G E N I

Global Environment for Network Innovations

ProtoGENI Control Framework Overview

Document ID: GENI-SE-CF-PRGO-01.3

January 14, 2009

Prepared by:
The GENI Project Office
BBN Technologies
10 Moulton Street
Cambridge, MA 02138 USA

Issued under NSF Cooperative Agreement CNS-0737890

TABLE OF CONTENTS

1 Document Scope

This section describes this document's purpose, its context within the overall GENI document tree, the set of related documents, and this document's revision history.

1.1 Purpose of this Document

This document provides an overview of the ProtoGENI control framework being implemented for Spiral 1, for use in Cluster C. It is a DRAFT, to be used for discussion in the GENI Facility Control Framework working group. (Note: A review of this document by the ProtoGENI team is underway, but has not yet been completed.) It provides a description of the ProtoGENI control framework structure, a summary of how it meets the requirements as presented in the "GENI Control Framework Requirements", and a view of its implementation at the start and the finish of Spiral 1.

Some of the material in this document is drawn from the GENI System Requirements document.

Some of the material in this document is drawn from the GENI System Overview document.

Some of the material in this document is drawn from the GENI Control Framework Requirements document.

Some of the material is drawn from a draft "Slice-based Facility Architecture (SFA)" document, dated August 8, 2008, and edited by Larry Peterson. In particular, Larry Peterson and the others who contributed to the SFA document are recognized for their significant contributions, including: Scott Baker, Ted Faber, Jay Lepreau, Soner Sevinc, Stephen Schwab, Leigh Stoller, Robert Ricci, and John Wroclawski.

Some of the material in this document was provided by Rob Ricci and Leigh Stoller.

1.2 Context for this Document

Figure 1-1. below shows the context for this document within GENI's overall document tree.

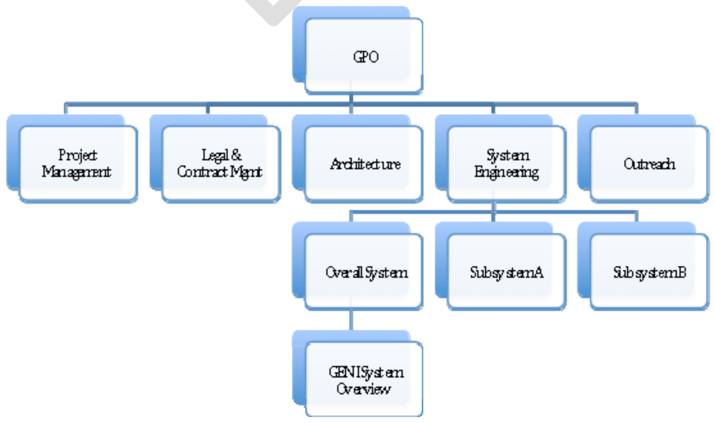


Figure 1-1. This Document within the GENI Document Tree.

1.3 Related Documents

The following documents of exact date listed are related to this document, and provide background information, requirements, etc., that are important for this document.

1.3.1 National Science Foundation (NSF) Documents

Document ID	Document Title and Issue Date
N / A	

1.3.2 GENI Documents

Document ID	Document Title and Issue Date
GENI-SE-SY-RQ-01.4	GENI System Requirements, September 18, 2008 http://www.geni.net/docs/GENI-SE-SY-RQ-01.7.pdf
GENI-SE-SY-SO-01.5	GENI System Overview, September 19, 2008, http://www.geni.net/docs/GENISysOvrvw092908.pdf
GENI-SE-CF-RQ-01.x	GENI Control Framework Requirements, November 21, 2008, http://geni.bbn.com:8080/docushare/dsweb/Services/Document-1234

1.3.3 Standards Documents

Document ID	Document Title and Issue Date
N / A	

1.3.4 Other Documents

Document ID	Document Title and Issue Date
GDD 06-10	"Towards Operational Security for GENI," by Jim Basney, Roy Campbell, Himanshu Khurana, Von Welch, GENI Design Document 06-10, July 2006. http://www.geni.net/GDD/GDD-06-10.pdf
GDD 06-23	"GENI Facility Security," by Thomas Anderson and Michael Reiter, GENI Design Document 06-23, Distributed Services Working Group, September 2006. http://www.geni.net/GDD/GDD-06-23.pdf
N/A	"GMC Specifications," edited by Ted Faber, Facility Architecture Working Group, September 2006. http://www.geni.net/wsdl.php
GDD 06-24	"GENI Distributed Services," by Thomas Anderson and Amin Vahdat, GENI Design Document 06-24, Distributed Services Working Group, November 2006. http://www.geni.net/GDD/GDD-06-24.pdf

GDD 06-38	"GENI Engineering Guidelines," edited by Ted Faber, GENI Design Document 06-38, Facility Architecture Working Group, December 2006. http://www.geni.net/GDD/GDD-06-38.pdf
GDD 06-42	"Using the Component and Aggregate Abstractions in the GENI Architecture," by John Wroclawski, GENI Design Document 06-42, Facility Architecture Working Group, December 2006. http://www.geni.net/GDD/GDD-06-42.pdf
N/A	"Slice Based Facility Architecture," v1.10, August 8, 2008, by Larry Peterson, et.al. http://groups.geni.net/geni/attachment/wiki/GeniControlBr/v1.10%20%20080808%20%20sfa.pdf
N/A	"Large Scale Virtualization on the Emulab Network Testbed," June, 2008, by Mike Hibler, et.al. http://www.cs.utah.edu/flux/papers/virt-usenix08-base.html
N/A	"Implementing the Emulab-PlanetLab Portal: Experience and Lessons Learned," December, 2004, by Kirk Webb, et.al. http://www.cs.utah.edu/flux/papers/portal-worlds04-base.html
N/A	"Decentralized Trust Management," 1996, by Matt Blaze, et.al, AT&T Research. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.6276
N/A	"Compliance Checking in the PolicyMaker Trust Management System," 1998, by Matt Blaze, et.al, AT&T Research. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.2525
N/A	"The Role of Trust Management in Distributed Systems Security," 1999, by Matt Blaze, et.al, AT&T Research. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.7726

1.4 Document Revision History

The following table provides the revision history for this document, summarizing the date at which it was revised, who revised it, and a brief summary of the changes. This list is maintained in reverse chronological order so the newest revision comes first in the list.

Revision	Date	Revised By	Summary of Changes
01.1	12/5/08	H. Mussman	Completed draft, for limited review, based on material provided by Rob Ricci and Leigh Stoller.
01.2	12/15/08	H. Mussman	Updated draft to follow structure from Control Framework Requirements document.
01.3	1/14/09	H. Mussman	Updated with small changes.
01.4			

2 GENI System Overview

2.1 Major Entities and their Relationships

Figure 2-1 presents a block diagram of the GENI system covering the major entities within the overall system. Optional (but desirable) parts are shown "grayed-out." See the GENI System Overview document at http://www.geni.net/docs/GENISysOvrvw092908.pdf for more details.

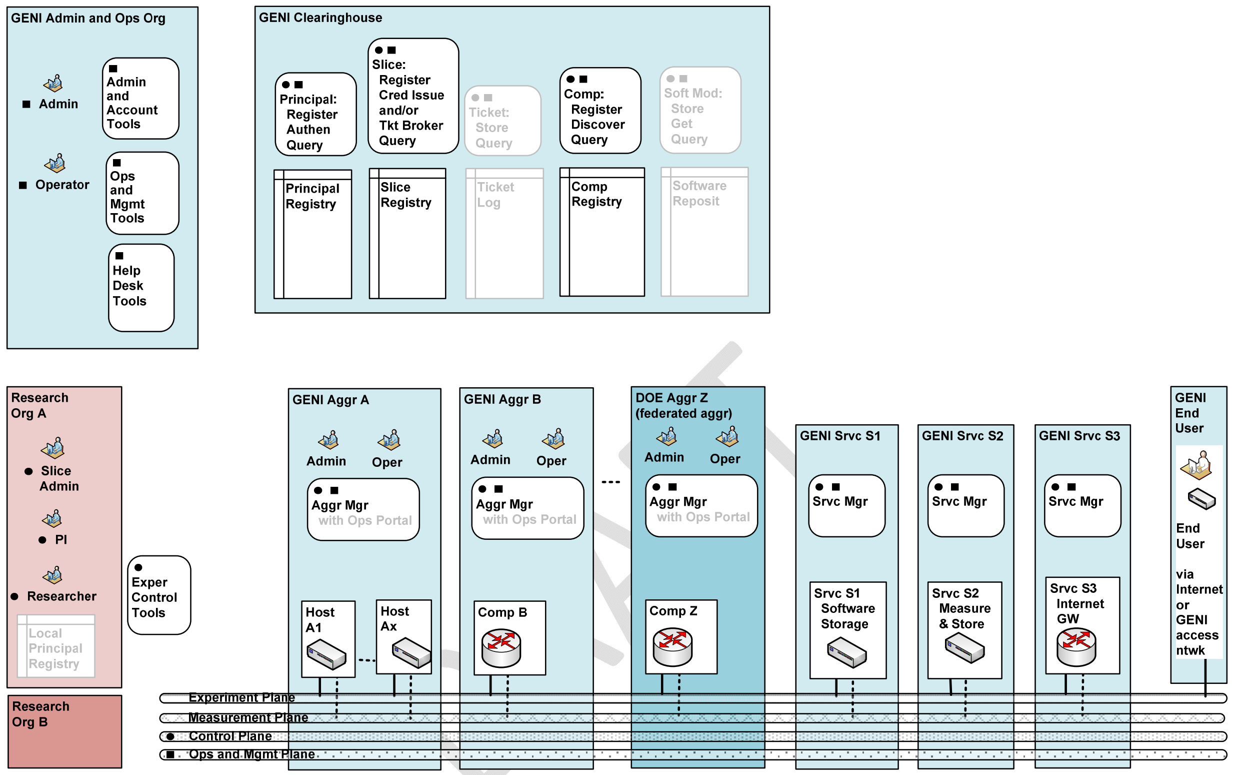


Figure 2-1. GENI System Diagram.

2.2 Federated Suites

Figure 2-2 provides a system diagram illustrating federation between one GENI suite and another. As a hypothetical example, it depicts federation between a US-based GENI suite and a compatible suite in the European Union (EU).

