ExoGENI Design

Ilia Baldine (ibaldin@renci.org), Jeff Chase (chase@cs.duke.edu),
Chris Heermann (ckh@renci.org), Brad Viviano (viviano@renci.org),
Jonathan Mills (jonmills@renci.org), Victor Orlikowski (vjo@cs.duke.edu)

Version 1.01
Jan 10, 2012

Revision History

Date	Performed by	Notes
1.00	ibaldin	*Initial version*
1.01	Ibaldin	*Corrected figure 2 iSCSI connectivity to mgmt switch instead of head node. Added a table for port counts.*

Overview

ExoGENI is based on an extended Infrastructure-as-a-Service (IaaS) cloud model with coordinated provisioning across multiple sites and a high degree of control over intra- and inter-site networking functions. Most researchers will use a standard cloud computing stack to instantiate and manage virtual machines.

The sites federate by delegating certain functions for identity management, authorization, and resource management to common coordinator services offered by the ExoGENI federation; ExoGENI in turn delegates some of these functions to the GENI federation and to identity systems operated by participating institutions (Shibboleth/inCommon).

This structure enables a network of private ExoGENI IaaS clouds to operate as a hybrid community cloud. ExoGENI combines this multi-domain cloud structure with rich networking capabilities through direct Layer 2 site connectivity to national circuit backbone fabrics, linkages to other national and international networks, and OpenFlow-enabled dataplanes within each site.

It provides unified access to these services to enable users to construct virtual network topologies on demand. The testbed software supports GENI APIs and extended APIs to enable users to create and manage a virtual network as a slice of virtualized resources within the infrastructure.

Basic operations envisioned within ExoGENI include

- Provisioning individual compute resources (virtualized and bare-metal) from rack resources. Users will be able to supply boot images for virtualized instances; bare-metal instances will be limited to a few vetted images.
    - We plan to support Linux (bare-metal and virtual) and Windows (virtual, possibly also bare-metal). Other operating systems will require further study.
- Creating, modifying and destroying slices consisting of compute resources belonging to one or more racks, tied together with VLANs provisioned from rack switches and intermediate circuit providers.
- Create slices with user-driven packet forwarding control via OpenFlow. OpenFlow slices will be restricted to VLANs provisioned within and between the racks
    - Using OpenFlow, in the longer term we plan to offer an on-ramp feature for allowing external traffic (from campuses or other slices) to transit existing slices (slice-as-a-service)
- Create slices that combine ExoGENI resources with other GENI resources (e.g. meso-scale OpenFlow and WiMax testbeds; via our switch at StarLight facility we plan to create experimental L2 topologies that involve international partners)

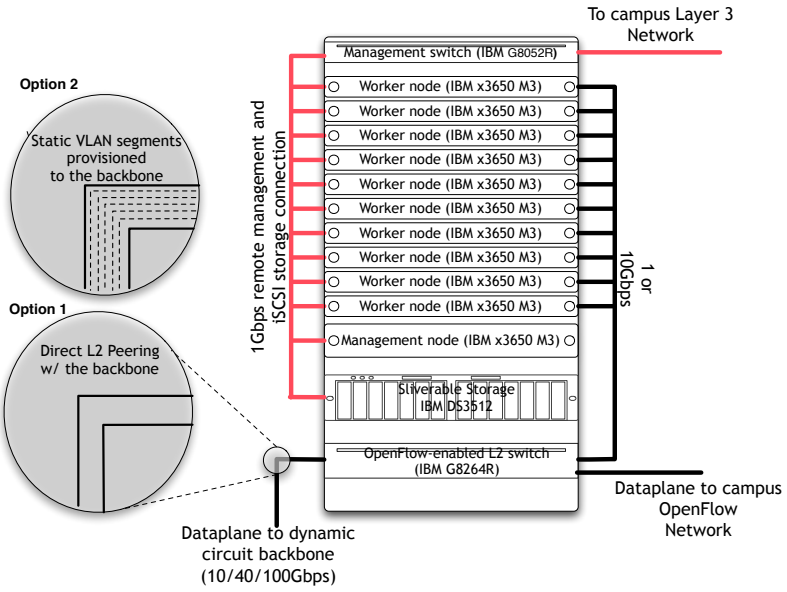
1. Hardware

Each ExoGENI site is a small cloud site capable of supporting about 100 virtual machines, based on a standard cloud cluster stack (e.g., Linux, KVM, and Eucalyptus). The site network links multiple interfaces for each cloud server and offers controlled IP connectivity through the host campus network.

Each site has a rack with multiple worker nodes, which are used as server substrate for provisioning experiments. A single management node in each site runs provisioning software and GENI services (like various CF interfaces, OF FlowVisor etc). Approximately 6TB local iSCSI storage appliance is provided at each site for image storage, measurement data storage and other experimental needs. All components are connected to a management switch, which combines the function of a rack backplane for iSCSI access and out-of-band management connectivity to RENCI for remote control and management needs (via a campus Layer 3 connection).

