The Experimenter's View of GENI

Niky Riga, Sarah Edwards, and Vicraj Thomas

Abstract GENI is a *federated* infrastructure that provides GENI experimenters with access to multiple different testbeds, enabling networking and distributed systems research. Although GENI resources are owned and operated by different organizations from a users perspective GENI appears as a unified virtual laboratory. An experimenter can instantiate custom Layer 2 topologies that include a variety of compute and network elements. This ability is achieved through the use of tools, as well as common APIs and shared authentication and authorization procedures across the federation.

GENI is a *federated* infrastructure that provides GENI experimenters with access to multiple different testbeds, enabling networking and distributed systems research. Although GENI resources are owned and operated by different organizations from a user's perspective GENI appears as a unified virtual laboratory. An experimenter can instantiate custom Layer 2 topologies that include a variety of compute and network elements. This is achieved by tools (see [5]) with the use of common APIs and shared authentication and authorization procedures across the federation.

In more detail, a GENI user can obtain compute resources from locations around the United States,[1] connect them using Layer 2 networks and can program every aspect of their topology including how traffic is routed within their network. It is important to note that all networking in GENI (wireless and wired) is done at Layer 2, allowing experimenters to run non-IP protocols.

All reservations in GENI are limited in time. When a reservation expires, resources are returned to the pool of available resources. The federated design of

[1]Through common APIs and policy agreements, GENI users can actually access resources from around the globe.

N. Riga (✉) • S. Edwards
GENI Project Office, Raytheon BBN Technologies, 10 Moulton St. Cambridge, MA 02138, USA

V. Thomas
GENI Project Office, Raytheon BBN Technologies, 5775, Wayzata Blvd., Suite 630, St. Louis Park, MN 55416, USA
e-mail: vthomas@bbn.com

R. McGeer et al. (eds.), *The GENI Book*, DOI 10.1007/978-3-319-33769-2_15

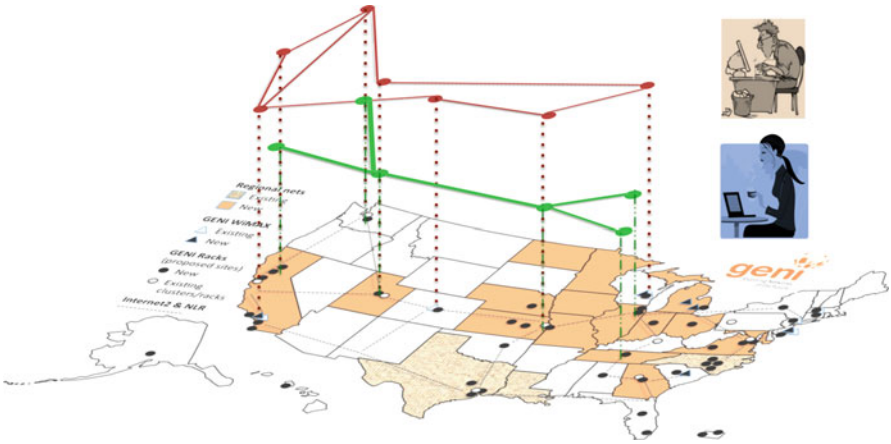

Fig. 1 Multiple GENI experiments run concurrently in isolated slices of the infrastructure

GENI makes it feasible to scale to a testbed that is much larger than one typically found in a laboratory. It provides the geographic diversity often needed in network research and the resource variety (from VMs to bare metal machines, from switches to WiMAX and LTE base stations) to make new configurations possible and to spur innovation.

Two of the major design principles in GENI that affect the interactions of users with the testbed are:

1. *Sliceability.* Each experiment is instantiated within a separate *slice* of the testbed, see Fig. 1. All slices are isolated from each other, i.e. the traffic of one experiment is not accessible or visible to an experiment running in another slice. This enables experiments that might be incompatible with each other to run concurrently on the same physical resources. For example one experimenter might be exploring the performance of a TCP variant that runs on top of IP, while a second experimenter might be investigating the feasibility of a new non-IP internet architecture in another slice. It is worth noting that several new internet architectures have been deployed and evaluated on GENI [1, 7, 15, 16, 26, 28], running concurrently in different slices. Although GENI does not provide any hard performance isolation guarantees, its architecture and resource slicing[2] provides best effort performance isolation between experiments. Sliceability not only enables different experimenters to use the testbed concurrently but also enables one user to run multiple experiments at the same time.

[2]Network slicing is done by VLAN with the appropriate bandwidth limits and there is no over-provisioning of network capacity. Some resource providers may over-provision compute resources by allocating more virtual machines on a server than available cores or memory. GENI also provides a limited number of bare metal machines that users can reserve in their experiments.

2. *Deep Programmability.* A user is allowed to program the behavior of as many elements in a slice as possible. This includes hosts at the edge of the topology (the user can choose an operating system and have full root access on the hosts to further customize them, including modifying the kernel as needed), as well as switching and computing elements in the core of the network. GENI has deployed programmable switches—mainly OpenFlow [19]—in the edge and core networks, as well as computation and storage in centrally located network exchange points, e.g. within a regional network. From the user's point of view, the slice includes a topology that can be programmed at different layers of the networking stack and allows for functionality (e.g. caching) to be placed in the middle of the network.

Accessing the GENI Testbed GENI is free for use in research and education. For users from many academic institutions accessing GENI is as simple as logging in any other service offered by their university. As described in [5], GENI has outsourced, when possible, the authentication of the users to their home institutions. This design choice not only makes user management much simpler for the federation, but also simplifies the user experience by allowing them to use their institution's credentials to login and use GENI. The single sign-on mechanism used in GENI is very similar to the prevalent practice in the web today of using well known identity providers such as Google or Facebook to access third party services. The technology used for single sign-on in GENI is Shibboleth [21]. For institutions that do not support this technology the GENI Project Office runs a Shibboleth Identity Provider to manage and authenticate users from these institutions.

1 Useful GENI Concepts

Before we delve into more details about how a user accesses GENI and instantiates experiments, we will go over some basic concepts and terminology.

1.1 GENI Resources and Resource Aggregates

Resource in GENI is used to describe elements that can be reserved by users and used in their experiments. Examples of resources include virtual machines (VMs), bare metal machines, storage, VLANs, OpenFlow datapaths, flowspace in OpenFlow-enabled devices, NetFPGAs, switches, sensors, monitoring cards and cameras. Resources can be contained within one physical device (e.g. VM) or distributed across a set of devices (e.g. VLAN), depending on the nature of the resource.

The following is a list of major GENI resource types. The elements in the list are not necessarily mutually exclusive i.e. a resource may belong to more than one type.

- *Compute resources*: Compute resources in GENI can be Virtual Machines, Linux containers or bare metal hosts. Depending on the requirements of an experiment the user can choose the resources that are most appropriate. For example, if performance is critical to the experiment, the user can reserve bare metal hosts. On the other hand if geographic diversity is more important then containers might be the right choice since they are much more widely available.
- *Wireless resources*: GENI Wireless sites have one or more WiMAX or LTE antennae that provide 4G coverage. Resources are sliced at Layer 2 by VLANs. For more information see [29]. Also each site has 2 or more wireless devices, usually referred to as *yellow nodes* that support regular *nix operating systems and can be reserved as bare metal nodes to run remote wireless experiments using WiMAX or WiFi interfaces. For users local to the sites there are 3G Android phones available for mobile experiments.
- *Storage resources*: Some sites also provide external storage that can be reserved and attached to an experiment, providing extra storage space when needed.
- *Network resources*: In addition to the wireless resources described above, GENI provides a variety of wired network resources that can be used to (1) connect resources from different locations in Layer 2 topologies and (2) allow the user to control forwarding within the network. Many of the network providers in GENI, including Internet2 [13] that provides the GENI backbone network, have deployed OpenFlow [19] switches and allow users to control the forwarding of their traffic as it traverses the network.
- *Unique resources*: The architecture GENI allows participants to connect unique resources into GENI and provide access to them to remote users. For example some of the compute nodes also have NetFPGAs or NPUs, that a user can program. This capability is not limited to networking resources and sites can connect diverse devices such as advanced microscopes and weather radars.

Resources are made available to experimenters by *Aggregate Managers*. An Aggregate Manager (AM) is a service that manages a collection (an aggregate) of physical devices in order to provide users with the requested resources. For example an AM can manage one or more VM servers providing VMs to users, it can manage a set of switches and provision links across them or manage a unique resource like a microscope. The AM is responsible for provisioning the resources, slicing shared devices ensuring isolation, providing remote access to the users when appropriate, reclaiming the resources for expired reservations, and enforcing local and global usage policies. An AM can manage any number of different devices from computation servers to switches to storage. The set of devices to be managed is an implementation and policy decision by the owners of the devices and the operator of the AM.

All AMs implement a GENI standard API called the AM API. This API is used by experimenter tools to learn about and reserve resources at the AM. See Sect. 1.2 for details.

