2014

The GpENI testbed: network infrastructure

Medhi, Deep; Ramamurthy, Byrav; Scoglio, Caterina; Rohrer, Justin P.; Cetinkaya, Egemen K.; Cherukuri, Ramkumar; Liu, Xuan; Angu, Pragatheeswaran; Bavier, Andy; Buffington, Cort...

The GpENI Testbed: Network Infrastructure, Implementation Experience, and Experimentation. Deep Medhi, Byrav Ramamurthy, Caterina Scoglio, Justin P. Rohrer, Egemen K. Çetinkaya, Ramkumar Cherukuri, Xuan Liu, Pragatheeswaran Angu, Andy Bavier, Cort Buffington, James P.G. Sterbenz, In Computer Networks, vol. 61 iss. 0, March, 2014, pp. 5174. Special issue on Future Internet Testbeds -- Part I

http://hdl.handle.net/10945/59962

Computer Networks 61 (2014) 51–74

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

The GpENI testbed: Network infrastructure, implementation experience, and experimentation

Deep Medhi [a,g,*], Byrav Ramamurthy [b], Caterina Scoglio [c], Justin P. Rohrer [d], Egemen K. Çetinkaya [d], Ramkumar Cherukuri [a], Xuan Liu [a], Pragatheeswaran Angu [b], Andy Bavier [e], Cort Buffington [f], James P.G. Sterbenz [d,h]

[a] University of Missouri–Kansas City, Kansas City, MO, USA
[b] University of Nebraska–Lincoln, Lincoln, NE, USA
[c] Kansas State University, Manhattan, KS, USA
[d] The University of Kansas, Lawrence, KS, USA
[e] Princeton University, Princeton, NJ, USA
[f] KanREN, Lawrence, KS, USA
[g] Indian Institute of Technology–Guwahati, India
[h] Lancaster University, Lancester, UK

ARTICLE INFO

Article history:
Received 19 December 2012
Received in revised form 29 November 2013
Accepted 26 December 2013
Available online 3 January 2014

Keywords:
Programmable future Internet testbed
Network virtualization
Dynamic circuit network

ABSTRACT

The Great Plains Environment for Network Innovation (GpENI) is an international programmable network testbed centered initially in the Midwest US with the goal to provide programmability across the entire protocol stack. In this paper, we present the overall GpENI framework and our implementation experience for the programmable routing environment and the dynamic circuit network (DCN). GpENI is built to provide a collaborative research infrastructure enabling the research community to conduct experiments in Future Internet architecture. We present illustrative examples of our experimentation in the GpENI platform.

1. Introduction

Deploying large-scale network testbeds brings significant benefits to researchers to conduct scalable network experiments and evaluate the performance. Those testbeds provide fundamental capabilities such as high-speed infrastructure, programmable network nodes, and open access permission to registered researchers so that they are allowed to use the network resources through the open API provided by the testbed. Two prominent network research programs in the past five years are GENI (pronounced as 'genie') [12,41] and FIRE [9], and both focus on the Future Internet architecture design and relevant technological development, while supporting creation of experimental testbeds. However, it is important to remember that the idea of large-scale testbeds on which to conduct networking research is not new. In this section, we summarize a few of the most relevant previous efforts on network research testbeds.

- Gigabit Testbeds: A set of testbeds was constructed in the early 1990s to further the state of high-speed networking research, funded by the US NSF and DARPA (Defense Advanced Research Projects Agency), managed by CNRI (Corporation for National Research Initiatives). Five separate testbeds were constructed, Aurora, Blanca, Casa, Nectar, and Vistanet [49], later supplemented by MAGIC [20].

* Corresponding author at: University of Missouri–Kansas City, Kansas City, MO, USA. Tel.: +1 816 235 2006.
E-mail addresses: dmedhi@umkc.edu (D. Medhi), byrav@cse.unl.edu (B. Ramamurthy), caterina@ksu.edu (C. Scoglio), rohrej@ittc.ku.edu (J.P. Rohrer), ekc@ittc.ku.edu (E.K. Çetinkaya), ramkumar.ch03@gmail.com (R. Cherukuri), Xuan.Liu@umkc.edu (X. Liu), acb@cs.princeton.edu (A. Bavier), cort@kanren.net (C. Buffington), jpgs@ittc.ku.edu (J.P.G. Sterbenz).

1389-1286/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.bjp.2013.12.027

The Gigabit Testbeds were a platform for research in high-speed networking, new bandwidth-enabled applications [58], and networked supercomputing.

