A Modular Architecture for Cloud Federation

by

Rizwan Panjwani
B.Sc., University of Victoria, 2007

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

A Modular Architecture for Cloud Federation

by

Rizwan Panjwani

B.Sc., University of Victoria, 2007

Supervisory Committee

Dr. S. Ganti, Supervisor

(Department of Computer Science)

Dr. Y. Coady, Departmental Member

(Department of Computer Science)

Supervisory Committee

Dr. S. Ganti, Supervisor

(Department of Computer Science)

Dr. Y. Coady, Departmental Member

(Department of Computer Science)

ABSTRACT

Cloud Computing is the next step in the evolution of the Internet. It provides seemingly unlimited computation and storage resources by abstracting the networking, hardware, and software components underneath. However, individual cloud service providers do not have unlimited resources to offer. Some of the tasks demand computational resources that these individual cloud service providers can not fulfill themselves. In such cases, it would be optimal for these providers to borrow resources from each other. The process where different cloud service providers pool their resources is called Cloud Federation. There are many aspects to Cloud Federation such as access control and interoperability. Access control ensures that only the permitted users can access these federated resources. Interoperability enables the end-user to have a seamless experience when accessing resources on federated clouds. In this thesis, we detail our project named GENI-SAVI Federation, in which we federated the GENI and SAVI cloud systems. We focus on the access control portion of the project while also discussing the interoperability aspect of it.

Contents

Supervisory Committee **ii**

Abstract **iii**

Table of Contents **iv**

List of Tables **viii**

List of Figures **ix**

Acknowledgements **xii**

Dedication **xiii**

1 Introduction **1**

1.1 GENI-SAVI Federation Goals 3

1.2 Agenda 4

2 Related Work **5**

2.1 PlanetLab 6

2.1.1 PlanetLab Federation using Para-Virtualization 6

2.1.2 PlanetLab-NITOS Federation 7

2.2 The Reservoir Model for Cloud Computing 8

2.3 Mobile Agent Based Cloud Federation 10

2.4 Peer to Peer Cloud Computing 11
2.5 Broker-Based Cross-Cloud Federation Manager 12
2.6 Federation: The Next Phase of Cloud Computing 14
2.7 Summary 15

3 GENI-SAVI Federation Architecture 16
3.1 Overall Architectural Goal 16
3.2 An example 17
3.3 OpenStack Architecture 18
3.4 GENI 20
3.4.1 Key GENI Concepts 20
3.4.2 GENI Federation of Testbeds 23
3.5 SAVI 24
3.5.1 SAVI Architecture 24
3.6 GENI-SAVI Federation Architecture 28
3.6.1 User Authentication 29
3.6.2 SAGEFed Interface 32
3.7 Summary 34

4 Implementation 35
4.1 User Authentication 35
4.1.1 Third Party Authentication 35
4.1.2 SAVI Keystone for Third Party Authentication 36
4.1.3 Authenticating SAVI Users on GENI 37
4.1.4 GENI as an SP 38
4.1.5 SAVI as an IdP 38
4.1.6 Creating a SAVI User Account for a GENI User 44

4.2 Accessing GENI Resources 46
4.3 Accessing SAVI Resources 47
4.4 SAGEFed 47
4.5 Summary 50

5 Experimentation and Evaluation **51**
5.1 Experiments 51
5.1.1 User Study 51
5.1.2 Experiment 1: A SAVI User Accessing GENI 52
5.1.3 Experiment 2: A GENI User Accessing SAVI 54
5.1.4 Conferences 59
5.2 Evaluation 59
5.3 Summary 61

6 Conclusions **62**
6.1 Future Work 63
6.1.1 UACS 63
6.1.2 SAVI IdP 64
6.1.3 Housekeeping Notes 64
6.2 Final Thoughts 65

A Additional Information **67**
A.1 GENI Racks 67
A.1.1 GENI Rack Specifications 68
A.2 Accessing Resources on GENI 71
A.3 SAVI Usage 80
A.4 GENI-SAVI Federation Tutorial 82
A.4.1 GENI User Accessing SAVI Resources 82

A.4.2 SAVI User Accessing GENI Resources 84

Bibliography **91**

List of Tables

Table 5.1 Experiment 1 - comparing federated and non-federated approach for an experienced user (Participant A) accessing GENI 55

Table 5.2 Experiment 1 - comparing federated and non-federated approach for a new user (Participant B) accessing GENI 55

Table 5.3 Experiment 2 - comparing federated and non-federated approach for an experienced user (Participant A) accessing SAVI 58

Table 5.4 Experiment 2 - comparing federated and non-federated approach for a new user (Participant B) accessing SAVI 58

List of Figures

Figure 2.1 Edwards and Harwood's proposed PlanetLab federation architecture [2]. 7
Figure 2.2 PlanetLab-Europe and NITOS federation architecture [14]. . . 8
Figure 2.3 Reservoir site architecture [5]. 9
Figure 2.4 Mobile agent based open cloud computing federation architecture [6]. 11
Figure 2.5 Peer to Peer cloud federation architecture as proposed by Babaoglu et al. [13]. 12
Figure 2.6 Ranjan and Buyya's Grid-Federation framework [1]. 13
Figure 2.7 Broker-Based Cross-Cloud Federation [17]. 14
Figure 2.8 Broker-Based Cross-Cloud Federation [17] 15

Figure 3.1 Architecture Goal - Overview 17
Figure 3.2 OpenStack Conceptual Architecture (Havana Version) [23] . . 19
Figure 3.3 A GENI Slice [41] 21
Figure 3.4 Relationship between a Sliver, a Slice, and a Project 23
Figure 3.5 SDI resource management architecture [21] 24
Figure 3.6 SAVI entities hosting two sample applications [18] 25
Figure 3.7 Control and management system for a SAVI Node [21] 26
Figure 3.8 SAVI entities hosting two sample applications [21] 27

Figure 3.9 SAVI and GENI users being authenticated and accessing the federated system using the SAGEFed interface. 28
Figure 3.10 A SAVI user being authenticated by SAVI onto GENI 31
Figure 3.11 A GENI user being authenticated onto SAVI 32
Figure 3.12 A user accessing both GENI and SAVI resources using SAGEFed 34

Figure 4.1 A user being authenticated using third party authentication . . 36
Figure 4.2 GENI authenticating a user by redirecting them to SAVI IdP . 39
Figure 4.3 An authentication request being processed by Shibboleth IdP. . 41
Figure 4.4 A SAVI user modification request flowchart 43
Figure 4.5 SAVI account creation for a GENI user 44
Figure 4.6 SAVI User Account Creation Service 46
Figure 4.7 SAGEFed Command Stages 48
Figure 4.8 Anatomy of a SAGEFed command 49

Figure A.1 OpenGENI Software Architecture [50]. 69
Figure A.2 Organization Selection 72
Figure A.3 GENI Login Screen 73
Figure A.4 List of User's GENI Projects 73
Figure A.5 Naming a GENI Slice 74
Figure A.6 Adding resources to a GENI Slice 74
Figure A.7 Choosing a GENI site to add resources. 75
Figure A.8 Selecting a resource type. 75
Figure A.9 Selecting resource properties. 76
Figure A.10 Geni resource creation log. 76
Figure A.11 Geni resource domain name and login information. 77
Figure A.12 SSHing into a GENI VM 77

Figure A.13Geni resource domain name and login information. 78
Figure A.14Creating an instance on a SAVI node 81
Figure A.15GENI user accessing SAVI using federation - Step 1 83
Figure A.16GENI user accessing SAVI using federation - Step 2 83
Figure A.17GENI user accessing SAVI using federation - Step 3 84
Figure A.18GENI user accessing SAVI using federation - Step 4 84
Figure A.19GENI user accessing SAVI using federation - Step 5 85
Figure A.20GENI user accessing SAVI using federation - Step 6 85
Figure A.21SAVI user accessing GENI using federation - Step 1 86
Figure A.22SAVI user accessing GENI using federation - Step 2 86
Figure A.23SAVI user accessing GENI using federation - Step 3 87
Figure A.24SAVI user accessing GENI using federation - Step 4 88
Figure A.25SAVI user accessing GENI using federation - Step 5 88
Figure A.26SAVI user accessing GENI using federation - Step 6 89
Figure A.27SAVI user accessing GENI using federation - Step 7 89
Figure A.28SAVI user accessing GENI using federation - Step 8 90

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Sudhakar Ganti, and Dr. Rick McGeer for mentoring, support, and patience throughout this project.

Dr. Yvonne Coady, Dr. Hadi Bannazadeh, Tom Mitchell, and GENI and SAVI teams for help, support, and guidance with this project and thesis.

Our technology, our machines, is part of our humanity. We created them to extend ourselves, and that is what is unique about human beings.

Ray Kurzweil

DEDICATION

To my family and friends for their love and support.

Chapter 1

Introduction

The Internet now has over 3 billion users [47] or consumers who use it to read, watch videos, shop, engage in social media and much more. The producers such as businesses that provide these services have to worry about the infrastructure of their service. They either have to make a considerable investment in the hardware, software, and network infrastructure by installing it on-site, or they find a web-hosting service that would meet their needs. In addition, they have to hire professionals who would administer the infrastructure.

The needs of Internet consumers and producers have continued to grow. This growth poses considerable infrastructure challenges in terms of scalability, security, and availability. A growing online business should not have to put majority of its time and resources into maintaining and scaling its hardware and software infrastructure. It should be able to focus on its product and its users. Cloud Computing solves exactly this problem. It allows producers to focus on their products without worrying too much about the underlying infrastructure. When a producer subscribes to a cloud service, they get the flexibility of hosting their application on any or as many servers as the cloud service provider makes available. The producer can specify

exactly the amount of the resources they need and scale up as their needs grow. All the infrastructure is managed by the cloud service provider, whereas the producer only needs to worry about their application. The cloud service providers also provide service-level agreements (SLAs) that guarantee certain amount of resources and availability so the producer can rest easy knowing their application won't suddenly become unavailable for hours or days due to some technical issues that the producer may not have the expertise to be able to resolve quickly.

The cloud service providers have continued to grow. There are not only commercial cloud providers such as Amazon, Google, and Microsoft [31, 35, 32], but also academic clouds such as SAVI and GENI [64, 40]. However, the number of consumers and their needs keep growing. There are many applications which need more resources than many of the cloud providers can provide on their own. For example, the GENI network only has servers available in the United States. In this scenario, if a researcher wanted to run an experiment on GENI and wanted to include servers located in Canada, they would be out of luck. To resolve this situation, GENI service provider could pool their resources with SAVI service provider, who have their servers located in Canada. This way, a GENI user could gain access to SAVI resources. This process of cloud service providers pooling their resources together to allow their users to have access to combined resources is called Cloud Federation. The benefits of federating clouds are many: access to more geographical locations, storage and computational resources, etc. Cloud Federation makes it possible for the users to run large scale and distributed computing tasks, which would otherwise be out of their reach.

In this thesis, we discuss our implementation of the GENI-SAVI federation. GENI Project is a cloud system comprised of several academic sites in United States, while SAVI is its counterpart in Canada. We discuss GENI and SAVI in depth in Chapter

3.

1.1 GENI-SAVI Federation Goals

The GENI-SAVI federation project had two goals:

1. Accomplish the GENI-SAVI federation without making any major changes to the existing GENI and SAVI architectures.

2. Enable existing GENI and SAVI users to access both GENI and SAVI resources using a common interface which is intuitive and extensible.

These goals were quite broad, but they served as a guide when we set out to design the system. In the end, we settled on two main components that we had to implement:

1. ***Access Control***: Ability to authenticate GENI and SAVI users so that they can access resources provided by GENI-SAVI federation. Concretely, this means that GENI users should be able access SAVI resources, and SAVI users should be able to access GENI resources using some sort of a seamless authentication mechanism.

2. ***Interoperability***: Enable users to interact with GENI and SAVI resources using a common interface. GENI and SAVI both have their own tools such as Omni and Nova which their users leverage in order to access resources [55, 33]. Our interoperable interface would allow the users to access resources across both GENI and SAVI using a client which can interface with both Omni and Nova APIs (Application Programming Interface) [70].

Throughout the thesis we will demonstrate that the GENI-SAVI Federation project increased the resources available to both GENI and SAVI users. We also demonstrate that our resulting architecture was modular, and required minimal changes to the existing architectures of both the systems. Finally, we demonstrate that our prototype improves the overall user experience by reducing the amount of steps required to access resources on GENI and SAVI as compared to original Nova and Omni tools.

1.2 Agenda

We now describe the outline of this thesis to guide the reader.

Chapter 1 Introduces the reader to GENI and SAVI cloud systems in the context of Cloud Computing Federation and provides justification for their federation. Likewise, it provides a summary of the project and its goals.

Chapter 2 Provides background information on Cloud Federation and discusses some past work on the topic while contrasting it with our GENI-SAVI federation architecture and implementation.

Chapter 3 Discusses GENI and SAVI systems in depth and delves into the GENI-SAVI Federation architecture and its goals.

Chapter 4 Describes the implementation of the GENI-SAVI Federation prototype and gives an overview of the technologies involved.

Chapter 5 Evaluates the project based on the original goals, the resulting prototype, and the user experiments.

Chapter 6 Concludes the thesis by providing a summary of the GENI-SAVI federation project and its accomplishments. Furthermore, it provides future researchers guidance on improvements.

Chapter 2

Related Work

Cloud computing has been around for a while, but it is now reaching mainstream usage [30]. As the technology continues to improve, we see it being used in many areas of business and government [19]. Not only is cloud usage growing in the business and manufacturing [16] sectors, but it is also being used in heavily regulated areas such as healthcare [26]. This is in part driven by widespread embrace of Service Oriented Architecture (SOA), which lets organizations provide ubiquitous services as software [25]. Since these services have high-availability, low-investment requirements, cloud computing seems like a natural solution.

This growth in cloud computing parallels the growth of the Internet [4], where it evolved by isolated networks combining their networks to become a "network of networks". We now see clouds combining their resources to provide benefits such as increased availability of resources, load balancing during peak traffic, and larger geographical coverage [12]. This process of combining and regulating cloud resources is also known as Cloud Federation. This chapter provides an overview of various cloud federation attempts and their architectures.

2.1 PlanetLab

PlanetLab is a globally distributed network currently comprising of 1353 nodes at 717 sites [75]. Since 2003, PlanetLab has enabled researchers to conduct experiments on large-scale distributed topics such as distributed storage, peer-to-peer systems, and network mapping on its global network. From 2003 to 2005, PlanetLab grew from 100 nodes to around 600 nodes globally [3]. There have been a few attempts at federating PlanetLab with other existing cloud systems, which we detail in the sections below.

2.1.1 PlanetLab Federation using Para-Virtualization

As early as 2006, when cloud computing was still relatively young, Edwards and Harwood suggested an approach they called Para-Virtualization in order to federate PlanetLab [2]. Para-Virtualization is a process where the host machine's operating system is modified to make virtualization more efficient. The modification is only made to privileged instructions so that any software running under user space can run unmodified. When applied to PlanetLab, this approach isolates the software that manages virtualized resources - called Management Authority (MA).

Edwards and Harwood propose an architecture which allows users to add nodes from multiple MAs to their *slices* via a Slice Authority (SA)[2]. A *slice* is a way of grouping a user's resources across all the sites. They propose adding a Site Manager (SM) to manage sites and multiple nodes on each machine. In addition, Edwards and Harwood propose adding a Federation Authority (FA) in order to control user access to MAs. Figure 2.1 illustrates the proposed architecture. Under this scheme, a user would gain access to resources by sending a request through the SA, which would then negotiate with MA, SM, and FA to finally grant the user access to the requested resources.

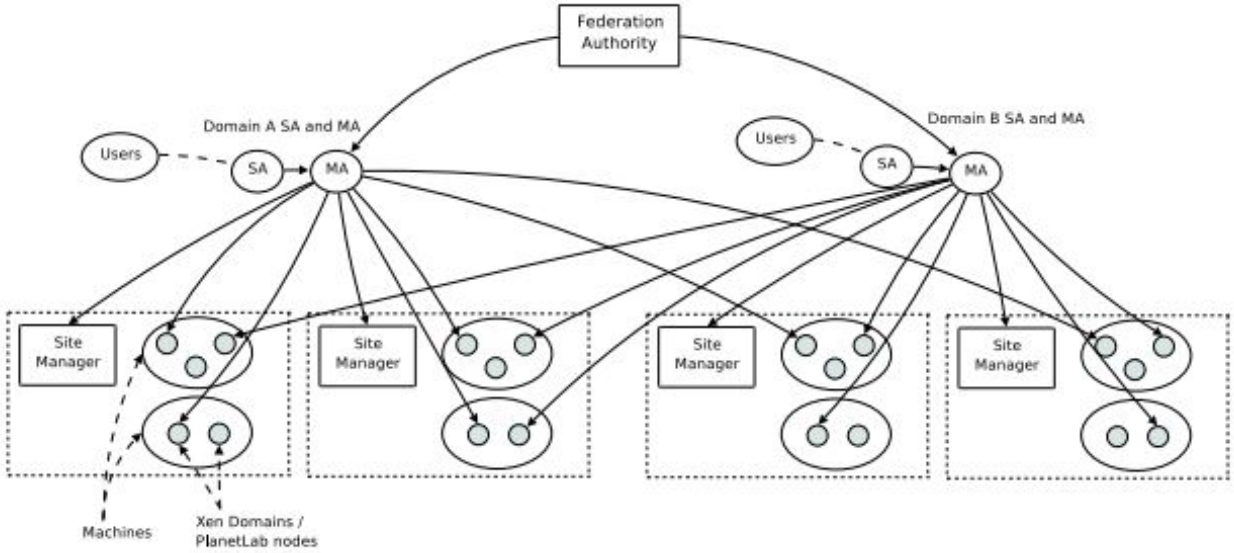

Figure 2.1: Edwards and Harwood's proposed PlanetLab federation architecture [2].

2.1.2 PlanetLab-NITOS Federation

PlanetLab-NITOS federation demo was a part of a larger project called *Onelab2* [14], which focused on federating PlanetLab with multiple other systems. In this demo, Markis et al focus on the federation between PlanetLab Europe (PLE) and NITOS. PLE is the European portion of PlanetLab and consists of around 100 nodes. NITOS is a testbed focusing on wireless experimentation built on OMF [10], which is a control and management framework for networking testbeds.

PLE and NITOS federation is based on three components:

- ***Single Sign Up***: Allows an user of PLE to log into NITOS portal without registering separately.
- ***Deployment of OMF at PLE resources***: Incorporting OMF support for PLE.
- ***XMPP Communication using slices***: Essentially allows a user to connect to a single testbed and access resources on both PLE and NITOS through using

OMF commands as illustrated in Figure 2.2.

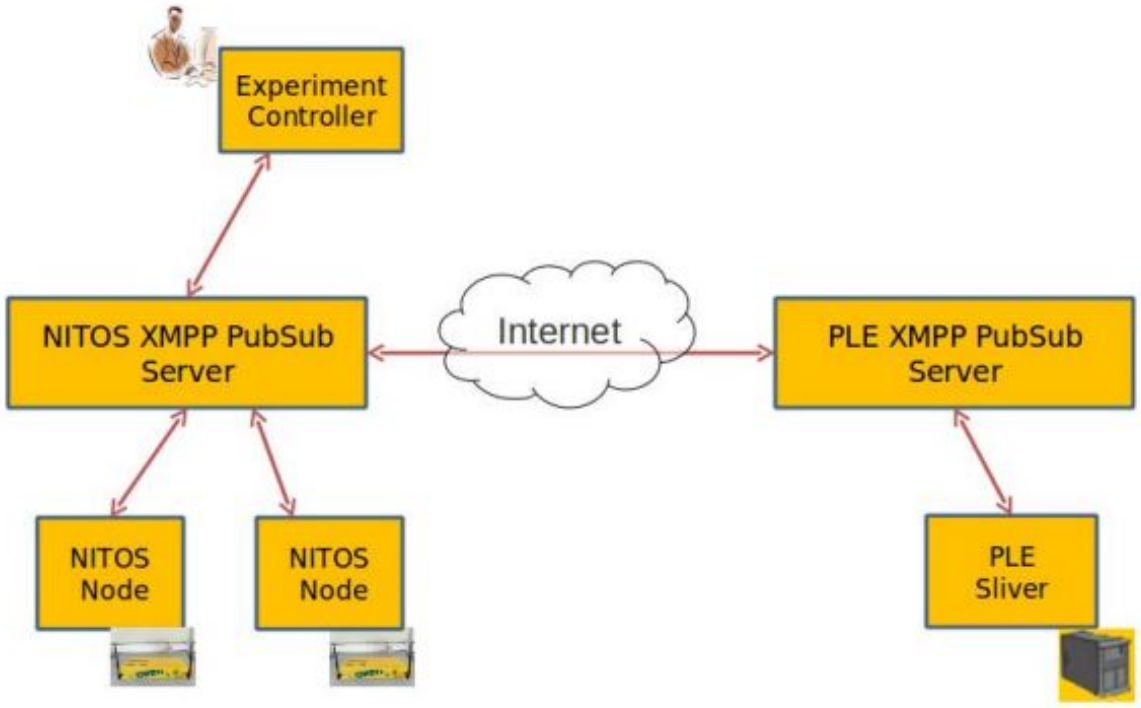

Figure 2.2: PlanetLab-Europe and NITOS federation architecture [14].

2.2 The Reservoir Model for Cloud Computing

The Reservoir Model for cloud computing aims for an ambitious goal: provide enterprise grade federation between cloud services where the cloud providers can maintain autonomous control of their clouds while being able to support infrastructure service-level agreements (SLAs) [5]. The key challenges include servicing thousands of different service components, consolidating many applications on the same infrastructure while minimizing operational cost, and guaranteeing the individual customer SLAs.

In their proposed architecture, Rochwerger et al separate the functional role of service providers and infrastructure providers. Service providers interface with the

customer and lease resources from infrastructure providers accordingly. A Reservoir site contains the physical resources and is owned and operated by infrastructure providers. Figure 2.3 illustrates four major components to each reservoir site:

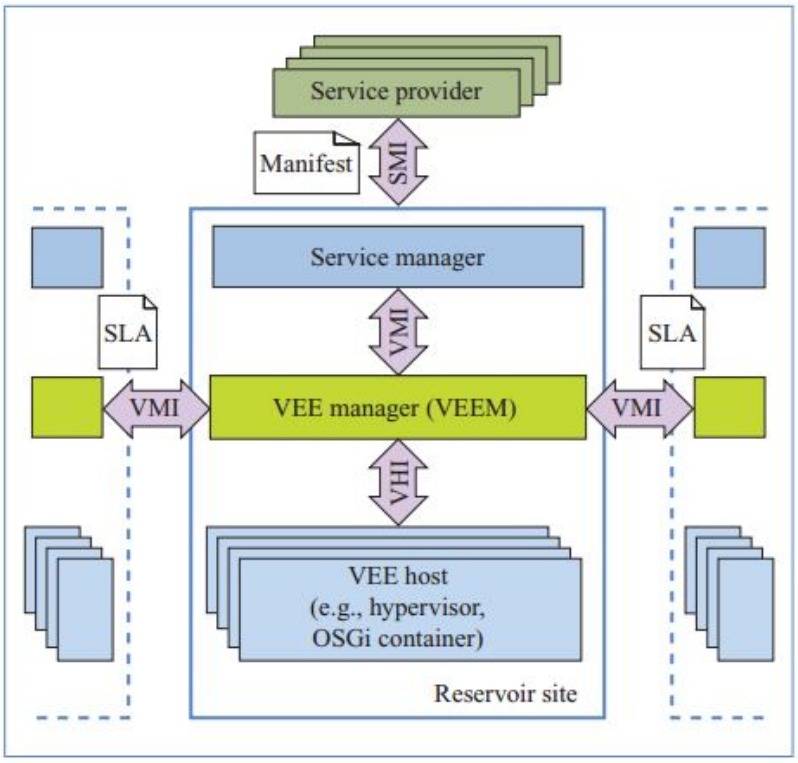

Figure 2.3: Reservoir site architecture [5].

- ***VEE Host***: Virtual execution environments (VEEs) are fully isolated runtime environments, which along with virtualization layer, and management enabling components are referred to collectively as a VEE host.

- ***VEE Manager***: VEE Manager (VEEM) is responsible for optimal placement of VEEs into VEE hosts. VEEM acts intelligently in placing and moving VEEs as long as they satisfy the constraints as dictated by the Service Manager.

- ***Service Manager***: The Service Manager interfaces with the Service Provider to receive service manifests, negotiate pricing, and handle billing. It provisions

and deploys VEEs based on the service requirements and enforces SLAs by either increasing or throttling capacity of a service application.

- ***Service Provider***:A Service Provider interfaces with the Service Manager and translates the requests of customers into service requests that the Service Manager can understand.

In conclusion, the Reservoir architecture takes a unique enterprise approach to cloud federation focusing on satisfying business needs around service-level agreements.