Following are examples of some existing Aggregates that will clarify their role in the GENI ecosystem:

GENI Racks. These are the most widely deployed GENI resources. A GENI Rack consists of compute, storage and networking devices, all controlled by one or more Aggregate Managers. Details on the design and deployment of GENI Racks can be found in [3, 8, 18].

Network Providers. Several network providers that provide connectivity between GENI sites have deployed GENI-compliant Aggregate Managers for users to obtain and configure networking resources using the GENI AM API. Some characteristic examples are:

- *Internet2* and *MAX* that allow GENI users to dynamically configure Layer 2 circuits across their network.
- *SOX (Southern Crossroads), StarLight, CENIC, and MOXI regional networks* that allow GENI users to dynamically reserve flowspace on their OpenFlow switches and control the specified traffic with a user-defined OpenFlow controller.

Federated testbeds. Several existing testbeds have modified their resource managers to support the GENI AM API and thus allow users to reserve resources using the same tools they would use to resources GENI resources. Notable examples include PlanetLab [24] and Emulab [36].

1.2 GENI RSpecs and the GENI AM API

To allow interoperability among different Aggregate Managers(AMs) and a variety of tools, GENI has specified a standardized API called the GENI Aggregate Manager API (GENI AM API) [5]. The GENI AM API specifies the interactions between an AM and tools (which are a proxy for users). It defines methods to manage resource reservations (create one, expand the duration of the reservation, delete it), get status of reserved resources and discover resources offered by AMs.

Resources in GENI are described using a standardized language called *Resource Specification (RSpecs)* [27]. RSpecs are XML documents following an agreed upon schema to represent resources. The schema supports aggregate or resource specific extensions. As shown in Fig. 2, there are three different types of RSpecs:

1. **Advertisement RSpec.** Used by an AM to describe the resources it makes available to users.
2. **Request RSpec.** The document a user (usually a tool) uses to describe the resources to be reserved and how they should be configured including network topology.

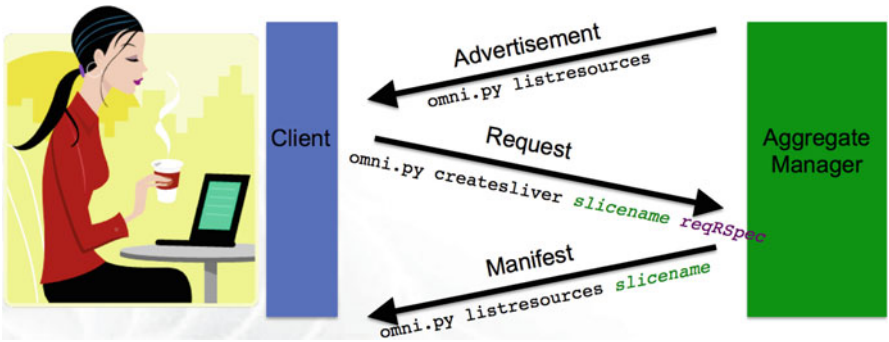

Fig. 2 There are three different types of RSpecs in GENI

3. **Manifest RSpec.** Describes the resources a user has already reserved at an AM.

Figure 2 shows the API calls that use RSpecs to communicate between the tools and the AMs. Since (1) all experimenter tools and AMs use the same API and RSpec schema, and (2) all information about resources is stored at the AMs, an experimenter may use different tools at different times for the same experiment. For example, the experimenter may reserve resources using tool A, check on their status using tool B and release them using tool C.

1.3 Slice

As mentioned earlier, one the major design pillars of GENI is *sliceability*, the ability to share the same infrastructure among multiple users and the ability to concurrently run multiple experiments. To achieve this GENI adopted, and expanded on the concept of a slice from PlanetLab [24] and the SFA architecture [23].

A GENI slice is:

- *A container for resources used in an experiment.* Users add GENI resources (compute resources, network links, etc.) to slices and run experiments that use these resources.
- *A unit of access control.* The experimenter that creates a slice can determine who has access to the slice i.e. are members of the slice.
- *The unit of isolation for experiments.* Resources assigned to the slice are dedicated to that slice and protected from access or interference from other slices, up to the capabilities of the specific virtualization technology used to slice each specific resource.

All slices in GENI are ephemeral, i.e. they have an expiration time. It is worth noting that although the resource reservations within a slice can not outlive the slice,

the expiration times can be different, i.e. a slice can (and usually does) outlive the resource reservations.

A slice can contain resources from any number of federated aggregates. Although slice is an abstraction, it is the concept that allows an experiment to span multiple administrative domains. Before starting an experiment, the user has to register a slice with a trusted Slice Authority.[3] Using this registered slice, the user can request resources from individual aggregates. In some sense, the aggregates trust and grant resources to slices.

From an operator's perspective, slices are the primary abstraction for accounting and accountability—resources are acquired and consumed by slices, and experiment behavior can be traced back to a slice.

1.4 GENI Projects

GENI is a shared, federated infrastructure that is used by experimenters around the globe at no cost. However, when running experiments in GENI, a user accesses and instruments real physical devices that are located within administrative domains usually not owned by the user or his institution. To address the accountability issues that arise in such a federated environment GENI expands on Emulab's [36] concept of a *Project*.

Projects organize research in GENI. Projects contain both people and their experiments. A project is led by a single responsible individual, known as the *Project Lead*. At the time of this writing, only academic faculty, senior technical staff and GENI Rack administrators can be Project Leads. The Project Lead takes responsibility for all experiments running within his projects and agrees to respond appropriately if a problem is discovered.

Any user who meets the requirements to be a Project Lead can request to be one. Leads can create GENI projects without the need for further approvals. Although only Project Leads can create projects, a lead can choose to have other individuals administer a specific project by making them project admins. Project admins have the same privileges within a project as the Project Lead, but they can not create new projects.

A project may or may not have an expiration time depending on the purpose of the project. A project that will be used by the students of a class is typically set to expire after the end of the semester. Research projects on the other hand tend not to have an expiration date.

[3]GENI's architecture supports multiple Slice Authorities. For example GENI currently has three Slice Authorities that can register slices used in the federation: The GENI Slice Authority operated by the GPO and the PlanetLab and Emulab Slice Authorities.

2 The GENI Experimenter Workflow

The GENI Experimenter workflow is a structured approach to running experiments on GENI. It serves as a framework for experimenters to be systematic with their experimentation. For GENI tool developers it serves as a framework for describing the steps of the workflow supported by their tools and their interfaces with other tools.

The GENI experimenter workflow is illustrated in Fig. 3. The major phases of running any experiment—Design/setup, Execute and Finish—are the three large areas of the figure, The ovals represent the steps in the workflow. The white boxes are experiment artifacts produced or consumed at each step.

Even though the workflow is depicted as a linear set of steps, in reality there will be loops in the workflow with the experimenter going back to an earlier step or skipping some steps altogether.

2.1 Design and Setup Experiment

Experiment Design In this step, experimenters lay out a detailed plan in advance of running their experiments. When experimenters come to GENI to run an

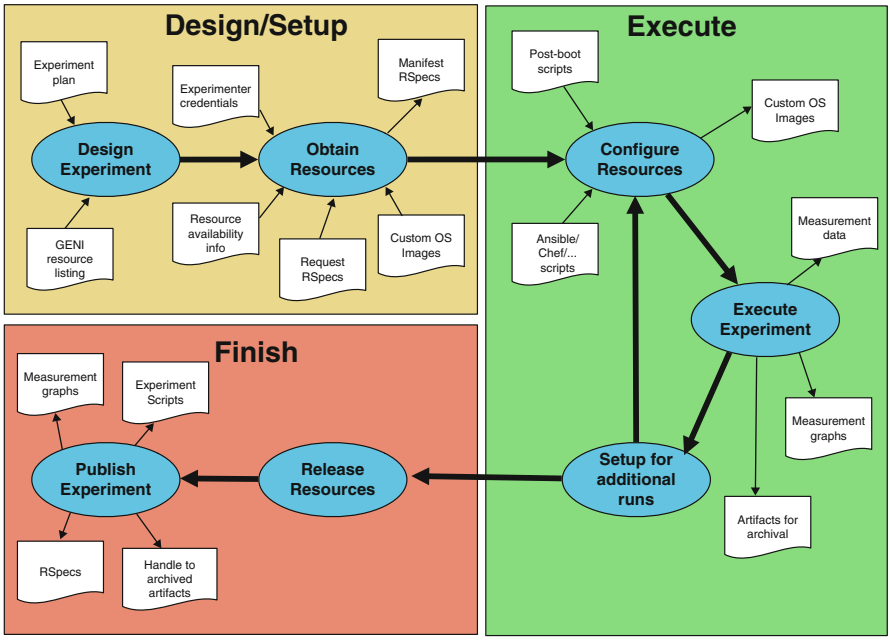

Fig. 3 Major steps in the GENI workflow and artifacts associated with each step

experiment, they typically have certain objectives in mind, the kind of experiment that will achieve the objectives and measurements needed to test if the objectives have been met.

