Secure operations as congestion control mechanism within OpenStack based Cloud Laboratory

Franko Hržić and Domagoj Poljančić and Tihana Galinac Grbac
Faculty of Engineering
University of Rijeka
Vukovarska 58, 51000 Rijeka
Email: tihana.galinac@riteh.hr

***Abstract*—In this paper we present a service that provides secure operation for OpenStack based compute operations. It is suitable for governing sensitive configuration operations in dynamic untrusted Cloud environments. We provide an example that is implemented as part of SEIPLab Cloud Service Monitoring System. SEIPLab Cloud is virtual laboratory solution developed for use of Software Engineering and Information Processing Laboratory equipment at Faculty of Rijeka that is used for student Software Engineering projects. To enable reliable simultaneous use of number of students and students groups sharing the same physical SEIPLab infrastructure we developed a smart management system for SEIPLab Cloud. The notification service, is a function that is realized as a service within OpenStack based Cloud solution that implements notification and validation of user activates trough SMS and e-mail communication. This service restricts user behavior in operations related to configuration of user laboratory environments. Thus, the service avoids possible cloud management congestion situations initiated by simultaneous users and irregular user behaviour.**

I. Introduction

Infrastructure as a service (IaaS) is a paradigm that provides opportunity to end users to use infrastructure resources (CPU, I/O, memory) of a Cloud provider on demand and as a service. With this opportunity the end user is offloaded with resource care activities. In a case of a service malfunction the end user may just switch to another service deployment solution within the same Cloud or to another Cloud provider. However, some applications require the process of transition to be reliable and secure and based on timely notification and decision.

Cloud providers are challenged to offer its infrastructure as a service, to many of its users and simultaneously. Success of Cloud provider greatly depends on reliability, availability, elasticity and performance of services offered to its users [1]. Even best business models may fail due to unreliable and unavailable service and that is why Cloud providers are competing among themselves to attract and maintain customer trust. We define the provider's Cloud resiliency as its ability to predict failure or service unavailability and act accordingly to avoid and mitigate problems related to proper service executions. One important method to do that is by monitoring and controlled guiding of user initiated operations executed within the Cloud system.

In most cases the Cloud providers provide services specialised to offer the hardware and software management to its end users. They may serve numerous end users simultaneously and their management activities are focused on proper operation of Cloud resources in common. Unfortunately, there are still lack of management operation functions that secure reliable Cloud service operation and Cloud provider resiliency. Malfunctions or Service Level Agreement (SLA) violations that are related to single customer may be undetected to Cloud provider and thus not properly handled to end users satisfaction. Autonomous services may be of critical importance in such situations [2]. An autonomous system is capable to exhibit a large degree of self-governance without any support of external entities and is completely independent while autonomic system do the same but based on 'internal policies and principles which can be described as autonomic [3]. The complexity of operations that are simultaneously executed within Cloud environments that are under supervision of Cloud provider are the main cause of failing to address SLA and quality issues with every single end user. New concepts that would be applied within new generation of intelligent computer system need to be developed [4]. Focus is on management procedures that would minimise the need of human administrator. In such scenario, control procedures based on the monitoring systems of resource usage and operations executed per each end user are investigated.

In aim to investigate development of autonomic management procedures and introduce these concepts into education system within the university, a number of projects have been established to introduce virtualisation into student lab environment and enable students to experiment with management procedures over the Laboratory resources [5], [6], [7]. This work was motivated by the same goal when virtualisation was introduced into Computing study programme of Faculty of Engineering at University of Rijeka. The Software Engineering and Information Processing Laboratory (SEIPLab)[1] at Faculty of Engineering, University of Rijeka host numerous physical but also software resources that were virtualized using OpenStack technology. These virtualized resources were provided to students 'as a service' to experiment in their student projects within the Software Engineering Management course at graduate level of Computing study programme of Faculty

[1] $http://www.seiplab.riteh.uniri.hr/$

978-1-5090-3720-9/16/$31.00

of Engineering, University of Rijeka. The SEIPLab resources are limited and a number of unexperienced student groups experiment with SEIPLab virtual resources simultaneously. By executing a number of student projects on SEIPLab Cloud we identified that student unexperience is negatively related to students patience while waiting response from SEIPLab Cloud platform and that was identified as the root cause of huge workloads and delays in SEIPLab Cloud service operation. Therefore, we were motivated to develop a guided and authorized procedure for critical operations triggered by students while managing and orchestrating SEIPLab virtual resources. This work is aligned with work of research project executed within the SEIPLab on Evolving Software Systems: Analysis and Innovative Approaches for Smart Management (EVOSOFT) with aim to investigate and develop new and innovative approaches for autonomic and smart management of evolving software systems.

