ExoGENI: Principles and Design of a Multi-Domain Infrastructure-as-a-Service Testbed*

Ilia Baldine
RENCI
ibaldin@renci.org

Yufeng Xin
RENCI
yxin@renci.org

Anirban Mandal
RENCI
anirban@renci.org

Paul Ruth
RENCI
pruth@renci.org

Aydan Yumerefendi
Dept. of Computer Science
Duke University

Jeff Chase
Dept. of Computer Science
Duke University
chase@cs.duke.edu

ABSTRACT

NSF's GENI program seeks to enable experiments that run within virtual network topologies built-to-order from testbed infrastructure offered by multiple providers (domains). GENI is often viewed as a testbed integration effort, but behind it is an ambitious vision for multi-domain infrastructure-as-a-service (IaaS). This paper presents ExoGENI, a new GENI testbed that links GENI to key advances in virtual infrastructure services outside of GENI: cloud computing systems and network circuit fabrics. ExoGENI orchestrates a federation of independent cloud and circuit providers through their native IaaS interfaces, and links them to other GENI tools and resources.

ExoGENI will deploy cloud sites on host campuses within the US beginning in 2012 and link them with national research networks and other circuit networks through programmable exchange points. The ExoGENI sites and control software are enabled for software-defined networking using OpenFlow. ExoGENI offers a powerful unified hosting platform for deeply networked, multi-domain, multi-site cloud applications. We intend that ExoGENI will seed a larger, evolving testbed linking other campus cloud sites and transport networks, and that it will enable real-world incremental deployment of innovative distributed services and new visions of a Future Internet.

1. INTRODUCTION

ExoGENI is a new testbed at the intersection of networking and cloud computing, funded through the US National Science Foundation's GENI project. The testbed is designed to support research and innovation in networking, operating systems, distributed systems, future Internet architectures, and deeply networked, data-intensive cloud computing. Each ExoGENI site is reconfigurable "down to metal and glass" providing a flexible substrate for repeatable experiments. The testbed can also serve as a platform for novel gigabit applications and services, e.g., for the US IGNITE initiative. ExoGENI was funded in December 2011 and is scheduled to become operational in September 2012.

ExoGENI is based on an extended Infrastructure-as-a-Service (IaaS) cloud model with orchestrated provisioning across multiple sites. Following the standard NIST terminology, each ExoGENI site is a private IaaS cloud that may be operated independently on behalf of a local user community, using a standard cloud stack to instantiate and manage virtual machines. The sites federate by delegating certain functions for identity management, authorization, and resource management to coordinator services in the ExoGENI federation. This structure enables a network of private IaaS clouds to operate as a hybrid community cloud. Thus ExoGENI is an example of a *multi-domain* or *federated* cloud system, which some have called an InterCloud.

*This work is supported by the US National Science Foundation through the GENI initiative (Raytheon BBN Labs) and NSF awards OCI-1032873, CNS-0910653, and CNS-0720829; by IBM and NetApp; and by the State of North Carolina through RENCI.

ExoGENI combines this multi-domain cloud structure with a high degree of control over networking functions: OpenFlow networking, multi-homed cloud servers that can act as virtual routers, site connectivity to national circuit backbone fabrics through host campus networks, and linkages to national and international networks through programmable exchange points. The initial funding will deploy at least 12 ExoGENI sites at universities and research labs across the United States. Each site includes a packaged rack with a cloud server cluster and an integrated OpenFlow network, built by our industry partner IBM. The sites also have connectivity to dynamic point-to-point Ethernet services and other multi-layer circuit services offered by regional and national research networks: most sites have 10Gbps access to the National Lambda Rail (NLR) and/or other transport networks including Internet2, the BEN network in North Carolina, LEARN in Texas, and the 100Gbps ANI testbed operated by the US Department of Energy.

The ExoGENI testbed is designed to combine cloud computing and advanced networking with OpenFlow in a unified testbed. The ExoGENI testbed software supports GENI APIs and extended APIs to enable users to instantiate and program *slices* of virtualized resources within the infrastructure. A slice is an interconnected set of resources under the control of a particular project. A slice may serve as a container or execution context to host an application or network service. ExoGENI slices may contain layer-2 network topologies with programmable nodes: the topologies may span multiple sites and link to other external resources as permitted by peering and interconnection agreements. ExoGENI slices are isolated: they interact with the outside world through controlled interfaces, and the virtual resources in a slice may have defined quality-of-service properties.

The project aims to enhance US research cyberinfrastructure capabilities in four inter-related ways:

- **The missing link.** ExoGENI interconnects clouds to high-speed circuit networks, enabling a range of networked cloud applications and services, including data-intensive interaction, distributed data sharing, and location-aware services.
- **On-ramps to advanced network fabrics.** ExoGENI shows how to manage campus clouds as a bridge to national transport network fabrics from campus networks, overcoming a key limitation identified by NSF's CF21 vision. ExoGENI cloud sites can act as *virtual colocation centers* that offer on-demand cloud services adjacent to fabric access points. ExoGENI sites at fabric intersection points can also act as *virtual network exchanges* to bridge "air gaps" between fabrics stemming from lack of direct connectivity or incompatible circuit interfaces.
- **Cloud peering and data set mobility.** ExoGENI enhances the potential for peering and sharing of private clouds deployed by institutions and research groups. It offers a means to bring data and computation together by migrating datasets to compute sites or placing computation close to data at rest. An important goal of the project is to provide a foundation for organic and sustainable growth of a networked intercloud through a flexible federation model that enables private cloud sites to interconnect and share resources on their own terms.
- **Networking as a service.** ExoGENI brings flexible network configuration to cloud computing. It also enables experimental deployments of new packet networking models over a flexible link substrate. Built-to-order virtual networks can implement routing overlays using IP or other packet-layer protocols within the slice. Slice owners may deploy custom node operating systems with alternative networking stacks into their nodes, and use programmable datapaths and/or virtual routers to implement new network services at the cloud edge and at network intersection points.

The control software for ExoGENI was developed and refined in a collaboration between RENCI and Duke to create a GENI testbed "island" around RENCI's Breakable Experimental Network (BEN), based on the Open Resource Control Architecture (ORCA). ORCA is an outgrowth of earlier NSF-funded projects in networked cloud computing at Duke. It is based on the SHARP federation model [8] and the plugin architecture of the Shirako resource leasing core [11], with extensions for automated control policies developed in the Automat project [15]. During the project we developed ORCA plugins to implement a circuit service over BEN, interface to NLR's Sherpa FrameNet service and other third-party circuit services, and drive standard open-source cloud stacks (e.g., Eucalyptus or OpenStack), replacing our older cloud software called Cluster-on-Demand [5]. These steps led us to the ExoGENI architecture, which orchestrates a collection of independent virtual infrastructure providers through their native IaaS interfaces.

