Policy Based Context Aware Service Level Agreement (SLA) Management in the Cloud

Mhammed Chraibi
Abdelilah Maach
Dept of Computer Engineering
Ecole Mohammadia d'Ingénieurs
Universite Mohammed 5, Rabat, Morocco
e-mail: M.Chraibi@aui.ma

Souhail Meftah
Hamid Harroud
School of Science and Engineering
Al Akhawayn University in Ifrane
Ifrane, Morocco
e-mail: S.Meftah@aui.ma

***Abstract*—The lack of security and the inexistence of Quality of Service tracking mechanisms limit the success of cloud computing as a new technology, even if it demonstrates great capabilities of solving a number of problems that almost all organizations suffer from. This paper presents a novel way of expressing Service Level Agreements (SLAs) and tracking them to assure the client about the security and Quality of Service that are provided by the Cloud Service Provider. Allowing the client to combine specific security and Quality of Service metrics with context information within SLAs, when they are expressed as software policies, increases tremendously their expressiveness and precision.**

Keywords—Security; Quality of Service; Service level Agreement; software policies;

I. INTRODUCTION

Cloud computing as a technology is changing the way Information Technology (IT) is seen by private, public, and independent organizations. None of them can survive in today's environment without heavily relying on IT. Therefore, all of them are looking for innovative ways to have their data and applications run. The quality of IT management might give them the edge that they need over the competition. Cloud computing, with the advantages it brings is, for many organizations, the most viable alternative. Having their data and applications managed by experts guaranteeing security and Quality of Service (QoS) on a pay per use basis is an incredible opportunity. Unfortunately, the fear of losing control of data and information that is not hosted locally anymore is stopping the organizations from migrating their IT to the cloud.

Our research group firmly believes that, if organizations were provided with the means to express their security and QoS needs and were able to track how well the cloud service provider is doing in taking care of those needs, they would be more willing to migrate to the cloud. The research we present in this paper consists of proposing software policies as a way to represent and manage Service Level Agreements (SLAs). The SLAs are the contract between the client and the Cloud Service Provider (CSP). The SLAs must allow the clients to express in terms of metrics what QoS and what security mean for them. In addition to that, software policies, the way we designed them, allow the integration of context information. Context awareness does not only increase the expressiveness of SLAs, but it also allows them to tackle any metrics identified to increase security and Quality of Service.

In this paper, we start by describing what we mean by context information within the cloud environment. Then, in Section III, we present our policy based, context aware Service Level Agreements design. In Section IV, we explain the reasons why we opted for a middleware to incorporate the management of SLAs in the cloud. Then, in Section V, we discuss the testing of the prototype that we built as a proof of concept. Finally, in the conclusion, we describe the future work.

II. CONTEXT AWARENESS IN CLOUD COMPUTING

The definition of the context of an application within the cloud is an exercise that has not been done by many researchers. The reason behind this is the fact that every application within the cloud has a different role and a different context. When the cloud is used for offloading, the context of the processing done in it relates to the domain where the application operates. In this section, we will see how context information is used to improve QoS in terms of efficiency, performance, or security.

A. Context information for performance improvement

Mobile Cloud Computing, for example, is currently a research hotspot. Scientists are looking into ways to allow mobile applications run their processing and data analysis in the cloud. They are aware that one of the major challenges facing them, the moment they consider offloading as an alternative, is the performance of the network. CloudAware is a context aware framework that contains a context manager entity responsible for collecting and analyzing network data to predict the status of the network at a point in time and make sure that the processing and communication of the data is done in a reasonable amount of time [1]. The context manager, once it has received the data from network sensors and other tools, performs a set of intelligent data mining operations in order to predict the future situation of the network [1]. This leads to the improvement of the overall network performance.

B. Context information for security management

Context aware role based access control is the solution described by [2]. The use of context information in order to provide a personalized service and "dynamic adaptation" of access control requires collecting continuously context data. To move into useful information, a few steps are described, such as:

- Context pre-processing
- Context analyzing
- Context providing

The huge amount of data received from the different context acquiring tools (both hardware and software) is considered big data. Therefore, intelligent, machine learning

 ISBN: 978-1-61208-529-6

algorithms are utilized by [2] in order to filter and have readily available context information. Such quality context information allows for state of the art role based access control without adding a big load to the overall system performance.

