A Study of Cloud Mobility in a Mobile Cloud Network based on Future Internet Approach

Dongha Kim, Hyunjun Kim, Gijeong Kim, Sungwon Lee
Department of Computer Engineering
Kyung Hee University
Young-in, South Korea
{dongha, kimhyunjun, kimgijeong, drsungwon}@khu.ac.kr

***Abstract*— In this paper, we extend the limited functionality of the GENI Cloud project as follows. First, we use the OpenStack cloud platform instead of the Eucalyptus cloud platform used in the GENI Cloud. Second, we develop the FiRST Cloud aggregate manager (AM) based on GENI AM Application Programming Interface (API) for the federation between a future internet test-bed and the OpenStack cloud platform. Third, we develop a Cloud Mobility Server and Client for mobile cloud management in order to control the zero-client service. Thus, we confirm that the proposed FiRST Cloud AM is feasible through zero-client mobile cloud service.**

Keywords-cloud; mobile cloud; cloud mobility; future internet; FiRSTCloud AM.

I. INTRODUCTION

Since 1974, when the Internet was first proposed, the Internet has become a global network. Since 2000, however, rapid change of communication environments and various user requirements trigger numerous researches for Future Internet to overcome conventional Internet's problems [1].

A new trend of these Future Internet research is harmonization between the conventional Future Internet and cloud computing. Disadvantages of current cloud computing include limited bandwidth and highly variable latency due to the conventional Internet limitations is able to complemented by Future Internet concept [1].

Cloud computing includes aspects such as grid computing, utility computing, thin-client based computing. Cloud computing requirements for client hardware and software are continually being simplified. Mobile handheld devices are able to take special advantage of these simplified requirements.

An optimized, robust network is needed for cloud computing, along with an optimized protocol for communication between the client device and the cloud server. Researchers are studying projects which use cloud computing in large-scale global test-beds.

In this paper, we develop zero-client based mobile cloud service with open-source cloud platform, Openstack [8]. Furthermore, to improve this service, we develop cloud mobility server, client and FiRST cloud Aggregate Manager (AM).

First, cloud mobility server and client support a client for receives service not from 'local site' but from 'remote site'. Local site is an original server that has client's data and remote site is a closest cloud server to the client.

Second, FiRST Cloud AM is an API based on GENI AM API to interwork with Future Internet test-bed.

As a result, we develop a Cloud Mobility server and client for mobile cloud management to support the zero-client cloud service and to confirm the feasibility of the proposed FiRST Cloud AM with a zero-client mobile cloud service.

This paper organized as follows: related works are introduced in following section. After that section, detailed information about proposed cloud mobility and interaction with future Internet are described. Next section presents result data of performance evaluation. Finally, conclusion and future work is presented

II. RELATED WORK

A. Future Internet Test-bed: GENI

The Global Environment for Network Innovations (GENI) [3] is designed to support experimental research in network science and engineering. This research challenges us to understand networks broadly and at multiple layers of abstraction, from physical substrates through architecture and protocols to networks of people, organizations, and societies. The intellectual space surrounding this challenge is highly interdisciplinary, ranging from new research in networking and distributed system design to understanding the theoretical underpinnings of network science, policy, communication networks and economics,. Such research may generate new knowledge about the structure, behavior, and dynamics of the most complex systems – networks of networks – with potentially huge social and economic impacts [2][3].

B. GENI AM API

The GENI Aggregate Manager API is a common API for reserving disparate resources from multiple GENI aggregates. Prior to this API, each control framework specified a unique interface between aggregates and experimenters.

The GENI Aggregate Manager API specifies a set of functions for reserving resources and describes a common format for certificates and credentials to enable compatibility across all aggregates in GENI. The aggregate is an abstract concept represents set of resources. This API has been implemented in multiple control frameworks, and will serve as the basis for ongoing integration among GENI control frameworks and tools. Using this document, new GENI-interoperable aggregate managers, tools, and clearinghouses may be constructed [4].