- Active Network Testbeds: In the late 1990s, testbeds were constructed in the US and Europe to support active networks research. *Active networks* are programmable networks in which one of the programming modalities includes *capsules* of mobile code that can dynamically program network nodes. In the US, the ABone [42] was constructed as part of the DARPA-funded Active Networks Program [1], to permit experimentation on programmable network languages, management and control [54], node operating systems [65], and security mechanisms [50]. The ABone had the goal of open access to the research community. In Europe, the EU FP5 FAIN (Future Active IP Networks) [52] and related projects (e.g., LARA++ [47]) also investigated active and programmable networks, with testbeds constructed for experimentation. These active network architectures and testbeds permitted sharing infrastructure by the simultaneous execution of active applications (AAs) in execution environments (EEs) on a network node operating system (NodeOS). While not generally recognized in this manner, the active network testbeds had many goals similar to that of GENI, and should be considered a conceptual precursor.
- Modern Large-Scale Testbeds: More recently, two large scale testbed infrastructures have been constructed with the explicit goal of permitting open access for networking research. PlanetLab [26] is a worldwide infrastructure that permits users to run networked experiments on a large scale. The infrastructure is shared using the *slice* paradigm. It is important to note that while PlanetLab permits experimentation in networked applications and end-to-end protocols, the network itself is not programmable, and experiments in lower-layer protocols can only be performed on overlays. VINI [29] provides a virtual network infrastructure that is built on PlanetLab. VINI allows researchers full control to create virtualized arbitrary network topologies where routing software can be invoked for experimentation. Emulab [6] is a network testbed consisting of a cluster of computing nodes interconnected by flexible network infrastructure, which permits researchers to experiment with network protocols and applications with complete root access to the systems. A number of Emulab facilities are located throughout the world, some of which provide access to external researchers in addition to the main facility at the University of Utah. Both PlanetLab and EmuLab are the basis for GENI control frameworks; GpENI uses the PlanetLab control framework.
- Current Future Internet Initiatives: While a number of researchers proposed alternatives to the Internet architecture as early as the 1980s (including research programs such as DARPA Next Generation Internet – NGI), there is now a general consensus in the research community that the current architecture is limiting in scale and support for emerging application paradigms such as mobile and nomadic computing and communications. Recent research initiatives include NSF FIND (Future Internet Design) [23] in the US, EU FP6 SAC (Situated and Autonomic Communications) [11], and the research component of FP7 FIRE (Future Internet Research and Experimentation) [9]. These research initiatives aim to investigate clean slate (greenfield) as well as incremental (brownfield) architectures to evolve the Future Global Internet architecture. A key problem remains how to experiment with Future Internet architectures on a reasonable scale. For this reason, the NSF GENI (Global Environments for Network Innovation) program [12], the experimental component of the EU FP7 FIRE programme [9], and the Japanese JGN2plus [18] testbeds plan to deploy large-scale programmable testbeds for experimentation of the Future Internet research.

The scope of this paper is to give a comprehensive presentation on the GpENI testbed from three different aspects: network infrastructure, implementation experiences, and experimentation. This comprehensive work is built on our earlier conference and workshop papers [36,48,60,68]. The rest of the paper is organized as follows. In Section 2, we present the motivation and overview about the GpENI testbed. In Section 3, we present the physical topology of the GpENI testbed over the United States, Europe and Asia, as well as the infrastructure design. In Section 4, we give a high level description on the GpENI node cluster architecture. In Section 5 and Section 6, we present a detailed discussion on the network layer and the optical layer programmability of the GpENI testbed, including both architecture design and preliminary results. We discuss our current federation status in Section 7. In Section 8, we discuss the experimentation work we have done with the GpENI testbed. A recent GpENI extension to KanREN-GENI is briefly described in Section 9. We summarize this paper in Section 10.

2. GpENI testbed: Motivations and overview

The Great Plains Environment for Network Innovation – GpENI (pronounced as 'japini', rhyming with GENI) is an international programmable network testbed centered on a regional optical network between The University of Kansas (KU) in Lawrence, the University of Missouri–Kansas City (UMKC), the University of Nebraska–Lincoln (UNL), and Kansas State University (KSU) in Manhattan associated with the Great Plains Network, in collaboration with the Kansas Research and Education Network (KanREN) and the Missouri Research and Education Network (MOREnet). GpENI has been extended to several sites in Europe. GpENI started with funding provided initially through the NSF GENI program.

The goals of GpENI are to:

- Build a collaborative research infrastructure, and construct an international programmable network infrastructure enabling GpENI member institutions to conduct experiments in Future Internet architecture and autonomic management.

