

GEC 10 Report

March 15-17th, 2011

Darwin Witt

1 Status of Control Frameworks

Several CF groups reported their current status, recent developments, and efforts that will be made going forward with the construction of the GENI global framework.

1.1 ProtoGENI

ProtoGENI highlighted several key efforts that they have been working on since GEC 9. They have begun major strides to developing an understandable concept of stitching to generate a fully federated testbed supported by the major control frameworks. In close relation to this work, they have also begun establishing cross-aggregate links with PlanetLab and hope to extend to ORCA in the coming months. They cited a major hurdle in the efforts to provide cross-aggregate resources, the inability to hide details from the client or make the process automatic without needing the client to specify cross-aggregate connections. A significant leap forward is the addition of an aggregate-to-aggregate stitching call in the ProtoGENI CM API, which has yet to be advertised and made public knowledge. They plan to disseminate a new version of their APIs shortly.

Negotiations are being made to run backbone processes through Internet2 ION, establishing a native layer 2 connection to supplement the preexisting National Lambda Rail services. In relation to Internet2 deployment, new equipment has been established in LA, Houston, and Atlanta, adding more 1Gbps connections cross-continent.

With regards to federation attempts, there is now an established PlanetLab federation and work has begun on tools that work with both APIs. They have exchanged root keys with GpENI as apart of new negotiations to establish a federation with their testbed.

ProtoGENI has begun to focus on cleaning up bugs that might have escaped them earlier on during the first two spirals of development and generating documentation to fully describe the behavior and capabilities of their control framework.

ProtoGENI addressed the successful mitigation of a GENI AM API that works effectively and addressed the need to start discussing a GENI SA API. Ted Faber's document on SA API suggestions was cited as a starting point for such an idea.

1.2 PlanetLab

PlanetLab currently has fully-operational federations with the following testbeds: PlanetLab Europe, VINI, M-Lab, GpENI, PLJ, PlanetLab Korea and Emanics Lab via PLE. Soon to be added to this list are two cloud computing resources: GENICloud and VICCI. These two future federations are the beginnings of PlanetLab's focus on next-gen cloud construction.

The current status of their SFA, a standard API based entirely off of the GENI API, is on version 1.0-20. This standard has been deployed fully on PLC and PLE testbeds. Current work has been focused on the interoperability effort with ProtoGENI. The main concern with this interoperability is a difference in RSpec formats. An agreement has been made to make changes to PLC's RSpec to accommodate ProtoGENI tools. ProtoGENI has a tool that understands both formats; however, PlanetLab does not. In effect, this creates a serious behavioral problem: ProtoGENI provides access to PlanetLab, but PlanetLab does not offer access to ProtoGENI.

Essentially, a ProtoGENI user can request PlanetLab resources, but the reverse has yet to be established.

Recent efforts have been made in the interest of user accessibility. This has led to the Sface GUI tool, a client for the SFA for PlanetLab Federated Resources. Written in PyQT, the code is currently available for download to be used by registered PlanetLab experimenters.

As apart of a growing interest in next-generation cloud infrastructures, work has begun on a PlanetLab cloud resource named VICCI. The current plan is to establish VICCI clusters at several international sites including the following: Princeton, GA Tech, UW, Stanford, MPI-SWS, ETH Zurich, and U Tokyo. These resources will be managed by PlanetLab CF; however, financial issues have arose, delaying deployment time.

In effect, the current plan for VICCI deployment is to have the first cluster online and available in less than two weeks' time, with plans to establish federation connections to PlanetLab and to be accessible to PlanetLab users through the newly-established Sface GUI. Other clusters will become available soon after, extending the availability of VICCI resources based on the needs of the user. Currently, plans for VICCI extension are modeled heavily on design ideas originating from the GENICloud project.

PlanetLab concluded with a set of goals for the coming months. These goals include the previously mentioned items of advertising the Sface GUI service to users and the beginnings of the VICCI project. Other items of interest include establishing a federation with GENICloud resources and advertising ProtoGENI resources to PlanetLab users as soon as the PlanetLab to ProtoGENI connection becomes reliable.