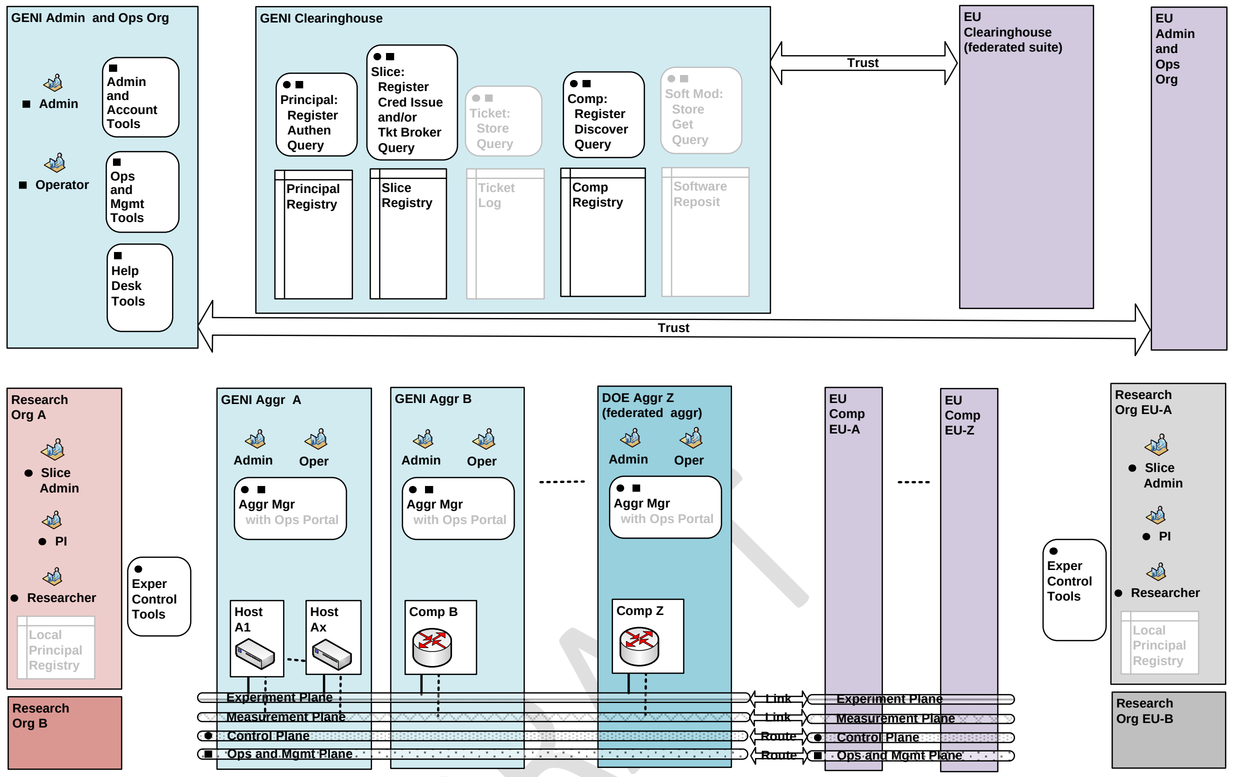


Figure 2-2. System Diagram with Federated Infrastructure Suites.

2.3 Slices

Figure 2-3 shows two researchers from different organizations managing their two experiments in two corresponding slices. Each slice spans an interconnected set of slivers on multiple aggregates and/or components in diverse locations. Each researcher remotely discovers, reserves, configures, programs, debugs, operates, manages, and teardowns the “slivers” that are required for their experiment. Note that the clearinghouse keeps track of these slices for troubleshooting or emergency shutdown.

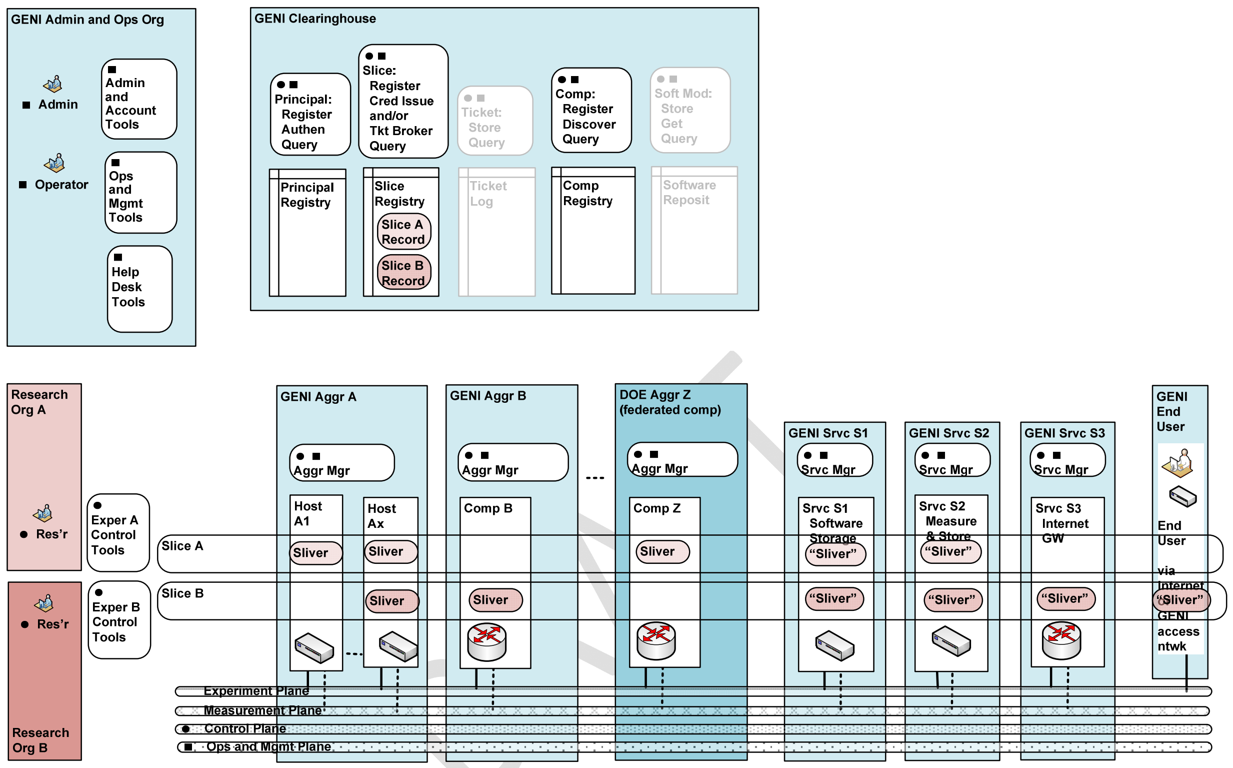


Figure 2-3. Two GENI Slices.

An aggregate manager a) interacting with the researcher (or her proxies) via the control plane and b) configuring the devices over internal interfaces establishes Slivers. Components may be virtualized, and can thus provide resources for multiple experiments at the same time, but keep the experiments isolated from one another. In addition, each slice requires its own set of experiment support services. Furthermore, as shown in Slice B, “opt-in” users may join the experiment running in a slice, and thus be associated with that slice.

3 GENI Control Framework Overview

3.1 Definition

The GENI control framework is defined in the GENI Control Framework Requirements document at http://geni.bbn.com:8080/docushare/dsweb/Services/Document-1234 .

It includes these entities:

- A "clearinghouse" consisting of principal, component and slice registries, plus related services.
- Services associated with each aggregate.
- Principals typically using tools, and acting as clients.

The GENI control framework defines:

- Interfaces between all entities.
- Planes for transporting messages between all entities.
- Message types, including basic protocols and required functions.
- Message flows necessary to realize key experiment scenarios.

3.2 Requirements

The GENI control framework requirements are presented in the GENI Control Framework Requirements document at http://geni.bbn.com:8080/docushare/dsweb/Services/Document-1234 .

3.3 Implementation Approach for Spiral 1 Prototypes

Five control framework implementations are being developed for Spiral 1 prototypes, based on the following systems and software packages:

• PlanetLab, a system that allows researchers to conduct experiments on hosts located at various sites; see http://svn.planet-lab.org/ and project [PlanetLab].

• ProtoGENI, which is based Emulab, a system that allows researchers to conduct experiments on hosts and other equipment located in various sites; see https://www.protogeni.net/trac/protogeni and project [ProtoGENI].

• ORCA resource allocation software; see http://nicl.cod.cs.duke.edu/orca/ and project [ORCA/BEN].

• ORBIT, a system that allows researchers to conduct experiments on a federated arrangement of wireless networks, utilizing peridically disconnected resources; see http://www.orbit-lab.org/wiki/WikiStart and project [ORBIT].

• DETER, a system that allows researchers to conduct experiments on a federated arrangement of hosts located in various sites; see http://seer.isi.deterlab.net/ and project [DETER].

Each control framework will be used for one cluster of projects, and typically provides the clearinghouse and a reference implementation of the aggregate manager for its cluster. Researchers in a given cluster will typically be able to conduct experiments only on the prototype aggregates and components within that cluster.

At the completion of Spiral 1, these control framework prototypes will be compared and evaluated.

For Spiral 2 prototyping during the following year, improvements and/or consolidations are expected. Useful features in one control framework may be adopted by another. A particular project may choose to move from one control framework to another. And, it is also possible that two (or more) control frameworks may merge, or possibly just federate.

Sections 4 though 11 present an overview of the ProtoGENI-based control framework implementation.

4 ProtoGENI Control Framework Structure

A block diagram of the ProtoGENI control framework structure is shown in Figure 4-1, which includes:

- Principal, Slice and Component Registries, which are the key entities in the Clearinghouse, and which include, respectively, Principal, Slice and Component Registry Services, plus Principal, Slice and Management Authority Services.
- Aggregates, each of which includes zero, one or many Components.
- Principals, such as Administrators, PIs, Operators, and Researchers.

