A COMPARATIVE PERFORMANCE ANALYSIS OF GENI CONTROL FRAMEWORK AGGREGATES

By

NIDHI TARE

B.E., Shri. G. S Institute of Technology and Science, India, 2007

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:

Major Professor
Dr. Caterina Scoglio

Abstract

Network researchers for a long time have been investigating ways to improve network performance and reliability by devising new protocols, services, and network architectures. For the most part, these innovative ideas are tested through simulations and emulation techniques that though yield credible results; fail to account for realistic Internet measurements values like traffic, capacity, noise, and variable workload, and network failures. Overlay networks, on the other hand have existed for a decade, but they assume the current internet architecture is not suitable for clean-slate network architecture research. Recently, the Global Environment for Network Innovations (GENI) project aims to address this issue by providing an open platform comprising of a suite of highly programmable and shareable network facilities along with its control software. The aim of this report is to introduce GENI's key architectural concepts, its control frameworks, and how they are used for dynamic resource allocation of computing and networking resources. We mainly discuss about the architectural concepts and design goals of two aggregates, namely the BBN Open Resource Control Architecture of the (BBNORCA) of the ORCA control framework and Great Plains Environment for Network Innovations (GpENI) belonging to the PlanetLab control framework. We then describe the procedure adopted for hardware and software setup of individual aggregates. After giving an overview of the two prototypes, an analysis of the simple experiments that were conducted on each of the aggregates is presented. Based on the study and experimental results, we present a comparative analysis of control framework architectures, their relative merits and demerits, experimentation ease, virtualization technology, and its suitability for a future GENI prototype. We use metrics such as scalability, leasing overhead, oversubscription of resources, and experiment isolation for comparison.

Table of Contents

List of Figures v
List of Tables vi
Acknowledgements vii
CHAPTER 1 - Global Envrionement for Networking Innovations 1
1.1 Introduction 1
1.2 Existing Facilities for Network Research 2
1.2.1 Network Simulation 3
1.2.2 Network Emulation 4
1.2.3 Overlay Networks 5
1.3 Design Goals of GENI 5
1.4 GENI Control Frameworks 6
1.5 Report Outline 8
CHAPTER 2 - The PlanetLab Control Framework 9
2.1 Introduction 9
2.2 Role of PlanetLab in GENI 9
2.3 The Great Plains for Networking Innovations (GpENI) Project 11
2.3.1 The GpENI Node Cluster 12
2.3.2 Setup and maintain GpENI Aggregate 14
2.4 Slice-based Facility Architecture 15
2.5 Geniwrapper - Exposing PlanetLab GENI interfaces 16
2.6 GpENI Integration with other Projects in GENI 17
2.6.1 Vertical Integration with GUSH 18
2.6.2 Horizontal Integration with Seattle 18
CHAPTER 3 - The Open Resouce Control Architecture (ORCA) Control Framework 20
3.1 Introduction 20
3.2 Role of ORCA in GENI 20
3.3 The BBNORCA Project 20

3.4 Set up and maintain BBNORCA Aggregate 22
3.5 Integrationof BBNORCA Aggregate with Cluster D Clearinghouse 29
CHAPTER 4 - GENI Experiment Design 30
4.1 Introduction 30
4.2 GpENI Experiment Design and Performance Analysis 30
4.2.1 Experiment Procedure 31
4.2.2 Results and Conclusions 34
4.3 ORCA Experiment Design and Performance Analysis 38
4.3.1 Results and Conclusions 39
CHAPTER 5 - Comparative Performance Analysis of CFs 40
5.1 Introduction 40
5.2 Metrics 40
CHAPTER 6 - Conclusion and Future Work 43
Bibliography 45

List of Figures

Figure 1.1 Internet and Web Hourglass 2
Figure 1.2 Design goals of GENI 3
Figure 2.1 The PlanetLab GENI Control Framework Projects with GpENI components 10
Figure 2.2 GpENI optical backbone over the Great Plains Network (GPN) 11
Figure 2.3 GpENI European underlay Topology 12
Figure 2.4 GpENI Node Cluster 13
Figure 2.5 MyPLC Web Interface 14
Figure 2.6 Add a PlanetLab node to a site 15
Figure 2.7 The Geniwrapper conceptual diagram 17
Figure 2.8 GUSH running clients on GpENI and PlanetLab nodes 18
Figure 3.1 ORCA actors and messages exchanged 21
Figure 3.2 ORCA software pieces 23
Figure 3.3 ORCA web portal showing Online Actors 25
Figure 3.4 ORCA web portal showing Active machine geni-ORCA assigned to site actor 26
Figure 3.5 ORCA web portal showing Xen VM Broker as Ticketed 26
Figure 3.6 ORCA web portal showing ticket for creating a reservation 27
Figure 3.7 ORCA web portal showing Reservation successful and lease Active 28
Figure 3.8 ORCA web portal shows brokers of Cluster D as Online 29
Figure 4.1 GpENI experiment topology 31
Figure 4.2 MyPLC web portal to reserve a slice 32
Figure 4.3 Login into GpENI nodes 32
Figure 4.4 Iperf TCP test between KSU and KU 35
Figure 4.5 Iperf TCP test between KSU and Cambridge 36
Figure 4.6 Iperf TCP test between KSU and BERN 36
Figure 4.7 Iperf TCP test between KSU and Lancaster 37
Figure 4.8 Interaction between ORCA actors and their leasing state 38
Figure 4.9 Leasing overhead in an ORCA reservation 39

List of Tables

Table 4.1 Average ping times in msecs between GpENI sites..... 34
Table 4.2 Average UDP delays in msecs between GpENI sites 34
Table 4.3 Ratio of UDP and ping delays between GpENI sites..... 35

Acknowledgements

I wish to extend my heartfelt thanks to my advisor Dr. Caterina Scoglio who has guided me throughout my research work both as a teacher and as a mentor. She has motivated me to make persistent efforts towards my goals, with hard work and dedication, however challenging the task may be. Like many other graduated students, she has influenced my life by her encouraging words.

I would also like to thank members at the GENI Project Office (GPO), especially Heidi Dempsey for her guidance on the Intern project at BBN Technologies. Special thanks to Head of the Department and my co-advisor, Dr. Don Gruenbacher in providing us with necessary equipments in setting up the testbed.

Finally, I would like to thank my fellow graduates John Sherrell, Ali Sydney, Yunzhao Li, Chunhui (Evan) Zhang, and Ihsan Qazi for the regular discussions on the GENI project and their valuable suggestions during my work.

.

CHAPTER 1 - Global Environment for Networking Innovations

1.1 Introduction

Academic and industrial research communities have been addressing a number of problems associated with the current Internet [1] and the ossification of the network protocol stack. The proposed solutions to these research problems are narrowly focused and to implement these ideas on the operation Internet is a very difficult task.

For more than 25 years, the networking community has been designing a range of systems to enable distributed systems. Most of the projects involved are about redesigning Internet Systems like DCE [2], DCOM [3], and CORBA [4] that have been complete, elegant, and correct. However, none of them have been widely successful. In the case of TCP/IP protocol, which started out as a simple protocol implemented by DARPA and NSF research projects suffer from limitations and rigidity in adding new capabilities and supporting billions of end hosts.

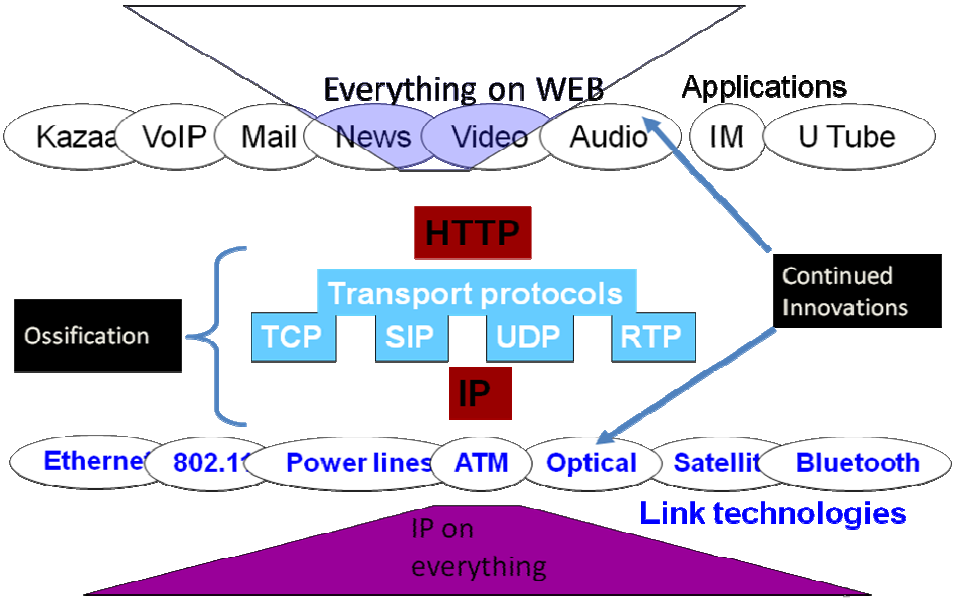

Figure 1.1 The Internet and Web Hourglass [11]

In short, the Internet has become very complex and requires significant investment (time, money, and perseverance) to build, modify, and use. The simplicity and ability to use it with minimal new investments made it easy to get going, but difficult to stop. And many limitations including lack of security and programming capabilities did not matter. Even today, the narrow waist of IP continues to support the evolving protocols at the top and bottom layers as shown in the Figure 1.1.

All this thinking paved the way for a clean slate of next generation network research to redo the national networking infrastructure, also called the Global Environment for Network Innovations (GENI) [5]. GENI is a National Science Foundation funded research project. GENI is a suite of network research infrastructure to support experimental research in network science and engineering. This novel research initiative involves understanding networks broadly and at multiple layers of abstraction from the physical substrates through the architecture and protocols that define networking principles.

GENI is currently in the stage of prototyping and early experimentation and its goal is to provide a networked testbed for researchers to conduct networking and systems experiments at scale using the backbone networks (Internet 2 and National Lambda Rail) coupled with regional networks. In this chapter, we introduce the existing networking testbeds and give a detailed overview of the GENI research initiative.

1.2 Existing Facilities for Network Research

To understand the significance of GENI we will first discuss the existing facilities for networking research and their pros and cons, which led to the idea of building an at-scale testbed such as GENI. A large fraction of the ongoing research in the field of computer science is to design new protocols and applications for future Internet which improves the quality of service and efficiency catering to the ever increasing demands. These research experiments involve different network layers (of ISO-OSI network stack), edge resources (servers, sensor nodes, routers), and network media (wireless, optical fiber, Ethernet, Internet backbone). To

support the experiments at-scale, with the introduction of realistic network conditions, there is need of an appropriate testbed supporting the right environment. In general, the testbed choice decision is largely based on:

1. The types of resources that form a testbed
2. How realistic the experiment measurement data is
3. What scalability level the experiments want to achieve
4. How easy it is to use the testbed for experimentation
5. Availability of resources comprising the testbed
6. How many users a testbed is capable of supporting at a time without causing interference

The testbed facilities currently available are: network simulators, network emulators, and overlay network testbeds. NS-2/NS-3 [6], called the network simulator, is the most popular open-source network simulation software. Emulab [7] is a local network emulator that supports arbitrary user-defined network topologies consisting of PCs, switches, network processor cards, and wireless nodes. Two most common examples of overlay networks are Resilient Overlay Networks [8], developed by Massachusetts Institute of Technology and PlanetLab [9], and developed by Princeton University. In addition to the above facilities there are various resource specific testbeds like ORBIT (mobile wireless testbed) [10], DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks) [12], DETER (Defense Technology Experimental Research Laboratory Testbed) [13], and Kansei (wireless sensor networks) [14]. We now discuss the existing facilities in detail and point out the limitations that GENI is addressing.

1.2.1 Network Simulation

A network simulator provides users with an environment that can be used to simulate a real network with nodes, topology, traffic, and other realistic circumstances using the program. The program is generally characterized by a mathematical or statistical model. Various attributes of the simulation environment can also be modified in a controlled manner to assess how the network would behave under different situations. As we mentioned earlier, NS-2 is one of the most popular network simulators. With it, we can implement nearly anything regarding various

network aspects (e.g. brand new network stack, novel network applications, etc.) as modules and run them with a large number of simulated nodes. However, it obviously has its own shortcomings.

1. As a generic downside of simulation, it fails to obtain the realistic measurement data that could be generated with real computing nodes and links.
2. Potential innovations run in a closed experimental circle and there is no way to introduce real user traffic.
3. The software is slower than hardware and it may not be possible to get realistic performance values with simulations.

1.2.2 Network Emulation

In terms of realistic measurements a network emulator is an improvement over a network simulator by allowing computer programs to run on the platform other than the one they were originally written for. Unlike network simulation, a network emulator is usually a LAN-based PC cluster with special nodes responsible for imitating the realistic link behavior. Emulab is a widely used network emulator supporting network experiments by managing a suite of locally wired switches, PCs, network processor cards, and wireless nodes. By supporting arbitrary, user-defined topology, Emulab enables experiments in a wide range of networked and distributed systems. Comparing to network simulation, by conducting emulated experiments, researchers get a direct feeling of their running application and more realistic measurement results since their code runs on real PCs. However, the network links are still emulated in a manner of introducing the special delay node between end-PCs. We conclude the shortcomings as follows:

1. Emulated links fail to provide native characteristics of real links.
2. Emulators are usually small scale.
3. Similar to network simulation, the experiments are still ‘research-only’ and there is no easy way to deploy them in the real world or in Wide Area Networks (WANs).

1.2.3 Overlay Networks

An overlay network is a computer network built on top of another network. For example, the Internet is an overlay network built over telephone networks. Nodes in the overlay can be

thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. For example, many peer-to-peer networks are overlay networks because they run on top of the Internet at the application layer. As a popular overlay network testbed, PlanetLab and Resilient Overlay Networks (RONs) enable multiple application layer research including content distribution networks, peer-to-peer video steaming, and overlay routing. Experimenters are able to deploy their new applications at Internet-scale with it and possibly introduce real user traffic from other participants of PlanetLab sites. One unique feature of PlanetLab is that its nodes are virtualized. This feature enables the simultaneous experiments on the same node and thus enhances the resource utilization and supports green computing. However, due to poor resource isolation at the network layer, this feature does result in one of the shortcomings of PlanetLab.