Developing a detailed plan for an experiment on GENI includes deciding on:

1. The types, numbers and locations of resources needed. For example, experimenters must choose between virtual machines and raw-PCs for computation; configuration of these resources including memory, disk space, network interfaces and operating system; the location of the resources; the experiment topology and types of links; and the need for specialized resources such as hardware switches. Experimenters may consult the GENI web pages to find aggregates that have the resources they need.
2. How these resources will be programmed. They may choose to: (1) log into each resources separately and configure it, (2) write scripts that are automatically installed and executed when the resources come up, or (3) use custom OS images that have the needed software installed and possibly configured. Section 5 has a discussion on why experimenters should use scripts and custom images to make experiments repeatable and reproducible.
3. How the necessary network traffic will be generated and what will be measured. Options for network traffic generation include standard networking tools such as iperf and ping; more sophisticated traffic generation tools such as Tmix [35]; and the use of real user traffic (also know as opt-in user traffic). Experimenters may collect their own measurements or use GENI Instrumentation and Measurement tools such as LabWiki and GENI Desktop (Sect. 4.4).

Obtain Resources The next step in the experiment lifecycle is obtaining the resources needed for the experiments. Experimenters specify the resources they need including compute resources, network topology, operating systems to be loaded, bandwidth of network links, etc. by creating a Request RSpec. RSpecs are typically created using a tool such as Jacks Sect. 4.1. Figure 4 shows Jacks being used to create a Request RSpec for two virtual machines from two different aggregates.

Experimenters need to pick the aggregates where they want to reserve resources. This is typically a subset of the aggregates identified in the Experiment Design step of the workflow. They may pick aggregates based on factors such as availability and load. The GENI Meta-Operations Center (GMOC) maintains a calendar that shows scheduled and unscheduled maintenance events. The GENI monitoring service has information on resource utilization at each aggregate.

Experimenters submit their RSpecs to the selected aggregates using tools such as the GENI Portal and Omni. If their reservation is successful, they get a *Manifest RSpec*, an XML document with information needed to use the resources. For example, for VMs the manifest includes login information, OS installed, and MAC and IP addresses of the interfaces.

Since GENI is a shared testbed, resources have expiration times on them i.e. these resources are released when they expire. Experimenters can extend the expiration

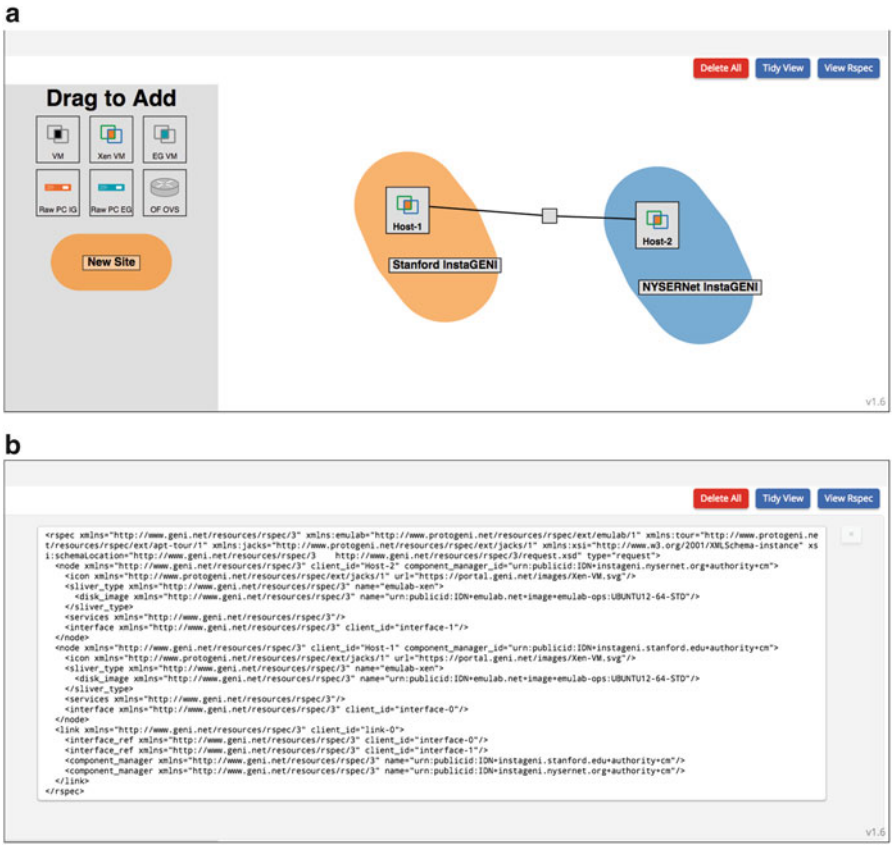

Fig. 4 Tools such as Jacks are commonly used to created request RSpecs. (**a**) Experiment topology drawn using Jacks. (**b**) Corresponding request RSpec generated by Jacks

time using their GENI resource reservation tools. Policies on default expiration times and the maximum duration by which a reservation can be extended at one time are set by the aggregate owners.

2.2 Execute Experiment

Configure Resources After resources have been obtained, experimenters configure them for their experiment. They may do this by installing software, modifying configuration files, changing settings on network interfaces, etc. Experimenters may automate the configuration of their resources by:

1. The use of install and execute scripts (also called post-boot scripts). These scripts, specified in the request RSpec, are installed and executed on the resources when they are setup by the aggregate.
2. Using system administration tools such as Ansible [2] and Chef [6].

After the resources are configured, experimenters may choose to create *custom images*, which are snapshots of operating systems they have configured. For future experiments they can specify these custom images as the operating system to be loaded on their resources. The operating system to be loaded is specified in the Request RSpec.

Execute Experiment Execution can be triggered manually by logging into each resource and starting up the experiment. Experiments can also be started up automatically using execute scripts.

GENI experiment orchestration tools such as LabWiki [17] and OMF/OML [22, 25] allow experimenters to describe and instrument an experiment, execute it and collect its results.

GENI tools for instrumenting experiments and collecting measurements are the GENI Desktop and LabWiki (Sect. 4.4). These tools allow experimenters to specify measurements to be collected, view graphs of these measurements and save the measurements.

Experiments may archive measurements and other experiment related artifacts such as RSpecs and scripts using a GENI-provided iRODS service Sect. 4.6.1. Items archived on this service survive the releasing of resources used in the experiment, the expiration of slice or project.

Setup for Additional Runs Experimenters may run the same experiment multiple times with the same or with different inputs, or resource configurations. The changes they make to the experiment before each run may be based on measurements gathered during an earlier execution.

2.3 Finish Experiment

Release Resource Since GENI is a shared testbed, experimenters are expected to release their use as soon as they are done. Experimenters can use any of the resource reservation tools to release resources. If resources are not explicitly released by the experimenter, they will automatically be released when they expire.

Publish Results The final step of the workflow is the publication of the results of the experiment. Experiment reprodicibility is a tenet of scientific research and GENI provides mechanisms for researchers to make the experiments reproducible. The artifacts required to reproduce the experiment may be archived on the GENI iRODS service and made accessible by other researchers. The RSpecs used for the experiment may be uploaded and shared on the GENI Portal. Any custom image used in the experiment can also be made public and available for others to use.

Section 5.1 describes how experimenters can make their experiments reproducible by others.

3 Case Study: GENI Cinema, Implementing an Advanced Service on GENI

This section is based on the GENI Cinema Architecture document [14] written by Ryan Izard, Parmesh Ramanathan and KC Wang.

GENI Cinema is a persistent live video streaming service that capitalizes on the advanced capabilities of GENI. It allows any user (organization or individual) with access to the GENI network to publish a live video stream through this service. The users can also search, select and watch video streams. Being a geographically-distributed testbed, the GENI infrastructure provides an ideal platform to implement a content delivery network for efficient broadcasting of video content to customers at the edges. Combined with Software Defined Networking (SDN), this allows both network and compute resources to be conserved while users from different areas choose between the available video "channels" hosted by GENI Cinema.

In this section we describe the deployment story of GENI Cinema from an idea to a prototype service and highlight some of the design choices made. GENI Cinema was developed by teams of researchers at the University of Wisconsin (Principal Investigator Parmesh Ramanathan) and at Clemson University (Principal Investigator Kuang-Ching Wang).

3.1 Designing GENI Cinema

Designing such a complicated service is an iterative process, where the design is constantly being improved as the service is developed and deployed.

GENI Cinema consists of two main subsystems: one addressing end-to-end video/audio stream handling and the other addressing optimal forwarding in the network. Both subsystems heavily leverage GENI SDN capabilities. Each subsystem was developed separately by each of the universities on the project. Each group ran single site experiments and updated its design to optimize for performance. Once both systems were fully developed they were integrated into one system. Figure 5 shows the combined architecture that consists of the many functional blocks that comprise the GENI Cinema service. There are ingress and egress gateways for receiving and sending video streams, ingress and egress VideoLAN Client (VLC) servers for hosting video streams on the backend and providing them on the frontend, a global OpenFlow controller, hardware and software OpenFlow switches for controlling the flow of video streams internal to the GENI Cinema service, and a web server for customer interaction.

