Research Project 1

Evaluating the Network Performance of ExoGENI Cloud Computing System

Andreas Karakannas
Andreas.Karakannas@os3.nl

Anastasios Poulidis
Anastasios.Poulidis@os3.nl

Supervisors: Dr. Paola Grosso
& ing. Ralph Koning, M.Sc.

University of Amsterdam
February 10, 2014

Abstract

ExoGENI is a federated cloud computing system that offers a multi-domain Infrastructure as a Service (IaaS) testbed for research and innovation in networking and future internet applications. It is mostly used for data intensive applications where the network performance and stability is very critical. In this research project we present a measurement study to evaluate the networking performance for both the Inter-Rack domain and the Intra-Rack domains of ExoGENI. For the Intra-Rack domains we also evaluate if the network performance is reproducible among repetitions of the same experiment when the virtual topology of the experiment is reconstructed from scratch with the same attributes. Our results show that the network performance on the Inter-Rack domain is different when the end points are in short and long geographical distance. For the Intra-Rack domains we find out that the network performance is not affected by the virtual link used for the interconnection of the virtual machines and the server upon on which the VMs lie. Moreover, our results showed us that the reproducibility of the network performance for the Intra-Rack domains is always feasible.

Contents

1 Introduction

Cloud Computing Systems, the next generation distributed systems, is a set of hardware, networks, storage, operating systems and applications that are combined together to deliver computing services and resources to the end user. Cloud Computing Systems are usually deployed through the Internet (eg. Amazon EC2) but can also be implemented within a corporate firewall completely independent from the internet, known as private cloud computing systems. The fundamental technology upon which cloud computing systems are based is virtualization. With virtualization the cloud computing systems infrastructure can be shared among multiple users according to each user needs on computing power and storage capacity, thus optimizing the usability of the physical infrastructures. One of the service models that Cloud Computing Systems offer is the Infrastructure as a Service (IaaS) model.

In Infrastructure as a Service (IaaS), the cloud computing system provides to the end user the capability to use hardware, network, and storage resources from the physical infrastructure in order to create his own virtual network and I.T systems, on which he can install his own operating system, deploy his own applications and run his own experiments. The end user has control over his virtual network operating system and applications but he does not manage or control the physical cloud infrastructures upon which his virtual network runs.

It is clear from the description of the cloud computing service models that the end user knows nothing about the physical infrastructure upon which his virtual infrastructure lies. This is an important consideration especially for IaaS where the end user wants to construct a virtual network according to his needs. If the virtual network is not efficiently mapped to the cloud computing physical infrastructure then the end user will not have the network performance that the physical equivalent would have provided. For this reason data-intensive applications that need specific network topologies to be executed correctly are difficult to be implemented efficiently in Cloud Computing Systems, especially when the cloud is distributed and not within one unique data center.

In the cloud computing landscape we see an emergence of distributed clouds where resources are spread geographically where the problem of mapping virtual topologies into the physical infrastructures is even more evident. In our research we focused on ExoGENI. ExoGENI [1] is a distributed cloud computing system that offers a multi-domain Infrastructure as a Service (IaaS) testbed that was designed to support research and innovation in networking, operating systems, distributed systems, future Internet architectures, and deeply networked, data-intensive cloud computing. As in all cloud computing system the user does not have the capability to manage and control the physical infrastructure and knows nothing about the underlying infrastructure.

ExoGENI is mostly used for data-intensive e-Science applications that require the transfer of large volume of data between the computer nodes, thus the network performance among virtual machines plays a decisive role on the performance of these applications. Specifically, data-intensive applications require specific network performance and limitation on network performance variation in order to be implemented efficiently and approximate the optimal performance. For this reason the virtual links that interconnect and transfer data among the computer nodes in ExoGENI has to be efficiently virtualized from the physical infrastructure in order to provide the requested network performance. Moreover, the stability of the network performance on ExoGENI is also very critical especially for data-intensive applications that use the network performance statistics from previous implementations to be auto-reconfigured or manually configured from the user in order to optimize their performance. Given the network performance and stability on ExoGENI the experimenter can determine whether his data-intensive application can be executed efficiently on ExoGENI.

The structure of this paper is as follow: In Chapter 1 the research questions, our approach, the related work and the scope of our research are presented followed by Chapter 2 where the ExoGENI testbed, the ORCA framework and the Flukes tool are described. In Chapter 3 the methodology and the experimental scenarios are defined. Finally in Chapter 4 the results of our experiments along with their evaluation are presented followed by the Conclusion and the Future Work that can be made subsequently to our research.

1.1 Scope

The scope of our research project aims to evaluate the network performance on the ExoGENI Cloud Computing System in order to provide quantitative measurement results that will be valuable for experimenters that use the ExoGENI for data-intensive e-Science applications. Although some simple applications will be used to evaluate the performance of our virtual network topologies, the actual implementation of data-intensive e-Science applications is out of the scope of this research project.

1.2 Research Questions

- What is the network performance on ExoGENI and how suitable is for data-intensive applications?
- Is the network performance on ExoGENI reproducible when the virtual network topologies are reconstructed from scratch with the same attributes?

1.3 Approach

Our approach on this research project is to evaluate the network performance of the ExoGENI under different scenarios. In order to evaluate the network performance of ExoGENI we wrote a simple python script that makes use of free network performance tools in order to measure and analyze the TCP/UDP throughput, round trip time (RTT) and packet loss among a simple machine/computer. We then created virtual network topologies according to our scenarios on ExoGENI using ORCA-Flukes and we evaluated the network performance of these virtual topologies by running our software on each virtual machine. In order to have a better understanding of ExoGENI we measure the network performance in a long-time period for each scenario. Finally in order to evaluate the reproducibility of the network performance of these virtual network topologies after reproducing them, we attempt to re-create them from scratch with the same attributes (image, domain rack etc) multiple times with specific interval time periods and repeated all the aforementioned scenarios experiments for a short time period.

1.4 Related Work

As far as we know, there is no scientific paper evaluating the behavior of virtual network topologies on the ExoGENI Cloud Computing Systems. There are some scientific papers in the literature evaluating the performance of the virtualized cloud computer systems networks. Specifically G. Wang et al. [24] investigated the impact of machine virtualization on networking performance for Amazon EC2 Data Center found out that virtualization can cause significant throughput instability and abnormally large packet delay variation.

Also, D. Battre et al. [25] created an algorithm that reconstructs likely network tree topologies used for data-intensive applications connecting a set of star VMs topologies in those so far opaque IaaS clouds and made end-to-end measurements to evaluate their experiments. They observed large variations in the measured delay between the individual VMs and a distinct gap between VMs being hosted on the same physical server, and those running on different servers.

Moreover, D. Battre et al. [26] studied to what extent the underlying network topology of virtual machines inside an IaaS Opaque Cloud Computer Systems can be inferred based end-to-end measurements found out that common assumptions for end-to-end measurements do not hold in presence of virtualization and that RTT-based measurements in paravirtualized environments lead to the most accurate inference results.

Finally, Bill Howe in his paper [27] talks about the importance of the reproducibility of experiments, how cloud computing can improve this and which are the remaining challenges that should be overcome.

2 ExoGENI

2.1 ExoGENI

ExoGENI is a multi-domain Infrastructure as a Service test-bed developed under the Global Environment for Network Innovations (GENI) [2][3] project and it was launched in order to support research and innovation in networking and future internet applications. ExoGENI [1] orchestrates a federation of independent cloud sites that are geographically spread (Figure 1) across the United States, Europe (Amsterdam) and Australia (Sydney) and circuit providers (Internet2,NLR,SURFnet) through their native IaaS API interfaces and links them to other GENI tools and resources thus creating a federated cloud computing system that offers a unique virtual laboratory for experimenting and evaluating future internet technologies and applications. Each cloud site is a private IaaS cloud located on a host campus and uses a standard cloud stack to manage a pool of servers. Each cloud site is linked with national research networks through programmable exchange points. The ExoGENI's layer 2 connectivity between the ExoGENI private cloud sites and the circuit providers is shown on Figure 2.The interconnection of these private clouds through national research networks to a distributed cloud computing system is what enables ExoGENI federation to operate as a federated cloud computing system.

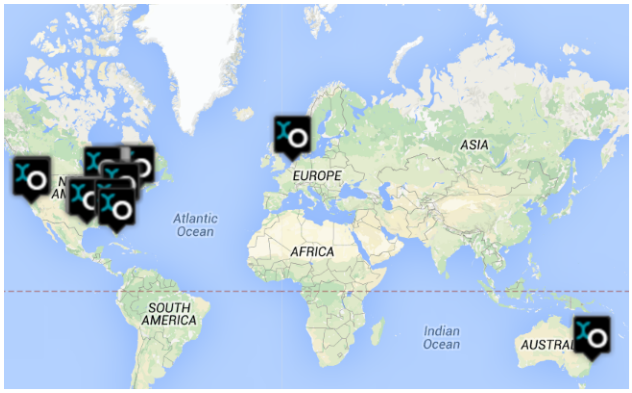


Figure 1: Geographic Location of the Indivudual Private Clouds [16]

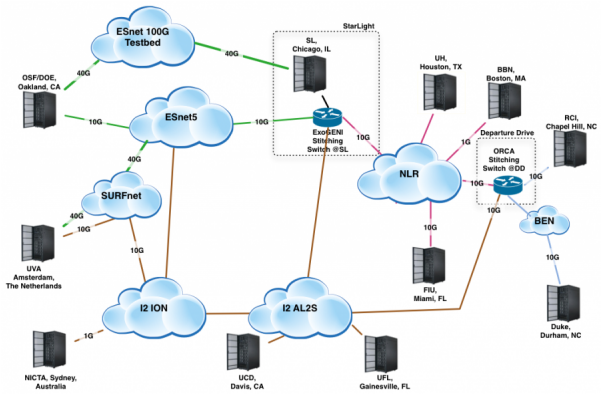


Figure 2: Layer 2 Interconnection between the ExoGENI private cloud sites and the National Research Networks [17]

The provisioning and control of the private clouds is achieved by delegation [1][5]. Each private cloud delegates certain functions for identity management, authorization, and resource management to common coordinator services offered by ExoGENI which in turn delegates, a number of these functions to the GENI federation and to identity systems operated by participating institutions.

ExoGENI test-bed software allows the users to combine resources from any private cloud within the infrastructure in order to create and manage virtual network topologies on demand. The basic operations offered by the ExoGENI testbed are [5]:

- Create, modifying and destroying virtual networks topologies consisting of compute resources that may belong to one (Intra-Rack) or more private clouds (Inter-Rack). The user can choose the private cloud in which each virtual achine (VM) of his topology belongs but he cannot specify which national research network will use to interconnect the nodes within the virtual topology. This is done automatically by the ExoGENI testbed.
- Create virtual network topologies with user-driven packet forwarding control via OpenFlow. These virtual network topologies are restricted to VLANs provisioned within and between the private cloud.
- Provisioning and control of individual compute resources (virtualized and bare-metal) from the private cloud resources. Users have the capability to load their own boot image for the virtualized instances. The boot images that can be loaded to bare-metal instances are limited.
- Create slices that combine ExoGENI resources with other GENI resources.

2.1.1 Private Clouds Architecture

Each Private Cloud also called an ExoGENI Rack is an IBM cluster (Figure 3) [12][13][14] which is consisted of 11 x3650M4 servers from which the one is configured to be the head node on which no user access is allowed and the rest 10 are configured to be worker nodes from where the user is allowed to use resources when constructing his virtual network topology. Moreover, the rack is armed with a separate iSCSI storage space of 1TB HDD for storing user OS images and saving measurement data, an 8052 1/10G management switch for provisioning and managing the rack and an 8264 10/40/100G OpenFlow-enabled dataplane switch for interconnecting the rack directly with a circuit provider either directly or through an intermediate Layer 2 provider. The interconnection between all the machines is achieved through 1/10G Ethernet infrastructure.