An IBM G8264R 10G/40G OpenFlow-enabled Layer2 dataplane switch with VLAN capabilities connects the nodes to the backbone (see below discussion of hybrid OpenFlow/VLAN operation). This switch serves as a termination point for incoming Layer 2 or Layer 1 dataplane connections (for supporting experiment traffic) that each campus presents to the racks.

Figure 1: ExoGENI Rack Overview



Management/head node will be equipped with a dual-socket 4 core 2.66GHz Intel Westmere CPU, 12G RAM and 8 1Gbps ports. Worker nodes will be equipped with dual-socket 2.66GHz Intel Westmere CPUs, each with 6 cores and 48GB of RAM. Each worker node will be equipped with two 1Gbps and two 10Gbps interfaces. Revision 1 of ExoGENI rack architecture assumes 10G interfaces are used for the dataplane (1G interfaces are used for management connectivity).

Figure 2: ExoGENI rack connectivity detail

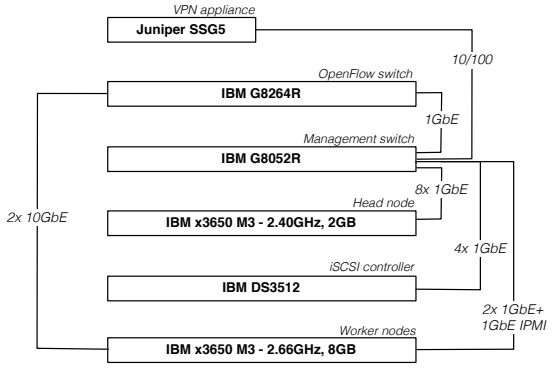
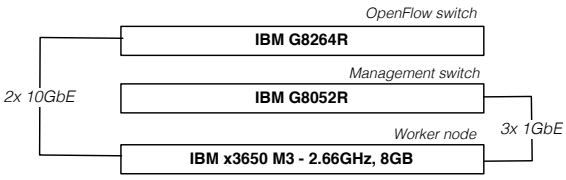


Figure 3: Worker node connectivity detail

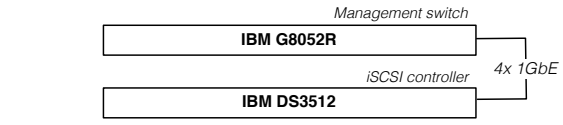


Notes:
1. Direct Attach SFP+ connectors provide 10GbE connectivity
2. Assign 2x 1GbE connection to Mngnt Vlan
3. Assign 1GbE RSA connection to Mngnt Vlan

Figure 2 shows intra-rack connectivity detail (shown is a half-rack for Rev 1 rack design based on IBM xM3 servers; Rev 2 racks will be based on IBM xM4 servers and will have the option of a less expensive 1G/10G dataplane OpenFlow switch). All elements of the rack connect to the management switch. Worker nodes are dual-homed into the management switch in a port-bonded fashion for improved throughput, they also dual-homed into the dataplane OpenFlow switch (these ports may be used for different types of slices – VLAN vs. OpenFlow). For additional remote management access, all nodes' IPMI interfaces are also connected to the management switch (See Figure 3).

The iSCSI storage (see Figure 4) will be connected through multiple bonded interfaces to the management switch to improve the throughput when transferring OS images or large datasets within the rack.

Figure 4: iSCSI storage connectivity detail



Notes:
1. Link aggregate 4 ports; assign to iSCSI Vlan

Local management can be accomplished using the KVM switch and an integrated console with keyboard provided in each rack.

Table below summarizes port occupancies on the management and dataplane switches:

Rack Element	Management Switch (max 48x1GbE ports)		Dataplane Switch (max 48x10GbE ports)	
	Ports	Description	Ports	Description
Head node	8	Bonded connections for improved throughput		
SSG5 security gateway	1	Connection to commodity L3 network		
iSCSI storage	4	Bonded 2-by-2 for improved throughput		
Dataplane OpenFlow Switch	1	Management connection into the switch		
10 Worker Nodes	3x10	2x1GbE bonded connection + IPMI connection	2x10	Dataplane connection
TOTAL	44		20	

2. External connectivity

2.1. Management plane connectivity

The connectivity of individual racks to the commodity Internet is envisioned to be flexible to accommodate possible restrictions on the availability of public IP addresses at individual sites. The connections to the commodity Internet via the campus network is expected to serve management access by staff as well as experimenters. The use of commodity network connections for experimental traffic will have to be a subject of explicit discussions between experimenters, operational staff and participating campuses (the preferred option is to use OpenFlow connectivity into the campus network to serve such needs and have a formal procedure for doing so).

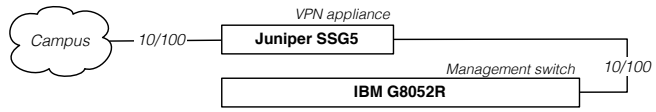
For remote management access, racks will be linked together into an IPSec-secured private network via a mesh of Juniper SSG5 gateways. These will provide secured remote access into each device within the rack. In addition, the head node will be connected to the campus network to provide a more limited access and in some cases proxy for compute resources created within the racks (VMs and bare-metal).