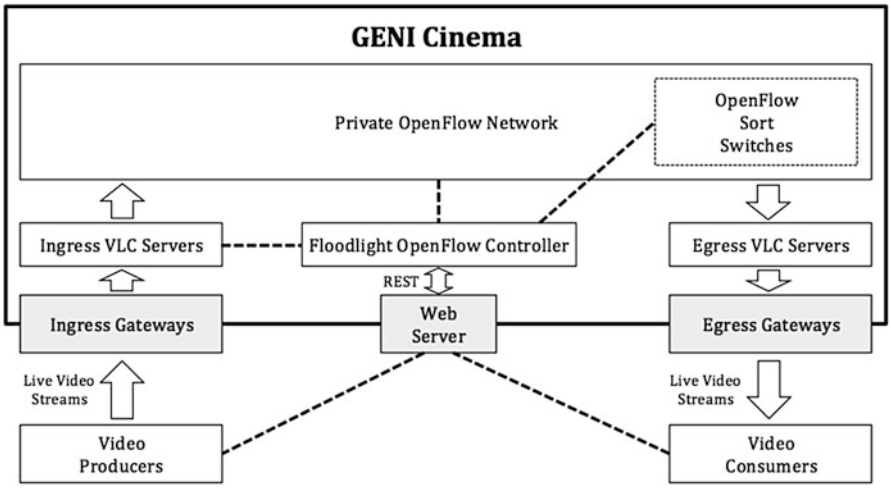

Fig. 5 Logical components of the GENI Cinema architecture

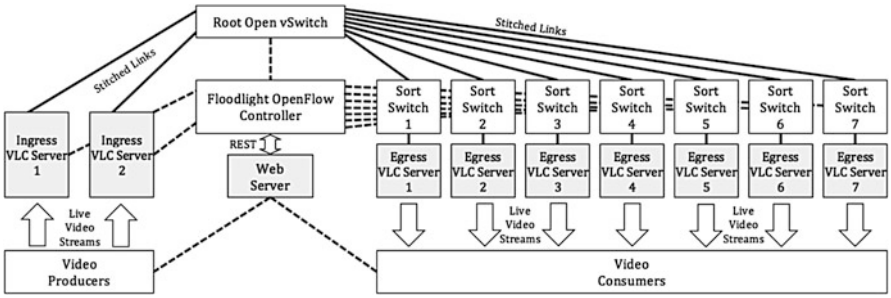

Fig. 6 The GENI Cinema SDN architecture. Note that each stitched link also contains two physical OpenFlow switches under the control of a floodlight controller—one at each end of the link

3.2 Use of Software Defined Networking

The GENI Cinema implementation heavily relies on the software defined capabilities of the GENI network and in particular on the deployment of OpenFlow switches.

All video traffic output from the ingress VLC servers into the private GENI Cinema network is unicast UDP in order to allow fast video stream switching without regard for connection state, sequence, or source as TCP would impose. Each UDP video stream is directed through the network toward all egress VLC servers where there is at least one video consumer wishing to watch that particular stream. Prior to each egress VLC server is an OpenFlow switch called a "sort switch", as depicted in Fig. 6.

Each sort switch is responsible for taking the UDP video streams supplied as input on the northbound interface, duplicating these streams if appropriate, and sending them to the VLC instances on the associated egress VLC server. This involves rewriting the destination MAC, IP, and/or UDP port numbers in order for the network stack on the egress VLC server to accept the packets and send them to the VLC instances running as applications, which is enabled by OpenFlow.

GENI Cinema reduces duplicate transmissions of video streams until the last hop at the egress point where the consumers are connected. For example, if there are 100 video consumers on a particular egress VLC server and all 100 video consumers wish to watch the same video stream, a single stream will be sent by the private GENI Cinema network to the sort switch, using 1 Mbps bandwidth. This single stream will be made into 100 copies where each copy's destination headers are rewritten such that the packets are routed to the VLC instance of each video consumer on the associated and nearby egress VLC server. This means 1 Mbps of traffic enters the sort switch and 100 Mbps exits. On the other hand, if there are 100 video consumers that collectively select all 20 of 20 available channels, then each channel's stream enters the sort switch for a total of 20 Mbps. The sort switch will make copies of each stream and rewrite the destination headers of each stream to send it to the VLC instance of the video consumer that wishes to watch that particular stream. After duplication, the total exit traffic is still 100 Mbps leaving the sort switch. The exit traffic is directly proportional to the number of video consumers presently attached to that particular egress gateway. The traffic entering can be no more than the total number of video channels available or the number of consumers at the egress point—whichever is less. Note that if there is no consumer watching a particular video stream at an egress point, this stream is not sent to the sort switch.

As described, when a video consumer selects a channel to watch, the sort switch is responsible for selecting the appropriate input stream. OpenFlow 1.1+ groups and buckets are used at the sort switch to implement this channel changing feature (Fig. 7). Every video is classified as an OpenFlow group, and every video consumer has a single OpenFlow bucket. An OpenFlow bucket is a list of actions, which in this case each action list rewrites the destination MAC, IP, and/or UDP port in the headers of the packets. If a video consumer switches video channels, its bucket is removed from the previous video group of the channel it was viewing, and the bucket is simply added to the group of the new video channel. In this way, only one connection and video stream per consumer is ever present at a given time within the private GENI Cinema network, and no connection is set up or torn down upon a channel change. This optimizes the bandwidth usage, as well as reduces the load on the server resources during frequent channel changes.

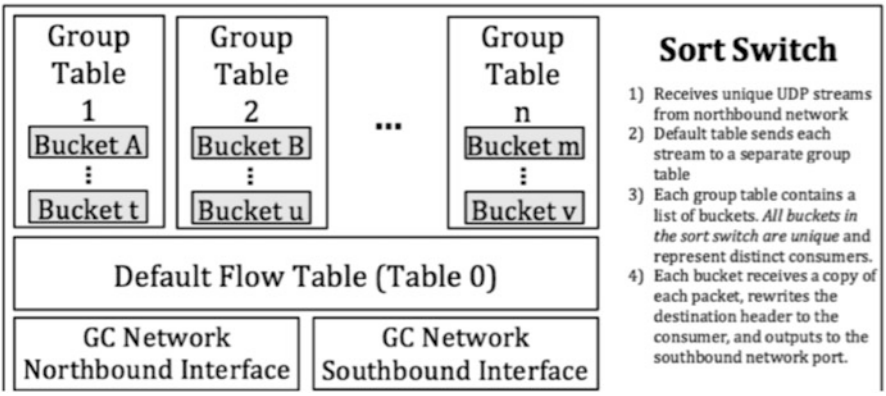

Fig. 7 The use of OpenFlow groups in the sort switch

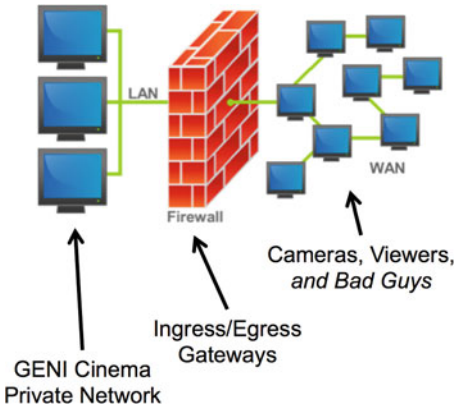

Fig. 8 The egress/ingress gateways can also serve as firewalls to the GENI Cinema private network

3.3 Deploying GENI Cinema

GENI Cinema started with a couple of single site deployments, one at Clemson University and the other at the University of Wisconsin. Originally the team broadcast local classes while debugging and enhancing the system.

While running in a single site, the team also developed the ingress and egress gateways that not only bypass any local issues due to NATing but can also secure the GENI Cinema system by acting as a firewall (Fig. 8).

Once GENI Cinema was stable, the deployment expanded to multiple sites. The first multi-site deployment was to connect the two prototype systems, the one at Clemson and the one at Wisconsin. After the two-site system was operational the team started working on a multi-site deployment. The first step was to enhance the system architecture to clearly identify which systems needed to be deployed on each site, how they are connected and how they interact with the rest of the system, i.e. they designed a distributed version of their system. Each new site can be an ingress

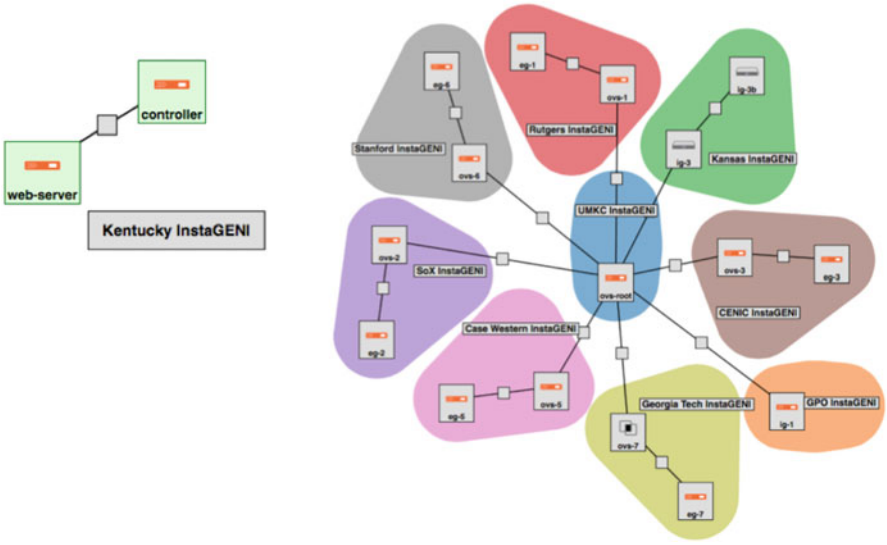

Fig. 9 Current GENI Cinema deployment

site (where new video streams are connected), an egress site (where new consumers are connected) or both. One or more sites are chosen to run central programmable switches responsible for routing the video streams from producers to consumers.