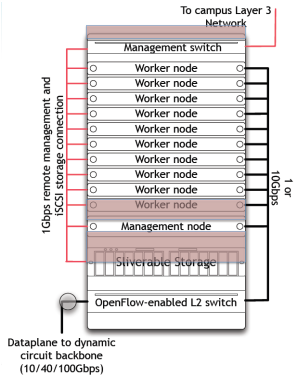


Figure 3: IBM rack [18]

2.2 ExoGENI domains and limitations

As we mentioned before ExoGENI individual domains operate as a private cloud and the layer 2 interconnections between the private clouds through circuit providers allows ExoGENI to operate as a Federated Cloud Computing System. The experimenter can construct intra-rack virtual network topologies within a private cloud and global inter-rack virtual network topologies that combine resources from multiple private clouds. Table 1 shows the ExoGENI Racks that are available on ExoGENI testbed [11].

RENCI	Chapel Hill, North Carolina, USA
BBN	Boston, Massachusetts, USA
NICTA	Sydney, Australia
FUI	Florida, Miami, USA
UFL	Florida, Gainesville, USA
UH	Texas, Huston, USA
Duke University	North Carolina, USA
SL	Illinois, Chicago, USA
UCD	California, Davis, USA
OSF	California, Oakland, USA
UVA	Netherlands, Amstedam, Europe

Table 1: ExoGENI Racks

From the above mentioned Racks only a few were provided Inter-Rack topologies due to technical problems over the Virtual Lans (VLANS) that provide the stitching between the different available racks in ExoGENI testbed. Specifically, only the network connections between NICTA and UFL were available most of the days while the connections between RENCI, UH, BBN and FUI where available only for a few days. Table 2 below provides the rack/network connection availability through our research period.

Date	RENCI	UFL	NICTA	BBN	UH	FUI	Reason	Details
12/01/2014	Yes	Yes	Yes	No	No	No	Resources	Not available VLans
13/01/2014	Yes	Yes	Yes	No	No	No	Resources	Not available VLans
14/01/2014	Yes	Yes	Yes	No	No	No	Resources	Not available VLans
15/01/2014	No	Yes	Yes	Yes	Yes	Yes	Rack and Cloud network fail	RENCI & BBN Network fail
16/01/2014	No	Yes	Yes	No	No	No	Rack and Cloud network fail	RENCI & BBN Network fail
17/01/2014	No	Yes	Yes	No	No	No	Rack and Cloud network fail	RENCI & BBN Network fail
18/01/2014	Yes	Yes	Yes	No	No	No	Resources	Not available VLans
19/01/2014	Yes	Yes	Yes	No	No	No	Resources	Not available VLans
20/01/2014	No	Yes	Yes	No	No	No	Cloud network fail	BBN Network fail
21/01/2014	No	No	Yes	No	No	No	Cloud network fail	BBN & ION Network fail
22/01/2014	Yes	No	Yes	No	No	No	Cloud network fail	ION Network fail
23/01/2014	Yes	No	Yes	No	No	No	Cloud network fail	ION Network fail
24/01/2014	Yes	No	Yes	No	No	No	Cloud network fail	ION Network fail
25/01/2014	Yes	No	Yes	No	No	No	Cloud network fail	ION Network fail
26/01/2014	Yes	No	Yes	No	No	No	Cloud network fail	ION Network fail
27/01/2014	Yes	No	Yes	No	No	No	Cloud network fail	ION Network fail
28/01/2014	No	No	Yes	Yes	Yes	Yes	Cloud network fail	BBN & ION Network fail
29/01/2014	No	No	Yes	Yes	Yes	Yes	Cloud network fail	BBN & ION Network fail
30/01/2014	No	No	Yes	Yes	Yes	Yes	Cloud network fail	BBN & ION Network fail
31/01/2014	No	Yes	Yes	Yes	Yes	Yes	Cloud network fail	ION Network fail
01/02/2014	No	Yes	Yes	Yes	Yes	Yes	Cloud network fail	ION Network fail
*/02/2014	No	Yes	Yes	No	No	No	Cloud network fail	Stitching problems

Table 2: Rack availability during the period of our research

A lot of problems were observed in BBN and ION Cloud Networks where the virtual connections and the connected racks that used these networks were affected. Also, RENCI rack was failing to be constructed due the failure of the BBN Network that connected all the adjacent racks. The most serious problem was observed in the network connections between any rack and one of FUI, BBN, HU due to the limited vlans

that were available resulting in the inability to build the dependency tree. All these problems affected the number of virtual topologies that we were be able to construct because the inter-rack topologies that can be built with 2 or 3 racks are limited.

Bearing in mind the limited resources and the technical problems that we were usually facing, only a few topologies could be built for the inter-rack slices and present them here:

- Point-to-point topology implemented by two different racks connected to each other
- Mesh/Ring topology implemented by three different racks fully connected to each other
- Star topology implemented by one rack connected with two different racks or one rack connected with two other servers of a different rack

From the above mentioned topologies we chose specific the Point-to-Point and the Star topology:

- The Point-to-Point topology was chosen because it is the simplest topology that could be built and it will help us undestand in what extent the virtualization of ExoGENI operates and how much is the network performance of data-intensive applications affected for different physical distances.
- The star topology was chosen in order to make use of data parallelism applications that is widely used in data-intensive applications and let us understand how this will affect the network performance of its.

2.3 ORCA

ORCA is a control framework and open-source platform originally developed by the NICL Lab at Duke University [6], and currently being developed jointly with Duke University by the Networking Group [7] at The Renaissance Computing Institute (RENCI) [8]. ORCA is implemented as a webapp intended to run inside a Tomcat Java servlet engine. Most of its code is written in Java, although its tools that are used to speak with GENI are written in Python.

Orca is used to provision virtual networked systems by managing distributed heterogeneous resources over a shared substrate. This substrate may contain nodes as servers, nodegroups as a collection of servers, storages, network links and other components.

ORCA is the software used to control the resources on GENI/ExoGENI testbed. It primarily uses OpenStack to provision the virtual machines but some older racks use Eucalyptus for the same reason. ORCA is also integrated with xCAT to support baremetal node provisioning.

The ORCA deployment is a dynamic collection of interacting control servers (actors) that work together to provision and configure resources for each guest according to the policies of the participants. ORCA uses leases to provide resources to their participants. In order to make this possible needs a resource provider, a consumer and one more brokering intermediaries. Each server (actor) can manage a large number of leases consumed by different participants which are independent and may have different boot images, resources reserved from different racks and/or different resource types.

2.3.1 Terminology & Operation

Substrate - "a collection of resources under specific administrative ownership."
Sliver - "a smallest unit of some independent resources."
Slices - "groups of multiple slivers from multiple substrate providers."

The basic actors that are used in all emulations are three:

- **Authority or Aggregate Manager (AM)** - "controls access to some subset of the substrate components."
- **Slice/Service Manager (SM)** - "is responsible for creating, configuring, and adapting one or more slices."
- **Broker** – "act as go-between resource discovery and arbitration by controlling the scheduling of resources at one or more substrate providers over time."

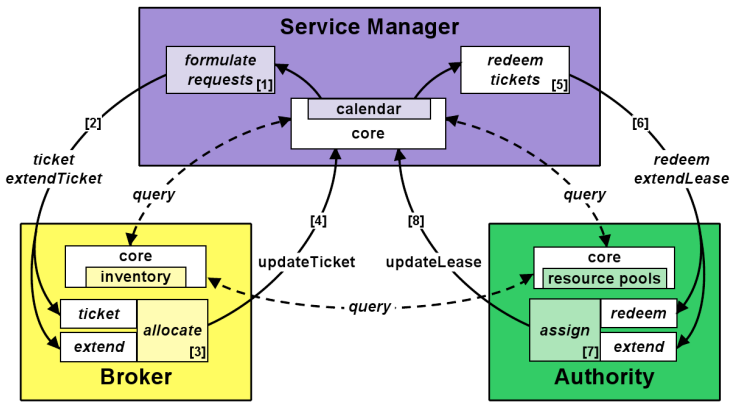


Figure 4: ORCA Operation [19]

Authorities delegate resources to brokers. Brokers hold the promised resources until SMs request them. Brokers issue tickets for resources from different authorities to an SM, which redeems those tickets at AMs. AMs instantiate the resource slivers and pass the control of them to the SM. Pluggable control and access policies help mediate these transactions. Query interfaces allow actors to query the state of other actors [9].

2.3.2 ORCA deployment in ExoGENI

Each rack in ExoGENI runs its own Slice Manager actor that acts as GENI AM and exposes ORCA native API (Flukes) and GENI AM API (Omni). From the one hand, each local Rack SM can only create slices with resources within that rack and cannot use the resources from the other Racks in ExoGENI testbed. It runs interdependently of the other racks and cannot communicate with them. From the other hand, ExoSM has global visibility of the racks and can access resources from all the available racks of the ExoGENI testbed. Thus, can create topologies using resources from different racks and using network backbone resources can stitch them together creating a distributed network [9]. The ExoSM allows experimenters to select in which ExGENI Rack each virtual machine of his topology will be placed but does not allows to configure in which worker node inside the ExGENI Rack each virtual machine will be placed. Moreover, the user is allowed to configure the desired bandwidth of the virtual connections between his virtual machines but he is not allowed to configure which ExGENI Rack virtual links will be used to connect virtual machines that lie on the same Rack and which circuit providers and VLANS (inside these circuit providers) will be used for interconnecting virtual machines that lies in different ExGENI Racks.

2.4 Flukes

Flukes is a java-based graphical tool for submitting and creating topologies by requesting them to ORCA, managing and editing the substrate resources and graphically inspecting the ExoGENI resources. It uses ORCA interfaces and NDL-OWL to describe the resources, allows the user to submit and inspect requested topologies in a graphical environment called “manifest” and gives the ability to login at the selected image of each node and configure it at any time [10].

2.4.1 Features

Flukes allows the user to create his desired topology by reserving ExoGENI resources including [10]:

- **Nodes** that play the role of VMs in the topologies and they are specified by selecting their size that is related to the resources that will be reserved, the desired OS using one of the available images and the rack from where the resources will be reserved. Also, they use IP addresses which have access over the Internet as well as interfaces connecting them with other nodes via Layer 2 circuits. Moreover, nodes can have dependencies on other nodes and post boot scripts that is being executed immediately after the instance boots.

- **Nodegroups** that is a set of nodes with common attributes which have all the functionalities of ordinary nodes with the only limitation that uses the same vlan when it is connected to other Nodegroups/Nodes.

  Flukes allows the automatically binding of Nodes and Nodesgroups in specific domains (racks) or selectively node binding to different domains, provisioning inter-domain links on demand among them and stitching them to built the requested slice.

- **Links** are the Layer2 circuits that connects the nodes in the topology. There are two types of links, the point-to-point and the broadcast. The point-to-point is used to connect the nodes/nodegroups and the broadcast can be used to create isolated vlans inside a nodegroup or to connect multiple nodes together.

- **Stitch Ports** that is being used to connect a slice that was created using resources from ExoGENI to a physical external network that is not part of ExoGENI.
- **Storage** that is an iSCSI volume assigned from the same rack that the node is reserved and attached to it after being formatted to the desired specifications.