Several options, offering varying degrees of flexibility, are envisioned. They are outlined in the table below:

Option	Number of Public IPv4 addresses required from site	Description
A	2	Only the SSG5 and the head node have public IP addresses. All GENI resources are proxied/NATted by the head node onto a single IP address. This option may prove limiting for user tools that expect GENI slivers to be fully publicly addressable with the entire TCP/UDP port space exposed.
B	/24 or more	SSG5 and head node have a public IP, head node has a pool of public IP addresses to use for proxying resources within the rack.
C	/24 or more	SSG5, head node and elements of the rack have public IP addresses. GENI slivers (VMs and bare-metal instances) have public IP addresses. This option has the greatest performance along with significant security implications by allowing public visibility of rack elements.

Our current preference is for option (B) wherever possible. Figure 5 shows the detail of VPN connectivity. We prefer to have two redundant management connections into each rack – one via SSH through the head node and one via SSG5 (in case the head node fails).

Figure 5: VPN connectivity detail



Notes:
1. 10/100 upstream, campus connection requires public IP address

2.2. Data plane connectivity

ExoGENI sites will be interconnected by transit aggregates. Each transit aggregate offers a point-to-point Ethernet service among specified point-of-presence locations on the edge of the provider's network. Transit aggregates may offer dynamic VLAN services or pools of static VLANs connecting racks to other aggregates. ExoGENI will use dynamic circuit services offered by NLR (Sherpa), Internet2 (OSCARS) and ESNet (OSCARS) and native ORCA multi-layered circuit service in BEN (Breakable Experimental Network) built by the state of North Carolina and managed by RENCI. BEN has a direct connection to a 10Gbps NLR FrameNet with Sherpa service.

This capability is implemented through a combination of mechanisms. Sites will connect to the backbones through either a dedicated fiber or a static pool of VLANs that traverse campus networks and RONs to connect rack dataplane switches to the backbone interfaces (see Figure 1). The static VLANs coming out of individual sites can be thought of as available degrees of connectivity that can be dynamically connected to other sites using the national Layer 2 backbone. We will negotiate with sites and NLR on the details of VLAN tag assignments to the different sites.

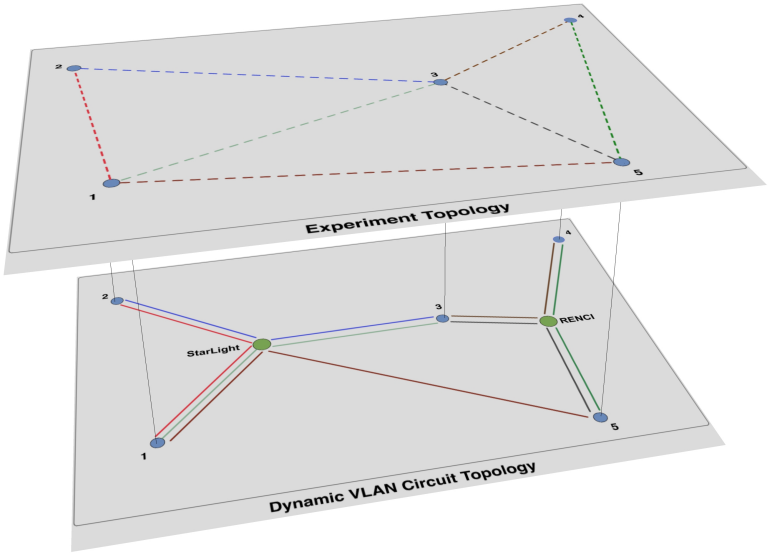
Since the static VLANs incident on individual ExoGENI sites have fixed tags, connecting them together requires a stitching facility that can take one VLAN tag and remap it onto another. In its initial stages, ExoGENI will rely on the already deployed ORCA capability to remap VLAN tags that exists at StarLight facility and at RENCI. We are in discussions with NLR to integrate VLAN translation into Sherpa feature-set as the permanent solution to this problem.

ORCA-BEN team has deployed a L2 switch to the StarLight facility in Chicago. This facility is a meeting point for many national fabrics. The deployed switch facilitates connectivity between members of GENI Cluster D. Some members of the cluster connect directly to NLR, while others could only negotiate dedicated connectivity to StarLight. ORCA-controlled switch at StarLight fills this 'gap' by dynamically establishing connectivity between different Cluster D members. The switch serves a

dual purpose of interconnecting multiple fabrics as well as performing VLAN tag remapping where necessary.