Besides the cloud computing security issues and challenges, the student Lab environments have some additional specific challenges that we are addressing in this paper [8]. There are number of security models and their implementation may vary depending on the addressed needs within the context of application. Furthermore, there are studies working on congestion control mechanisms aiming to overcome load situations [9]. For example the congestion control mechanisms are focused on fair resource allocation among users during congestion [10]. In our study we approached the congestion control by employing additional authorisation of end user and thus providing early feedback that would encourage the student to be patient and gaining more time to user request to be executed. The *Notification service* presented in this paper provide guided execution of student operations, in particular creation of virtual machine instance, with additional authorisation and authentication while operating over Cloud resource configuration and thus enabled better availability and reliability of SEIPLab Cloud resources. Our proposal is based on using the standard security modules available within the implementation of OpenStack Keystone service management module and related Python application platform interface that helped us to automate OpenStack scripts. Massive delays of working system are minimised with help of Notification service. As part of the SEIPLab experimental platform we are developing a number of management and orchestration services for executing student projects.

II. Technology

A. Cloud technologies

Cloud computing [11] is a paradigm where any physical resource can be virtualized and provided in 'as a service' fashion over the Internet network to its users. The key element of cloud computing paradigm is *virtualisation* where network resources are separated from their logical software representation. Logical abstraction of network resources enables easier and dynamic management. With this paradigm the services can be enabled and tailored per user, while the costs of management activities are drastically decreased. We may vitalise following network resources and provide in 'as a service' fashion: physical hardware (Infrastructure as a service, IaaS), operating system (Platform as a Service, PaaS) and application software (Software as a Service, SaaS).

There is lot of work done on virtualisation of network resources and numerous commercial Cloud environments (Amazon EC2, Google App Engine, Microsoft Azure, VMware, KVM, etc.) are offering all network resources as a service. At the same time there is ongoing development of open platforms one example is OpenStack. However, there is limited related work on managing these virtual network services. For dynamic use of network resources by numerous simultaneous users it is of crucial importance to implement reliable, secure operation of these network function virtualisation NFV services.

OpenStack has been developed in number of projects. Each project has implemented specific functionality in its separate module that communicate in between through RabitMQ messaging system for reliable communication. The modules are implementing following; compute services in nova, networking in quantum, storage services in cinder, identity services in keystone, image services in glance. These modules are forming the main core elements of OpenStack. Also, these services may be accessed and used through standardized Application Platform Interfaces (API), through Graphical User Interface that provides the most standard set of functions, but also through Python libraries[2] and HEAT resource module for orchestration of resources[3]. The main idea is to provide a standardized set of interfaces to the end users and enable them to develop its own customized cloud management scripts and services.

Since there are a number of possibilities that OpenStack provide 'as a service' and huge market penetration for use of these services, the complexity of management operations within the Cloud environments is growing. Therefore, there is ongoing work for autonomous control for large scale and reliable services within and across Cloud environments. Such example is an European project Autonomous Control for a Reliable Internet of Services (ACROSS[4]). The student project presented in this paper is one of the numerous student projects performed within Software Engineering and Information Processing Laboratory (SEIPLab) aiming to establish experimental platform for research on predicting and modelling of evolving software systems, effect of service composition structure and performance, quality and reliability within EVOSOFT project.

B. Standardisation

The European Telecommunications Standards Institutes Network Functions Virtualization (ETSI NFV) Industry Specification Group 13 is defined to work on Management and Orchestration (MANO) of NVF deployments in virtualized environments. The work of this group is focused on abstraction models and Application Platform Interfaces, provisioning and

[2] $https://docs.python.org/3/library/$
[3] $http://docs.openstack.org/developer/heat/$
[4] $http://www.cost-across.nl/$

configuration of NFV resources, operational management, interworking with existing Operating systems within the network. The ongoing activity is to develop an open source NFV management and orchestration software platform. The main objective of this working group is to consolidate activities by numerous Cloud platform developers and unify its common operation to enable its secure and reliable interworking functions.