C. *Eucalyptus Cloud Platform*

Eucalyptus [9] stands for 'Elastic Utility Computing Architecture Linking Your Programs To Useful Systems' and is an open source platform in the Infrastructure as a Service (IaaS)-style based on Linux. Eucalyptus is software available under GPL that helps in creating and managing a private or publicly accessible cloud. It provides an EC2-compatible cloud computing platform and a S3-compatible cloud storage platform [9].

D. *Openstack Cloud Platform*

OpenStack is a global collaboration of developers and cloud computing technologists aiming to produce a ubiquitous open source cloud computing platform for public and private clouds. The project aims to deliver solutions for all types of clouds with simplicity, ease of implementation, scalability, and feature selection.

Founded by Rackspace Hosting and NASA, OpenStack has become a global software community of developers who collaborate on a standard and massively scalable open source cloud operating system. All of the code for OpenStack is freely available under the Apach e 2.0 license, and anyone can run it, add to it, or submit changes back to the project. An open development model is the only way to foster badly needed cloud standards, remove the fear of proprietary lock-in for cloud customers, and create a large ecosystem that spans cloud providers.

The current OpenStack project has been divided into two kinds of software. The first, OpenStack Compute (Nova), is cloud management software used to operate and manage the infrastructure for large-scale provisioning of virtual machines. Second, OpenStack Object Storage (Swift) is storage system software that offers the reliable distribution of a store of objects.

E. *GENICloud*

GENICloud's goal is to allow the federation of heterogeneous resources like those provided by Eucalyptus, an open-source software framework for cloud computing, to coexist with GENI. Under the federation of Eucalyptus and GENI, a more comprehensive platform is available to users; for example, development, computation and data generation can be completed within the cloud, and deployment of the applications and services can be conducted on the overlay (e.g., PlanetLab).

By taking advantage of cloud computing, GENI users can not only dynamically scale their services on GENI depending on demand, they can also benefit from other services and uses of the cloud. GENICloud is complementary to Future Internet test-bed by federating heterogeneous resources, for example, a cloud platform with PlanetLab. Both PlanetLab and Eucalyptus architectures offer some insights into some of the similarities between the two seemingly disparate systems. PlanetLab comprises nodes scattered around the globe, and Eucalyptus consists of clusters. Both PlanetLab and Eucalyptus start out with some computing resources, namely, physical machines that can be provisioned to users [7].

F. *PlanetLab*

To provide a more realistic platform for researchers, PlanetLab is a test-bed for exploring disruptive technologies on a global scale. Testing distributed applications and network services on a global scale has always been difficult because deploying such applications and services could have adverse effects on the Internet. Also, PlanetLab is built as an overlay network to be positioned over the Internet. [5]

PlanetLab defines the treatment of a set of distributed virtual machines as a single, compound entity called a slice. The concept comes from the fact that, whenever a service is running on PlanetLab, it receives a slice (virtual machines running on different nodes) of the PlanetLab overlay network. An individual virtual machine within a slice is called a sliver. GENICloud has expanded the concept of slices to include Eucalyptus virtual machines and, in the future, storage capability. Therefore, a slice in GENICloud can have both PlanetLab resources and virtual machines from a Eucalyptus cloud. The users can log into individual slivers in a GENI Cloud slice to conduct their experiments.

G. *Eucalyptus Aggregate Manager*

Most of the implementation effort of GENICloud is concentrated on implementing the aggregate manager over Eucalyptus. In addition, a resource specification format is formulated for Eucalyptus.

The aggregate manager acts as a mediator between PlanetLab and a Eucalyptus cloud. The manager manages the creation of Eucalyptus instances for the slice and maintains a map of the slices and instances so when the users query the sets of resources allocated for their slices, the information is readily available.

H. *Resource Specification (RSpec)*

The resource specification is an XML document that can be used by the aggregate manager to return information to the users. The users can then use the specification to send information to the aggregate manager. Since the resource specification is in XML format, the format of the RSpec for a specific network is completely open for the network to define. With such openness, the RSpec can encompass many different types of resources and different network topologies. As a result, many networks (e.g., PlanetLab, VINI, ProtoGENI) have different RSpec formats [4][7].

