an experiment is computed as the average value of the elements of the correction vector. Then, the overall customer's

credibility is updated according to lines 15-16. For each KPI, we adapt the customer's opinion if the difference between the opinion and the monitoring data is greater than the respective threshold value. The modified opinion is based on the monitoring value, the updated customer's credibility and the threshold value (lines 17-27). The modified opinions on KPIs are used in Phase 3 of HRS.

6. Proof of Concept, Experimentation and Evaluation

The collaborative SLA and RTM solution has been deployed, tested and evaluated in the FED4FIRE+ [3] platform. FED4FIRE+ initiative is the largest testbed federation in Europe, designed to facilitate experimentally driven research in the context of Future Internet Research and Experimentation (FIRE). Currently, the federation consists of sixteen core testbeds, offering wired, wireless, OpenFlow and cloud testbeds, recently extended to support big data experimentation. Furthermore, FED4FIRE+ is federated with other initiatives worldwide, i.e., GENI [35] and CloudLab [2]. The federation allows the experimenter to book and utilize resources from different testbeds at the same time in order to provide real life networking conditions for Future Internet experimentation, thus, it is suitable for evaluating the cloud applications' performance in federated environment. In this paper, for demonstration purposes, Virtual Wall [36] and NETMODE [37] testbeds - both being part of the FED4FIRE+ federation - are used in order to test, evaluate and validate the proposed SLA and RTM services. The Virtual Wall testbed offers cloud resources and its SLA template includes two offerings; Service Availability and Response Time, while the NETMODE wireless testbed defines Node Availability (the degree of up-time of a wireless node) as SLA performance indicator. Regarding the RTM service, the above SLA offerings are used as QoS metrics of both testbeds and they are combined with four QoE KPIs; namely Support Satisfaction, Documentation Readability, Usability and Operability, in order to compute the reputation score of each testbed - provider.

The collaborative SLA and RTM solution is validated through two illustra-

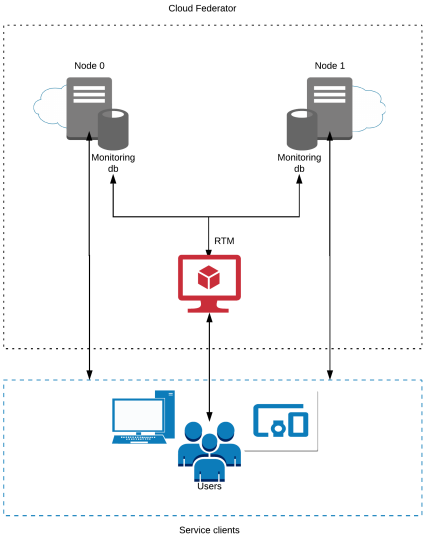


Figure 8: Scenario 1 Architecture

tive use cases. Scenario A architecture is depicted in Figure 8. Scenario A uses three physical nodes from Virtual Wall. Two of the physical nodes, named *Node0* and *Node1*, act as cloud providers, while the third one, *Node2*, hosts the RTM Service. Both *Node0* and *Node1* offer a dummy cloud application that just returns a 200 OK HTTP status code. The response time of the web server can be regulated for the purpose of the experiment. So, for *Node0*, it is configured to produce responses between 100ms and 400ms (stable node), while for *Node1* varies between 100ms and 650ms (unstable node). These values are selected for demonstrating the case of a stable cloud provider, which never breaches the SLA, and an unstable provider, which does breach the SLA, in order to validate the SLA mechanism and show the exploitation of SLA assessments by the RTM Service. In the Scenario A, the two SLA offerings guarantee: (i) the response time of the service should be lower than 500 ms, (ii) the service availability, measured as the average of availability values in ten minute intervals, should be higher than 90%. Finally, the Monitoring adapter obtains the values of the response time and the availability metrics, whose values are generated every ten

seconds and ten minutes respectively, for *Node0* and *Node1*.