1. It assumes the current internet architecture and is not suitable for clean-slate network architectural research.
2. The system is best effort by oversubscribing its nodes so the performance cannot be guaranteed.

1.2 Design Goals of GENI

GENI addresses these shortcomings of the testbeds as described above, as well as proposes a unified testbed solution for at-scale experiments, supporting a spiral development approach to conceptualize, build, deploy, and test for the clean slate Internet architecture design. The goal is to build a research testbed that leverages Network Science and Engineering (NetSE) experiments, which have been underway since 2005. Figure 1.2 explains the GENI vision.

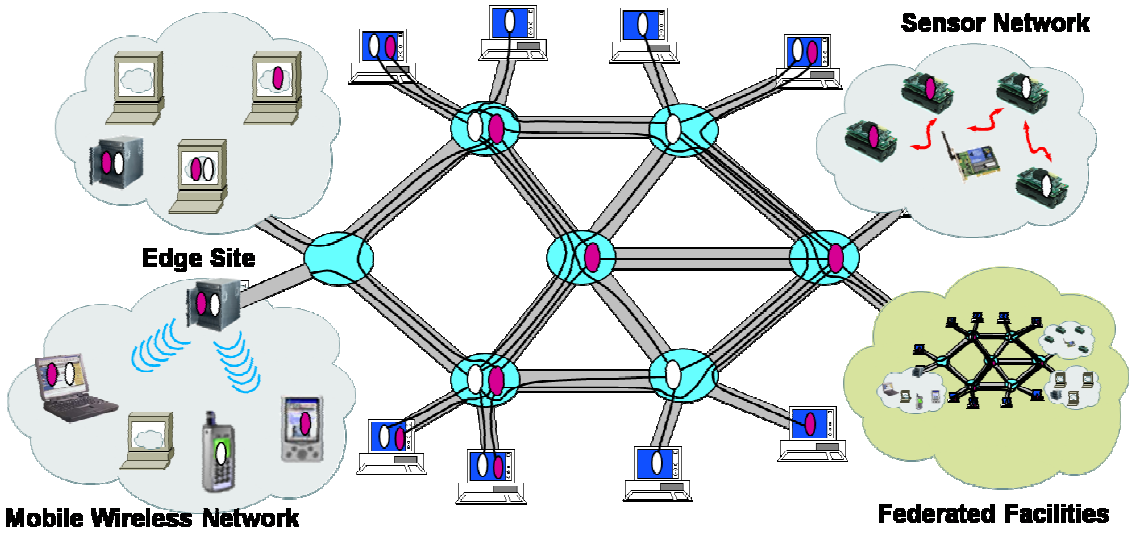

Figure 1.2 Design goals of GENI [11]

As the conceptual design of GENI evolved over the Spiral I of planning and prototyping phase, GENI has put forth certain design goals or core concepts as follows:

1. **Programmability:** To be able to download node compatible software and execute them remotely. A programmable infrastructure allows setting up and tearing down of experiments.
2. **Virtualization or resource sharing**: To be able to reserve a portion or sliver of the computing and network resources. This idea saves infrastructure cost and reduces NxN mapping of resources and experimenters. While the resources are shared between experiments, isolation is provided such that the experiments can run without interference from other experiments.
3. **Federation**: To be able to join GENI "islands", owned by different organizations to form a common suite of NSF owned GENI infrastructure. These GENI islands are independent testbeds in themselves that support a variety of edge and backbone resources.
4. **Slice based Experimentation:** To be able to remotely discover, reserve, configure, program, debug, operate, manage, and teardown distributed systems established across parts of the GENI suite.

1.3 GENI Control Frameworks

Researchers use GENI by acquiring resources from components through GENI control frameworks. To support GENI design goals as stated above, there are currently four software suites that implement a subset of GENI features. Conceptually, the control framework can be divided into four different planes – control plane, data plane, management and operations plane, and measurement plane.

1. **Control plane**: It is defined as infrastructure and distributed intelligence that controls the establishment and maintenance of connections in the network, including protocols and support mechanisms to relay this information. Some of the functionalities include creating VMs, destroying VMs, and populating public keys in the slice. From an experimenter's point of view, the setup and tear down of experiments fall under the control plane.
2. **Data Plane**: Once the experiment is setup, the researcher deploys the experiment on the slivers. The data plane carries the actual experimental data. A true measure of data plane connectivity is the number of computational units consumed in the VMs and the network bandwidth consumed during the experiment.
3. **Management & Operations Plane**: To ensure the validity of experiments in terms of overuseage of resources, misbehaving slices and security threats the management plane comes into play. The GENI Management Authority (MA) is responsible for a subset of substrate components: providing operational stability for components, ensuring the components behave according to the acceptable use policies, and executing resource allocation at the wishes of the site owner. Operations plane includes functions related to interoperability of independently-owned and autonomously-administered infrastructures. This plane is also related to infrastructure monitoring using services for instrumentation and control.
4. **Measurement Plane**: One of the main goals of building GENI is to provide the ability to instrument the testbed for a real time monitoring of a diverse set of physical substrates. The substrates include the fiber optic backbone, programmable core switches, campus networks, metro area networks, wireless mobile networks and sensor

networks.. At present, there are four control frameworks in operation that provides support for the four planes as mentioned above:

1. Cluster B or PlanetLab control framework
2. Cluster C or ProtoGENI control framework
3. Cluster D or ORCA control framework

This report deals with two control frameworks – cluster B and cluster D. A technical overview of each one of them is presented in the subsequent sections along with their software setup details.

1.4 Report Outline

This report consists of six chapters. Chapter 2 talks about the PlanetLab control framework. We first outline the PlanetLab architectural features and design goals. Then we talk about setting up the GpENI aggregate. We then present a brief overview of other Cluster B projects that GpENI has integrated vertically and horizontally. Chapter 3 describes the ORCA control framework. We first outline the architectural features of ORCA and its design goals. A detailed summary of setting up a local ORCA aggregate, which was part of my internship at BBN Technologies, is presented in the subsequent sections. In Chapter 4, we describe a GENI experiment that was designed for both the aggregates in PlanetLab and ORCA control frameworks. In the last section, we compare the two control frameworks based on the architectural design and experiments conducted on the aggregates. In the end we present the limitations of the current work and steps for the future work.

CHAPTER 2 - The PlanetLab Control Framework

2.1 Introduction

The PlanetLab Consortium is a collection of academic, industrial, and government institutions cooperating to support and enhance the PlanetLab overlay network [15]. It is responsible for development and maintenance of PlanetLab's hardware infrastructure; designing and evolving its control software to make it usable in GENI by researchers and providing day-to-day operational support. PlanetLab provides a platform for distributed systems research by creating new network protocols, evaluating new and existing services, and deploying novel services that enhance capabilities of Internet.

PlanetLab continues to support at-scale experiments in the areas of network measurements, Distributed Hash Tables (DHT) and P2P, resource allocation, security, multicast, and broadcast. Some of the services that are currently running are CoMon – PlanetLab monitoring infrastructure [16], CoDeen – an academic testbed Content Distribution Network (CDN) build on the top of PlanetLab [17], and PlanetFlow – PlanetLab's traffic monitoring system [18].

2.2 Role of PlanetLab in GENI

The public PlanetLab offers a testbed for computing and networking resources with the explicit goal of permitting open access. The PlanetLab infrastructure is shared using the slice paradigm such that multiple users can share the testbed for at-scale experimentation. Every user gets a slice for his own experiment. In the GENI context, PlanetLab extends as a control framework or a software suite for on-demand resource provisioning and control.

Based on the design of the GENI control framework architecture, the PlanetLab control framework (also called GENI Cluster B) implements the following features -

1. Integrating a variety of aggregate-like edge clusters, high-performance backbone nodes, enterprise-level nodes, and edge-sitting wireless nodes
2. Federating across multiple, independently-controlled aggregates OneLab or PlanetLab Europe [19], Great Plains Environment for Network Innovations (GpENI) [20].
3. Operating the Cluster B, GENI prototype clearinghouse

4. Provisioning experiments at L2 by connecting several VINI [21] nodes. Also, providing Supercharged PlanetLab (SPP) [22] nodes in the backbone at 3 Internet2 PoPs.

Figure 2.1 below shows the PlanetLab control framework along with other projects falling under Cluster B. The PlanetLab, GpENI, SPP, VINI, Open flow [23], and Mid Atlantic Crossroads (MAX) [12] are the aggregates that implement PlanetLab control software for their specific resources. GUSH [24] and Raven [25] are the experimental control tools that run as a service on the aggregates for resource discovery and provisioning. They will be explained in the later sections.

The GpENI aggregate (shown in the bigger box) in Figure 2.1 shows the node cluster at each of the American and European GpENI sites. The diagram also shows the connectivity of the PlanetLab GENI interface with the GpENI interface.

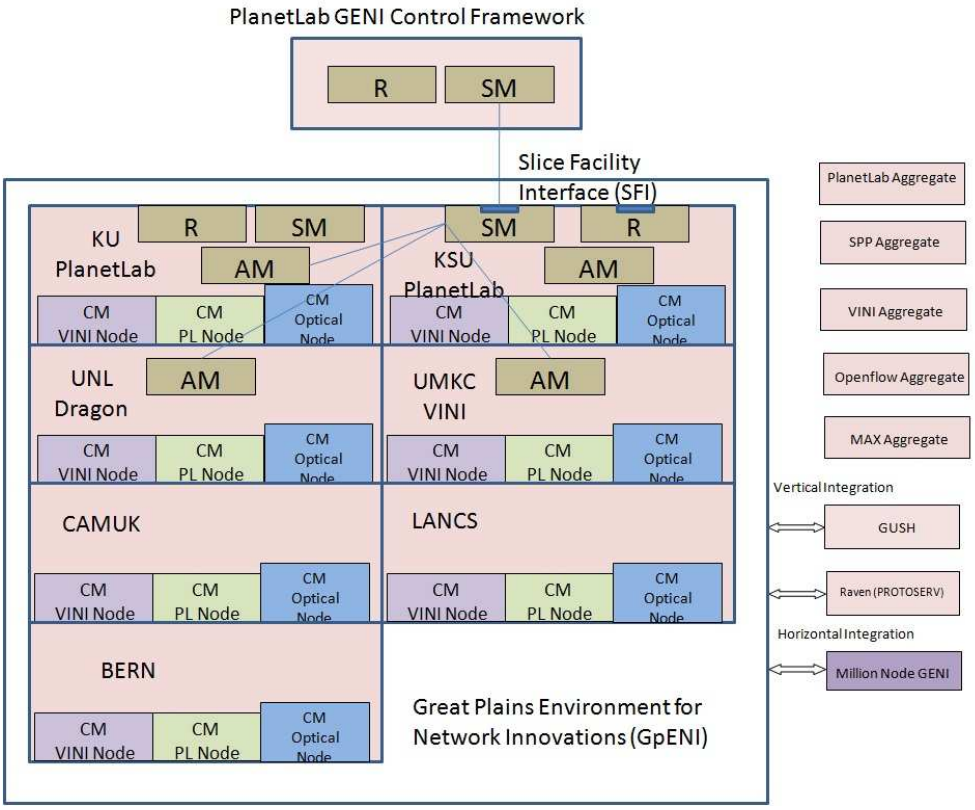

Figure 2.1 The PlanetLab GENI Control Framework Projects with GpENI components

2.3 The Great Plains Environment for Network Innovations

GpENI is an international programmable testbed centered on the regional multi-wavelength fiber interconnection between four Midwest universities – The University of Kansas (KU), Kansas State University (KSU), University of Nebraska – Lincoln (UNL), and University of Missouri – Kansas City (UMKC) within the Great Plains Network (GPN). The GpENI infrastructure provides programmability across the entire network protocol stack.

The main goals of GpENI are – 1) Build an at-scale programmable testbed suite enabling experiments in future Internet architecture, supporting projects such as PoMo: Post Modern Internetwork Architecture [26] and ResumeNet [27], 2) Deploy experimental tools and testbed monitoring infrastructure at Layer7, 3) Provide programmability at Layer 3 by using software routers like Quagga [28], XORP [29], Click [30] and deploy VINI to provide flexible network-layer topologies, 4) Leverage optical infrastructure for GpENI experiments and provide Layer 2 programmability by using DCN-enabled Gigabit-Ethernet switches at each GpENI site.

The GpENI topology consists of the Midwest optical backbone coupled with Canadian, European, and Asian GpENI infrastructure. The four GpENI sites, also the GPN institutions, at KU, KSU, UMKC, and UNL are connected over the optical backbone centered at Kansas City and extended by KanREN (Kansas Research and Educational Network). The optical backbone consists of a fiber optic running from KSU to KU to the Internet2 POP in Kansas City, interconnected with dedicated wavelengths to UMKC and UNL as shown in the Figure 2.2.