C. Multiplicity of cloud context providers

Within a cloud environment, any context aware platform, middleware, or application needs to be able to handle incoming context data from a wide variety of sources. Depending on the nature of the source, different processing of the data may be done to be transformed into information that will be used in order to perform the appropriate actions. In [3], the authors raise the point that heterogeneity of incoming context data must be handled and they review the literature and analyze the way it is done. In the framework that they present, named MobiLife, the context data is all received by an entity called the context provider. That entity has three main responsibilities:

- It receives data from the different context sources, analyzes it, checks the ontology being used
- It advertises the availability of the context information
- It responds to requests of context information

Such a central entity is necessary because of the different sources of context data. They also mention how new needs of modeling of context data are there since context aware applications do not rely anymore on location only as it was the case previously.

III. Context Aware Service Level Agreements for Security and Quality of Service

SLAs are the contract that exists between the cloud service provider and the cloud application. The accuracy of the SLA is the key to a healthy relationship between those entities. Therefore, there is a need for a way that allows the detailed expression of SLAs and monitoring to know if the SLAs are being respected or not. In this section, we present our policy specification language that is used to represent SLAs. Then, we go deeper into the metrics that the policy specification language must allow the SLAs to express. Finally, we show examples of SLAs and how they are represented.

A. SLA Specification Language

In a previous work, our team developed a policy specification language that allows the expression of policies to manage security in quickly changing environments [4]. Figure 1 shows the structure of the Service Level Agreement using our policy specification language.

The first Attribute of the SLA is the ID. It represents a unique identifier to each Service Level Agreement. It allows the tracking and modification of SLAs. It is the only attribute that is not assigned a value by the client. Second, every SLA has a Subject. It is the entity responsible for enforcing the policy's action. Usually, the subject is a Policy Enforcement Point (PEP) that wraps the client application. More details about the PEP are given in Section IV.

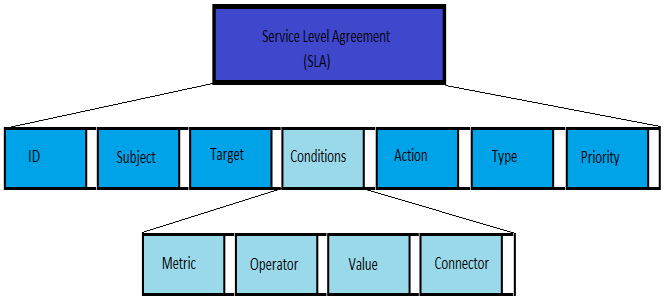

Figure 1. SLA policy based structure

The next attribute is the Target. It is the entity on which the action defined by the SLA is executed. Obviously, every Service Level Agreement contains the action itself that is triggered in case the conditions are met, and a priority that allows conflict resolution between policies. Finally, every SLA has a type. Obligation SLAs are triggered when a change in the context happens and a notification is generated. On the other hand, Authorization policies are triggered when a request, from the client application, is received asking to verify if an SLA is being respected or not.

The condition set, as shown in Figure 1, is composed of four different attributes. The major attribute is the metric that is being evaluated. It also contains the value against which the metric is compared. The comparison is done through an operator such as: greater, smaller, equal. Finally, since an SLA can have a set of conditions, they are linked using connectors. These connectors are based on First Order Logic (FOL) in order to allow for as much flexibility as possible.

B. SLA Metrics

SLAs describe for both parties (the client and the cloud provider) expectations and act as a roadmap for change in the cloud service. Actually, just as an IT project needs a roadmap that comprises a set of clearly defined deliverables, an SLA is also crucial for working with cloud infrastructure. In fact, to develop a consistent and an effective SLA, a list of important criteria needs to be mentioned [5]. The following are some of the most important criteria:

- Availability: describes the percentage of the availability of the service agreed upon during working and non working days. For example: 99.9% during work days, 98.5% for nights/weekend.
- Performance: this element describes the maximum response times for a specific service.
- Security/privacy of the data: this element is related to the section described above concerning the confidentiality, integrity, availability, and accountability of the data stored within the cloud. An example of a rule regarding security is: encrypting all stored and transmitted data.
- Disaster Recovery expectations: this element describes the commitment sated by the cloud provider to ensure the recovery of data in case of disaster that may affect the main data center.