Scenario B emulates the deployment of an application within a Mobile Edge Computing service delivery paradigm, which requires the orchestration of wireless and cloud resources, and illustrates the applicability and the efficiency of the proposed collaborative solution to any type of resources. The architecture of this scenario is illustrated in Figure 9. Three Raspberry Pi devices are connected on the wireless nodes of NETMODE testbed and generate requests of a cloud application. These requests are directed to a physical node of Virtual Wall testbed, which act as cloud provider and hosts the cloud application, which is identical to Scenario A and the same SLA offering is used. The cloud application is instantiated every half hour and 750 customers' ratings are submitted. In order to highlight the importance of the credibility mechanism, the 20% of these ratings are biased while the rest customers submits objective ratings.

6.1. Proof of Concept

6.1.1. SLA Validation

Scenario A provides an SLA validation where two federated providers offer identical services and guarantee terms to the customer. As mentioned before, *Node1* is parameterized to produce SLA violations, while *Node0* fulfills the performance requirements, according to the agreement defined above. The cloud application runs for one and half hour. Each provider has a monitoring process, which stores the measured values into a KairosDB database, and the SLA Management module retrieves these monitoring data using a specific KairosDB Adapter. The SLA assessment process runs every minute and detects possible violations on the response time and service availability in different time scales. Regarding the first KPI, six values of response time are used in each assessment process, since it is polled every ten seconds. The monitoring value of the service availability is updated every ten minutes, thus the assessment is executed accordingly. Figures 10a and 10b show the response times and the corresponding violations for both cloud providers. As it was expected, there are no violations for the application running on *Node0*, while several violations are generated

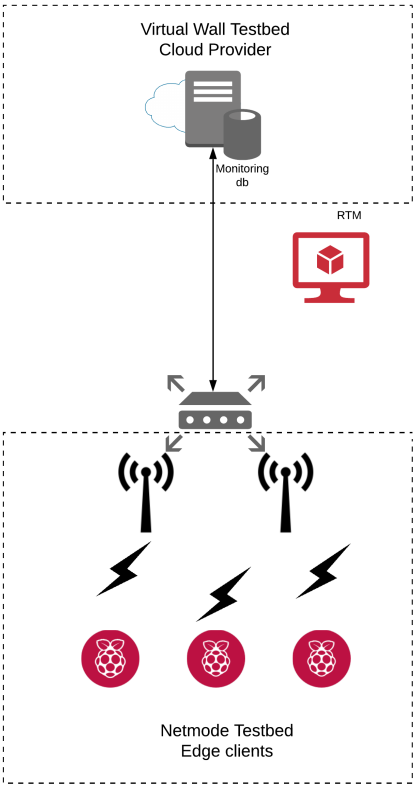
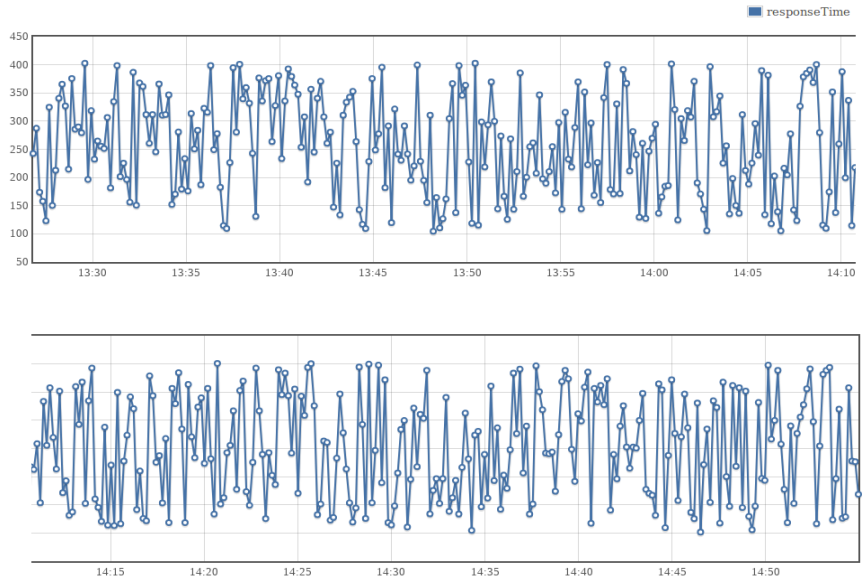