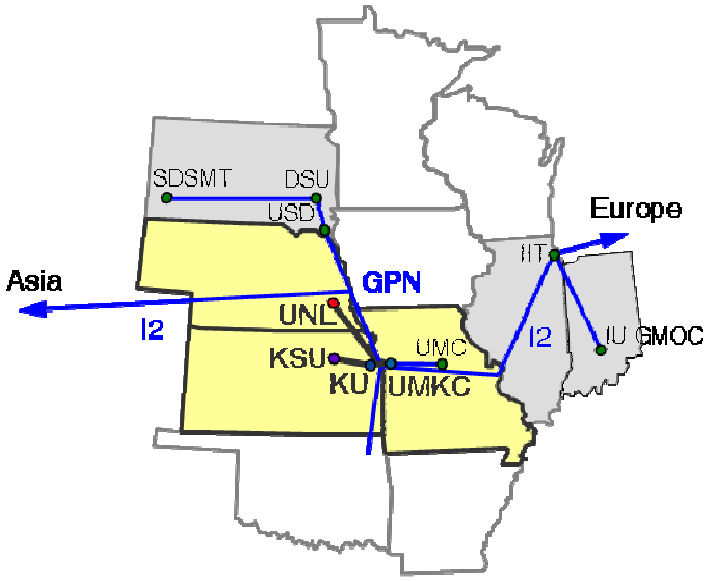

Figure 2.2 GpENI optical backbone over the Great Plains Network (GPN) [52]

The European topology is extended across Internet2 to GÉANT2 [31] and NOR-DUnet [32] and then to regional or national networks. Currently, the connectivity is achieved using L2TPv3 and IP tunnels. A direct fiber link over JANET [33] is deployed between Lancester and Cambridge Universities. The topology diagram showing the optical backbone connectivity is show in Figure 2.3.

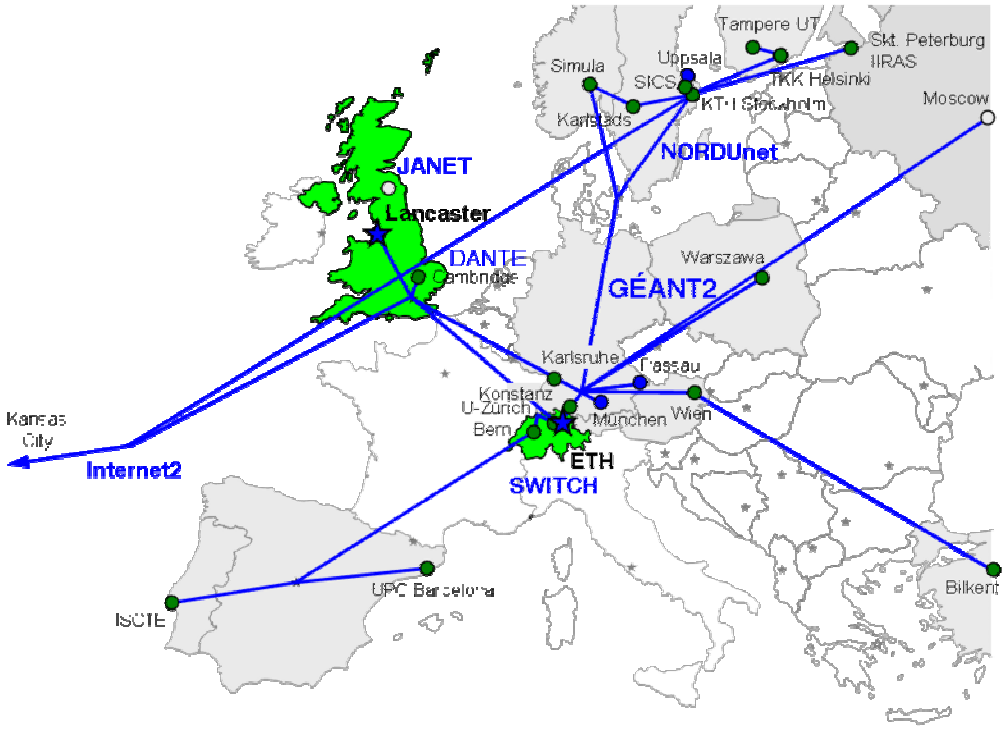

Figure 2.3 GpENI European underlay Topology [52]

2.3.1 GpENI Node Cluster

Each GpENI node cluster consists of several components, physically interconnected by a Gigabit Ethernet switch. GpENI uses a private IP address space (198.248.240.0/21) provided by KanREN. The node cluster is designed to be as flexible as possible at every layer of the protocol stack, and consists of the following components:

1. GpENI management and control processor: general-purpose Linux machine for backing up MyPLC database and hosting the GpENI overlay demo [51]. Monitoring tools like CoMon, Cacti [34], Nagios [35], Zenoss Core [36].
2. PlanetLab control framework consisting of aggregate managers: MyPLC with GENIwrapper SFA (at KSU), MYVINI (at UMKC), and DCN (at UNL)
3. PlanetLab programmable nodes (enabling layer 4 and 7 experimentation)
4. VINI-based programmable routers (providing flexible network topologies), with Quagga and other extensions such as XORP and Click (enabling layer 3 experimentation)
5. Site-specific experimental nodes and testbeds, including software defined radios (such as the KUAR), optical communication laboratories, and sensor testbeds Managed Gigabit Ethernet switch, providing L2 VLAN programmability and connectivity to the rest of GENI
6. Ciena optical switch running DCN providing L1 interconnection among GpENI node clusters on the Midwest US optical backbone

The arrow overlaid on the figure shows a conceptual flow of an experiment in which the GENI experiment controls the configuration of the PlanetLab, which in turn configures a custom routing protocol, which in turn configures the optical switch configuration.

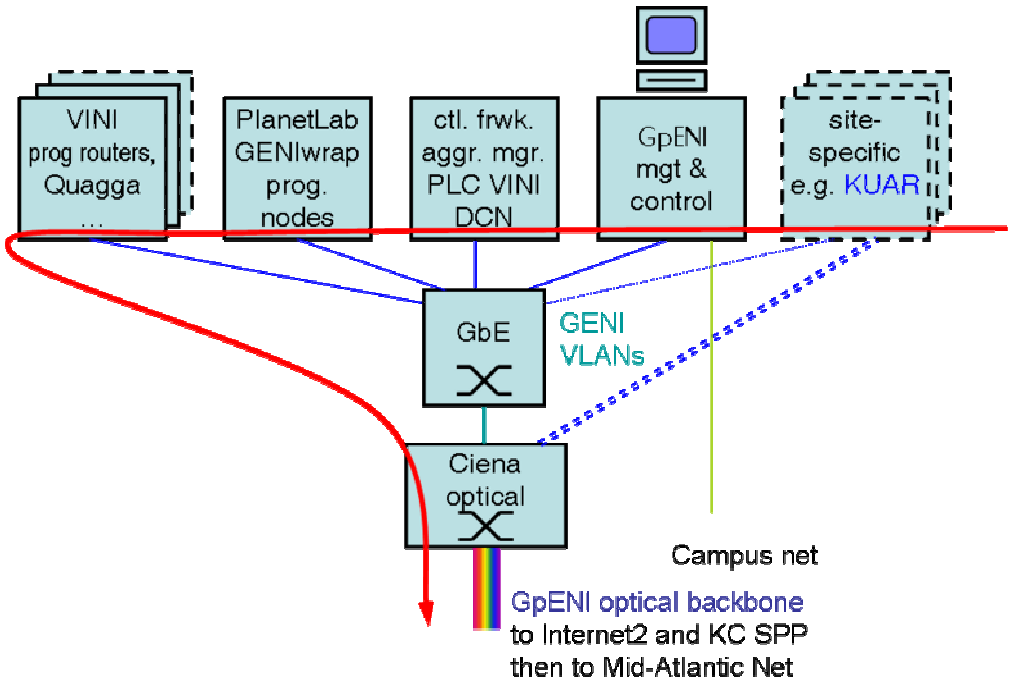

Figure 2.4: GpENI Node Cluster [52]

2.3.2 Setting up a GpENI PlanetLab Aggregate

The primary design goal of the PlanetLab testbed is centralized management and control. A PlanetLab Controller (PLC) is the central control authority that provisions resources and maintains records of users, nodes, slices, sites, and accounts. Earlier in Spiral I, the PlanetLab consortium released MyPLC, an open source complete PlanetLab Central (PLC) portable installation that users can adapt to create their own PlanetLab testbed. The default installation consists of a web server, an XML-RPC API server, a boot server, and a database server that providing the software pieces necessary to host a distributed testbed of PlanetLab nodes.

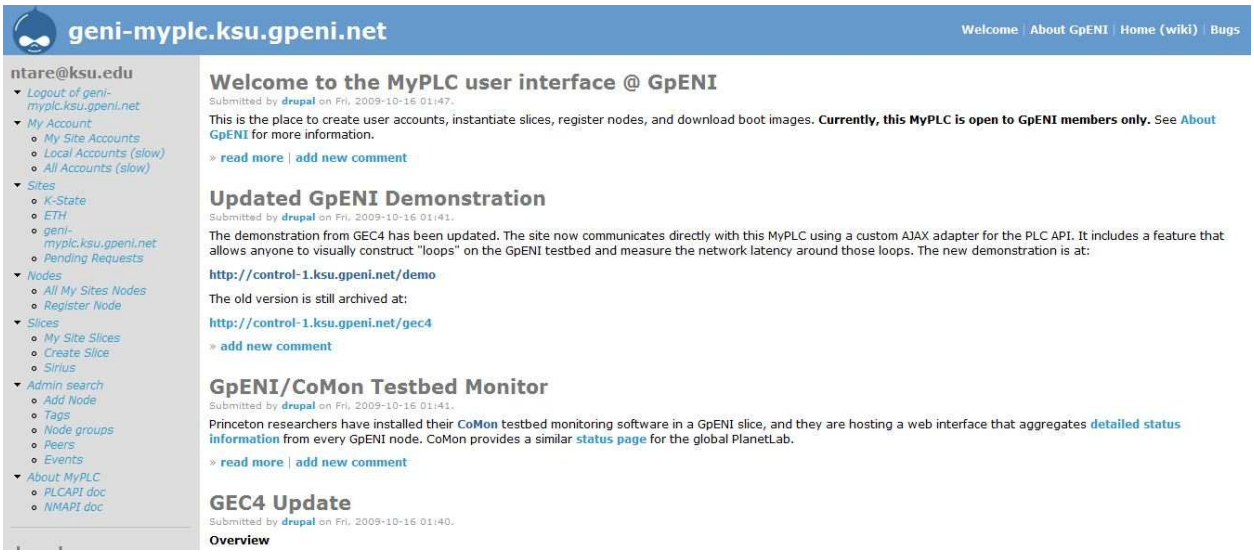

Figure 2.5: The MyPLC Web Interface

The GpENI testbed is based on MyPLC version 4.3 and is currently hosted at Kansas State University. The installation procedure, software releases, and build process are located at the SVN website of PlanetLab [37]. The controller provides a graphical user interface, as shown in the Figure 2.4, for researchers to view the Cluster sites, users, manage accounts, create slice and destroy slice. The PLC defines “roles” to control accessibility to the testbed. These roles are –

1. **User:** The user has the minimum privileges of all and is only allowed to upload his/her public key and add nodes to the slice created by the PI.
2. **Principle Investigator**: The Principle Investigator (PI) is responsible for overall Site management. They can enable, disable and delete user accounts, create a slice, delete a slice, and assign slices to users.
3. **Technical Contact:** A Technical Contact works under a PI and is responsible for installing and administrating PlanetLab nodes at a site.

Once the controller MyPLC is installed, the next step is to install the PlanetLab nodes. The nodes are brought up by running a Live boot CD that provides the minimum PlanetLab image. This image is downloaded from the MyPLC interface with proper network configurations as shown in the Figure 2.5.

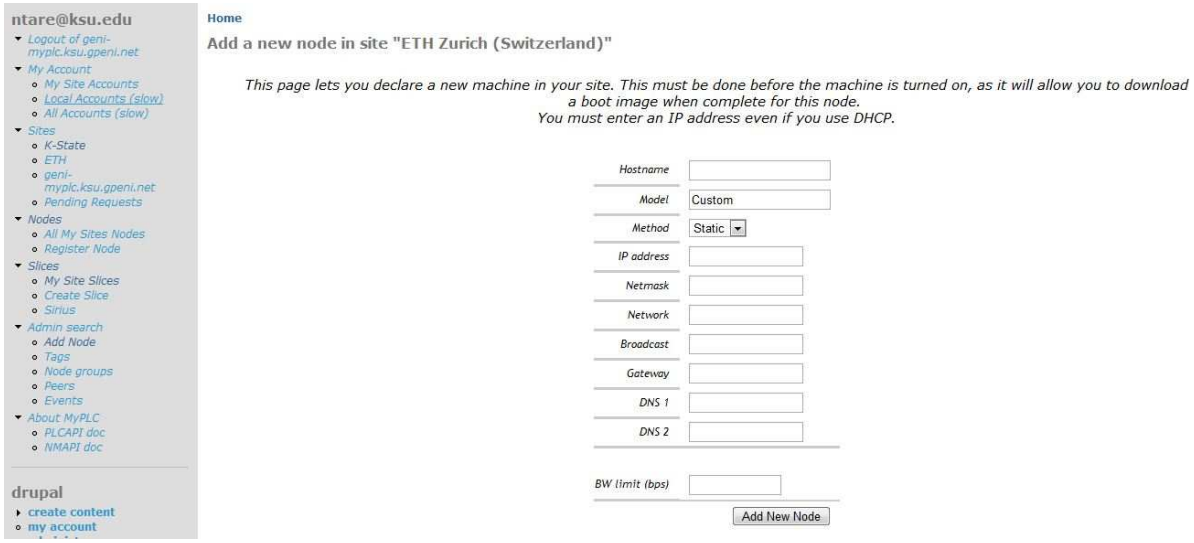

Figure 2.6 Add a PlanetLab node to a site

Using this webportal, a researcher (with user privileges) can create an account, upload public keys, and have a PI create a slice for him. The user adds nodes to the slice and can start deploying their experiment. We will see in the later chapters on how an experiment is designed and deployed.

2.4 Slice-based Facility Architecture (SFA)

The SFA architecture [38] defines a framework to allow slice based facilities in PlanetLab to federate and interoperate with each other. SFA bundles together three software entities: Slice Manager(SM), Aggregate Manager (AM), and Registry in a vanilla PlanetLab configuration. These entities are well defined in a structured code for the MyPLC in the form of python scripts. Each of these entities operates on the set of variables, also the user attributes, like the credentials, certificates, and UIUDs. For instance, when a user logs in, the SM exports an interface for user to verify its private key with the public key stored in the Registry's database. After the user credentials have been verified, the user gets access to the resources defined by the resource specification or the rspecs.