  Flukes allows the experimenters to create their own filesystem, kernel and ramdisk images to be used for a slice or use the available ones. Also, it allows to affix your OpenFlow controller to your slices and create your own SDN.

2.4.2 Getting Started

The first step in order to create slices in ORCA-Flukes is to generate a public and a private key and a certificate through GENI Portal that will be used for authorization. Next, you have to configure the Flukes by editing the "$HOME/.flukes.properties" file adjusting the desired options and save the keys and certificate to "$HOME/.ssh" and "$HOME/.ssl" respectively . Finally, the user has to select a Slice Manager/controller that will allow him to reserve resources from specific rack(s) that the SM supports. ExoSM supports domain binding of nodes or node groups to different provider domains based on resource availability while the other SMs supports only to specific provider domains [10].

Flukes follows a simple workflow to construct slices and inspect/experiment on them [10]. The various steps are described here and shown in the Appendix:

- Create or load existing slices A.1
- Set Name, Node Type, Image, Domain, Dependencies and IP addresses manually on each node or use "Auto IP" A.2
- Reserve your topology for specific time A.3
- Type the name of the slice A.4
- Submit request and observe the reserved resources that are listed A.5
- Switch to Manifest view and select your slice from 'My Slices' or type the name of the slice and "Query for Manifest". A.6
- Continue querying for manifest until slice is ready, meaning that all the resources were reserved. A.7
- Inspect or login on each node of your topology and start your experiments. A.8

3 Experiments

In this chapter, we describe our experiments methodology and our experiment scenarios along with the corresponding experiments based on our motivation explained just before 2.3.

3.1 Experiments Methodology

In this section, we describe the methods we used for our measurements. Firstly, we describe the attributes we selected for each construction of the virtual topologies on ExoGENI. Then, we describe the tools we used for our measurements and finally the experiments setup. Before that we explain the basic metrics for evaluating the network performance, what are the expected results and what factors affect the network performance on a virtualized environment.

3.1.1 Network Performance

In computer networks, network performance is defined as measurements for estimating the service quality of a telecommunication connection. The basic metrics for evaluating the network performance are bandwidth, throughput, round trip time (RTT) and packet loss.

- Bandwidth is the theoretical maximum rate of information that can be transferred through the physical channel that connects two end points and it is measured in bits per second (bps).
- Throughput refers to the actual rate of information that is transferred through the physical or logical link. It is also measured in bits per second. Throughput depends on the link bandwidth and the additional overhead that each layer protocol adds. For this reason, throughput is always a proportion of the maximum bandwidth of the link, meaning that if you have a channel with X bps bandwidth you will get $(X - Overhead)$ bps throughput.
- Round trip time (RTT) is the time required for a data packet to travel from a specific source to a specific destination and back again.
- Packet Loss is the number of data packets send across a computer network and failed to reach their destination.

3.1.2 Expected Results

- **TCP/UDP Layers Overhead:** Layers overhead refers to the extra information that is added in each layer and it is send along with the data. In ExoGENI, the Ethernet technology with VLAN tags is used on Data Link layer thus an overhead of 42 bytes is added on the data. On the Network Layer the Internet Protocol version 4 (IPv4) was used during our measurements, thus an overhead of 20 bytes was added on the data. On the Transport layer, TCP and UDP add an overhead of 20 and 8 bytes respectively. In order to get the maximum TCP/UDP payload data rates we used 1500 bytes of packets/datagrams (the maximum transmission unit allowed by the Ethernet Technology) during our measurements. The overall overhead rate along with the maximum TCP/UDP payload rate [28] that could be achieved during our measurements are shown on Table 3.

Transport Layer Protocol	Maximum Payload Data Rate (%)	Overhead Rate (%)
TCP	93.9	6.1
UDP	95.4	4.6

Table 3: Transport Layer Protocols Overhead

During our experiments, we used 10Mbps and 100Mbps virtual links, thus the maximum TCP and UDP throughput that we expected is shown on Table 4 and 5 respectively.

Maximum TCP Throughput	9.39Mbps
Maximum UDP Throughput	9.54Mbps

Table 4: TCP/UDP Throughput Expected Results for 10Mbps Virtual Links

Maximum TCP Throughput	93.9Mbps
Maximum UDP Throughput	95.4Mbps

Table 5: TCP/UDP Throughput Expected Results for 100Mbps Virtual Links

- **Round Trip Time(RTT):** RTT value is affected by the geographical distance of the two end points. During our measurements we expect to see RTT values that will be analog to the geographical distance of the two end points.
- **Packet Loss:** In a reliable connection packet loss rarely occur ($< 0.1\%$). Since we are sending UDP traffic with the maximum virtual link bandwidth we expect to see a low packet loss rate ($< 0.1\%$).

3.1.3 Virtualization Overhead

In a non-virtualized machine [23] the Operating System (OS) runs on bare-metal hardware. The communication of the machine with other machines is achieved through the network interface card (NIC). The OS uses the TCP/IP stack and the NIC driver in order to pass the application data to the NIC which in turn transmits this data to the network. In a virtualized environment, the virtual machine operating system runs on top of a VM monitor. In this case the VM OS does not communicate with the NIC directly through the NIC driver. Instead, the VM has a virtual NIC which emulate NICs' functions and resources in software. This virtual NIC are completely decoupled from NIC hardware. The communication between them is achieved through a software switch implementation that forwards the traffic from the virtual NIC to the hardware NIC and vice versa. This adds extra layers of packet processing and overhead. ExoGENI virtualization technology is based on KVM and VMWare hypervisors [20][21]. These hypervisors run on bare metal and achieve a lower portion of virtualization overhead. However, the virtualization overhead is not completely removed in ExoGENI, thus depending on the implementation of the virtualization mechanisms on ExoGENI we expect to see or not the impact of virtualization on our measurements.

3.1.4 Resources Attributes

Each ExoGENI resource has some attributes that can be configured by the user from a pool of available values. The resources that we use for constructing our topologies are virtual machines and virtual links. The attributes that can be configured by the user for the aforementioned resources are shown on Table 6.

Virtual Machine	Name	Type	Image	Domain	Dependencies
Virtual Link	Name	Bandwidth	Label Tag		

Table 6: Resource Attributes

For our virtual machines attributes we selected the following values:

- The Name of the Virtual Machine is just used for identification by the user so not any parameters affected this value.
- For the Virtual Machine Type we selected XOsmall Virtual Machines. XOsmall are virtual machines that have 1 dedicated CPU 1GB of RAM and 10GB of hard disk space. This choice was based upon the fact that the tools used for network performance measurements do not need a powerful compute machine.

- No desktop Ubuntu 12.04 was chosen as the OS that will run on our Virtual Machine. No desktop Ubuntu 12.04 recommended minimum system requirements [15] are 64MB of Ram and 1GB HDD thus the specific OS can run smoothly on our selected VM type. Moreover, the measurement tools that we used were supported by this image.
- The domain on which the VM was located was selected according to the needs of the specific experiment.
- Dependencies between nodes where not need for our experiments thus no dependency were selected.

For our Virtual Links attributes we selected the following values:

- The Name of the Virtual Link is just used for identification by the user so not any parameters affected this value.
- The bandwidth was set to the maximum that the system available resources allowed us. Specifically for inter-domain virtual links the system available resources allowed us a maximum of 10Mbps and for intra-domain virtual links 100Mbps.

3.1.5 Measurements Tools

The measurements used to evaluate the network performance on ExoGENI included TCP/UDP throughput, Packet Round Trip Delay (RTD) and packet loss. For extracting these measurements we developed a simple python script that makes use of network performance measurement and network packet analyzer tools. The tools used are described below.

- Iperf was used for measuring the TCP/UDP throughput and Packet Loss. Iperf is an open source software that uses the client server model in order to measure the network performance between end points. By default iperf used the TCP Window scaling and fills TCP packets over the connection as fast as possible, thus it measures the maximum TCP throughput of a connection. It also allows to set up the target bandwidth for UDP connections thus it can measure the maximum UDP throughput of a connection. Moreover, it measures packet loss and the packet loss rate of a UDP connection. For these reason it was chosen our measurement tool for the aforementioned metrics.
- Ping was used for measuring the round trip time (RTT) between our VMs. Ping sends Internet Control Message Protocol Packets (ICMP) echo request packets to the destination, waits for the response and reports the minimum, maximum and mean RTT. It can be configured to measure RTT in specific time intervals for specific amount of time.
- Tcpdump is an open source network packet analyzer tool that gives the user the ability to observe the packets traversing through the network on which a machine is connected. We used tcpdump for observing the network traffic of our virtual links.
- Iostat and Top are computer system monitor tools that collect data about a computer I/O devices and processor and present statistics for these metrics. We used these tools for checking whether the behavior of the VM CPU and I/O devices affected the performance of our network.

3.1.6 Experimental Setup

For the first 4 experiments we set up on ORCA-Flukes the topologies described above. In order to have a better idea of the network performance we conducted each measurement for a long time period as shown on Table 7. TCP/UDP and Packet loss were measured using Iperf and RTT was measured using ping. For TCP connections we had TCP Window scaling enabled and for UDP we configured iperf in order to send UDP traffic equal to our virtual link bandwidth. These configurations were done in order to be able to get the maximum TCP/UDP throughput. During these measurements we also had Tcpdump collecting information about the network packets traversing through our virtual connections and Iostat and Top

Metric	Repetitions(times)	Time(second)	Interval Between Measurements
TCP Throughput	100	60	10 min
UDP Throughput	100	60	10 min
Packet Loss	100	60	10 min
RTT	100	60	10 min

Table 7: Experiment 1-4 Measurement Configuration

collecting CPU and I/O devices usage statistics from each VM. This procedure was automated with a python script that was running on every virtual machine of our topology.

For our last experiment (Exp. 5) we repeated each one of the previous experiments 100 times with an interval of 10 minutes between each experiment repetition. Every time one experiment was repeated, it was reconstructed from scratch and run for a small time period as shown in Table 8.

Metric	Repetitions	Time(second)	Interval Between Measurements
TCP Throughput	100	20	5 min
UDP Throughput	100	20	5 min
Packet Loss	100	20	5 min
RTT	100	20	5 min

Table 8: Experiment 5 Measurement Configuration

For this experiment we used OMNI which is another client tool that uses scripts for interacting with ExoGENI. It has a command-line interface through which the user can make use of the scripts to create, delete, modify and get status of his topology. In order to automate this experiment we developed a simple python script that uses OMNIs scripts and repeated all the previous experiments 100 times.

Using the above measurement metrics as we described above give us reliable results for evaluating the network performance of ExoGENI.

3.2 Experimental Scenarios - Experiments

In order to evaluate the network performance of ExoGENI we conducted a number of experiments based on four different scenarios that emerged from ExoGENIs architecture. In this section we describe our scenarios and the experiments that we conducted for investigating each one. Table 9 shows an overview of our experimental scenarios and their purpose.

Cloud	Scenario	Topology/Communication	Purpose
Inter-Racks	1	Point to Point	Short/Long Distance End Points Network Performance
Inter-Racks	2	Star/Point to Multipoint	Competition Upon Circuit Provider Physical Infrastructure
Intra-Racks	3	Point to Point	Same/Different Worker Node End Points Network Performance
Intra-Racks	4	Star/Point to Multipoint	Competition Upon ExoGENI Rack Physical Infrastructure
Both	5	All	Reproducibility of Network Performance

Table 9: Experimental Scenarios Overview

3.2.1 Scenario 1 - Experiment 1

- **Scenario:**ExoGENI Racks are distributed geographically among the world from United States to Netherlands and to Australia. On this scenario we aim to evaluate the network performance of a virtual link that connects VMs from ExoGENI Racks that are in a short geographical distance (eg. within US) and in long geographical distance (eg. US – Australia). Virtual links that connect short geographical distance ExoGENI Racks are expected to have better network performance than virtual links that connect ExoGENI Racks that have long geographical distance. We investigate this scenario to find out whether our hypothesis is true.