Figure 6 shows an example of how VLAN translation at fixed points in the network topology (StarLight and RENCI) can serve as an enabler for instantiating dynamic slice topologies for GENI and ExoGENI. In this figure, the desired experiment topology is shown in the top plane, while the circuits and their remapping is shown in the bottom plane. Using the RENCI switch at the StarLight facility, ExoGENI will be able to connect 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

Figure 6: Instantiating dynamic Layer 2 experiment topologies



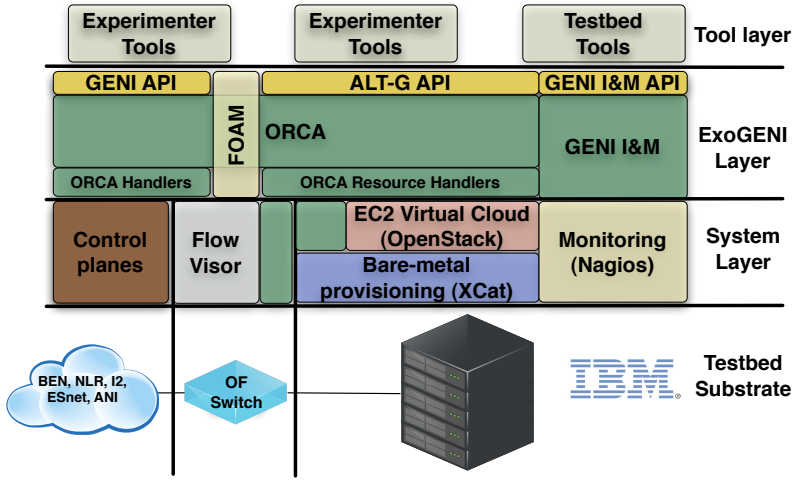
3. Software

3.1. Resource Provisioning

The testbed offers multiple levels of provisioning interfaces for user access and resource management, including standard cloud interfaces (EC2 and xCAT), OpenFlow, and layered GENI control and monitoring functions. One goal is flexible, automated deployment of customized software stacks on shared servers, with secure isolation and manageable quality of service. We expect the majority of users to rely on virtualization as this offers higher degrees of freedom in the choice of the OS, kernel and filesystem. But it will also have a bare-metal imaging capability based on xCAT provisioning tool (open-source xCAT, developed and maintained by IBM) with a small number of vetted images.

Figure 7 shows the ExoGENI software stack. For compute element provisioning it will use XCat (bare-metal instances) and OpenStack (virtualized instances). To support OpenFlow, an instance of FlowVisor will be running on the head node such that both FOAM and ORCA can communicate with it to create slices. ORCA will also directly communicate with the OpenFlow switch, once the hybrid mode (a mode combining a traditional switch operation with OpenFlow) is implemented by the vendor.

Figure 7: ExoGENI software stack



The interface to control planes like OSCARS and Sherpa will not be distributed, but rather will be run from a single reliable server at RENCI (see Section 3.3).

3.2. Resource Monitoring

Nagios – an established, versatile open-source monitoring software suite will be used as a low-level monitoring solution for operations staff (it can also be used to feed GENI Instrumentation and Measurement). We have demonstrated an initial implementation that showed how a number of Nagios instances (one from each rack) can be aggregated into a single view for operations staff, in order for them to monitor the health of individual resources and instantiated slivers (VMs) in each rack in an easy-to-understand fashion. This model permits RENCI staff (and potentially GPO staff and GMOC) to view the health of each rack, while on-site staff can view the health of just their rack.

Nagios collects information on most common performance metrics (CPU, memory, disk utilization, network traffic, temperature readings). The IBM x3650 M3 server family has extensive probes for server health monitoring (including power consumption) which we will work to enable and expose via Nagios and to GENI users.

While at the time of the writing the final decision has not been made, we are considering adding smart PDUs to racks to ensure we capture the power consumption of those elements in the rack that do not have internal mechanisms to do so. As this represents an additional cost per rack, this decision awaits the initial trial period with the first rack deployments.

3.3. Software components

ExoGENI may be viewed as a group of independent resource providers within a larger GENI federation. A resource provider is represented by an ORCA AM (not to be confused with GENI AM; ORCA AMs currently do not expose GENI AM API, as ORCA internal interfaces rely on tickets – a feature currently in discussion for future GENI API). In order to achieve the functionality required for creating complex slices with resources from multiple providers, a coordinating function is needed, which in ORCA is fulfilled by a Broker actor. ORCA AMs delegate some portion of their resources to one or more brokers. The users interact with ORCA via the SM actor, which exposes the GENI AM API as well as ORCA's native user-oriented XMLRPC interface. SMs receive tickets from brokers for the needed resources that they redeem with ORCA AMs to instantiate those resources.

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. An ORCA cloud AM includes a handler plugin to invoke an EC2-compatible IaaS cloud service such as Eucalyptus or OpenStack. The handler also invokes an extension to the cloud interface 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 also ported it to OpenStack. For bare metal provisioning we will rely on xCAT – xCAT plugin for ORCA is currently under development.

• 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 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 plugins for NLR's Sherpa FrameNet service, Internet2 ION, and the OSCARS circuit reservation service used in ESNet.

• 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.