1.3 ORCA

The biggest news coming from the ORCA control framework group was the establishment of a new tool to convert NS2 resource specification documents to NDL. This tool is established and freely available via an XMLRPC server service, and offers conversions between ORCA and ProtoGENI formats. Also announced was the successful establishment of interoperability with the ProtoGENI control framework through an automated use of the conversion tool.

As part of a refactoring effort, ORCA has been focusing on a concept of embedding, or hiding resource location from clients and establishing cross-aggregate connections without having the client to specify the connections directly. This unspecified stitching operation is now established across multiple federated resources for networking layers 0, 1, and 2.

Other successful tools implemented by the ORCA control framework include work on several proxies that they plan to distribute in their control framework system in order to facilitate cloud computing resources. ImageProxy allows the user to share virtual appliance images across multiple independent cloud sites, thereby decreasing the overhead of re-imaging resources for a previously established experiment. This ImageProxy works as an interface between the ORCA Site Authority and the Eucalyptus private Cloud

DNAT Proxy, another proxy system recently configured, makes firewall resources available to users via public Ips. DNAT Proxy establishes a firewall between the Client and the Site Authority/Eucalyptus Slice. This system also shows a Public IP Proxy to mask resource location. The DNAT Proxy uses ShoreWall Software for the actual firewall, but does not replace the preexisting firewall established for the cluster.

These proxies are a part of ORCA's new focus on cloud technology. In accordance with this

new effort, ORCA has published interfaces for remote access. These interfaces are available to the experimenters as well as for overall system applications such as the proxies listed above.

Negotiations are underway to convince substrate resource testbeds to dedicate their resources to more substantial cloud infrastructure. These cloud systems are established via Eucalyptus cloud software and NEuca software, an extension added to ORCA equipment to allow for connections between ORCA clusters and Eucalyptus clouds. Note that NEuca does not affect the normal functionality of a private Eucalyptus Cloud, allowing for the unspecified nature of the internal topology that is intrinsic to the operations of a cloud. Also, it is important that of these three proxy systems developed by ORCA,

NEuca is the only Eucalyptus-specific software. The ImageProxy and DNAT Proxy both could be utilized in cloud-to-cluster systems that depend on other clouds such as those provided by Amazon.

2 Interesting Projects

Presentations were given about certain projects that the GPO highlighted as being interesting innovations occurring as a result of the GENI project.

2.1 Data-Intensive Cloud Computing

The goal is to extend Orca with Data-Centric Slices and connection to Amazon Elastic Compute Cloud (EC2) from GENI . From their system architecture, the following process occurs:

- 1) GENI provides network for sensors to collect data
- 2) Di S3 Proxy stores data to the cloud ,

3) AWS accounting processes S3 data on EC2 servers and stores results back to S3 data – actions logged in a database

4) S3 Proxy distributes data on S3 storage cloud

Major implementation and implications from this project include the demonstrated ability of DiCloud to implement building blocks for using Clouds , and the use of a proxy to monitor cloud API calls. All of the software conforms to the GENI AM API standard, due to the dependence on the ORCA testbed. This project also allows ORCA to connect to cloud resources via DiCloud.

DiCloud has two main concerns that they will improve upon in the coming months. One is that their CloudWatch software suite monitoring cloud API calls does not differentiate between free and paying network traffic, which raises financial concerns. The other is the dangerous absence of a security model to protect storage functionality.

2.2 WiMAX Projects and Deployment:

With respect to the meso-scale deployment integration goals of Spiral 2, WiMax technology has been distributed to six universities and several private companies that showed interest in collaborating on furthering the WiMax technological capabilities.

The eventual goal of the WiMax project is to establish an open source managerial software for deployed resources at university campuses. In the interest of achieving this goal, the software is based on NEC hardware, with little to no modification of the network architecture.