- Enable the capability of dynamically creating circuits of specified requested bandwidth not only across GpENI testbed among participating universities, but also support inter-domain dynamic circuit creation.
- Provide a flexible infrastructure to support the GENI program as part of GENI node Cluster B, which uses a PlanetLab-based control framework.
- Provide an open environment for networking research community experiments.
- Provide an interface for the GpENI community to use resources from other testbeds via federation.

GpENI aims to enable programmability not only at upper layers such as the application or transport layer, but also at the network layer and even at the optical layer. As shown in Table 1, application and transport layer programmability are provided by a private instance of PlanetLab (MyPLC). At the network layer, programmable routers are implemented in the routing software suite (i.e., Quagga or XORP). Flexible network-layer virtualization is provided by GpENI-VINI, which is a customized private instance of VINI [29]. At the optical layer, dynamic VLAN configurations are provided by dynamic circuit network (DCN) enabled Gigabit-Ethernet switches at the center of each GpENI node cluster. GpENI institutions directly connected to the optical backbone use DCN-enabled Ciena switches to provide dynamic lightpath and wavelength configuration.

3. GpENI network infrastructure and topology

The core of GpENI is the regional optical backbone centered around Kansas City. This is extended by KanREN (Kansas Research and Education Network) to various GPN (Great Plains Network) institutions located in the Midwest region of the US. Connectivity in Kansas City to Internet2 provides tunneling access to the European GpENI infrastructure. GpENI is growing, currently with about 38 node clusters in 17 nations, including KanREN, G-Lab, and NorNet. Institutions may connect to GpENI if they are interested in becoming part of the GpENI community, and manage a node cluster. GpENI runs a PlanetLab implementation of Slice Facility Architecture (SFA) [59] to allow the application/transport layer and the network layer federation, and GpENI also allows federation at Layer-2 with the dynamic circuit network (DCN) software suite.

GpENI is built around the core GpENI optical backbone centered in the great plains, shown in Fig. 1, among the principal institutions of KU, UMKC, UNL, and KSU, including the GMOC (GENI Meta-Operations Center). The optical backbone consist of a fiber optic run from KSU to KU to the Internet2 PoP in Kansas City, interconnected with tunnels to UMKC and UNL, as shown in Fig. 2.

Table 1
GpENI programmability layers.

Layer	GpENI Layers	Programmability
Application	Application	PlanetLab
Transport	End-to-End	
Network	Router	XORP & Quagga
	Topology	GpENI-VINI
Layer-2	VLAN	DCN
	Lightpath	

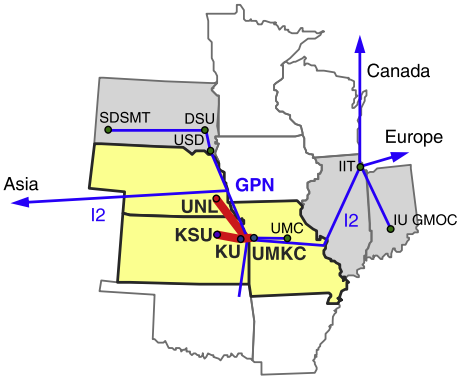


Fig. 1. GpENI US Midwest topology.

Each of the four core institutions will have a node cluster that includes optical switching capabilities provided by a Ciena CoreDirector or CN4200, permitting a flexible spectrum, wavelength, and lightpath configurations.

GpENI is extended to Europe across Internet2 to GÉANT2 and NORDUnet and then to regional or national networks, as shown in Fig. 3. Currently, connectivity is achieved using L2TPv3 and IP tunnels. A direct fiber link over JANET is deployed between Lancaster and Cambridge Universities. The principal European GpENI institutions are Lancaster University in the UK and ETH Zürich in Switzerland.

4. GpENI node cluster architecture

Each GpENI node cluster consists of several components, physically interconnected by a managed Netgear Gigabit-Ethernet switch to allow arbitrary and flexible experiments. GpENI uses a KanREN 198.248.240.0/21 IP address block within the gpeni.net domain; management access to the facility is via dual-homing of the Node Management and Experiment Control Processor. The node cluster is designed to be as flexible as possible at every layer of the protocol stack, and consists of the following components, as shown in Fig. 4 (the curved arrow shows the logical flow):

- GpENI management and control processor: general-purpose Linux machine.
- Control framework consisting of aggregate managers: MyPLC with SFA, MyVINI with SFA, and DCN.
- MyPLC programmable nodes.
- GpENI Virtualized Network Infrastructure (GpENI-VINI) providing flexible virtual network topology creation with programmable routers allowing an experimenter to choose a routing software suite, either Quagga or XORP. GpENI participants are able to use node resources from public PlanetLab and public VINI with the SFA client.
- Ciena optical switch running DCN providing Layer-2 programmability among GpENI node clusters, and inter-domain Layer-2 programmability between GpENI and MAX testbed.