This paper gives an overview of the ExoGENI testbed and its control software. The research contribution of this paper is to summarize the design principles of ExoGENI resulting from our three-year experience in the ORCA-BEN project, and the relationship and integration with the GENI architecture as it currently stands (Section 2). Section 3 summarizes the testbed hardware and software stack. Some capabilities of the control software are currently unique within the GENI project; Section 4 describes these aspects of the design and implementation in more detail. The software described in this paper has been implemented, deployed, and demonstrated at various GENI events, with the exception of the OpenFlow capability

discussed in Section 4.5. Some partial results from the project have been reported in previous research publications [3, 2]. This is the first full-length paper with a complete overview of the work.

2. GENI AND EXOGENI

ExoGENI may be viewed as a group of resource providers within a larger GENI federation. Resource providers are called *aggregates* in GENI. Each aggregate makes some infrastructure (*substrate*) available to users associated with the federation, through a server called an *Aggregate Manager* (AM). Aggregates may be independently owned and administered.

The name "ExoGENI" reflects our view of how GENI will evolve, and what capabilities are needed to deliver on the promise of GENI to "explore networks of the future at scale". GENI is evolving alongside cloud technologies and open network control systems, whose goals overlap with GENI. The rate of investment in developing and deploying these systems is quite a bit more than an order of magnitude larger than the GENI effort. One purpose of ExoGENI is to define a path to leverage these technologies and substrates in the GENI project. At the same time, GENI control software offers new ways to combine and extend them as a unified deployment platform for advances in network science and engineering. Another goal of ExoGENI is to define the right principles for GENI control software to serve this purpose, and in doing so to address important orchestration challenges for networked cloud computing. The "Exo" (outside) prefix captures five inter-related themes and principles.

E1 **Decouple infrastructure control from testbed orchestration.** ExoGENI uses a modular structure at agregates, in which each aggregate runs a standard front-end server (AM) with a plug-in interface for a back-end infrastructure control service. The AM front-ends orchestrate the invocations of these back-end infrastructure control services through their various APIs. They handle the details of cooperating with other services in a federation, including identity management, authorization, provider discovery and selection, and a common resource model for tools (e.g., declarative resource representations).

E2 **Use off-the-shelf software and IaaS services for infrastructure control.** Standard services and APIs offer a ready back-end solution to instantiate and release virtual resources in cloud sites, circuit services, and other virtual infrastructure services. Cloud sites may be deployed quickly with "turnkey" open-source cloud stacks, such as Eucalyptus and OpenStack, which support the de facto standard Amazon EC2 IaaS cloud API. Common APIs such as OSCARS are also emerging for transport network circuit services.

E3 **Leverage shared third-party substrate through their IaaS interfaces.** This compatibility with standard back-end infrastructure control services offers a path to bring independent resource providers into the federation, beyond the core GENI-funded substrate. The providers deploy some virtual infrastructure service using standard software and IaaS APIs. To join the federation, the provider deploys an AM to "wrap" the IaaS service. The AM acts as a customer of the service and exposes it testbed users. The provider may serve other customers concurrently through its native IaaS APIs rather than dedicating its substrate for use by the testbed.

E4 **Enable substrate owners to contribute resources on their own terms.** Participating providers are autonomous: they are empowered to approve or deny any specific request according to their policies. ORCA resource allotments are visible to all parties and are controlled by the providers. Providers allocate virtual infrastructure resources with attached QoS properties for defined intervals; the slice owners determine what resources to request and how to expose them to applications. These principles are similar to those put forward for the *exokernel* extensible operating system [13] developed at MIT in the 1990s, and the ExoGENI name is partly an homage to that project [12]. They also suggest how to share common substrate among multiple testbed models. For example, ExoGENI sites may hold back resources from the testbed for use by other software through the native IaaS services.

E5 **Accommodate abstract views of dynamic substrate.** Testbed control software should not assume that it has a complete physical view of the underlying hardware substrate, or that the visible substrate is static. In essence, providers advertise virtual infrastructure services (the *domain models* in Section 4) rather than physical substrate. Some providers may offer physical substrate for direct allocation, but this is a special case and not necessarily the most common or most important case. The testbed's view of provider substrate may change with time, since the substrate may be shared with other uses outside of GENI.

ExoGENI is significant in part because it offers our first opportunity to evaluate this model in a production testbed. We intend that ExoGENI will serve as a nucleus for a larger, evolving testbed that encourages participation from independent cloud sites, transport networks, and testbed providers. As the capabilities of these infrastructure providers continue to advance, the ExoGENI model will enable real deployment of not just innovative distributed services but also new visions of a Future Internet.

2.1 ExoGENI and the GENI Federation

These principles represent a departure in the GENI effort, whose current standards evolved from testbeds (PlanetLab and Emulab) that were already established and accepted by the research community. Their control software manages substrates that are permanently dedicated to the testbed and under the direct control of a central testbed authority. ExoGENI is the first GENI-funded substrate whose control software departs from that model and instead uses standard virtual infrastructure services, which may be deployed and administered independently and/or shared with other uses.

These predecessors of GENI have provided years of valuable service to researchers in our community, but they do not serve the need for a deployment platform for IGNITE applications or evolution of a Future Internet. Their primary uses have been to evaluate new ideas under controlled conditions (for Emulab) and to measure the public Internet as it currently exists (for PlanetLab). PlanetLab has enabled development of innovative distributed services in the real world, but it is limited as a deployment platform because it supports only best-effort resource allocation, it does not offer a path for providers to expose their QoS capabilities to testbed users, and it depends on the existing layer-3 Internet as its dataplane.

ExoGENI extends the GENI federation model. GENI aggregates implement standard GENI APIs for user tools to request resources from aggregates. Other services, which we will call *coordinators*, help aggregates to cooperate and function together to serve users associated with the federation. Aggregates may delegate certain powers and trust to coordinators. For example, the current GENI architecture defines coordinators to endorse and monitor participating aggregates, authorize and monitor use of the testbed for approved projects, and manage user identities and their association with projects. The GENI coordinators are often described together under the umbrella of a GENI *Clearinghouse*, but they act as a group of distinct services endorsed by a common GENI root authority, rather than as a monolothic entity.

The GENI federation architecture allows aggregates to choose for themselves whether to accept any given coordinator and what degree of trust to place in it. These choices are driven by federation governance structure. GENI has opted for a rigid hierarchical governance structure for its initial trial deployment, for reasons of safety and simplicity. To join the GENI federation an aggregate must enter into certain agreements, including acceptance of GENI-endorsed coordinators and compliance with their policies.

The aggregates in the initial ExoGENI deployment will enter into agreements with GENI and accept and trust all GENI coordinators. Specifically, ExoGENI trusts GENI coordinators and their policies to certify users, issue keypairs to users, authorize projects, approve creation of empty slices, authorize users to operate on approved slices, and endorse other aggregates in GENI. The GENI coordinators in turn delegate some identity management functions to identity systems operated by participating institutions (Shibboleth/inCommon).

2.2 ExoGENI Control Structure

ExoGENI provides additional coordinators and APIs for managing resources and configuring end-to-end virtual topologies within the ExoGENI testbed itself. The ExoGENI services are based on the ORCA control framework.