3 Software Development Focus Areas

As apart of an experimental refocusing of objectives moving forward with a federated GENI,

the GPO organized a track of key software development issues needing to be resolved. These issues were grouped into four major categories (Identity & Attributes, Authorization, Rspec, Stitching) and one exploratory group for defining a solid future development area (Instrumentation & Measurement). As my focus is on GENI architectures in general and specifically what mechanisms are being developed in the architecture to improve security, I did not attend the Instrumentation & Measurement meetings. The following are accounts of the proceedings of the Authorization, Rspec, and Stitching Meetings. These meetings were formatted as proposals to the GENI community to decide critical issues moving forward in these three areas. The proposals were followed by presentations from key stakeholders, most notably the administrators of the three main control frameworks, and finally agreements were reached as to how to proceed with the GENI project.

3.1 Authorization

The currently used authorization protocol is based on the credential protocol specified in the SFA 2.0 document. This familiar protocol is comprised of the exchange of signed XML documents. Most notably, the content in these documents can be specified accurately in a three tuple consisting of Roles which are defined by a list of Privileges which in turn specify available Operations. The second thing to note about these credentials that are overly complicated is that most of the decision processes about what user has what Roles or Privileges or if they are authorized to perform a certain Operation are delegated to the slice authority.

Both of these concerns, the unnecessary complexity of the credentials and the overbearing trust placed on the slice authority, inspired the proposal that was presented below.

3.1.1 Proposal

The Attribute Based Access Control protocol specified in ABAC Rules for GENI Authorization document should be added to the GENI AM API specification, but caution should be taken as to whether or not the previously established SFA credential protocol should be replaced.

Some of the goals outlined in the ABAC Authorization specifications include the need to support different authorization policies, users, and groups of users. Another major benefit of the ABAC protocol is that it establishes a uniform language for authorization policies.

Currently, as specified above, credentials hard-code the info of authorization in (Subject, Target, Privileges). Due to the general nature of GENI, these types, rights, and operations are continuously expanding and their definitions vary based on differing opinions by each control framework and localized testbed.

Attribute Based Authorization specifications seek to bridge this gap and allow for extendibility. The ABAC document referenced above establishes a formal attribute credential logic, the nature of which allowing for expansion of types/rights/operations definitions and establishing a cross CF credential format. Control framework definition expansion is expected, but a basic ABAC form should be kept uniform to allow cross CF authorization to depend upon it.

Furthermore, the plan moving forward is proposed that ProtoGENI will follow a previously specified integration plan laid out in the authorization-plan-v0.4 document available on the GENI wiki.

In this model, the Aggregate Managers are specified as the grantor of rights. ProtoGENI will establish ABAC credentials in conjunction with existing credentials to determine whether shifting to ABAC completely will fulfill all authorization needs. The results of this testing will hopefully be

available by GEC 11.

3.1.2 ORCA Perspective

The currently operating ORCA control framework is in agreement with SFA 2.0 and the GENI AM API. They propose that GENI use the ABAC protocol to encode attributes in a less literal way, and to distribute the ability to be the grantor of rights to more than the Aggregate Manager.

ORCA supports the policy of establishing a separate security architecture in order to support interoperability between control frameworks. This concept of a separate security architecture would thereby separate policy concepts from the mechanism that enforces it.

ORCA outlined the general implications of the ABAC agreement. The agreement endorses a concept of user identity, with assertions as to the definitions of their attributes, the need to delegate rights to these identities, and the need of an internal system depending on trusted root systems to validate a user's claims to attributes and rights.

According to the document outlining ABAC, any entity may serve as a trust anchor for its own name space of attributes. This could allow for flexibility instead of putting credential signing onto one entity. One of these trust anchors, Identity Providers are just trust anchors established by institutions.

Identity Providers thereby authenticate the user agent.

ORCA also highlights the fact that the concept and issues of Authorization are related to Resource Allocation . ORCA has resource allocation policy mechanisms in Aggregate Managers and brokering services which may consider ABAC; however, ABAC is not rich enough to support the

complexities of resource allocation.

The Aggregate Manager trusts and delegates control privileges for the objects they create, and ABAC delegation primitives are powerful enough to do that without any inference procedure required.

Despite these benefits, ORCA raised some concerns with ABAC. ABAC needs a limited form of parameterized roles/attributes. This type is the type specified in the ABAC document as RT1, and is suggested in order to keep the roles/attributes from constantly expanding and being incompatible when dealing with more than one control framework. Root ownership attributes should be in the object creator, not the Aggregate Manager as a security-related precaution.