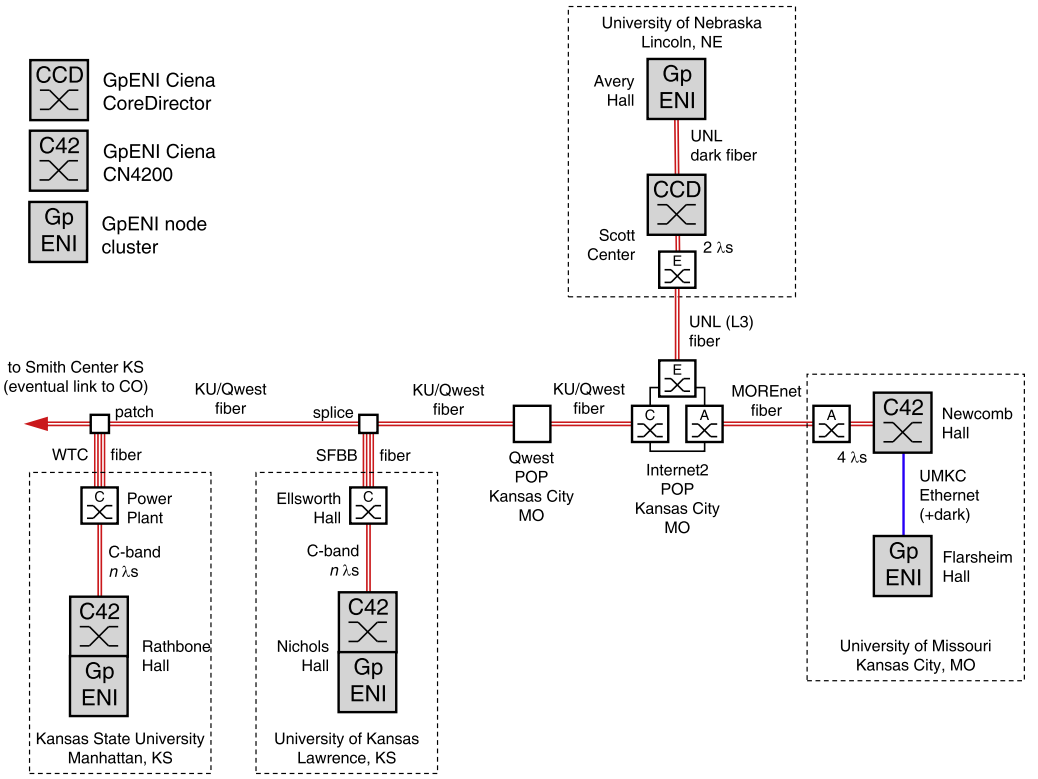


Fig. 2. GpENI US Midwest optical backbone.