C. SEIPLab Cloud Environment

In the SEIPLab the Cloud environment is established with Mirantis Open Stack software distribution [5].

For the purpose of student projects the limited system administrative access rights are provided to students. Available Cloud resources are distributed among number of students within the course of Software Engineering Management at Computing study programme of Faculty of Engineering, University of Rijeka. Since students work in pairs the resources are assigned to the student pairs organized in student projects. A local SQL database was created with student project, student name, surname, id, user id, and password. The restricted usage of SEIPLab Cloud resources are assigned to each student project and resource management is performed with tenant quota assignments through Keystone Application Platform Interface. Students access the SEIPLab resources over the secure SSH connection with provided user credentials that are verified at user login. Also, all user actions and resource use is restricted per student project that students belong. Therefore, user actions and resource usage is verified at every user action activation. The configurations created by student projects is removed periodically from the SEIPLab Cloud environment. Project actions are monitored through SEIPLab Cloud environment and their lifetime is limited by SEIPLab administrator. All actions with timeout are deactivated and removed during the daily system scan.

III. Notification Service

During the execution of student projects using SEIPLab Cloud environment number of students simultaneously used the SEIPLab Cloud resources. There were problems with congestion of student requests while attempting to execute Actions within the SEIPLab Cloud. The Notification service is developed to overcome this limitations.

The implementation of Notification service is done by using OpenStack Nova–api resources and Python scripts. Nova, OpenStack Compute service that is used for hosting and managing cloud computing systems, provides its own API for the Python programming language. In Notification service, Nova is used for controlled creating a computing instance on OpenStack cloud. So, to create these instances and use Nova, the Notification service is using following resources:

$novaclient.client$ A resource that provides access to the Nova part of OpenStack. The resource is used for scanning OpenStack cloud flavors, images, security keys and networks, respectively all the settings that are needed to create the instance. Also, when all of the settings needed for the instance to perform are installed, the $novaclient.client$ resource creates that instance on OpenStack.

Keystone service is identity service used by OpenStack for authentication and high-level authorization. It provides a central directory of OpenStack service users and acts as a common authentication system across the cloud operating system. When establishing SEIPLab environment we used the Keystone service for defining students and student groups with access rights to access the SEIPLab Cloud resources over OpenStack. The student identity is stored in Keystone module and this information is used within Notification service. Without providing user credentials (his environmental user credentials) Notification service script cannot make any changes to OpenStack.

The Notification service uses Python scripts that implement and collect all the necessary data for sending an e-mail or an SMS to the user, such as chosen instance settings, users e-mail address, a mobile phone number. Python scripts use the following libraries:

- time The library that provides time delay. It is necessary to give OpenStack a few seconds for creating and setting up the instance.
- os/environ The library used for collecting environmental variables that are needed for authentication.
- Os The library provides many useful functions to work with a terminal. The Notification service uses os library to edit the terminal and to communicate with the user.
- Random The library that provides random generated numbers. The Notification service generates the verification code with random characters.
- smtplib The smtplib module defines an SMTP client session object that can be used to send an e-mail to any Internet machine with an SMTP or ESMTP listener daemon, generally, it is used for sending an e-mail to the user.
- Clockwork Clockwork is a Python wrapper for Clockwork SMS API a service that provides easy SMS sending to any mobile number specified in the script.
- $Validate_email$ Validate email is a Python package whose task is to check if the e-mail is valid, formed properly and if the e–mail truly exists.

Notification service pseudo code will be elaborated in the following lines:

1) Import all the necessary libraries and modules.
2) Provide user credentials to OpenStack
3) Fetch all available properties (flavor, image, security key, network) from OpenStack server for the creation of a Computing instance.
4) Prompt the user to set up all properties for his instance.
5) Prompt the user to give an e-mail for the verification code.
6) Generate the e-mail with basic information about the instance and the verification code.