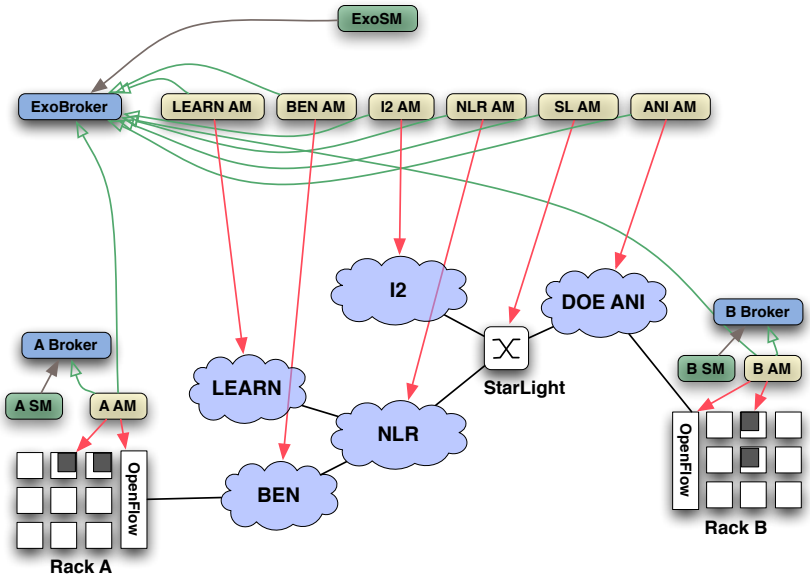
Each virtual link instantiated from these aggregates appears as an atomic link (an Ethernet pipe or segment) in the slice's virtual topology. At the layer below, the aggregate may perform internal stitching operations to construct a requested virtual pipe or segment from multiple stitched links traversing 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 OSCARS.

Figure 8 demonstrates the proposed ExoGENI ORCA software deployment. Each rack will have its own ORCA AM that delegates resources to the local broker (for coordinating intra-rack resource allocations of compute resources and VLANs) and to the global broker (ExoBroker), which coordinates allocation for slices spanning more than one rack. Each rack will also run an ORCA SM that will expose GENI AM API to allow the allocation of resources from the rack. An ORCA AM running on the rack can stitch resources within one rack, however any stitching of resources external to the rack has to be done by GENI tools externally.

ORCA has demonstrated a powerful slice embedding and stitching engine that can take under-specified (unbound or partially bound topologies) and create global slices across multiple network providers. In order to use this engine, an additional global SM (more can be deployed later) will be deployed. This ExoSM will use the global ExoBroker to acquire resources from multiple racks as well as intermediate network providers in a coordinated fashion (including pre-negotiated VLAN tag assignment) and stitch them together into a single slice.

A global broker will also receive delegations of resources from ORCA AMs controlling the intermediate network providers like Internet 2, NLR, ANI, LEARN and BEN. These resources will be used by ORCA stitching engine via ExoSM to create global slices. The ExoBroker and ORCA AM actors responsible for the network providers will run in VMs on RENCI-owned VMware cluster with hourly backup and high-degree of hardware redundancy.

Figure 8: ORCA software deployment in ExoGENI



Each rack will include several additional components external to ORCA but integrated into its operations:

- ImageProxy – is a component that helps distribute user-created filesystem/kernel/ramdisk images to different sites. Today's cloud software (Eucalyptus, OpenStack, xCAT) is built on a single site model in which each site has a separate image repository from which compute instances are booted. When multiple sites are involved, a user must somehow specify which image is to be used and the image must be registered with the selected sites. ImageProxy fulfills this function by allowing the user to specify a URL of the image descriptor meta-file and its hash (for security purposes). When ORCA processes a slice request and decides on a slice binding to particular sites, the ImageProxies at those sites download and register the user image based on a URL of the metafile so the user's image can be booted on compute slivers within the slice. More on ImageProxy can be found here: https://code.renci.org/gf/project/networkedclouds/wiki/?pagename=ImageProxy
- Shorewall DNAT Proxy – is a component that helps sites with limited public IP address availability to proxy services on TCP/UDP ports running on compute slivers using a single IP address. Its operation is fully integrated with ORCA and is configurable by the site operator. More information can be found here: https://geni-orca.renci.org/trac/wiki/shorewall-with-orca
- FlowVisor – the software provided by Stanford for creating OpenFlow slices. ORCA will communicate with FlowVisor directly via its XMLRPC interface.

ExoGENI will also run several global components:

- ORCA Actor Registry (running on https://geni.renci.org:12443/registry/actors.jsp) - a secure service that allows distributed ExoGENI ORCA actors to recognize each other and create security associations in order for them to communicate. All active actors are listed in the web view and an actor requires ExoGENI operations staff approval in order to start communicating with other actors.
- RSpec/NDL conversion service (running on http://geni.renci.org:12080/ndl-conversion/convert.jsp and http://geni.renci.org:12080/ndl-conversion/r-convert.jsp) used by all ORCA SMs to convert RSpec requests to NDL and NDL manifests into RSpec.