Figure 2 depicts the services and their trust structure. Each ExoGENI site and each participating aggregate runs an ORCA Aggregate Manager (AM). A *service registry* maintains a directory of binding information and public keys for all ExoGENI coordinators and AMs. ExoGENI AMs accept every coordinator listed in the registry. The registry is controlled by the testbed operator (RENCI).

User requests enter the ORCA control system through a *Slice Manager* (SM). The ExoGENI SMs run plugins that support the GENI standard APIs to enable use of the ExoGENI aggregates by GENI users and tools. In general, an ORCA slice manager runs on behalf of the owners of one or more slices, with no special trust from other services. The ExoGENI deployment configures the registry-endorsed SMs to also function as coordinators that provide a trusted interface at the edge of ExoGENI to the rest of the GENI federation. The ExoGENI AMs trust these SMs to acts as authorization servers to validate the GENI authorization for the slice and the user identity in each request. The ExoGENI SMs appear as GENI AMs to GENI users and tools.

There are two distinct SM configurations in ExoGENI. Each ExoGENI site runs an SM called a *site SM*. Requests to a site SM can operate only on the aggregate at that site. Since the SMs export the GENI API, each site's SM exposes that site as a distinct aggregate within GENI. In addition, a global ExoGENI SM called an *ExoSM* can access all aggregates within the ExoGENI testbed. There may be any number of ExoSMs, but the initial deployment has a single ExoSM. The ExoSM exposes ExoGENI as a single GENI aggregate. The ExoSM offers a unified view of the testbed, and supports virtual topology mapping across the ExoGENI aggregates, including the circuit providers.

Internally to ExoGENI, the SMs interact with other entities using ORCA-defined interfaces. These entities include the ExoGENI AMs and ORCA coordinators called *brokers* that collect and share information about ExoGENI aggregates, including their resources, services, and cross-aggregate connections. A broker may also coordinate resource allocation across aggregates

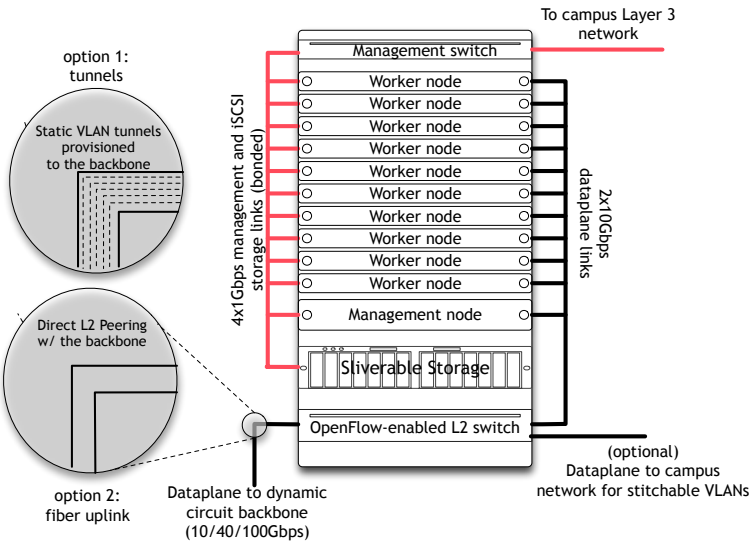

Figure 1: Components and connections in an ExoGENI site rack.

and guide or limit the flow of requests to the aggregates. An ORCA broker is trusted by some set of ORCA AMs to enforce resource allocation and scheduling policies, which may span multiple aggregates. Each request for resources from an ExoGENI aggregate must be approved by a broker before it is passed to the aggregate's AM. The ExoSMs in the ExoGENI deployment use a common testbed-wide broker that is accepted by all ExoGENI aggregates.

ExoGENI also runs a central depository for virtual machine boot images that can be named by URL and fetched as needed to the ExoGENI cloud sites (Section 4.6).

2.3 Integration with GENI

GENI users and their tools choose for each slice whether to access the ExoGENI testbed as a single aggregate (through the ExoSM) or as a collection of distinct site aggregates (through the site SMs). External GENI tools can interact with ExoGENI site aggregates based on the current GENI architecture, in which aggregates are loosely coupled except for common authorization of users and slices. Each ExoGENI slice may also link to other GENI resources under the direction of the GENI standard tools and APIs.

At the same time, the ExoSM allows ExoGENI to offer capabilities for automated cross-aggregate topology embedding and stitching within the ExoGENI testbed. These capabilities are currently unique within GENI. They are based on coordinator services, APIs, resource representations, and tools that are not part of a GENI standard. In particular, GENI defines no coordinators for resource management, so cooperation among GENI aggregates is based on direct interaction among AMs or exchanges through untrusted user tools. GENI is developing new extensions that would offer similar capabilities for automated configuration of cross-aggregate virtual networks.

ExoGENI also differs from current GENI practice with respect to the usage model for OpenFlow networks. GENI views an OpenFlow datapath as a separate aggregate that "allocates" the right to direct network traffic flows matching some set of specified flow patterns, which are approved manually by an administrator. In ExoGENI, OpenFlow is an integrated capability of the ExoGENI aggregates, rather than a distinct aggregate itself. ExoGENI slices may designate OpenFlow controllers to direct network traffic within the virtual network topology that makes up the slice's dataplane. The OpenFlow-enabled ExoGENI aggregates authorize the controllers automatically for flows in the slice's dataplane. As an option, ExoGENI may also allow GENI users to program the OpenFlow datapaths as separate aggregates (using FOAM), with manual approval by GENI administrators.

These are two areas in which ExoGENI departs from initial directions of the GENI architecture and standards. One purpose of this paper is to outline the approaches we use to other members of the GENI community as a candidate path forward for GENI. Section 4 discusses these ExoGENI approaches in more detail.

3. EXOGENI SITES

Each ExoGENI site is a small-scale cloud site capable of supporting about 100 virtual machines, based on a standard cloud

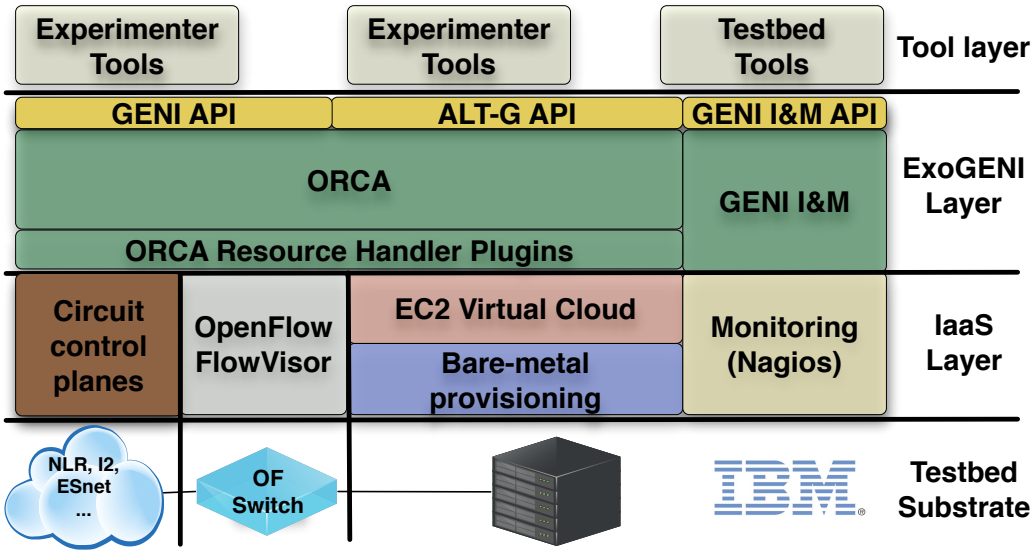

Figure 2: Conceptual view of the ExoGENI software stack. This diagram combines components that run in different process and (in some cases) on different servers.