Functionality should be added to allow for a parameterization of authorization options. Local attributes need to be added to objects and local object attributes need to be added to the ABAC.

There needs to be added functionality enabling

User Delegation of Authority. The suggested addition was a “SpeaksFor” attribute for automated controllers to determine who is responsible. Standard conventions need to be agreed upon for basic attributes and their flow through the ABAC protocol.

3.1.3 ProtoGENI Perspective

Currently, credentials are based on capabilities rather than roles. Standard Policies are good, but ABAC should be focused on a simple logic rather than trying to completely define such policies. There needs to be a way to communicate to users their set of capabilities or the extent of their role.

Tool support needs to be maximized in order to understand clearly what the authorization

credentials state.

On a more logistical perspective, a set time limit of dual-credential support needs to be established to keep focus on the reason for the dual-credential testing and to prevent the possibly antiquated SFA protocol to continue as background legacy code.

3.1.4 Discussion

An example of ABAC code is already implemented and located on the TIED wiki in the Reference Aggregate Manager for those who are interested.

A large supporting feature of ABAC is the active concept of a "partial proof" to show a user why they are not authorized to access a resource. This allows for the tracking of any authorization-related issues and for the user to determine how to proceed in getting access to something they believe they should be authorized to access. ProtoGENI does this through an error return, ORCA does it by returning an array of strings.

ORCA has an architectural mechanism within the broker that can manage resource allocation policies cross aggregate manager, and RT1 has the capability to support some form of resource allocation policy. Despite this, the ORCA stance is that ABAC RT1 is not rich enough to address resource allocation policy, and that the resource allocation policy should be implemented in the framework itself. ORCA claims that resource allocation policies do not have a place currently in GENI, but it needs to be addressed

3.1.5 Agreement

ProtoGENI and perhaps ORCA will try out ABAC for a year

3.2 Resource Specifications

Several Resource Specification formats have been proposed for GENI-wide use: RDF according to ProtoGENI RSpecs, NML, and NDL.

The basis of the GENI AM API is a need for a common RSpecs format to allow for interoperability.

The issue of developing a way to specify resource specification is not a new problem. Several other case studies have been suggested and were presented during the discussion. There has been work with perfSonar and NML to describe measurement resources. NDL, a format originating from the University of Amsterdam, is currently being used through the ORCA framework.

The final driving force for this discussion is the need for some basic concepts to form a common language so that solid, generalized software tools can be built.

3.2.1 Proposal

GENI will use ProtoGENI RSpec V2.

Aggregates can use Translators, using their own languages internally.

Establish Basic Concepts for Semantics in order to pursue the goal of defining a unified Resource Specification Language. Chances are that the use of ProtoGENI RSpec V2 across GENI will be temporary while a common language is being built.

3.2.2 ORCA Perspective

The problem with semantics for slicing a network is the existence of multiple dimensions of specification. Luckily, there currently exists an agreement on the resource representation cycle, which is a start. Translation is possible, but the differences in concepts of resource specification in different CFs poses the greatest difficulty. As we approach interoperability, translators might not be an option for a formalized GENI, therefore, precaution should be taken to not depend fully upon these converters.

Currently, there are Four Families of RSpecs: OMF (ORBIT), PlanetLap Rspec ,ProtoGENI Rspec, ORCA NDL-OWL. Diverse resources are also available in each CF and RSpec Family. This reality is the source of the concept differences in each Family's Semantics. NDL-OWL might be too extensive in descriptions due to the need it addresses to allow for allocation of resources on several different networking layers.

A driving force for developing this common language is that Stitching depends on a core RSpec Syntax.

Guidelines have been thought out for a practical solution. The descriptive focus should stay within the Ethernet layer and accept only one format. In the meantime, availability should be established to perform bi-directional conversions; however, only partial might be possible due to the fact that conversion to ProtoGENI RSpec V2 is limited to layer 2 and above . There is currently an example of such a converter available through ORCA.