Figure 9: Scenario 2 Architecture

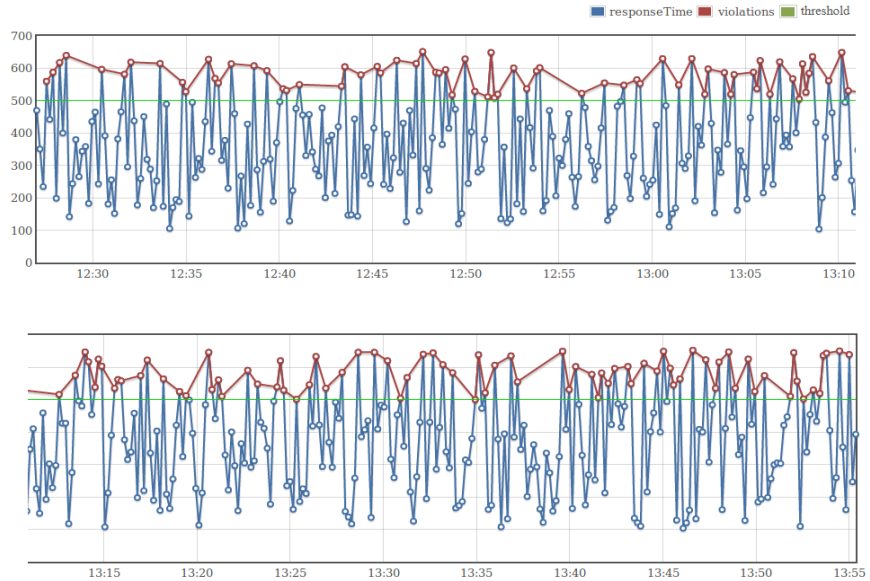
for *Node1*. On Figure 10b, the threshold value of the agreement is marked with green colour, while the red line highlights the generated response time violations. The created violations are forwarded to the cloud provider and the customer. Also, they are used by the HRS for the computation of the reputation score.

6.1.2. Reputation Example

In this section, we demonstrate a simple example of the behaviour of the reputation algorithm to biased evaluations. After the end of the experiment described above, we compute the reputation score of the two nodes-providers. As described in the section above, *Node1* violates the agreed standards, contrary to *Node0*, which fulfills completely the agreement. Since the inputs of the cred-



(a) Node0: SLA Assessment



(b) Node1: SLA Assessment

Figure 10: Application's Response Time

Table 3: Evaluation data

	ratings	previous reputation	evaluation	monitoring data	modified opinion	updated reputation
Node0	7	82.1432	65	100	90	83.3227
Node1	6	74.5615	55	72.7777	55	73.4554

ibility mechanism are normalized values, for the response time KPI, we utilize the SLA violations to define an alternative KPI, which actually measures the fragment of time interval where no violation occurs, with the following formula,

$$rt = (1 - \frac{violations}{samples}) \cdot 100$$

As it shown in Table 3, we assume that the submitted rating is the seventh and sixth sequential experiment for *Node0* and *Node1* respectively, while the respective reputation score is 82.1432 and 74.5615. In this scenario, the customer is not satisfied with the overall application's performance, thus, he submits poor evaluations, 65 and 55 respectively, for both nodes. The rating of *Node0* is unfair while the evaluation of *Node1* is closer to its actual performance. Leveraging the monitoring data, the credibility mechanism modifies the customer's opinion, thus the rating of *Node1* has changed to 90, while the rating of the second node remains the same. This implies that the credibility mechanism fairly mitigates the effectiveness of misleading ratings on the reputation score of a provider. For the service availability KPI, for both nodes, the ratings and the monitoring values are 100% and are not modified by the credibility mechanism. Thus, they are omitted from the Table 3. Additionally both nodes receive the "VERY GOOD" linguistic value for all the QoE KPIs.