2.3 Mobile Agent Based Cloud Federation

Despite cloud computing's growing use, it suffers from limited scalability and unreliable availability [6]. In addition, the user can not move their application easily from one cloud to another, since different clouds have different requirements and configurations. Zhang and Zhang propose a mobile agent based open cloud computing federation to solve these issues. They propose a Travelling Bag mechanism which encapsulates a user's application code and can run on a MAP (Mobile Agent Place). This effectively makes the user application portable to any cloud.

Figure 2.4 illustrates the mobile agent architecture. A user application is embedded into a mobile agent which is managed by the task manager to run on the MAP. The MAP sits on top of a Java Virtual Machine (JVM) [60], which runs on the physical machine. This architecture is flexible enough to allow other regular non-mobile agent applications to run. Task Manager is able to distinguish such applications and route them to a suitable virtual machine. Each computing region has a Task Manager, and all the task managers can communicate and negotiate with each other to best match the computing requirements of an application to available resources.

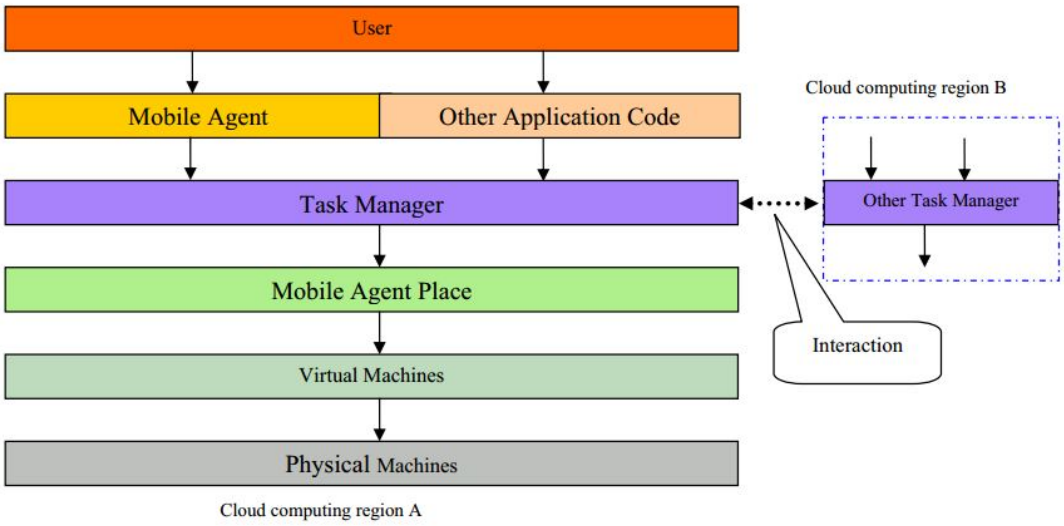

Figure 2.4: Mobile agent based open cloud computing federation architecture [6].

Mobile agent based cloud computing scheme as proposed by Zhang and Zhang allows for unlimited federation between computing regions. It also enables users to switch service providers easily thereby avoiding vendor lock-in.

2.4 Peer to Peer Cloud Computing

Peer to Peer (P2P) cloud allows companies or individuals to build a computing infrastructure by pooling together existing resources [13]. While a P2P cloud is not able to guarantee Quality of Service (QoS), it is still a useful distributed architecture which can turn idle resources into computing infrastructure. Figure 2.5 describes a P2P cloud model by Babaoglu et al. The areas shaded in gray have been fully implemented.

Ranjan and Buyya have combined P2P model with Grid computing to propose a new model for distributed resource management called Grid-Federation [1]. In this model, users interact with individual grid-clusters, which combine together to pool

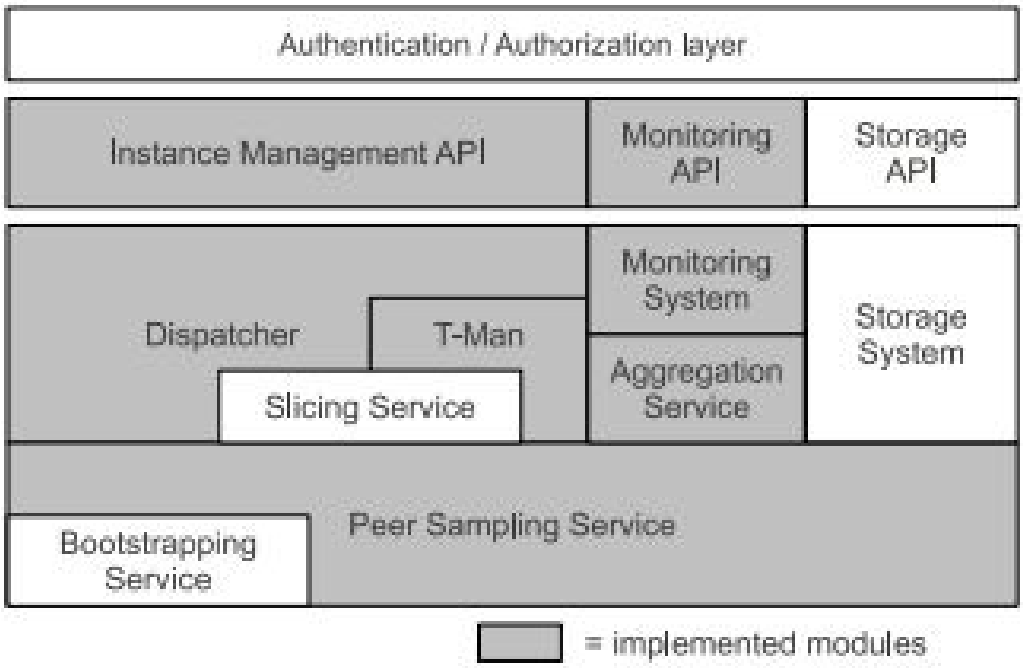

Figure 2.5: Peer to Peer cloud federation architecture as proposed by Babaoglu et al. [13].

resources. A Grid Federation Agent (GFA) acts as a resource coordinator in the federated space, spanning over all clusters. Each cluster must instantiate a component of the GFA in order to join the cloud federation. Ranjan and Buyya use experimental and statistical analysis to show that their proposed framework leads to better overall utilization of cluster resources and improves the realization of QoS constraints imposed by resource consumers.

2.5 Broker-Based Cross-Cloud Federation Manager

Abdo et al. propose a broker-based federation architecture in order to allow different cloud providers to dynamically cooperate in order to gain economies of scale, make efficient use of assets, and become more competitive [17]. They propose an improve-

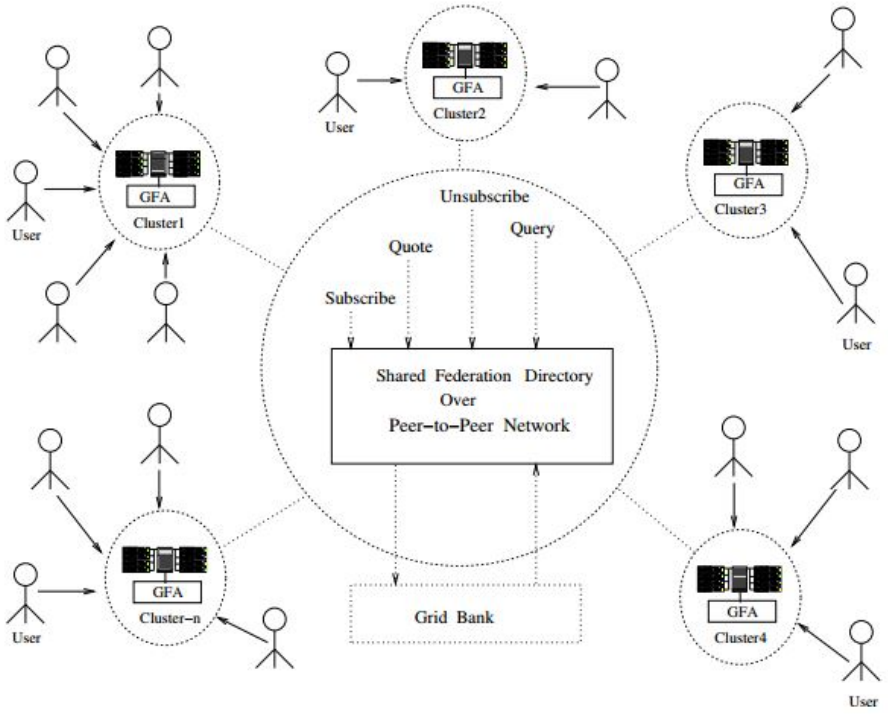

Figure 2.6: Ranjan and Buyya's Grid-Federation framework [1].

ment over the Three-Phase Cross-Cloud Federation Model (CCFM) as proposed by Celesti et al [9, 8]. In this model, a centralized server acts as a "broker", which contains updated information about each present cloud provider. As illustrated in Figure 2.7, the broker interacts with each cloud that wants to join the federation rather than the entrant cloud exchanging information with every other cloud in the federation.

Under the broker-based CCFM business model, the broker can intelligently negotiate with service providers wanting to dynamically join the federation on price points and demand. As illustrated in Figure 2.8, a federation request through three phases: discovery, match-making, and authentication. The Discovery Agent looks for any available clouds. The Match-Making Agent then picks the cloud that offers the best deal. The Authentication Agent establishes a trust context between the matched clouds.

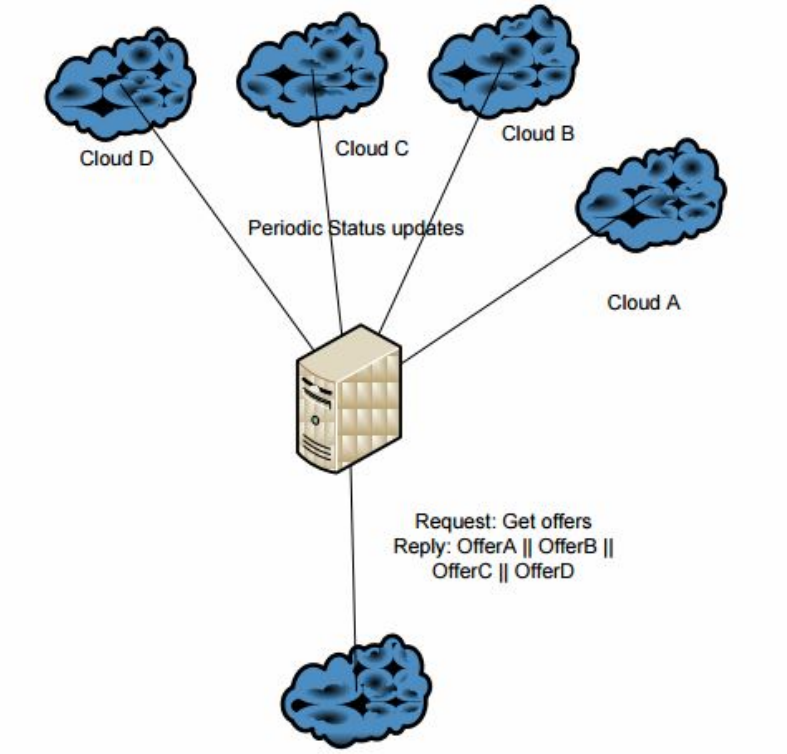

Figure 2.7: Broker-Based Cross-Cloud Federation [17].

2.6 Federation: The Next Phase of Cloud Computing

It is apparent that there are many different ways by which federation between clouds can be achieved. Many of these federation models are necessitated by the nature of the underlying system, such as P2P clouds in Section 2.4. Upon closer examination, the trend seems clear: combining or federating cloud resources by different service providers, organizations, and even individuals can lead to a more efficient, versatile, and user-friendly cloud economy. As argued by Celesti et al. [9, 8], and Abdo et al. [17], we are approaching stage 3 of the cloud computing market's ecosystem: Horizontal Federation, where providers will dynamically cooperate to gain economies of scale, make more efficient use of their assets, cater to a diverse global market, and

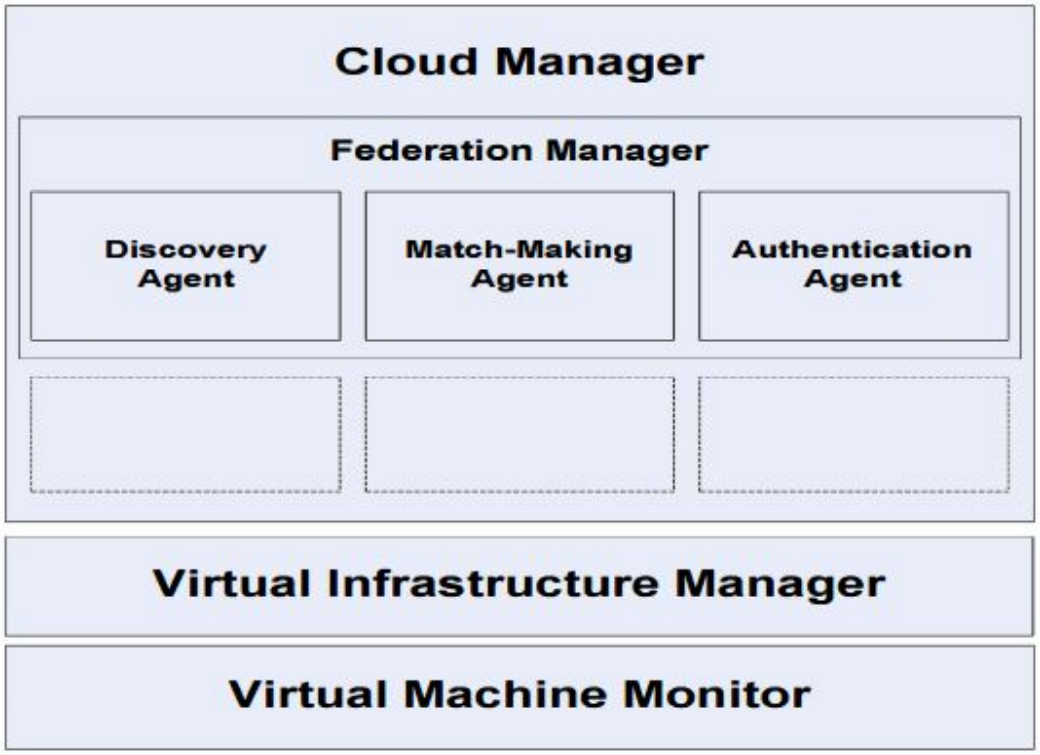

Figure 2.8: Broker-Based Cross-Cloud Federation [17]

increase their capabilities and availability.

2.7 Summary

In this chapter, we have detailed different areas and projects involving Cloud Computing with a focus on federation. All of the approaches that we have described involve customizing or specializing the cloud software for federation. In contrast, our approach achieves federation between two cloud systems by overlaying services and tools without modifying the underlying architecture of the systems being federated. We describe our proposed architecture in Chapter 3.

Chapter 3

GENI-SAVI Federation Architecture

This chapter describes the approach we took when federating GENI and SAVI cloud systems. We will first discuss the overall architecture of GENI-SAVI cloud federation. In the later sections, we will dive deeply into the authentication approach we took in order to allow users access to the federated system.

3.1 Overall Architectural Goal

GENI and SAVI are disparate cloud systems built on different underlying systems [21, 22]. In addition, these systems have several other differences such as their own APIs (Application Programming Interface) which allow clients to access these systems. We outline these differences in detail in sections 3.4 and 3.5. Our goal was to allow a user to be able to access resources on both the cloud systems without changing their existing architecture significantly. Figure 3.1 shows a very broad and simplified version of our architectural goal. In this architecture, a user should simply be able to send commands to a common interface in order to request or release resources

without worrying about system intricacies underneath. We dubbed this interface as SAGEFed (SAGEFed is a combination of words SAVI GENI Federation) [34]. The SAGEFed interface then translates the user commands into commands that GENI or SAVI APIs can understand, and also translates the results back to the user.

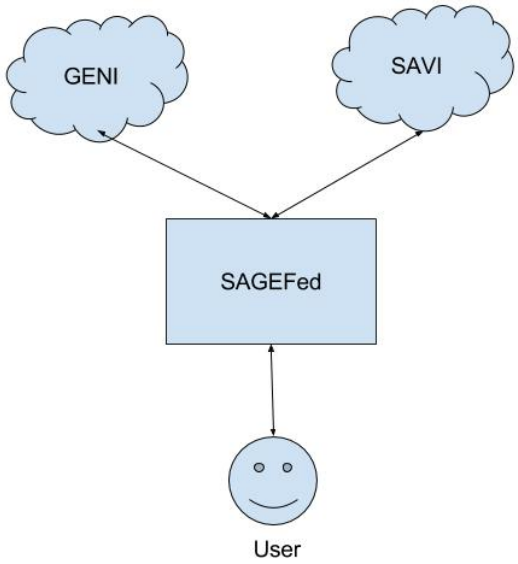

Figure 3.1: Architecture Goal - Overview

3.2 An example

Let us use a concrete example. Let us assume that our user has access to the resources on both GENI and SAVI systems. Now imagine a scenario where the user is using GENI client tool to create some virtual machines, but partway through, GENI runs out of resources or the user wants to host part of their service in Canada, where there are no GENI servers. In such cases the user would like to create the rest of the virtual machines on SAVI, which has servers in Canada. The user must now launch SAVI client tool separately, and then reserve those virtual machines on SAVI. This requires the user to deal with an entirely different set of client commands and makes their job

more tedious.

Our solution solves exactly this problem of managing resources on these two disparate systems through the SAGEFed interface as described in Section 3.1. In order to be able to use the SAGEFed interface, the user needs to be authenticated on both GENI and SAVI systems. This thesis, and by extension this chapter focus on the authentication architecture of the GENI-SAVI federation. We also cover the SAGEFed interface architecture in some detail.

3.3 OpenStack Architecture

Before we discuss the GENI and SAVI systems, it is important to discuss OpenStack [54], which underlies SAVI. Certain implementations of GENI such as ExoGeni also rely on OpenStack components [38]. OpenStack is a cloud operating system which allows control over underlying hardware resources in a neatly packaged manner in form of services and APIs [23]. The hardware resources may be distributed geographically such as in the case of SAVI. The OpenStack conceptual architecture is described in Figure 3.2.

Some of the OpenStack components/services are described below:

- ***Compute (Nova)***: Handles creating, scheduling, and terminating virtual machines (VM) on demand [23]. Nova is the OpenStack API which allows clients the ability to use Compute service in such a manner.

- ***Block Storage (Cinder)***: Provides persistent block storage to running VM instances. The block storage behaves like a file storage system or a hard drive for the VM instance in question [24].

- ***Object Storage (Swift)***: Swift API provides object storage as a service.

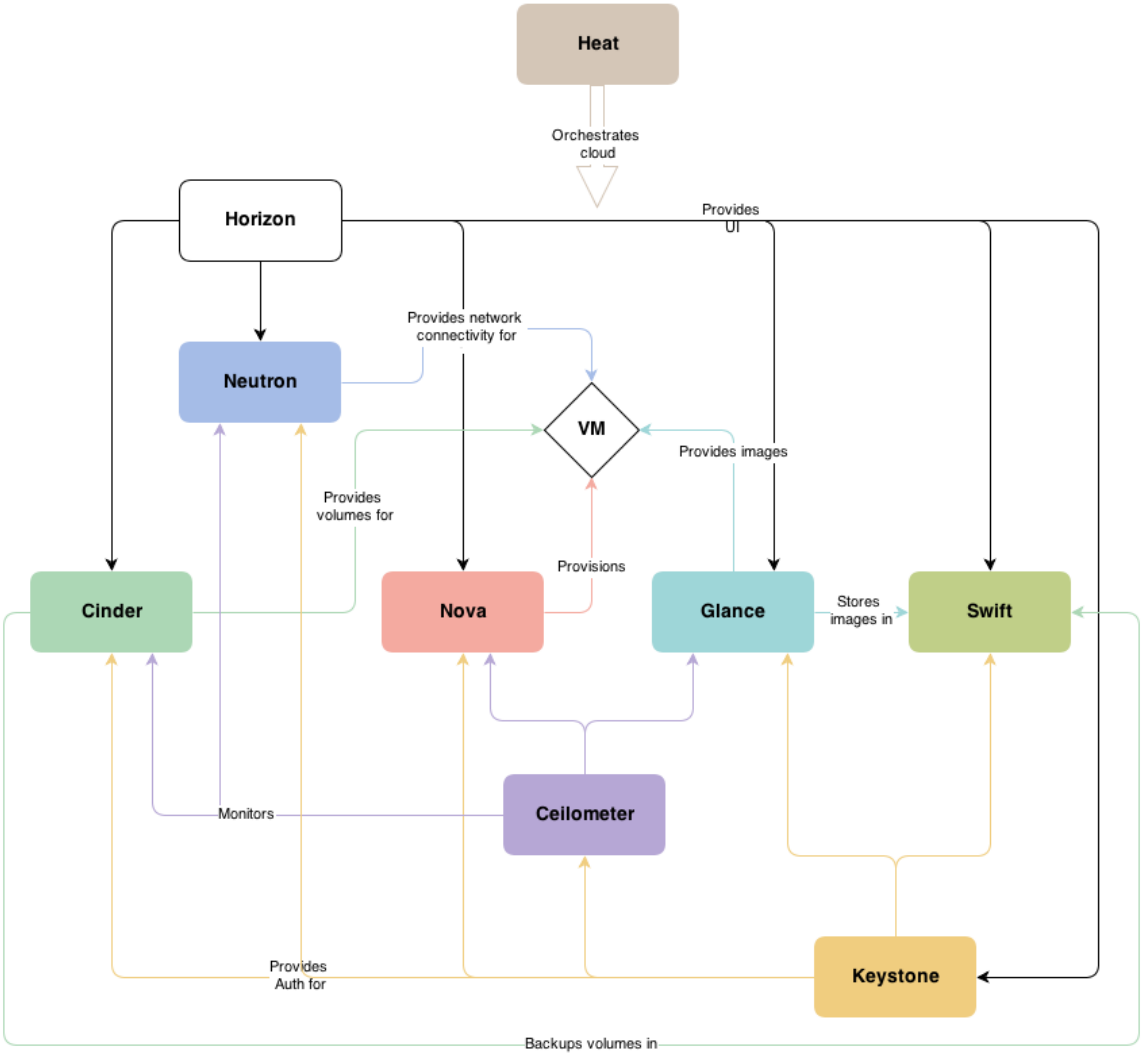

Figure 3.2: OpenStack Conceptual Architecture (Havana Version) [23]

Object storage is a unit of data which is treated like an object with its own metadata and unique identifier in the cloud [24]. OpenStack provides object storage that is scalable, and redundant.

- ***Identity Service (KeyStone)***: KeyStone provides Identity related service such as authenticating, authorizing, and storing users and groups for use across an OpenStack implementation [23].

Above are just some of the OpenStack components and services that were relevant

to our research. The OpenStack documentation contains the full description of all the OpenStack components [23].

3.4 GENI

GENI (Global Environment for Network Innovations) Project is a federation of academic sites across the United States that provide cloud based resources and are implemented using one of the custom GENI hardware and software systems, also known as GENI racks [45]. GENI racks must meet certain GENI hardware and software specifications [40] in order to be part of the GENI cloud. The GENI Project lets a user access computing and network resources across many sites in the United States. It is the system in question that we federated with SAVI.

3.4.1 Key GENI Concepts

Before we proceed, we need a basic understanding of some of the key concepts and terminology with regards to GENI resources [41].

- ***Experimenter***: An experimenter is a GENI user with credentials to access certain GENI resources.
- ***Experiment***: An experiment is a general way to describe any research activity performed on GENI network that requires certain resources by certain users.
- ***Project***: A GENI project encompasses users, experiments, and the resources involved. A project is created and led by the Project Lead. An experimenter can belong to different projects at the same time.
- ***GENI Aggregate***: A GENI Aggregate is a GENI rack hosted on a physical site (for example, a university). A user can request resources from a GENI

Aggregate if they have permissions to do so.

- ***Slice***: A slice is a unit of isolation for experiments. A slice can can be granted access to resources across several aggregates. For example, if an experiment needed 5 virtual machines across 5 physical sites where the experimenters wanted to measure network flow between the virtual machines, they would create a slice and reserve access to the resources across those 5 aggregates. Figure 3.3 illustrates the concept of a slice.