Currently the prototype deployment spans nine sites (Fig. 9). The deployment of new sites is completely automated and they can add new sites on demand. This helps them manage occasional unavailability of sites due to failures or maintenance events.

3.4 Connecting Users to GENI Cinema

GENI Cinema is open to users without GENI accounts. Connecting users to the GENI Cinema network is challenging, since the deployment lives within GENI. While the deployment was within the Clemson and Wisconsin Universities, the labs of the researchers were connected to the GENI deployment through their local GENI Rack. Classrooms in GENI-enabled campuses can be connected in a similar way, by expanding connectivity to the GENI network through the local GENI Rack. However, users should be able to access GENI Cinema from anywhere. To achieve this goal, users (producers or consumers) connect to the GENI Cinema service through the egress and ingress gateways using the public facing interface of these gateways. To avoid overloading the public interface, the users are load balanced across multiple gateways.

The workflow for video publishers and consumers is as follows:

- A producer wishing to publish a video on GENI Cinema makes a request on the GENI Cinema web service. The request is relayed to the OpenFlow controller, which responds with an ingress gateway IP address and transport port number. The producer connects and sends the video stream to the assigned address and port. The incoming video stream is relayed to an ingress VLC server where the live stream is hosted.
- When a consumer requests a video stream on the GENI Cinema web service, the request is relayed to the OpenFlow controller, which responds with an appropriate egress gateway IP address and port number where the consumer can connect and watch the video. The video selected is routed, duplicated and rewritten within the private network of GENI Cinema from the ingress VLC server on which it is being hosted to the private interface of the egress VLC server where the customer is connected. A VLC instance on this egress VLC server outputs the video on the public interface and relays it to the video consumer.

4 Experimenter Tools

The experimenters' main interface to GENI are the *experimenter tools* that serve to support the experimenter workflow (Sect. 2). Some tools support the experiment design/setup parts of the workflow by helping create Request RSpecs. Other tools support experiment execution by helping with installing and configuring software, orchestration (the automation and scheduling of the steps in the experiment), and instrumentation and measurement (the taking of and collection of data related to or produced by the experiment). Finally, other tools support the archiving and sharing of experiment results.

4.1 RSpec Creation Tools

Jacks, *Flack* and *jFed* are all graphical user interface (GUIs) tools for creating and editing Request RSpecs. They are used to define topologies and set properties of nodes and the links that connect them. Node properties include node name, OS and scripts to be installed and executed at boot time. Link properties include link type (VLAN or GRE tunnel) and IP addresses of end-points and others.

Flack and jFed can also be used as resource reservation tools (Sect. 4.2); they can be used to submit RSpecs to specified aggregates using the GENI AM API. While Jacks does not do resource reservations, the RSpecs it generates can be exported for use with other tools.

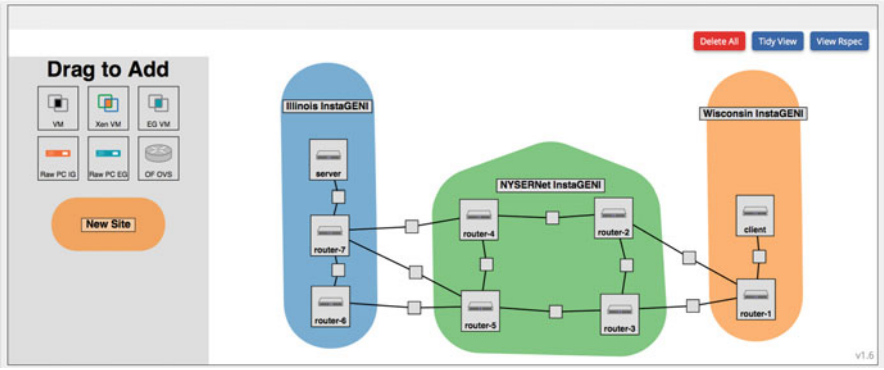

Fig. 10 Jacks GUI showing a topology spanning three aggregates

4.1.1 Jacks and Flack

Jacks and Flack are created by the Flux Research Group at the University of Utah. Both are browser-based: Jacks is written in HTML5 and Flack in Flash. Flack is no longer maintained.

Jacks is primarily an RSpec creation and viewing tool and is usually embedded in another tool such as the GENI Portal or GENI Desktop. A unique feature of Jacks is its constraint system that prevents experimenters from creating invalid RSpecs. For example, it will warn experimenters if they try to create a Layer 2 link between sites that do not support it or load an OS image in an incompatible compute resource (Fig. 10).

4.1.2 jFed

The jFed tool [12] is created by the iMinds Research Institute in Belgium. It is a Java application that runs on the experimenter's workstation. It can be used to create and view RSpecs, make resource reservations, launch ssh clients to log into nodes and do some experiment orchestration (Fig. 11).

4.1.3 geni-lib

geni-lib [4] is a python library from Barnstormer Softworks. It provides an object oriented scripting interface to both the AM API and GENI RSpecs. The purpose of geni-lib is to allow developers to build custom GENI tools. This is particularly helpful for advanced GENI experimenters. An example is the *scaleup* tool distributed with geni-lib which allows experimenters to write small topologies using standard node types (e.g. a topology might consist of multiple client, server,

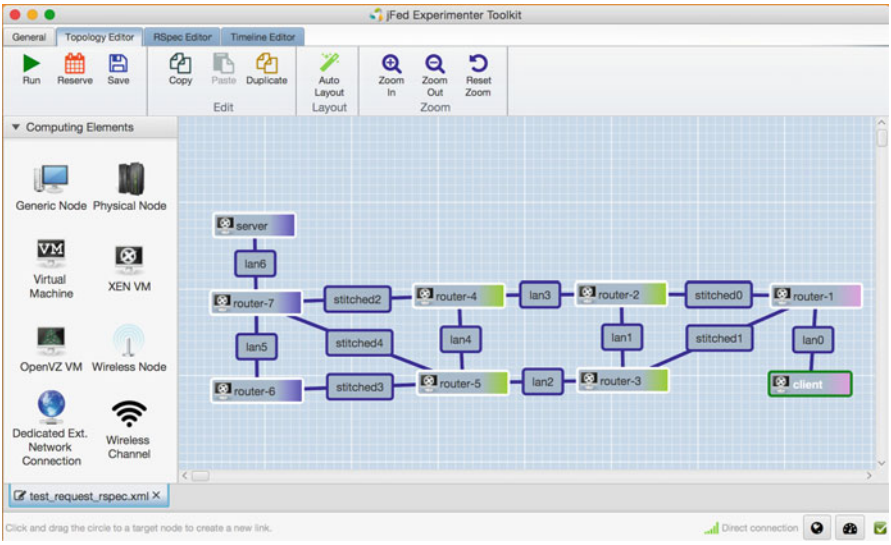

Fig. 11 jFed GUI showing a topology spanning three aggregates

```
>>> ad = IGAM.MAX.listresources(context)
>>> for node in ad.nodes:
...   if node.available and IGUtil.shared_xen(node):
...     print node.component_id
...
urn:publicid:IDN+instageni.maxgigapop.net+node+pc3
urn:publicid:IDN+instageni.maxgigapop.net+node+pc1
urn:publicid:IDN+instageni.maxgigapop.net+node+pc2
```

Fig. 12 Using *geni-lib* to list all available Xen servers at an InstaGENI rack

and router nodes) which can then be easily scaled up to a larger number of nodes in a wide range of topologies (e.g. ring, grid, random) (Fig. 12).

4.2 Resource Reservation Tools

The *Omni*, *geni-lib*, the *GENI Portal*, *jFed*, and *Flack* allow experimenters to communicate with resource providers (i.e. aggregates) using the GENI AM API (Sect. 1.2).

These tools allow experimenters to determine the resources advertised by an aggregate (i.e. to request an advertisement RSpec), to reserve resources in a slice (i.e. to submit a request RSpec), and to determine the resources reserved at an aggregate in a particular slice (i.e. to retrieve a manifest RSpec).