GENICloud defined an RSpec for Eucalyptus, so that its resources and requests from users can be expressed in XML format. During the workflow, users interact with the slice manager using RSpec devised for Eucalyptus.

I. *Definition of the Problem*

Mobile cloud service has problems like packet loss, low bandwidth, bandwidth fluctuation, and delay fluctuation based on broadband communication and delay based on WAN. These problems obstruct users who want to use mobile cloud services. GENI Cloud supports interaction among heterogeneous resources on the Future Internet and Eucalyptus cloud computing platform. It provides better communication circumstance. However, the GENI Cloud project provides limited functionality, which includes few features of cloud computing capabilities.

III. CLOUD MOBILITY CONTROL FOR A MOBILE CLOUD

A. *Key Features of Cloud Mobility Control*

Mobile cloud service has problems like Packet loss, lower bandwidth, bandwidth fluctuation and delay fluctuation based

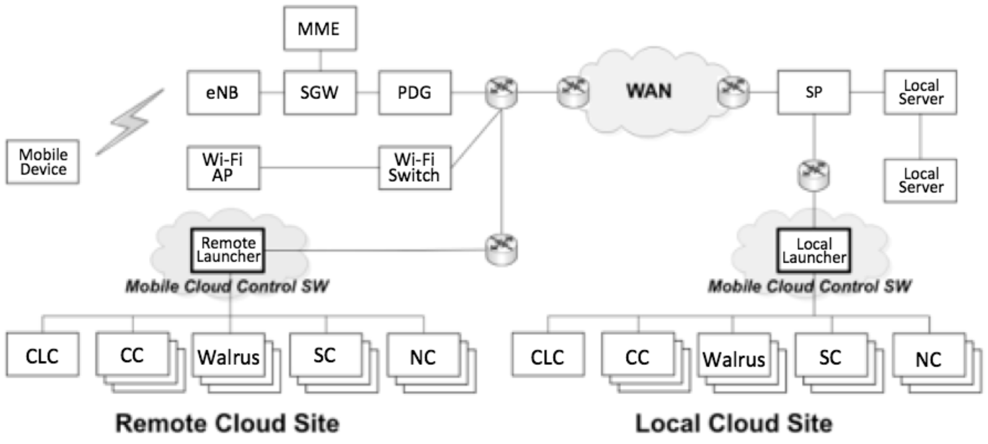

Fig. 1. The concept of proposed remote cloud based on mobile cloud architecture

on broadband communication and delay because of the WAN area. To solve problems with mobile cloud service, we proposed software which has a remote control function to improve cloud service. Followings are explained based on Eucalyptus as a cloud platform because of our first cloud mobility concept is established using Eucalyptus. In fact, it's more easy and clear to explain our concept of cloud mobility using Eucalyptus. However, the Openstack is very similar with Eucalyptus; it's very easy to apply this concept from Eucalyptus to Openstack.

B. Cloud Platform Based on Remote Cloud Mobility Control

In the mobile cloud platform based on a remote cloud, the mobile cloud provides an on-demand/pre-reserved virtualized service by incorporating the cloud server into the telecommunications and wireless carrier networks rather than using the server outside of the WAN, as shown in Figure 1.

Remote cloud's environment-information managing function is able to provide cloud service environment-information (which is originally at the 'local' cloud server on the outside of WAN) to cloud server in 'remote' mobile operator network. When user requests cloud service, Cloud Controller (CLC) determines the location of the new Virtual Machine (VM) to create. If service is requested through a mobile operator, CLC requests user environment-information from the remote cloud server and uses the information to create a VM. If service is terminated, CLC returns the user environment-information to the cloud server. Both the local cloud and the remote cloud provide mobile cloud services which use proxy server software based on the Eucalyptus cloud platform. It is simple software shown as 'Remote Launcher' and 'Local Launcher' in Figure 1. Each launcher substantially controls cloud mobility as explained in the next sections.