All the calls between the external entity and PLC components take place through the XMLRPC protocol on a secured connection over the Internet. However, the above structure corresponds to a vanilla PlanetLab, in reality there can be multiple SMs and AMs and they can be integrated through a GENI like interface called the Geniwrapper [38]. Much of the Geniwrapper code is obtained from the PlanetLab and it provides a development environment to host aggregates using Cluster B control software. In other words, the Geniwrapper module can be bundled with MyPLC and can be accessed in isolation for other AMs or CMs development. Currently, the most functional part of the Geniwrapper is the Slice Facility Interface (SFI) [38] command line that is a UNIX shell for users to access PlanetLab resources. The SFI commands act on the home registry and slice manager to verify credentials and allocate resources.

2.5 Geniwrapper - Exposing PlanetLab GENI interfaces

Geniwrapper module envelopes the MyPLC code-base and exposes three GENI interfaces (in the Slice Facility Architecture) called Registry (that hold user records like username, slicename, keys, UIDs etc), Slice Manager (that exports the MyPLC management interface - the web-portal) and Aggregate Manager (that exports interfaces relevant to aggregate which in our case corresponds to vanilla PlanetLab) as shown in the Figure 2.7

The SFI provides a UNIX command-line interface to a federation of PlanetLab-based networks that export the programmatic interfaces of Slice Facility Architecture (SFA). It is configured to invoke operations on a “home” registry and slice manager. It takes user’s credential in a configuration file as input and uses them to invoke various slice or registry operations.

At present Geniwrapper provides following set of commands –

1. List
2. Show
3. Add
4. Update
5. setRecord

The output of the configuration steps, along with starting the Geniwrapper module is shown in a small demo in Appendix 1.

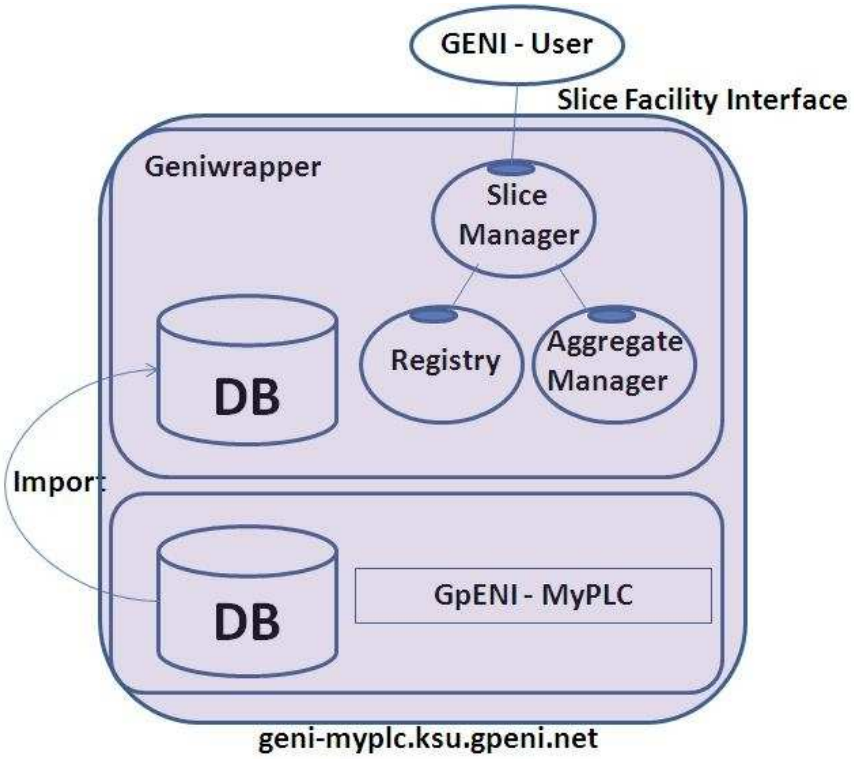

Figure 2.7 The Geniwrapper conceptual diagram

1.6 GpENI integration with other projects in GENI

As explained before, GpENI is a part of a Cluster B that includes different resource aggregates and services (refer Figure 2.1). As a part of Cluster B collaboration activity, GpENI completed vertical integration with Gush and Raven projects. In another collaboration activity, GpENI completed horizontal integration ProtoGENI [39] control framework project called Million Node GENI [40]. In this section we discuss the details of the integrations.

2.6.1 Vertical Integration with GUSH

The GENI User Shell (GUSH), an extension of the PlanetLab user shell (plush), is a remote execution management system for resources on GENI. Every time a user wants to reserve resources on GENI and manage them remotely, GUSH comes into play by locating (resource discovery) them through interaction with GENI Clearinghouse and subsequently managing them through deploy, run, debug, and cleanup
commands.

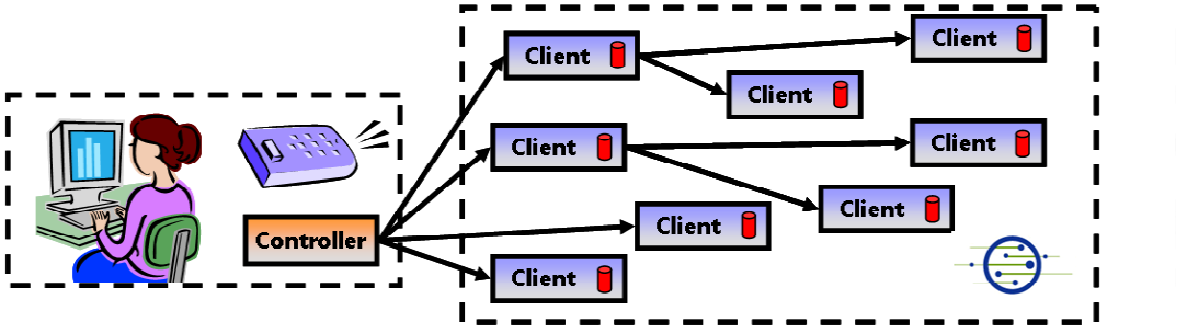

Figure 2.8 GUSH running clients on GpENI and PlanetLab nodes [24]

GUSH demonstrated a simple experiment with GpENI and PlanetLab nodes. As part of the experiment, three XML files were created to inform the GUSH controller about node locations and the slice *gpeni_gush* along with user attributes. GpENI nodes were first added to the GENI Clearinghouse database available through GUSH command line interface and software was deployed on those nodes. With the help of agents installed on nodes (contained in the *gpeni_gush* slice) the controller machine deploys a tarball on both the PlanetLab and GpENI nodes simultaneously using a GUSH command prompt. Once the necessary files are deployed, the command "cat" is executed at the controller to show the files deployed.

2.6.2 Horizontal Integration with Seattle

Seattle [40] is a worldwide networking testbed that runs on computers donated by individuals. Like PlanetLab, Seattle helps users reserve "vessels" on the machines that consume part of the resources in a Seattle-like sandbox environment. Unlike PlanetLab vservers, Seattle allows a restricted execution environment with limited capability by using the programming language of restricted python. Through a very simple web-interface called the Million Node GENI, any user can register and create vessels.

As a part of simple experiment on GpENI nodes, a slice *gpeni_seattle* was created and all the nodes at the four sites were added. Using the script seash.py a command line interface was used to browse and list the added nodes. A pair wise ping test was conducted which displayed latency information on a web-page.

CHAPTER 3 - The ORCA Control Framework

3.1 Introduction

The Open Resource Control Architecture (ORCA) is an extensible architecture for on-demand networked computing infrastructure. Its purpose is to manage diverse computing environments on a common pool of networked hardware resources such as virtualized Xen clusters, storage, sensor motes, actuators, and network elements. The ORCA framework is designed to support utility computing environments [41] where users are able to reserve resources on-demand and subsequently release them.

Standing up a BBNORCA instance was a part of my Intern project at BBN Technologies. This chapter gives a brief overview of the ORCA software architecture and design with a step by step process to set up and maintain a BBNORCA aggregate.

3.2 Role of ORCA in GENI

The role of ORCA is to extend its control software as a GENI control framework, as well as Cluster D, to include the optical resources available in the Breakable Experiment Network (BEN) metro optical testbed [42]. As an integration plan, ORCA aims to connect with the National Lambda Rail (NLR) backbone network and FrameNet service to facilitate end-to-end VLAN connections into BEN and provide external connectivity to GENI researchers.

Relatively less matured compared to other control frameworks, ORCA recently stood up their Clearinghouse integrating components from its cluster projects. Currently, five projects are a part of ORCA control framework – ORCA/BEN, ViSE[43], DOME (an outdoor mobile network testbed) [44], Embedded Real-time Measurement (ERM) (a service to provide access to real-time measurements in ORCA testbeds [45]) and KanseiGENie (a testbed for wireless sensor networks)**.**

3.3 The BBNORCA Project

The ORCA system is a collection of actors that interact using the leasing protocols. Logically, each actor is a server with private internal state that persists in a database. Each provider site or

domain is represented by an authority actor. Each guest environment that consumes resources is represented by an actor called a service manager. Other actors represent brokering intermediaries.

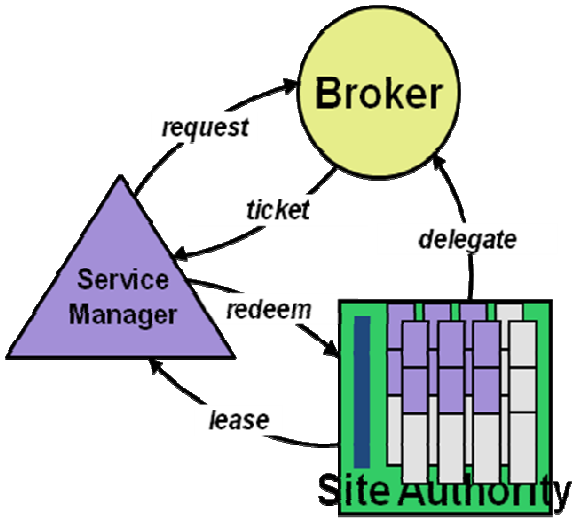

Figure 3.1 ORCA actors and messages exchanged [55]

The three actors represented in the Figure 3.1 are explained as below –

a) **Service Manager (SM)**: This is the guest actor that hosts the Automat web portal for user interaction and is closest to the user. SM is responsible for monitoring application demands and resource status, and negotiates to acquire leases for the mix of resources needed to host the guest [46]. The GENI experimental tool such as Gush, introduced before, uses a SM as an interface point. A service manager negotiates with the broker and sites for leases on behalf of the guest.

b) **Broker (agent or clearinghouse)**: A broker maintains inventories of resources offered by sites, and match requests with their resource supply. A site may maintain its own broker to keep control of its resources, or delegate partial, temporary control to third-party brokers that aggregate resource inventories of multiple sites.

c) **Site (authority)**: A site controls resource allocation at each resource provider site or domain, and is responsible for enforcing isolation among multiple guests hosted on the resources under its control. When a site comes up or is active, it

registers its resources or exports tickets to the broker. The broker in turn advertises them to the slice controller.

3.4 Set up and maintain BBNORCA Aggregate

I set up a BBNORCA aggregate as a part of my Intern project in the summer and fall of 2009. The goal was to configure a simple ORCA subsystem that coordinates machine management consisting of Xen Virtual Machines. The result is an easy-to-use web portal that end-users or administrators access to create and destroy VMs and disk images, bind and unbind machine resources (CPU, memory, and bandwidth) to VMs, and manage VM lifetime. An up-to-date ORCA control software code is provided by the GENI ORCA developers that support resource allocation and management.

Setting up an ORCA aggregate involves several steps that are outlined below. The entire set up consists of an ORCA master, or controller (hake.bbn.com) and an ORCA inventory (geni-ORCA.bbn.com) as shown in the Figure 3.2. The controller machine hosts the container of three actors as explained above. The inventory runs a small piece of ORCA aware software called the Xen drivers, as Xen is the virtual machine hypervisor used. Figure 3.2 shows the functional pieces of the ORCA software tied together.

The controller hosts handlers that talk to the Xen cluster sends inventory specific calls to create and destroy VMs. The node agent on the inventory is a java process that translates the calls from handlers and converts them to Xen specific calls to create DomU images. Cluster-on-Demand is the authority site Shirako [46] plug in.

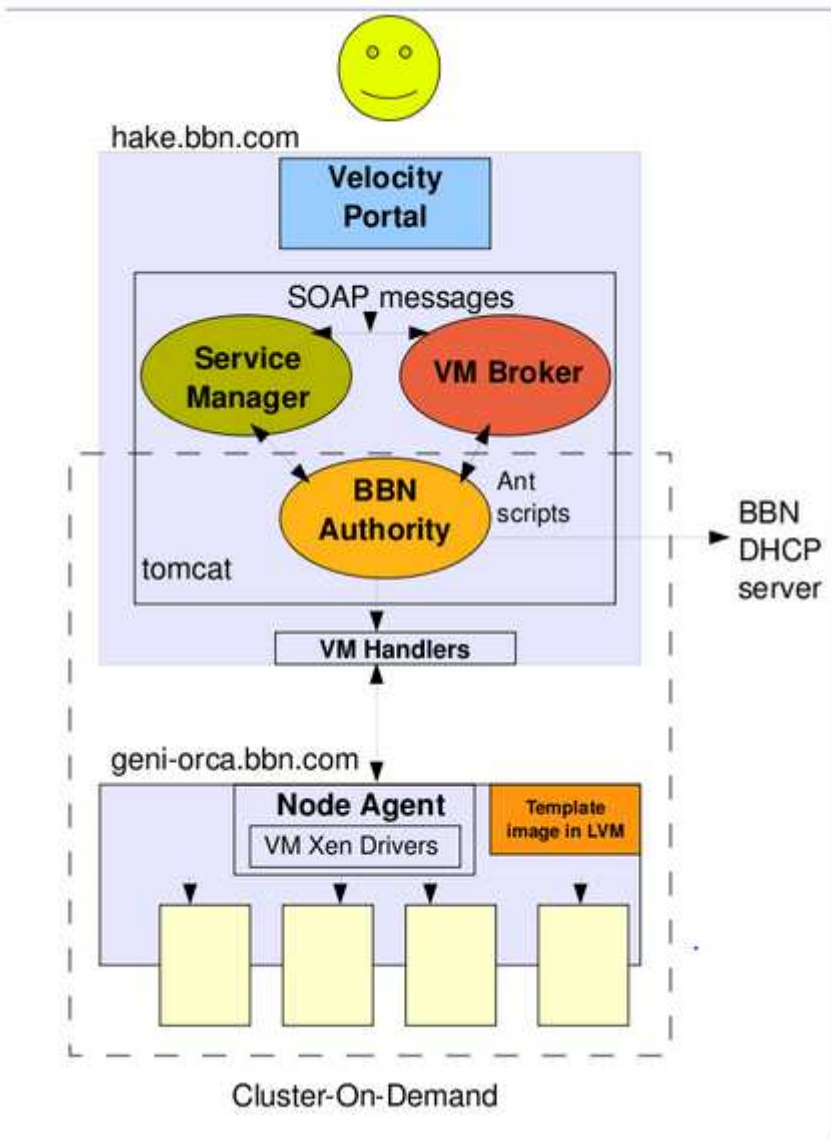

Figure 3.2 ORCA software pieces

1. **Set up ORCA container of three actors**

   This is the first step to set up ORCA with null resources, also called setting up emulated version of ORCA. The detailed steps to set up are also available on the geni website [47] with the current release of their software.