 ISBN: 978-1-61208-529-6

- Location of the data: this element describes the location where data are stored. This rule should be consistent with local legislation.
- Access to the data: this rule defines the way the client will be using to access its data. An example of this rule would be: data retrievable from provider in readable format.
- Portability of the data: this element describes the identity of another provider that may own the client's data whenever the main provider encounters a problem. In fact, it is possible for the cloud provider to not mention any other cloud provider.
- Change management process: this part deals with process a service should go through to be updated or add new functionalities.
- Exit strategy: this part describes how smooth the exit from the data center of the cloud provider is.

While going through the literature, we have identified two major categories of metrics that can be expressed within SLAs. The first category contains metrics to assess the Quality of Service offered by the CSP, while the second one contains metrics that are used to assess the security of the environment offered by the CSP. In both categories, we could subdivide the metrics based on the level of service offered by the CSP: Software as a Service (SaaS), Storage as a Service, Platform as a Service (Paas), and Infrastructure as a Service. In Figure 2, we classify the metrics that we can assess when we are considering the Quality of Service offered by the CSP.

Since security is one of the most important aspects that clients consider before making the decision to move the management of their data and services to the cloud, we have identified, in Figure 3, the different metrics that need to be expressed in an SLA to assure clients of the security of their assets.

Identifying the different metrics that need to be specified and detailed in SLAs is an important step that will allow us to design our policies. In fact, our team is still working and making progress in detailing the metrics and bringing them to a lower level of granularity. In the next section, we present two policy based SLAs in order to show the way they are represented using our policy specification language.

C. Examples of SLAs

Figure 4 shows two policies that represent SLAs. We intentionally decided to give the example of a QoS SLA and the example of a security SLA.

From the first policy, the user is enforcing the fact that as agreed with the CSP, in case of maintenance work, the services of the client applications must not be down for more than 2 hours and it has to be between 3 and 5 AM. This example shows clearly how a policy can be used to combine metrics extracted from Service Level Agreement with context information (in this example, time) to express the users preferences in terms of Quality of Service. The second policy, on the other hand, deals exclusively with a security issue management. The user wants to be notified in the case of an intrusion detection where the latency of response is more than 80ms. The choice of then 80ms is specific to the client's own knowledge about the application. Finally, from the two examples, we demonstrate how clients can express their Quality of Service requirements and security requirements using software policies. These policies are a representation of the Service Level Agreement between the client and the cloud service provider.

In the next section, we will show the architecture of our system and explain how the policy based SLA management system fits within the cloud environment.

IV. SLA Management Within the Cloud

A. Opting for a Middleware Solution

Our policy based security management system was previously used within the context of mobile computing. When we started thinking of re-modeling it to adapt to the needs of cloud environment, the question of how to insert the software in the cloud was raised. When we were dealing with mobile environments, one of the major concerns that arose was the amount of processing needed by the system versus the mobile device's computing power and battery life. We are not the only ones who struggled with such an issue. In [8], the authors show that there is a direct impact of offloading on energy saving in mobile devices. Computation offloading is defined as "sending heavy computation to resourceful servers and receiving the results from these servers" [9]. In other words, instead of having the heavy computations take place at the level of the mobile device, given that there is an Internet connection available, and a safe medium to transmit the data and instructions, the computations can be delegated to a powerful server (or the cloud). The results can then be sent back to the mobile device. In the same line, researchers have proposed frameworks for developing software to make use of offloading.