- **Experiment:**In order to investigate scenario 1 we constructed the virtual network topology shown in Figure 5. This virtual network topology connects one VM located in ExoGENI Rack A with a VM located in ExoGENI Rack B. In the first case the virtual link connects two VMs that lie on ExoGENI Racks that have long geographical distance whereas the second case the virtual link connects VMs that lie on short geographical distance. After constructing this topology, we performed end-to-end network performance measurements for both cases.

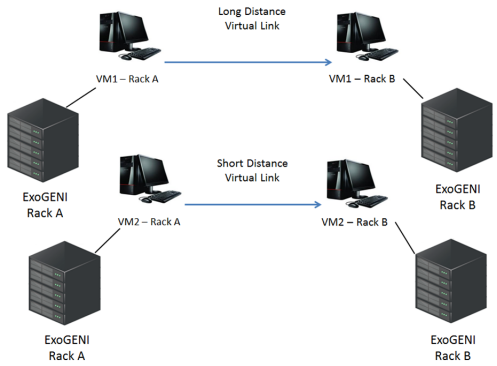


Figure 5: Experiment 1

3.2.2 Scenario 2 - Experiment 2

- **Scenario:**ExoGENI Racks are inter-connected through virtual lans provided by circuit providers. With this scenario we intent to investigate whether two or more virtual links that connects VMs on ExoGENI Rack A with VMs on ExoGENI Rack B are competing upon the underlying network.
- **Experiment:**To investigate the behavior of the network performance of Scenario 2 we constructed the virtual network topology shown in Figure 6. This virtual network topology connects VM1 that is located on ExoGENI Rack A to VM1 and VM2 located on ExoGENI Rack B through virtual links 1 and 2 respectively. VM1 on Rack A sends simultaneously network traffic to VM1 and VM2 on Rack B. By implementing this we can conclude if the competition upon the physical underlying network exists or not.

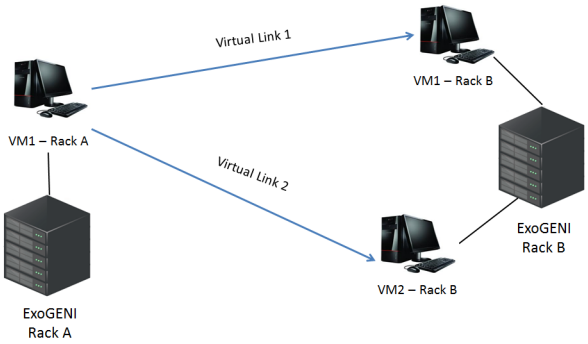


Figure 6: Experiment 2

3.2.3 Scenario 3 - Experiment 3

- **Scenario:**On ExoGENI the experimenter is allowed to construct a virtual network topology in which all the resources are derived from the same ExoGENI Rack. The problem here is that the experimenter is not allowed to choose whether his VMs lie on the same server inside the Rack or in separate servers. When VMs lie on the same server the virtual link is implemented upon the servers' bus, thus it is expected to have better network performance comparable to VMs that lie on different servers on which the virtual link is implemented upon the Ethernet infrastructure that connects the servers inside the Rack. In this scenario we investigate this hypothesis.
- **Experiment:**The virtual network topology shown on Figure 7 was constructed in order to examine Scenario 3. This virtual network topology connects two VMs located in ExoGENI Rack A. In the first case the virtual link connects two VMs that lie on the same worker node whereas the second case the virtual link connects VMs that lie on separate worker node. After constructing this topology, we performed end-to-end network performance measurements for both cases.

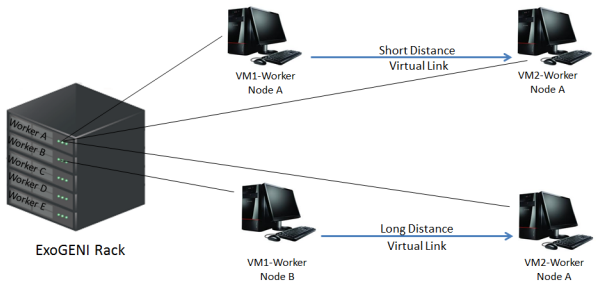


Figure 7: Experiment 3

3.2.4 Scenario 4 - Experiment 4

- **Scenario:**This scenario goal is to examine whereas two or more virtual links that are implemented on the Ethernet infrastructure that connects the machines on the ExoGENI Rack are competing upon the underlying network.
- **Experiment:**For investigating Scenario 4 we constructed the virtual network topology shown in Figure 8. In this topology the VM of worker node A is connected with 3 VMs that are on different worker nodes among them and the VM of worker node A. VM on worker node A sends simultaneously network traffic to all the other VMs. By implementing this we can conclude if the competition upon the physical underlying network exists or not.

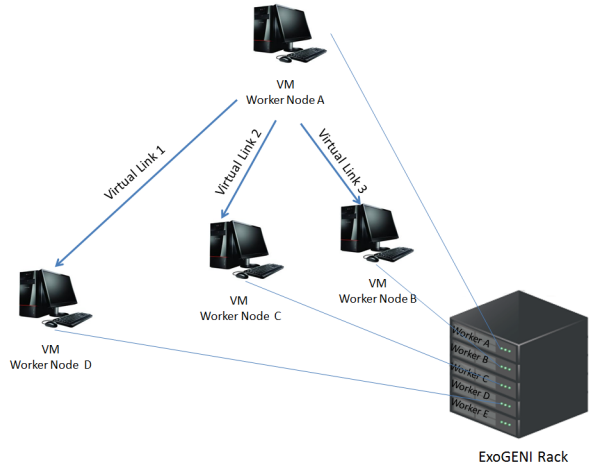


Figure 8: Experiment 4

3.2.5 Experiment 5

Reproducibility plays a big role to experimenters and researchers. They ideally want a system which will allow them to repeat their experiment a lot of times and achieve the same results and thus the overall performance of the system to be stable and accurate. The ExoGENI Cloud System must perform the virtualaziation of the physical machines and connections in that way that will allow the achievement of these demanding results without unexpected behaviours. It must also allow the experimenters to trust the system in that way that they can relay on the results of an experiment that will be used as a guideline for other more complicated experiments where anything can go wrong and the system should not be their concern. Finally, the network performance must be stable in order to allow data intesive application to perform well and in an expected way. The purpose of our final experiment is to evaluate the network performance reproducibility on ExoGENI. In order to evaluate the network performance reproducibility of ExoGENI we will repeat all the aforementioned experiments multiple times with a specific interval for a long time period. Each time an experiment is repeated it is re-constructed from scratch. A re-constructed experiment follows that the virtual network topology used for that experiment will have different values on some attributes that the experimenter cannot configure. Virtual links used for the interconnection of the VMs and the worker node inside an ExoGENI Rack upon which a VMs lie are parameters that may vary among re-constructed from scratch topologies. By applying these experiments we are able to get insights on the variation of network performance and we are able to characterize the reproducibility of the network performance.

4 Results & Evaluation

In this chapter, we present and analyze the results of our experiments. Firstly, we present and analyze the results for RTT and TCP throughput and then the results for UDP and packet loss together. As we mentioned on the experimental methodology section on the previous chapter, UDP throughput and packet loss were measured using iperf. Due to the fact that when you do a UDP connection with iperf you get results for both the UDP throughput and packet loss we intentionally present and analyze these two metrics together since they are timely related in our experiments. Finally, we have a discussion about the unexpected packet loss rate that occured in all our experiments.

4.1 Experiment 1

For experiment 1 scenario we manage to gather results for 5 end-to-end points pairs as shown on Table 10.

A/A	End Point 1	End Point 2	Distanes
1	RENCI	NICTA	Long
2	UFL	NICTA	Long
3	RENCI	UFL	Short
4	BBN	NICTA	Long
5	BBN	UH	Short

Table 10: Experiment 1 Pairs

Due to the unavailability of resources on ExoGENI the experiments for the pairs 1,2 and 3 were conducted in different time periods than the experiments for pair 4 and 5. Between these time periods maintenance was performed on the ExoGENI.

4.1.1 RTT

The measurements of RTT for all the inter-rack point-to-point connections are shown in Figure 9. By analyzing the graphs we observed that the Average RTT of the long distances (UFL – NICTA, BBN – NICTA, RENCI - NICTA) are way higher than the Average RTT of the short distances (RENCI – UFL, BBN – UH) connections as we were expecting (3.2.1). This is the result of the time required for the ICMP packet to travel to a distance of 15500, 15000, 16258 and 931, 2584 km respectively. The problems that these connections encountered was that in a few time intervals there was a number of high RTT events (up to 20 times higher than average) that was correlated with the increased packet loss rates over the period of 10seconds with the result of increased average RTT. Also sometimes, we observed a 'destination host unreachable' event due to the inability to complete the ARP resolution of the destinations address. These events which happened prior to the maintenance was only a few and led as to the conclusion that there was a temporarily congestion over the underlying connection which didn't affect the overall RTT of the connection. Moreover, these events weren't representative of the actual RTT due to their nature and they were removed from Table 11 values. In Figure 9 we observe the cumulative distribution of our short distance point-to-point connections that is 4-5 smaller compared with the one of the long distance. This is the result of the lower Average RTT of the small distances over the long distances.

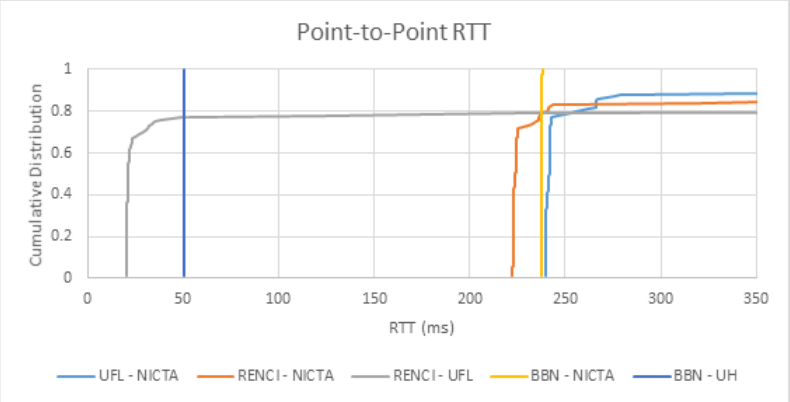


Figure 9: Experiment 1 - RTT CDF

The point-to-point measurements of the our topologies that include RENCI, NICTA and UFL racks was made prior to an important maintenance that was being held in during our research. But, the measurements made on BBN, NICTA and UH racks was being made after the maintenance. As you can see the previous network problems we encountered was fixed and the connection between these racks was more stable and the RTT values as well as the Standard Deviation (Table 11) was as expected.