C. Design of Cloud Mobility Control

We establish a design based on a remote cloud system for the proposed cloud mobility control. Table I represents the common message header for cloud mobility control. First, we divide local and remote cloud systems into categories based on 'Kind of cloud system.' In addition, we should be able to support seamless mobility control by including 'Type of component,' 'Type of message-passing,' 'Message order,' and 'Source and Destination IP addresses' in the header.

We propose a design for cloud mobility control software based on the following four basic functions for controlling local and remote cloud systems.

D. Cloud Computing Mobility Environment Configuration

Local launcher is a gateway for administering the local cloud system in a remote cloud architecture environment. Each cloud system (local and remote) exchanges environment-information with another cloud system and utilizes the appropriate information from service requests and communicates with another.

TABLE I. COMMON MESSAGE HEADER FOR CLOUD MOBILITY CONTROL

Component	Size (Octet)	Default	Meaning	
Kind of cloud system	1	0x00	0x00	Local cloud system
			0x01	Remote cloud system
Source component type	1	0x00	0x00	Default(App launcher)
			0x01	CLC(Cloud controller)
			0x02	CC(Cluster controller)
			0x03	SC(Storage controller)
			0x04	Walrus
			0x05	NC(Node controller)
Type of message passing	1	0x00	0x00	Default
			0x01	Request
			0x02	Response
Message order	1	0x00	Message order	
Source IP	4	0x00000000	Source IP address of a message	
Destination IP	4	0x00000000	Destination IP address of a message	

E. Mutual Recognition and Authentication between Cloud Systems

The cloud systems also process mutual recognition and authentication. If a remote user makes a request to the cloud system, the local cloud exchanges authentication information for remote cloud service between the local cloud and the remote cloud. By exchanging authentication information, the connection setting is established.

Each cloud system checks for available components on its own system. CLC shows the process for periodically checking for system components through Eucalyptus API to CC, SC, Walrus, and NC. The remote launcher and local launcher return information about the available system to CLC using a query.

F. Activation of the Remote Cloud Server

The remote cloud server's activation function shows remote cloud activation in the cloud environment. If a remote user requests cloud service from the cloud system, the local cloud system activates the remote cloud service function. After a recognition step between the local cloud and the remote cloud, each cloud's connection settings are established, and the remote cloud system activates cloud service. Connection is established between the user and the remote cloud system. Users can utilize cloud service in the remote cloud system.

If the user requests a service, the local cloud launcher uses the user network. After that, the system request to the remote cloud system regarding OS image information, the remote cloud and local cloud synchronize the image list and transfer image files. The local cloud system requested information from the remote cloud system about the user requested application.

The user's operating information consists of a kernel, ramdisk, and image file. The remote application launcher registers on the eucalyptus cloud system. If operating information is registered on the system, the system returns the ID values of EKI (Eucalyptus Kernel Image), ERI (Eucalyptus Ramdisk Image), and EMI (Eucalyptus Machine Image).

If the operating image file completes registration on the Eucalyptus system, the remote application launcher creates a keypair with a key value for communication with each Openstack instance. After creating the keypair, the remote application launcher requests creation of an instance on the Eucalyptus cloud system.

During generation of the instance, the Eucalyptus system uses the appropriate needed parameters like key-pair's name, EMI ID and VM type. Upon completion of instance creation, the Eucalyptus system returns the ID of the instance for registration.

If the instance is normally driven on the cloud system, the application launcher periodically checks the status of the instance. In this process, the remote application launches a connection with the instance and transfers the user's application. After this, the remote application launcher returns an IP address for the instance from the local application launcher to receive cloud services. A user re-requests the cloud service based on the IP address, which returns the remote application launcher.

Each cloud's application launchers check the CPU usage, RAM usage, and HDD usage for status information in order to manage resources.

G. Deactivation of the Remote Cloud Server

If usage of cloud resources is low, the local cloud system is deactivated from the remote cloud system, and the user requests cloud service termination. After this, the instance in operation is stopped on the remote cloud system, and the user's image and instance information is transferred to the local cloud system.