4.2.1 Omni

Omni [11] is a command line tool that can be used to invoke any AM API method on a GENI aggregate. It was developed by the GENI Project Office. Benefits of Omni include:

1. Omni is usually the first tool to make new AM API versions or functionality available to experimenters. This is because it originated as a developer tool and is still used to test new AM API functions.
2. Omni works well with aggregates that use atypical or novel RSpec extensions and features. This is because it does very little parsing of the RSpecs.
3. Omni is a command line tool and can be used in shell scripts and/or over poor Internet Connections.
4. Omni is written in Python and can be used by other Python scripts. Examples of commonly used tools that take advantage of this are *Stitcher* and *readyToLogin*. The Stitcher tool is used for dynamically connecting compute resources on different aggregates using VLANs. ReadyToLogin is used to determine the status of reserved resources and to get information needed to log into those resources. Additionally, tools such as the GENI Portal and GENI Desktop use Omni behind the scenes to make AM API calls.

The downside to Omni is that much of the burden of manipulating RSpecs (generating Request RSpecs, parsing Advertisement and Manifest RSpecs) falls on experimenters. Of course, experimenters can use other tools for RSpec manipulation and use Omni for resource management.

4.2.2 The GENI Portal

The GENI Portal [10] is probably the most widely used of GENI experimenter tools because it is the only tool for account and project management. It is a web-based tool that requires no software installation on the experimenters' computers, it supports much of the experimenter workflow and it serves as an identify provider for other tools and services. The GENI Portal was developed by the GENI Project Office.

The GENI Portal can be used for account management (requesting accounts, requesting Project Lead status), project and slice management (creating projects and slices, adding and removing users from projects and slices), resource management (reserving and deleting resources, extending resource reservations) and sharing of RSpecs. The GENI Portal embeds the Jacks tool for creating and viewing RSpecs.

The GENI Portal also serves as an OpenID identity provider for tools, services and testbeds hosted by other organizations. Experimenters log into the Portal and then click from the Portal to access these tools and services without having to separately log into those tools. Some examples of tools and services that are accessible from the portal inlcude JFed, GENI Desktop, the Canadiatn SAVI testned, GENI wireless, CloudLab.

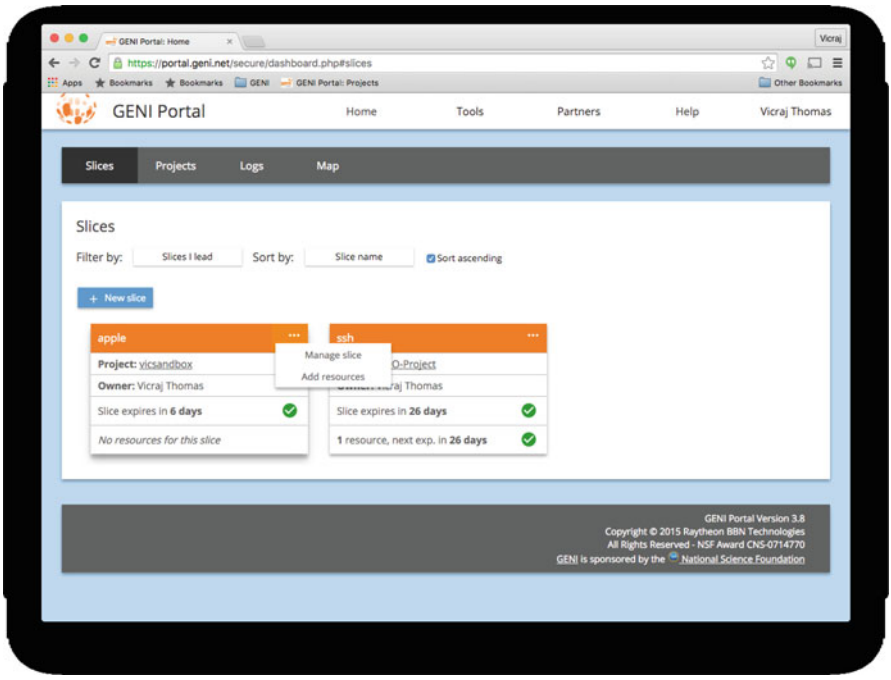

Fig. 13 Slice dashboard view of the GENI portal

Figure 13 shows the slice dashboard view for a user. In this figure the user has filtered the slices he has access, to only view the ones he leads. He can manage a slice or add resources to the slice by clicking on the dots by the name of the slice.

4.3 Experiment Orchestration and Scripting Tools

Experiment orchestration allows experimenters to automate or script their experiment procedure: start/stop data collection, start/stop traffic, schedule network events, etc. As such, orchestration is critical to the repeatability of experiments by allowing an experimenter to do multiple runs of the same procedure and to vary parameters as necessary.

While trivial procedures can be orchestrated with simple scripts (for example install scripts), GENI supports more complicated procedures using *OEDL* which is language to script and instrument data collection.

4.3.1 OEDL

OEDL is a domain-specific language for the description of an experiment's execution [33]. It is based on the Ruby language with domain specific extensions for experiment-oriented commands and statements. An OEDL script consists of two main parts: (1) A part where resources used in the experiment and their configurations are declared, and (2) a part where events are defined along with tasks to be executed when those events occur. An *experiment controller* interprets OEDL scripts to orchestrate experiments. The LabWiki tool (Sect. 4.4.2) uses OEDL as its scripting language.

4.4 Instrumentation and Measurement Tools

Measurement is a key to scientific experimentation and to this end GENI provides experimenters with a couple of Instrumentation and Measurement (I&M) tools: GENI Desktop/GEMINI and LabWiki/GIMI. Both tools allow experimenters to specify the measurements to be collected, and to graph, view and archive measurements.

4.4.1 GENI Desktop

GENI Desktop [34] is a web-based experimenter tool that, like the GENI Portal, can be used to create projects and slices, create Request RSpecs using the embedded Jacks tool, and manage resources. It was developed by the University of Kentucky.

A key feature of the GENI Desktop is the ability to *instrument* a slice to collect and view live measurements. It includes a number of pre-defined measurements such as CPU load on the hosts and number of packets sent/received on a network interface. Experimenters may also provide scripts to collect and view their own custom measurements. To select pre-defined measurements, the experimenter simply clicks on a host or link in the "Topology View" of the GENI Desktop and then selects the measurements of interest. Figure 14 shows the Topology View of an experiment and a graph of traffic on one of the interfaces attached to the link in the experiment.

4.4.2 LabWiki

LabWiki [17] is a web-based tool to design, describe and run GENI experiments. It was developed by NICTA, Australia's Information Communications Technology Research Centre of Excellence and the University of Massachusetts at Amherst. LabWiki is designed to help experimenters develop experiments that are repeatable and reproducible. LabWiki includes a panel where experimenters write experiment scripts using the OEDL scripting language (Sect. 4.3.1), a second panel for running and viewing graphs, and a third panel for recording notes and saving experiment

Fig. 14 GENI desktop showing graph of traffic on a network link

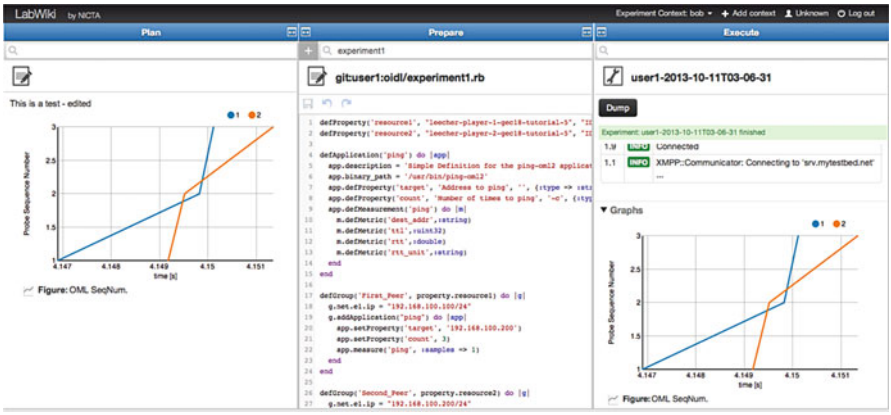

Fig. 15 Scripting and running an experiment using the LabWiki tool

results including graphs. Experiment scripts can be shared with other LabWiki users wishing to reproduce or extend the experiment.

Figure 15 shows the three panels of the LabWiki tool: The panel labeled "Prepare" is used to write or load experiment scripts, the "Execute" panel is where experimenters drop scripts to be executed and view graphs of experiment data and the "Plan" panel is used for notes, observations and saving graphs from the Execute panel.

4.5 Software Installation and Resource Configuration

Almost any experiment in GENI involves installing software or configuring compute resources. Automating this process helps experimenters easily and quickly repeat

experiments or share them with others. It also makes large-scale experiments feasible as experimenters do not have to log into each resource to configure it.