Connection	Average RTT (ms)	Standard Deviation (ms)
RENCI - UFL	23,767	10,392
BBN - UH	50,424	0,030
RENCI - NICTA	225,047	12,986
UFL - NICTA	243,785	8,336
BBN - NICTA	237,869	0,070

Table 11: Experiment 1 - Average RTT & SD

4.1.2 TCP Throughput

Figure 10 shows the cumulative distribution of TCP Throughput among all the point to point connections. From these results we can see that the long distance end to end point connections achieve on average 9,3 Mbps TCP throughput whereas the short distance connections achieve on average 8,2 Mbps (Table 12). Moreover the standard deviation of the short distance pairs is times smaller than the standard deviation of the long distance pairs. These results come in agreement with our initial hypothesis that the network performance on short distance connections is better than in the long distance connections In addition, in this experiment we observed two unexpected measurements from RENCI to UFL (Figure 10). Specifically, in two measurements the TCP throughput falls below 8 Mbps. We couldn't get any insights on these two measurements from tcpdump and iostat analysis so we conclude that during these two time periods the underlying link may had congestion that resulted in packet loss, packet delay and retransmissions.

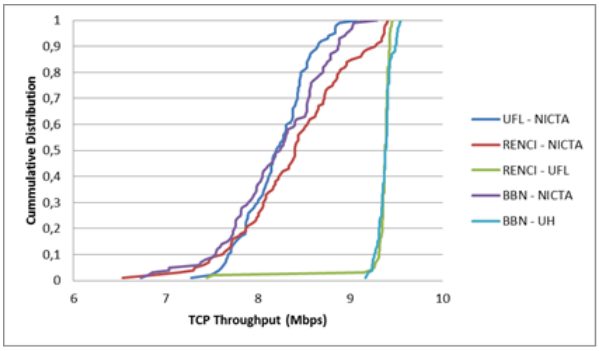


Figure 10: Experiment 1 - TCP Throughput CDF

Connection	Average TCP Throughput(Mbps)	Standard Deviation
RENCI - UFL	9,3384	0,271166
BBN - UH	9,3787	0,07804
RENCI - NICTA	8,386	0,607711
UFL - NICTA	8,2014	0,364523
BBN - NICTA	8,1979	0,542064

Table 12: Experiment 1 - TCP Throughput Results

4.1.3 UDP Throughput – Packet Loss

During our experiments for measuring the UDP throughput and Packet Loss for the first three pairs before the maintenance of the system we observed the abnormal behavior shown on Figure 11. For these measurements the UDP throughput had an average of 9.5Mbps on the first two measurements but from the third measurement and on UDP throughput dropped to approximately 1% of the link bandwidth. That was due to the packet loss that increased from 1-2% to 99%. From our insights we couldn't find a logical explanation for this behavior so we conducted the IT Engineers of ExoGENI who also observed this behavior of UDP and they couldn't find a good reason for that. But, after a maintenance they performed on ExoSM Slices, this abnormality was solved as we will show later.

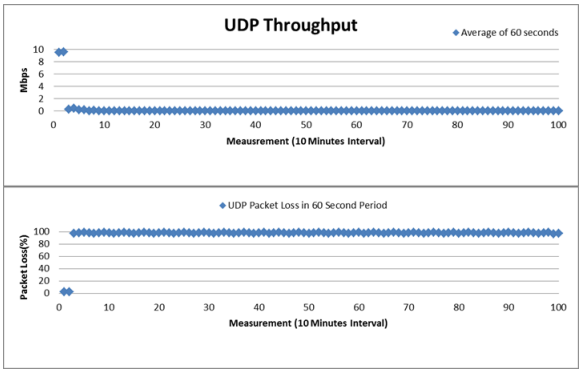


Figure 11: Experiment 1 - UDP Throughput - Packet Loss

Experiment 1 for pairs 4 and 5 was performed after the aforementioned maintenance. Table 13 shows the average UDP throughput along with its standard deviation and the average packet loss. Figures 12 and 13 show the average UDP throughput and the percentage of packet loss for each one of the 100 measurements. From these we can observe that the UDP throughput and packet loss for both short and long distance connections is approximately the same. This comes in contrast with our initial hypothesis and our findings about the TCP throughput for short-long distance relation. However, in TCP connections the sender has to wait for an acknowledgment (ACK) before sending the next packet where in UDP connection the sender sends packets continuously without having to wait for an ACK. These ACKs introduce more delay for every packet send via a TCP connection so we conclude that the different outcomes for TCP/UDP throughput are related to this factor. It is also important to mention that the UDP throughput from BBN – UH has three specific measurements where the average UDP throughput is below 8 Mbps due to higher packet loss rates on these time intervals. After analyzing the tcpdump and iostat for that time periods we couldn't observe something that could be related to this, thus we conclude that the underlying link was overloaded and introduced this high packet loss rates. On the contrary, the UDP connection from BBN – NICTA has no unexpected behavior in any measurement and can be considered more stable.

Connection	Average UDP Throughput(Mbps)	Standard Deviation	Average Packet Loss(%)
BBN - UH	9,3787	0,634186	5,456
BBN - NICTA	9,6011	0,078040	3,966

Table 13: Experiment 1 - UDP Results

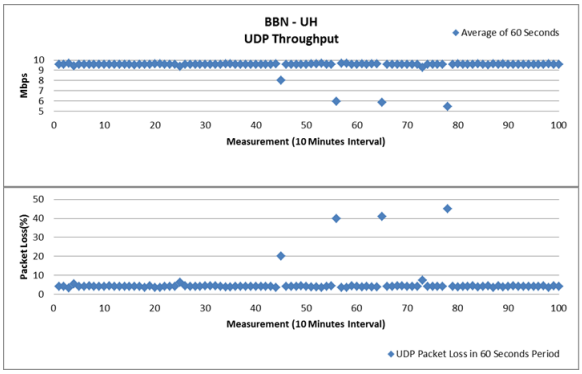


Figure 12: BBN-UH UDP Throughput- Packet Loss Results

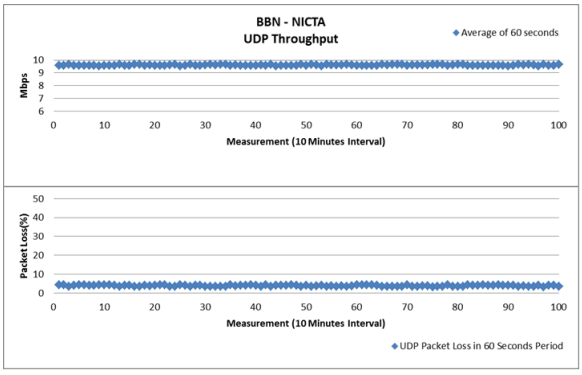


Figure 13: BBN-NICTA UDP Throughput - Packet Loss Results

4.2 Experiment 2

For this experiment we manage to gather results between BBN and UH ExoGENI Racks. Specifically, we use one Virtual Machine from BBN Rack to generate network traffic simultaneously to 2 Virtual Machines that where both located in UH Rack. Our results are presented and analyzed below.

4.2.1 RTT

As shown on table 14 the average RTT has a value close to 50ms (50,85514) for the connection between BBN and UH1 and also a value close to 50ms (51,07125) for the connection between BBN and UH2. In both graphs the Average RTT is stable without any significant malfunctions and its value is comparable with the value of the previous topology, between the same Racks. The existence or not of the competition that we investigate will be shown during our TCP and UDP throughput measurements.

Connection	Average RTT (ms)	Standard Deviation (ms)
BBN - UH1	50.85514	0.778143842
BBN - UH2	51.07125	0.817486384

Table 14: Experiment 2 - RTT Results

4.2.2 TCP Throughput

As shown on Figures 14 we observe an unexpected behavior on TCP throughput for both connections. For the connection BBN – UH 1 and BBN – UH 2 the average TCP throughput is 1.34 Mbps and 1.38 Mbps respectively. This unexpected behavior on the TCP throughput could not be explained, so in order to further investigate this behavior we reconstructed the topology and performed another experiment. We generated TCP traffic from BBN to UH1 for 20 minutes and on the middle of this interval we started generating TCP traffic from BBN to UH2.

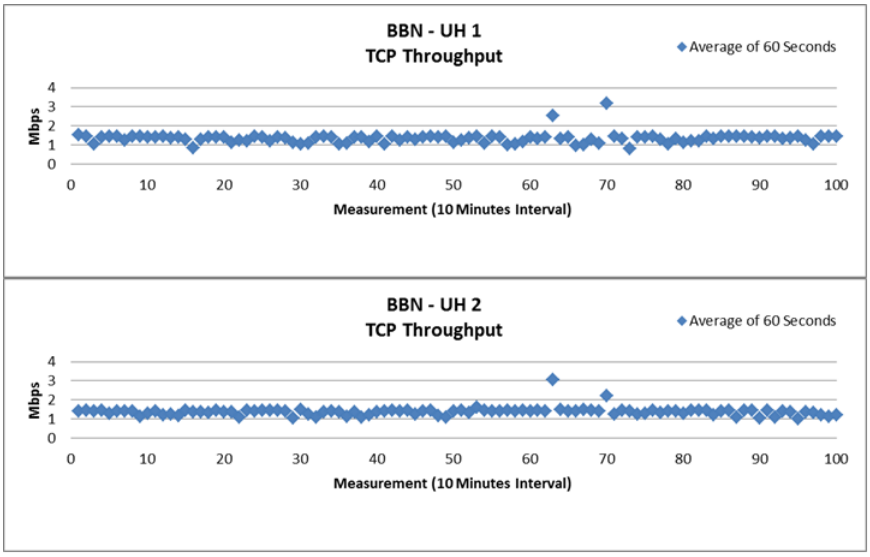


Figure 14: Experiment 2 - TCP Throughput

The results of this experiment as shown on Figure 15 didn't give us any insights on what could be the reason that affected our initial experiment unexpected results. Specifically, when the VM on BBN starts sending data to the VM 2 on the UH Rack the TCP throughput of the connection BBN UH 1 remains stable and the connection BBN – UH2 achieves approximately the same TCP throughput with the connection BBN – UH1. The results of this experiment were as expected thus we can conclude that the unexpected behavior of our initial experiment was not related to competition upon the underlying network. Moreover, the results of this experiment showed us that for TCP connections there is no competition upon the physical infrastructure.

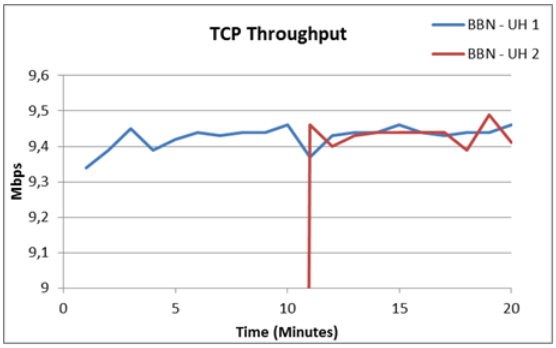


Figure 15: Experiment 2 - Repeated

4.2.3 UDP Throughput - Repeatability of Experiment 2

UDP throughput results shown on Figure 16 can be characterized as normal. The average UDP throughput is 9.6 Mbps for both connections. These results imply that there is no competition between the two virtual connections on the underlying network.