These components are running on a VM in a production VMWare cluster at RENCI, that is snapshotted hourly (for backup) and has high degree of hardware redundancy.

4. GENI Integration

4.1. APIs and RSpecs

Since internal ORCA inter-actor APIs (APIs used to communicate between ORCA AMs, brokers and SMs) are ticket-based and operate using signed SOAP messages and not XMLRPC, it is ORCA SMs that create the GENI AM API compatibility layer by implementing GENI AM API XMLRPC interface for the users, while speaking ORCA APIs on the back end. They also perform the necessary RSpec-to-NDL conversions between GENI RSpec and ORCA's internal semantic NDL-OWL resource representations. The conversion is hosted as a standalone stateless XMLRPC service described in Section 3.3.

This approach allows ExoGENI to evolve its architecture while maintaining compatibility with the GENI standards.

ORCA's GENI AM API implementation mostly conforms to GENI AM API v1, with the exception of NDL-OWL ad conversions to RSpec and minor return parameter differences. We plan to have a GENI AM API v2 compliant implementation, including the ad conversion, available by GEC13.

For interoperability with the traditional model of using OpenFlow in GENI, each rack will run and instance of FOAM. Manual approval of FOAM slices will be performed by the GPO or their delegate. Section 5.4 describes ExoGENI model for using OpenFlow without requiring manual opt-in approval.

4.2. 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 is a link connecting 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.

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, 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, with directed edges from producers to consumers,

as it plans a slice's virtual topology and the mapping to aggregates. It 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, some 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.

ORCA produces RSpec manifests consistent with the prototype stitching schema, that should be sufficient for external tool stitching. Inter-domain tag negotiation to the best of our knowledge is currently not part of the GENI AM API and is not supported by the GENI AM API implementations in ORCA SMs. ORCA's internal stitching engine follows what is commonly referred to in GENI as a 'tree' model (actually a 'forest') with an entity external to ORCA AMs (an ORCA SM) determining acceptable common tags and reserving them with one or more brokers, and utilizing tag translations in those domains that permit this function.

4.3. Monitoring/GMOC interface

Monitoring within ExoGENI is layered, just like resource provisioning – at the bottom there is a hardened industry-standard open-source solution (Nagios) that monitors the health of the infrastructure and some of the sliver characteristics (like VMs coming and going and their state as visible through the hypervisor). It should be possible to construct an interface from Nagios to support GMOC's internal interface.

To achieve better visibility and fidelity a feed from ORCA is needed notifying of slice creation/teardown/modification events as well as any decisions the emebedding and stitching logic has made in connection to those. In our conversations with GMOC we have proposed using a pubsub interface based on XMPP XEP-0060 for that purpose as it would permit GMOC to collect and correlate necessary data while at the same time opening the monitoring platform to other uses. At this point we need to complete the pubsub implementation on our end.

Finally, for improved monitoring of virtualized compute instances, better tooling is required as the ability of basic hypervisors to see inside the VM is limited, and forcing users to install specific packages onto their images so their instances can be monitored is not a practical option as it is hard to enforce and is intrusive and open to software versioning conflicts.

We have been in discussions with the University of Alaska at Fairbanks team (Brian Hay and his students), regarding using the VIX (Virtual Machine Introspection http://assert.uaf.edu/geni/vmi.html) tool in ExoGENI and have outlined the preferred way to proceed, which was to integrate VIX with libvirtd – an open-source package sitting atop hypervisors like Xen and KVM, which serves as a basis of many cloud software stacks, including Eucalyptus and OpenStack. We would then work to integrate VIX monitoring information into Nagios and ORCA, as needed, to expose the additional monitoring information to GMOC and GENI users.

4.4. Reachability/Continuity testing

ExoGENI will be a distributed facility that relies primarily on dynamic circuits for intra-slice connectivity. With many disparate provisioning mechanisms in play it will be essential to perform periodic reachability/continuity tests for both the management and the data planes. Nagios will provide probes for basic reachability testing over the management plane that will inform operators of any commodity L3 connectivity problems. The testing for L2 dataplane connections is more involved and will require slices to be periodically created within and between racks to verify the operations of the rack elements and intermediate network providers.

This is the evolving part of ExoGENI operations. It would be reasonably straightforward to create some scripts that submit periodic slice requests and do a very basic verification of slice functionality. If more sophisticated tools are available, we're open to adopting them into ExoGENI operations.

4.5. User authorization

ExoGENI delegates user authorization to GENI (GPO) and its current authorization policy as implemented in ORCA SM matches that implemented in ProtoGENI (the code was written by Prateek Jaipuria – a Duke student who interned at the GPO over the summer of 2011). The current model presumes that users will use inCommon to get their GENI credentials that can be verified.