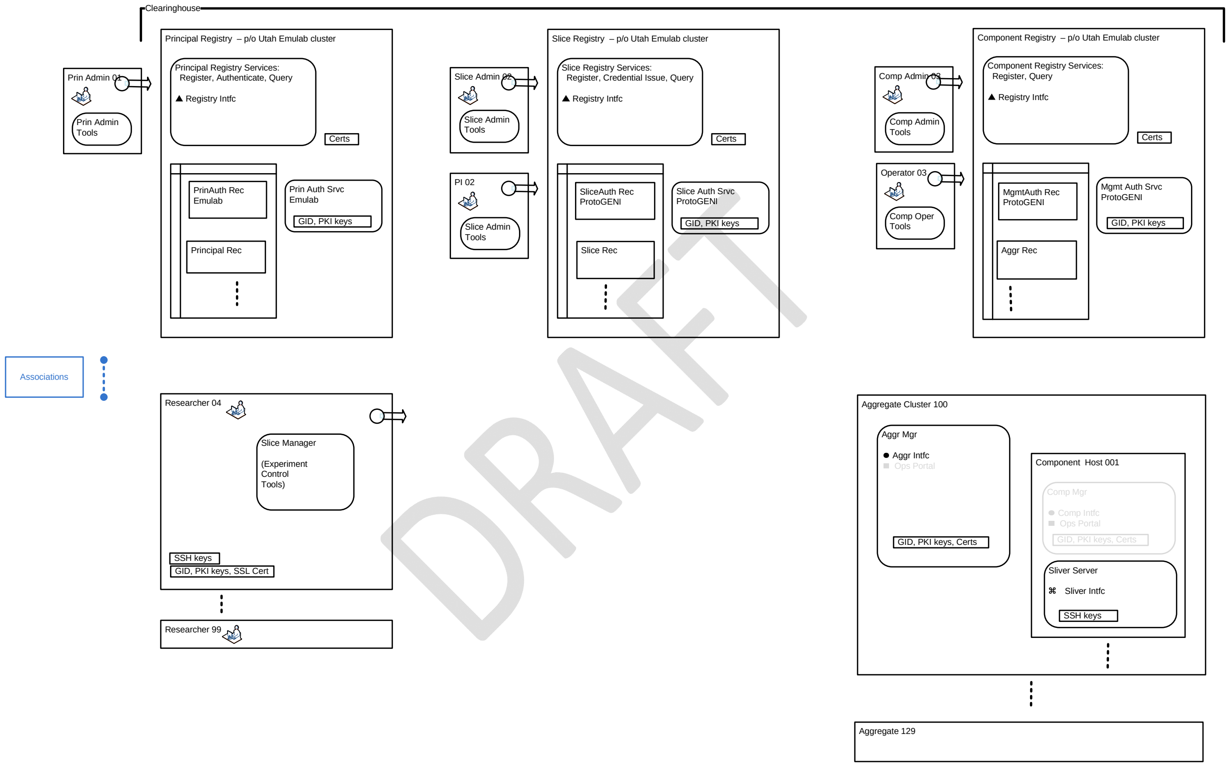


Figure 4-1. ProtoGENI Control Framework Structure.

The ProtoGENI control framework structure is based on the “Slice-based Facility Architecture” (SFA) summarized in http://groups.geni.net/geni/attachment/wiki/GeniControlBr/v1.10%20%20080808%20%20sfa.pdf , but it includes a number of engineering decisions that are consistent with a "prototype" implementation.

(Note that the SFA is based on earlier GENI efforts, including: "GMC Specifications," "GENI Distributed Services," and "GENI Engineering Guidelines.")

4.1 Registries

The registries hold all of the records plus associated services which are necessary for the operation of the ProtoGENI suite. Each registry exports a Registry Interface.

ProtoGENI implements a centralized Clearinghouse registry on Utah's Emulab cluster.

Users and Slices are registered in the Clearinghouse by the Slice Managers at each Emulab instance.

Question: Aggregates are registered in the Clearinghouse by?

Question: Are there multiple registries, or is there just one registry?

Per the SFA, each registered item has a Global Identifier (GID) that includes a Global NAME (GNAME) and a UUID. The UUID is a random number, generated following X.667 (RFC4122), that is guaranteed to be unique. The registry Resolve() and GetCredential() functions take either a GNAME or a UUID as the target of the lookup operation.

Issue: A problem with the above relates to the use of GNAMES. It is defined to correspond to a chain of authorities, but a slice is not an authority, nor is a node. One possibility is to treat the last N components of the GNAME as non-authoritative when verifying the chain of credentials (verification stops when a component does not correspond to GID in the chain). But this approach needs more thought.

4.2 Aggregates and Components

ProtoGENI, when running on an Emulab instance, is fundamentally an aggregate of resources, consisting of cluster nodes, switches and other resources within an Emulab installation.

An (aggregate) Component Interface is exported from an Emulab cluster, but individual resources such as nodes and switches, do not export a component interface; all operations on individual resources are via the aforementioned Component Interface at the cluster. See Figure 4.2

Private communication channels are then invoked to handle any operations required on individual resources.

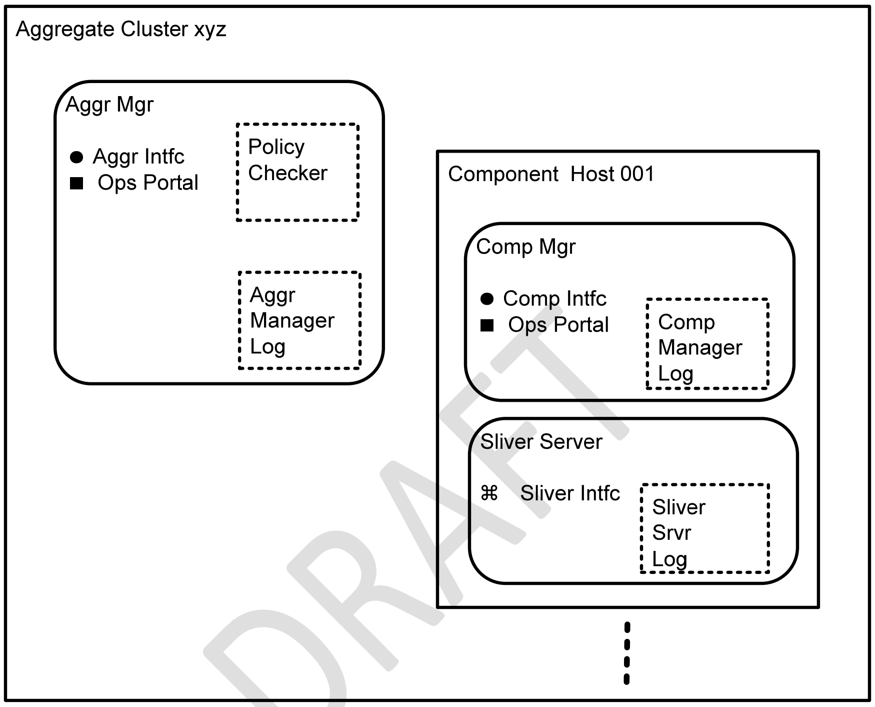


Figure 4-2. An Aggregate and Its Internal Structure.

4.3 Principals

A ProtoGENI principal (user) can be:

- A principal (user) acting from a server utilizing a browser.
- A principal (user) acting from a server utilizing a set of helper tools, such as a Researcher utilizing a Slice Manager (Experiment Control Tools).

Question: Is the Slice Manager located with the Resaercher as shown, or is it an entity associated with an Emulab instance?

A principal in ProtoGENI has privileges to allow it to play one (or more than one) of the following roles in the ProtoGENI suite:

- Principal administrators, who act for the ProtoGENI suite or a research organization, and are responsible for principal records and the authentication of principals.

- Aggregate administrators, who act for the ProtoGENI suite or an owning organization, and are responsible for aggregate records.
- Slice administrators, who act for the ProtoGENI suite or a research organization, and are responsible for slice records.
- PIs, who act for a research organization, and are responsible for slice records, the researchers assigned to a slice, and for managing slices, including all of their slivers.
- Operators, who act for the ProtoGENI suite or an owning organization, and are responsible for operations and management functions within an aggregate (or component).
- Researchers, who utilize the ProtoGENI suite for running experiments, deploying experimental services, etc.

Question: Is this list complete and correct?

4.4 Services

(Not yet defined in ProtoGENI.)

4.5 Slices

From a Researcher's perspective, a slice is an interconnected set of reserved resources, or slivers, on heterogeneous substrate aggregates (components). Researchers can remotely discover, reserve, configure, and program, debug, operate, manage, and teardown resources on these aggregates (components) to setup, utilize and then teardown the slivers necessary to complete an experiment. See Section 2.3.

From an Operator's perspective, slices are the primary abstraction for accounting and accountability—resources are acquired and consumed by slices, and external program behavior is traceable to a slice, respectively.

Question: Does a ProtoGENI slice map to an Emulab experiment?

4.6 Message Flows

The current approach to message flows in the ProtoGENI control framework is summarized by:

- Principals that periodically connect and communicate with Registries and Aggregates via defined interfaces and APIs.
- In particular, Researchers periodically connect and communicate with Registries and Aggregates so that they can acquire the resources necessary for them to setup and execute experiments. See Figure 4.3. In these transactions, Aggregates supply resources, and Researchers consume resources.
- Since the Aggregates are expected to be widely distributed, and connections are made over an IP network that may be the open Internet, the message flows must be secure, and the Principals must be properly authenticated.