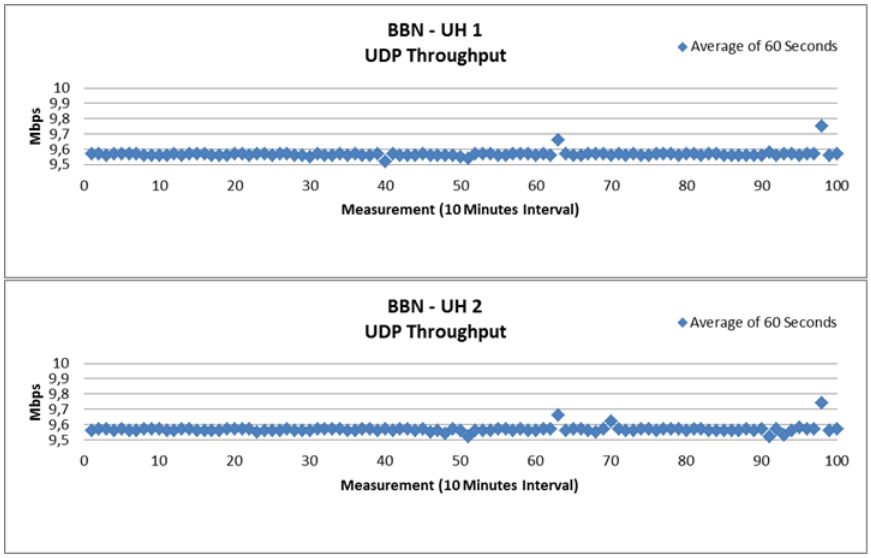


Figure 16: Experiment 2 - UDP Throughput

4.3 Experiment 3

We implemented Experiment 3 on UFL and UH ExoGENI Racks. Since all ExoGENI Racks have the same architecture and hardware we believe that our results should be the same for all the ExoGENI Racks. Below, we present and analyze these results.

4.3.1 RTT

The results of our measurement can be shown here. We observed an average RTT of 0.75738 in the first long-distance connection between different servers and an Average RTT of 0.33727ms in the short-distance connection in the same server. From the one hand, the differences between short and long distances are not big, but this behaviour is normal because we investigating connections between servers in the same physical machine. From the other hand, the dissimilar RTT between the short and the long distance workers is clear as we were expecting.

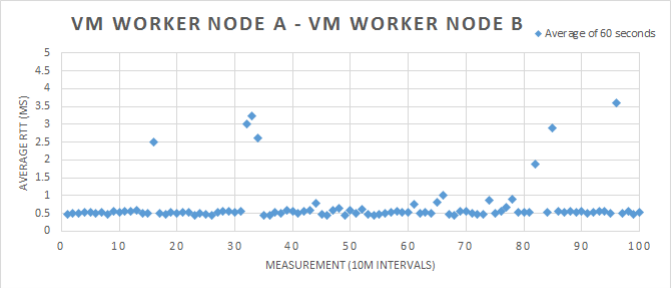


Figure 17: Experiment 3 - RTT VMA-VMB

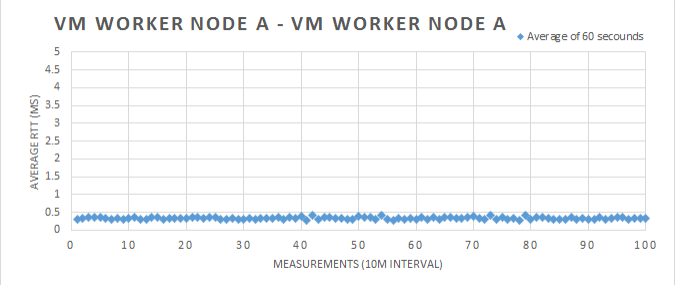


Figure 18: Experiment 3 - RTT VMA-VMA

4.3.2 TCP Throughput

Table 15 second column shows the TCP throughput of a connection where the VMs are on the same worker node inside the Rack (VMA-VMA) and the third column shows the TCP throughput for a connection that the VMs are on a different worker node (VMA – VMB). In both cases the average Throughput results are displayed.

The average TCP throughput for the VMA – VMA connection is 100 Mbps for both ExoGENI Racks where the average TCP throughput for the VMA – VMB connection on UFL and UH are approximately 99.7Mbps. The difference is minimal, thus we can conclude that the worker node on which the VMs are lying does not impact the TCP throughput and that the virtualization inside these two ExoGENI Racks does not affect this metric.

ExoGENI Rack	VM A – VM A TCP Throughput (Mbps)	VM A – VM B TCP Throughput (Mbps)
UFL	100	99,734
UH	100	99,732

Table 15: Experiment 3 - TCP Throughput Results

4.3.3 UDP Throughput – Packet Loss

In Table 16 we present the average UDP throughput and packet loss for both UFL and UH. For UFL the UDP throughput and packet loss for both connections VM A – VM A and VM A – VM B is approximately the same. The same result can be observed for UH Rack. From these results it is clear that the worker node where each VM lies does not affect these metrics. Moreover the results for both Racks are approximately the same so we can believe that all the ExoGENI Racks have the same behavior since they all have the same architecture. In order to prove this assertion we have to repeat this experiment in all ExoGENI Racks. Due to the limited time available for this research project and the unavailability of resources for most of ExoGENI Racks during this time period we didn't manage to repeat this experiment in all the Racks.

ExoGENI Rack	VM A – VM A UDP Throughput (Mbps)	VM A – VM B UDP Throughput (Mbps)	VMA – VMA Packet Loss (%)	VM A – VM B Packet Loss (%)
UFL	95,937	95,971	4	4
UH	95,911	95,929	4	4

Table 16: Experiment 3 - UDP Throughput/Packet Loss Results

4.4 Experiment 4

4.4.1 RTT

In this experiment the head node of our star topology sends packets to the destination nodes at the same time and the purpose of this experiment was to investigate the existence or not of competition. The average RTT between the head Node0 and the Node1, Node2 and Node3 that was measured, was equal to 0.51338, 0.57622, 0.57411 respectively. As we can, the three graphs have a lot similarities as a result of the virtual machines belonging to the same server.

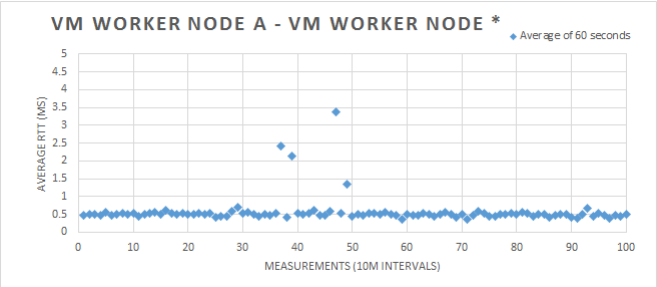


Figure 19: Experiment 4 - Average RTT

4.4.2 TCP Throughput

Experiment 4 was also implemented on UFL and UH ExoGENI Racks and the results are shown on Table 17. The average TCP throughput for all the connections for both UFL and UH is approximately 99.9Mbps. This result implies that there is no competition of the virtual links upon the physical Ethernet infrastructure that interconnects the worker nodes inside the Rack.

4.4.3 UDP Throughput – Packet Loss

On the other hand UDP throughput has a much different behavior. In particular, the 3rd link that connects the head node with the VM on worker node D has an average UDP throughput of 72Mbps and 77Mbps for UFL and UH Racks respectively. On Figure 20 we show the average UDP throughput of each measurement along with its' corresponding packet loss. As shown on Table 14 the UDP connection has some time intervals where the packet loss is increased to 60% on average. Moreover, during these time intervals the RTT was increased up to 5 times on average. In order to investigate this unexpected behavior we search on our iostat statistics and tcpdum network traffic file. We find out that on the time intervals that this unexpected behavior was observed the cpu load had an average value of 70% where in other time intervals the cpu overload had an average value of 50%. This insight looked suspicious so we repeated the same experiment but this time we use XO Large type for our VMs. XO large has 2 cores, and 6G of Ram thus we could prove if this unexpected behavior of UDP was related to the compute power of the VMs we use. The results from this experiment were the same as the initial one so we couldn't conclude on a logical explanation for this unexpected behavior. In order to further research this behavior we contact the RENCI engineers which asked us to repeat our experiment and let them know in order to check if these changes on the UDP throughput are caused from other topologies that are constructed or deleted during this time intervals. We are still waiting for their findings.

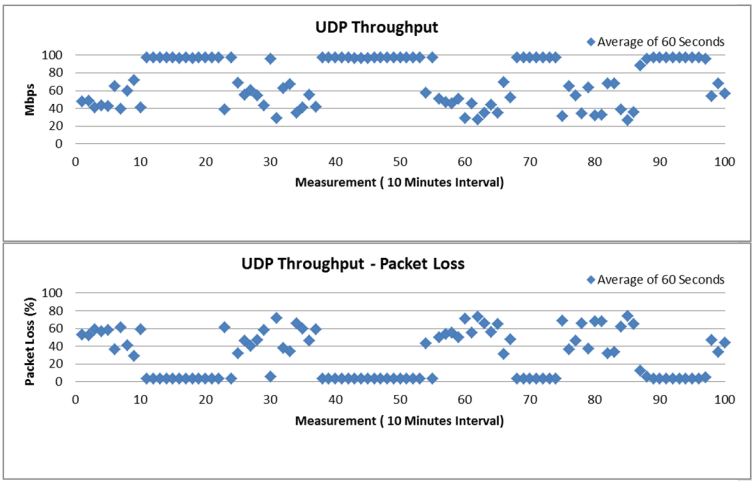


Figure 20: Experiment 4 - UDP Throughput

ExoGENI Rack	TCP Throughput			UDP Throughput		
	VM B	VM C	VM D	VM B	VM C	VM D
UFL	99,926	99,949	99,908	96,656	95,379	72,186
UH	99,904	99,942	99,893	94,925	94,338	77,596

Table 17: Experiment 4 - UDP Results

4.5 Packet Loss - Discussion

For all our experiments we observed a high packet loss rate of approximately 4%. After analyzing the cpu statistics, we observed that when iperf sends/receives UDP traffic it consumes much more cpu power than when it sends/receives TCP traffic (table 18 and 19).

Network Traffic	Sender CPU Load (%)	Receiver CPU Load (%)
UDP	4	1.5
TCP	0.1	0

Table 18: Iperf TCP/UDP CPU Load (10Mbps Network Traffic)

Network Traffic	Sender CPU Load (%)	Receiver CPU Load (%)
UDP	17	6
TCP	0.3	0.1

Table 19: Iperf TCP/UDP CPU Load (100Mbps Network Traffic)

These results gave us some insights on what could be the reason of the high packet loss rate on all UDP connections. In order to find out if the high packet loss rate is related to this observation we need to research how the scheduler of cpu operates for high cpu power consuming processes and in what degree it can affect the performance of these processes. Due to the limited available time for our research project we didn't manage to research this hypothesis.

4.6 Experiment 5

Table 20 shows the results of the reproducibility of our experiments that we explained in the previous chapter.

Experiment	Reproducibility	Results
1	Not Available Resources	-
2	Not Available Resources	-
3	Possible	Same as Initial Experiment
4	Possible	Same as Initial Experiment

Table 20: Reproducibility Results

In Experiments 1 and 2 where we used Inter-Racks domains to build the point-to-point and star topologies the recreation of our topologies was not possible due to unaivalability of resources. Thus, the repeatability of the network performance for the Inter-Racks domains could not be evaluated. For Intra-Rack domains the recreation of of our topologies was achieved for all the 100 times as a result of the availabity of resources and the reproducibility of the network performance was evaluated succesfully. Our results showed that the reproducibility of the network performance for the intra-rack domains was the same as the initial experiments, thus we can conclude that the network performance reproducibility is possible for intra-rack domains.

5 Conclusion

In this research project we evaluate the network performance of ExoGENI. For the Inter-Rack domain we evaluate the network performance for point to point connections when the ExoGENI Racks are in a short and long geographical distance (4.1). For the intra-rack domains we evaluate the network performance when the VMs are on the same and separate worker node (4.3) inside an ExoGENI Rack. Furthermore we investigate if multiple virtual connections compete upon the underlying network for both Inter (4.2) and Intra-Rack (4.4) connections. Finally, we attempt to evaluate the repeatability the network performance for all the aforementioned cases.