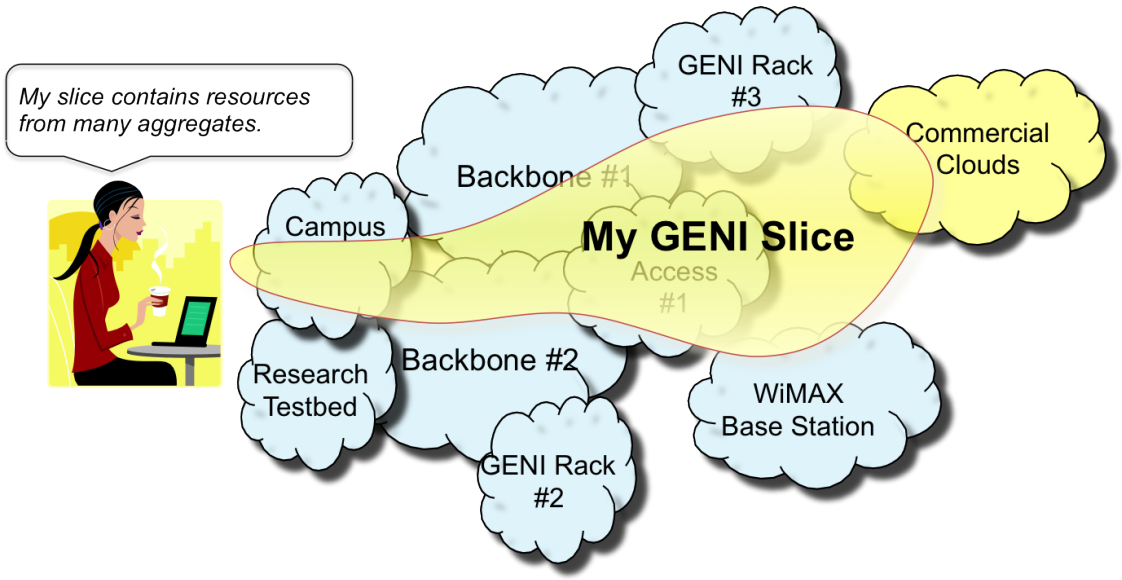

Figure 3.3: A GENI Slice [41]

- ***Aggregate Manager API (AM API)***: GRAM is an implementation of AM API which allows an experimenter to list, request, find status of, and delete resources available at an aggregate. Every aggregate must publish its resource information as per the GENI AM (Aggregate Manager) API standards. The way it accomplishes this is by using an Advertisement RSpec document to describe its resources.

- ***RSpec***: An RSpec (Resource Specification) document is a resource specification

document in an XML format. An experimenter generates an RSpec document describing his/her resource needs. An Advertisement RSpec is a document which describes resources at an Aggregate. An RSpec manifest is a document that describes the resources that a user has reserved at an Aggregate Manager.

- ***Experimenter Tools***: GENI provides a set of tools that an experimenter can use to request resources from aggregates and run experiments. These tools automatically generate RSpec documents and make appropriate GENI AM API calls so that the experimenter doesn't have to be intimately aware of the details of the RSpec and GENI AM API.

- ***Sliver***: A Sliver is one or more resources provided by an aggregate in form of virtual or physical resources. Sliver represents a unit of resource where experiments can be performed. When a user reserves a Sliver, they reserve resources on an Aggregate. Figure 3.4 visualizes the relationship between a Sliver and the rest of the components mentioned above.

- ***Omni Client***: Omni Client is a command line tool that offers a user more flexibility in accessing GENI resources via their command shell or a terminal [33].

- ***Federation***: Federation permits the interconnection of independent infrastructures in a way that allows their owners to declare usage policies and resource allocation. From a GENI user perspective, federation allows a user to gain access to GENI resources across multiple sites by using one set of credentials [41].

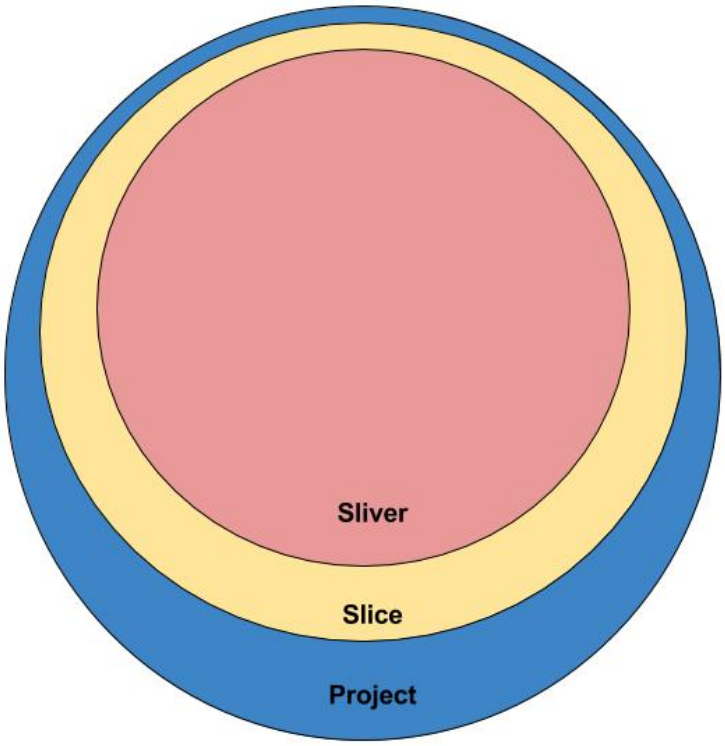

Figure 3.4: Relationship between a Sliver, a Slice, and a Project

3.4.2 GENI Federation of Testbeds

GENI (Global Environment for Network Innovations) is a federation of testbeds which provides a computing environment for networking and distributed systems research and is supported by the National Science Foundation [39, 40]. These testbed sites are maintained by their respective institutions, many of which are academic. Each GENI site contributes hardware and network resources in form of GENI Racks (see Section A.1 on GENI Racks). GENI allows users to obtain computing resources from these sites, connect those compute resources, install custom software and operating systems on these resources, among many other things. This allows users to perform experiments ranging from small-scale computations on a single machine, to large-scale experiments involving several machines over virtualized network topologies.

3.5 SAVI

SAVI (Smart Applications on Virtual Infrastructure) is a multi-tier cloud consisting of virtualized resources focused on computing and networking [21]. SAVI's goal is to provide these network and computing resources for experimental research. SAVI is built around the concept of SDI (Software Defined Infrastructure) [21]. An SDI provides open programmable interfaces to control and manage various types of resources in different types of infrastructures. As such, SAVI's major selling point is that it is a programmable infrastructure which allows its users to modify its underlying network and computing resources as they see fit. For example, a researcher can reprogram SAVI's underlying routers in order to do routing experiments. Figure 3.5 provides an illustration of how SDI can be used to manage virtual and physical resources.

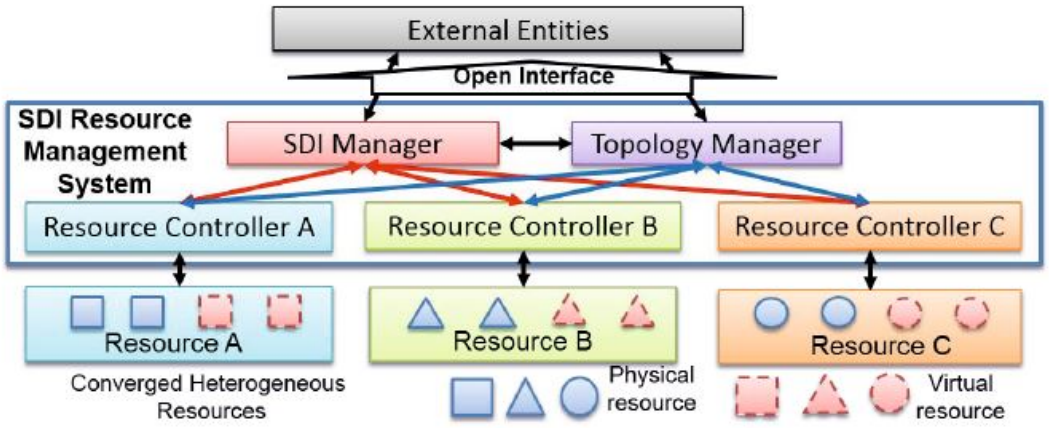

Figure 3.5: SDI resource management architecture [21]

3.5.1 SAVI Architecture

SAVI Testbed is comprised of several different types of physical entities which we describe below. Figure 3.6 illustrates these entities hosting two sample applications.

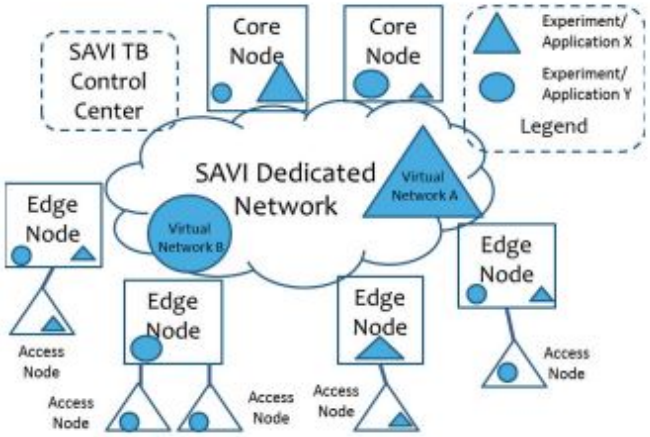

Figure 3.6: SAVI entities hosting two sample applications [18]

SAVI Physical Entities

SAVI consists of five main physical entities:

1. ***Core Nodes***: Core nodes provide conventional cloud computing resources such as compute, storage, and basic networking [18]. These Core nodes reside on the SAVI backbone infrastructure at University of Toronto.

2. ***Smart Edge Nodes***: Edge nodes provide more advanced resources such as reconfigurable hardware resources. Edge nodes reside at the sites which are a part of the SAVI network, such as University of Victoria.

3. ***Access Nodes***: Access nodes are similar to Smart Edge Nodes, but they are more focused on virtualized resources. The Access nodes are connected to Smart Edge nodes via dedicated links.

4. ***SAVI Network***: SAVI network is a dedicated research network which connects Core nodes, Smart Edge nodes, and the SAVI Testbed Control Center. SAVI network is also itself considered a resource.

5. ***SAVI Testbed Control Center***: A software suite that allows monitoring, measurement, and allocation of resources and also acts as a web portal server.

SAVI Node

As mentioned in Section 3.3, SAVI is built on top of OpenStack and leverages many of its components in order to manage the SAVI resources [21]. For example, it uses Keystone for Identity management. Each SAVI node controls and manages virtual and physical resources using some of these OpenStack components which are customized for SAVI purposes. A control and management system for a Smart Edge node is illustrated in Figure 3.7. In this system, OpenStack components along with the SDI manager provide access to the underlying physical and virtual resources.

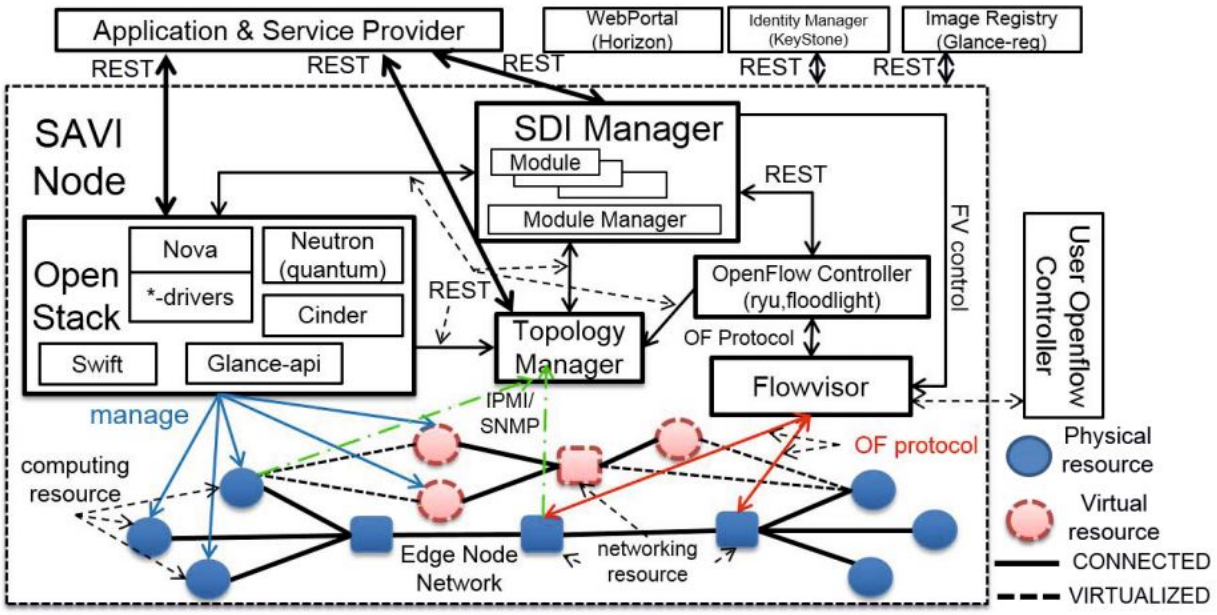

Figure 3.7: Control and management system for a SAVI Node [21]

Current SAVI Deployment

SAVI is currently deployed across multiple University sites in Canada. SAVI testbed has 550+ CPU cores, and 50+ terabyte storage among other network and compute resources [21]. Figure 3.8 gives an overview of the SAVI network deployment. The SAVI testbed control center is located in University of Toronto and acts as a resource manager across the entire infrastructure. ORION (Ontario Research and Innovation Optical Network) is a 1GE L2 ethernet link which connects all the SAVI nodes in Ontario.

Nova Client

Nova Client [55] is a command line tool which allows a user to access resources on an OpenStack system which also works with SAVI. Nova Client is used as a flexible and powerful alternative to the SAVI Portal's web interface when accessing SAVI resources.

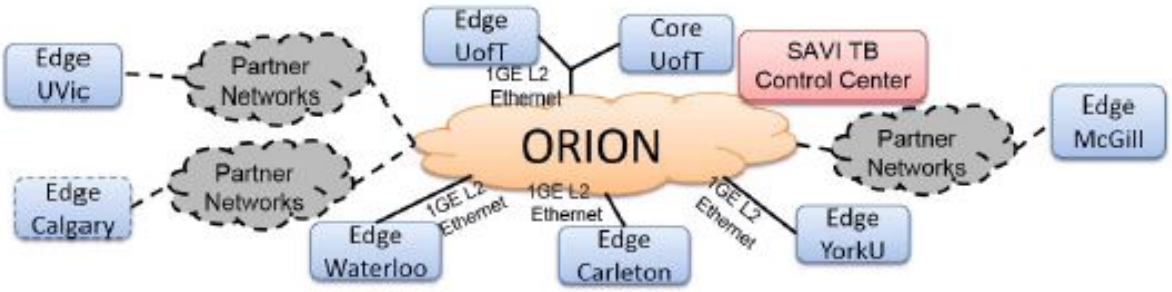

Figure 3.8: SAVI entities hosting two sample applications [21]

3.6 GENI-SAVI Federation Architecture

In this section, we present the GENI-SAVI Federation Architecture. When considering federating these disparate systems, we divided our approach into two major pieces:

1. Authenticating the user so that they have the permissions to access the federated resources.

2. Presenting a unified interface to the user via which they can access the federated resources in a seamless manner.

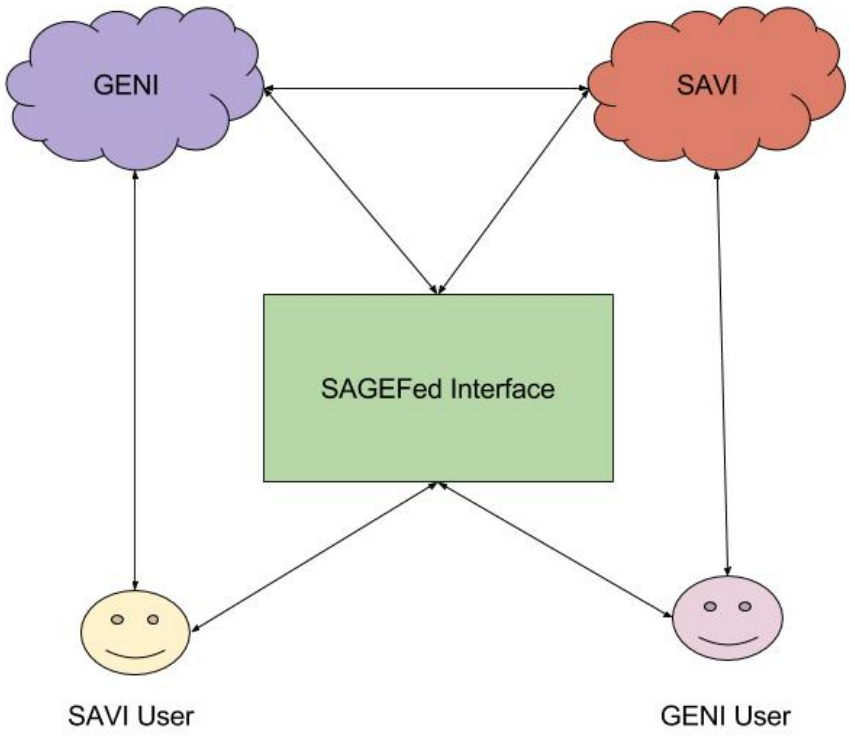

Figure 3.9: SAVI and GENI users being authenticated and accessing the federated system using the SAGEFed interface.

Figure 3.9 illustrates a very simplified overview of the complete architecture. In this figure, a GENI user is someone who already has access to GENI, and a SAVI

user is someone who has access to SAVI. We describe the process where the GENI user is able to access SAVI and a SAVI user is able to access GENI below:

1. A GENI user authenticates themselves onto SAVI when SAVI verifies with GENI that the user indeed is a GENI user.

2. A SAVI user authenticates themselves onto GENI when GENI verifies with SAVI that the user indeed is a SAVI user.

3. Both the users may now use the SAGEFed interface tool to access resources on either SAVI or GENI.

We now dig into this architecture further and look at the design and process flows involved in this approach.

3.6.1 User Authentication

In order for a user to access resources on either GENI or SAVI, they must first be authenticated via user credentials. In addition to having user credentials, the user must be authorized to access the resources in question, such as the ability to create a virtual machine. This is done by allowing the user to join projects or tenant groups with access permissions in place as described in Sections A.2 and A.3. In order for a SAVI user to access resources on GENI or vice-versa, we would not only need to authenticate the user on that particular system, but we would also need to have them join groups with certain access permissions in order to allow them access to certain resources.

As mentioned in Section 3.5, SAVI is based on OpenStack. GENI on the other hand, uses various implementations of the AM API as mentioned in Section 3.4. Consequentially, we needed two separate mechanisms in place in order to authenticate users:

1. Authenticating SAVI users onto GENI.
2. Authenticating GENI users onto SAVI.

Authenticating SAVI users onto GENI

SAVI stores its users in a centralized location in its Keystone database, which is a part of its underlying OpenStack implementation as mentioned in Section 3.3. Keystone is an OpenStack identity service [36] which manages its user database and also acts as an API to support multiple authentication mechanisms. The following is an outline of steps we determined should be followed by a SAVI user needing access to GENI (see figure 3.10):

1. Visit the GENI web portal.
2. Get redirected by GENI portal to a SAVI authentication site with a request to authenticate the user.
3. Get authenticated by SAVI authentication site as a SAVI user. SAVI authentication site would check the user credentials against the Keystone database.
4. Get redirected back to GENI portal along with an authentication response
5. GENI portal verifies the response and grants the SAVI user access

As mentioned in Section 3.4, GENI is a federation of different sites and uses the InCommon federation approach in order to authenticate users [42]. In this regard, SAVI is not much different from a typical GENI organization when it comes to granting SAVI users access to GENI resources. On a side note, GENI is also able to store and authenticate users directly using the GENI Clearing House for the users who do not have access to a site which can authenticate them.

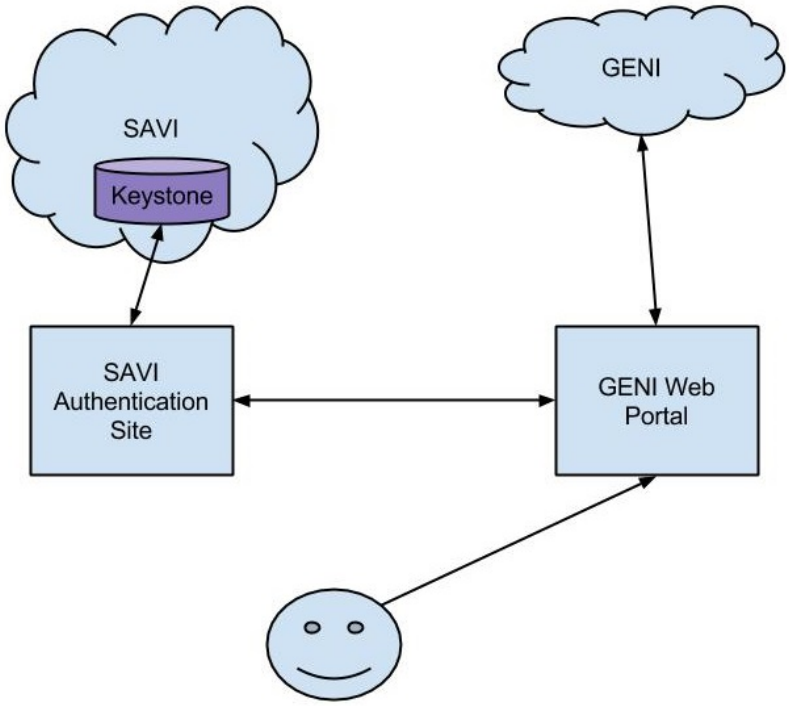

Figure 3.10: A SAVI user being authenticated by SAVI onto GENI

This was just a high level view of the GENI-SAVI federation architecture that we initially designed. We had to make some modifications to this design, which we will further elaborate in Chapter 4.

Authenticating GENI users onto SAVI

As opposed to GENI, SAVI has a centralized Keystone database which stores all its users and their credentials. As such, SAVI didn't originally have a way of authenticating external users. We created a service hosted on SAVI network that would authenticate users coming from GENI with a certificate. We describe this in greater detail in Chapter 4. The following is an outline of steps we determined should be followed by a GENI user needing access to SAVI (see Figure 3.11):

1. Login to the GENI web portal via a GENI organization/site.

2. Get forwarded to SAVI authentication service along with a user certificate that the service can use to verify the user.

3. Login to SAVI.

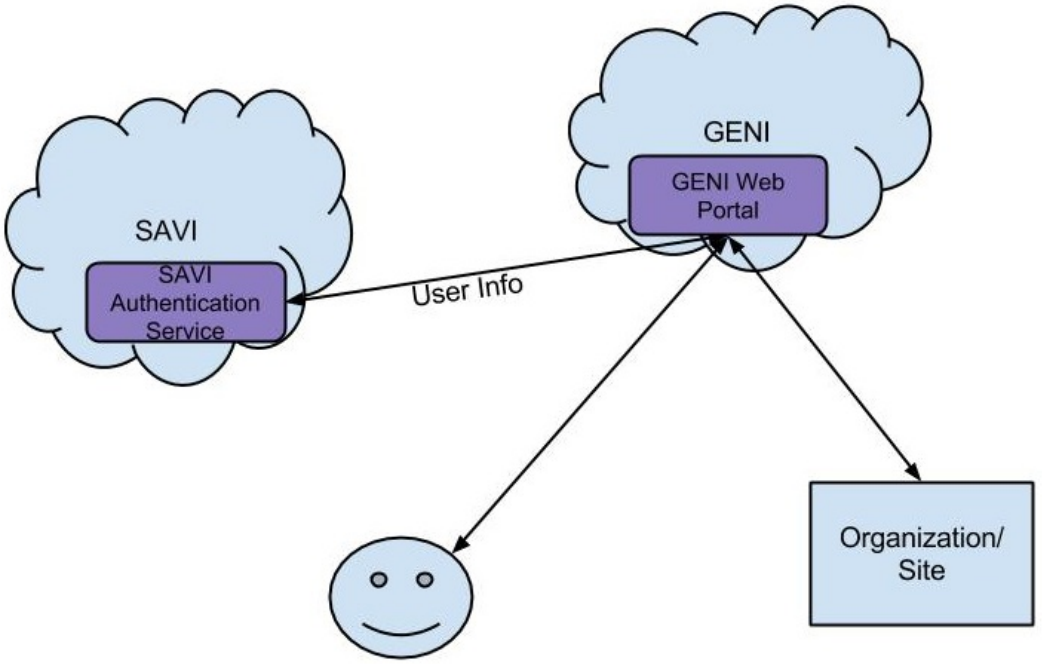

Figure 3.11: A GENI user being authenticated onto SAVI

This was a high level view of the process involved in authenticating a GENI user onto SAVI. There were some modifications to this design when implementing it, which we describe in Chapter 4.

3.6.2 SAGEFed Interface

One of the goals of the GENI-SAVI Federation project was to make the integration between the two systems as transparent to the user as possible. In addition, we had to make sure that the user interface was going to be extensible. As mentioned in

Sections 3.4 and 3.5, both GENI and SAVI have web interfaces as well as command line tools that a user can leverage to access the resources on these systems. We considered creating a common web interface to access the federation resources, but it seemed that a command line interface would be much more extensible and functional, and it would let us leverage the existing Nova and Omni client tools. One of the major benefits of using the command line is the ability to script several commands together. For example, if an experiment calls for repeatedly creating several virtual machines on SAVI as well as GENI nodes, running some tests, and then deleting those virtual machines, it is much more convenient to do so by creating a script that executes SAGEFed commands and then running it as many times as we would like. In addition, the command line interface can be quickly modified or extended to integrate any other systems in the future.