cluster stack (e.g., Linux/KVM, and Eucalyptus or OpenStack). Figure 1 depicts the site hardware. Each site has a rack with multiple host nodes or *worker nodes* which are the server substrate for provisioning virtual nodes for slices. The initial worker nodes are configured with dual-socket 2.4GHz Intel Westmere CPUs, each with 6 cores and 24GB of RAM. A single *management node* (head node) in each site runs the management software stack described below. The rack also includes an iSCSI storage appliance with about 7TB of storage for images, measurement data, and other experimental needs.

All components are connected to a *management switch*, which has a Layer 3 connection to the campus network and from there to the public Internet. This switch is used for intra-site access to the iSCSI storage, for remote management through a VPN appliance, and for slice dataplane connectivity to the public Internet if permitted by the host campus. Each node has multiple 1Gbs ports on this switch for these various management uses.

A L2 *dataplane* switch connects the worker nodes to the backbone. The dataplane switch is the termination point for incoming L2 links that a host campus presents to a site rack, as a bridge to external circuit providers and (optionally) to VLANs or subnets on the host campus network. ExoGENI uses an IBM G8264R 10G/40G OpenFlow-enabled hybrid L2 switch with VLAN capabilities. Section 4.3 outlines the VLAN services that ExoGENI sites offer to construct virtual slice dataplanes through this switch. All traffic in an ExoGENI slice dataplane can be directed via OpenFlow on ingress and egress, at the option of the slice owner. In addition, slices may construct virtual topologies with internal switching and routing nodes on OpenFlow datapaths. Each worker node has two 10Gbps interfaces (ports) connected to the dataplane switch for the L2 slice dataplanes. We expect to configure one port of each pair for OpenFlow and the other for standard VLAN operation. Section 4.5 outlines our views on more flexible hybrid OpenFlow/VLAN operation in the future.

Slice owners have low-bandwidth management access to their instantiated nodes (virtual machines) over the public Internet through a management port. Each ExoGENI site runs an SNAT/DNAT network proxy (not shown) to provide this access. Management access to a node is by root ssh for a public key specified by the slice owner. Nodes may also be assigned optional proxied public IP addresses, at the discretion of the site's hosting campus. This access is provided by the underlying cloud service. In the case of Eucalyptus the public IP access is proxied through the head node.

3.1 Site Software

The testbed offers multiple levels of provisioning interfaces for user access and management, including standard cloud interfaces (EC2 and xCAT), OpenFlow, and layered GENI control and monitoring functions. The head node runs IaaS provisioning software (the cloud stack) and various network proxies and services for GENI and ORCA. These services include the two aggregate managers for the site: the ORCA AM and the site SM, which acts as a GENI aggregate manager supporting the standard GENI API.

Figure 2 gives a conceptual view of the ExoGENI software stack. Tools for experimenters and operators run outside of

the testbed; they invoke remote APIs on ORCA servers running on the site head node through its public IP address on the management network. The SMs support the GENI API and an alternative API for the ExoGENI testbed, which is labeled as *ALT-G* in the figure. This native API describes resources using semantic network models (discussed in Section 4), and is evolving independently of the GENI APIs. The instrumentation and monitoring (I&M) API also continues to evolve to meet the needs of production GENI testbeds, and is outside the scope of this paper.

The ORCA aggregate managers receive calls from the SMs and invoke *handler* plugins for one or more typed resource pools managed by each aggregate. The handlers invoke native interfaces of the IaaS layer. For the ExoGENI site aggregates this layer runs off-the-shelf open-source management software for clusters and networks, including Linux/KVM, EC2 cloud managers (Eucalyptus or OpenStack), and monitoring agents based on Nagios. Most activity by a site operator occurs at this layer. For circuit aggregates the handlers invoke a native circuit API. These IaaS services may also be exposed directly to users or other orchestration software, independently of ORCA. From the perspective of these services the ORCA AMs run with a user identity created by the site operator. They are unaware that the AM is exposing the resources through its own APIs and acting on behalf of testbed users.

ExoGENI sites run other software components not shown in Figure 2. A key goal of ExoGENI is to support flexible, automated deployment of customized software stacks on shared servers, with secure isolation and manageable quality of service. The EC2 cloud handlers invoke an ImageProxy service to fetch images from outside the site and register them with the local cloud manager (see Section **??**). Most testbed users will use a virtual machine service, but we also have a bare-metal imaging capability based on the xCAT provisioning tool (open-source xCAT [7], developed and maintained by IBM). Currently we use xCAT only to deploy the Linux distributions on worker nodes, and do not expose it through the AM. Programmatic bare-metal imaging will be useful for performance-critical applications, such as virtual routers, and it is on our roadmap.

3.2 Circuit Backbones

Each circuit provider offers a point-to-point Ethernet service among specified point-of-presence locations on the edge of the transport provider's network. ExoGENI will use dynamic circuit services offered by NLR (Sherpa), Internet2 (ION) and ESNet (OSCARS). Two of the sites (RENCI, Duke University) are connected to an optical testbed called BEN (**Breakable Experimental Network** [1]) built by the state of North Carolina and managed by RENCI. The BEN sites have 20Gbps capacity on BEN. BEN has a direct connection to a 10Gbps NLR FrameNet port. Other BEN-connected rack sites are available pending funding.

Rack sites connect to the circuit backbones either through dynamic VLANs on a dedicted fiber or a static pool of VLAN tunnels that traverse campus networks and RONs. Each host campus presents these circuit VLANs to the local site dataplane switch. We negotiated with most sites that the racks will have access to a significant fraction of the backbone interface bandwidth, some of which will be upgraded to 10Gbps. The static VLAN tunnels can be viewed as available **channels of connectivity** for dynamic connections to the other ExoGENI sites through the circuit fabrics. Circuit VLANs may also connect to other resources outside of ExoGENI, including other GENI resources through the existing static GENI Waves.

3.3 Network Exchange Points

ExoGENI has connectivity to a wider range of circuit providers through exchange points, primarily a RENCI-operated exchange on BEN and the StarLight facility in Chicago, which lies at the intersection of a number of national and international networks. RENCI has deployed L2 switches (Cisco 6509) under ORCA control to these exchange points. The exchange points are also useful in part because they provide VLAN tag translation. The control software is described in more detail in Section 4.4.