After our experiments for the Inter-Rack domain we find out that the short distance virtual connections have an efficiently and stable network performance for both TCP and UDP. Moreover, we observed that the long distance virtual connections also have an efficiently and stable network performance for UDP. These connections can be considered suitable for data-intensive applications since they achieved both efficient and stable network performance. For the case of long distance TCP connections we observed 11% lower throughput than the short distance TCP connections. Furthermore the long distance TCP connections had a throughput standard deviation of approximately 5 times higher than the short distance TCP connections. Bearing this in mind we conclude that long distance TCP connections are not suitable for data-intensive applications. The second experiment was only implemented among the pair BBN and UH. The results from this experiment showed that point to multipoint UDP connections are efficient and stable while TCP connections are not. However, due to the fact that results for this experiment were only extracted only for one pair of ExoGENI Racks we cannot have a general idea for this scenario so we cannot conclude. The reproducibility experiment was not able to be implemented due to unavailability of resources, so no conclusion can be made upon the repeatability of the network performance.

Our experiments for the Intra-Rack domains showed us that the worker node upon on which the VMs are lying does not affect the network performance for both TCP and UDP connections. For both cases the TCP and UDP performance were efficiently and stable thus we conclude that data-intensive applications that use point to point connections can be efficiently implemented in Intra-Rack domains. On the other hand, point to multipoint connections inside an ExoGENI Rack gave us different results for TCP and UDP connections. Specifically, for TCP connections we observe an efficient and stable network performance while for UDP connections the network performance was insufficient and unstable. This observation makes the point to multipoint connections suitable for data-intensive applications that uses TCP connections and inappropriate for those who use UDP connections. The reproducibility experiment (4.6) was successfully implemented and gave us the same network performance results as the initial experiments. From this, we can conclude that the network performance in the Intra-Rack domains is reproducable when the virtual network topology is reconstructed from scratch.

Finilazing, besides our results during our research project we faced a lot of difficulties in reserving resources and constructing Inter-Rack topologies on ExoGENI. In general, the system could not provide us what it was designed for (eg. high bandwidth virtual links, multi-domain topologies) the most of the days and during the days that we were able to construct Inter-Rack topologies we observed some cases of unexpected behaviors that neither we or ExoGENI IT enginners could explain. From our experience during our research project we can say that the Inter-Rack domain needs some improvements on resource provisioning and managing in order to offer a suitable experimental environment for the experimenters.

6 Future Work

On our research project we evaluate the network performance for four scenarios that emerged from ExoGENI architecture. For each of our scenarios we only implemented experiments for a small number of ExoGENI Racks. In order to have the full picture of ExoGENI network performance each of our scenarios experiment has to be implemented for all the possible combinations of ExoGENI Racks. Especially the network performance of the Inter-Rack domain where each interconnection between the Intra-Rack domains is provided by a different circuit provider needs to be further researched. Moreover, the experiment for the repeatability of the network performance for the Inter-Rack domain wasn't able to be implemented due to unavailability of resources thus it is proposed for future work. Finally, the high packet loss rate observed during our experiments needs to be further researched.

References

[1] Ilia Baldine, Yufeng Xin, Anirban Mandal, Paul Ruth, Chris Heerman and Jeff Chase, *"ExoGENI: A Multi-Domain Infrastructure-as-a-Service Testbed"*, RENCI, Department of Computer Science Duke University

[2] *"GENI : Exploring the Networks of the Future"*, http://www.geni.net/

[3] *"Global Enviroment for Networking Innovations (GENI)"*, http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=501055

[4] *"ExoGENI Rack Details"*, https://wiki.exogeni.net/doku.php?id=public:hardware:start

[5] *"ExoGENI Introduction"*, https://wiki.exogeni.net/doku.php

[6] *"NICL Lab Duke"*, http://nicl.cod.cs.duke.edu/

[7] *"BEN Renci"*, https://ben.renci.org/

[8] *"Renci"*, http://www.renci.org/

[9] *"ORCA"*, https://geni-orca.renci.org/trac/

[10] *"Flukes"*, https://geni-orca.renci.org/trac/wiki/flukes

[11] *"Available inter-racks"*, https://wiki.exogeni.net/doku.php?id=public:experimenters:rspecs

[12] *"ExoGENI Rack Campuses"* http://groups.geni.net/geni/attachment/wiki/GEC12GENIDeploymentUpdates/GEC12-ExoGENI-Racks-campuses.pdf

[13] *"Rack Operators"* https://wiki.exogeni.net/doku.php?id=public:operators:start

[14] *"ExoGENI Hardware"* https://wiki.exogeni.net/doku.php?id=public:hardware:start

[15] *"Hardware Requirements"* https://help.ubuntu.com/lts/installation-guide/i386/minimum-hardware-reqts.html

[16] *"Rack Location"* http://www.exogeni.net/locations/

[17] *"ExoGENI Interconnections"* https://wiki.exogeni.net/doku.php?id=public:experimenters:topology

[18] *"ExoGENI Racks Architecture"* http://groups.geni.net/geni/attachment/wiki/GEC12GENIDeploymentUpdates/GEC12-ExoGENI-Racks-campuses.pdf?format=raw

[19] *"ORCA Operation"* https://geni-orca.renci.org/trac/attachment/wiki/orca-introduction/orca-arch.png

[20] *"Virtualization Overhead 1/2"* http://sdqweb.ipd.kit.edu/publications/descartes-pdfs/HuQuHaKo2011-CLOSER-ModelVirtOverhead.pdf

[21] *"Virtualization Overhead 2/2"* http://www.cc.iitd.ernet.in/misc/cloud/hypervisor_performance.pdf

[22] *"ICMP Packet size"* http://searchnetworking.techtarget.com/answer/What-are-the-minimum-and-maximum-sizes-of-an-ICMP-packet

[23] Yong Wang *"Networking Virtualization"*, VMWARE

[24] G. Wang, T. S. Eugene Ng, *"The Impact of Virtualization on Network Performance of Amazon EC2 Data Center"*, Dept. of Computer Science, Rice University

[25] D. Battre, N. Frejnik, S. Goel, O. Kao and D. Warneke, *"Evaluation of Network Topology Inference in Opaque Compute Clouds Through End-to-End Measurements"*, T.U. Berlin, T.U. Munich

[26] D. Battre, N. Frejnik, S. Goel, O. Kao and D. Warneke, *"Inferring NetworkTopologiesin Infrastructure as a Service Cloud"*, T.U. Berlin, T.U. Munich

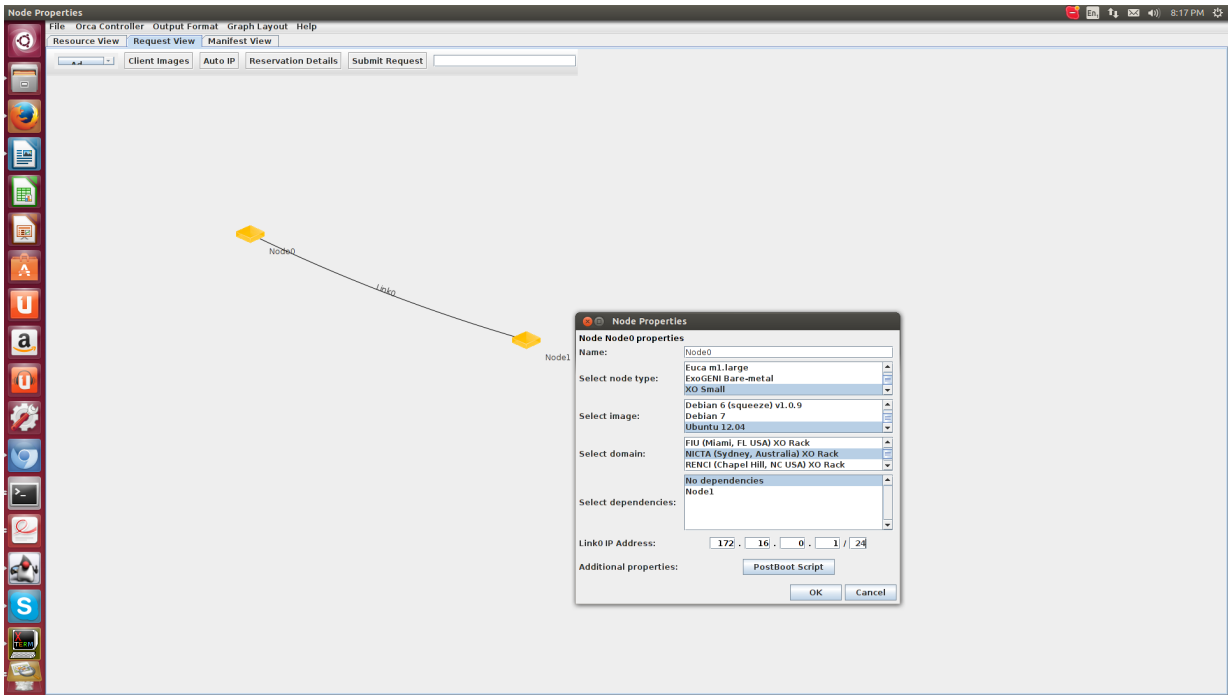
[27] Bill Howe, *"Virtual Appliances, Cloud Computing, and Reproducible Research"*, University of Washington, Seattle, WA