Figure 3.12 illustrates how a user can access both SAVI and GENI resources using the SAGEFed. In order to use the SAGEFed, the user must first have access to either GENI and/or SAVI and have the corresponding Omni and Nova client tools setup on their machine along with their credentials as mentioned in Sections 3.4 and 3.5.

1. User launches SAGEFed and issues a command.

2. SAGEFed first validates the command to ensure that it is valid.

3. Next, SAGEFed translates the command according to which system it is aimed.

4. SAGEFed then sends the translated command along with user credentials to either Omni or Nova client tools.

5. Omni and Nova client tools then execute the command onto their respective GENI or SAVI systems and get back the response.

6. Finally, SAGEFed translates the response and relays it to the user in an appropriate format.

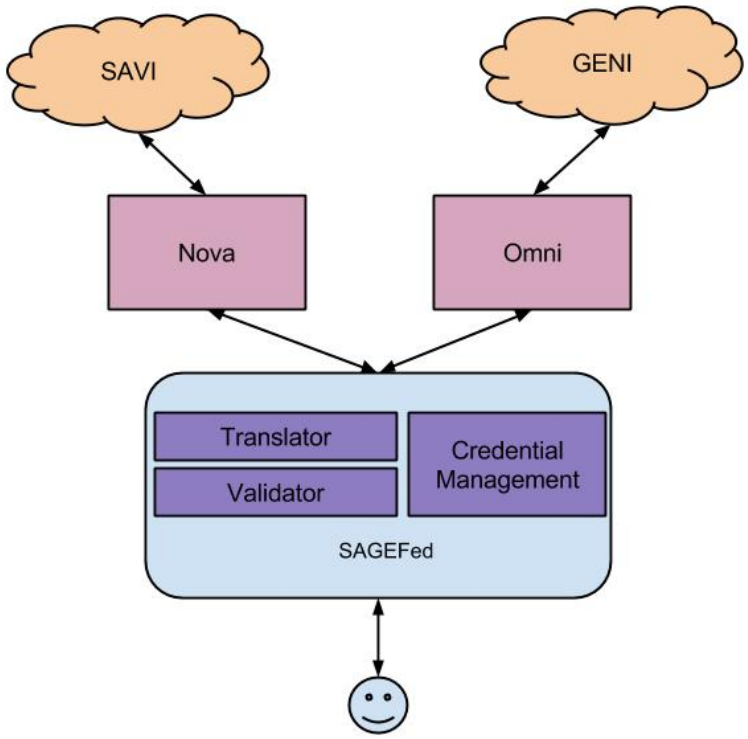

Figure 3.12: A user accessing both GENI and SAVI resources using SAGEFed

This concludes our high level architecture of the GENI-SAVI federation. We look at the implementation details including trade-offs in the next chapter.

3.7 Summary

In this chapter, we discussed the overall architecture for GENI-SAVI federation and the technologies and concepts underlying GENI and SAVI. In Chapter 4, we go into implementation details and trade-offs we made when implementing the architecture we had originally conceived.

Chapter 4

Implementation

4.1 User Authentication

When authenticating a user on either GENI or SAVI, we needed to consider user account creation and subsequent access to the system. One of our design goals was to minimize any modifications to the underlying system.

4.1.1 Third Party Authentication

One of the main goals behind federating GENI and SAVI was so that a user wouldn't have to create separate accounts on both systems. We chose to use third party authentication very early on as one of the design components. Third party authentication involves two major components: an Identity Provider (IdP) and a Service Provider (SP). An SP controls access to resources that a user is trying to access, and defers the user to an Identity Provider (IdP) that has been setup to work with the SP. An IdP verifies a user against an underlying user credential database by prompting the user for their credentials. Please note that an SP can accept verification from more than one IdP. We now briefly describe how third party authentication works at a high

level. Please see Figure 4.1 for reference.

1. A user tries to access certain services and is greeted by an SP and is redirected to an IdP which can recognize and verify the user.

2. After successfully verifying the user, the IdP redirects the user back to the SP along with a verification token and user information.

3. SP then verifies the token and grants the user access to its underlying services.

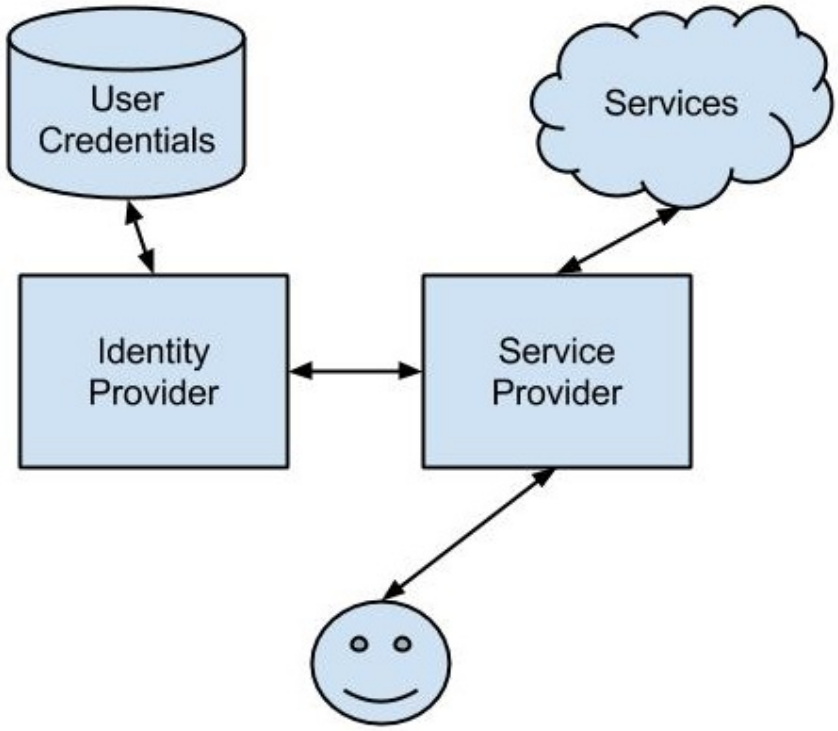

Figure 4.1: A user being authenticated using third party authentication

4.1.2 SAVI Keystone for Third Party Authentication

Keystone can be configured to act as an Identity Provider as well as a Service Provider. We considered using SAVI Keystone for third party authentication, but we decided against it after some investigation as described below.

Keystone as an Identity Provider

OpenStack Keystone can be configured to be used as an Identity Provider (IdP) [56] as of the OpenStack Kilo release. It can be setup to run inside a web container such as Apache and respond to verification requests [28]. However, at the time we implemented the project, OpenStack Kilo had not yet been released and Keystone support for IdP was unstable. Therefore we decided not to use this option, but to setup a separate service backed by a database to act as an IdP. We describe this architecture in section 4.1.5.

Keystone as a Service Provider

OpenStack Keystone can be used as a Service Provider (SP) [56]. Keystone provides options to be setup for third party authentication to use either of Shibboleth, OpenID Connect implementations [51, 68]. GENI portal can act as an OpenID 2.0 IdP [43]. However, OpenID is now obsolete and has been superceded by OpenID Connect [52]. We considered setting up Keystone as a Shibboleth SP, however it proved challenging to configure SAVI Keystone as an SP, and we decided to go with a more flexible architecture by setting up a service in the middle that acts as an SP and interfaces with SAVI Keystone. We describe this architecture in section 4.1.5.

4.1.3 Authenticating SAVI Users on GENI

Authenticating SAVI users onto GENI requires us to first allow SAVI users to be able to log on to the GENI web portal. Once the user has successfully logged onto the GENI web portal, they can be treated as a regular GENI web portal user and access GENI resources as described in Section 3.4.

4.1.4 GENI as an SP

In order for GENI to grant a SAVI user GENI resources, it has to act as a Service Provider (SP) which accepts authentication verification from an Identity Provider (IdP), namely SAVI. Luckily for us, GENI is already a third party authenticated federation and GENI Portal already acts as an SP [42] for participating sites which in turn act as Identity Providers. SAVI can just act as one of these already existing Identity Providers in order to federate with GENI Portal.

4.1.5 SAVI as an IdP

According to GENI Portal specifications, an IdP must be implemented using Shibboleth because GENI Portal SP is implemented using Shibboleth. We used OpenLDAP as a user database against which Shibboleth IdP would authenticate the user. OpenLDAP is an implementation of LDAP, which is a protocol for accessing and maintaining distributed user directories and user information such as user credentials [61, 53]. The OpenLDAP database first needs to be populated with users from SAVI's Keystone user database. Figure 4.2 illustrates how a SAVI user can get authenticated onto GENI Portal.

We now describe the setup process for Shibboleth IdP and OpenLDAP.

Hosting SAVI IdP

Before configuring Shibboleth and OpenLDAP, we first had to decide how and where to host these services. SAVI IdP is hosted on a SAVI core server located on University of Toronto campus. The server is able to handle more than 1000 concurrent requests, but we don't expect more than 100 concurrent users to request this service in the foreseeable future.

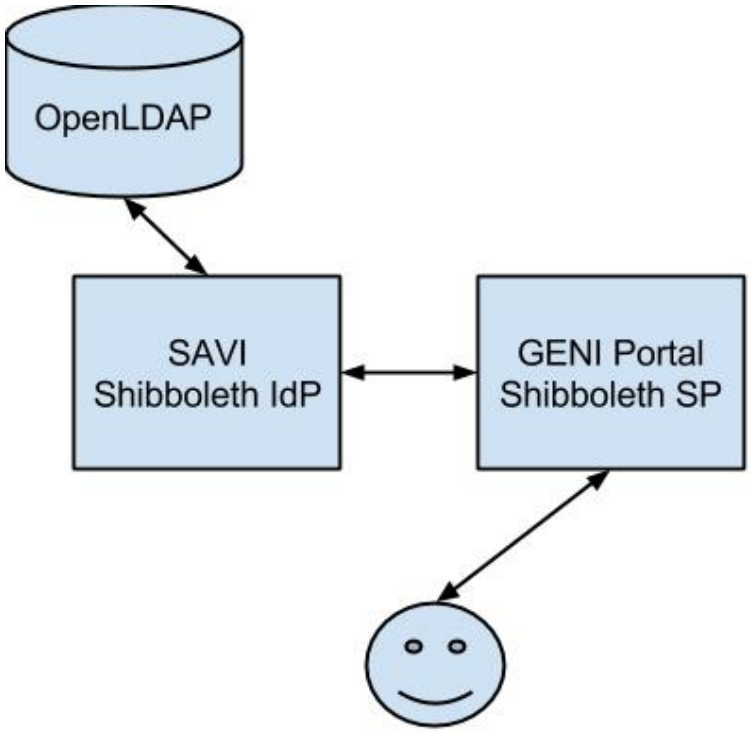

Figure 4.2: GENI authenticating a user by redirecting them to SAVI IdP

The hosting server also needs a routable IP address, a domain name, and an SSL certificate to encrypt and secure the connections to the server [72, 46].

Setting up an OpenLDAP User Database

Shibboleth IdP service needs to be able to query the OpenLDAP database to authenticate users. Slapd [53] is an OpenLDAP service and a database wrapped into one package. It manages connections and queries to the database. The database itself is an implementation of the BerkelyDB. Slapd is the actual service which Shibboleth IdP interfaces with. In addition to Slapd, we need Ldap-utils so that we can configure and query the OpenLDAP database. Once we have setup OpenLDAP, we can query it, or add users as shown in Listings 4.1 and 4.2.

Listing 4.1: Adding a user to LDAP

```
ldapadd -x -D "cn=admin,dc=ldap,dc=yourdomain,dc=com" -w
    supersecretpassword
dn: cn=SOME_USERNAME,dc=ldap,dc=yourdomain,dc=com
objectClass: inetOrgPerson
cn: SOME_USERNAME
uid: SOME_USERNAME
UserPassword: SOME_PASSWORD
sn: SOME_SURNAME
```

Listing 4.2: Querying a user on LDAP

```
ldapsearch -LLL -x -D 'cn=admin,dc=ldap,dc=yourdomain,dc=com' -b
    dc=ldap,dc=yourdomain,dc=com -w supersecretpassword
```

Setting up Shibboleth IdP

Shibboleth IdP is built using Java, and as such must be hosted in a web container such as Tomcat [60, 29, 27]. We opted to use the latest stable release of Shibboleth IdP at the time, which was Shibboleth 2.4.2. The Java release version was Java 7, and Tomcat version was Tomcat 7. Finally, we used Apache 2, as the web server to host the Shibboleth service [28].

After installation, we had to follow the steps below in order to setup the Shibboleth IdP to work properly with GENI Portal SP:

1. Setup a domain name entry which can route to the server's IP address [72].

2. Purchase and setup an SSL certificate with Apache web server in order to encrypt the authentication [46].

3. Setup and map appropriate Tomcat ports to Apache ports in order to intercept incoming requests.

4. Setup Tomcat to interface with and host Shibboleth Service

5. Setup Shibboleth configuration files to interface with the OpenLDAP database service that has already been setup on the server.

6. Retrieve SP metadata from GENI Portal Shibboleth and configure our Shibboleth IdP to process incoming requests from the GENI Portal SP [42].

Figure 4.3 illustrates an authentication request making its way through our Apache/-Tomcat/Shibboleth/OpenLDAP setup:

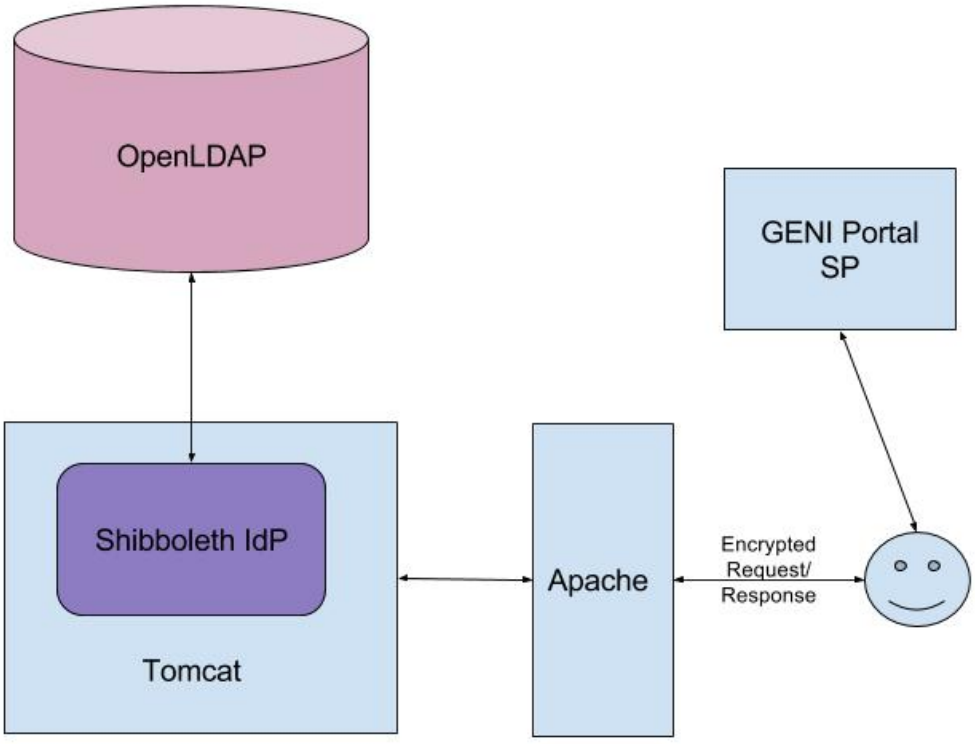

Figure 4.3: An authentication request being processed by Shibboleth IdP.

1. A user is redirected by the GENI Portal SP (Service provider) to SAVI Shibboleth IdP (Identity Provider), where the user enters their SAVI credentials. The

user credentials, along with the GENI Portal SP metadata, form an encrypted request which is first received by Apache Web Server.

2. Apache redirects the request to Tomcat.

3. Tomcat allows Shibboleth IdP application to process the request.

4. Shibboleth IdP verifies the user credentials against OpenLDAP and crafts a response for GENI Portal SP verifying the user. The user is redirected to GENI Portal SP along with this response.

5. GENI Portal SP verifies the response and authorizes the user.

Transferring Data from SAVI Keystone to OpenLDAP Database

After verifying that our current Shibboleth and OpenLDAP setup works properly, the next step was to transfer the relevant user data from SAVI Keystone to our OpenLDAP database. We generated an SQL script which contained existing SAVI users by exporting the relevant user data such as user login id, first name, last name, and encrypted password. The user encrypted password is an implementation of Python SHA512 password scheme [65]. The script creates an LDIF file (LDAP understandable format) with all the necessary user data, and then executes the LDAP command to make the necessary user data modifications to the OpenLDAP database.

Once OpenLDAP database has been initialized with existing SAVI user data, it must be kept synchronized with the SAVI Keystone database. We present the process below accompanied by a flowchart diagram in Figure 4.4. Please note that only an administrator or someone with sufficient privileges is able to setup this process.

1. A request to add/modify/delete a user comes in to SAVI user management service.

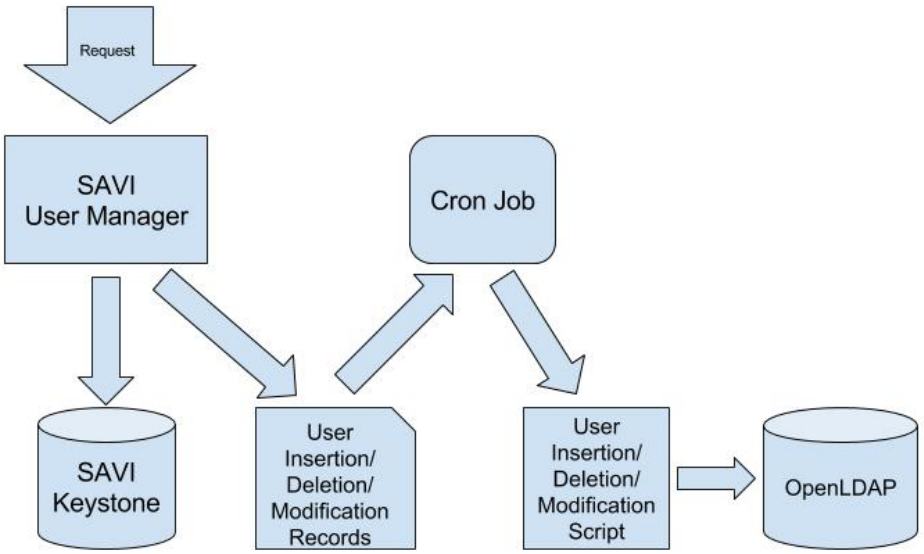

Figure 4.4: A SAVI user modification request flowchart

2. The request is forwarded to the SAVI Keystone, where the request is fulfilled. It is also recorded in a user record file.

3. A Cron job, which is a scheduled task [71], runs on its scheduled time and it reads the user record file which consists the user records to be added/modified/deleted.

4. The Cron job then executes the LDAP script which modifies the OpenLDAP database by adding/modifying/deleting users.

After importing the user data into our OpenLDAP database, we had to configure OpenLDAP to use the same password encryption scheme to ensure that the users would be able to log in using their old credentials. When the user types their password when logging in, the password is encrypted using the same scheme and then the resulting encrypted value is matched against the existing encrypted password value in the database.

4.1.6 Creating a SAVI User Account for a GENI User

Unlike GENI, SAVI is not configured as a Service Provider(SP) which can leverage third party authentication. SAVI user credentials are stored centrally in an OpenStack Keystone database. For this project, we decided to automatically create SAVI accounts for GENI users when they tried to access SAVI, rather than reconfigure SAVI as an SP. We now describe this process below. Please see the accompanying Figure 4.5.

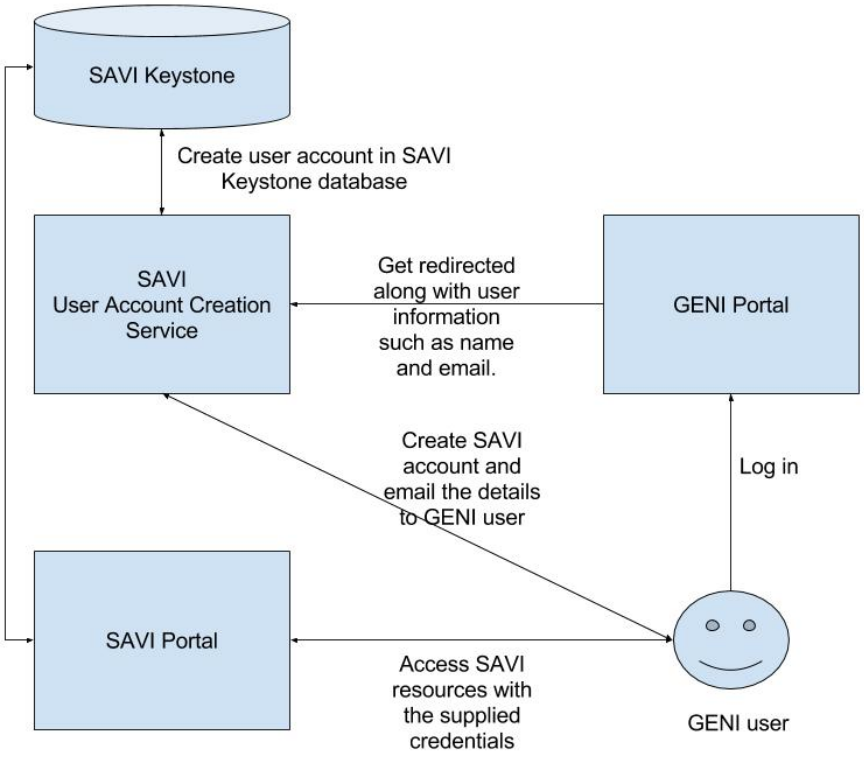

Figure 4.5: SAVI account creation for a GENI user

1. A GENI user logs into GENI Portal and chooses to access SAVI resources.
2. The GENI user is redirected to SAVI User Account Creation Service (UACS) along with the GENI user name and email.

3. SAVI UACS creates a unique user id in SAVI Keystone database after checking that a user with the same email doesn't already exist.

4. SAVI UACS emails the GENI user their credentials in order to access the SAVI Portal. Please note that a SAVI administrator must approve this step.

5. The GENI user accesses the SAVI Portal in order to access SAVI resources using their newly granted credentials.

SAVI User Account Creation Service (UACS)

We now describe UACS in a bit more detail. UACS is hosted on a virtual machine on a SAVI edge. In fact, it's hosted on the same virtual machine as the SAVI shibboleth IdP, and is programmed using Python and Flask [63] web application development framework. The process illustrated in Figure 4.6 is as follows:

1. A GENI user sends an http request, which contains the user name, email, and a GENI X509 certificate. X509 is an authentication protocol, and X509 certificates are used to verify the authenticity of the originator of an incoming request [62]. In this case, an X509 certificate in form of a PEM file [73] is acquired from GENI beforehand and placed on the server so that UACS can access it and extract the GENI public key from it.

2. UACS then extracts the user name, email and the GENI X509 certificate and verifies that the X509 certificate is authentic by checking it against the GENI public key extracted from the resident GENI certificate.

3. UACS then sends a user account creation request to SAVI Identity Access Management service [20], which creates the user account in SAVI Keystone.

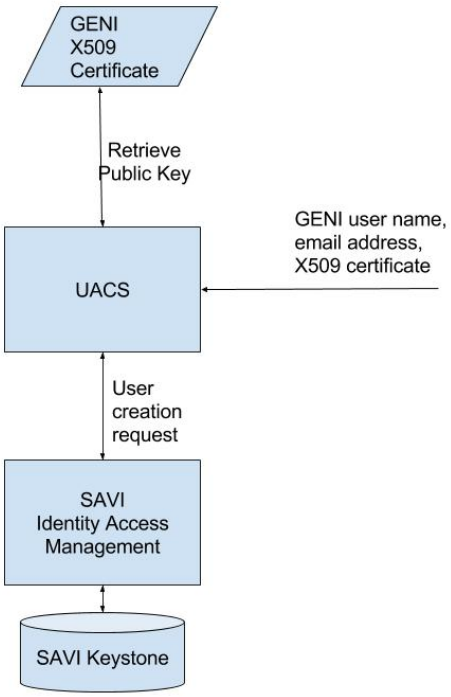

Figure 4.6: SAVI User Account Creation Service

4. If the user account creation is successful, UACS sends back a success response and emails the user their new SAVI credentials.