2. **Setup MySQL database configuration**

   ORCA bootstraps from the MySQL database and stores information about users, machines, VMs, and disks. The database allows ORCA master to restart itself in the

event of a failure (e.g. power outage). There are standard settings for loading a MySQL database and connecting it with the ORCA software.

3. **Priming ORCA inventory**

   The BBNORCA set up uses Xen and Logical Volume Manager (LVM) for configuring inventory machines. The idea is to compile a Dom0 kernel or the Xen hypervisor and run Xen DomU kernel that supports kernel-level IP auto-configuration so that ORCA master can set the IP address of each VM it creates on the kernel command line.

4. **Installing node-agent and drivers on Inventory Machine**

   ORCA ships with a set of tools which make the process of installing and securing node agent easy. A node agent is analogous to the component manager of PlanetLab nodes that talk to the controller machines. By securing node agent, we mean to pass public key with the node image.

5. **Setting up proper configuration files**

   ORCA comes with configuration files that set the rspecs [48] of the inventory machine in terms of number of virtual units, CPU share, lease start and end time, and IP related information of the inventory. The configuration files set up the web-portal contents by presenting users with the information related to inventory.

6. **Adding inventory machines to the ORCA database**

   Once all the IP related information regarding the inventory machine is added to the configurations files in the step above, the next step is to add the machine information to the ORCA database. Before adding these machines we need to generate globally unique identifiers (GUIDs) for them. ORCA uses globally unique identifiers to ensure names are unique. For this, ORCA comes with a tool to generate GUIDs. Each inventory machine, for now, is characterized by a machine name, hostname, GUID, IP address, memory, and Boolean flag that specifies its availability.

7. **Using the web portal to manage resources**

   The webportal exposes the three actors which makes it is easy to see the communication taking place between them. The three tabs are the User, Broker, Site, and Admin as shown in the Figure 3.3. After the initial build process, we package the ORCA files in tomcat webapps directory that invokes the actors - broker, site and

machines active in the setup. After the machine becomes active in the Site tab, public key for creating a reservation is added. A series of calls takes place between the SM, Broker, and Site where the end result is the virtual machine attains the Active state.

8. **Example of creating a reservation with Automat**

   Now we discuss the process of creating a reservation once the aggregate is active and ready for operation. The admin tab shows the Actors that get instantiated in one tomcat container. When we click on "View actors", their status is shown as "Online" or "Offline". If the status is "Offline", we need to repackage the web application and bring it to the "Online" state.

Figure 3.3 ORCA web portal showing Online Actors

Next, we verify whether the inventory machines that were added are active. The View Machines in Figure 3.4 shows the status of the machines. The screenshot in the Figure 3.4 shows the machine geni-ORCA as Active and assigned to the actor Site.

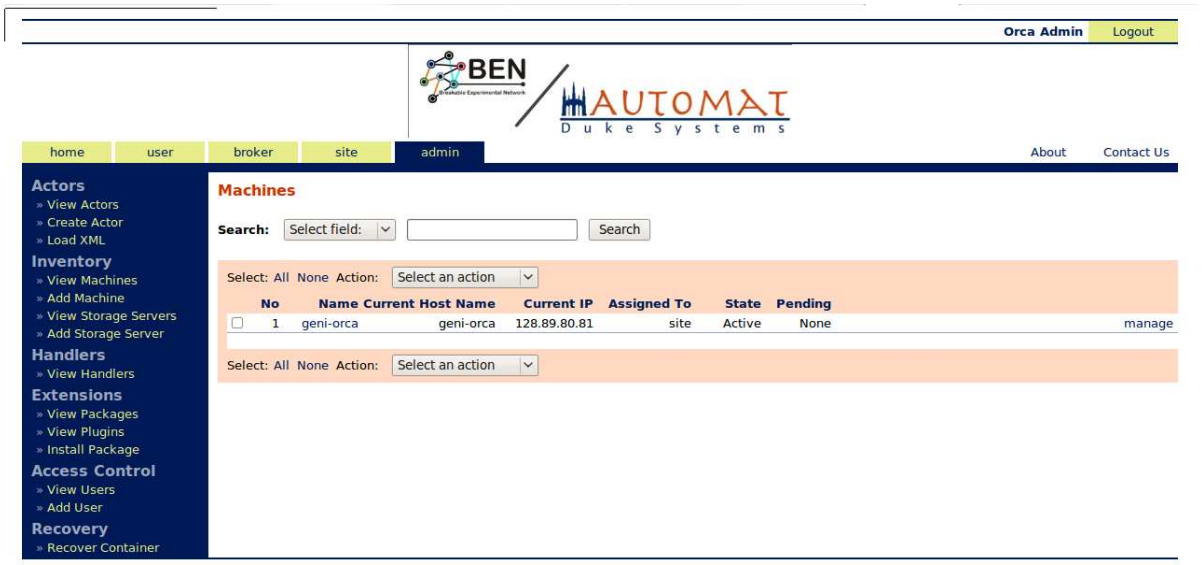

Figure 3.4 ORCA web portal showing Active machine geni-ORCA assigned to site actor

Next, in the broker tab's of the "View Inventory" shows that the Site actor has been assigned to the Broker. Figure 3.5 shows one Site consisting of Xen Virtual Machine. The ticketed state indicates that the site has provided broker with the tickets to allow users see the machines and subsequently reserve them.

Figure 3.5 ORCA web portal showing Xen VM Broker as Ticketed

After verifying the actors and the site status, we create a reservation. ORCA refers to this process as leasing, or getting a ticket for a reservation. A ticket has all the details such as resource type, memory, CPU share, units, lease start, lease end, user public key, image, and OS/kernel.

Figure 3.6 ORCA web portal showing ticket for creating a reservation

The reservation parameters are shown in the Figure 3.6. Creating a reservation process takes a some definite time, where the reservation goes from -

Obtaining Ticket, Ticketed, Redeeming Ticket, Post-Installing, Active

Figure 3.7, shows that the reservation turns out to be Active, or the VM is created. A user can now log in to the VM with the private key corresponding to the public key uploaded during the Create Reservation Process.

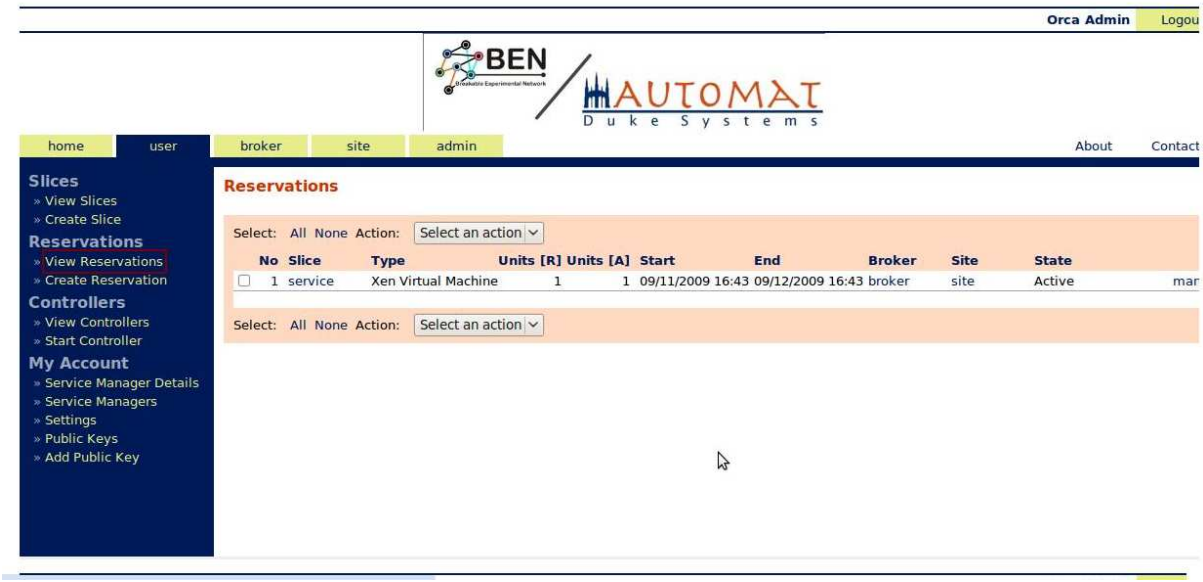

Figure 3.7 ORCA web portal showing Reservation successful and lease Active

3.5 Integration of BBNORCA Aggregate with Cluster D Clearinghouse

A clearinghouse is defined as the "operational grouping of trust anchors for management authorities and slice authorities, slice and component registries, portal for resource discovery and a portal for managing GENI-wide policies" [5]. The ORCA clearinghouse that was stood up recently is a functional integration of Cluster D brokers belonging to individual projects. Prior to the integration, each Cluster D projects ran brokers locally. The federation step enables their brokers to run remotely using secured SOAP protocols over public Internet.

Figure 3.8 shows the remote brokers of the Cluster D projects forming a coherent software entity called Clearinghouse. The steps described above work for running a local ORCA aggregate of self contained actors. In order to run a remote broker, we need to run ORCA in two separate instances of tomcat (also container). One container hosts the SM and the Site actors and the other container hosts the Broker. Later in the process, the broker container is moved to remote Clearinghouse at RENCI.

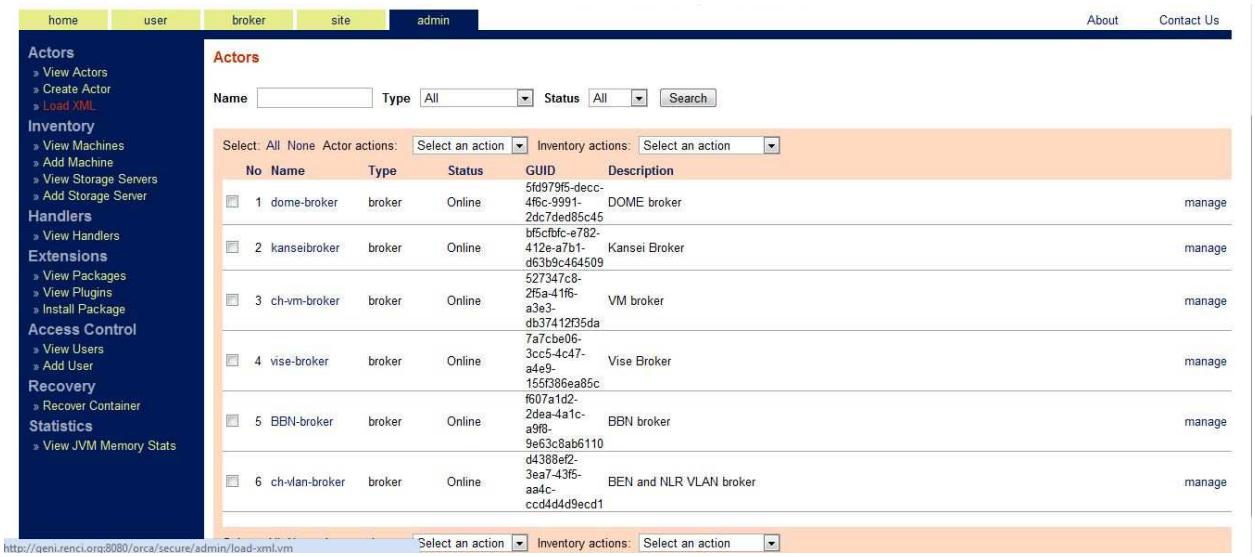

Figure 3.8 ORCA web portal shows brokers of Cluster D projects as Online

.

CHAPTER 4 - GENI Experiment Design

4.1 Introduction

The ultimate goal of a testbed is its usability that helps to achieve the goals of the experiment. To make experiments repeatable a programmable testbed such as GENI is useful. Another important characteristic of experiments is their isolation from each other on a shared testbed environment which testbeds like GpENI provide. This chapter discusses the experiments that were deployed on two testbeds – GpENI PlanetLab testbed and the BBNORCA. In the GpENI experiment, we conducted tests measuring TCP throughput and latencies in the underlying network. In case of the BBNORCA testbed, we describe performance analysis of an experiment measuring leasing overhead for active and failed VMs.

4.2 GpENI Experiment Design and Performance Analysis

Overlay Networks attempt to improve end-to-end application performance and availability. In an overlay network, traffic is routed by nodes that act as application level routers. Overlay networks attempt to leverage the inherent redundancy of the Internet's underlying routing infrastructure to detour packets along an alternate path when the Internet chosen path is congested or unavailable or offers high latency or low throughput. The Overlay experiment designed for GpENI testbed studies the end-to-end network performance between GpENI nodes spanning at five different sites – KSU, KU, UNL, CAMUK, and BERN. The performance metrics used are Round trip time and application TCP throughput.