[5]https://www.mirantis.com/

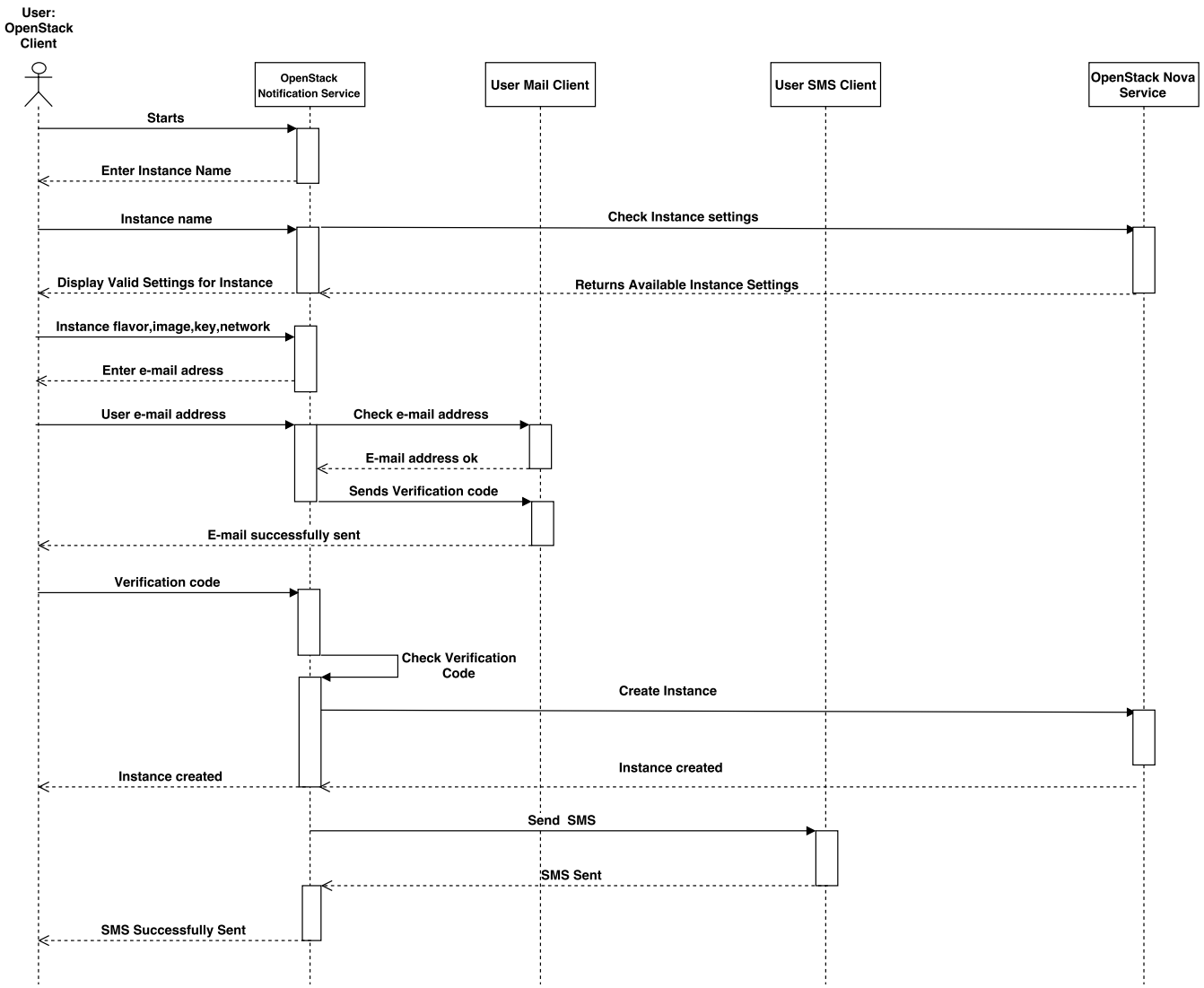

Fig. 1. Message Sequence Chart diagram of Notification service

7) Send an e-mail to the user and wait for him to enter the verification code.
8) If the verification code is correct, the instance will be created, otherwise the script will be stopped.
9) If the instance is successfully created, send an SMS to the administrator, otherwise stop the script.
10) Inform the user about the script created.