We have demonstrated how to use the StarLight switch to bridge connections among different circuit providers and to interconnect member sites of GENI's "Cluster-D", including the ViSE steerable weather radar testbed at U. Mass Amherst. Some sites without NLR access have dedicated connectivity to StarLight through regional and commercial providers, and can bridge circuits through the "air gap" to other networks via StarLight under ORCA control.

ExoGENI will use this point-of-presence at StarLight to enable dynamic connectivity to other GENI substrates, DOE ANI, Internet-2, ESNet, NLR, various regional networks in the US, and resources at international testbed partners. Using the RENCI switch at StarLight, ExoGENI will be able to link its NLR-connected sites (representing the majority of ExoGENI) to the **NERSC/LBL site**, which is connected to ESNet and the ANI 100Gbps DOE testbed. Similarly through StarLight, GLORIAD and KOREN we will be able to connect at Layer 2 to the **FIRST@PC OpenFlow testbed** located in Korea. We will also have the opportunity to connect to our partners from Fraunhofer FOKUS and their Teagle control framework [4] via StarLight and GEANT.

Figure 3 shows an example of how VLAN translation at fixed points in the network topology (StarLight and RENCI)

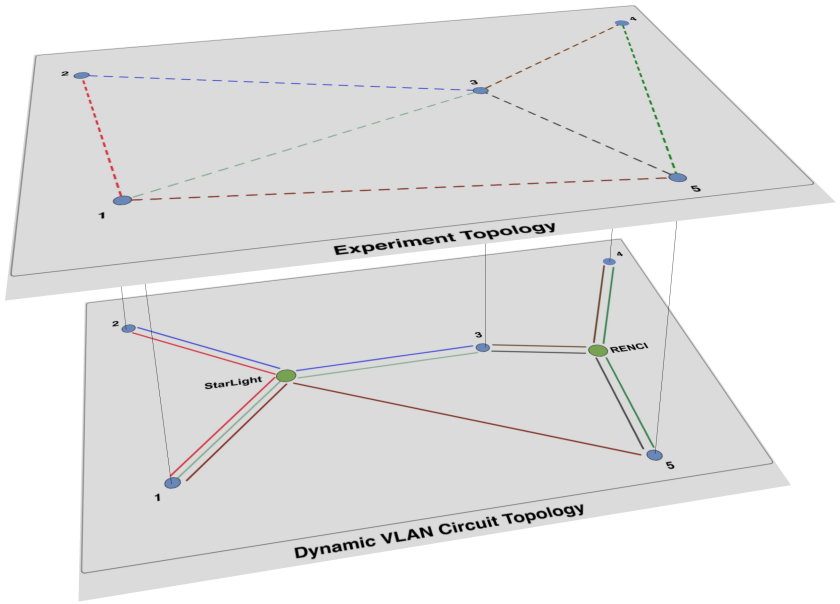

Figure 3: A virtual topology mapped to a multi-provider circuit substrate with exchange points.

can serve as an enabler for instantiating dynamic slice topologies for GENI and ExoGENI. This figure shows the desired experiment topology in the top plane and the circuits and their remapping in the bottom plane.

4. MULTI-DOMAIN ORCHESTRATION

This section introduces an architecture and model to instantiate and program a virtual topology for a slice. ExoGENI is an implementation of the model based on ORCA and an evolving set of technologies, including point-to-point Ethernet services, OpenFlow-enabled hybrid Ethernet switches, and standard stacks for virtual cloud computing (e.g., Eucalyptus, OpenStack, xCAT, Perceus). An ExoGENI slice may contain a virtual topology consisting of virtual nodes, which include virtual machines and programmable switch datapaths, and virtual network links, which are based on Ethernet standards. ExoGENI is VLAN-sliced: each virtual link corresponds to a unique VLAN tag at any given point in the network.

ExoGENI supports virtual infrastructure resources, which are instances of "fundamental computing resources, such as processing, storage, and networks", according to the NIST definition of Infrastructure-as-a-Service. We use the generic term *sliver* (a piece of a slice) to refer to any typed virtual resource, such as a virtual node or virtual link, that is instantiated from a single aggregate for a single slice, and managed independently of other slivers. Each sliver may be bound to various capabilities or properties offered by the aggregate, such as QoS properties. Each aggregate exports services to instantiate and manage slivers of one or more types. We use the term *controller* to refer to an entity that requests sliver operations on behalf of slice owners. Both ORCA and OpenFlow use the term "controller" in this way: in this section it will be clear from context if the term "controller" refers specifically to an OpenFlow controller or to an ORCA slice controller. An ORCA slice controller is a plugin module running within some ORCA Slice Manager (SM).

As discussed in Section 2 each infrastructure provider within ExoGENI is represented by an ORCA Aggregate Manager (AM), which acts as an authority for the provider site or domain. From the perspective of an ORCA AM each slice has a single controller. The controllers receive requests from users and their tools through external APIs, and make calls to the aggregates to process those user requests. In ExoGENI these controllers are part of the testbed software and are not user-replaceable. The ExoGENI AMs and brokers trust the SMs to run valid controllers and verify that the requests are legitimate, i.e., that each request is a valid API call on an approved slice originating from a user who is authorized to operate on the slice. An ExoGENI AM will attempt to serve any request from any ExoGENI SM.

4.1 Managing Virtual Topologies: an Overview

ORCA slice controllers running within an ExoGENI-wide SM (ExoSM) can manage complete virtual topologies across all ExoGENI aggregates. Each user request at an ExoSM controller specifies a new virtual topology or changes to an existing virtual topology. The controller is responsible for planning mappings of slivers onto aggregates to instantiate requested virtual

topologies. The controller runs a map agent that analyzes the users's specification and generates a partially ordered set of sliver operations on target aggregates. The combined effect of these operations, if they are successful, is to instantiate or modify the virtual topology as requested.

The ExoSM controller and other ORCA plugins for ExoGENI use semantic Web ontologies to describe networks. These ontologies extend the Network Description Language (NDL [9, 6, 10]), which is based on the G.805 taxonomy of multi-layer network elements. NDL specifications (models) can represent network topologies at multiple levels and at varying levels of detail. Each entity in a model has a global name given by a URI, so a model can represent linkages to other models. In particular, a model describing a network can include links to named nodes in other networks.

We call our extended ontology suite NDL-OWL.[1] NDL-OWL extends NDL in two key ways. First, it uses the OWL standard to specify certain relationships among predicates, which enable us to specify key network management functions as declarative SPARQL queries over NDL-OWL models. One example is multi-layer path finding. Second, it adds a new class hierarchy and predicates to describe edge resources such as compute servers, virtual machines, cloud services, and block storage servers.

The user request to a controller (SM) is captured in an NDL-OWL *request model*, which specifies the abstract virtual topology graph and the types of the nodes and links, and any additional constraints. It may constrain the mapping to bind specific slivers in the topology to specific named aggregates, or to aggregates that offer specific capabilities named or described in the models. For example, these capabilities might include quality of service, measurement, monitoring, programmability, or actuation.