[28] *Layers Overhead* `http://sd.wareonearth.com/~phil/net/overhead/`

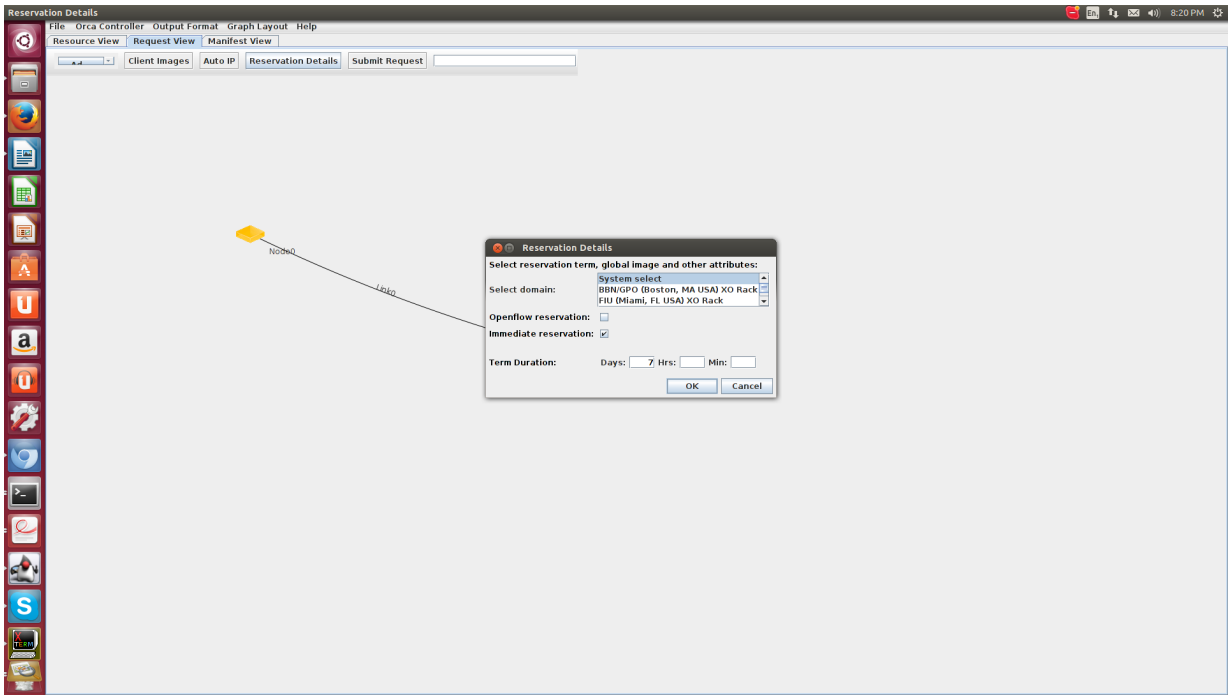
A WorkFlow

A.1 Add Topology

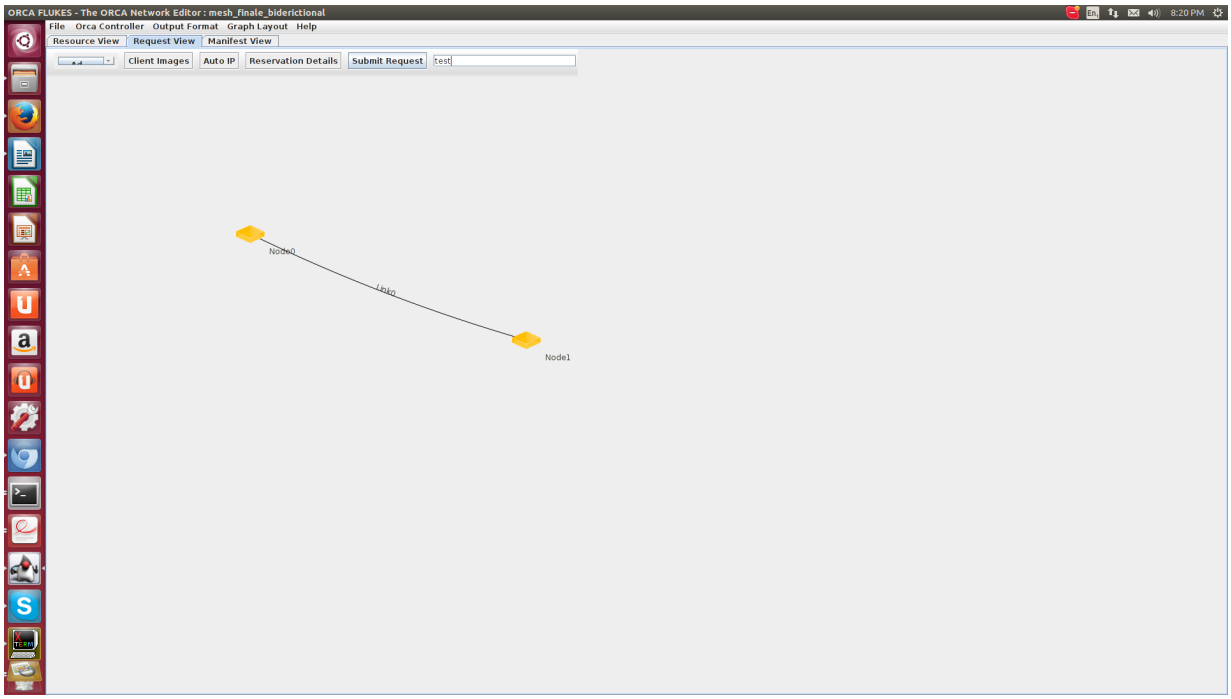
A.2 Node properties



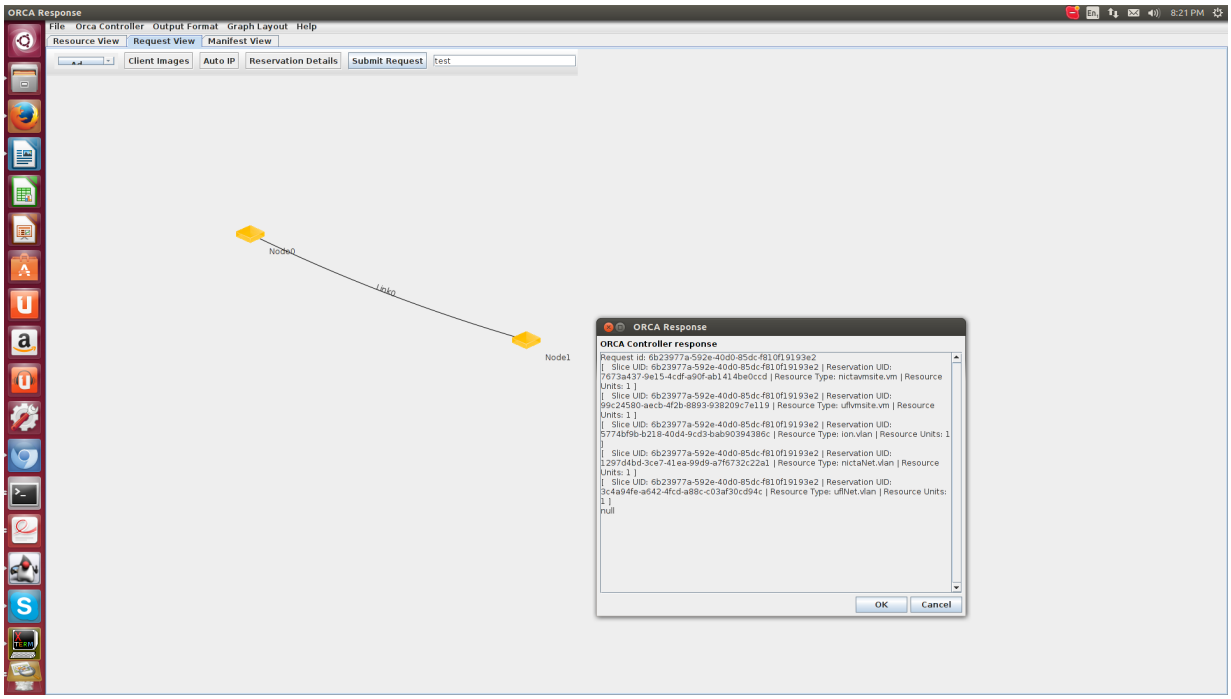
A.3 Slice reservation



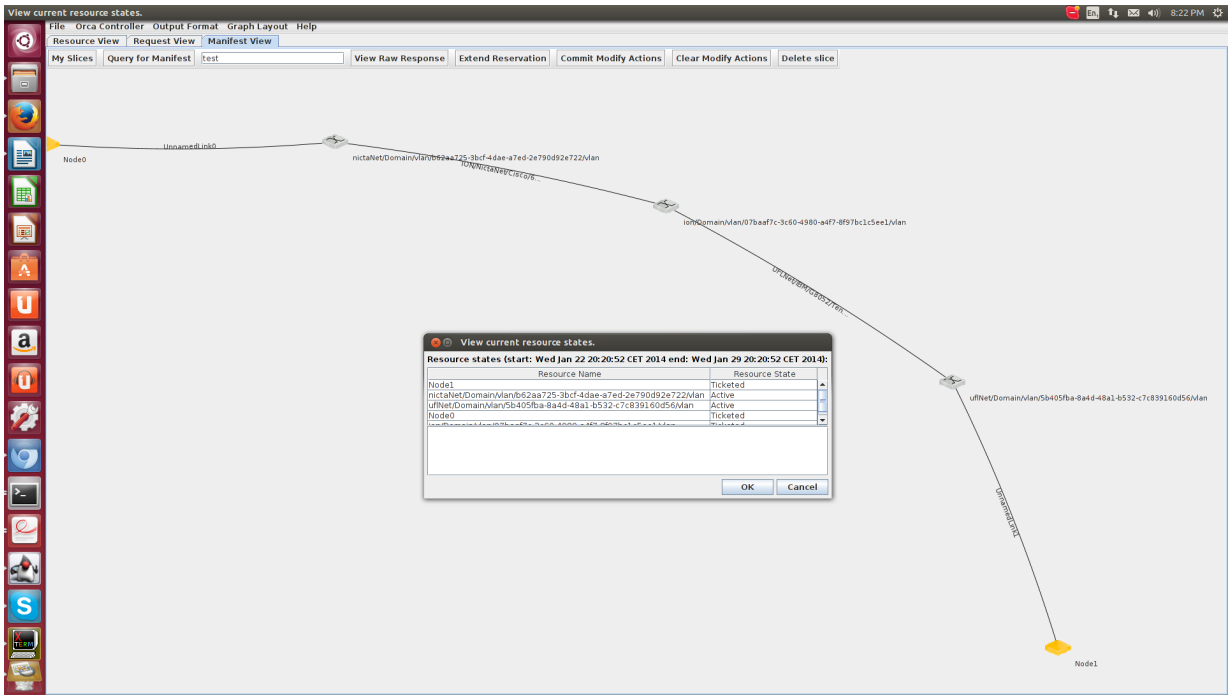
A.4 Choose slice name



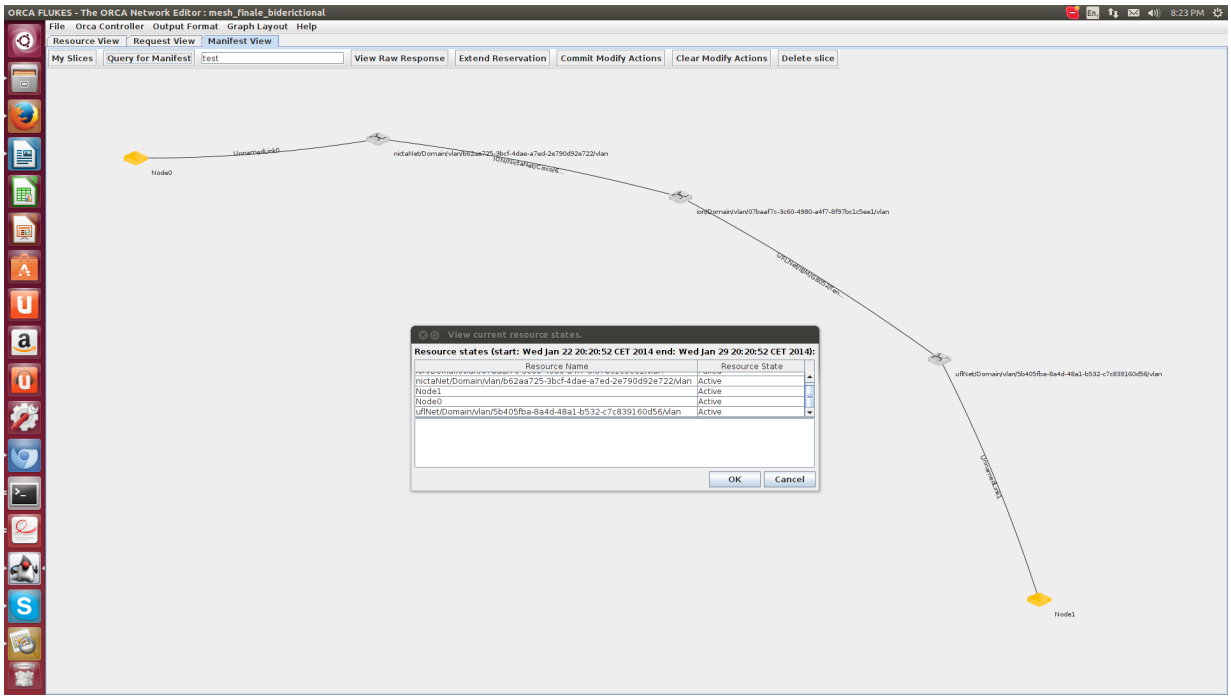
A.5 Submit slice



A.6 Resource state



A.7 Resource active



A.8 Node login