We believe the future of GENI lies in adopting ABAC (and its libabac implementation), however this decision has not been made yet. ORCA already demonstrated modifications necessary in its SM code to use libabac and Duke students have been developing a convenient certificate storage facility, which can keep user credentials and permit libabac to perform its inferences in order to determine the validity of attribute assertions. These components can be added to ExoGENI if ABAC becomes part of GENI operations.

At present the security procedures associated with storing user private keys and certificates are not discussed since under the current GENI authorization it is not necessary in ExoGENI and is the responsibility of GENI central.

5. How users will use ExoGENI

5.1. Getting compute resources

ExoGENI will offer two basic classes of compute slivers: VMs and bare-metal servers. VMs will come in a number of subclasses, differing by the number of CPU cores, amount of RAM and disk available to them (VM classification will be based on the EC2 instance type hierarchy adapted to the ExoGENI hardware and OpenStack). The basic hierarchy has been agreed on as part of the compute ontology definition done over the summer of 2011. The class.subclass will be specified as part of *sliver_type* element of node RSpec.

Since ExoGENI permits users to use the images of their own creation in slices (rather than using only a set of pre-approved images pulled from a common repository), the images must be specified differently: they must be specified by a URL of an ImageProxy metafile and its hash (used to detect malicious modifications to a metafile). The current RSpec convention used by ORCA is to use *disk_image* element of RSpec and to place the URL in the *name* attribute, while the hash goes into the *version* attribute. (More information on RSpec/NDL-OWL conversion conventions can be found here:

https://geni-orca.renci.org/trac/wiki/orca-and-rspec).

When getting a bare metal instance ExoGENI will likely use 'raw-pc' type name (we can also adopt 'x3650.m3' and 'x3650.m4') and conform to existing RSpec conventions used in protogeni for specifying the name/version of a pre-approved image for bare-metal slivering that must exist in each rack. These images will be periodically synchronized out-of-band between the racks and a small central repository.

Users will be able to submit their slice requests using GENI tools (and GENI AM APIs) to specific SMs or they will be able to use Flukes – ORCA's GUI that operates on native interface and semantic web resource representations. Both types of requests are processed in the same pipeline within the SM such that similar authorization policies will be respected and they will operate on the same credential sets.

ORCA supports post-boot script mechanism, similar to the *execute* service specified in RSpec. When using ORCA's native interface, this script can be templated for use with nodegroups – large groups of

nodes sharing common characteristics that can be (but don't have to be) split between provider sites. This feature has no RSpec equivalent.

More information on ORCA native interface and Flukes is available here:

https://geni-orca.renci.org/trac/wiki/flukes

5.2. Intra-rack slices

The SM within each rack can allocate resources only from this rack, however they will still be stitched together (i.e. compute slivers will be connected to each other by VLANs in the user-specified topology within the rack). The SM will then produce an RSpec manifest with stitching schema extensions that can be used by external tools to operate on the slice or, potentially, stitch it to other slivers via e.g. FOAM.

5.3. Inter-rack slices

Users will be able to request intra-rack slices by submitting their requests to the ExoSM. This SM has visibility into a wider pool of available resources and will perform necessary binding of request to available substrate (in case that the user topology is unbound or only partially bound). In case of an explicitly bound topology ExoSM will simply attempt to fulfill the request as stated, performing the necessary stitching on the way if resources are available.

This request can be submitted either through the GENI compatibility interface on ExoSM or via Flukes and ORCA native interface.

The alternative to using ExoSM is to use GENI tools to do the stitching external to ORCA using the manifest produced by the individual rack SMs.

5.4. OpenFlow slices

Due to low penetration of OpenFlow in core circuit network providers and lack of quality of service support (i.e. performance isolation) in currently available implementations, ExoGENI differs from current GENI practice with respect to the usage model for OpenFlow networks. In GENI, an OpenFlow datapath is viewed as a separate aggregate that ``allocates'' the right to direct network traffic flows matching some set of specified patterns, which are approved manually by an administrator.

In ExoGENI, OpenFlow is an integrated capability available from some ExoGENI network providers, including the sites available on top of basic VLAN slicing mechanism, rather than a distinct aggregate. ExoGENI slices may designate OpenFlow controllers to direct network traffic within the virtual network topology that makes up the slice's dataplane. The ExoGENI aggregates authorize the controller automatically based on their assignments of virtual network resources to the slice's virtual topology. 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.

The user will specify the URL of their OF controller in the RSpec request (the specific extension to RSpec to support it is under discussion). A slice will be created with typically a single broadcast domain for an entire slice (with VLAN tag remapping as needed to connect slivers from multiple ExoGENI domains). The user's controller will be able to perform forwarding of packets using OpenFlow rules in OF datapaths included in the slice by matching on header fields other than the VLAN tag, which will be 'locked in' to this slice via FlowVisor running in the rack. ORCA will communicate with FlowVisor via its XMLRPC interface to communicate slice information (ports and vlan tags involved in the slice) and controller URL.

It is important to understand that all slices in ExoGENI are based on VLANs. Some slices have OpenFlow enabled in them for explicit user control of packet forwarding within those vlans. This behavior is optional and is specified in the request.