There are a few emerging standards to keep in mind: OVF will be standardizing policy agreements and certification of portable appliance images across heterogeneous environments (inter-operable CFs), and there is work being done to establish higher-level programming environments.

In continuing issues surrounding resource specification, we should arrange for aligning edge compute resource descriptions. ORCA is willing to make their conversion software available and are willing to give it as a GENI-wide service, perhaps even extending it to convert between other formats.

Ontologies (abstraction models) for certain types of resources need to be developed by specialists in the domains. These domains are the following: Wireless resources, Measurement resources, Storage (physical and cloud), and OpenFlow technologies.

3.2.3 UNIS and the work with perfSONAR

UNIS came out of the need for the same schema for control and measurement resources. There was a driving need to express common attributes and elements, expressing complexity when necessary.

A small set of basic elements was established, with extensions available for layer and application specific properties. Strict distinctions were needed between layers. Namespaces in XML were used, but consideration was needed to maintain the ability to map the namespaces to URIs. If we can agree on basic elements, then translation becomes easy.

XSD and RNG formats have been decided against because they lack the needed semantics. YANG, a recent development, provides useful additional semantic power in a simplified format.

Observations resulting from UNIS experimentation was discussed in depth. Within the problem domain of Resource Specification, there exists a natural conflict between Expressiveness of the resources and the Complexity that such expressiveness can create. There exists a need to represent things beyond allocatable resources, which is a concern with measurement applications.

The main issue is how to represent the subnetwork location within the surrounding public network. Issues also exist with expressing substrate topologies .

UNIS has also developed their own translator, and supports the extended use of translators. However, this usage comes with a caveat: internal representation languages would have to be maintained independently.

3.2.4 Work with NDL

NDL, developed by Univ. of Amsterdam for use within their lab testbed, is currently being used as the standard for Resource Specification in Europe. In addition, an extensive Python toolkit has been developed to work with NDL documentation. The major goal of NDL is to accurately describe a Semantic Networking Workflow. There is also the current development of the OWL graphical interface that displays information expressed in Resource Specification Files.

3.2.5 ProtoGENI Perspective

ProtoGENI relies on XML with expandable namespace definitions. These namespace definitions form the cornerstone of their "External References" concept. The ProtoGENI team made it clear that their RSpecs were not based on a language, but a declarative format. Their RSpec documents come in three types, each with a different purpose and each with different information (advertisement, request, manifest).

ProtoGENI doesn't see any issues with converting RSpecs to and from the PlanetLab format because the only difference is that PlanetLab includes a <site> tag that details to which testbed site a resource belongs. Issues that might arise with ORCA's NDL is that there are extensions

mismatches causing difficulty for ORCA to access some of the extended federated resources in ProtoGENI.

ProtoGENI wants RSpec principles to continue to be at the forefront of any design decision. The format agreed upon should continue to be flexible. Limits need to be created on the amount of flexibility, with a defined standard base that can be extended. If there is a need to have different versions of this format running at once, efforts should be made to bound the number of active versions. Finally, as useful extensions are developed, there should be a way for these extensions to be added to the standard base upon approval.

To clarify what ProtoGENI means by “Extension” and “External References,” they provided a list of example extensions currently active in the ProtoGENI federation: Emulab, BGPmux, Stitching, OpenFlow, Proprietary.

3.2.6 OpenFlow Perspective

Having addressed a similar issue with resource specifications pertaining to switches, OpenFlow was willing to share its expertise and viewpoints. Their experience is relevant because switches are essentially nodes with DPID and ports numbered starting from 1. Links between these switches are two dpid:port pairs . Each link has an allowed “FlowSpace” or bandwidth which is the main difference in their RSpecs.

In their advertising RSpecs, they focus strictly topology, no flowspaces defined, due to Opt-In manager message flow. This allows for FlowSpace allocation to be dealt with separately.

The OpenFlow Request RSpec is a subset topology that notes the FlowSpace per link.

Unless otherwise specified, this FlowSpace allows for the same amount of traffic across the entire network. A major thing to note is that no bandwidth reservations are allowed.