In this section we will discuss GpENI network measurement experiment. In the first part we measure the delays using ping and UDP between each node pair in a mesh topology. We begin the experiment by reserving a slice *ksu_ron* using the MyPLC controller interface and add nodes to the slice. For the RTT tests, an application script named runclient.sh is deployed on the nodes. The script consists of ping command for a count of 1000 packets. The application's output is written into files that are later transferred to a central machine for further processing. In the second test using UDP, the round trip delays are calculated for a UDP connection between node pairs. We perform TCP throughput test using Iperf[57] tool that was run between

node pairs. TCP tuning was performed to get better results. The GpENI test topology is show in figure 4.1.

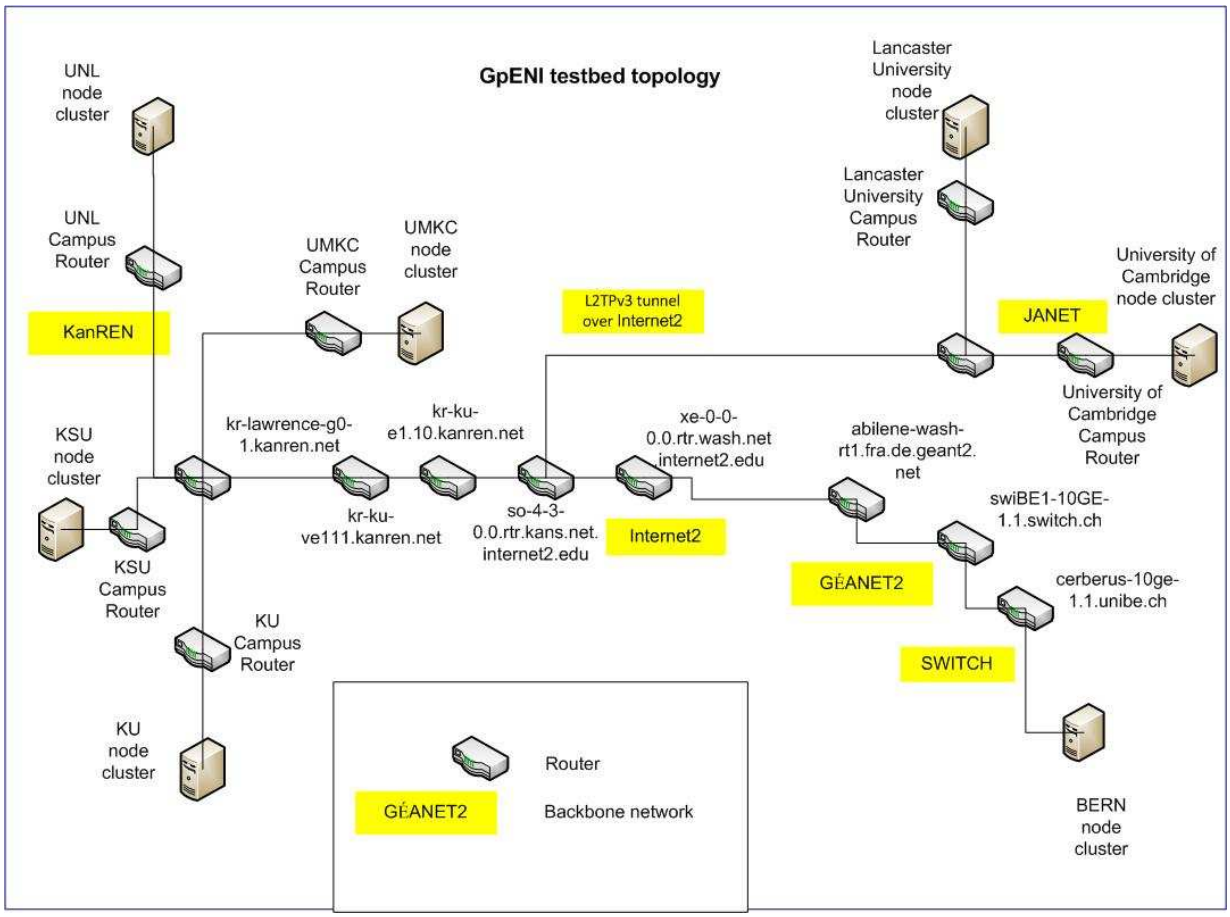

Figure 4.1 GpENI experiment topology

4.2.1 Experiment Procedure

Network measurements are significant tests to characterize the links on a large scale testbed like GpENI. PlanetLab provides an overlay network and network measurements play an important role to characterize the underlay network. In our experiments, we conducted latency and throughput measurements of the underlay links in GpENI.

The first step in the experiment is to reserve a slice using the MyPLC web portal. A slice is constitutes of some compute resources such as processing, memory, storage and network resources such as bandwidth. In this step, the slice named *ksu_ron* is created and 15 nodes at 5

sites are added using the web portal. This is show in figure 4.2. The nodes can be added and deleted based on the experiment's need. Once the slice is instantiated, the MyPLC exchanges a bunch of messages with the Node's component manager and populates user information (public key and login) along with slice attributes to the nodes. A virtual machine corresponding to the user is set up in about 15 minutes and the nodes are accessible via ssh as shown in figure 4.3.

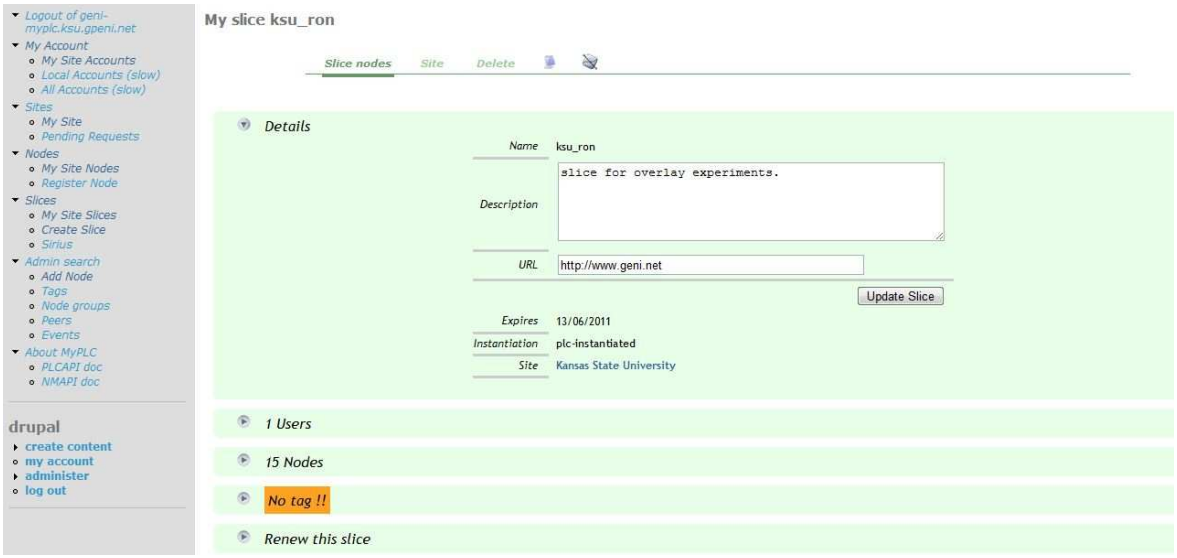

Figure 4.2 MyPLC web-portal to reserve a slice

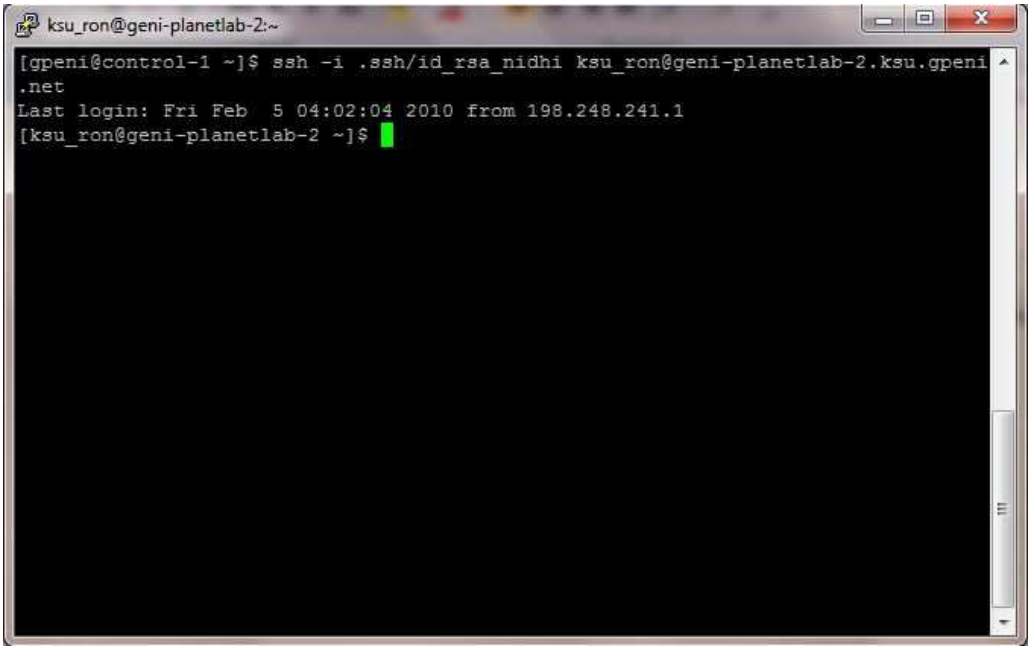

Figure 4.3 Login into GpENI nodes

After the public key propagates to the nodes, we begin the experiments. In the first part of test 1, we measured Round Trip Time in an in a mesh topology using *ping* tool. A shell script was copied to all the nodes using the CoDeploy [58] synchronization tool. The script records the all pair wise ping values in the local files on every node and which are retrieved from a central server for further processing. The results are summarized in table 4.1. Although ping tool gives us round trip delays, we are more interested in finding the end-to-end delays between PlanetLab VMs we created in the slice. To do this, we introduce a second experiment.

In the second part of test 1 we calculated end to end latency using a UDP application tool. A set of python based client and server scripts were deployed on pair of nodes and small UDP packets were sent out on the network. The delay is calculated as the difference in the timestamps of the sent and received packets. Table 4.2 shows the values. As we can see the UDP delays are slightly higher than the ping delays. Table 4.3 displays the ratio of delays of a UDP Vs. ping test. The reason for seeing this variation is the difference in the way application tools work and the network protocol on which each one of them is based on. A detailed explanation on this variation is covered in section 4.2.3.

In the test 2 we measured throughput using Iperf. Iperf is an application level tool to measure maximum network performance using TCP for throughput and UDP for jitter and datagram loss. In order to determine the maximum throughput of GpENI links provisioned by KanREN, we needed to alter TCP window parameters in the PlanetLab kernel 2.6.22. We vary certain parameters such as rmem_max (receiver window) and wmem_max (sender TCP window) for this purpose. The relevant files are located at /proc/sys/net/ipv4. We increase the sender's window size from the default size of 131071 bytes (default window size) to 4194304 bytes. After the window augmentation, we are able to set window size as high as 4MB. Ideally window size is set to Bandwidth Delay Product of the network for optimal performance. The results section gives the outcome of tests conducted between KSU and KU, Bern, Cambridge, Lancaster nodes.

4.2.2 Results and Conclusions

Table 4.1 shows the shows the average delay between a set of five GpENI node clusters, measured on 23 Nov. 2009, beginning at 09:00. For each cluster pair, a ping command was issued 6 times at 2 hour intervals. The packet size used for ping is 64KB packets, with a repeat count of 1000 and an interval of 1 sec. We note the average of latencies for 6000 packets were sent over a 9 hour period.

	KSU	KU	UNL	CAMUK	BERN
KSU	-	2	9	145	155
KU	2	-	6	143	153
UNL	9	6	-	152	158
CAMUK	145	143	152	-	295
BERN	155	153	158	295	-

Table 4.1 Average ping times in msecs between GpENI sites

Clearly, the overlay topology is not well represented by the layer -2 underlay topology since the traffic between the CAMUK and BERN is backhauled through KU [59] as shown in figure 4.1.

In our next experiment, we determined the delays using UDP based application. A packet of 64bytes was transmitted from the source to destination and then back. The time to transfer is calculated for a round trip. Table 4.2, shows the average values of the 1000 UDP delays (in msecs) calculated between node pair.

	KSU	KU	UNL	CAMUK	BERN
KSU	0.47	3.35	11.94	159.45	150.23
KU	3.35	0.41	7.55	154.96	148.94
UNL	11.94	7.55	0.35	166.56	158.32
CAMUK	159.45	154.96	166.56	0.39	298.66
BERN	150.23	148.94	158.37	298.66	0.23

Table 4.2 Average UDP latency in msecs between GpENI sites

We now present the ratio of UDP over ping delays in the mesh topology in Table 4.3.

	KSU	KU	UNL	CAMUK	BERN
KSU	-	1.36	1.36	1.03	1.02
KU	1.23	-	1.17	1.01	1.01
UNL	1.36	1.17	-	1.05	1.05
CAMUK	1.03	1.01	1.05	-	1.00
BERN	1.02	1.01	1.05	1.00	-

Table 4.3 Ratio of UDP latency to ping latency between GpENI node clusters

As seen from Table 4.3, the ratio is greater than 1 which shows that UDP latency involves UDP packet processing overhead over ping packets. The other reason which could induce this behavior is the filtering action taking place in the intermediate network devices based on protocols – ICMP and UDP which is hidden from the users.

In the second experiment we measure TCP throughput using Iperf test. The figure 4.4 shows the variation of Throughput achieved over variable sender window size. Likewise, other Iperf tests were conducted between KSU and KU, Bern, Cambridge, Lancaster nodes and results are summarized in Figure 4.4, 4.5, 4.6, 4.7 respectively.