The high level message flow among the entities is depicted in the Figure 1. We have following entities: the user as OpenStack Client, the OpenStack Notification service, the User Mail Client, the User SMS Client and the OpenStack Nova Service. As already explained, we assume the user is a student that has been authenticated to OpenStack service by using Keystone module and that he gained necessary credentials. Once the credentials are set up, user starts the OpenStack Notification service by simply starting $Notification\_service.py$ script. When script is started, it welcomes user and asks him for confirmation that he wants to start the script. When $Notification\_service.py$ script is started, it instance novaclient object that scans OpenStack server and store all properties required for creating instance. Once when all properties are acquired and stored in separate lists (images, flavors, keys, networks) script prompts user to enter instance name. After the name is entered, script prompts user to choose properties for his instance from properties lists that are generated in step before. When all necessary data for instance creation is prepared, validation procedure starts. First of all, sender e–mail address is provided by $Notification\_service$. Secondly, script asks user to enter his email address on which he will receive validation code. Email address from user is form validated. The verification code is generated by the Notification service using uppercase and lowercase letters, also including numbers. Once when validation code is generated, for example:$A01wnTr3$, it is sent to user email address in the form depicted in figure 2.

Afterwards the user enters the verification code. The Notifi-

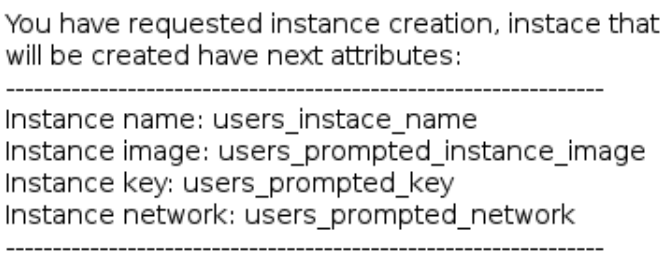

Fig. 2. Output message

cation service checks the verification code by simply matching user entered code against generated one. If the code is correct, Notification service generates new nova object that creates new instance with properties that user has entered on the OpenStack server. During the time the instance is being created, the Notification service sends an SMS to the administrator informing him of the taken actions (the instance being created) providing instance name and user's email address. Once the procedure is completed, the Notification service informs the user that the instance is successfully created.

IV. Benefits and discussion

The main benefit of using Notification service arises from the fact that it is very scalable service that can find its place in most applications. Here we discuss identified core benefits for using it:

1) **Authorisation:** Notification service provides an additional authorisation system. Even if someone were to steal users credentials, he would still need to provide an email address to execute any action on Cloud. Also every action taken by the user is monitored by the cloud administrator via SMS. That would provide additional security mechanism for users of SEIPLab Cloud resources.
2) **User control:** Sometimes users that have access to cloud resources unintentionally overuse them without caution. In that case, the cloud administrator can react by sending an email to them, since he knows who is overusing the cloud resources. This scenario is very common if cloud is shared by many people who need to use limited resources. On that way, the student user is aware of this monitoring service and because of that the assumption is that she or he would more use Cloud resources with more care and patience. We expect that this awareness would somehow limit the number of unfinished jobs.
3) **Awareness:** By forcing the user to open an email and enter the verification code, Notification service puts an additional effort for the user who now needs to rethink about the actions he is going to take. Often users quit their action, because by the time they open the email and enter the verification code, the action they were going to take is no longer needed.
4) **Performance:** Additional verification procedure that will divide a bigger job into smaller two jobs introducing an individual sms sent to student personal and professor equipment may result with student grater patience while using Cloud resources because she/he was able to communicate with system and thus introduce a confidence into service. We expect that more confident user would be more patient with a service waiting time and that would eliminate unnecessary load from forced service shut down by the student user. Therefore, we may have positive effect on enhancing server performances. Furthermore, splitting one bigger task the Cloud platform has to execute into two tasks with introduced new communication service is expected to have positive effect on performance although some additional time would be introduced in total service delivery but with smaller response time for each job within the service.
5) **Monitoring and statistics:** additional verification may improve resource use monitoring and statistics from server resources that may be further used for student evaluation and future Cloud configuration enhancements and Cloud resource planing.
6) **Division of access rights and policies** Because of additional authentication procedure the operation that use such Notification service may employ additional policies that may be independent of user properties. The management of such policies is now much easier since these may be tied to operations and users independently.