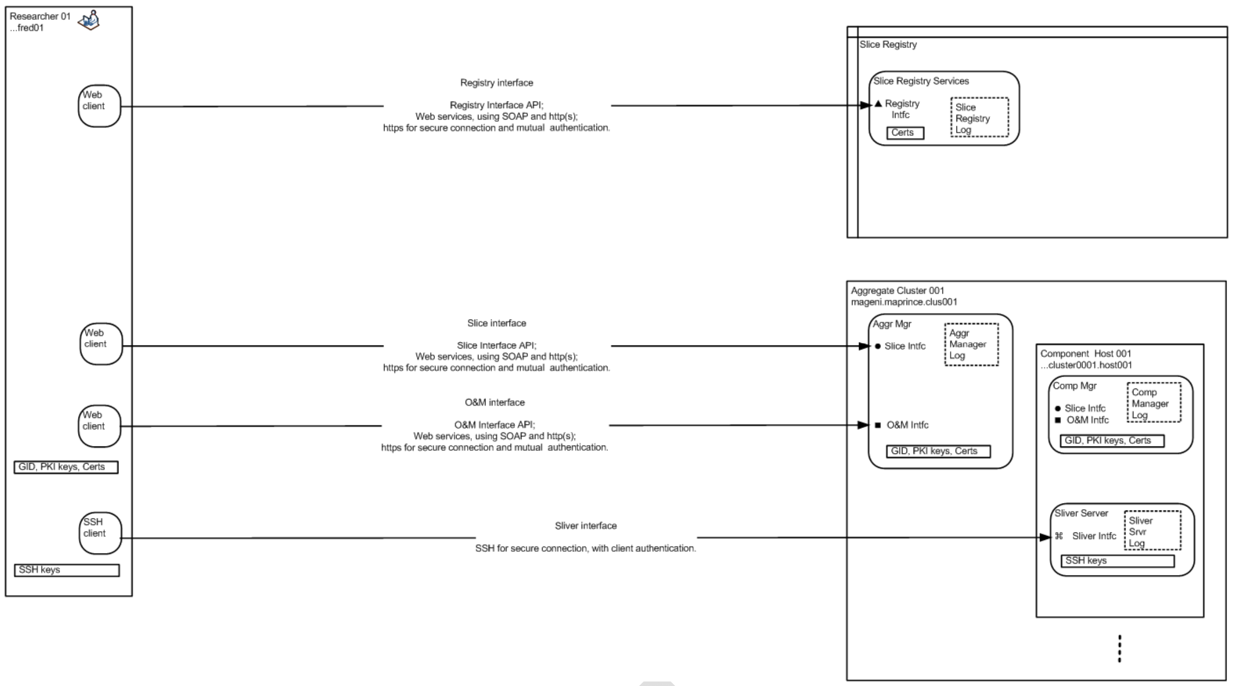
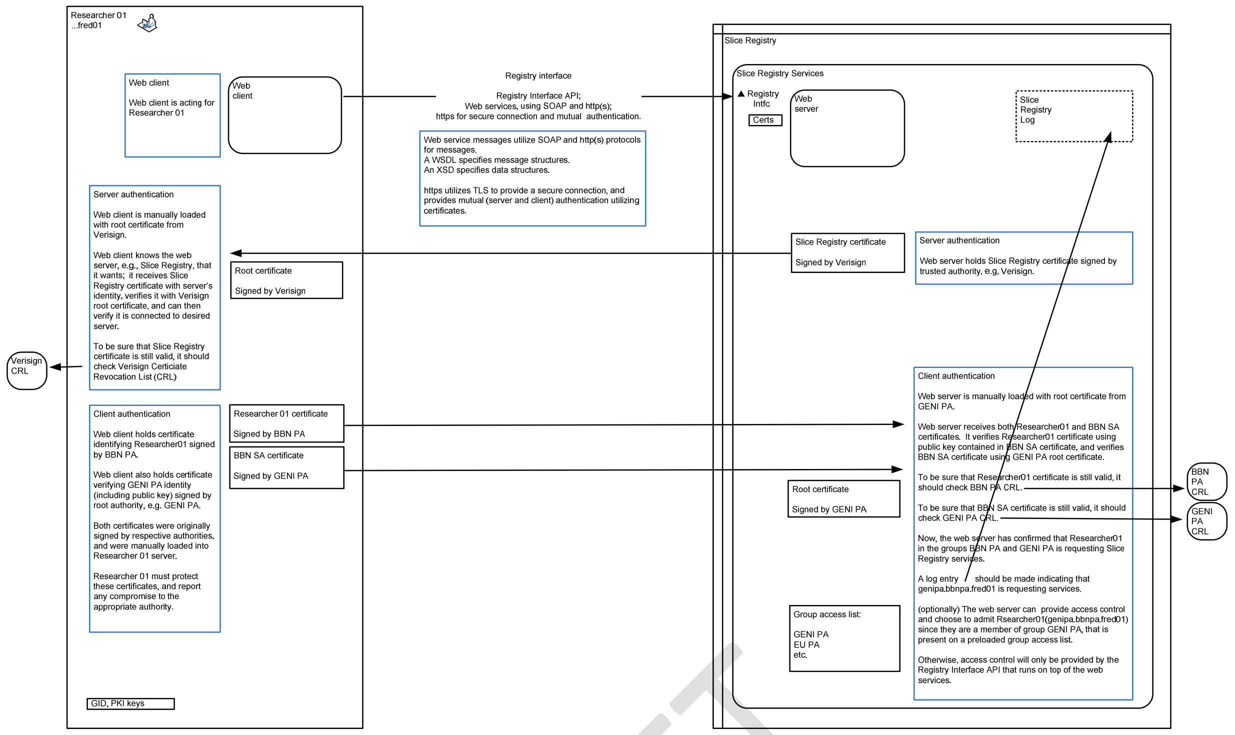


Figure 4.3 – Researcher to Registry and Aggregate Message Flows

4.7 Control Interfaces

In ProtoGENI, communications between a Principal's server and the Registry and Component Interfaces follows a web services model. The current decision is to utilize:

- A custom API for each of these interfaces which utilizes web service messages.
- The Principal is on the client side, and uses an xmlrpc client to talk to the servers, such as python.
- The Registry and Component Interfaces are on the server side.
- Web service messages that utilize XML RPC and http(s) protocols for messages.
- A WSDL that specifies the message structures.
- An XSD that specifies the data structures.
- http(s) utilizes TLS to provide a secure connection.
- http(s) provides mutual authentication.
- To permit mutual authentication, certificates in both servers can be verified by the other server using a root certificate. See Figure 4.4.



Web client

Web client is acting for Researcher 01

Server authentication

Web client is manually loaded with root certificate from Verisign.

Web client knows the web server, e.g., Slice Registry, that it wants; it receives Slice Registry certificate with server's identity, verifies it with Verisign root certificate, and can then verify it is connected to desired server.

To be sure that Slice Registry certificate is still valid, it should check Verisign Certiciate Revocation List (CRL)

Client authentication

Web client holds certificate identifying Researcher01 signed by BBN PA.

Web client also holds certificate verifying GENI PA identity (including public key) signed by root authority, e.g. GENI PA.

Both certificates were originally signed by respective authorities, and were manually loaded into Researcher 01 server.

Researcher 01 must protect these certificates, and report any compromise to the appropriate authority.

Registry Interface API;
Web services, using SOAP and http(s);
https for secure connection and mutual authentication.

Web service messages utilize SOAP and http(s) protocols for messages.
A WSDL specifies message structures.
An XSD specifies data structures.

https utilizes TLS to provide a secure connection, and provides mutual (server and client) authentication utilizing certificates.

Server authentication

Web server holds Slice Registry certificate signed by trusted authority, e.g, Verisign.

Client authentication

Web server is manually loaded with root certificate from GENI PA.

Web server receives both Researcher01 and BBN SA certificates. It verifies Researcher01 certificate using public key contained in BBN SA certificate, and verifies BBN SA certificate using GENI PA root certificate.

To be sure that Researcher01 certificate is still valid, it should check BBN PA CRL.

To be sure that BBN SA certificate is still valid, it should check GENI PA CRL.

Now, the web server has confirmed that Researcher01 in the groups BBN PA and GENI PA is requesting Slice Registry services.

A log entry should be made indicating that genipa.bbnpa.fred01 is requesting services.

(optionally) The web server can provide access control and choose to admit Rsearcher01(genipa.bbnpa.fred01) since they are a member of group GENI PA, that is present on a preloaded group access list.

Otherwise, access control will only be provided by the Registry Interface API that runs on top of the web services.

Figure 4.4 – Researcher to Registry Message Flow with Mutual Authentication

4.8 Ops and Mgmt Interfaces

(Note yet defined in ProtoGENI.)

4.9 Sliver Interfaces

Once a Sliver has been created on a component, a Sliver Interface is sometimes provided on the component to allow the Researcher to program, configure and operate a server within the underlying component, here designated the Sliver Server. The Sliver Server may be a physical server, or a virtual machine. The Sliver Server may represent a component (host) itself, or a part of the component, e.g., a controller acting as a protocol engine.

Typically, a Researcher on their server connects with the Sliver Server via the Sliver Interface using a Secure Shell (SSH) login. See Figure 4.5.

An SSH login provides for almost complete control of the Sliver Server within the component, and it is important that that server by well isolated from other slices to prevent security breaches.

Mutual authentication is required in an SSH login. ProtoGENI utilizes public and private key pairs, where the public key is loaded into the Sliver Server, and both public and private keys are held in the SSH client.

And, in ProtGENI, a new UpdateSliceAttributes() function has been added to allow for the specification of user keys, init scripts, and other as yet undefined items, to be associated with a slice on a component. The call can be made prior to any RedeemTicket() or InstantiateSlice() calls (before a sliver exists on the component) so that keys and the like can be pre-staged.

```
UpdateSliceAttributes(Credential, AttrNames[], AttrValues[]);
```

where AttrNames is currently one of "keys" or "initscript". When specifying keys for a slice, the value is another array that allows multiple keys to be associated with multiple researchers, thus binding a set of researchers (and their keys) to the slice, on that component.

4.10 Authentication

The current approach to authentication of registries, aggregates, principals and services in the ProtoGENI suite is summarized by:

- One Public Key Infrastructure (PKI) that covers all ProtoGENI registries, aggregates and principals.
- This PKI provides all necessary certificates, and allows verification to be done using a limited number of root certificates.

ProtoGENI employs a much simplified trust model that is more consistent with a prototype implementation. Since the number of trusted "roots" will be small at first, we exchange root SSL certificates out of band, and populate a certificate directory that can be used for verifying client certificates when they are presented. In short, SSL's built in client verification is used to step back through the chain of issuers, which in the current prototype is usually of length one. As a result, ProtoGENI does not currently use or implement GID chains.

Note that we will support GID chains in the future, so that we conform to the SFA spec and so that we can inter-operate more easily with other GENI Spiral 1 implementations. The current approach was taken so that we can develop the other parts of ProtoGENI, while we wait for the specification of GIDs to be finalized.

4.11 Authorization

The current approach to authorization in the ProtoGENI suite is summarized by:

- The exchange of credentials, including tickets.
- To mediate resource authorization by Aggregates, plus assignment and control.
- Credentials that are signed (certified) by the appropriate authorities (slice) and objects (aggregates and slivers) to give them some intrinsic value.
- To certify a credential (ticket), an authority or object signs the token using its own private key, which is then followed by signatures from its responsible authorities, up to the root authority. In the current implementation, there is always only one signature.
- The Public Key Infrastructure (PKI) that is used to authenticate principals, etc., provides all of the keys and other structure to sign and verify credentials.
- The Aggregate that receives this token can then verify it using a limited set of root certificates.
- This follows the approach that was developed in "trust management system" and in early GENI designs.
- This follows the SFA; see http://groups.geni.net/geni/attachment/wiki/GeniControlBr/v1.10%20%20080808%20%20sfa.pdf .

4.12 Credentials

In ProtoGENI, a Researcher requires a Credential to access an Aggregate via the Component Interface to have resources authorized and assigned to create slivers, and to control slivers.

A credential is always received by an Aggregate Manager, and then the requested function is either authorized and completed, or not.

Slivers in ProtoGeni are first class objects; every sliver is controlled via a credential that is created when the sliver is instantiated. RedeemTicket() and InstantiateSlice() both return a credential to the caller. While a credential can be requested via the MA interface, we feel that to be an unnecessary additional step.