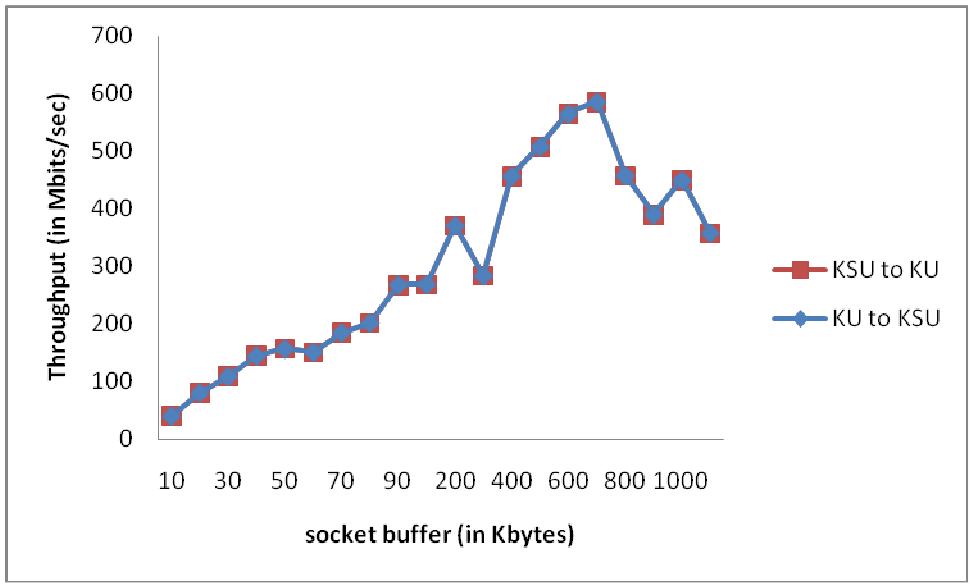

Figure 4.4 TCP Iperf test between KSU and KU

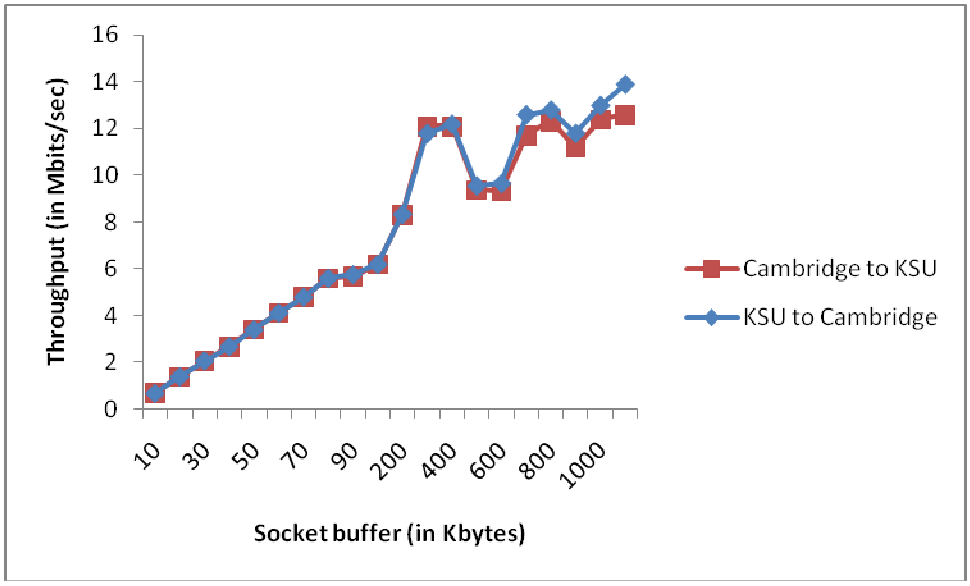

Figure 4.5 TCP Iperf test between KSU and Cambridge

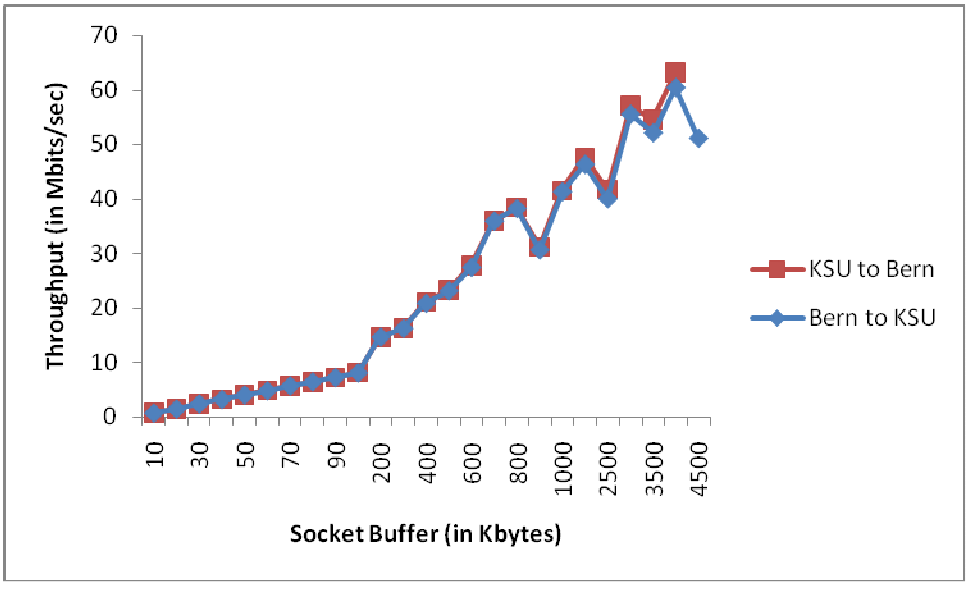

Figure 4.6 TCP Iperf test between KSU and BERN

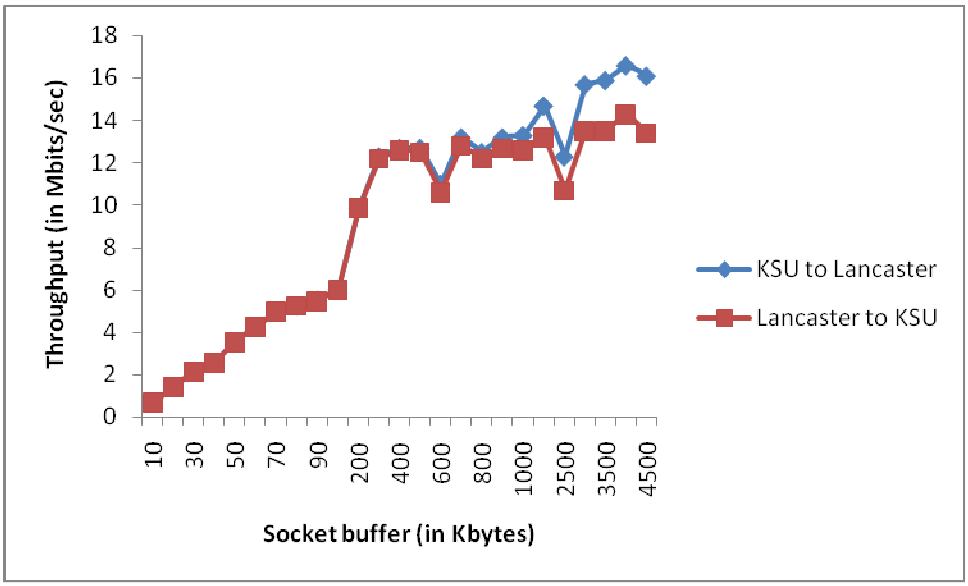

Figure 4.7 TCP Iperf test between KSU and Lancaster

The throughput values increase with an increase in the sender window size as more and more packets are pushed into the network resulting in the higher utilization of the link. Even though the KSU-KU link capacity is 1Gbps (as provisioned by KanREN) we always get a lower throughput value due to presence of bottleneck links in the network. For example the traffic from KSU nodes to KanREN pass through multiple campus POPs that further reduce the throughput calculation. For example, a typical IP traffic path (POPs shown) between KSU and UNL is - (from KSU campus) -> kr-ksu-pplant -> kr-ku -> kr-kumc -> kr-bryant -> (to GPN switch and UNL fiber system).

Another observation from the graphs is the variation in the bi-directional throughput in the links which is attributed to the difference in the paths taken by the traffic. This effect is more prominent where trans-continental links are involved in the test. For example the test involving GPN nodes and European nodes the throughput drops down to 100Mbps due to presence of multiple links in the Wide Area Network.

4.3 BBNORCA Experiment Design and Performance Analysis

We evaluate the BBNORCA aggregate, by conducting an experiment to determine leasing overhead in getting a new VM. We examine the latency and overhead to lease a Xen cluster in a local testbed environment. By local, we mean the broker stays in local ORCA container and not remote as in the integrated case. We begin the experiment by getting the web portal up and creating a reservation for 1 VM. During the reservation messages are exchanged between actors and the leasing state changes as the lease completes that are recorded in a log file. We study these log files based on that we identify the key messages exchanged during leasing mechanism s described in Figure 4.4.

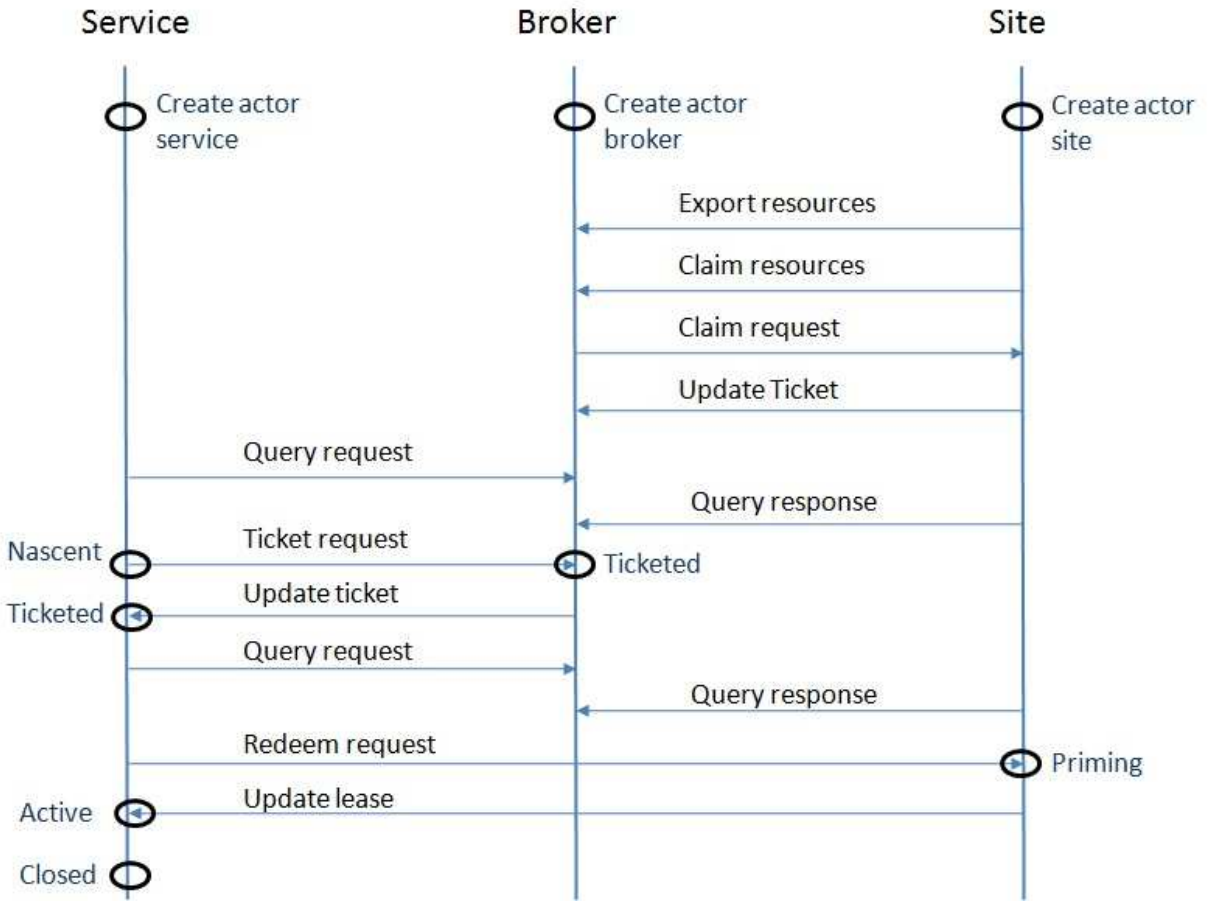

Figure 4.8 Interaction between ORCA actors and their leasing state

4.31 Results and Conclusions

In the first phase resources are exported from site to broker so that broker can advertise them to the experimenters. In the next phase, the service manager starts querying broker on the available resources and after getting a response from broker, it sends out another request for ticket called the ticket request. A ticket includes the units of the VMs requested, CPU %, and user's public key. When it gets the ticket accepted, the service manager redeems the request for the site actor as shown in Figure 4.4. The broker sends back an update lease message confirming the reservation. The active state of the lease indicates the successful VM reservation. This whole process of leasing introduces an overhead which is what our experiment is investigating.

We identify key events in the leasing process of a successful VM and a failed VM. Based on the messages we find out the time stamps and record it in a table. The Figure 4.5 shows the leasing overhead incurred in creating a reservation. All the events are marked on the x axis and the delays are indicated on the Y axis.