4.2 Accessing GENI Resources

Once the SAVI IdP has been setup and configured properly, a SAVI user can log into the GENI Portal by being redirected to SAVI IdP and authenticating themselves. After logging in, the user can access and procure GENI resources using the GENI Portal web interface, or they can do so using their computer's command line shell by downloading and setting up Omni Client as described in section A.2.

4.3 Accessing SAVI Resources

Once the GENI user has acquired SAVI credentials, they can log into SAVI Portal and access SAVI resources. Another way to access SAVI resources is by downloading and using the NOVA client tools from the command line. Accessing SAVI resources is covered in more detail in section A.3.

4.4 SAGEFed

SAGEFed acts as a wrapper around both NOVA and OMNI client tools. Rather than learning the NOVA commands for accessing SAVI and OMNI commands for accessing GENI, the user can just use the SAGEFed commands to access both the resources. Main component of SAGEFed is a Linux Bash script which translates valid user commands to either NOVA or OMNI commands and executes them against SAVI or GENI respectively. In order to use SAGEFed, NOVA and OMNI tools must be setup and configured with user account credentials for SAVI and GENI respectively. Sections A.2 and A.3 discuss how to setup OMNI and NOVA client tools on a user's machine.

SAGEFed has five major components in the form of a directory structure:

- ***SAGEFed.sh***: SAGEFed.sh is a Linux Bash script as described above. A user can execute this script by passing in a valid SAGEFed command which is then translated by the script and executed against the underlying NOVA or OMNI client tools.

- ***image***: Image folder consists of unique identifiers for all SAVI operating system images. A SAGEFed command will refer to one of these image ids when creating a virtual machine instance on SAVI.

- ***rspecs***: Rspecs folder consists of all GENI rspec files discussed in section 3.4.1. A SAGEFed command will refer to one of these rspec file identifiers, which determine a virtual machine's operating system characteristics when creating a virtual machine instance on GENI.

- ***tmp***: Tmp folder stores all the temporary logs that are generated when executing a SAGEFed command. If a command fails, the user can investigate the issue by looking at the files in the tmp folder.

- ***logs***: Logs folder stores timestamped logs of the SAGEFed commands executed by the user and their results.

There are four stages that a SAGEFed command goes through when it is executed by a user as shown in Figure 4.7:

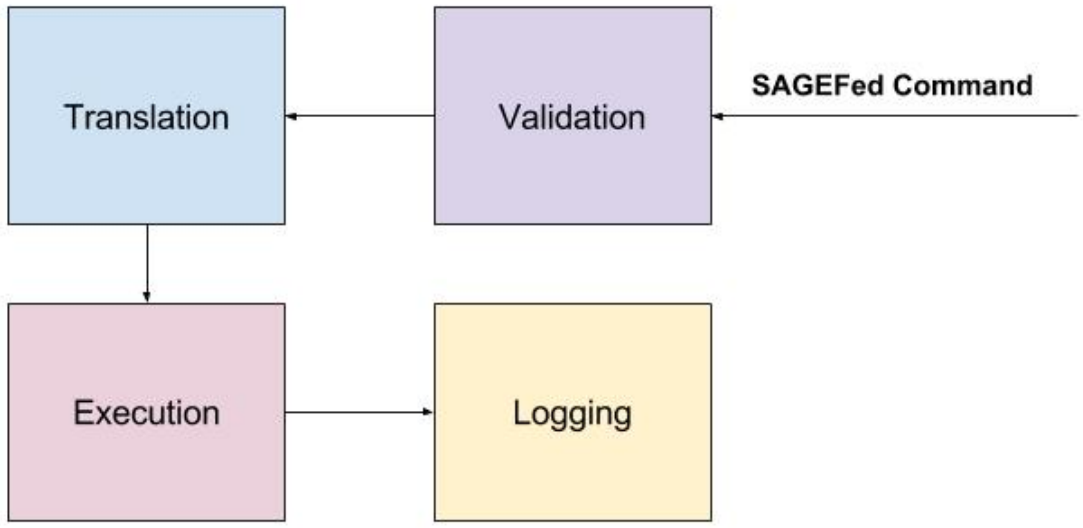

Figure 4.7: SAGEFed Command Stages

1. ***Validation***: SAGEFed first validates a command. As indicated in Figure 4.8, a command can be divided into three components: the command, the system,

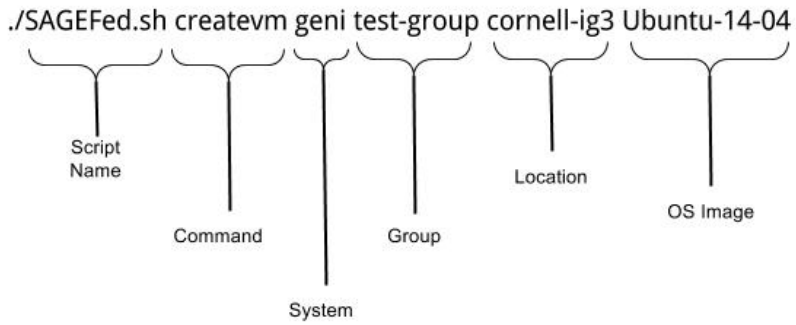

Figure 4.8: Anatomy of a SAGEFed command

and the parameters. The parameters differ according to the command and the system it is being executed against. During validation, SAGEFed must make sure that the command is valid, the system is valid, and the parameters are valid for the particular command and the system.

2. ***Translation***: Once a command has been validated, it gets translated into either an OMNI command or a NOVA command depending on whether the target system is GENI or SAVI respectively. Please note that this isn't always a one-to-one translation. A SAGEFed command may be translated into a set of multiple OMNI or NOVA commands to accomplish the task.

3. ***Execution***: The translated command is then executed using either OMNI or NOVA tools and the result is returned back to the translator to be translated in a user-friendly format.

4. ***Logging***: Any executed command and its result is logged in timestamped log files for troubleshooting.

SAGEFed currently supports the following commands:

1. ***Createslice***: Creates a slice on GENI. This command is only available to be used on GENI.

2. ***Deleteslice***: Deletes a slice on GENI. This command is only available to be used on GENI.

3. ***Createvm***: Creates a particular type of virtual machine on either SAVI or GENI. It requires arguments such as a group name, type of operating system, location, etc.

4. ***Deletevm***: Deletes the virtual machine that was created on either SAVI or GENI in order to free up resources.

5. ***Listinstance***: Lists instances of existing virtual machines for a particular user on both GENI and SAVI. This command can take additional arguments in order to restrict the list to either GENI or SAVI or even just a group.

4.5 Summary

We provided a comprehensive look the GENI-SAVI Federation architecture in this chapter. We discussed the Shibboleth-IdP and the UACS services which authenticate existing GENI and SAVI users onto each others' systems. We also looked at the architecture of the SAGEFed tool, that lets users access the resources on both GENI and SAVI systems.

Chapter 5

Experimentation and Evaluation

5.1 Experiments

In addition to testing the SAVI-GENI federation implementation on our own, we had the opportunity to test our setup on two separate occasions:

- ***Tridentcom***: An annual conference sponsored by European Alliance for Innovation which showcases computing and network testbeds and research infrastructures [37]. Researchers are invited to submit articles related to the conference theme and present their work.
- ***SAVI AGM (Annual General Meeting)*** : An annual conference sponsored in part by SAVI and NSERC which exhibits a variety of research projects that leverage SAVI resources [67]. Researchers present their work in the form of presentations, demos, and tutorials.

5.1.1 User Study

We followed an exploratory mixed-methods approach when doing our user study. We wanted to measure user satisfaction level from two different user perspectives:

an experienced user, and a new user. Our goal was to determine whether our work enhanced the user experience by making it easier for users to accomplish typical tasks. We selected two participants, whom we shall refer to as Participant A and Participant B. Participant A was an experienced SAVI and GENI user, whereas Participant B had no prior experience with either of the two systems.

5.1.2 Experiment 1: A SAVI User Accessing GENI

The purpose of this experiment was to measure the experience of a user who already had a SAVI account, and wanted access to GENI resources. We had our two participants follow the steps as outlined in section A.4.2: SAVI to GENI. Please note that both the participants were provided SAVI credentials before beginning this experiment. The goal of this task is to be able to create and then delete a virtual machine on GENI.

The Steps Without Federation

1. Visit GENI Portal and request an account.
2. Fill out a form with various fields.
3. Get an email with login access.
4. Log in to GENI Portal.
5. Go to Profile, and generate SSH Keys and SSL certificate.
6. Download and configure OMNI Tools from OMNI Tools section.
7. Request and procure access to an existing GENI Project. This must be approved by a GENI Project Lead.

8. Create a VM on one of the available GENI racks using the OMNI tool (1 command).
9. List all user instances using the OMNI tool (5 commands).
10. Delete the VM using the OMNI tool (1 command).

The Steps With Federation

1. Visit GENI Portal and select SAVI (Smart Applications on Virtual Infrastructure) [44]. This will redirect you to SAVI's Shibboleth IDP login screen.
2. Enter your SAVI credentials. This will redirect you back to the GENI Portal and you will automatically be logged in to the portal.
3. Go to Profile, and generate SSH Keys and SSL certificate.
4. Download and configure OMNI Tools from OMNI Tools section.
5. Request and procure access to an existing GENI Project. This must be approved by a GENI Project Lead.
6. Download and setup SAGEFed.
7. Create a VM on one of the available GENI racks using SAGEFed tool (1 command).
8. List all user instances using SAGEFed tool (1 command).
9. Delete the VM using SAGEFed tool (1 command).

The Results

We asked both the participants regarding their overall experience with both the approaches when accomplishing the given task. Both participants were highly satisfied with the federation approach. They liked the fact that they did not have to create and remember new account details when accessing GENI. They also liked how fast they were able to get access to GENI. They also liked the simplicity and clarity of SAGEFed. They were made aware that currently the available commands in SAGEFed were limited, but can be extended in the future.

We recorded the number of steps and the time it took in seconds for each participant to finish a corresponding step. Table 5.1 summarizes this data for the experienced participant (Participant A) and 5.2 summarizes the same data for the new participant (Participant B). It is clear from both the tables that the federated approach has fewer steps overall and takes less time for the users to perform the given task on GENI. Please note that we only recorded the time for steps while the user was actively performing some action. For example, we disregarded the time the users were waiting to be added to a GENI group.

5.1.3 Experiment 2: A GENI User Accessing SAVI

The purpose of this experiment was to measure the experience of a user who already had a GENI account, and wanted access to SAVI resources. We had our two participants follow the steps as outlined in section A.4.1: GENI to SAVI. Please note that both the participants were provided GENI credentials before beginning this experiment. The goal of this task is to be able to create and then delete a virtual machine on SAVI. Below are the steps that the participants followed:

Stage	With Federation + SAGEFed		Without Federation + OMNI	
	Number of Steps	Time (seconds)	Number of Steps	Time (seconds)
Logging into GENI	3	18	5	127
Tool Setup	3	66	2	45
Creating a VM	1	60	1	60
Listing Instances	1	130	5	180
Deleting a VM	1	60	1	60
Totals	9	334	14	472

Table 5.1: Experiment 1 - comparing federated and non-federated approach for an experienced user (Participant A) accessing GENI

Stage	With Federation + SAGEFed		Without Federation + OMNI	
	Number of Steps	Time (seconds)	Number of Steps	Time (seconds)
Logging into GENI	3	22	5	131
Tool Setup	3	75	2	55
Creating a VM	1	60	1	70
Listing Instances	1	130	5	200
Deleting a VM	1	60	1	60
Totals	9	347	14	516

Table 5.2: Experiment 1 - comparing federated and non-federated approach for a new user (Participant B) accessing GENI

The Steps With Federation

1. Log into GENI Portal and get redirected to SAVI User Account Creation Service.
2. Receive credentials in email.
3. Log into SAVI.
4. Request and procure access to an existing SAVI Tenant. This must be approved by a SAVI administrator.
5. Download and install NOVA client.
6. Download and setup SAGEFed.
7. Create a VM on one of the SAVI sites using SAGEFed (1 command).
8. List all user instances using SAGEFed (1 command).
9. Delete the VM using SAGEFed (1 command).

The Steps Without Federation

1. Request SAVI credentials via email.
2. Receive credentials in email.
3. Log into SAVI.
4. Request and procure access to an existing SAVI Tenant. This must be approved by a SAVI administrator.
5. Download and install NOVA Client Tools.
6. Create a VM on one of the SAVI sites using the NOVA client (7 commands).

7. List all user instances using the NOVA client (2 commands).

8. Delete the VM using the NOVA client (2 commands).

The Results

Both the participants preferred the federated approach to using SAVI. They liked the fact that they quickly received their new account credentials rather than having to wait for the administrator. However, they were not as satisfied as they were during Experiment 1. They still had to have new account credentials when accessing SAVI. The new user expressed security concerns over receiving the password in her email. When the experienced user was asked about the security concern, she mentioned that SAVI sends new user credentials over the email anyway.

Both the users were highly satisfied with SAGEFed when performing the given task on SAVI. They liked the fact that they didn't have to enter as many commands to do a typical task of creating a virtual machine as they would with NOVA. The experienced user was particularly happy as she was still able to use NOVA client tools if she wanted to perform a task that was not supported on SAGEFed yet.

We recorded the number of steps and the time it took in seconds for each participant to finish a corresponding step. Table 5.3 summarizes this data for the experienced participant (Participant A) and 5.4 summarizes the same data for the new participant (Participant B). It is clear from both the tables that the federated approach has fewer steps overall and takes less time for the users to perform the given task on SAVI. Please note that we only recorded the time for steps while the user was actively performing some action. For example, we consider the time it took for the users to receive credentials in email or while they were waiting to be added to a SAVI tenant.

Stage	With Federation + SAGEFed		Without Federation + NOVA	
	Number of Steps	Time (seconds)	Number of Steps	Time (seconds)
Access SAVI	3	45	3	67
Tool Setup	2	69	1	58
Creating a VM	1	120	7	180
Listing Instances	1	25	2	30
Deleting a VM	1	25	2	30
Totals	8	284	15	365

Table 5.3: Experiment 2 - comparing federated and non-federated approach for an experienced user (Participant A) accessing SAVI

Stage	With Federation + SAGEFed		Without Federation + NOVA	
	Number of Steps	Time (seconds)	Number of Steps	Time (seconds)
Access SAVI	3	52	3	71
Tool Setup	2	73	1	62
Creating a VM	1	120	7	200
Listing Instances	1	25	2	60
Deleting a VM	1	25	2	60
Totals	8	295	15	453

Table 5.4: Experiment 2 - comparing federated and non-federated approach for a new user (Participant B) accessing SAVI

5.1.4 Conferences

We showcased our federation project at Tridentcom [37] and SAVI AGM conferences [67]. Our team prepared a tutorial in which the participants would perform several tasks by creating virtual machines on SAVI and GENI using the federation approach [69]. The tutorial was executed by a total of over 110 users: over 80 users in SAVI AGM, and 30 users in Tridentcom. Although we did not record any specific feedback, we were able to guide everybody through the tutorial without encountering any errors while authenticating users and executing commands on SAGEFed to perform several tasks such as creating virtual machines, and communicating information between them.

5.2 Evaluation

Based on the the experiments and the federation architecture, we now evaluate our work with respect to the following criteria: functionality, usability, productiveness, and security.

Functionality

Based on the experiments that we have done so far, we can say with confidence that our prototype does what it is supposed to do. We are able to allow a GENI user to be authenticated and get access to SAVI and a SAVI user to be authenticated and get access to GENI. The ***createslice*** and ***createvm*** commands provided by SAGEFed generate the virtual machines to their exact specifications on both SAVI and GENI. The ***listinstance*** command lists the current instances under the user accurately. The ***deleteslice*** and ***deletevm*** commands delete the slice and the virtual machine respectively.

Usability

Based on the user study in section 5.1.1, we determined that the users find the federation approach more user-friendly. They also find SAGEFed more usable than either NOVA or OMNI tools when using the functionality provided so far.

Productiveness

Based on the data produced by the user study in section 5.1.1, the federation approach reduces the number of steps and time taken in authenticating the user, and procuring resources on both SAVI and GENI using SAGEFed. We therefore conclude that our approach increases user productivity.

Security

One of our goals when undertaking this project was to accomplish our task with minimal changes to the underlying systems. To that end, we have introduced a few services in order to facilitate federation as described in Chapter 4. As a result of adding these services to the system, we have increased the endpoints for an attacker to target our system. This is indeed a trade-off, and we realize the security implications. Having said that, the services that we have deployed are based on systems that are widely used and have been around for a while.

One area of concern is the SAVI User Account Creation Service (UACS) as described in 4.1.6. Currently, the service sends user credentials, including plain text passwords, to the user's email account when creating a SAVI account for them. This is a potential security issue in the case if a user loses control of their email account to an attacker. This issue is not unique to UACS. The existing SAVI process for user account creation uses the same mechanism. We recommend that this process should be evaluated and a better process for user account creation should be put in place.

5.3 Summary

In this chapter we performed an experiment where the participants compared the use of GENI-SAVI federation approach versus the non-federated approach when accessing resources on the cloud. We then evaluated our federation project using a set of criteria. Our conclusion is that our federated approach increases usability, and productivity of users and that we have met our original goals. As we continue to further develop this prototype, we must take into account the security aspect and make some improvements.

Chapter 6

Conclusions

The GENI-SAVI federation project resulted in a working proof-of-concept prototype, which we were able to experiment several times with positive results and have already integrated it into the live GENI and SAVI systems. This has proven that we can combine resources of two different cloud systems by avoiding making any significant changes to the underlying systems by using a highly modular design. We have shown that by using well-proven authentication technologies such as Shibboleth and X.509, we can let users access both the systems very quickly and easily. By creating an intuitive client interface such as SAGEFed, we can allow the user to easily manipulate resources on both GENI and SAVI. At the same time, SAGEFed is flexible enough to allow the users to use Nova and Omni clients for more advanced tasks. With this federation in place, users will now have access to the combined resources of both the systems. The users can now scale their tasks not just in terms of physical resources, but also in terms of geographical boundaries and availability.

We divided the federation project into three parts: authenticating SAVI users onto GENI, authenticating GENI users onto SAVI, and the SAGEFed interface. Authenticating SAVI users onto GENI was accomplished by creating a separate service

using Shibboleth, which could verify to GENI that the incoming user was indeed a SAVI user. Authenticating GENI users onto SAVI was accomplished by creating yet another service which would authenticate a GENI user based on an X.509 certificate contained within the user request. SAGEFed interface was a set of commands created using a shell script on top of already existing command line tools namely, Nova and Omni. SAGEFed validates and translates the incoming user commands and passes them onto Nova and Omni to execute.

Once we finished with all the three portions of the project and had a working prototype, we had two participants interact with it and give us feedback. We found that our prototype saved them a significant amount of time and reduced the steps they had to take in order to accomplish certain tasks. The participants found the user experience of our prototype to be of higher quality than the existing GENI and SAVI tools. We also got the opportunity to showcase our prototype in two different conferences, where over 100 users tested it without any issues.

6.1 Future Work

Our prototype is functioning correctly and has been tested rigorously, but there are still a few improvements we could make in the future to make it more robust and secure. There might be some trade-offs involved when making some of the changes, so future researchers should first evaluate the suggested changes accordingly.

6.1.1 UACS

As mentioned in Section 5.2, the User Account Creation Service (UACS) currently emails the user credentials, including passwords, to the user. This process is not very secure, and as such it should be modified to just sending the user a secure link

which allows the user to choose their password. Since the UACS is hosted using Python-Flask, there is a package called "itsdangerous" we can use to accomplish this [58].

6.1.2 SAVI IdP

In order to authenticate SAVI users onto GENI, we host a service called Shibboleth IdP on SAVI as described in Section 4.1.5. The Shibboleth IdP verifies the user credentials against an OpenLDAP database, whose user data is imported from the SAVI Keystone database on a regular basis. If in the future we would like to do away with this dependency, we can modify SAVI Keystone itself to act as an Identity Provider as detailed in [56]. The trade-off here is that we would have to modify a major component of SAVI in exchange for simplifying the architecture.

6.1.3 Housekeeping Notes

All the information regarding the software, versioning, and dependencies related to the prototype is listed below:

Shibboleth IdP/OpenLDAP Installation Guide

We followed the guide on *cybera* in order to install and run the Shibboleth IdP service along with OpenLDAP [59]. This guide was very helpful as the official Shibboleth documentation can be very confusing and hard to follow. We used Shibboleth version 2.4.2 as suggested by the guide. We used a SAVI VM to host the service. Due to security reasons, we can not provide the IP address or credentials for this machine in this document, but these can be made available upon request. The exact folder locations for Shibboleth and Tomcat are also as suggested by the guide, so they are easy to find on the VM.

UACS Guide

The UACS is hosted on the same SAVI VM as the Shibboleth IdP service. The IP address and credentials are available upon request. UACS is located in **~/portal** folder on the VM. We used Python and Flask to create the service. The service uses X509 certificates to authenticate the user. The service documentation and code can be found in the file **~/portal/server.py**.

SAGEFed Script

SAGEFed was created using Bash scripting and can be directly downloaded from **https://github.com/rickmcgeer/GENI**$_SAVI/$.

6.2 Final Thoughts

Cloud computing has now arrived at its third stage of evolution, called "Horizontal Federation" as outlined by Celesti et al [8]. In this stage, service providers pool their resources according to the need of their users to provide unlimited resources not just in terms of computing power or storage, but also in terms of location and quality of service. Cloud computing has a tremendous potential to transform the way we do business and research. Researchers can run large scale experiments with ease which would surpass the power of centralized super computers of the past. Businesses can afford to expand their reach worldwide. Individuals can harness the power of cloud computing to start small businesses and achieve independence.

While there are many benefits to cloud computing, we must also consider the risks involved in our approach. For example, with the shared

nature of the cloud, several applications from several users can be hosted on a single physical machine. If security boundaries are not set correctly, this can pose major security risks [11]. There is also the issue of trust when federating with different cloud providers. Can the cloud provider we are federating with be trusted enough to host our customer's data? Who is responsible for any data loss or breaches? There are several other issues we must face such as loss of control, and data ownership with certain kinds of cloud computing services such as Google Docs [7].

Cloud computing has a huge promise for the benefit of humanity. It can allow us to collaborate in ways never before possible. It can transform the way we live and at the rate we make progress. However, we must also develop and follow proper standards and practices in order to reduce the known and unknown risks of this promising technology.

Appendix A

Additional Information

A.1 GENI Racks

A GENI Rack is a server which has custom hardware and software designed to run as part of the GENI network or cloud. A GENI Rack that has been deployed to a site can be referred to as an aggregate as described in Section 3.4.1. There are a few different types of racks deployed on various sites as part of the GENI network [76]. These Racks have custom hardware provided by hardware vendors as well as custom software as mentioned in the above sections. It is beyond the scope of this thesis to cover all of these in detail. We cover OpenGENI below to give the reader a good idea of a GENI Rack.

OpenGENI

OpenGENI can be described as a hardware installation (provided by Dell) rack with Openstack based custom software to manage its resources [50]. An academic organization can order an OpenGENI rack which can be

installed with relative ease, which is the main appeal of OpenGENI. Below is an overview of OpenGENI software architecture. Please see Figure A.1 for reference.

- Openstack/Openstack API: Custom OpenGENI software components use Openstack software to manage virtual environments and resources via the Openstack API.
- GRAM/GRAM-AM: GRAM (GENI Rack Aggregate Manager) is a collection of services that allows management of virtual environments and network links between them for experimentation purposes. OMNI client is one of many tools which interfaces with a GRAM based system and provides commands that a user can use to manage resources on such a system [33].
- OpenFlow/OpenFlow API: GRAM controls bi-directional flow between virtual environments on different hosts by interfacing with OpenFlow switches using OpenFlow API.

Above offers a quick glance at OpenGENI software architecture. The main purpose of the software is to provide management of virtual and network resources.

A.1.1 GENI Rack Specifications

GENI has certain specifications that all custom racks must meet to be a part of the GENI network [45]. These specifications encompass the following areas:

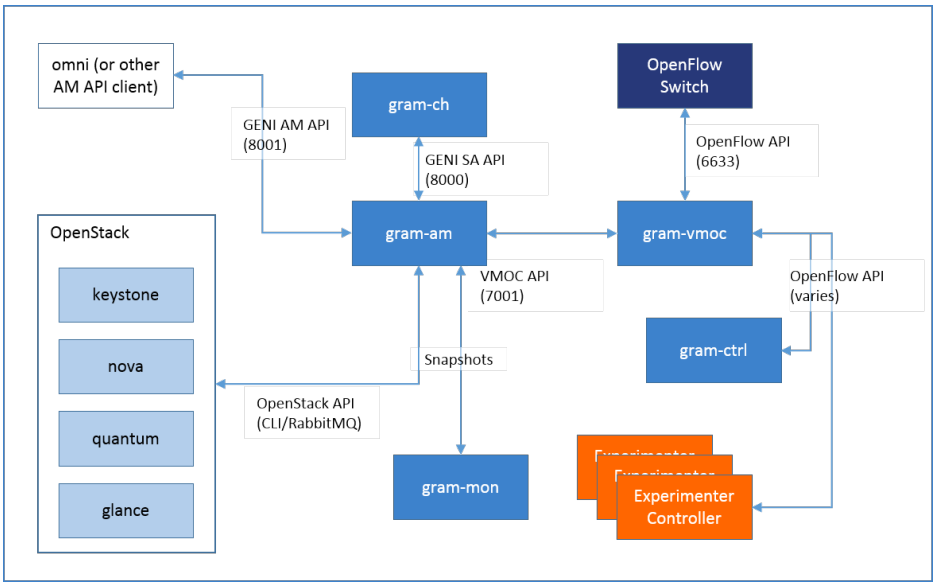

Figure A.1: OpenGENI Software Architecture [50].