A ProtoGENI credential is currently defined as:

```
## A credential granting privileges or a ticket.
credentials = element credential {
   ## The ID for signature referencing.
   attribute xml:id {xs:ID},
   ## The type of this credential.
   element type { "privilege" | "ticket" },
   ## A serial number.
   element serial { xsd:string },
   ## GID of the owner of this credential.
   element owner_gid { xsd:string },
   ## GID of the target of this credential.
   element target_gid { xsd:string },
   ## UUID of this credential
   element uuid { xsd:string },
   ## Expires on
   element expires { xsd:dateTime },
   ## Privileges or a ticket
   (PrivilegesSpec | TicketSpec ),
   ## Optional Extensions
   element extensions { anyelementbody }*,
   ## Parent that delegated to us
   element parent { credentials }?
}

signatures = element signatures {
   element sig:Signature { ... }+
}
SignedCredential = element signed-credential {
   credentials,
   signatures?
}
```

A credential is signed using the XMLSIG specification, located at http://www.w3.org/TR/xmldsig-core/ . To facilitate delegation, a credential can have an optional chain of parent credentials, each one signed. An original (un-delegated) credential will not have a parent. A credential is delegated by creating a new credential, setting the parent to the original credential, and then signing the entire blob. The credential can then be verified by reversing the operation, verifying the signature at each level. An existing tool called xmlsec1 (http://www.aleksey.com/xmlsec/) is used for the verification. A secondary step is then used to ensure that the rules of delegation were not violated (ie: an inner credential does not include a privilege that an outer credential did not allow to be delegated).

Each credential has its own UUID to allow for easier bookkeeping, and for simple revocation. A UUID is also useful when supporting unmediated splitting of tickets; the CM needs to be able trace back the split ticket to the original ticket.

We currently feel (although not very strongly) that delegation should be at the individual privilege level, not at the credential level, except when the credential is really a ticket.

Credentials are extensible, using the "extensions" field above. No format is defined as of yet, but the intent is to be able pass along data in the credential that has been signed along with the rest of the credential data. When delegating a credential, the extension data must remain unchanged.

Question: What is in a PrivilegesSpec? Does this include a Resource Spec (RSpec)?

Question: What is in a TicketSpec? Does this include a Resource Spec (RSpec)? Does this include start time and duration for a reservation? If so, how?

4.13 Tickets

In ProtoGENI, a Ticket is a specific kind of credential. See Section 4.12.

4.14 Resource Specification (RSpec)

A *resource specification* (RSpec) describes a component in terms of the resources it possesses and constraints and dependencies on the allocation of those resources. The exact form of an RSpec is still being defined elsewhere.

The ProtoGENI rspec is described at https://www.protogeni.net/trac/protogeni/wiki/RSpec . It is based on the ptop/vtop format designed for Emulab's "assign" resource mapper. This format is fairly simple, and has the advantage of being in production use for 8 years now in Emulab. This format does have some parts that are specific to Emulab and assign: these will simply be ignored to begin with, and we will remove them from the RSpec or implement them as extensions as we move to a full ProtoGeni rspec.

Per the ProtoGENI implementation, an RSpec can include a request for a link between two nodes, or a LAN between a set of nodes. ProtoGENI supports independent control of these links and LANs, and even the individual interfaces on nodes attached to the links. As a result, ProtoGENI treats these sub resources the same as slivers, creating credentials that the caller can use to control them. In addition to the defined sliver operations in the API, ProtoGENI will export additional APIs that are specific to these other resources.

Since Emulab clusters exporting the ProtoGNEI APIs, are exporting a Component Interface, the resulting sliver (credential) might encompass a collection of resources (nodes and links). To operate on an individual piece of that sliver, such a link, it must be possible to extract a credential that refers to that link. ProtoGENI provides an extraction operation that takes a credential and a GNAME, and returns a new credential for that specific resource. For example, to extract a credential for a link, you would use the extraction operation on the credential for geni.emulab.slice0.link0, to get a credential for an interface, geni.emulab.slice0.node0.eth0.

5 Principals in the ProtoGENI Control Framework

5.1 Identification

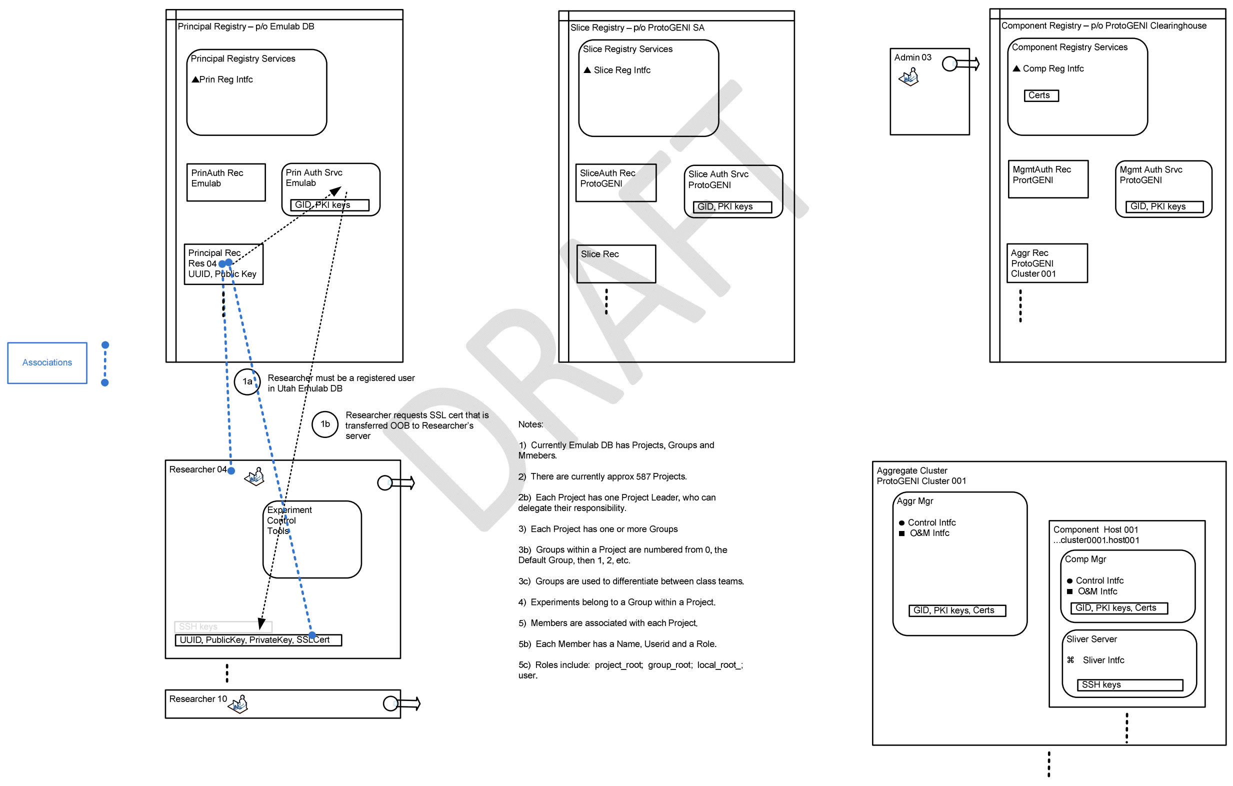
Each principal (user) in ProtoGENI, such as a Researcher, has a unique Global Identifier (GID) that does not change, and that includes a Global NAME (GNAME) and a UUID.

Question: Who creates the UUID, and names the principal?

5.2 Registration

Each principal (user) is registered within the ProtoGENI suite, so that they can be identified and authenticated. They are given a set of privileges to precisely define what they can and cannot do within the ProtoGENI suite, and in a particular situation.

See Figure 5-1 for the steps toward registering a Researcher.


Principal Registry – p/o Emulab DB
Principal Registry Services
▲Prin Reg Intfc
PrinAuth Rec Emulab
Prin Auth Srvc Emulab
GID, PKI keys
Principal Rec Res 04 UUID, Public Key
Associations
1a
Researcher must be a registered user in Utah Emulab DB
1b
Researcher requests SSL cert that is transferred OOB to Researcher's server
Researcher 04
Experiment Control Tools
SSH keys
UUID, PublicKey, PrivateKey, SSLCert
Researcher 10
Slice Registry – p/o ProtoGENI SA
Slice Registry Services
▲ Slice Reg Intfc
SliceAuth Rec ProtoGENI
Slice Auth Srvc ProtoGENI
GID, PKI keys
Slice Rec
Admin 03
Component Registry – p/o ProtoGENI Clearinghouse
Component Registry Services
▲ Comp Reg Intfc
Certs
MgmtAuth Rec PrortGENI
Mgmt Auth Srvc ProtoGENI
GID, PKI keys
Aggr Rec ProtoGENI Cluster 001
Aggregate Cluster ProtoGENI Cluster 001
Aggr Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Component Host 001 ...cluster0001.host001
Comp Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Sliver Server
⌘ Sliver Intfc
SSH keys

Notes:

1) Currently Emulab DB has Projects, Groups and Mmebers.

2) There are currently approx 587 Projects.

2b) Each Project has one Project Leader, who can delegate their responsibility.

3) Each Project has one or more Groups

3b) Groups within a Project are numbered from 0, the Default Group, then 1, 2, etc.

3c) Groups are used to differentiate between class teams.

4) Experiments belong to a Group within a Project.

5) Members are associated with each Project.

5b) Each Member has a Name, Userid and a Role.

5c) Roles include: project_root; group_root; local_root_; user.

Figure 5.1 – Researcher Registration

Step 1a: Currently, the Researcher must first register as a user in the Utah Emulab DB.

Note: The Emulab DB has Projects, Groups and Members.

There are currently approx 587 Projects. Each Project has one Project Leader, who can delegate their responsibility.

Each Project has one or more Groups. Groups within a Project are numbered starting from 0 for the Default Group, then 1, 2, etc. In some cases, a Project corresponds to a class, and then Groups are used to differentiate between class teams.

Experiments belong to a Group within a Project.

Members are associated with each Project. Each Member has a Name, UserID and a Role. Roles include: project_root; group_root; local_root_; user.

Step 1b: Researcher requests an SSL cert that is transferred out-of-band (OOB) to the Researcher's server.

Question: What about other ProtoGENI principals?

Question: Must a ProtoGENI Researcher have a particular role in the Emulab DB?

5.3 Authentication

Each ProtoGENI principal (e.g., Researcher) can be authenticated using their SSL certificate.

6 Aggregates and Components in ProtoGENI Control Framework

6.1 Identification

Each aggregate in ProtoGENI has a unique Global Identifier (GID) that does not change, and that includes a Global NAME (GNAME) and a UUID.

Question: Who creates the UUID, and names the aggregate?

6.2 Registration

Each Aggregate must be registered within the ProtoGENI suite, including pointers (URLs) to the Aggregate.

Question: Who does this? How is it done?

6.3 Resource Allocation

By registering an aggregate in the ProtoGENI suite, the administrator (owner) of the aggregate indicates that they are willing to allocate resources to experiments in the ProtoGENI suite.

However, the information currently contained in the Aggregate Record does not provide any information on the resources that can be utilized in ProtoGENI.