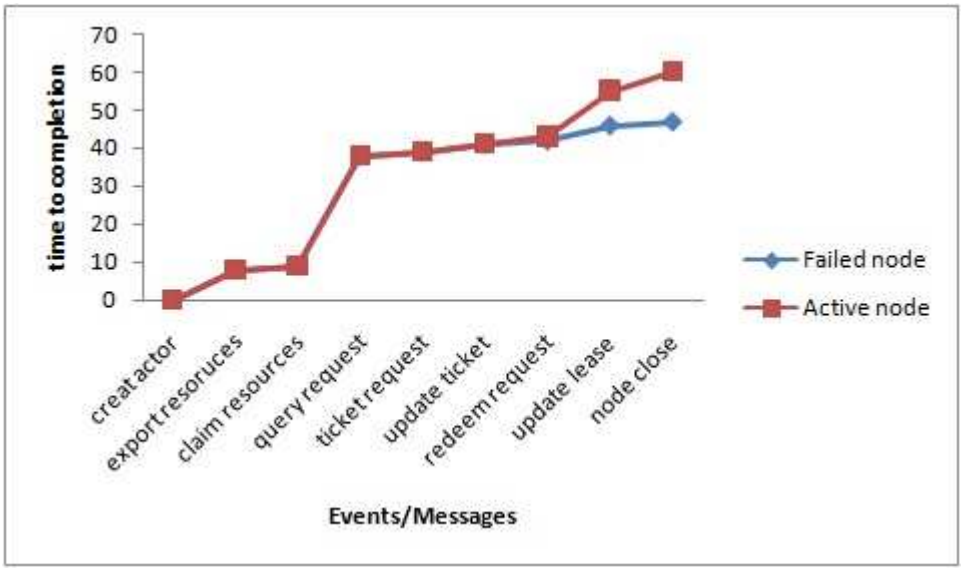

Figure 4.9 Leasing overhead in an ORCA reservation, time in secs

CHAPTER 5 - Comparative Performance Analysis of BBNORCA and GpENI Aggregates

5.1 Introduction

So far we discussed two control frameworks – the PlanetLab and the ORCA. We also explained the process of setting up the GpENI PlanetLab and the BBNORCA aggregate. In the previous chapter we demonstrated simple experiment on the testbeds. With the goal of GENI in mind, we will now investigate the technology readiness and suitability of the competitive control frameworks. The former is determined by the operational stability of the aggregate and ease of usability. In this chapter, we discuss the performance metrics used to evaluate the two aggregates and subsequently the experiment conducted on the same.

5.2 Metrics for Comparison

A GENI control framework defines interfaces between all entities, message types including basic protocols and required functions, and message flows necessary to realize key experiment scenarios. [53]. we evaluate the BBNORCA and GpENI aggregates along with their control frameworks capabilities based on the following metrics –

1. **Management**: PlanetLab implements a centralized management while ORCA implements a distributed management. For example, a site administrator in ORCA has the total control over the resources and can set control policies for testbed users. The broker is an intermediate authority that simply advertises the resources. On the other hand, a GpENI PlanetLab site administrator has to relinquish the control of resources to the controller that control the resource allocation for testbed users. Once the PlanetLab node (aggregate) comes up, the site administrator has no control over the node usage.
2. **Resource Types**: The BBNORCA aggregate uses a Xen capable inventory machine as a resource. As discussed in chapter 3, Xen handlers are invoked when resources enter

or leave slice. These handlers in turn invoke the Xen drivers installed within the node agent (component). ORCA provides a flexibility to run plug in resource specific handlers and drivers for that aggregate without significantly changing the control software code. In case of PlanetLab, the Component Manager (CM) runs as the live boot CD on the PL nodes that provide a static node image. The Node Manager cannot control resources across a site or cluster.

3. **Leasing Overhead:** Leasing Overhead is defined as the total number of messages exchanged in obtaining a resource or the total time required in obtaining a slice. In case of ORCA, it is defined as the time required in starting a tomcat container to completing a reservation. Its value is approximately 50 s in case of ORCA in a local aggregate setup.
4. **OS Environment**: In the GENI speak, once the slice is claimed, we need to prepare the sliver environment as the first step to begin the experiment. The BBNORCA's service manager passes a string to identify an OS configuration from among a menu of options approved by the site authority as compatible with the machine type. On the other hand, in the PlanetLab GpENI nodes, the OS type is fixed and is the same that comes along with the boot CD.
5. **Resource Scheduling**: PlanetLab uses leases to manage the lifetime of its guests, rather than for resource control or adaptation. The lease coming with the reservation has a start and stop time to indicate the time duration the resources are reserved. While ORCA allows actors to negotiate lease contracts at any time. Lease diagram in ORCA is show in the Figure 4.4.
6. **Isolation of experiment slices**: PlanetLab emphasizes best-effort open access over admission control; there is no basis to negotiate resources for predictable service quality or isolation. The reason for this behavior is due to virtualization technology. PlanetLab nodes use Linux Vservers that provide Operating System-level virtualization where the VMs created share the hardware resources. Currently Vservers do not support virtual network device drivers and all the VMs get the same IP. While BBNORCA node uses Xen virtualization that provide hardware level virtualization (paravirtualization) to achieve higher performance [50]. It provides separate virtual network driver for the VMs and hence each VM gets a different IP addresses.

7. **Oversubscription of resource**: As explained in previous paragraph, the Vservers virtualize servers on operating system (kernel) level which permits the creation of many independent Virtual Private Servers (VPS) that run simultaneously on a single physical server at full speed, efficiently sharing hardware resources. However the performance isolation and security is compromised [54]. The performance isolation is superior in the case of the ORCA aggregate however the maximum possible VMs are limited to fifty.

CHAPTER 6 - Conclusion and Future Work

In this work we discussed about the GENI research initiative and laid out the key design features and architectural concepts. We described two control frameworks aggregates GpENI and BBNORCA dealing with their technology readiness and role in GENI. ORCA is a framework that is less mature than PlanetLab as they do not have an operational setup up and running for outside GENI researchers. We then discussed about the experiments that were conducted as a part of the testbed utility. In the case of GpENI we performed a latency and throughput measurement test for the purpose of overlay routing. We conducted all pairs ping test to measure latency. We concluded the experiment by saying that we can infer an underlay topology on the basis of this overlay topology as in the actual physical topology the path between these pair of nodes is backhauled via KU. This experiment provided us with an insight into the technological readiness of GpENI testbed and the approach to create and teardown a distributed system application. In the case of an ORCA experiment, we conducted a test measuring the leasing overhead in reserving a slice or a VM instance. The overhead is measured in seconds and is found out to be 10 times lesser than the leasing overhead in GpENI. The reasons attributed to this behavior is due to message calls that are SOAP based on case of ORCA and are XMLRPC based in case of PlanetLab.

The future steps in the individual projects are to be able to conduct experiments layer 3 and below using the VINI nodes and VLANs provisioned by the Netgear switch. The infrastructure supporting this functionality is still building up and hardware resources are been provided. In the case of ORCA, an interesting experiment would be to establish an end-to-end slice with the BEN substrate using the physical machines at BBN and Duke. This would require establishing VLAN from NOX facility to the National Lambda Rail and all the way to BEN. Passing through multiple networks, stitching VLANs is the best option that requires conformity between campus and public Internet along with usage of appropriate Layer 2 equipments.

Broadly speaking, the lessons learned through plan, build, test, analyze, experiment and deploy phases assist in evaluating GENI's progress. In the year 2 of GENI, new and existing prototypes are providing enhanced features and interfaces for other prototypes to plug in. With

this incremental progress the goal of GENI is realizable with the hope of providing society with a robust, scalable and promising future Internet architecture.

References

[1] Definition of Internet http://en.wikipedia.org/wiki/Internet, December 2009

[2] Distributed Computing Environment

http://en.wikipedia.org/wiki/Distributed_Computing_Environment, December 2009

[3] Distributed Component Object Model

http://en.wikipedia.org/wiki/Distributed_Component_Object_Model, December 2009

[4] Common Object Request Broker Architecture

http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture , December 2009

[5] GENI Website http://www.geni.net, September 2008

[6] Network Simulator NS-2 http://www.isi.edu/nsnam/ns/, December 2009

[7] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, Abhijeet Joglekar, An Integrated Experimental Environment for Distributed Systems and Networks, December 2002, *5th Symposium on Operating Systems Design & Implementation*, OSDI' 02, pages 1-2

[8] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris, Resilient Overlay Networks, *Proc. 18th ACM SOSP*, Banff, Canada, October 2001 pages 1-2

[9] A Blueprint for Introducing Disruptive Technology into the Internet. L. Peterson, T. Anderson, D. Culler, and T. Roscoe. *(HotNets-I '02),* October 2002

[10] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh, Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols, *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2005)*

[11] Guru Parulkar: A Presentation on Internet at Crossroads: Controlled IP or Overlay Networks ?, *Next Generation Networks (NGN) Conference 2001.*

[12] DRAGON website - http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/WebHome

[13] DETER website - http://www.isi.edu/deter/

[14] Kansei Website - http://ceti.cse.ohio-state.edu/kansei/

[15] The PlanetLab Consortium Website http://www.planet-lab.org/

[16] CoMon Website http://comon.cs.princeton.edu/

[17] CoDeen Website http://codeen.cs.princeton.edu/

[18] Planetflow Website http://planetflow.planet-lab.org/#planetlab

[19] OneLab: Future internet test beds. http://www.onelab.eu/, December 2009.

[20] James P.G. Sterbenz, Deep Medhi, Greg Monaco, Byrav Ramamurthy, Caterina Scoglio, Baek-Young Choi, Joseph B. Evans, Don Gruenbacher, Ronqing Hui, Wesley Kaplow, Gary Minden, Jeff Verrant, *GpENI: Great Plains Environment for Network Innovation (Proposal),* ITTC technical report ITTC-FY2009-TR-0061349-01, The University of Kansas, October 2008

[21] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford, In VINI Veritas: Realistic and Controlled Network Experimentation, *In Proc. ACM SIGCOMM, Sept. 2006.*

[22] J. Turner, P. Crowley, J. Dehart, A. Freestone, B. Heller, F. Kuhms, S. Kumar, J. Lockwood, J. Lu, M.Wilson, C. Wiseman, D. Zar, Supercharging PlanetLab - High Performance, Multi-Application, Overlay Network Platform, *In Proc. SIGCOMM Proceedings* 2007

[23] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review, 38, 2 (April 2008)*

[24] Gush: Geni user shell, http://gush.cs.williams.edu/trac/gush, December 2009.

[25] Raven provisioning service, http://raven.cs.arizona.edu/, December 2009.

[26] Bobby Bhattacharjee, Ken Calvert, Jim Gri_oen, Neil Spring, and James Sterbenz. Postmodern *internetwork architecture. Technical Report ITTC-FY2006- TR-45030-01*, Information and Telecommunication Center, 2335 Irving Hill Road, Lawrence, KS 66045-7612, February 2006.

[27] ResumeNet http://www.resumenet.eu/project/index, December 2009

[28] Quagga routing suite. http://www.quagga.net/, December 2009.

[29] XORP: Extensible open-source routing platform. http://www.xorp.org/, December 2009.

[30] The Click modular router project. http://read.cs.ucla.edu/click/, December 2009.

[31] G_EANT2. http://www.geant2.net/, December 2009.

[32] NORDUnet: Nordic infrastructure for research & education. http://www.nordu.net/, December 2009.

[33] JANET: The UK's education and research network. http://www.ja.net/, December 2009.

[34] Cacti: the complete rrdtool-based graphing solution. http://www.cacti.net/,December 2009.

[35] Nagios. http://www.nagios.org/, December 2009.GpENI 15

[36] Zenoss: Unlegacy enterprise it management. http://www.zenoss.com/, December 2009.

[37] MyPLC user guide http://svn.planet-lab.org/wiki/MyPLCUserGuide, January 2009.

[38] Larry Peterson, Soner Sevinc, Jay Lepreau, Robert Ricci, John Wroclawski, Ted Faber, and Stephen Schwab. Slice-based facility architecture. Technical report, Princeton, November 2007.

[39] ProtoGENI control framework http://www.protogeni.net/trac/protogeni, Dec 2009

[40] Seattle Testbed, https://seattle.cs.washington.edu/wiki, December 2009

[41] Utility Computing, http://en.wikipedia.org/wiki/Utility_computing, December 2009

[42] BEN: Breakable Experimental Network, https://ben.renci.org/, December 2009

[43] Virtualized Sensing Environment, http://vise.cs.umass.edu/trac/, December 2009

[44] Diverse Outdoor Mobile Environment, http://prisms.cs.umass.edu/dome/, December 2009

[45] Embedded Real Time Measurements, http://groups.geni.net/geni/wiki/Embedded%20Real-Time%20Measurements

[46] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, K. Yocum, Sharing Networked Resources with Brokered Leases, In USENIX Annual Technical Conference, June 2006.

[47] Configure Xen ORCA aggregate http://groups.geni.net/geni/wiki/BBN_ORCA, November 2009.

[48] Ted Faber (USC/ISI) & Rob Ricci (University of Utah), Resource description in GENI: Rspec model, Second GENI Engineering Meeting 3 March, 2008.

[49] D. W. Junghee Han and F. Jahanian. *Topology aware overlay networks*. In *IEEE INFOCOM 2005*.

[50] Linux Vservers Overview http://linux-vserver.org/Overview, December 2009.

[51] GpENI demonstration website. http://control-1.ksu.gpeni.net/demo/, November 2009.

[52] James P.G. Sterbenz, et al., *GpENI: Great Plains Environment for Network Innovation*, www.gpeni.net, 2009.

[53] GENI Project Office (GPO), GENI Control Framework Requirements, January 2009

[54] Paper of Linux Private Servers (VPS) http://linux-vserver.org/Paper December 2009

[55] Jeff Chase*, Open Resource Control for a Programmable Network Substrate,* RENCI BEN Day 08/15/2008

[57] Iperf Website - http://dast.nlanr.net/Projects/Iperf/

[58] CoDeploy Website - http://codeen.cs.princeton.edu/codeploy/

[59] James P.G. Sterbenz1, Deep Medhi, Brav Ramamurthy, Caterina Scoglio, David Hutchison, Bernhard Plattner, Tricha Anjali, Andrew Scot, Cort Buffington, Greg Monaco, Don Gruenbacher, Rick McMullen, Justin P. Rohrer, Pragatheeswaran Angu, Hiyang Qian, Nidhi Tare: The Great Plains Environment for Network Innovation (GpENI): A Programmable Testbed for Future Internet Architecture, *6th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities2010.*