In planning a mapping the controller is assisted by one or more brokers. The brokers collect *domain models* of aggregates. There is one domain model for each aggregate, describing its slivering and related capabilities, edge interfaces to cross-domain links, and status. These models are described in more detail below. A controller can query a broker for the domain models of the aggregates the broker knows. Aggregates advertise their domain models to brokers they select. In general, not every ORCA broker has a model for any given aggregate, but the ExoGENI deployment has a central broker with a model for every ExoGENI aggregate.

An ORCA broker can also run resource allocation policies, e.g., based on capacity constraints, as described in previous work on SHARP resource peering [8, 11]. The domain models capture capacity constraints using OWL-based extensions for NDL-OWL.

4.2 Basic Network Model

We outline a vocabulary for networks that is consistent with NDL-OWL (G.805) and with standard modern IETF usage (e.g., RFC 4861), and is applicable to multi-layer networks. The building blocks of networks are *nodes*, which are devices, and *links*, which are network channels connecting nodes at some layer. A network attachment point between a node and a link is an *interface*. Every interface is attached to exactly one node and at most one link. Interfaces are *adjacent* if they are attached to the same node. Links are adjacent if they are attached to adjacent interfaces, i.e., they are linked to the same node. Nodes are adjacent if some link has an interface on both of them.

If a node has multiple interfaces, then it may function as a network element that passes network traffic among its interfaces. Without loss of generality, let us call such a node a *switch*. A switch supports some defined set of network functions and primitives, including standardized ways to connect its interfaces. A basic control primitive of a switch is to join a pair of interfaces to form a CrossConnect relationship traversing the switch. We call this operation joining or *stitching* the pair of adjacent interfaces and links attached to those interfaces. If an interface is not stitched (joined) then the attached link *terminates* at that interface and at its attached node.

Ethernet is the foundational layer-2 network technology visible to ExoGENI slices. Ethernet defines standard behaviors for links including automated MAC learning. An Ethernet link may be a *pipe* (path), which is a point-to-point bidirectional channel linking exactly two interfaces, or a *segment*, which links two or more interfaces to form a broadcast domain. The result of stitching two pipes is to create a longer logical pipe (tandem path) that traverses the switch. The result of stitching a pipe to a segment is to extend the segment to the interface at the other end of the pipe. A node may act as a *router*, which forwards packets among its interfaces based on higher-layer headers, or as a *host*, which sources and/or sinks packets but does not forward them.

A node implements some function or service defined by its *program*. Some nodes are programmable: the program can be changed. A programmable node has a named programming model that defines the set of legal programs for the node and the protocols for installing programs and changing them after the node is active.

[1] `http://geni-orca.renci.org/owl`

4.3 Virtual Cloud Networks

The terms and relationships outlined above comprise a general vocabulary for describing networked infrastructure at multiple layers. The NDL-OWL models use extensions of this vocabulary to represent physical substrate, domain models for resource providers, request models for virtual topology requests, and active virtual topologies. The models represent resources at each layer as instances of a common set of base classes for compute, storage, and network elements. The NIST definition of Infrastructure-as-a-Service implies this multi-layer class hierarchy.

Aggregate Managers (AMs) provide virtual infrastructure services (slivering services) to the controllers to allocate virtual nodes and program them as needed, and/or to allocate virtual links and stitch them as needed. Virtual nodes and virtual links are slivers: a controller may instantiate and manage them independently of other slivers. The domain models represent these slivering services as *adaptations* from resources at one layer onto resources of the same base class at the layer below, e.g., adaptations from the virtual topology layer onto the substrate components. Each adaptation defines limits on the number of instances that may be allocated on a given component. For example, the hosts in a virtual topology are virtual machines (VMs) allocated from a cloud site. The adaptation is to instantiate a VM over a hypervisor, such as KVM, running on a physical server host. We can represent "bare metal" provisioning as a null one-to-one adaptation from a virtual host to a physical host (e.g., "no-hype").

An ORCA AM is a generic ORCA server configured with local policies and plug-in handler scripts to control the aggregate's resources or invoke the underlying IaaS interfaces to create and manipulate slivers. The initial ExoGENI deployment includes four kinds of aggregates offering network services:

- **Cloud sites.** A cloud site AM exposes a slivering service to instantiate virtual machines (VMs) on its hosts and virtual links (VLANs) over its internal network for slice dataplanes. *Note: can it allocate a VLAN without any VMs?* An ORCA cloud AM includes a handler plugin to invoke an EC2-compatible IaaS cloud service and an *ImageProxy* server to obtain VM images named by URL in the request (Section 4.6). The handler also invokes a cloud service extension with a command set to instantiate interfaces on VMs when they are requested, stitch interfaces to adjacent virtual links, and configure interface properties such as a layer-3 address and netmask. This extension is known as "NEuca": we first implemented it for Eucalyptus, but we have since ported it to OpenStack. Both NEuca and ImageProxy can be used independently of ORCA.

- **Native ORCA-BEN circuit service.** The AM for the Breakable Experimental Network (BEN) offers a multi-layer circuit service. For ExoGENI it provides Ethernet pipes: point-to-point VLANs between pairs of named Ethernet interfaces (*named by URL?*) in the BEN substrate. It uses a suite of ORCA plugins, including NDL-OWL queries to plan the paths from a substrate model. The handler scripts for BEN manage paths by forming and issuing commands to switch devices over the BEN management network.

- **External circuit services.** For these services, the AM invokes a provider's native provisioning APIs to request and manipulate circuits. The AM authenticates with its own identity as a customer of the provider. A circuit is a pipe between named Ethernet interfaces on the provider's network. We have developed ORCA handler plugins for NLR's Sherpa FrameNet service and the OSCARS circuit reservation service used in ESNet and ION.

- **Static tunnel providers.** A provider can pre-instantiate a static pool of tunnels through its network, and expose them as VLANs at its network edge. The AM runs a simple plugin that manages an exclusive assignment of VLANs to slices, given a concrete pool of legal VLAN tags that name the prearranged static tunnels. This technique has proven to be useful for tunneling through campus networks and regional networks that do not offer dynamic circuit service.

Each virtual link instantiated from these aggregates appears as an atomic link (an Ethernet pipe or segment) in the slice's virtual topology. At the layer below, the aggregate may perform internal stitching operations to construct a requested virtual pipe or segment from multiple stitched links traversing multiple substrate components within the aggregate's domain. A virtual link may even traverse multiple providers if the host aggregate represents a multi-domain circuit service, such as ION.

Network providers and cloud providers typically do not expose their internal topology in their domain models. In our current implementation, the domain model abstracts each network domain as a single logical "big switch"; each interface on the switch connects to a named substrate link, which is typically a pipe to a named switch belonging to some other aggregate.

4.4 Cross-aggregate Stitching

A critical challenge in distributed cloud orchestration is stitching of slivers that cross domain boundaries. A key example is stitching links across multiple network aggregates. A cross-aggregate network stitch involves a link that crosses a border

between two adjacent providers. The border link connects a switch owned by one provider to a switch owned by the other. The network stitching problem is also faced by inter-domain circuit services [14].