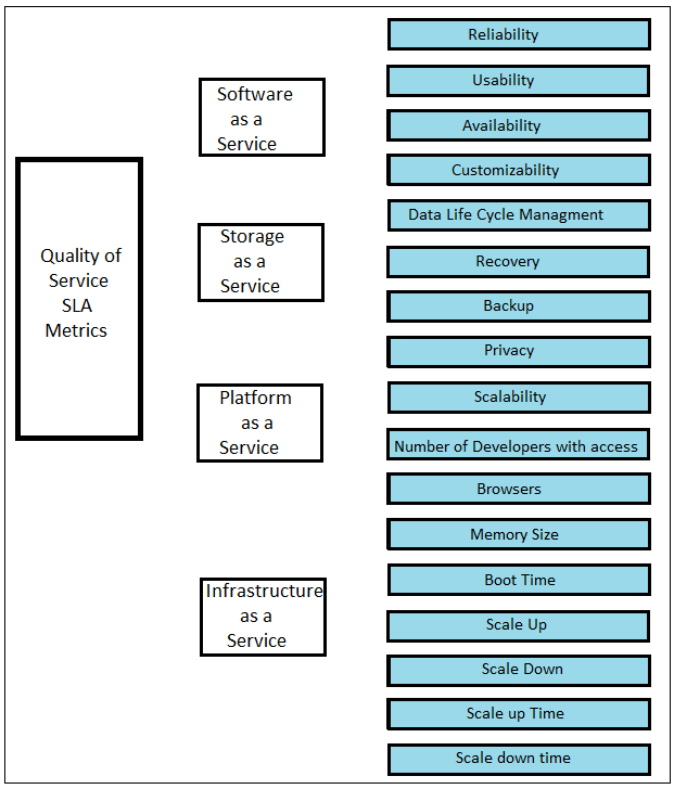

Figure 2. Quality of Service SLA metrics [6]

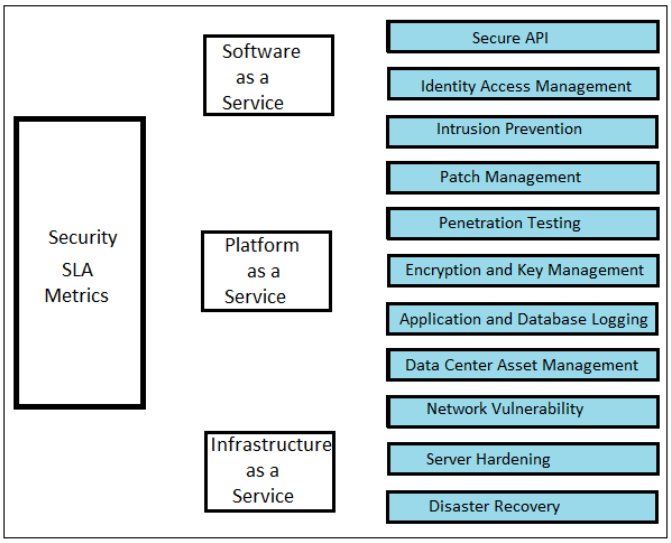

Figure 3. Security SLA metrics [7]

In [10], for example, a framework was tested to considerably reduce execution time of different types of applications given that a fast and reliable network connection is available between the mobile device and the server.

Learning from that experience, we decided that the policy based security management in the cloud should offload client applications from SLA management and the processing it incurs. Using a middleware is the best way to offer to cloud client applications. Since both the middleware and the applications are run by the same physical platform, the issue of latency in response that existed in mobile environments [11][12] does not exist anymore. Figure 5 shows how our system fits within the cloud.

B. *Middleware Architecture*

The middleware contains three main components, as we can see from Figure 5. We have described in detail each one of them in our previous work [4]. The major tasks performed by each one of them are summarized as follows:

- Tool Abstraction Layer (TAL): This component is responsible for collecting context data. This context data can be obtained by software tools just like it can be obtained by Radio Frequency IDentification (RFID) readers or other hardware tools. The tool abstraction feeds the received data to the context and services management system.

- Context and services management system receives data from the TAL and transforms it into useful (needed) context information. It does so by processing the data through the following services:
  - Data transformation services
  - Data dissemination services
  - Data filtering services
  - Data aggregation services
  - Duplicate removal services
  - Data replacement services