V. Conclusion and future work

The virtualisation technologies have huge potential for future use of computation resources in 'as a service' fashion. This concept is expected to introduce revolution not only in computer science domain but also in numerous other application domains of information and communication technologies. The autonomic and autonomous management functions within virtualised infrastructure environments are crucial for its future wider adoption. In this paper, we discuss problems in simultaneous experimentation of unexperienced students on the same SEIPLab infrastructure through virtualised solution based on Mirantis and OpenStack. The service unavailability caused by numerous student requests has been addressed with the proposed notification service. Each time a student wants to execute instantiation operation his identity is verified not only within the Open Stack environment with usual student credentials but also by SMS or email notification and authorisation service. Thus, students get confirmation that their request is processing and more controlled execution of critical operations is introduced. This approach is very useful in the case of student project virtual laboratory and reduced the impact of workload caused by eager students. We discussed benefits in terms of additional security, better control of user operations and behaviour, user awareness of controlled environment, performance and monitoring and statistics. Still, our discussion

was based just on the facts that arise from the service design. Our future work would focus more on quantifying these benefits with concrete measurements of using the service by many users in future operations of SEIPLab Cloud. As our future work we aim to develop an autonomic student platform system that would be capable to self manage and control SIPLab Cloud platform while executing student projects. This work contributes to our goal within the EVOSOFT project to understand complex system behaviour and develop sound design principles for building complex and distributed systems.

ACKNOWLEDGMENT

This work has been supported in part by Croatian Science Foundation's funding of the project UIP-2014-09-7945 Evolving Software Systems: Analysis and Innovative Approaches for Smart Management (EVOSOFT), the University of Rijeka Research Grant 13.09.2.2.16 and COST Action 1304 Autonomous Control for a Reliable Internet of Services (ACROSS).

REFERENCES

[1] M. Armbrust, et. al., A view of cloud computing, Commun. ACM 53(4), 5058, 215 (2010)

[2] I. Brandic, Towards self-manageable cloud services, in Proc. of the 2009 33rd Annual IEEE International Computer Software and Applications Conference - Volume 02, ser. COMPSAC 09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 128133.

[3] N. Agolumine, "Autonomic Network Management Principles, From Concepts to Applications", Elsevier, London, UK, 2011.

[4] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland. "A concise introduction to autonomic computing", Adv. Eng. Inform. 19, 3 (July 2005), 181-187.

[5] K. Krishna, W. Sun, P. Rana, T. Li, and R. Sekar. V-netlab: A cost-effective platform to support course projects in computer security. In Proceedings of 9th Colloquium for Information Systems Security Education, June 2005.

[6] W. D. Armitage, A. Gaspar, and Matthew Rideout. 2007. Remotely accessible sandboxed environment with application to a laboratory course in networking. In Proceedings of the 8th ACM SIGITE conference on Information technology education (SIGITE '07). ACM, New York, NY, USA, 83-90.

[7] B. R. Anderson, A. K. Joines, and T. E. Daniels. 2009. Xen worlds: leveraging virtualization in distance education. In Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE '09). ACM, New York, NY, USA, 293-297.

[8] K. Popović and Ž. Hočenski, "Cloud computing security issues and challenges," MIPRO, 2010 Proceedings of the 33rd International Convention, Opatija, Croatia, 2010, pp. 344-349.

[9] K. S. Reddy and L. C. Reddy, A Survey on Congestion Control Mechanisms in High Speed Networks, International Journal of Computer Science and Network Security (IJSNS), Vol.8, No.1, Jan. 2008.

[10] T. Tomita and S. i. Kuribayashi, "Congestion control method with fair resource allocation for cloud computing environments," Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on, Victoria, BC, 2011, pp. 1-6.

[11] Cloud Computing: The New IT Paradigm, 2010, available at http://itechthoughts.wordpress.com/2010/02/23/cloud-computing-the-new-it-paradigm