Following the phases of Section 5.1, we calculate the updated reputation score of both nodes. We demonstrate the step by step calculation of the proposed method, for *Node0*. Table 4 shows the structure of KPIs in the hierarchical model of HRS. The user assigned weights are written next to every attribute and KPI. The third column contains the experimenter's evaluation, while the ideal rating of the virtual user lies in the last column. As described in Phase 2 and 3 of subsection 5.1, the bottom-up procedure with the intermediate computations

Table 4: Ratings for Node0

Top Level	First Level	Second Level	Experimenter	Virtual User
Reputation	Computing Performance (0.8)	Service Availability (0.5)	1	1
		Response Time (0.5)	0.65	1
	User Experience (0.25)	Support Satisfaction (0.25)	[VG]	[E]
		Operability (0.2)	[VG]	[E]
		Usability (0.25)	[VG]	[E]
		Document Readability (0.25)	[VG]	[E]

will be presented briefly. For example, the fuzzy RACM of the Support Satisfaction KPI is computed using the user's value $\tilde{A}_u = (6, 7, 8)$ and the virtual user's rating $\tilde{A}_v = (7, 8, 9)$,

$$RACM_{SupSat} = \begin{bmatrix} (1,1,1) & (0.67,\ 0.87, 1.14) \\ (0.87,\ 1.14, 1.5) & (1,1,1) \end{bmatrix}$$

Then, the fuzzy synthetic extent is computed for the virtual and the actual user is computed according to $RACM_{SupSat}$ and (3),

$$D_1 = (0.36, 0.47, 0.60)$$
$$D_2 = (0.40, 0.53, 0.71)$$

Following the procedure in Phase 3 and using (4) and (5), we get the degree of possibility for the experimenter and the virtual user, $d_1 = 0.75$ and $d_2 = 1$ respectively. Finally, the normalized comparison vector is obtained (6),

$$c_{SupSat} = [\ 0.43\ 0.57]^\top$$

Similarly, for the rest of the QoE KPIs, we calculate the following comparison vectors are computed,

$$c_{Oper} = [\ 0.43\ 0.57]^\top$$
$$c_{Usab} = [\ 0.43\ 0.57]^\top$$
$$c_{DocRead} = [\ 0.43\ 0.57]^\top$$

Then, the comparison vector for the User Experience attribute is obtained,

$$c_{UsExp} = \begin{bmatrix} 0.43 & 0.43 & 0.43 & 0.43 \\ 0.57 & 0.57 & 0.57 & 0.57 \end{bmatrix} \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.429 \\ 0.571 \end{bmatrix}$$

For the QoS KPI Service Availability the comparison vector is,

$$c_{Avail} = [\ 0.5\ 0.5]^{\top},$$

as the rating is equal to the perfect evaluation of 1 (100%). The QoS KPI Response Time is modified according to the user's credibility. So the modified rating for Response Time is 0.9. The comparison vector of Response time is,

$$c_{RespTime} = [\ 0.474\ 0.526]^{\top},$$

Following the same procedure with the QoE KPIs, we compute the comparison vector for the Computing Performance attribute,

$$c_{CPerf} = \begin{bmatrix} 0.5 & 0.474 \\ 0.5 & 0.526 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.428 \\ 0.512 \end{bmatrix}$$

At the top level, the comparison vector for the Reputation attribute is,

$$c_{Reput} = \begin{bmatrix} 0.428 & 0.429 \\ 0.512 & 0.571 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0.475 \\ 0.525 \end{bmatrix}$$