Three mechanisms are widely used in GENI to automate software installation and configuration of compute resources: (1) *Install and execute scripts* specified in the Request RSpecs, (2) *Custom OS images* with the desired configuration or software installed and (3) *Configuration management tools* such as Chef [6] and Ansible [2].

4.5.1 Install and Execute Scripts

Install and execute scripts (also called *postboot scripts*), are listed in the Request RSpec as part of the specification for a compute resource. Install scripts are bundled in tarballs (.tgz files) posted on a publicly accessible web server. They are downloaded and installed on compute nodes by the Aggregate Manager when the resources are provisioned. Install scrips are typically executable scripts but any type of data can be bundled in tarballs and installed on the nodes. ExoGENI aggregates support templating in the scripts so they can be customized based on attributes such as slice name, hostname and node type.

Execute commands specified in the RSpec are run in the compute resource; they may be used to configure the resource or run the installed scripts. Multiple install and execute scripts may be specified in the RSpec; the order of installation of the scripts and execution of the commands is not specified though all installations will be completed before any commands are executed.

Install and execute scripts can also be used as a primitive means to orchestrate experiments by scripting actions such as starting traffic or data collection.

4.5.2 Custom Images

Custom images are bootable operating system images with the configurations or software needed for an experiment. Experimenters may create their own custom image by starting with a standard OS image, configuring it as needed, and taking a "snapshot" of the image. They can then specify this snapshotted image in their Request RSpecs as the operating system to be loaded when their compute resources are provisioned.

Custom images are particularly useful if configuring and installing software on a compute resource takes a long time since the experimenter has to do this just once on an instance of the operating system and then snapshot it. They are also useful if it is important that a certain version of the operating system be used for the experiment as standard images provided by the aggregates tend to keep up with newer releases of the operating system.

4.5.3 Configuration Management Tools

Industry standard configuration management tools such as Ansible and Chef are a user-friendly way of installing and configuring software on the nodes in an experiment.

Configuration management tools ensure an experiment is in a known configuration regardless of it's original state. The experimenter usually writes a *playbook* or *recipe* that describes the desired state of the node. When the playbook is run the tool uses the playbook to bring the resource to the desired known configuration. The commands in the playbooks are *idempotent* which means that the commands can be run repeatedly without concern for the initial state and no harm will result from the repeated invocations. These playbooks are usually easier to write than shell scripts or install scripts, because the experimenter is only required to describe the final intended state (e.g. Apache is installed, file.txt is present) and not how to get the node into that state (e.g. install Apache) or error handling (e.g. if Apache is not installed, then install Apache).

Configuration management tools make it easy to reproduce experiment configurations and therefore make it easy to do multiple runs with the same setup or with systematic variations such as changing parameters and scaling topologies.

4.6 Archiving

4.6.1 The GENI iRODS Service

GENI provides experimenters a long-term archival service for experiment related data such as measurements. This is the main GENI-provided storage that outlives resource reservations, slices and projects.

The GENI archival service is based on iRODS [32], an open source data management system. iRODS enables data discovery using a meta-data catalog. IRODS meta-data may be attached to files, users, groups, collections (iRODS equivalent of sub-directories), and resources (e.g., a hard drive).

The GENI iRODS service is hosted by RENCI, a research institute in North Carolina. GENI experimenters get iRODS accounts through the GENI Portal. GENI tools such as the GENI Desktop and the GENI Wireless experimentation tools can be configured to use this iRODS account to archive the measurements they collect.

5 Experiment Repeatability and Reproducibility

GENI makes it relatively easy for experimenters to recreate their setup and rerun their experiments. This is important because it encourages experimenters to collect statistics on the *repeatability* [31] of their experiments by recreating and rerunning

their experiments multiple times. As a side-effect, they are less likely to hold on to resources between runs of their experiments, an important consideration in a shared testbed.

Reproducibility, an important principle of the scientific method, is the ability to run experiments created by others and verify their results [30]. GENI supports reprodicibility by: (1) providing tools and mechanisms that make it easy to recreate experiment setups, (2) defining a workflow that produces and consumes formally-defined artifacts such as experiment scripts and resource specifications, and (3) making it easy to share these artifacts for others to reproduce experiments.

5.1 Making Experiments Repeatable and Reproducible

5.1.1 Reducing Variability Across Runs of an Experiment

Picking Resources A measure of experiment repeatability is the variability in measurements across runs. Since GENI is a shared testbed this variability cannot be eliminated. However, experimenters can minimize this variability by picking non-shared resources such as bare machines and by picking the same set of aggregates for different runs of multi-aggregate experiments to minimize latency related variability.

They can also minimize variability by being specific in the Request RSpec about the characteristics of the resources being requested. For example, experimenters can specify the number of cores and memory assigned to compute resources, the locations of these resources down to the physical computer at the aggregate providing these resources and versions of operating systems installed.

Scripting Experiments To ensure resources are programmed and configured identically for every run, experimenters can use one of the techniques for software installation and resource configuration described in Sect. 4.5. In addition, experiment scripting and orchestration using tools such as OEDL and LabWiki (Sect. 4.3) can be used to reduce variability across runs of an experiment.

5.1.2 Sharing Experiment Artifacts for Reprodicibility

GENI supports experiment reprodicibility by making it easy to share artifacts such as RSpecs, custom images, experiment scripts and measurements. RSpecs and install scripts are plain files easily shared on web pages or websites such as GitHub designed for sharing programs and scripts. In addition, experimenters can choose to upload and make their RSpecs public on the GENI Portal. Experimenters can reserve resources from the Portal using RSpecs they or others have uploaded. They can also choose to make their custom images public for others to use. Likewise, the LabWiki tool allows scripts to be shared among experimenters.

Experiment related data including measurements can be archived on the GENI iRODS service (Sect. 4.6.1); experimenters can make these archives public or share them with specific people.

6 Scaling Up Experiments

GENI supports experimentation at scale by providing resources at about 50 geographical locations (as of 2015) connected with Layer 2 VLANs.

In addition, GENI makes it easy to repeatedly bring up similar topologies of different sizes. This supports best practices from software engineering and system administration. GENI experimenters can *start small* with a modest topology consisting of a trivial number of nodes which are representative of the larger topology. Then experimenters can *change one thing at a time* to bring up a sequence of larger topologies with more geographical diversity. Edwards et al. [9] provides advice for novice experimenters when dealing with these issues as well as illustrating this approach with a use case.

GENI tooling supports scaling experiments in a variety of ways. First, the use of the software installation and configuration techniques described in Sect. 4.5) makes it easy to set up and run large experiments without having to manually configure each resource.

Second, carefully crafted install scripts or configuration management playbooks, often make it possible to completely specify the configuration of a given *node type* (i.e. to use the same script to configure all nodes with the same purpose, OS image, software, and configuration). These node types can then be mixed and matched in different combinations to create topologies of different configurations and different sizes. GENI supports this with the following tooling:

- The *scaleup* tool distributed with geni-lib (Sect. 4.1.3) lets experimenters describe node types and one of several standard topologies (grid, ring, full mesh) or a custom topology using a file in INI format. The output of scaleup is a Request RSpec that can be used with any of the resource reservation tools (Sect. 4.2).
- In addition, the GENI Portal, jFed, and Flack all support a "copy and paste" feature in their graphical user interfaces so a given node type can be replicated to easily create large experiments that have a large number of a few node types.

Third, once an experiment has been tested in a single Aggregate it can be easily modified to run as a multi-Aggregate experiment. Tools such as Jacks and jFed allow a single Aggregate Request RSpec to be imported and then for different resources to be assigned to different Aggregates. This new RSpec can then be used to reserve resources and run a multi-Aggregate version of the original experiment.

7 Collaboration

GENI supports *collaborative* experimentation by allowing researchers from different institutions to operate on the same experiment and providing them the ability to add collaborators over the life of a project. This is important for large project teams such as the NSF Future Internet Architecture projects [20] and for long-running experiments.

7.1 Mechanisms for Collaboration

Research in GENI is organized into *projects*. A project contains both people and their experiments. A project may have many experimenters as its members and an experimenter may be a member of many projects. Every project has a *Project Lead* who can add or remove members. The project lead can designate one or more Admins who manage project membership as well.

Project members create slices in the context of a project; there can be many slices in a project. The person who created a slice and the Project Lead can choose to add other project members to the slice. Slice members can add and remove resources in the slice and run experiments using resources in the slice. Accounts for slice members are automatically set up on compute resources when the resources are instantiated.

This organization of research enables collaborative experimentation. A researcher can create a project and add collaborators as project members. When a new collaborator joins the team, she can be added to the project and to any slices to which she would need access.

Figure 16 shows a professor who is a project lead and has created separate projects for research and classroom use. For the class project, the professor has given his teaching assistant Admin privileges and has given the project an expiration which means the students will not be able to use the project after that date.

Figure 17 shows two slices created in the same project by the same person. The Project Lead is added to each slice by default. One of the slices contains an additional member. The two slices contain different resources and all members will have accounts to login to the resources when the resources are reserved.