ORCA provides a general facility for cross-aggregate stitching that applies for network stitching and other stitching use cases as well. This feature enables ORCA to orchestrate end-to-end stitching across multiple aggregates. It can incorporate inter-domain circuit services offered by third parties, as outlined above, but where "air gaps" exist between circuit services ORCA may bridge them by cross-aggregate stitching at exchange points.

For example, as discussed in Section 3.3, ExoGENI has an exchange aggregate at the StarLight exchange facility. The AM controls a single node (a switch), whose interfaces are connected to links to switches owned by other transport networks with a presence at StarLight, including NLR, Internet2, and ESNet/ANI. The switch is an Ethernet switch (a Cisco 6509) with a *swapping* capability, also called VLAN tag remapping or VLAN translation. To join two pipes to form a circuit spanning StarLight, two adjacent networks produce VLAN tags for the circuit on their adjacent links. The AM joins them by consuming the selected VLAN tags and remapping them at each interface (port) to the same internal tag.

All sliver stitching in ORCA follows this producer/consumer model. Creating a stitch is a two-step process. First, the controller allocates a sliver from the producer, which selects a tag or label for it, perhaps guided by a broker or other coordinator. The label is chosen from some namespace associated with the adaptation, such as the set of legal VLAN tags. Second, the controller passes the label securely in a request to join the labeled sliver to another adjacent sliver (the consumer). The controller queries the domain models of the aggregates hosting those slivers to infer which is the producer and which is the consumer, based on their capabilities (e.g., swapping). In this way, the controller constructs a directed dependency DAG as it plans a slice's virtual topology and the mapping to aggregates. The nodes of the DAG are slivers, and the DAG has directed edges from producers to consumers. The controller then traverses the DAG, instantiating slivers and propagating labels to their successors as the labels become available.

The BEN circuit service also has a swapping capability. At present, many of the external circuit services do not, but we hope and expect that they will add it in the near future. When two providers are adjacent at a link and neither has swapping capability, various standards and approaches exist to negotiate common tags. We view tag negotiation as a legacy function: when necessary, ExoGENI can bridge networks without swapping by configuring a broker to assign tags from a common preallocated range.

4.5 Programmable Nodes and Switches

If a node has no links terminating on it after all stitching operations are complete, then that node is not visible in the topology: virtual links may traverse it, but the node is hidden beneath them. Every remaining node in the virtual topology corresponds to a sliver, which exposes to a controller some protocol or API to control its functions and/or program it. For example, a slice controller programs a VM by specifying an image to boot: see Section 4.6 below.

VMs and some other nodes may be programmed to function as switches or routers. In particular, the ExoGENI switches are programmable with OpenFlow. Using our terminology, OpenFlow is a named capability of a programmable switch at the substrate layer or domain model, which appears as a node (a virtual OpenFlow datapath) in the slice's virtual topology. The slice controller may register (adopt) an OpenFlow controller to direct the slice's network traffic through the datapath. Multiple slices may adopt the same OpenFlow controller, enabling a third-party OpenFlow controller to manage network traffic belonging to other parties.

ExoGENI allows a slice to register a distinct OpenFlow controller at each aggregate. Without loss of generality we speak as if each slice adopts exactly one OpenFlow controller, and we assume that each distinct OpenFlow datapath resides entirely within some substrate switch.

If an ExoGENI slice registers an OpenFlow controller, then the datapath connects to that controller through a proxy (FlowVisor). The proxy permits the controller to install flow entries in the datapath to direct network traffic in the slice's dataplane. OpenFlow defines a flexible protocol for a controller to program its virtual OpenFlow datapaths. The controller may change its flow entries on the fly, and the datapath may fetch flow entries from the controller on demand (through the proxy) if there is no entry matching a given packet or frame. The proxy hides traffic belonging to other slices from the controller, and blocks the controller from directing traffic into other slices.

In ExoGENI, each site runs an OpenFlow proxy under the control of the local site AM. Every packet or frame passing through a switch in ExoGENI's VLAN-sliced network is bound to exactly one ExoGENI slice, which is given by the packet's VLAN tag. The tag corresponds to a virtual link (pipe or segment) allocated to the slice at some aggregate. Thus the local AM knows the mapping, and informs the local proxy which VLAN tags each controller is authorized to direct traffic for at each datapath. When an ExoGENI slice adopts a controller, the AM adds the slice's tags to the controller's tag set at the local proxy. When the slice is destroyed, the AM removes its tags from the tag set. Similarly, the AM may add a tag when it

creates a virtual link for a slice that has adopted the controller, and remove a tag when it destroys a virtual link. The proxy blocks a controller from observing or influencing packets outside of its set of authorized VLAN tags, or from setting a frame's VLAN tag to a value outside of its set.

OpenFlow offers a powerful platform for each controller to implement or redefine switch or router behaviors in software, within the confines of the ExoGENI slice dataplanes. An OpenFlow controller is empowered to arbitrarily redirect packets from one link to another within a virtual OpenFlow datapath, for all virtual links in its tag set, regardless of which ExoGENI slice owns the link. In particular, the controller must define rules to route or forward packets arriving on any virtual link that terminates in the datapath, or else the datapath must drop the packet. But OpenFlow controllers are restricted to direct traffic within their tag sets. In essence, they must "stay in their lanes", and ExoGENI enforces this isolation automatically. This model for OpenFlow differs from other GENI testbeds, which require manual approval for controllers and yet do not ensure this isolation. Note that an ExoGENI slice differs slightly from OpenFlow's usage of the term: OpenFlow considers a controller's "slice" to be the union of the tag sets for all of the ExoGENI slices that adopt the controller.

The current OpenFlow specifications leave much to the imagination of switch designers. OpenFlow switches differ widely in their capabilities and behaviors, and the firmware is evolving rapidly for switches on the market. We hope that hybrid OpenFlow switches will offer uniform behaviors that give controllers the freedom to extend network functions without burdening them to implement standard behaviors that are efficient and correct in hardware, such as simple VLAN stitching and swapping, or MAC learning on Ethernet segments. Ideally, packets traverse the datapath using standard Ethernet behavior unless flow entries change that behavior. A related question is whether a controller can define forwarding rules for its virtual topology without engaging the details of the underlying physical topology. This would be possible if hybrid switches allow flow entries to transform packets (e.g., to set the VLAN tag to some other value in the controller's tag set) and then forward the packet back through the normal processing pipeline, which selects output ports based on a packet's VLAN tag and MAC. Another option is to add virtualization support in the proxy.

4.6 Image Management

Another orchestration challenge for multi-domain networked clouds is uniform management of images, which are program files for nodes. With standard IaaS cloud stacks following the Amazon EC2 model, each cloud site requires some local user to pre-register each image with the site's cloud service, which generates an image token that is local to that site. Networked clouds require some uniform means to manage images across the member sites in the federation.