- **Software Requirements: such as delivering an Aggregate Manager for reserving compute and network resources, types of operating systems, complying with GENI AM API, and GENI standard RSpecs [41].**

- **Integration Requirements: ability to inter-operate with other GENI production aggregates, particularly with other GENI OpenFlow switches and compute resources [49]. This consists of compute, network, and rack resource requirements.**

- **Monitoring Requirements: include the type of data, frequency, and interface each rack must provide to allow for active monitoring of an aggregate.**

- **Production Aggregate Requirements: the requirements that the rack production team must meet for the initial 3 month period before handing it off to the site aggregate managers.**

- **Local Aggregate Owner Requirements: these include requirements**

such as backup, power, air conditioning, network, versioning, staffing requirements, etc. that site managers need to maintaining an aggregate.

- **Experimenter Requirements: these include the provisions an aggregate must have for experimenters such as ability to log in as admin into their virtual machines, network and flow control access, etc.**

The above is a very high level look at the GENI rack specifications. The GENI Rack Specification guide contains the full details on this topic.

A.2 Accessing Resources on GENI

Before a user can access resources on GENI, they must perform the following prerequisites:

1. **Get GENI credentials from a participating organization or from an organization that implements InCommon federation [42]. Any organization which implements the InCommon federation is able to authenticate its users onto GENI. Another option is to register with GENI directly and be authenticated with the GENI Identity Provider. Acquiring GENI credentials can be achieved by following the steps on the GENI Wiki [77].**
2. **Join a GENI Project. The user will need the permission of a GENI Project Lead [77].**
3. **Login to the GENI Portal and generate SSH Keys [44] [74].**
4. **Download Omni Client from the GENI Portal and configure on local system shell [44].**

A user can access GENI resources in multiple ways [78]. We will focus on two ways that are relevant to our project:

1. **Via the GENI Web Portal [44].**
2. **Via their local shell or terminal with the help of Omni Client Tools [48] [Omniapiclient].**

Accessing GENI Resources Using GENI Web Portal

While it is possible to create, manage, and delete resources using the GENI Web Portal, it is not as flexible as using the Omni Client. At some point,

the user must use their command shell or terminal to access their resources and do actual work. However, the GENI Web Portal does an excellent job of getting a user acquainted with the basic process of reserving resources on one of GENI sites. Below is a step-by-step guide to creating a slice and reserving resources on a GENI site. This also serves as a concrete review of the concepts covered in the above sections. Make sure to first create an account on GENI and join a GENI Project. See section A.2.

1. **Log in to GENI by clicking USE GENI and selecting your organization [44]. See figures A.2 and A.3.**

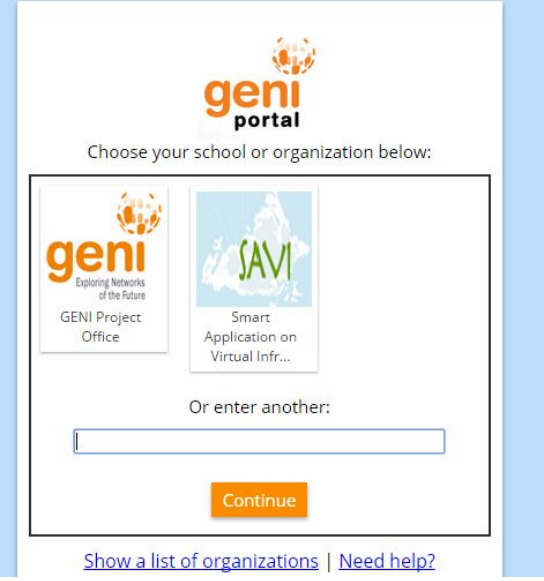

Figure A.2: Organization Selection

2. **Select a GENI project where you will add resources. See figure A.4.**

3. **Create a new Slice within the project by clicking "Create Slice" and give it appropriate name and description. See figure A.5.**

4. **Select and click "Add Resources" on the newly created Slice and Choose a site where you want to reserve resources as in figure A.7.**

Figure A.3: GENI Login Screen

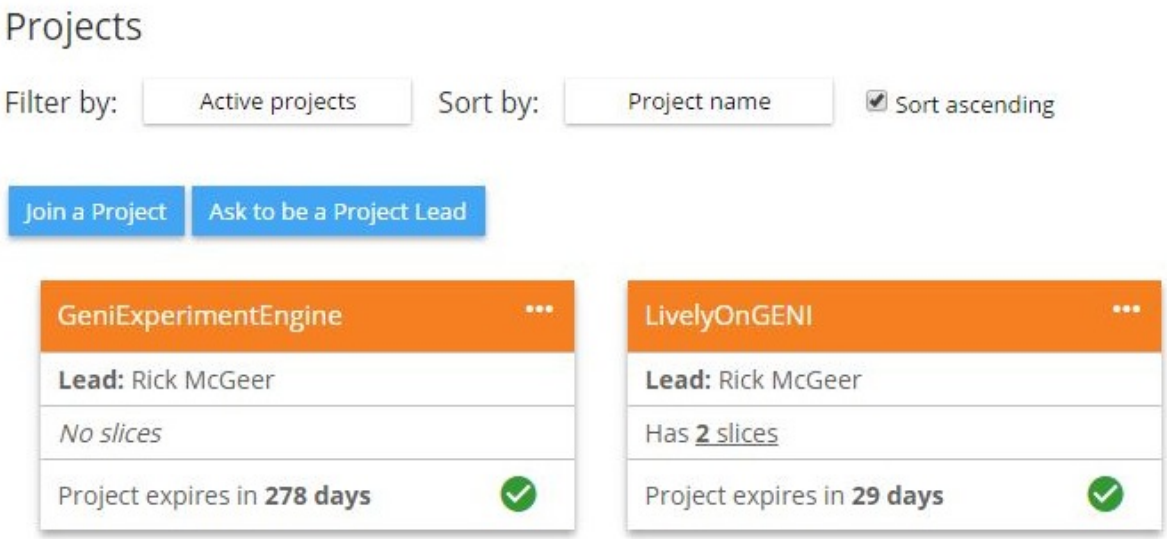

Figure A.4: List of User's GENI Projects

Under the covers, this reads an RSpec manifest file for the selected site in order to generate a resource on it as discussed in section 3.4.1.

5. **Select the type of resource you want and then drag and drop it onto the site. In our case, we selected a "VM" resource. See figure A.8.**

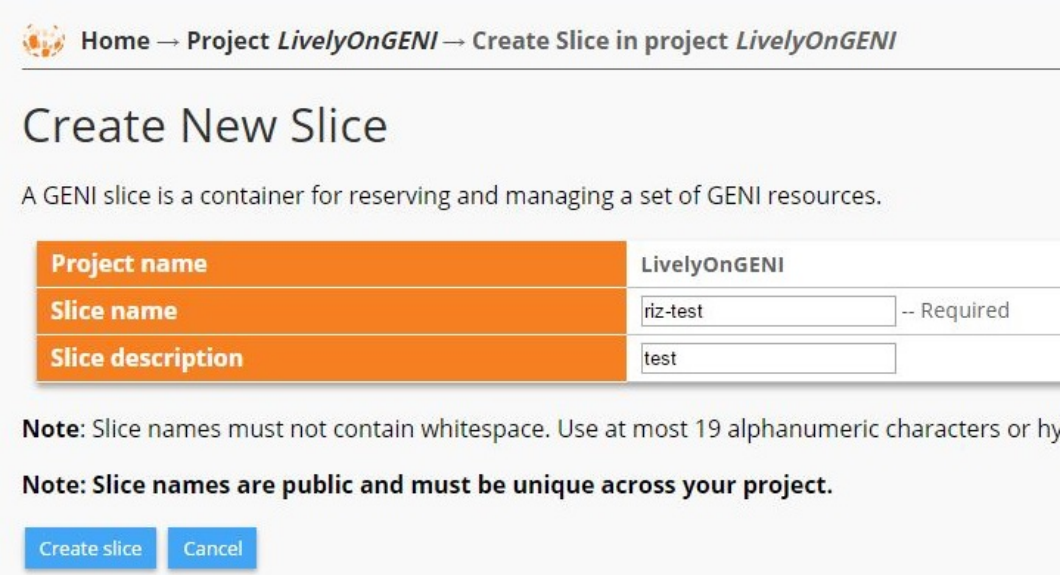

Figure A.5: Naming a GENI Slice

Figure A.6: Adding resources to a GENI Slice

6. **Select the resource properties such as name, hardware type, disk image and node type and click "Add Resources" button. We left everything at default in our example in figure A.9. This ends up creating a GENI Sliver on the selected site as discussed in section 3.4.1.**

7. **You can then view the log which shows the current status of the created Sliver as in figure A.10.**

Figure A.7: Choosing a GENI site to add resources.

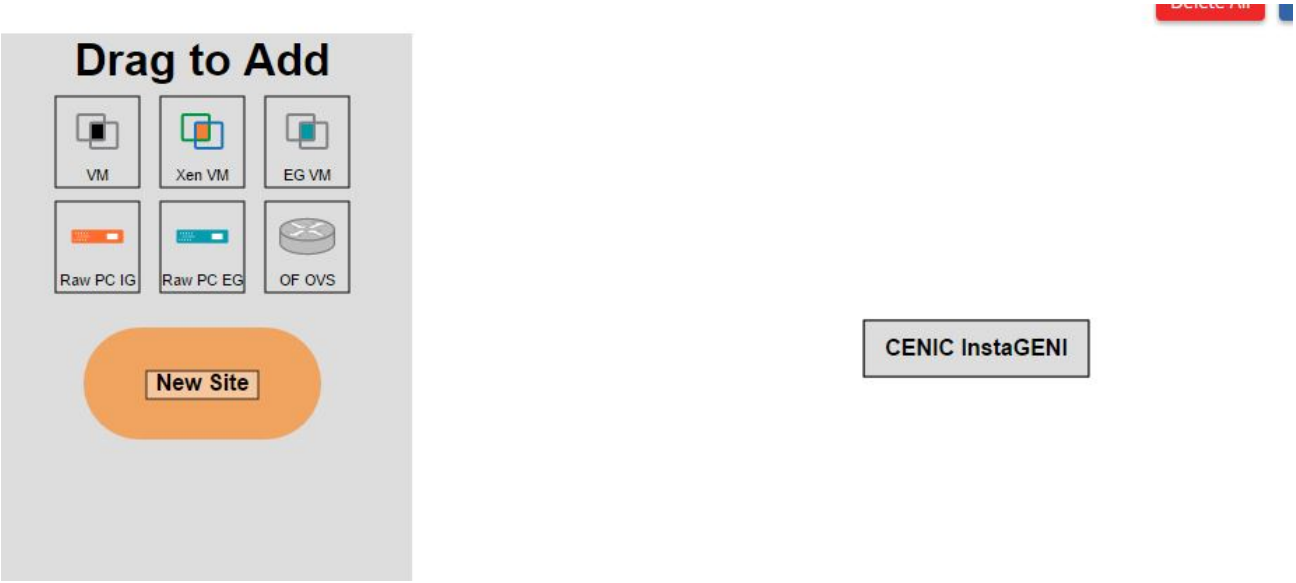

Figure A.8: Selecting a resource type.

8. **After the resource has been allocated, you can select the resource and view the domain names, user name accounts, and ports with which to connect to the resource. In our example, it's "rap@pc3.cenic.instageni.net" as in figure A.11.**

Figure A.9: Selecting resource properties.

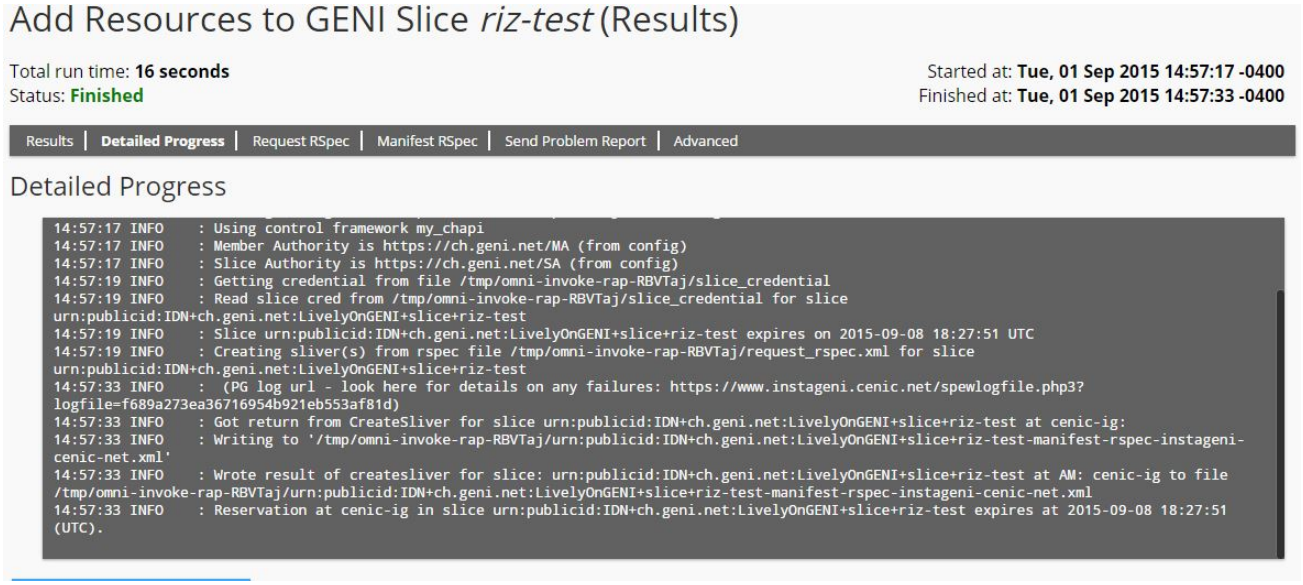

Figure A.10: Geni resource creation log.

9. **You can then ssh into the resource from a command shell. In our example, we use the command "ssh -p 34618 rap@pc2.cenic.instageni.net" to connect to our resource on "pc2.cenic.instageni.net" site under the account name "rap" on port "34618" as in figure A.12.**

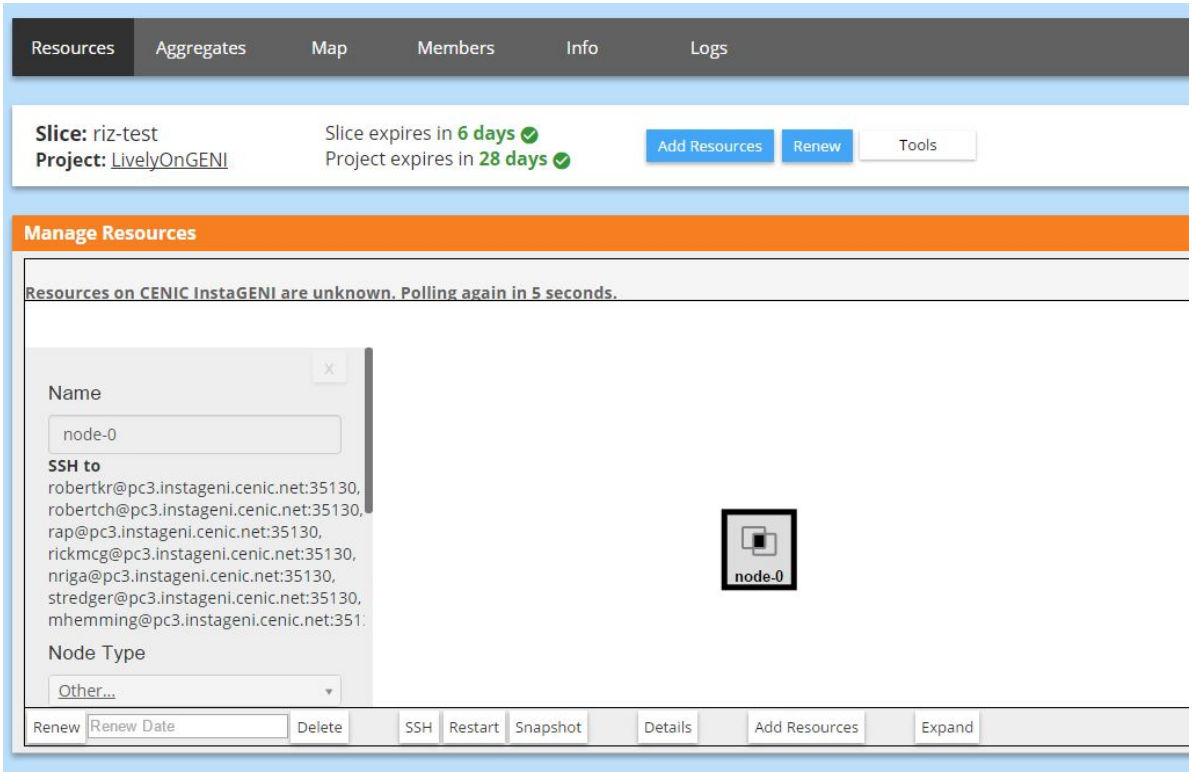

Figure A.11: Geni resource domain name and login information.

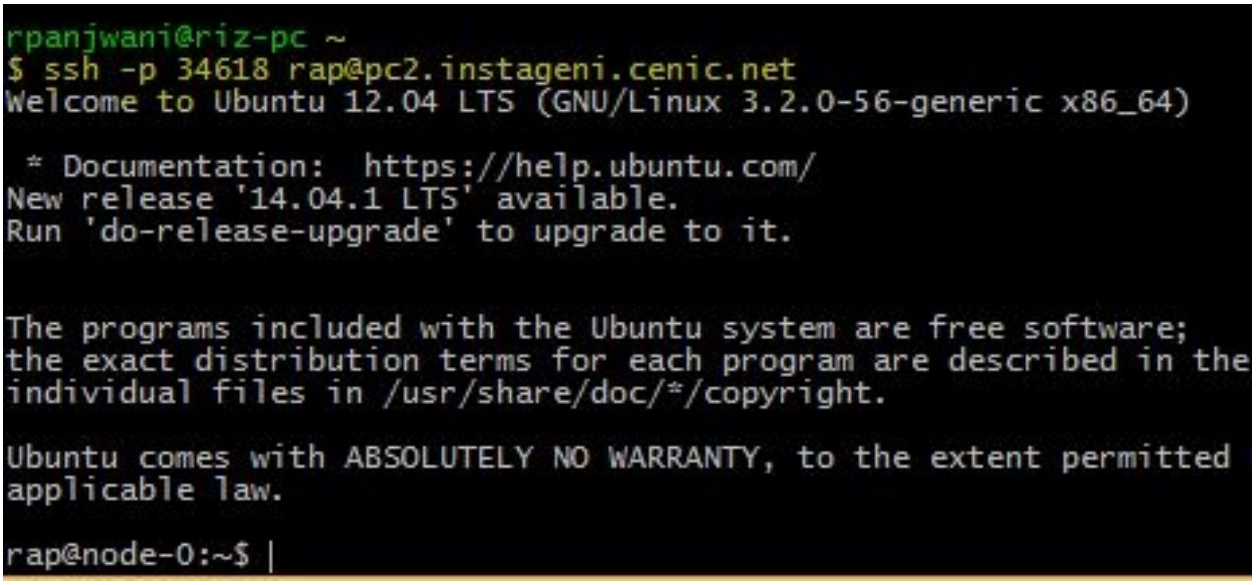

Figure A.12: SSHing into a GENI VM

10. **Once you are finished with the resource, you may delete the resource from the slice by selecting the slice and clicking "Delete Resources"**

button. Please note that you are not able to delete the slice itself on the Geni Web Portal. You can do so when using Omni Client, which gives you more fine-grained control over your GENI resources.

Figure A.13: Geni resource domain name and login information.

Acessing GENI Resources Using Omni Client Tool

Omni Client allows a user more flexibility in accessing GENI resources via their command shell or a terminal [33]. Below is an outline of a sample task a user can perform using Omni Client. Note that steps below are just a summary to give the reader an idea of the process. Please refer to a detailed Omni tutorial on the GENI website to try out Omni Client [79].

1. **Make sure you are part of a GENI Project.**
2. **Log in to the GENI Portal, configure an SSL certificate, and download Omni Client file.**
3. **From your command shell, configure Omni Client.**
4. **Issue the Omni command to create a GENI Slice for your experiment.**
5. **List all available GENI Aggregates or sites on the GENI network.**

6. **Issue the Omni command to create a GENI Sliver on a particular Aggregate as required for your experiment.**

7. **You can now ssh into the GENI Sliver or use other tools to run your experiment on the Sliver.**

8. **Issue the Omni command to delete the Sliver once you are done.**

9. **Issue the Omni command to delete the Slice if it is no longer needed.**

This concludes our tutorial on GENI-SAVI federation. We have demonstrated how a user with SAVI credentials can access GENI resources, and a user with GENI credentials can access SAVI resources.

A.3 SAVI Usage

In order to use the SAVI testbed, a user must first request access [66]. Once approved, the user will become a part of a project or a tenant. A tenant or a project in an OpenStack context is a group of users [57]. It is easier to manage users by specifying rules for a tenant around user access, security, permissions, quota, IP addresses and other resources.

SAVI Portal

Once a user has access to SAVI testbed, they can log into the SAVI portal and procure resources available to them such as creating Virtual Machine instances. Figure A.14 shows an instance creation dialog which allows the user to specify the flavor, image, access and security, and other options for the instance. The instance will then be created on the specified SAVI node. Once the instance is created, the user can assign it an IP address from a pool of IP addresses available and access the instance by using SSH (secure shell) [74].

Nova Client Shell

Another way to access resources on SAVI is by using Nova Client Shell [55]. Nova Client can be used via a command line shell, and as such provides much more flexibility and functionality to the user than the web interface. A user must first install and configure the Nova Client Shell onto their computer in order to issue Nova commands to create, access, and delete resources among other things on the SAVI testbed. SAVI group has provided a video tutorial which covers how to create and manage

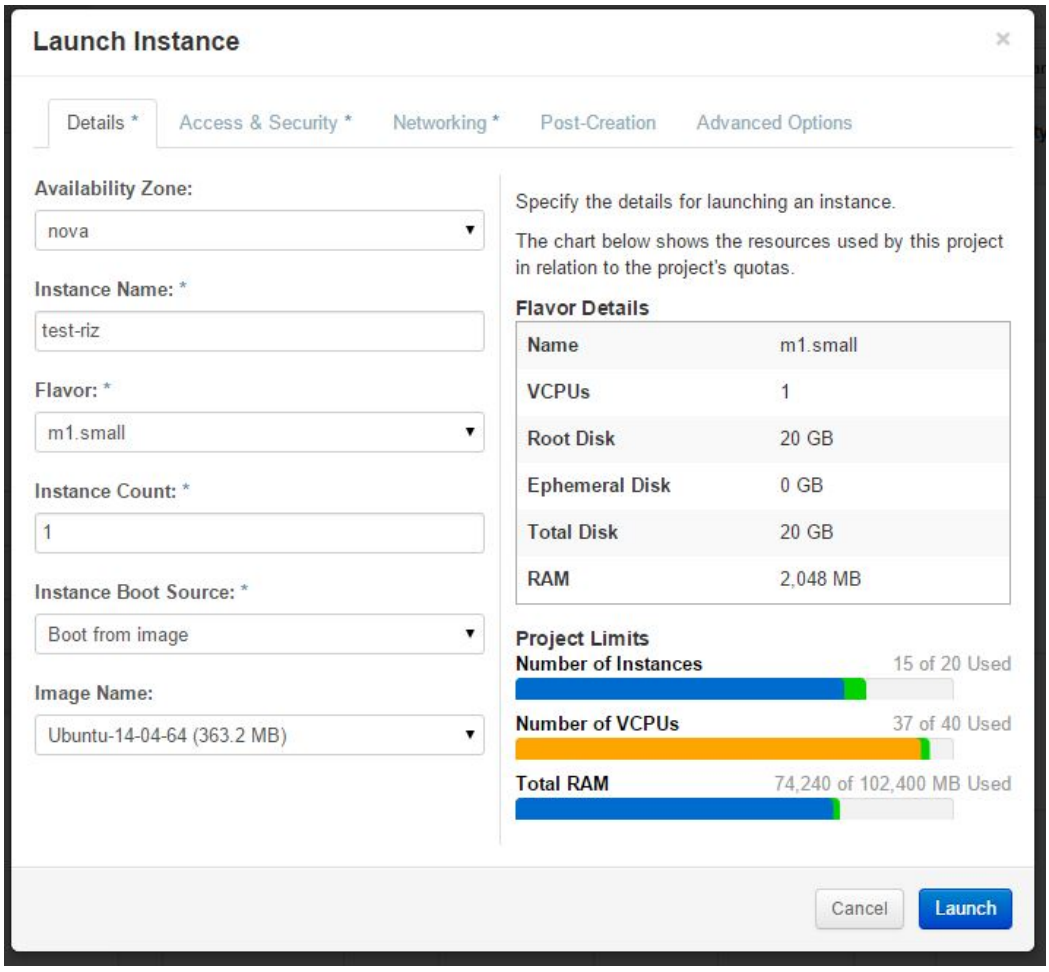

Figure A.14: Creating an instance on a SAVI node

resources on SAVI testbed using the SAVI Portal as well as Nova Client Shell [15].