  The way the services are managed is also through policies.

```
<?xml version="1.0" encoding="UTF-8"?>
<aui>
 - <SLA>
      <id>4</id>
      <type>obligation</type>
      <subject>client1_PEP</subject>
      <target>client1_NotificationManager.exe</target>
      <action>SendNotification.Code4111()</action>
      <priority>8</priority>
      <audit>yes</audit>
      <active>yes</active>
    - <conditions>
          <condition>Maintenance_status equal Ended AND</condition>
          <condition>System_timezone equal GMT AND</condition>
          <condition>Server_downLength greater 2hrs OR</condition>
          <condition>Server_downHour Less 3AM OR</condition>
          <condition>Server_downHour greater 5AM</condition>
      </conditions>
   </SLA>
 - <SLA>
      <id>5</id>
      <type>obligation</type>
      <subject>client1_PEP</subject>
      <target>client1_NotificationManager.exe</target>
      <action>SendNotification.Code4112()</action>
      <priority>6</priority>
      <audit>yes</audit>
      <active>yes</active>
    - <conditions>
          <condition>IntrusionDetection_Status equal True AND</condition>
          <condition>IntrusionDetection_latency greater 80ms OR</condition>
      </conditions>
   </SLA>
</aui>
```

Figure 4. QoS and Security SLA example

- Policy management system contains the following:
  - Policy Decision Point (PDP): Entity responsible for checking the data provided in a request or a context change notification against the client's policies. The PDP then enforces the action of the policy or not.
  - Policy manager: Entity responsible for adding policies, removing policies, or updating policies.
  - Policy conflict manager: Entity responsible for resolving conflict between different policies whose conditions are met and that need to be triggered.
  - Policy information base: this entity is a repository where all policies are stored.

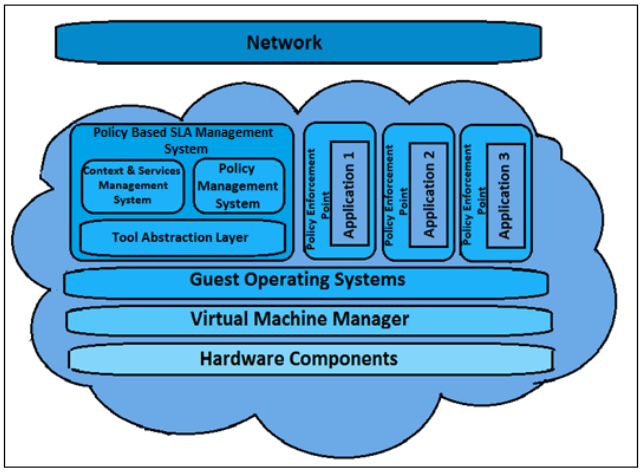

Figure 5. Policy based SLA management system in the cloud

 ISBN: 978-1-61208-529-6

- Policy Enforcement Points (PEP): are wrapping entities that have access to enforce policy actions on the target entities. This access is provided by the client applications through method calls.

Our team has developed a prototype for the middleware and we started proceeding with the evaluation and testing, as we show in the next section.

V. SLA MANAGEMENT MIDDLEWARE EVALUATION AND TESTING

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), in the standard ISO/IEC 9126, later on revised to ISO/IEC 25010:2011, identified the different criteria to evaluate the quality of a software as being: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability [13]. In this section, we will discuss how our middleware performs in each one of those categories keeping in mind that we are talking about a prototype meant for the sole purpose of building a proof of concept.

A. Functionality

The Policy based security management system is responsible for managing and enforcing the SLA policies provided by the client application. In that sense, once the policies are obtained from the client, they are stored in the policy information base and retrieved for evaluation when a request pertaining to the client is received. In our software, only the policies that have as a target the client's application are retrieved, each one of the conditions is checked against the data in our context base and a decision on whether the action of the policy is to be triggered or not is made. Therefore, the middleware fulfills the functionality for which it was designed in terms of suitability and accuracy.

The quality of the context information is managed by the software policies that deal with the requests incoming from the different software tools that provide us with raw data. There are some cases, where a piece of data comes only from one source where it might be given by the CSP itself. For example, the data about availability of the cloud services is posted every month on the website of the cloud service provider. We have not yet identified a tool that can provide us with such an information. Therefore, our context base is fed with data coming from the website of the service provider. Since this process is public, it is up to the client to choose whether to rely on the accuracy of that data in order to formulate their policies of Service Level Agreements. Finally, the policy based security management middleware still fulfills its functionality.

The system is reliable because no external entity can interfere with its processing. The policy management entity and context management entity only receive requests that come from the Policy Enforcement Point, which is part of the middleware. Therefore, the system is reliable and will perform the expected actions the way it was designed to.