IV. FIRSTCLOUD FOR FUTURE INTERNET

A. Key Features of FiRSTCloud

In this section, we propose the FiRST Cloud AM based on GENI AM API for the cloud computing platform to extend the limited functionality of GENI Cloud project [4][6][7].

FiRST Cloud AM acts as a moderator between the OpenStack cloud and the future internet test-bed. Also, FiRST Cloud AM manages mapping from an instance to the slice when a user queries about resource allocation on the slice. Therefore, FiRST Cloud AM maintains this mapping information between an instance and a slice. To moderate between instance and slice, FiRST Cloud AM creates a database of openstack instances and slice information, as in Table II.

FiRSTCloud AM provides six APIs (all except RenewSliver()) using GENI AM API: GetVersion(), ListResource(), CreateSliver(), DeleteSliver(), SliverStatus() and Shutdown(). Additional features may exist depending on the existing API.

FiRSTCloud AM defines the RSpec which is submitted by the user to describe instance-specific information and resource information. RSpec is managed differently depending on the items after parsing. The RSpec contains items such as cloud image information that includes image id and state, key-pair, instance and vm information.

B. GetVersion() API of FiRSTCloud AM

FiRST Cloud AM returns the version of the GENI Aggregate Manager API supported by this aggregate. Version information includes the OpenStack cloud version.

TABLE II. DB TABLE OF SLICE AND INSTANCE

Slice		
Name	***Type***	***Key***
ID	INTEGER	PRIMARY KEY
Slice urn	TEXT	

Openstack Insatance		
Name	***Type***	***Key***
ID	INTEGER	PRIMARY KEY
Instance ID	TEXT	
Kernel ID	TEXT	
Image ID	TEXT	
Ramdisk ID	TEXT	
Instance type	TEXT	
Key pair	TEXT	
Slice ID	INTEGER	REFERENCES Slice(ID)

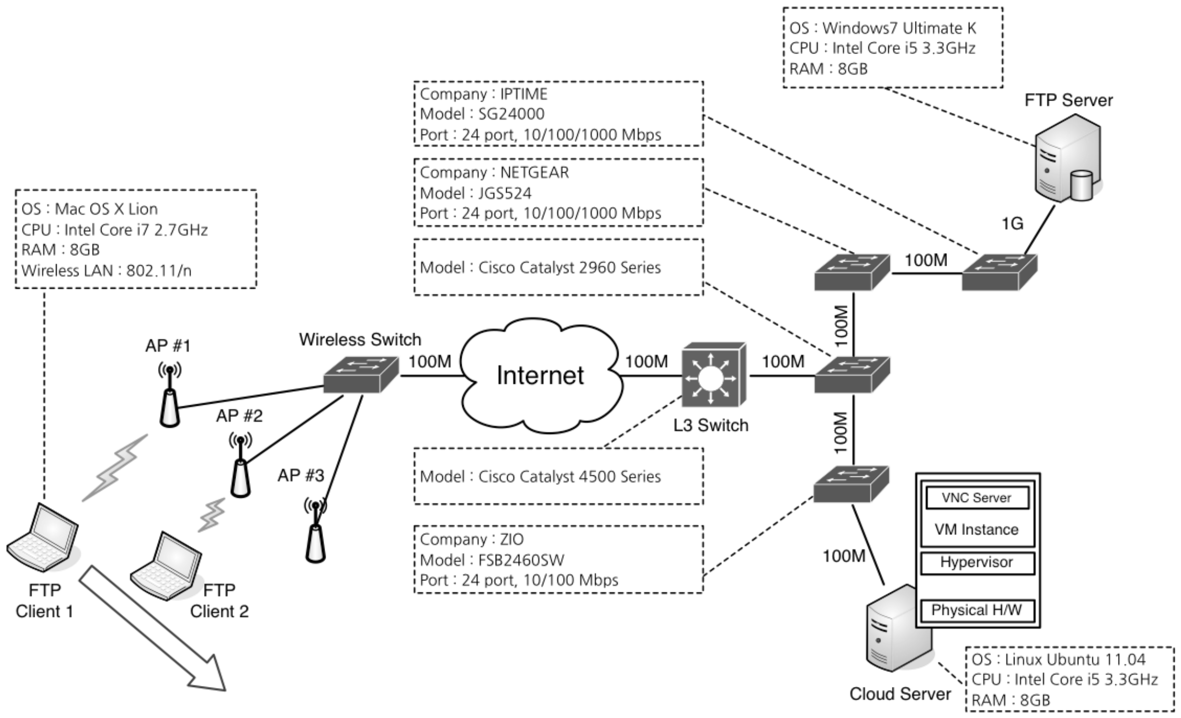

Figure 2. System environment for proposed mobile zero-client.