In general, OpenFlow is not dedicated to this specific format and has plans to move forward with experimenting on using the ProtoGENI RSpec V2 format. They will of course need to define a FlowSpace extension for the ProtoGENI format to support their switch links. They might even extend their capabilities by creating a bandwidth reservation extension, a feature which they are not able to provide with their current format.

3.2.7 OMF view

There needs to be a conceptual separation of object model from semantic meaning. Semantic meanings will change over time, but object models should be stable. Attributes should be used as little as possible, with focus being maintained on elements. With extensions, elements are easily validated compared to attributes.

3.2.8 Agreement

Refactoring will occur to determine how much of the ProtoGENI RSpec V2 will be considered for the standard GENI base. This RSpec format will then be moved to the GPO for maintenance, much like the GENI AM API. An Extension Mechanism will be established based on the ProtoGENI extension model. Extensions will be developed for each CF based on currently available converters. The GENI AM API will be edited to reflect this decision. Ontologies will be developed for different types of resources.

3.3 Stitching

Stitching has two distinct parts: automating the configuration of switches and negotiating stitching resources. This proposal focuses on number two. Consideration is given to the fact that Network Stitching is not a unique problem, and the proposal outlined is a modest beginning.

This proposal starts by addressing the way CFs do things normally, and therefore focuses on satisfying the 90% case. The Overview Architecture document lays out a common schema, 6 key functions, and a common API. Details need to be defined about how this schema needs to be adapted as well as how it will be implemented. The current proposed documents are open to debate and alteration as the control frameworks or other interested parties seem fit.

This proposal focuses on the mechanics of stitching and the workflows involved. As a beginning and in accordance with the 90% case limitation, this proposed Stitching focuses on Layer 2 (Ethernet VLANs), but should be extendable. Issues of External Network navigation are present, and the proposed architecture depends on heterogeneous Aggregate Managers and interconnected topology.

Therefore this implies the need to conform to the GENI AM API.

Architecture Components are defined as six distinct pieces:

Stitching Resource Element – dependent upon RSpec format decisions (Stitching information could be included in common RSpec, or could be developed as a standalone)

Common Stitching Topology Schema – pieces that create the global view

Stitching Topology Service – Communicates between AMs (AMs are not responsible for cross-aggregate communication), also allows for a global view by assembling CSTS pieces offered by the Ams.

Stitching Path Computation Function

Stitching Workflow Function

GENI AM API Stitching Extensions

Three models are proposed for the Stitching Topology Service: Chain, Tree, or Hybrid.

These are provided in more detail in the Architecture Overview document. The Stitching Workflows and Path Computations are dependent on the Stitching Topology Service, and as such are not yet fully detailed.

However, it is worth noting that the Chain model and the AM-Initiated Tree Model force the Stitching Functionality to be located in the Aggregate Manager. There is a possibility of implementing a Proxy AM to talk to native AMs in the Chain Model. This suggestion is interesting because Proxy AMs have other benefits outside of Stitching.

The goal is to detail a solid common API that could handle both a Chain Model and a Tree Model, in essence creating a Hybrid Model – This would force an extension of the GENI AM API to offer Stitching functionality. The Topology Service is meant to be static and to seed the Stitching Path Computation and Workflow Functions with data for developing a global view of the stitching and therefore establish a certain Workflow Model. There is an example of a standalone Stitching RSpec schema available on the GENI wiki ([stitching-protogeni-rspec-v2-style.xml](#)). This Schema is closely related to the UNIS Schema.

The following can be developed separately, but are recommended to be apart of a GENI Infrastructure Specification: Stitching Topology Service, Stitching Path Computation, Stitching Workflow Function.

3.3.1 ProtoGENI Perspective

The ProtoGENI architecture follows closely with the Chain Model for several reasons. The flows in the Chain Model avoid issues of global locks completely. In addition, the model hides information about stitching specifics from Clients. This type of stitching model is already common in other Network Stitching implementations. The needed extensions to Resource Allocation is explicit and visible to developers, allowing for them to deal with writing other software parts such as Client modules more easily.