B. Reliability

In the context of the discussion, the real threat to reliability is the interaction between the PEP and the client application, and the way the PEP intercepts the incoming requests, models them and forwards them to the policy decision point. The communication protocol between the PEP and the client application (it is also the process by which an application registers to the services of the middleware) is part of the future work. As for metrics such as maturity, and fault recovery we will only be able to test for them when we deploy on the cloud.

C. Maintainability

When we were thinking of the maintainability of the middleware, only one thing came to mind: we have designed the software for maintainability. Policy based systems are by definition maintainable. Since they are managed by policies, to maintain the software all that is required is remove the obsolete policies and replace them with updated versions. In terms of Analyzability, the accountability tag on the policies (audit tag) allows the system to keep track of all the policies whose actions have been triggered. Once new policies are in place it is straight forward to design requests that will test the impact of the policies on the system. Several scenarios have been described before showing how we can use requests to test the policies; therefore, we can say that the system is changeable and stable. As for testability, we have run several performance tests and the results are shown below.

D. Usability & Portability

The programming language and programming platform that we have used are synonyms of portability. The JAVA language and J2EE environment need no introduction and one of their major advantages is portability. As for usability, the client applications, once they have submitted their security policies, their interaction will remain with the Policy enforcement point. A graphical user interface is being developed in order to allow the clients to express their business rules and have them translated into software policies. The user interface is meant to be as user friendly as possible. All the translation into the policy attributes and XML will be transparent to the user.

E. Efficiency (Performance)

One of the major motivations behind opting for the middleware as a way to include the policy based SLA management middleware in the cloud is to offload the tracking and management of SLAs from the client applications. The moment we think about offloading, we want to know how much extra processing time will incur when the services of our middleware are being used. For performance testing, Table 1 shows the platform that we have used.

TABLE 1. TESTING ENVIRONMENT

CPU	Intel core i5 3221M 2.5Ghz
RAM	4 GB
Operating System	Windows 8 professional (64bits version)
IDE	Netbeans 8.0.2

 ISBN: 978-1-61208-529-6

In the first scenario, we wanted to investigate the impact of having multiple clients in our system on the performance of the system with regards to the requests received for a specific client. In the scenario, we have designed 100 policies and we sent 20 requests. During the first run, all the 100 policies belong to client 1. Then, we keep only 80 policies from client 1 (the one to whom the requests are directed) and we add 20 new policies from 4 other clients and we see the impact it has on the average request processing time. We continue using the same technique and, at each step, we reduce the number of policies of client 1 by 20 and increase the other clients' policies by 20. The results are shown in Figure 6.

We see from the graph that the processing time per request decreases with the number of client policies. The more policies we have, the more conditions we check the request against and therefore the more processing time is required. What is interesting for us to see is compare these results with the equivalent ones in the previous figures. That comparison can give us an idea about the impact of having multiple clients versus having one single client on the overall performance.

In the second scenario, we send 20 requests when client 1 owns 60 policies, when he/she owns 40 policies, and when he/she owns 20 policies. Figure 7 shows the average response time when the user is the only client in the environment versus when the environment is shared.

When we first look at Figure 7, we are surprised to see that the response time in shared environment is less than the one in the environment where a client is alone. But the tendency changes as we increase the number of policies. This is explained by the nature of policies. It just happens that the policies used in the first test (20 requests, 20 policies in a non-shared environment) contained more conditions leading to a higher processing time. When we increase the number of policies, it normalizes the number of conditions within a policy and makes the performance in shared environments less than the one in non-shared environments. The reason behind that is the time that is used in selecting the client policies.

Client 1 policies/ other client policies	Response time (ms)
100/0	128
80/20	127
60/40	112
40/60	62
20/80	47