A.4 GENI-SAVI Federation Tutorial

In this section we present the tutorial that we used at TridentCom to test our prototype. Before proceeding with the tutorial, please download the SAGEFed tool using the set of commands in Listing A.1.

Listing A.1: Downloading the SAGEFed tool

```
wget http://web.uvic.ca/~sushilb/federation/tutorial.tar
tar xvf tutorial.tar
cd tutorial
```

A.4.1 GENI User Accessing SAVI Resources

In this tutorial we demonstrate how a GENI user can create an account on SAVI and access SAVI resources. Please make sure you have a GENI account before starting this tutorial as detailed in A.2.

1. **Log into GENI, click on your name in the top right, select "Manage Accounts" and click "Create SAVI Account" as shown in Figure A.15.**

2. **You will be redirected to the SAVI service which will create a SAVI account to you and email you the credentials as shown in Figure A.16. The email used will be the one you have regisered under your GENI account.**

3. **Create a vm on SAVI as shown in Figure A.17 using SAGEFed.**

4. **Use your credentials when prompted and enter a tenant id as shown in Figure A.18.**

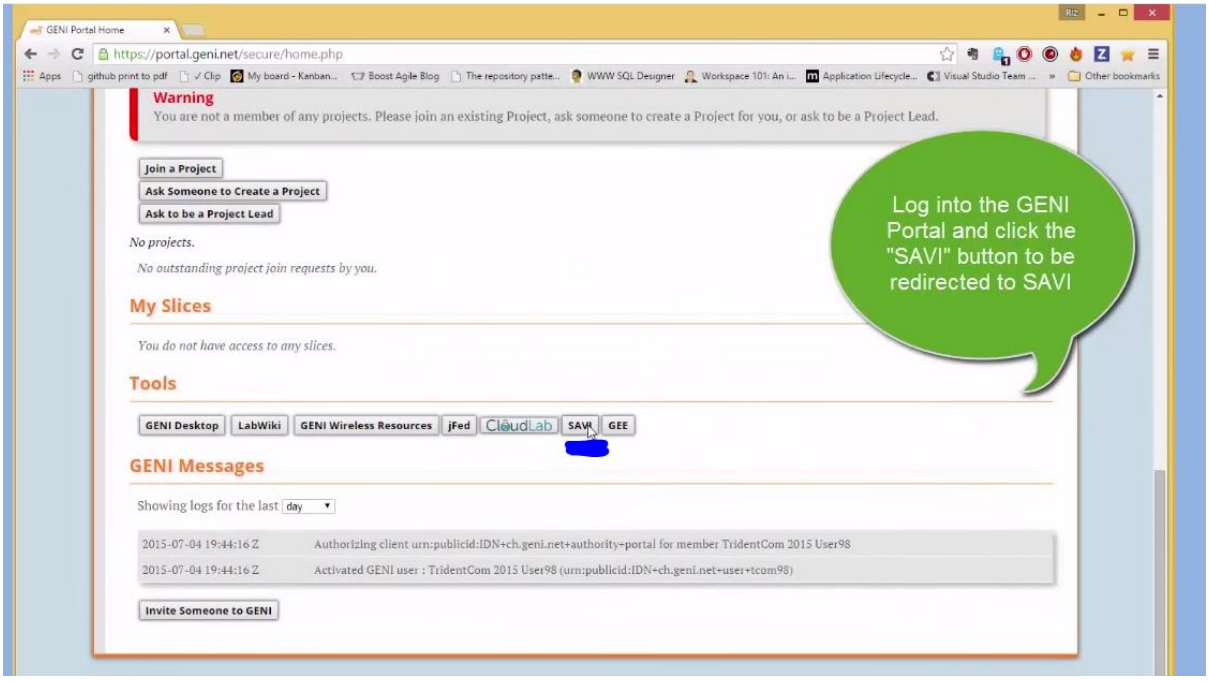

Figure A.15: GENI user accessing SAVI using federation - Step 1

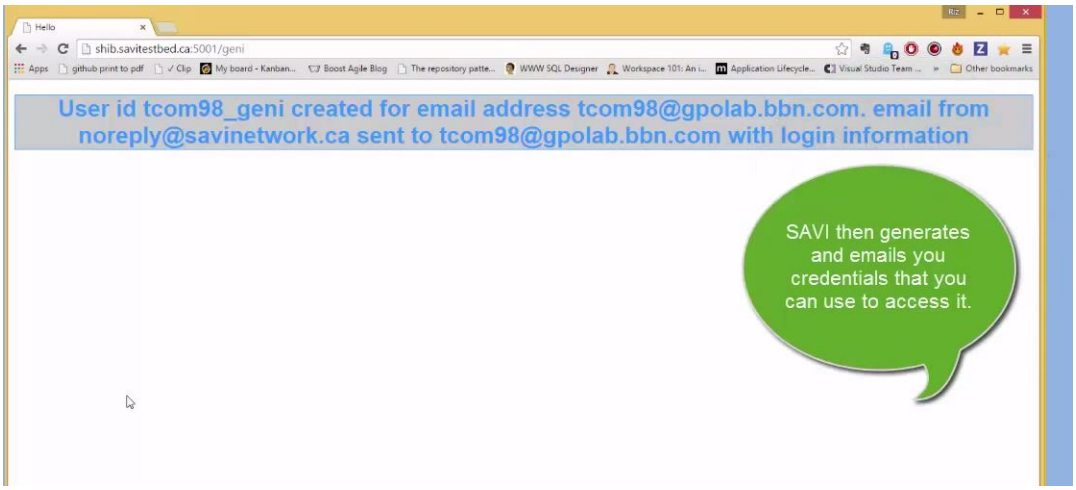

Figure A.16: GENI user accessing SAVI using federation - Step 2

5. **SAGEFed will create a VM on SAVI which has a public ip address and will display you the results as shown in Figure A.19.**

6. **SSH into the newly created VM as shown in Figure A.20.**

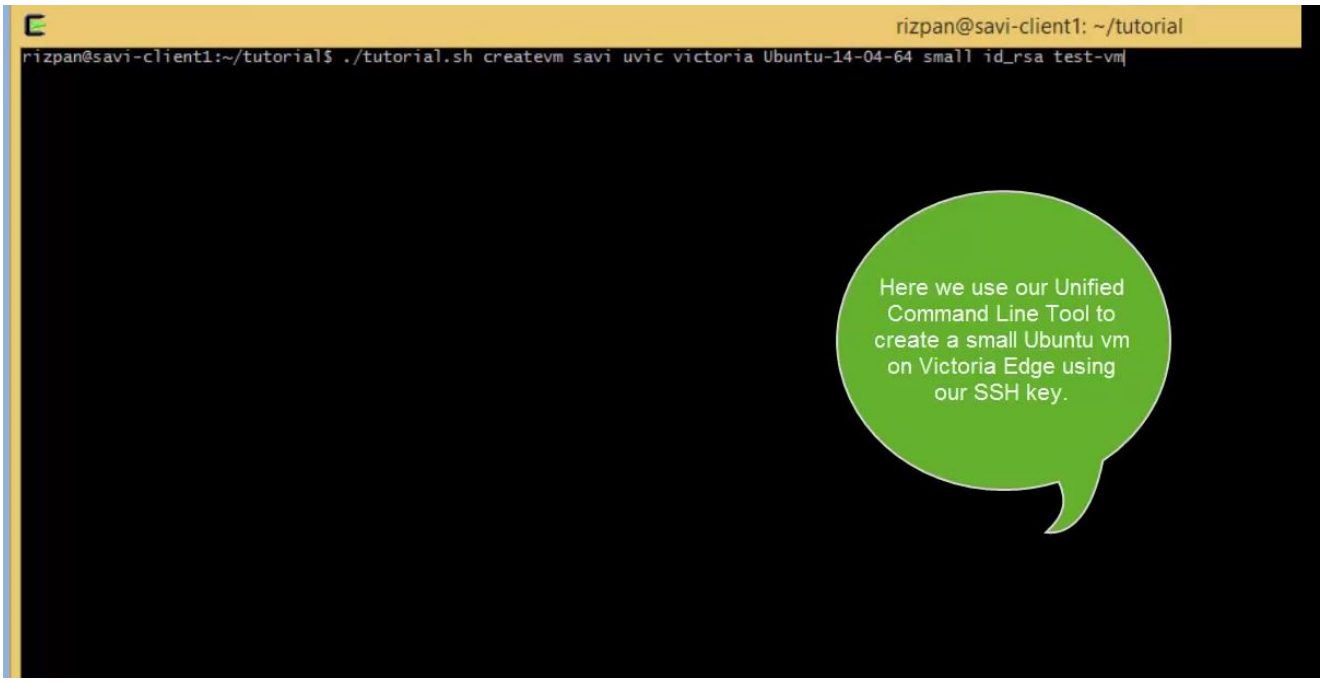

Figure A.17: GENI user accessing SAVI using federation - Step 3

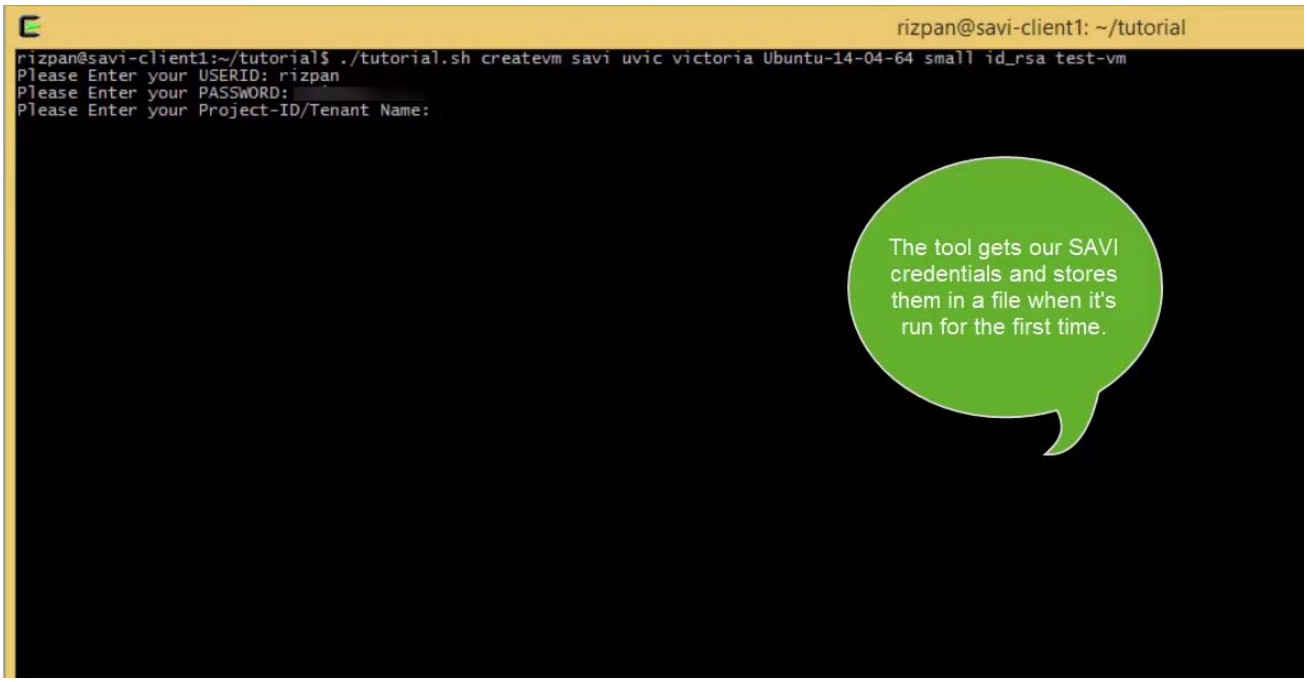

Figure A.18: GENI user accessing SAVI using federation - Step 4

A.4.2 SAVI User Accessing GENI Resources

In this tutorial we demonstrate how a SAVI user can create an account on GENI and access GENI resources. Please make sure you have a SAVI

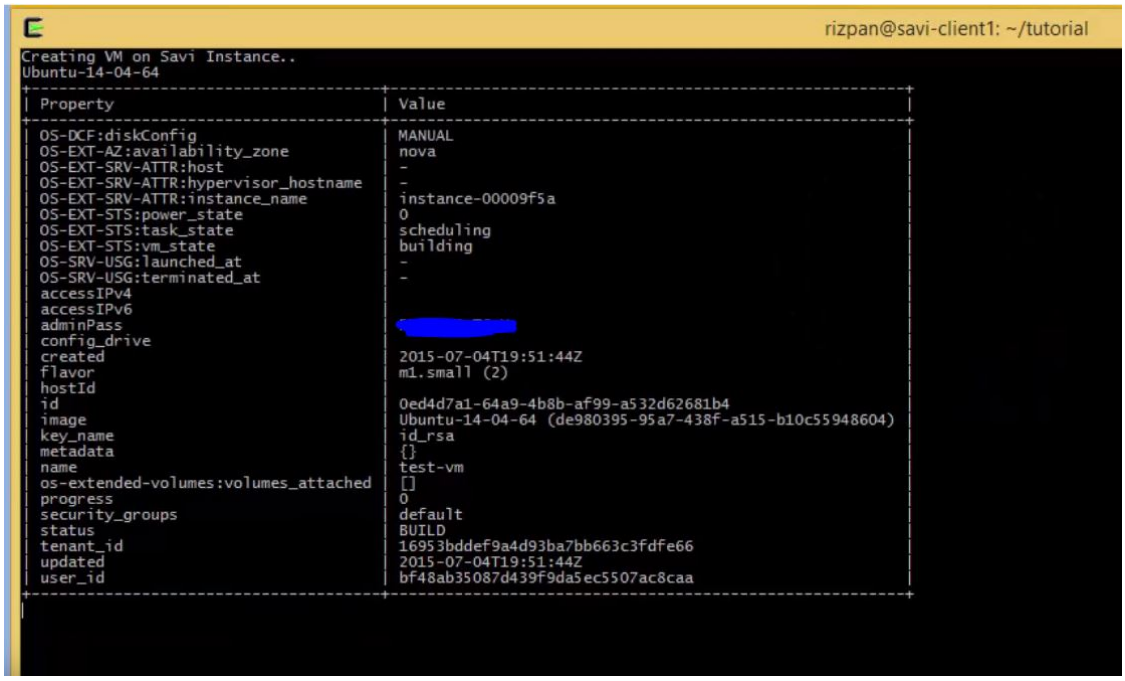

Figure A.19: GENI user accessing SAVI using federation - Step 5

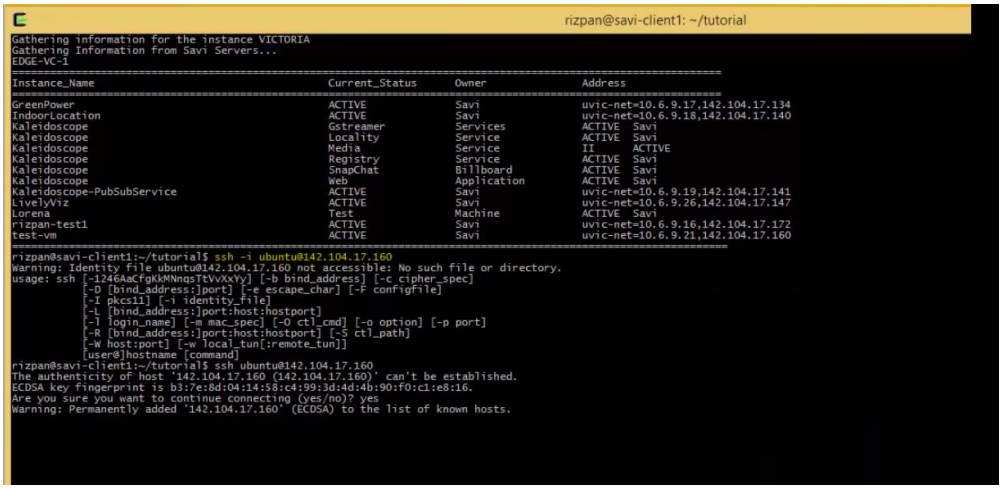

Figure A.20: GENI user accessing SAVI using federation - Step 6

account before starting this tutorial as detailed in A.3.

1. **Log into GENI, and click "Use GENI" button as shown in Figure A.21.**

2. **Click "SAVI" as shown in Figure A.22.**

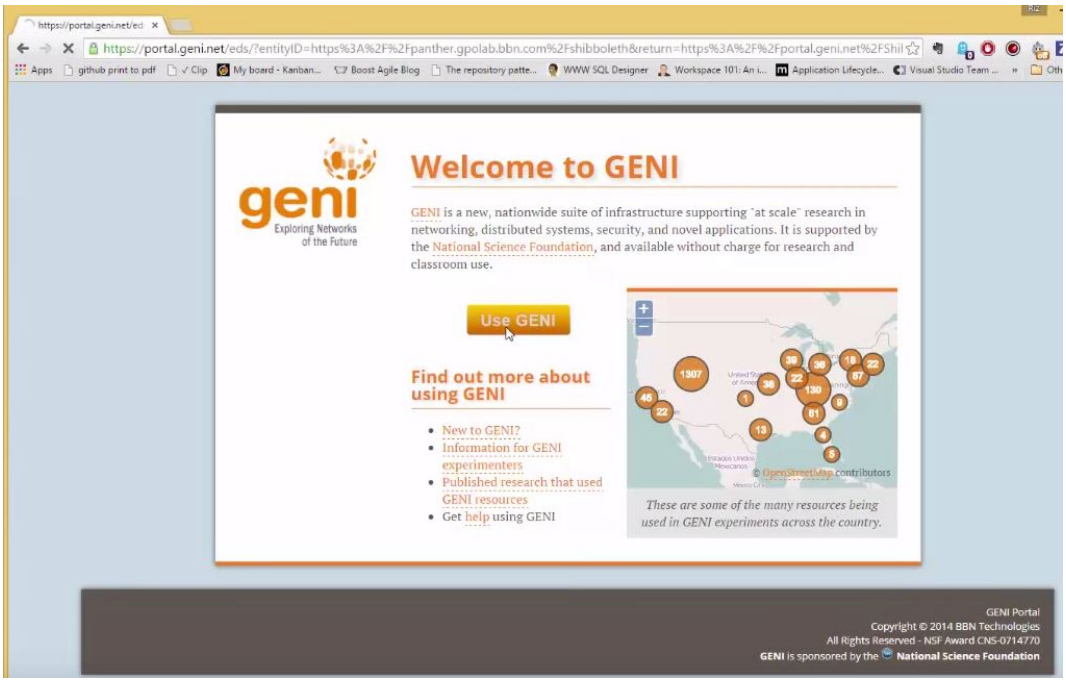

Figure A.21: SAVI user accessing GENI using federation - Step 1

Figure A.22: SAVI user accessing GENI using federation - Step 2

3. You will be redirected to the SAVI IdP service. Log in with your SAVI credentials as shown in Figure A.23.

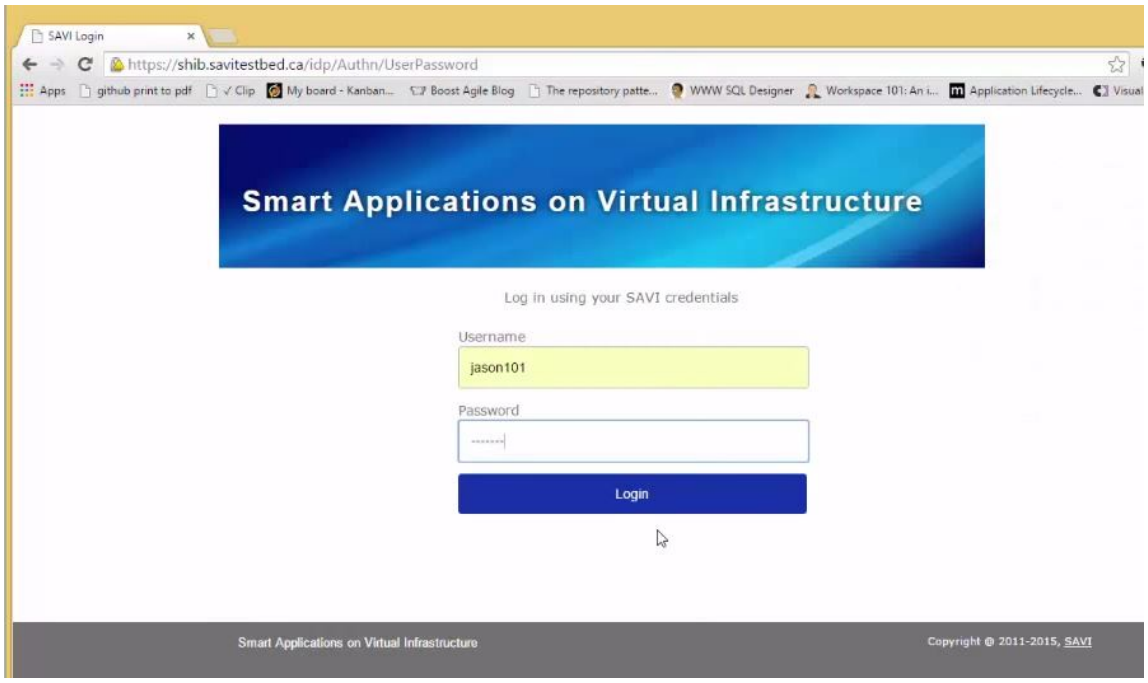

Figure A.23: SAVI user accessing GENI using federation - Step 3

4. **You will be redirected back to GENI Portal and logged in as shown in Figure A.24.**

5. **Click on your name in the top right, click "Profile", and click "Generate Public Key" as shown in Figure A.25. Once the key is generated, click on "Download OMNI Bundle". This will download the "omni.bundle" file onto your machine.**

6. **Configure OMNI as shown in Figure A.26.**

7. **Change into the "tutorial" directory which you downloaded before beginning this tutorial and type in the SAGEFed command to create a slice on GENI as shown in Figure A.27. Create a VM using the SAGEFed command to create a VM on the slice you just created as shown in Figure A.27.**

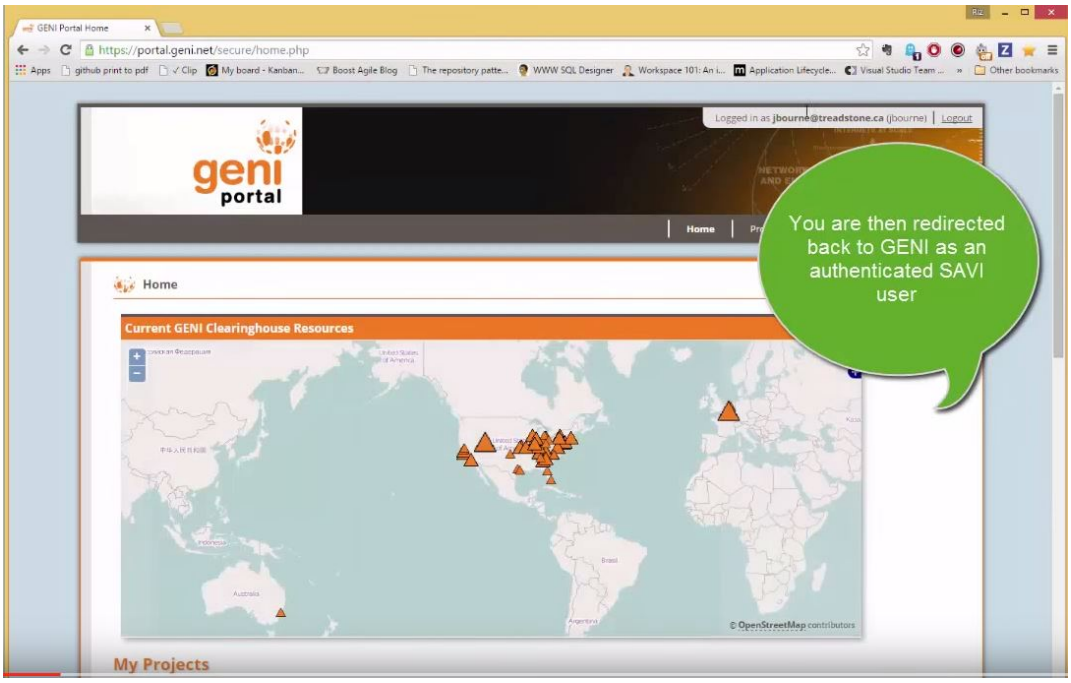

Figure A.24: SAVI user accessing GENI using federation - Step 4

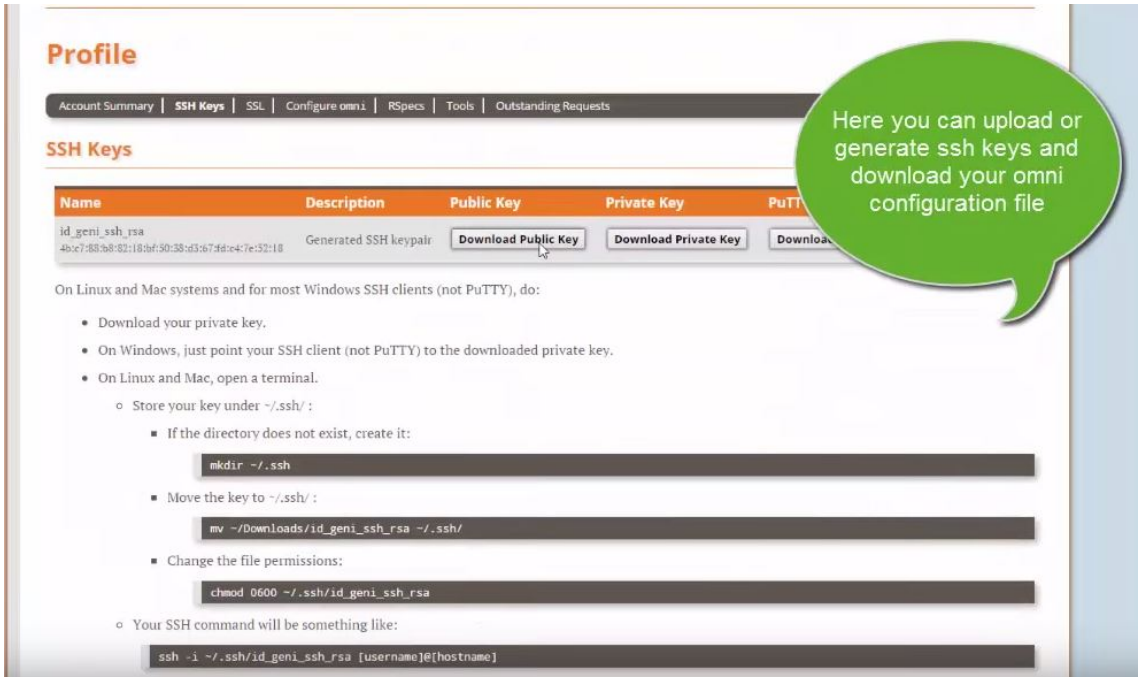

Figure A.25: SAVI user accessing GENI using federation - Step 5

8. Once the VM has been successfully created, you can SSH into the VM using its public IP address as shown in Figure A.28.