The ProtoGENI team went over their conceptual understanding of the interconnections involved in the proposed Stitching architecture and the previously discussed Resource Specification issues. Links, whose specifications were discussed earlier, are clearly defined as the objects that cross aggregates. Because of this, the Resource Specification implementation needs to be aware of Stitching concerns. Thankfully, the ProtoGENI RSpec V2 supports an Extension Mechanism, and a loose definition of such an Extension to support Stitching exists.

Things about this Stitching Extension should be known before other teams get their hopes up. Currently, the extension does not have a stitching topology service; however it would be easy to build because advertisements of AMs are publicly available in ProtoGENI. A Path Computation Function exists (SES) , but the really small API probably needs to be extended. A Stitching Workflow Function exists inside the AM, but could be removed and added somewhere else. This Stitching Workflow Function currently establishes a P2P connection with other AMs. The extension does not support loose requests (stitching points are explicitly identified). A new API call is available at AM to support the Stitching Workflow Function

Additions to this extension could be the building of a Proxy AM for ION connections. Also

there has been talk of testing on more than P2P connectivity between AMs.

3.3.2 ORCA Perspective

Stitching is more than topology. Stitching occurs within the sliver and logical linkages, therefore multilayer adaptations require a focus on cross-layer dependencies. There exists produce/consume labels (tags) common to other cases of stitching.

ORCA has a sequenced stitching model controlled by the SM. Should be able to infer the DAB from representations of substrates and slices; however, these complex substrates raise issues of their own.

ORCA has a stitching architecture that conforms to model (3a) the Tree Model that is Initiated by the SM. The reasoning for this decision that that putting the stitching function in the SM assists the Client and makes it easier, but the SM acts for the Client .

ORCA does not want to change parts of their CF architecture to conform to the new Stitching Architecture, and firmly supports the Stitching Functionality being supported by the Slice Manager.

3.3.3 OpenFlow Perspective

Most Stitching in the OpenFlow system occurs by hand, through a tree model. In addition, a service provided by OpenFlow, automatic topology discovery, currently depends on such a Stitching model. Of course, they are not happy with their current Stitching model because it does not scale, is error prone, and is difficult to use.

The OpenFlow architecture can be expanded to implement either one of these stitching

architectures, as long as the AM implementations are solid. Concerns were raised that they might need to add some sort of OpenFlow linktype to the Stitching functionality instead of defaulting to VLANs only.

The OpenFlow team noticed several problems that could occur in this Stitching architecture. How to make a common AM API to support both Chain and Tree stitching? Can a chain node pretend to be a stitching service for a tree node? This question is focusing on the implementation of stitching between AMs that are willing to negotiate links in the chain model and AMs that require a global identity to negotiate links for them, such as in the tree model.

There are two main use cases that exist in network stitching: Loose source routing, which encourages the chain model; and Strict source routing, which encourages the tree model. Due to this, we should probably allow for the implementation of both models, which is indeed the proposed hybrid model offered by a common AM API for Stitching functionality

3.3.4 Decisions

The Community decided to support the Architecture Overview Document with a focus on the Hybrid model. There seems to be a need to generate some use cases that fully flex stitching between Chain Model and Tree Model in order to understand what issues may arise.

Before next GEC, stakeholders should consider a few issues. A solid Common Stitching API for the AM needs to be considered. In turn, someone needs to draft a full schema based on the Architecture Overview Document. Other ideas might be proposed through the mailing list .

4 Proposals for Future Plans

We were able to make contact with Justin Cappos, the lead administrator for the Seattle Testbed project. He invited us to explore security concerns with his system and offered to provide source code and other resources for testing. Currently, we still need to maintain contact with him to further pursue this opportunity.

Aaron Falk, Engineering Architect and Lead System Engineer at the GENI Project Office, met with us to discuss our progress and concerns that he had with the lack of OpenFlow security tests. We plan to also reach out to those that work on the OpenFlow systems to obtain source code and other resources for testing.

In addition, it should be interesting to observe the implementation of several concepts discussed and agreed upon during GEC 10. These decisions will reshape the GENI AM API, provide for the establishment of a GENI SM API, and might even contribute to the construction of a more substantial security architecture inside of the control frameworks.