In our approach the creator of an image registers it at some shared image depository, and names it uniformly by its URL. ExoGENI runs an *ImageProxy* at each cloud site. ImageProxy is a stand-alone caching server that enables the cloud site to import images on demand from an Internet server. A slice controller's request to instantiate a virtual machine (VM) or other programmable node names an image *metafile* by a URL and content hash. The AM's cloud handler plugin passes the image URL and hash to the local ImageProxy server. The ImageProxy fetches and caches any image components specified in the metafile, if they are not already cached. It bundles the components into an image and registers it with the local cloud service (e.g., Eucalyptus), if the image is not already registered. It then returns a local token that the AM cloud handler may use to name the image to the local cloud service when it creates a VM instance to boot that image.

The ExoGENI deployment operates a central depository for images as a convenience to users. In principle, ImageProxy can work with any image server, including any of the various Virtual Appliance Marketplaces operating on the web. ImageProxy includes a BitTorrent client and supports BitTorrent URLs to enable scalable "content swarming" of images across many cloud sites.

The content hash is a basis for integrity checking and version checking. These hashes effectively make the image space content-addressable, which is useful for access control and flat naming for decentralized image storage schemes (e.g., DHTs). In the future we also plan to add support for additional safety checks based on signed assertions about the content hash. These include endorsement of image safety properties and types by entities in the authorization framework, and type-checking of images at the SM controller.

5. CONCLUSION

We believe that the right way to meet the goals of GENI is to embrace the challenge of federating diverse virtual infrastructure services and providers. This approach offers a path to leverage IaaS advances occurring outside of GENI and to allow GENI experiments to use infrastructure deployed outside of GENI. At the same time, it offers a path to bring GENI technology to bear on key problems of interest outside of the GENI community: linking and peering cloud sites, deploying and managing multi-site networked cloud applications, and controlling network functions in the cloud.

This paper describes the architecture and initial deployment of the ExoGENI testbed based on this approach. ExoGENI

offers a general architecture for federating cloud sites, linking them with advanced circuit fabrics, and deploying virtual network topologies on a multi-domain network cloud platform. The initial deployment combines off-the-shelf cloud stacks, virtual storage appliances, VLAN-capable network switches, integrated OpenFlow capability, and linkages to national-footprint research networks and exchange points with international reach.

The larger goal of the project is to extend the infrastructure-as-a-service vision to orchestrated control of pervasive virtualization: flexible, automated configuration and control of distributed cyberinfrastructure resources. ExoGENI and its control framework enable construction of elastic Ethernet/OpenFlow networks across multiple clouds and network transport providers. Once a provider allocates virtual resources to a customer slice, the customer has full control over how it uses those resources. For example, VM instances can be sized for an application and loaded with an operating system and application stack selected by the user. Built-to-order virtual networks can incorporate bandwidth-provisioned links and are suitable for flexible packet-layer overlays using IP or other protocols selected by the owner. IP overlays may be configured with routed connections to the public Internet through gateways and flow switches.

ExoGENI also serves important objectives beyond the network testbed community. The initial ExoGENI deployment is funded by GENI, offers standard GENI interfaces for tools to support research in network science and engineering, and leverages the GENI federation architecture to authenticate and authorize GENI-affiliated users. But ExoGENI can also serve a broader role as a model and platform for a wide range of deeply networked cloud services and applications. In particular, the ExoGENI architecture has a flexible structure for identity and authorization and it offers additional native interfaces and tools based on rich semantic network models. This paper only touches on those topics, which are the focus of companion papers.

6. ACKNOWLEDGMENTS

This paper should probably have about ten authors. We thank RENCI, NSF, IBM, and the GENI Project Office (GPO) at BBN for their support. Many colleagues at GPO and other GENI projects have spent a great deal of time to work through issues relating to ExoGENI and the material in this paper.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] I. Baldine. Unique optical networking facilities and cross-layer networking. In *Proceedings of IEEE LEOS Summer Topicals Future Global Networks Workshop*, 2009.

[2] I. Baldine, Y. Xin, D. Evans, C. Heermann, J. Chase, V. Marupadi, and A. Yumerefendi. The Missing Link: Putting the Network in Networked Cloud Computing. In *ICVCI: International Conference on the Virtual Computing Initiative (an IBM-sponsored workshop)*, 2009.

[3] I. Baldine, Y. Xin, A. Mandal, C. Heermann, J. Chase, V. Marupadi, A. Yumerefendi, and D. Irwin. Autonomic Cloud Network Orchestration: A GENI Perspective. In *2nd International Workshop on Management of Emerging Networks and Services (IEEE MENS '10) , in conjunction with GLOBECOM'10*, Dec. 2010.

[4] N. Blum, T. Magedanz, F. Schreiner, and S. Wahle. A Research Infrastructure for SOA-based Service Delivery Frameworks. In *Testbeds and Research Infrastructures for the Development of Networks and Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*, Washington DC, USA, April 2009. IEEE. ISBN: 978-1-4244-2846-5.

[5] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle. Dynamic Virtual Clusters in a Grid Site Manager. In *Proceedings of the Twelfth International Symposium on High Performance Distributed Computing (HPDC)*, June 2003.

[6] F. Dijkstra. *Framework for Path Finding in Multi-Layer Transport Networks*. PhD thesis, Universiteit van Amsterdam, 2009.

[7] E. Ford. From Clusters To Clouds: xCAT 2 Is Out Of The Bag. *Linux Magazine*, January 2009.

[8] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An Architecture for Secure Resource Peering. In *Proceedings of the 19th ACM Symposium on Operating System Principles*, October 2003.

[9] J. Ham, F. Dijkstra, P. Grosso, R. Pol, A. Toonk, and C. Laat. A distributed topology information system for optical networks based on the semantic web. *Journal of Optical Switching and Networking*, 5(2-3), June 2008.

[10] J. V. Ham. *A Semantic Model for Complex Computer Networks*. PhD thesis, University of Amsterdam, April 2010.

[11] D. Irwin, J. S. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum. Sharing Networked Resources with Brokered Leases. In *Proceedings of the USENIX Technical Conference*, June 2006.

[12] D. Irwin, L. Grit, A. Yumerefendi, and J. Chase. Underware: an exokernel for the internet? in submission, January 2007.

[13] M. F. Kaashoek, D. R. Engler, G. R. Ganger, H. M. Briceno, R. Hunt, D. Mazieres, T. Pinckney, R. Grimm, J. Janotti, and K. Mackenzie. Application Performance and Flexibility on Exokernel Systems. In *Proceedings of the Sixteenth Symposium on Operating Systems Principles (SOSP)*, October 1997.

[14] T. Lehman, X. Yang, C. P. Guok, N. S. V. Rao, A. Lake, J. Vollbrecht, and N. Ghani. Control plane architecture and design considerations for multi-service, multi-layer, multi-domain hybrid networks. In *IEEE INFOCOM Workshop on High Speed Networking*, May 2007.

[15] A. Yumerefendi, P. Shivam, D. Irwin, P. Gunda, L. Grit, A. Demberel, J. Chase, and S. Babu. Towards an Autonomic Computing Testbed. In *Workshop on Hot Topics in Autonomic Computing (HotAC)*, June 2007.