Then, the reputation score for this experiment is computed by (8),

$$R^T_{exp} = \frac{c^{\tilde{u}}_{Rep}}{c^{v}_{Rep}} 100\% = 90.4\%$$

So, as that was the seventh evaluated experiment, the total Reputation Score for the Node0 is updated according to (9),

$$R^T_7 = \frac{6 * R^T_6 + R^T_{exp}}{7} = 83.32\%$$

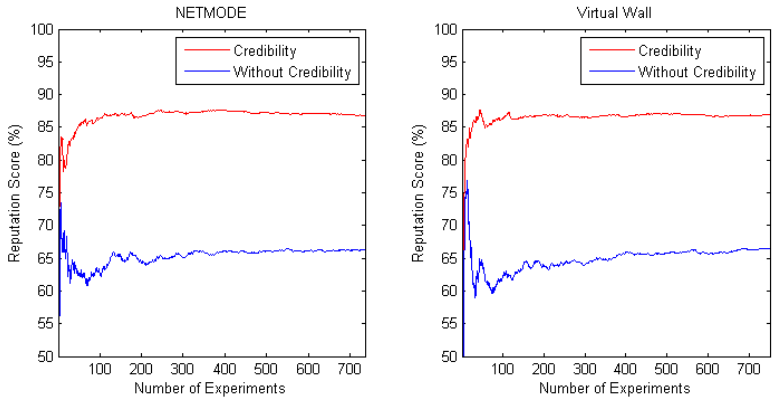


Figure 11: Credibility Mechanism's Effect in HRS

6.2. Effect of Credibility Mechanism

As it is mentioned above, scenario B illustrates the effectiveness of the RTM service on federated heterogeneous resources and focuses on the effect of credibility mechanism on the computation of the reputation score. For both testbeds and 750 ratings, the reputation score is computed for two different cases. As depicted in Figure 11, in the first one the credibility mechanism has been taken into account (red line), while in the second (blue line) the update of the reputation values is performed without the use of the credibility mechanism. For both cases, the output of HRS is drawn in Figure 11. As it is shown, for both testbeds, the activation of the credibility mechanism leads to 20% increase of the reputation score. The credibility mechanism leverages SLA and monitoring data and compares them with the subjective user's opinion. If a significant deviation is observed, the user's opinion is modified and the credibility value is decreased. Thus the influence of the malicious evaluation on the reputation score is minimal. On the other hand, when the reputation value is updated without the use of the credibility mechanism, malicious users and evaluations seem to significantly affect the HRS performance. In that case, the reputation score is generally lower and rapid fluctuations are observed, as shown in Figure 11.

7. Conclusions

This paper presents a collaborative SLA and trust management platform for federated cloud provider. The SLA service enables cloud providers to describe with specific indicators the performance of the deployed cloud application and provide the necessary tools for the assessment of the agreement. In case of performance degradation, SLA violations are generated and the interested entities are notified. Complementary, a reputation-based trust management service is developed in order to fairly depict the provider's reliability to provide specific resources and services. This service is based on both QoS and KPIs, and is suitable for federated environments since it scales easily. Alongside, a credibility mechanism leverages SLA and monitoring data to mitigate the effect of biased customers' evaluations. Apart from the evaluation of the cloud application performance, the RTM service facilitates to the future customers to select the appropriate providers and resources for deploying their applications. The experimental results demonstrate that the RTM service's output fairly represent the provider's reputation. Also, it was shown that the utilization of the credibility mechanism improves 20% the performance of the RTM service.

Our future plans contain the enrichment of the SLA service with an automatic negotiation function, which will allow customers and providers to customize the available SLA templates, while creating more fine-grained agreements. Finally, the reputation service could be further combined with a recommendation system in order to better promote the provider's resources to potential clients.

References

[1] K. Potvin, A. Akela, G. Atil, B. Curtis, A. Gorbachev, N. Litchfield, L. Nelson, P. Sharman, Cloud lifecycle management, in: Expert Oracle Enterprise Manager 12c, Springer, 2013, pp. 153–186. doi:10.1007/978-1-4302-4939-9_5.