5.5. Slices with components from other GENI resources

There are two ways in which other GENI resources can be stitched to slices within ExoGENI: over commodity Internet, which provides no isolation from other slices. Since ExoGENI slices have management network access via the commodity Internet, this is the default behavior.

The better alternative is to connect via a dataplane to a resource on some VLAN. In this case ExoGENI/ORCA will need to know the provider, VLAN tag on which the resource is available and, possibly, a port number and network element from which this VLAN can be accessed. Using this information ExoGENI staff can create a new NDL-OWL domain descriptor to enable ORCA's stitching engine to perform pathfinding and stitching to this resource.

This has been recently demonstrated at SC11 where ORCA was used to stitch a dynamic slice of cloud resources to the DOE Hopper super-computer located at NERSC. More information can be found here:
https://ben.renci.org/index.php?option=com_docman&task=doc_download&gid=25&Itemid=63 .

Same should be possible for meso-scale VLANs as long as proper points of entry (provider, network elements, ports) onto them are identified.

5.6. Slice Performance Isolation

ExoGENI relies on industry- standard mechanisms for resource provisioning (KVM-based virtual machines, bare-metal instances, quality-of-service provisioned VLANs) which all have well-documented performance isolation properties.

Several practical factors may serve to weaken those properties:

- Using pools of static best-effort VLANs for connecting a rack to a dynamic circuit provider. In our experience, while pools of static VLANs provide a practical means of connecting a rack to other racks (or intermediate service providers), the reality is that these VLANs are usually best-effort, since campus and RON providers don't want to permanently reserve bandwidth allocated to each VLAN, since it may remain unused. Even if a group of VLANs is given some portion of link bandwidth, they may still compete with each other for bandwidth. This means that portions of the intra-rack slices relying on such a pool of VLANs may loose performance isolation and retain only namespace isolation, thus affecting the repeatability of experiments. The solution is to opt as much as possible for a direct (dark-fiber) connection between the rack's dataplane switch and a dynamic circuit provider, which can then provide VLANs with reserved bandwidth.
- The current implementation of the OpenFlow in the IBM G8264R switch is high-performance and does flow matching and forwarding on the ASIC at 10Gbps speeds with thousands of flows simultaneously, however it is limited in that it does not support a hybrid mode in which some of the ports can operate in OpenFlow mode, while the rest of the ports operate as regular switching ports (hybrid mode is not part of OF 1.0 spec). Hybrid mode will become available as a firmware upgrade from IBM later in 2012. The OpenFlow implementation currently does not provide any performance isolation between flows (creating a virtual interface with a queue is an optional OF 1.0 spec feature and is not supported). When hybrid mode becomes available, we will use the fact that each worker node is dual-homed into the dataplane OpenFlow switch to connect one of the worker 10G interfaces to a port with OpenFlow enabled and the other to the port with traditional VLAN switching. This way if a user wants a slice without OpenFlow, we can provide strong performance isolation by provisioning a traditional QoS-enabled VLANs with specified bandwidth in the switch to the compute instances running within the node. If a user wants an OpenFlow slice, we will provision it on the OpenFlow-enabled port. We will continue working with the vendor on future implementations through the lifetime of ExoGENI to improve isolation properties in OpenFlow slices as the OpenFlow spec evolves and new implementations become available.

6. Management access and security procedures

6.1. Operations staff authorization

All elements in each rack will be configured to use a redundant LDAP server hosted at RENCI to authorize operations personnel to login to network elements. If needed, we can add RADIUS support for network element authorization (management and dataplane switches), as it is more commonly used with network equipment. This is the mode in which individual BEN sites operate today under RENCI control. Operations staff will belong to one or more authorization groups depending on their level of access (per rack or rack element type). Users will be added or removed to this LDAP database only the RENCI operations personnel.

Each element will also have a default admin login entry that will be known to a small number of operations staff for emergency logins.

6.2. RENCI operational access

For management access the operations staff will use the secure network built out of SSG5 gateways in each rack and use Juniper VPN software. Another way for operations staff to login will be to the head node using SSH. These two methods provide two redundant ways of accessing rack elements in case the other method fails (head node failure [more likely] or the SSG5 gateway failure [less likely, since it is a diskless device]).

From there the operations staff will be able to access all other rack elements to configure or check their health status.

As described above, Nagios will be used as a basis for low-level monitoring of rack element health and access to the Nagios portal reflecting the health of racks and their elements will be controlled using the same LDAP servers described in the previous section.

6.3. Site operational access

As a starting point, site staff will have limited access into the rack via SSH into the head node with some limited administrator privileges asserted via LDAP group membership. As the testbed evolves we will work to increase the level of privilege for local administrators.

Local administrators will also have emergency access to rack elements via a built-in KVM (Keyboard/Video/Mouse) console. This will require a physical access to the rack. This may be required in case of hardware failures to work with vendor representatives to diagnose and replace failed rack elements.