C. *CreateSliver() API of FiRSTCloud AM*

FiRST Cloud AM is able to allocate resources to a slice. Also, this operation is expected to asynchronously activate the allocated resources after the operation has been successfully completed.

Callers can check on the status of the resources using SliverStatus API.

To connect with OpenStack Cloud, first, use the boto library which is compatible with the EC2 proceeds. Then, initialize the database information of instance and slice. Create a new instance from RSpec by parsing the image information, virtual machine type, and keypair information. Finally, return the id of the created instance by creating a new RSpec.

D. *ListResources() API of FiRSTCloud AM*

FiRST Cloud AM returns information about available cloud resources or resources allocated to a slice. To connect with OpenStack Cloud, use the boto library which is compatible with the EC2 to connect. Then, request the available zone information, registered image information, and keypair information; instances of OpenStack Cloud AM are returned in the form of a list of values. Finally, return the cloud information by creating a new RSpec.

E. *DeleteSliver() API of FiRSTCloud AM*

FiRST Cloud AM is able to stop sliver and delete if the sliver is running. AM search instance information occurs in the DB which is mapped to slices to be deleted. If AM finds an instance, it can be terminated using the boto library, followed by a DB update.

F. *SliverStatus() API of FiRSTCloud AM*

FiRST Cloud AM is given the status of a sliver. Additionally, AM requests the connection to the OpenStack Cloud that verifies the status of the instance as well as the sliver. Returned status information is based on the instance information for the corresponding slice_urn (uniform resource name). Based on this instance information, the final status of the sliver is determined and returned to the client as in ListResources() API.

G. *Shutdown() API of FiRSTCloud AM*

FiRSTCloud performs an emergency shutdown of a sliver. This operation is intended for administrative use. In addition, this API is obtained from a database associated with slice_urn and the instance, then terminates and manages the instance.

V. PERFORMANCE EVALUATION AND ANALYSIS

A. *Key Features of the Mobile Zero-Client*

In this paper, we proposed the mobile zero-client based on cloud mobility control and FiRSTCloud AM. For mobile cloud service, we used Virtual Networking Computing (VNC) which includes a graphic desktop share system through a Remote Frame Buffer (RFB). Zero-client means end user device has no local storage and just has weak process power to communicate with server. Following evaluations, however, zero-client is substituted by common laptop running VNC viewer only. On the cloud server, there is a Linux OS installed instance to run VNC server application.

B. Performance Analysis of the Mobile Zero-Client

Network topology is constructed for mobile zero-client performance analysis as in Figure 2. In this performance analysis, we want to present mobile zero-client on mobile cloud performs better than normal mobile device.

There are two hypotheses before analyze performance of our proposal. First, we assume that the cloud mobility server and client are well operated. Therefore whole information and data in local site downloaded to remote site already.

Second, we did not consider about interworking with Future Internet test-bed in performance analysis. It's for comparison with normal mobile device and for convenience of experiment.

The performance analysis is operated by receiving a file from FTP server.

There are two different traffic cases. In the first case, no traffic is generated on the network. In the second case, another mobile device generates traffic at the second static AP. The second mobile device also downloads the same file from the FTP server. In these two network environments, the mobile device downloads a 700MB video file using 802.11n WLAN. Downloading occurs in two ways, through the use of a mobile device to download directly from the FTP server and connects to Openstack instance using VNC client as a mobile zero-client. When using mobile zero-client, the mobile device does not download from the ftp server but from an Openstack cloud to which the mobile zero-client is connected.