7 Slices in ProtoGENI Control Framework

7.1 Identification

Each Slice in ProtoGENI has a unique Global Identifier (GID) that does not change, and that includes a Global NAME (GNAME) and a UUID.

7.2 Registration

In ProtoGENI, a Slice is registered when a Researcher presents an SSL certificate and requests a Slice from the Slice Authority (SA) service in the Slice Registry. See Figure 7-1.

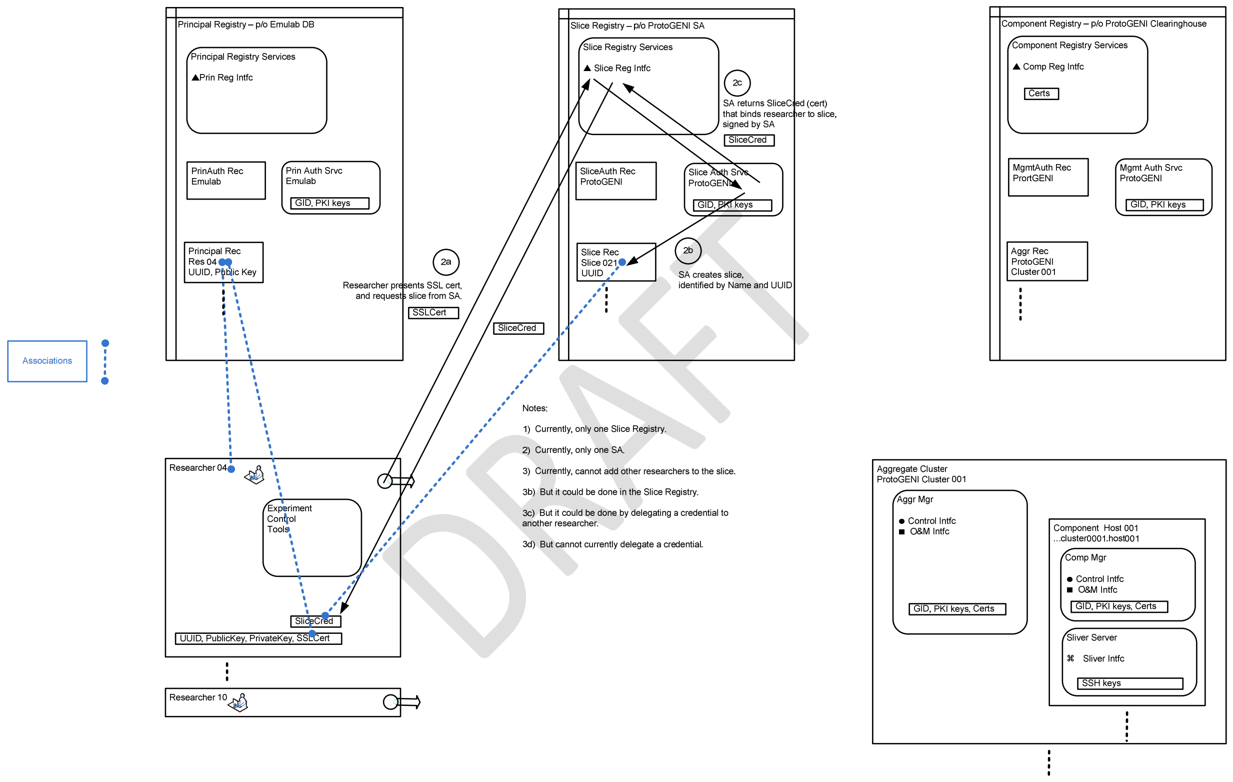

Principal Registry – p/o Emulab DB
Principal Registry Services
▲Prin Reg Intfc
PrinAuth Rec Emulab
Prin Auth Srvc Emulab
GID, PKI keys
Principal Rec Res 04 UUID, Public Key
2a
Researcher presents SSL cert, and requests slice from SA.
SSLCert
SliceCred
Slice Registry – p/o ProtoGENI SA
Slice Registry Services
▲ Slice Reg Intfc
2c
SA returns SliceCred (cert) that binds researcher to slice, signed by SA
SliceCred
SliceAuth Rec ProtoGENI
Slice Auth Srvc ProtoGENI
GID, PKI keys
Slice Rec Slice 021 UUID
2b
SA creates slice, identified by Name and UUID
Component Registry – p/o ProtoGENI Clearinghouse
Component Registry Services
▲ Comp Reg Intfc
Certs
MgmtAuth Rec ProtGENI
Mgmt Auth Srvc ProtoGENI
GID, PKI keys
Aggr Rec ProtoGENI Cluster 001
Associations
Notes:
1) Currently, only one Slice Registry.
2) Currently, only one SA.
3) Currently, cannot add other researchers to the slice.
3b) But it could be done in the Slice Registry.
3c) But it could be done by delegating a credential to another researcher.
3d) But cannot currently delegate a credential.
Researcher 04
Experiment Control Tools
SliceCred
UUID, PublicKey, PrivateKey, SSLCert
Researcher 10
Aggregate Cluster ProtoGENI Cluster 001
Aggr Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Component Host 001 ...cluster0001.host001
Comp Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Sliver Server
⌘ Sliver Intfc
SSH keys
DRAFT

Figure 7-1. Slice Registration and Slice Credential Issue.

Step 2a: Researcher presents SSL cert, and requests slice from SA.

Question: How does the SA, based on SSL cert, decide that it should issue a slice? Does it always issue a requested slice?

Step 2b: SA creates slice, identified by GNAME and UUID.

Question: Who creates the UUID and names the Slice?

In ProtoGENI, there is currently only one Slice Registry. And, there is currently only one SA.

7.3 Credential Issue

Step 2c: SA returns SliceCred (cert) that binds Researcher to Slice, and is signed by the SA.

Currently, you cannot add other researchers to the slice. However, this could be done in the Slice Registry with some additional features. And, it could be done by delegating a credential to another researcher, although you cannot currently delegate a credential.

Question: Is there only one Slice Credential that the Researcher holds and uses repeatedly, or must they repeatedly get credentials, that then have only “one use”?

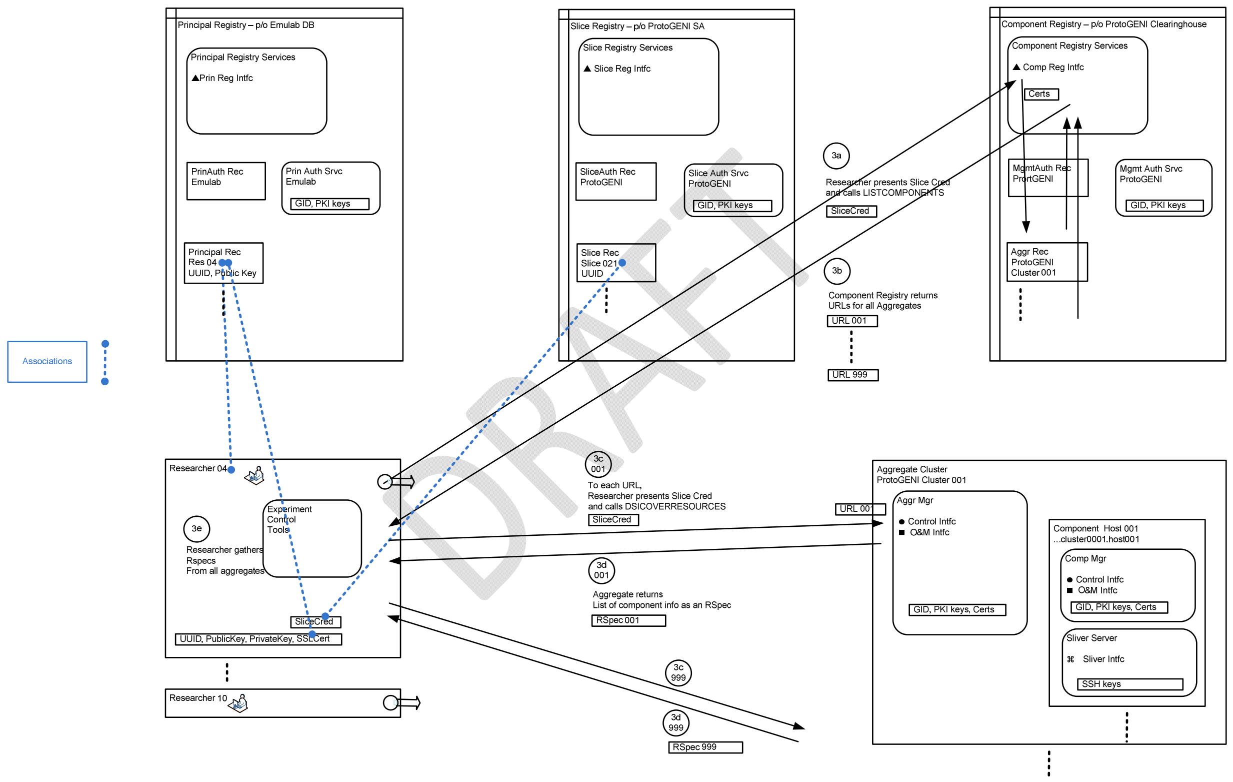
8 Experiment Setup in ProtoGENI Control Framework

The ProtoGENI control framework provides all of the basic functions necessary for a GENI researcher to setup an experiment.

8.1 Resource and Topology Discovery

The ProtoGENI control framework allows a Researcher, using the Component Registry, to discover all of the resources available to them from the Aggregates associated with the ProtoGENI suite. See Figure 8-1.

Question: How can they discover their interconnection topology?



Principal Registry – p/o Emulab DB
Principal Registry Services
▲Prin Reg Intfc
PrinAuth Rec Emulab
Prin Auth Srvc Emulab
GID, PKI keys
Principal Rec Res 04 UUID, Public Key
Slice Registry – p/o ProtoGENI SA
Slice Registry Services
▲ Slice Reg Intfc
SliceAuth Rec ProtoGENI
Slice Auth Srvc ProtoGENI
GID, PKI keys
Slice Rec Slice 021 UUID
Component Registry – p/o ProtoGENI Clearinghouse
Component Registry Services
▲ Comp Reg Intfc
Certs
MgmtAuth Rec ProtGENI
Mgmt Auth Srvc ProtoGENI
GID, PKI keys
Aggr Rec ProtoGENI Cluster 001
Associations
3a
Researcher presents Slice Cred and calls LISTCOMPONENTS
SliceCred
3b
Component Registry returns URLs for all Aggregates
URL 001
URL 999
Researcher 04
Experiment Control Tools
3e
Researcher gathers Rspecs From all aggregates
SliceCred
UUID, PublicKey, PrivateKey, SSLCert
Researcher 10
3c 001
To each URL, Researcher presents Slice Cred and calls DSICOVERRESOURCES
SliceCred
3d 001
Aggregate returns List of component info as an RSpec
RSpec 001
3c 999
3d 999
RSpec 999
URL 001
Aggregate Cluster ProtoGENI Cluster 001
Aggr Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Component Host 001 ...cluster0001.host001
Comp Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Sliver Server
⌘ Sliver Intfc
SSH keys
DRAFT

Figure 8-1. Resource Discovery.