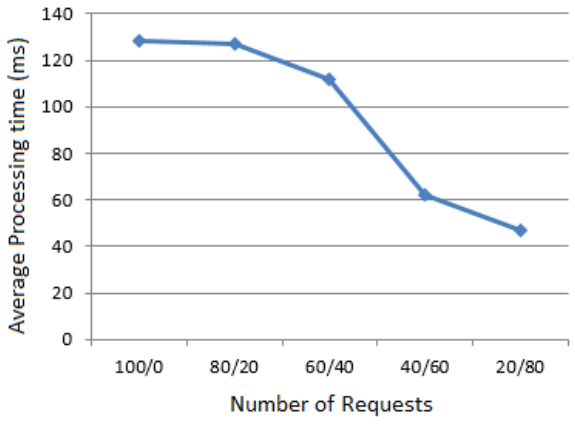

Figure 6. Response time based on the % of client policies in shared environment

VI. CONCLUSION

Policy based management has proven efficient in many environments. First, we have used policies to manage security in mobile environments. Then, we adapted our work to the context of security management in the cloud. Here, we are modeling and testing the management of Service Level Agreements. Next, we are investigating policy based management in mobile cloud computing. Also, the next step in our project is to devise a set of policies that would express the needs of Quality of Service and security for a real life client. At the university, we have a private cloud which is the ideal environment for us to perform the necessary set of tests in order to see how the system performs.

Number of Policies	Response time (ms) Shared Environment	Response time (ms) Non Shared Env
20	47	69
40	62	73
60	112	104

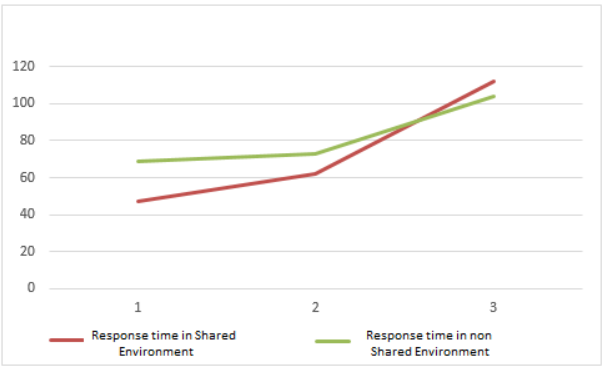

Figure. 7. Performance in shared environment Vs non-shared environment

REFERENCES

[1] G. Orsini, D. Bade, and W. Lamersdorf, "Cloudaware: Towards context adaptive mobile cloud computing," in IFIP/IEEE IM 2015: 7th Intern.Workshop on Management of the Future Internet (ManFI), 2015.

[2] S. Hiray and R. Ingle, "Context-aware middleware in cyber physical cloud (CAMCPC)". Proceedings of the 2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE)", Pune, India, 15–16 November 2013, pp. 42–47.

[3] N. Gupta and A. Agrawal, "Context Aware Mobile Cloud Computing: Review", 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1061–1065, 2015

[4] M. Chraibi, H. Harroud, and M. Maach, "Personalized security in mobile environment using software policies", UBICOMM 2011 : The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Nov 2011, Lisbon, Portugal.

[5] http://www.facilities.ac.uk/j/free-cpd/155-slas-and-service-specifications, retrieved: December, 2016

[6] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing", 4th IEEE International Conference on Digital Ecosystems and Technologies, 2010, doi:10.1109/dest.2010.5610586

[7] M. Hoehl, "Proposal for standard Cloud Computing Security SLAs - Key Metrics for Safeguarding Confidential Data in the Cloud", Used from:

 ISBN: 978-1-61208-529-6

https://isc.sans.edu/forums/news/Proposal+for+standard+Cloud+Computing+Security+SLAs+Key+Metrics+for+Safeguarding+Confidential+Data+in+the+Cloud/893991/studies on magneto-optical media and plastic substrate interface", IEEE, retrieved: December, 2016

[8] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" IEEE Comput, pp. 51–56, 2010

[9] K. Kumar, J. Liu, Y. Lu, and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems", In the Journal of Mobile Networks and Applications, Springer, pp. 129–140, 2012.

[10] E. Cuervo, "MAUI: Making Smartphones Last Longer with Code Offload", Proc. 8th ACM MobiSys, 2010

[11] C. Shi, K. Habak, P. Pandurangan, M. Ammar, E. Zegura, and M. Naik, "Cosmos: Computation offloading as a service for mobile devices", ACM MobiHoc, 2014

[12] H. Flores, S. N. Srirama and R. Buyya, "Computational offloading or data binding? bridging the cloud infrastructure to the proximity of the mobile user", Proceedings 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2014

[13] http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733, retrieved: December, 2016

 ISBN: 978-1-61208-529-6