Table III shows the result of performance evaluation. There are four scenarios, and each row represents a scenario. Scenario 1 means the data transfer rate of mobile devices when downloading a video file from an FTP server with no traffic on the network. In this scenario, data transfer rate is unstable because of the wireless network environment. The average data transfer rate measured 29.75 Mbps. Scenario 2 means the data transfer rate of the mobile zero-client download video file from an FTP server with no traffic on the network environment. Data transfer rate is stable because the mobile zero-client received the data through OpenStack cloud instance (VM). Average data transfer rate measured 67.65 Mbps. When using a mobile zero-client, we achieve a similar performance to that of a wired network user in our wireless network environment. Scenario 3 means the data transfer rate of the mobile device when it is directly downloading a video file from the FTP server. In addition, this and next scenario correspond to the second traffic case which is mentioned. Therefore the data transfer rate little bit decreased when wireless AP was shared with another client, producing an average data transfer rate of 27.19 Mbps. Scenario 4 means the data transfer rate of a Mobile Zero-Client downloading a video file from the FTP server with traffic using another mobile device. The average data transfer rate measures 65.89 Mbps.

TABLE III. AVERAGE DATA RATE ON 4 SCENARIOS

	Traffic load	**Client type**	**Average data rate (Elapsed time)**
Scenario 1	N/A	Normal client	29.75 Mbps (193.26 sec)
Scenario 2	N/A	Zero-client	67.65 Mbps (84.92 sec)
Scenario 3	Another device	Normal client	27.19 Mbps (211.40 sec)
Scenario 4	Another device	Zero-client	65.89 Mbps (87.23 sec)

VI. CONCLUSION AND FUTURE WORK

Future Internet research emphasizes harmonization of the conventional system with Future Internet research, network virtualization, and cloud computing. Providing high bandwidth and low delay is possible, but computationally intensive services or computing operations cannot be performed. A cloud computing platform can perform many service and computation operations. Its disadvantages include limited bandwidth and highly variable latency.

Researchers have begun to test cloud computing environments in large-scale global test-bed systems. In this paper, we developed cloud mobility for mobile communication between a device and the cloud server. Second, we developed the FiRST Cloud aggregate manager (AM) based on GENI AM API for interaction between the future internet test-bed and the OpenStack cloud platform. Third, we developed a Cloud Mobility Client/Server for mobile cloud management in order to control the zero-client service. We confirmed that the proposed FiRSTCloud AM works with zero-client mobile cloud service.

Through this work, mobile cloud service was shown to have a consistent quality regardless of mobile device performance or wireless environment.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A1006620).

REFERENCES

[1] M. K. Shin, "Trend on the Future Internet Technologies and Standardization," Electronics and Telecommunications Trends, vol. 22, no.6, pp.116-128, Dec. 2007.

[2] K. H. Nam, S.J. Jeong, M. K. Shin and H. J. Kim, "Technology and Trends of GENI Control Framework," Electronics and Telecommunications Trends, vol. 25, no.6, pp.157-166, Dec. 2011.

[3] GENI white paper, "GENI at a glance", [Online] http://www.geni.net/wp-content/uploads/2011/06/GENI-at-a-Glance-1Jun2011.pdf, <link> 07.16.2012.

[4] GENI API wiki page, http://groups.geni.net/geni/wiki/GeniApi, <link> 07.16.2012.

[5] PlanetLab, http://www.planet-lab.org, <link> 07.16.2012.

[6] S. W. Lee, S. W. Han, J. W. Kim, S. G. Lee, "FiRST: Korean Future Internet Testbed for Media-Oriented Service Overlay Network Architecture," Journal of Internet Technology, vol. 11, no. 4, pp. 553-559, Jul. 2010.

[7] M. Yuen, "GENI in the Cloud", University of Victoria, 2010.

[8] Openstack official homepage, [Online] http://www.openstack.org, <link> 07.16.2012.

[9] Eucalyptus official homepage, [Online] http://www.eucalyptus.com/, <link> 07.16.2012