Step 3a: Researcher presents Slice Cred to Component Registry and calls LISTCOMPONENTS.

Step 3b: Component Registry returns URLs for all registered Aggregates.

Step 3c: To each Aggregate at its URL, Researcher presents Slice Cred and calls DISCOVERRESOURCES.

Step 3d: Each Aggregate returns list of component info as an RSpec.

Step 3e: Researcher gathers RSpecs from all Aggregates.

8.2 Resource Sharing

In ProtoGENI, each aggregate provides for resource sharing among multiple researchers, referencing multiple slices, and assigns each researcher their own sliver or slivers.

8.3 Resource Authorization and Policy Implementation

The ProtoGENI control framework allows an Aggregate to authorize the assignment of resources to a Researcher referencing a particular Slice, following local policies. See Figure 8-2.

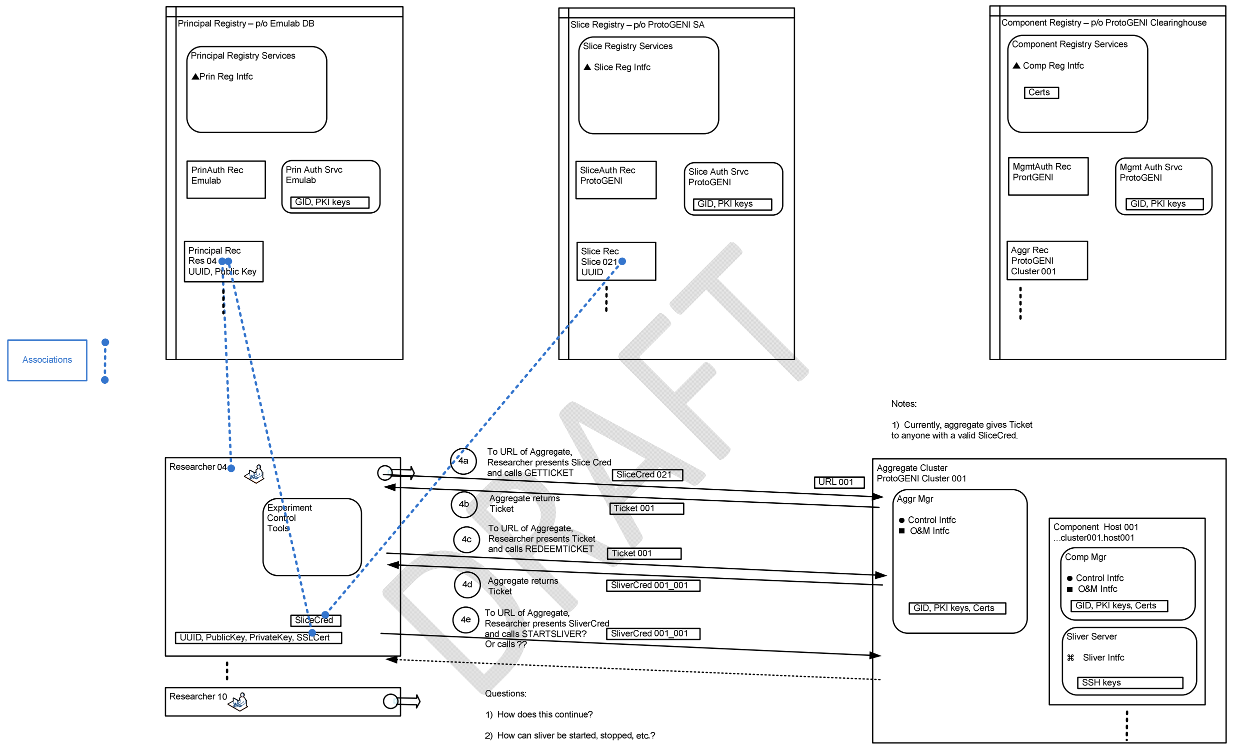

Principal Registry – p/o Emulab DB
Principal Registry Services
▲Prin Reg Intfc
PrinAuth Rec Emulab
Prin Auth Srvc Emulab
GID, PKI keys
Principal Rec Res 04 UUID, Public Key
Slice Registry – p/o ProtoGENI SA
Slice Registry Services
▲ Slice Reg Intfc
SliceAuth Rec ProtoGENI
Slice Auth Srvc ProtoGENI
GID, PKI keys
Slice Rec Slice 021 UUID
Component Registry – p/o ProtoGENI Clearinghouse
Component Registry Services
▲ Comp Reg Intfc
Certs
MgmtAuth Rec PrortGENI
Mgmt Auth Srvc ProtoGENI
GID, PKI keys
Aggr Rec ProtoGENI Cluster 001
Associations
DRAFT
Notes:
1) Currently, aggregate gives Ticket to anyone with a valid SliceCred.
Researcher 04
Experiment Control Tools
SliceCred
UUID, PublicKey, PrivateKey, SSLCert
Researcher 10
4a
To URL of Aggregate, Researcher presents Slice Cred and calls GETTICKET
SliceCred 021
URL 001
4b
Aggregate returns Ticket
Ticket 001
4c
To URL of Aggregate, Researcher presents Ticket and calls REDEEMTICKET
Ticket 001
4d
Aggregate returns Ticket
SliverCred 001_001
4e
To URL of Aggregate, Researcher presents SliverCred and calls STARTSLIVER? Or calls ??
SliverCred 001_001
Questions:
1) How does this continue?
2) How can sliver be started, stopped, etc.?
Aggregate Cluster ProtoGENI Cluster 001
Aggr Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Component Host 001 ...cluster001.host001
Comp Mgr
● Control Intfc
■ O&M Intfc
GID, PKI keys, Certs
Sliver Server
⌘ Sliver Intfc
SSH keys

Figure 8-2. Resource Authorization plus Assignment, and Sliver Control

Step 4a: To URL of Aggregate, Researcher presents Slice Credential and calls GETTICKET.

Question: Does this call include an RSpec?

Step 4b: Aggregate decides that it will authorize resources to this Researcher and the referenced Slice using its local policy, and then returns a Ticket to the Researcher.

Currently, an Aggregate gives a Ticket to anyone with a valid Slice Credential, i.e., there is an "always yes" local policy.

Question: What is included in Ticket? Does the Ticket specify starting time and duration? If so, how are they determined?

Question: If the Ticket specifies a starting time and duration, are the resources assigned on a "best effort" basis, or is there a firm commitment?

8.4 Resource Assignment

The ProtoGENI control framework allows an Aggregate to assign resources to a Researcher presenting a valid Ticket. See Figure 8-2.

Step 4c: To URL of Aggregate, Researcher presents Ticket and calls REDEEMTICKET.

Slivers in ProtoGENI are first class objects; every sliver is controlled via a credential that is created when the sliver is instantiated. RedeemTicket() and InstantiateSlice() both return a credential to the caller.

Step 4d: Aggregate assigns resources in a Sliver to the Researcher and their Slice, and then returns a Sliver Credential, that the Researcher can use to control the Sliver.

Question: How are starting time and duration determined?

Question: Are the resources assigned on a "best effort" basis, or is there a firm commitment?

8.5 Component Programming

The PortoGENI control framework allows a researcher to login to an assigned sliver (component), load code, and then boot it, etc. This is done via the Sliver Interface; see Section 4.13.

8.6 Disconnected Operation of Components

(Note yet defined in ProtoGENI.)

8.7 Resource to Resource Connections

When a researcher has been assigned resources from two (or more) aggregates that must be connected together, the ProtoGENI control framework provides a way for the researcher to learn about the connection points, request the connections, following the necessary sequence, and receive a verification that the connection has been completed.

This follows the Emulab "virtual network" approach described in http://www.cs.utah.edu/flux/papers/virt-usenix08-base.html .

For example, an RSpec can include a request for a link between two nodes, or a LAN between a set of nodes. ProtoGENI supports independent control of these links and LANs, and even the individual interfaces on nodes attached to the links.

To accomplish this, ProtoGENI treats these sub resources the same as slivers, creating credentials that the caller can use to control them. In addition to the defined sliver operations in the API, ProtoGENI will export additional APIs that are specific to these other resources.

Since Emulab clusters exporting the ProtoGENI APIs, are exporting a Component Interface, the resulting sliver (credential) might encompass a collection of resources (nodes and links). To operate on an individual piece of that sliver, such a link, it must be possible to extract a credential that refers to that link. ProtoGENI provides an extraction operation that takes a credential and a GNAME, and returns a new credential for that specific resource. For example, to extract a credential for a link, you would use the extraction operation on the credential for geni.emulab.slice0.link0, to get a credential for an interface, geni.emulab.slice0.node0.eth0.

8.8 Setup Verification

(Note yet defined in ProtoGENI.)

9 Experiment Execution in ProtoGENI Control Framework

9.1 Experiment Control

When a Researcher, associated with a designated slice, has been assigned resources on aggregates for an experiment, the ProtoGENI control framework provides a way for the researcher to control the slivers in the aggregates using commands appropriate to the nature of the sliver. For example, start, stop and reboot for a process running on a host. Or, connect, disconnect and loopback for a path in a network. See Figure 8-2.

Step 4e: To URL of Aggregate, Researcher presents SliverCred and calls STARTSLIVER, or other appropriate command.

9.2 Experiment Data Collection and Management

GENI will provide for experiment data collection and measurement, both locally within aggregates (components) and globally in designated measurement services. It is expected that large data files will be gathered by these services, and that they will need to be transferred to a software repository and/or an experiment analysis service after an experiment.

To accomplish this, the control framework should provide the mechanism(s) to allow a researcher to transfer large software records between components, software repositories, etc. For example, the control framework could provide a file transfer service based on ftp.

(Not yet defined in ProtoGENI).

9.3 Forensic and Usage Data Collection and Management

Forensic and usage data from a GENI suite has many uses, including:

- Finding anomalies that indicate errors, faults, malicious activity, etc.
- Allowing help desk functions to be provided to researchers.
- Permitting proper administration and management of suite resources.
- Permitting financial accounting where necessary.

The control framework should provide a structure for collecting and managing forensic and usage data, including formats and log structures.

(Not yet defined in ProtoGENI.)