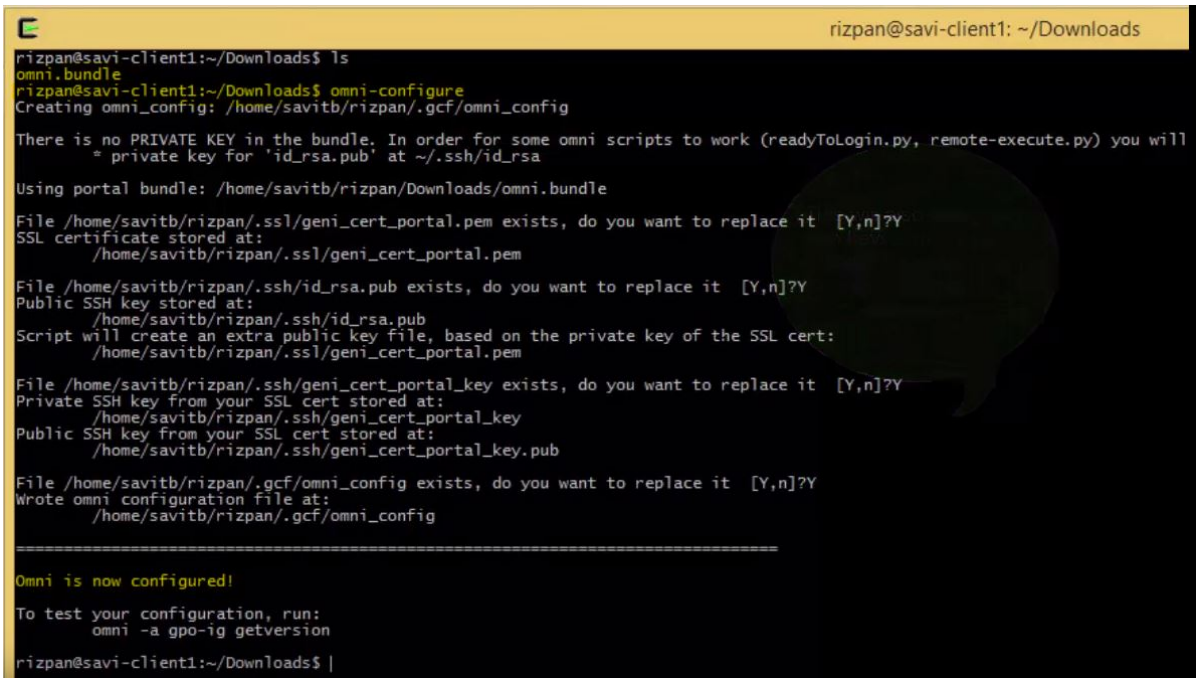

Figure A.26: SAVI user accessing GENI using federation - Step 6

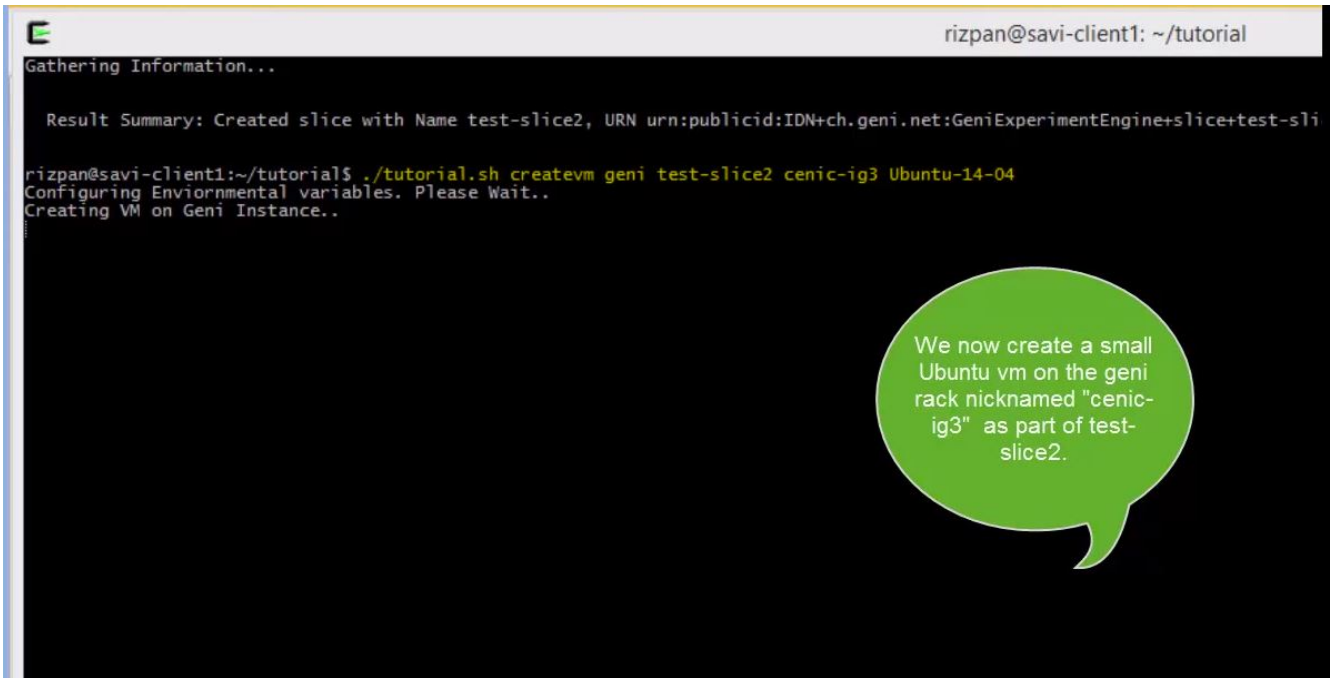

Figure A.27: SAVI user accessing GENI using federation - Step 7

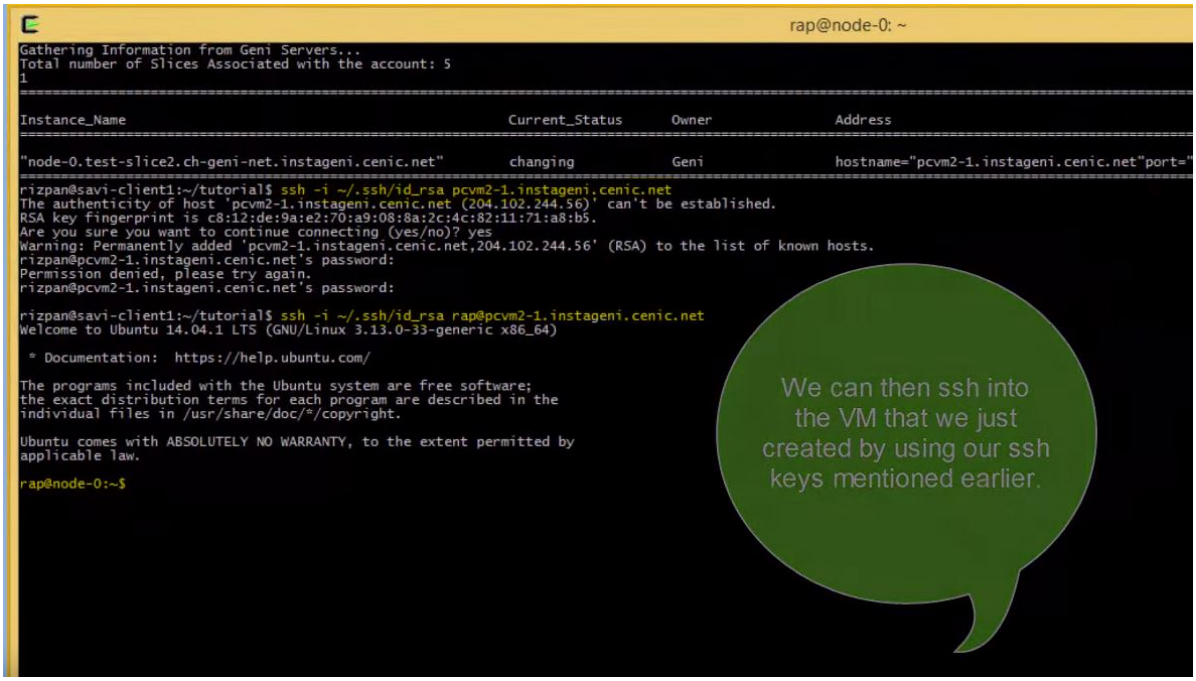

Figure A.28: SAVI user accessing GENI using federation - Step 8

Bibliography

[1] Rajiv Ranjan, Aaron Harwood, Rajkumar Buyya, et al. "Grid federation: An economy based, scalable distributed resource management system for large-scale resource coupling". In: *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia* (2004).

[2] C. Edwards and A. Harwood. "Using Para-Virtualization as the Basis for a Federated PlanetLab Architecture". In: *Virtualization Technology in Distributed Computing, 2006. VTDC 2006. First International Workshop on*. 2006, pp. 13–13. DOI: 10.1109/VTDC.2006.16.

[3] Li Tang et al. "Empirical Study on the Evolution of PlanetLab". In: *Networking, 2007. ICN '07. Sixth International Conference on*. 2007, pp. 64–64. DOI: 10.1109/ICN.2007.40.

[4] Barry M Leiner et al. "A brief history of the Internet". In: *ACM SIGCOMM Computer Communication Review* 39.5 (2009), pp. 22–31.

[5] B. Rochwerger et al. "The Reservoir model and architecture for open federated cloud computing". In: *IBM Journal of Research and Development* 53.4 (2009), 4:1–4:11. ISSN: 0018-8646. DOI: 10.1147/JRD.2009.5429058.

[6] Zehua Zhang and Xuejie Zhang. "Realization of open cloud computing federation based on mobile agent". In: *Intelligent Computing and Intelligent Systems,*

2009. ICIS 2009. IEEE International Conference on. Vol. 3. 2009, pp. 642–646. DOI: 10.1109/ICICISYS.2009.5358085.

[7] Daniele Catteddu. "Cloud Computing: benefits, risks and recommendations for information security". In: *Web Application Security*. Springer, 2010, pp. 17–17.

[8] A. Celesti et al. "How to Enhance Cloud Architectures to Enable Cross-Federation". In: *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. 2010, pp. 337–345. DOI: 10.1109/CLOUD.2010.46.

[9] A. Celesti et al. "Three-Phase Cross-Cloud Federation Model: The Cloud SSO Authentication". In: *Advances in Future Internet (AFIN), 2010 Second International Conference on*. 2010, pp. 94–101. DOI: 10.1109/AFIN.2010.23.

[10] Thierry Rakotoarivelo et al. "OMF: a control and management framework for networking testbeds". In: *ACM SIGOPS Operating Systems Review* 43.4 (2010), pp. 54–59.

[11] Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. "Security and privacy challenges in cloud computing environments". In: *IEEE Security & Privacy* 6 (2010), pp. 24–31.

[12] Tobias Kurze et al. "Cloud federation". In: *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2011)*. 2011.

[13] Ozalp Babaoglu, Moreno Marzolla, and Michele Tamburini. "Design and Implementation of a P2P Cloud System". In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC '12. Trento, Italy: ACM, 2012, pp. 412–417. ISBN: 978-1-4503-0857-1. DOI: 10.1145/2245276.2245357. URL: http://doi.acm.org.ezproxy.library.uvic.ca/10.1145/2245276.2245357.

[14] Nikos Makris et al. "Cross-Testbed Experimentation Using the Planetlab-NITOS Federation". English. In: *Testbeds and Research Infrastructure. Development of Networks and Communities*. Ed. by Thanasis Korakis, Michael Zink, and Maximilian Ott. Vol. 44. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2012, pp. 373–376. ISBN: 978-3-642-35575-2. DOI: 10.1007/978-3-642-35576-9_34. URL: http://dx.doi.org/10.1007/978-3-642-35576-9_34.

[15] SAVI Network. *SAVI Tutorial*. 2012. URL: https://www.youtube.com/watch?v=GSN9SEMtUDo (visited on 08/28/2015).

[16] Xun Xu. "From cloud computing to cloud manufacturing". In: *Robotics and computer-integrated manufacturing* 28.1 (2012), pp. 75–86.

[17] J.B. Abdo et al. "Broker-based Cross-Cloud Federation Manager". In: *Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for*. 2013, pp. 244–251. DOI: 10.1109/ICITST.2013.6750199.

[18] Joon-Myung Kang, Hadi Bannazadeh, and Alberto Leon-Garcia. "SAVI testbed: Control and management of converged virtual ICT resources". In: *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE. 2013, pp. 664–667.

[19] Bilal Charif and Ali Ismail Awad. "Business and Government Organizations Adoption of Cloud Computing". In: *Intelligent Data Engineering and Automated Learning–IDEAL 2014*. Springer, 2014, pp. 492–501.

[20] Mohammad Faraji et al. "Identity access management for Multi-tier cloud infrastructures". In: *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE. 2014, pp. 1–9.

[21] Joon-Myung Kang et al. "Software-defined infrastructure and the SAVI testbed". In: *Testbeds and Research Infrastructure: Development of Networks and Communities*. Springer, 2014, pp. 3–13.

[22] OpenGENI. *GENI Software Architecture*. 2014. URL: `http://www.opengeni.net/opengeni-software` (visited on 08/09/2015).

[23] OpenStack.org. *OpenStack Architecture*. 2014. URL: `http://docs.openstack.org/juno/install-guide/install/apt/content/ch_overview.html` (visited on 08/10/2015).

[24] OpenStack.org. *OpenStack Storage*. 2014. URL: `http://www.openstack.org/software/openstack-storage` (visited on 08/11/2015).

[25] Anatoly I Petrenko. "Service-Oriented Computing in a Cloud Computing Environment". In: *Computer Science* 1.6 (2014), pp. 349–358.

[26] Nabil Sultan. "Making use of cloud computing for healthcare provision: Opportunities and challenges". In: *International Journal of Information Management* 34.2 (2014), pp. 177–184.

[27] apache.org. *Apache Tomcat*. 2015. URL: `http://tomcat.apache.org/` (visited on 09/28/2015).

[28] apache.org. *Apache Web Server*. 2015. URL: `http://httpd.apache.org/` (visited on 09/28/2015).

[29] service architecture.com. *Web Container Technology*. 2015. URL: `http://www.service-architecture.com/articles/application-servers/j2ee_web_server_or_container.html` (visited on 09/28/2015).

[30] Danilo Ardagna. "Cloud and multi-cloud computing: current challenges and future applications". In: *Proceedings of the Seventh International Workshop on Principles of Engineering Service-Oriented and Cloud Systems*. IEEE Press. 2015, pp. 1–2.

[31] aws.amazon.com. *Amazon Web Services*. 2015. URL: http://aws.amazon.com/ (visited on 10/05/2015).

[32] azure.microsoft.com. *Microsoft Cloud Services*. 2015. URL: https://azure.microsoft.com (visited on 10/05/2015).

[33] Raytheon BBN. *OMNI API Client*. 2015. URL: http://trac.gpolab.bbn.com/gcf/wiki/Omni (visited on 08/24/2015).

[34] Sushil Bhojwani. "Interoperability in Federated Clouds". PhD thesis. University of Victoria, 2015.

[35] cloud.google.com. *Google Cloud Services*. 2015. URL: https://cloud.google.com/ (visited on 10/05/2015).

[36] Baojiang Cui and Tao Xi. "Security Analysis of Openstack Keystone". In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on*. 2015, pp. 283–288. DOI: 10.1109/IMIS.2015.44.

[37] eai.eu. *EAI Tridentcom 2015*. 2015. URL: http://blog.eai.eu/tridentcom-2015-a-must-for-experts-in-network-testbeds-and-research-infrastructures/ (visited on 08/27/2015).

[38] exogeni.net. *ExoGENI Software Architecture*. 2015. URL: https://wiki.exogeni.net/doku.php?id=public:software:start (visited on 08/27/2015).

[39] National Science Foundation. *National Science Foundation*. 2015. URL: http://nsf.gov/ (visited on 08/27/2015).

[40] geni.net. *About GENI*. 2015. URL: http://www.geni.net/?page_id=2 (visited on 08/11/2015).

[41] geni.net. *GENI Concepts*. 2015. URL: http://groups.geni.net/geni/wiki/GENIConcepts (visited on 08/27/2015).

[42] geni.net. *GENI Federation Guide*. 2015. URL: http://groups.geni.net/geni/wiki/InCommon/FederatingWithGENI (visited on 09/28/2015).

[43] geni.net. *GENI OpenId*. 2015. URL: http://groups.geni.net/geni/wiki/PortalOpenId (visited on 09/28/2015).

[44] geni.net. *GENI Portal*. 2015. URL: https://portal.geni.net/ (visited on 08/28/2015).

[45] geni.net. *GENI Rack Specifications*. 2015. URL: http://groups.geni.net/geni/wiki/GeniRacks (visited on 08/27/2015).

[46] globalsign.com. *SSL*. 2015. URL: https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/ (visited on 09/28/2015).

[47] internetlivestats.com. *Internet Live Stats*. 2015. URL: http://www.internetlivestats.com/internet-users/ (visited on 10/01/2015).

[48] linuxcommand.org. *Linux Shell*. 2015. URL: http://linuxcommand.org/ (visited on 09/30/2015).

[49] openflow.org. *Open Flow*. 2015. URL: http://archive.openflow.org/wp/learnmore/ (visited on 08/27/2015).

[50] opengeni.net. *OpenGENI Rack*. 2015. URL: http://www.opengeni.net/rack-overview/ (visited on 08/11/2015).

[51] openid.net. *OpenId*. 2015. URL: http://openid.net/developers/libraries/ (visited on 09/28/2015).

[52] openid.net. *OpenId is Obsolete*. 2015. URL: http://openid.net/developers/libraries/obsolete/ (visited on 09/28/2015).

[53] openldap.org. *Open LDAP*. 2015. URL: http://www.openldap.org/ (visited on 09/25/2015).

[54] *OpenStack*. 2015. URL: http://www.openstack.org/ (visited on 09/25/2015).

[55] openstack.org. *NOVA Client Shell*. 2015. URL: http://docs.openstack.org/developer/python-novaclient/shell.html (visited on 08/28/2015).

[56] openstack.org. *OpenStack Keystone Federation*. 2015. URL: http://docs.openstack.org/developer/keystone/configure_federation.html (visited on 09/28/2015).

[57] openstack.org. *OpenStack Tenant*. 2015. URL: http://docs.openstack.org/openstack-ops/content/projects_users.html (visited on 08/28/2015).

[58] openstack.org. *Python Its Dangerous, So Better Sign This*. 2015. URL: http://pythonhosted.org/itsdangerous/ (visited on 09/23/2015).

[59] openstack.org. *Shibboleth Setup Guide*. 2015. URL: http://www.cybera.ca/news-and-events/tech-radar/getting-started-on-shibboleth/ (visited on 09/23/2015).

[60] oracle.com. *Java*. 2015. URL: http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html (visited on 09/28/2015).

[61] oracle.com. *LDAP*. 2015. URL: http://docs.oracle.com/cd/A87860_01/doc/ois.817/a83729/adois09.htm (visited on 09/25/2015).

[62] oracle.com. *X509 Certificates*. 2015. URL: http://docs.oracle.com/javase/7/docs/api/java/security/cert/X509Certificate.html (visited on 09/26/2015).

[63] pocoo.org. *Python Flask Framework*. 2015. URL: http://flask.pocoo.org/ (visited on 09/26/2015).

[64] portal.savitestbed.ca. *SAVI Portal*. 2015. URL: http://portal.savitestbed.ca/ (visited on 09/05/2015).

[65] pythonhosted.org. *Python 512-bit Encryption Library*. 2015. URL: https://pythonhosted.org/passlib/lib/passlib.hash.sha512_crypt.html (visited on 09/26/2015).

[66] savinetwork.ca. *Accessing SAVI*. 2015. URL: http://www.savinetwork.ca/about-savi/request-access-to-savi-testbed/ (visited on 08/28/2015).

[67] savinetwork.ca. *SAVI AGM 2015*. 2015. URL: http://www.savinetwork.ca/wp-content/uploads/SAVI_AGM_poster_2015.pdf (visited on 08/28/2015).

[68] shibboleth.net. *Shibboleth*. 2015. URL: http://shibboleth.net (visited on 09/28/2015).

[69] tridentcom.org. *Tridentcom 2015 - GENI-SAVI Tutorial*. 2015. URL: http://tridentcom.org/2015/show/tutorials (visited on 08/27/2015).

[70] wikipedia.org. *Application Programming Interface (API)*. 2015. URL: https://en.wikipedia.org/wiki/Application_programming_interface (visited on 08/24/2015).

[71] wikipedia.org. *Cron Jobs*. 2015. URL: https://en.wikipedia.org/wiki/Cron (visited on 09/28/2015).

[72] wikipedia.org. *DNS*. 2015. URL: https://en.wikipedia.org/wiki/Domain_Name_System (visited on 09/28/2015).

[73] wikipedia.org. *PEM - Privacy Enhanced Mail*. 2015. URL: https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail (visited on 09/26/2015).

[74] Wikipedia.org. *Secure Shell*. 2015. URL: https://en.wikipedia.org/wiki/Secure_Shell (visited on 08/31/2015).

[75] URL: https://www.planet-lab.org/.

[76] URL: http://groups.geni.net/geni/wiki/GENIRacksHome.

[77] URL: http://groups.geni.net/geni/wiki/SignMeUp.

[78] URL: http://groups.geni.net/geni/wiki/GENIEducation/SampleAssignments.

[79] URL: http://groups.geni.net/geni/wiki/GEC15Agenda/IntroToOmni/Instructions.