References

1. Anand, A., Dogar, F., Han, D., Li, B., Lim, H., Machado, M., Wu, W., Akella, A., Andersen, D.G., Byers, J.W., Seshan, S., Steenkiste, P.: XIA: an architecture for an evolvable and trustworthy internet. In: Proceedings of the 10th ACM Workshop on Hot Topics in Networks, HotNets-X, pp. 2:1–2:6. ACM, New York (2011)
2. Ansible Inc. Ansible. http://www.ansible.com (2016). Accessed Jan 2016

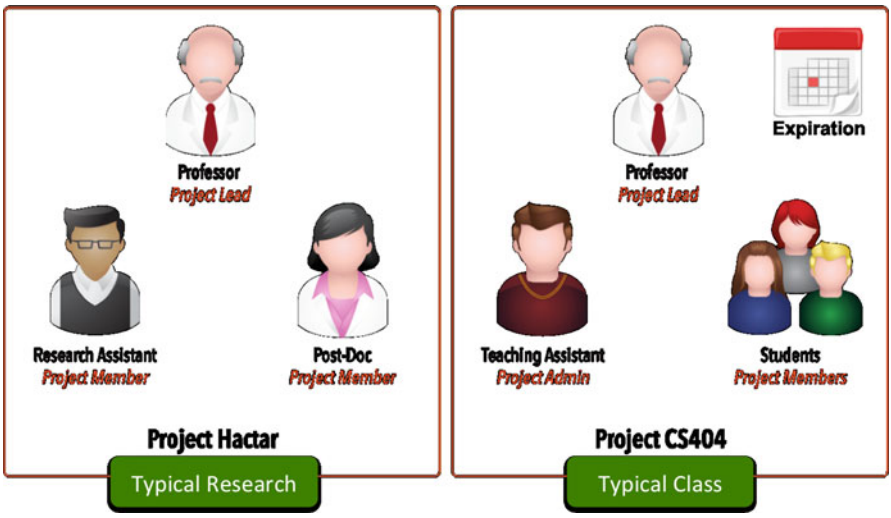

Fig. 16 A professor with separate projects for research and classroom use

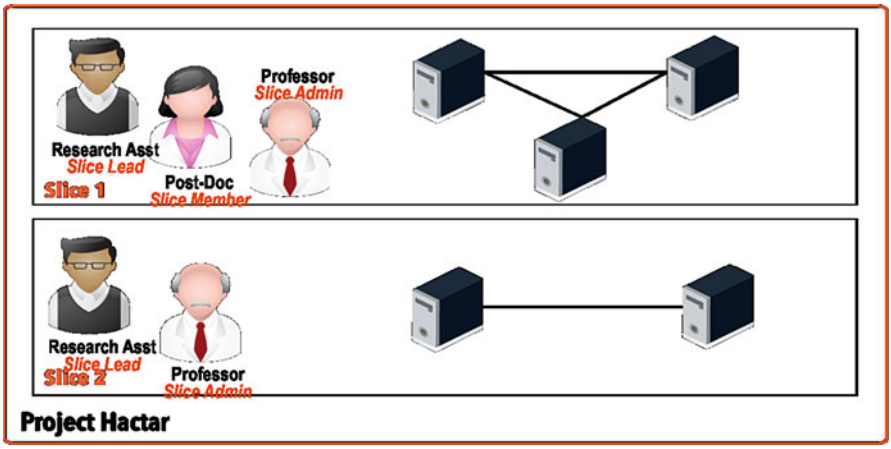

Fig. 17 Different slices can have different resources. The Project Lead is added to the slice by default as a Slice Admin. The slice creator (a.k.a. Slice Lead) can add additional people to the slice as desired. When resources are reserved, accounts will be created for all current slice members

3. Baldin, I., Castillo, C., Chase, J., Orlikowski, V., Xin, Y., Heermann, C., Mandal, A., Ruth, P., Mills, J.: ExoGENI: a multi-domain infrastructure-as-a-service testbed. In: The GENI Book. Springer, New York (2016)
4. Barnstormer Softworks. Welcome to geni-lib documentation! http://geni-lib.readthedocs.org/en/latest/ (2016). Accessed Jan 2016
5. Brinn, M.: GENI architecture foundation. In: The GENI Book. Springer, New York (2016)
6. Chef Software Inc. Chef. https://www.chef.io (2016). Accessed Jan 2016

7. Day, J., Matta, I., Mattar, K.: Networking is IPC: a guiding principle to a better internet. In: Proceedings of the 2008 ACM CoNEXT Conference, CoNEXT '08, pp. 67:1–67:6. ACM, New York (2008)
8. Dempsey, H.: The GENI mesoscale network. In: The GENI Book. Springer, New York (2016)
9. Edwards, S., Liu, X., Riga, N.: Creating repeatable computer science and networking experiments on shared, public testbeds. SIGOPS Oper. Syst. Rev. **49**(1), 90–99 (2015)
10. GENI Project Office. The GENI Portal. https://portal.geni.net (2016). Accessed Jan 2016
11. GENI Project Office. Omni. http://trac.gpolab.bbn.com/gcf/wiki/Omni (2016). Accessed Jan 2016
12. iMinds Research Institute. jFed is a java-based framework for testbed federation. http://jfed.iminds.be (2016). Accessed Jan 2016
13. Internet2. http://www.internet2.edu (2016). Accessed Jan 2016
14. Izard, R., Ramanathan, P., Wang, K.: GENI Cinema architecture. http://groups.geni.net/geni/raw-attachment/wiki/sol4/GENICinema/GENI-Cinema-Architecture.pdf (2016). Accessed Jan 2016
15. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. In: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '09, pp. 1–12. ACM, New York (2009)
16. Jain, S., Chen, Y., Zhang, Z.-L.: VIRO: a scalable, robust and namespace independent virtual Id routing for future networks. In: 2011 Proceedings IEEE INFOCOM, pp. 2381–2389 (2011)
17. Jourjon, G., Rakotoarivelo, T., Dwertmann, C., Ott, M.: Labwiki: an executable paper platform for experiment-based research. Proc. Comput. Sci. **4**, 697–706 (2011)
18. McGeer, R., Ricci, R.: The instaGENI project. In: The GENI Book. Springer, New York (2016)
19. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
20. National Science Foundation. NSF Future Internet Architecture Project. http://www.nets-fia.net (2016). Accessed Jan 2016
21. OASIS SAML Working Group. Shibboleth Federated Identity Solution. http://www.shibboleth.net (2016). Accessed Jan 2016
22. OMF Overview. http://omf.mytestbed.net/projects/omf (2016). Accessed Jan 2016
23. Peterson, L., Ricci, R., Falk, A., Chase, J.: Slice-Based Federation Architecture. http://groups.geni.net/geni/raw-attachment/wiki/SliceFedArch/SFA2.0.pdf (2016). Accessed Jan 2016
24. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A blueprint for introducing disruptive technology into the Internet. SIGCOMM Comput. Commun. Rev. **33**(1), 59–64 (2003)
25. Rakotoarivelo, T., Ott, M., Seskar, I., Jourjon, G.: OMF: a control and management framework for networking testbeds. In: SOSP Workshop on Real Overlays and Distributed Systems (ROADS) (2009)
26. Raychaudhuri, D., Nagaraja, K., Venkataramani, A.: MobilityFirst: a robust and trustworthy mobility-centric architecture for the future internet. SIGMOBILE Mob. Comput. Commun. Rev. **16**(3), 2–13 (2012)
27. Resource Specification Documents. http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs (2016). Accessed Jan 2016
28. Rouskas, G., Baldine, I., Calvert, K., Dutta, R., Griffioen, J., Nagurney, A., Wolf, T.: Choicenet: network innovation through choice. In: 2013 17th International Conference on Optical Network Design and Modeling (ONDM), pp. 1–6 (2013)
29. Seskar, I., Raychaudhuri, D., Gosain, A.: 4G cellular systems in GENI. In: The GENI Book. Springer, New York (2016)
30. Stodden, V.C.: The scientific method in practice: Reproducibility in the computational sciences. Technical Report 4773-10, MIT Sloan School of Management (2010)
31. Taylor, B.N., Kuyatt, C.E.: Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results, Chapter D.1.1.2 Repeatability (of results of measurements). Number Technical Note 1297. National Institute of Standards and Technology (1994)
32. The iRODS Consortium. iRODS. http://irods.org (2016). Accessed Jan 2016

33. The OMF Experiment Description Language (OEDL). https://mytestbed.net/projects/omf6/wiki/OEDLOMF6, Accessed Jan 2016
34. University of Kentucky. The GENI Desktop. http://genidesktop.netlab.uky.edu (2016). Accessed Jan 2016
35. Weigle, M.C., Adurthi, P., Hernández-Campos, F., Jeffay, K., Smith, F.D.: Tmix: a tool for generating realistic TCP application workloads in NS-2. ACM SIGCOMM Comput. Commun. Rev. **36**(3), 67–76 (2006)
36. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. SIGOPS Oper. Syst. Rev. **36**(SI), 255–270 (2002)