9.4 Experiment Status Monitoring

Experiment status can be monitored in a GENI suite by:

- Defining trigger events associated with the use of resources in a sliver, an aggregate or a component.
- Defining watchdog processes, to periodically verify functions in a sliver, an aggregate or a component.
- Sending notifications to a researcher, an administrator, an operator, or anyone who has requested receipt.

The control framework should provide a structure for experiment status monitoring, and sending notifications.

(Not yet defined in ProtoGENI.)

Question: What about the Emulab per-node watchdog process, with status message to Emulab central? See "monitor health" section in http://www.cs.utah.edu/flux/papers/portal-worlds04-base.html .

9.5 Experiment Status Commands

The control framework must provide a structure for commanding changes in the status of resources used by an experiment. It must be possible for changes to be commanded by a researcher or by an aggregate or component administrator or operator. For example, it must be possible to command "shutdown all slivers in this aggregate associated with slice x".

(Not yet defined in ProtoGENI.)

10 Federation in ProtoGENI Control Framework

10.1 Federated Aggregates and Components

A ProtoGENI control framework provides for the inclusion of a variety of federated aggregates (and their included components) to provide a wide range of resources to the Researchers.

Starting with a ProtoGENI cluster node and the Utah Emulab cluster node, it will add federated Emulab nodes at Carnegie Melon and Kentucky and thus realize a unified ProtoGENI suite.

Furthermore, the Utah Emulab cluster node already includes federated access to two PlanetLab Central (PLC) installations at Princeton and at Utah. This federation was done by implementing an Emulab to PlanetLab portal; see http://www.cs.utah.edu/flux/papers/portal-worlds04-base.html for more details.

10.2 Federated Suites

The control framework should provide for the federation of a GENI suite with other suites, where each suite has its own complete set of entities, but is independently owned and operated.

(Not yet defined in ProtoGENI.)

11 ProtoGENI Cluster C Spiral 1 Implementation

11.1 Start of Spiral 1

At the beginning of Spiral 1, the ProtoGENI Cluster C implementation includes a ProtoGENI cluster node and the Utah Emulab cluster node. See Figure 11-1.

Note that the Utah Emulab cluster node already includes federated access to two PlanetLab Central (PLC) installations at Princeton and at Utah. This federation was done by implementing an Emulab to PlanetLab portal; see http://www.cs.utah.edu/flux/papers/portal-worlds04-base.html for more details.

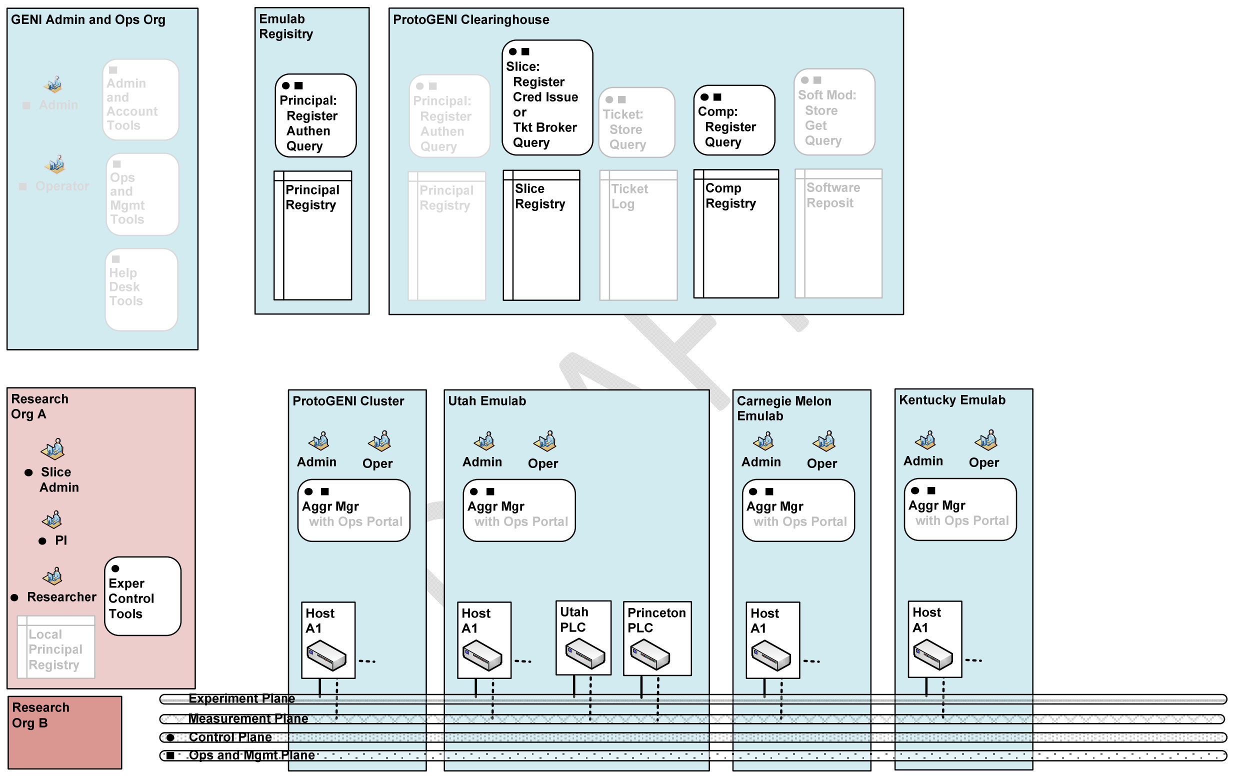


Figure 11-1. Start of ProtoGENI Cluster C Spiral 1 Implementation.

Next, it will add federated access to:

A programmable edge cluster, HomeNet?, and a wireless emulator at Carnegie Mellon University

Netlab at the University of Kentucky

Netlab at Georgia Tech

Schooner, part of WAIL at the University of Wisconsin-Madison

11.2 Completion of Spiral 1

Eventually, the following Cluster C projects will be integrated into the ProtoGENI implementation; see Figure 11-2.

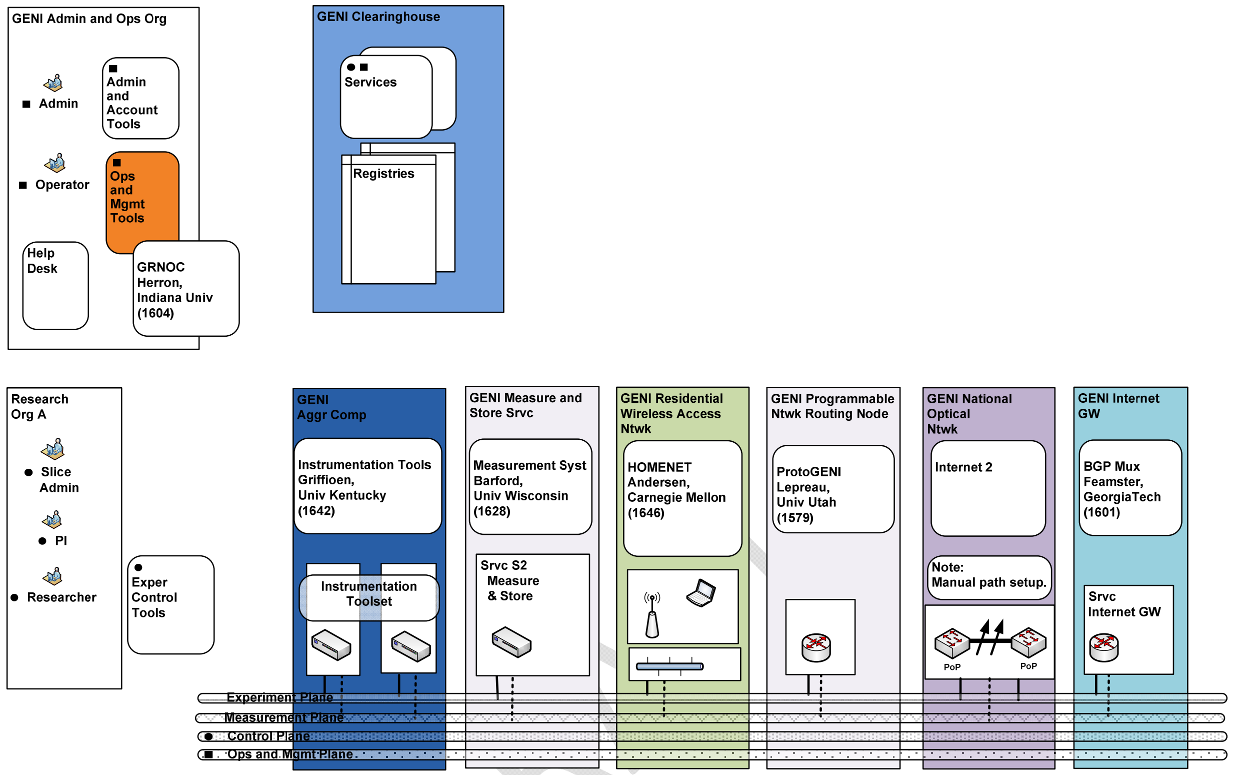


Figure 11-2. ProtoGENI Cluster C Implementation.

University of Kentucky: *Instrumentation Tools*

Tools for collecting and displaying measurements of hosts, such as traffic volumes and host load characteristics. Integration Plans

University of Wisconsin-Madison: *Measurement System*

Prototype of the GIMS measurement framework. A system for the capture and storage of packets, using capture devices separate from GENI components (eg. by employing optical splitters). An important part of this will be providing controlled access to measurements collected, possibly with anonymization. Integration Plans

Georgia Tech: *Virtual Tunnels*

Georgia Tech is a participant in the VINI project, which has as one of its goals improved network virtualization, allowing the creation of tunnels between sliced hosts, and allowing slivers more control over routing, etc. between their interfaces. Integration Plans

Georgia Tech: *BGPMux*

The BGP multiplexer can be used to share BGP feeds among researchers, for the purpose of advertising prefixes (a possible opt-in end user strategy) or getting vantage points into the global BGP topology. Integration Plans

Carnegie Mellon University: *HomeNet?*

HomeNet? will be a deployment of components throughout Pittsburgh on residential broadband (Cable and DSL), and will be equipped with 802.11 interfaces to create an urban wireless mesh. Integration Plans

Carnegie Mellon University: *WirelessEmulator?*

CMU's wireless emulator allows for the creation of controlled, repeatable wireless channel condition, under which real wireless hardware and protocols can be tested. Integration Plans

University of Massachusetts-Lowell: *Programmable Edge Node*

This project will produce a component that is sliced in a lightweight manner using the OpenVZ mechanism in the Linux kernel, and provides a highly-scalable set of virtual interfaces using a network processor. Integration Plans