[2] CLoudLab Project, https://www.cloudlab.us/#top.

[3] A. J. Ferrer, E. P. i Montanera, The role of SLAs in building a trusted cloud for Europe, in: IFIP International Conference on Trust Management, Springer, 2015, pp. 262–275. doi:10.1007/978-3-319-18491-3_22.

[4] D. Gambetta, Can we Trust, Trust? Trust: Making and breaking cooperative relationships, Ph.D. thesis, Ed, D. Gambetta, Oxford: Blackwell: 213-237 (1988).

[5] SLALOM project, http://slalom-project.eu/.

[6] Amazon Compute Service Level Agreement, https://aws.amazon.com/compute/sla/.

[7] Rackspace Cloud Service Level Agreement, https://www.rackspace.com/information/legal/cloud/sla.

[8] D. Kyriazis, Cloud computing service level agreements–exploitation of research results, European Commission Directorate General Communications Networks Content and Technology Unit, Tech. Rep 5 (2013) 29.

[9] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. DAndria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, et al., Cloud4SOA: A semantic-interoperability PaaS solution for multi-cloud platform management and portability, in: European Conference on Service-Oriented and Cloud Computing, Springer, 2013, pp. 64–78.

[10] A. J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, et al., OPTIMIS: A holistic approach to cloud service provisioning, Future Generation Computer Systems 28 (1) (2012) 66–77.

[11] W. Ziegler, M. Jiang, K. Konstanteli, OPTIMIS SLA framework and term languages for SLAs in cloud environment, OPTIMIS Project Deliverable 2.

[12] Open Grid Forum (OGF), Web Services Agreement Specification (WS-Agreement), http://ogf.org/documents/GFD.192.pdf.

[13] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, R. Yahyapour, Establishing and monitoring slas in complex service based systems, in: 2009 IEEE International Conference on Web Services, 2009, pp. 783–790. doi:10.1109/ICWS.2009.47.

[14] R. Cascella, L. Blasi, Y. Jegou, M. Coppola, C. Morin, Contrail: Distributed application deployment under sla in federated heterogeneous clouds, in: The Future Internet, 2013, pp. 91–103. doi:10.1007/978-3-642-38082-2_8.

[15] D. Ardagna, M. Ciavotta, G. P. Gibilisco, R. B. Desantis, G. Casale, J. F. Pérez, F. D'Andria, R. Sosa González, QoS Assessment and SLA Management, Springer International Publishing, Cham, 2017, Ch. 4, pp. 35–46. doi:10.1007/978-3-319-46031-4_4.
URL https://doi.org/10.1007/978-3-319-46031-4_4

[16] V. Casola, A. D. Benedictis, M. Rak, U. Villano, Sla-based secure cloud application development: The specs framework, in: 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2015, pp. 337–344. doi:10.1109/SYNASC.2015.59.

[17] E. Rios, M. Rak, E. Iturbe, W. Mallouli, et al., Sla-based continuous security assurance in multi-cloud devops.

[18] O. A. Wahab, J. Bentahar, H. Otrok, A. Mourad, A survey on trust and reputation models for web services: Single, composite, and communities, Decision Support Systems 74 (2015) 121–134. doi:10.1016/j.dss.2015.04.009.

[19] H. T. Nguyen, W. Zhao, J. Yang, A trust and reputation model based on bayesian network for web services, in: Web Services (ICWS), 2010 IEEE International Conference on, IEEE, 2010, pp. 251–258. doi:10.1109/ICWS.2010.36.

[20] C.-W. Hang, A. K. Kalia, M. P. Singh, Behind the curtain: Service selection via trust in composite services, in: 2012 IEEE 19th International Conference on Web Services, IEEE, 2012, pp. 9–16. doi:10.1109/ICWS.2012.96.

[21] Z. Yan, X. Li, M. Wang, A. V. Vasilakos, Flexible data access control based on trust and reputation in cloud computing, IEEE Transactions on Cloud Computing 5 (3) (2017) 485–498. doi:10.1109/TCC.2015.2469662.

[22] S. M. Khan, K. W. Hamlen, Hatman: Intra-cloud trust management for Hadoop, in: 2012 IEEE Fifth International Conference on Cloud Computing, IEEE, 2012, pp. 494–501. doi:10.1109/CLOUD.2012.64.

[23] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: Proceedings of the 12th international conference on World Wide Web, ACM, 2003, pp. 640–651. doi:10.1145/775152.775242.

[24] C. Zhu, H. Nicanfar, V. C. Leung, L. T. Yang, An authenticated and reputation calculation and management system for cloud and sensor networks integration, IEEE Transactions on Information Forensics and Security 10 (1) (2015) 118–131. doi:10.1109/TIFS.2014.2364679.

[25] P. Manuel, A trust model of cloud computing based on quality of service, Annals of Operations Research 233 (1) (2015) 281–292. doi:10.1007/s10479-013-1380-x.

[26] M. Chiregi, N. J. Navimipour, A new method for trust and reputation evaluation in the cloud environments using the recommendations of opinion leaders' entities and removing the effect of troll entities, Computers in Human Behavior 60 (2016) 280–292. doi:10.1016/j.chb.2016.02.029.

[27] M. M. Hassan, M. Abdullah-Al-Wadud, A. Almogren, S. M. M. Rahman, A. Alelaiwi, A. Alamri, M. A. Hamid, QoS and trust-aware coalition formation game in data-intensive cloud federations, Concurrency and

Computation: Practice and Experience 28 (10) (2016) 2889–2905. doi:10.1002/cpe.3543.

[28] W. Fan, H. Perros, A novel trust management framework for multi-cloud environments based on trust service providers, Knowledge-Based Systems 70 (2014) 392–406. doi:10.1016/j.knosys.2014.07.018.

[29] D. Dechouniotis, I. Dimolitsas, K. Papadakis-Vlachopapadopoulos, S. Papavassiliou, Fuzzy Multi-Criteria Based Trust Management in Heterogeneous Federated Future Internet Testbeds, Future Internet 10 (7) (2018) 58. doi:10.3390/fi10070058.

[30] T. Kaya, C. Kahraman, Multicriteria renewable energy planning using an integrated fuzzy VIKOR & AHP methodology: The case of Istanbul, Energy 35 (6) (2010) 2517–2527. doi:{}10.1016/j.energy.2010.02.051.

[31] D. Chang, Applications of the extent analysis method on fuzzy AHP, European Journal of operational research 95 (1996) 649–655. doi:10.1016/0377-2217(95)00300-2.

[32] W. D. A. V. S. Kubler, J. Robert, Y. L. Traon, A state-of the-art survey & testbed of fuzzy ahp (fahp) applications, Elsevier Expert Systems with Applications 65 (2016) 398–422. doi:10.1016/j.eswa.2016.08.064.

[33] I. Patiniotakis, S. Rizou, Y. Verginadis, G. Mentzas, Managing imprecise criteria in cloud service ranking with a fuzzy multi-criteria decision making method, in: K.-K. Lau, W. Lamersdorf, E. Pimentel (Eds.), Service-Oriented and Cloud Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 34–48. doi:10.1007/978-3-642-40651-5_4.

[34] S. K. Garg, S. Versteeg, R. Buyya, Smicloud: A framework for comparing and ranking cloud services, in: Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, IEEE, 2011, pp. 210–218. doi:10.1109/UCC.2011.36.

[35] Geni, http://www.geni.net/.

[36] Virtual Wall - FED4FIRE+, https://www.fed4fire.eu/testbeds/virtual-wall/.

[37] NETMODE - FED4FIRE+, https://www.fed4fire.eu/testbeds/netmode/.