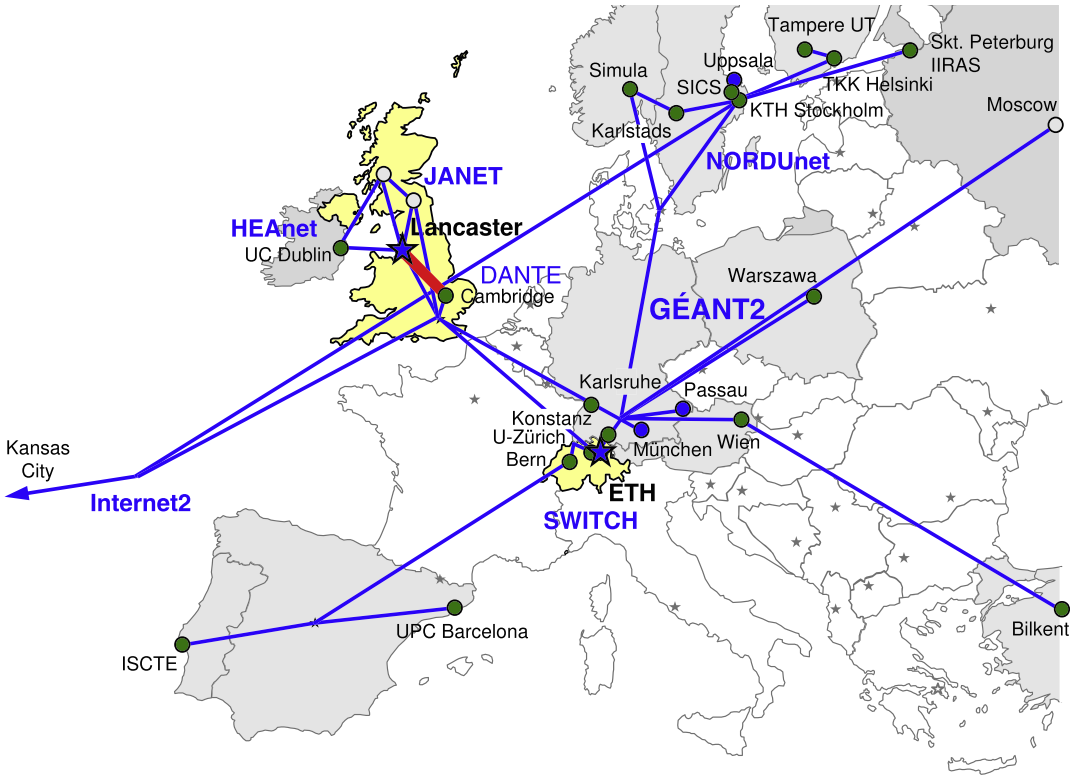


Fig. 3. GpENI European topology.

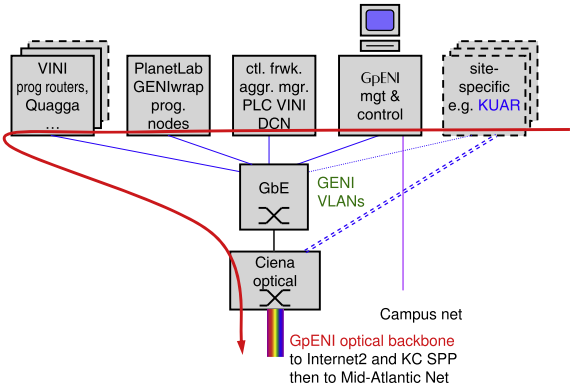


Fig. 4. GpENI node cluster.

4.1. GpENI management and control

The GpENI management and control services are distributed across the Linux machines dedicated for the purpose at each of the node clusters. Open-source tools are used wherever possible to minimize the amount of GpENI-specific software development and maintenance required. Some of these services are installed at every node cluster, for example the Cacti monitoring tool [2] is used to monitor the per-port network usage on each of the Netgear Gigabit-Ethernet switches. Nagios [21] is used to monitor the status of individual nodes and services across all the clusters. Zenoss Core [31] is also being evaluated as an alternative to Nagios.

The control node for each cluster also provides firewall and NAT services using Firestarter [10] for that cluster's private subnet, thereby protecting insecure devices, such as the Netgear switch telnet and SNMP management interfaces, from direct exposure to the public Internet.

4.2. GpENI-MyPLC control framework sub-aggregate

The GENI Project has four basic node clusters, and GENI Cluster B is based on the PlanetLab control framework. The PlanetLab Control Framework provides the control software to implement the control plane, data plane, management plane, and operations plane functionalities. To better understand this framework, it requires familiarity with a number of terminologies such as slice and sliver, and they are summarized in the Appendix A. There are a number of distinct *aggregates* belonging to Cluster B: PlanetLab nodes, VINI nodes, Supercharged PlanetLab Platform (SPP) backbone nodes [28], OpenFlow switches [25], and GpENI node clusters.

The GpENI aggregate consists of three sub-aggregates: the MyPLC sub-aggregate, the routing and topology sub-aggregate, and the DCN sub-aggregate (discussed in the following sections). MyPLC is a private instance of PlanetLab that runs the same control framework as PlanetLab does. GpENI-MyPLC sub-aggregate provides the programmability at both the application layer and transport layer, so that the researchers can run their application experiments within their slice on the GpENI-MyPLC sub-aggregate. Currently, GpENI-MyPLC has been federated with PlanetLab, so researchers are able to use the PlanetLab node resources from the GpENI-MyPLC aggregate interfaces. A tutorial on how to use GpENI-MyPLC is available from the GpENI wiki page [14].

4.3. GpENI-VINI sub-aggregate

For Layer-3 programmability, GpENI provides programmable topologies using MyVINI and an arbitrary number of programmable routers in each node cluster. We will use GpENI-VINI for a customized MyVINI to indicate the network layer virtualization infrastructure on the GpENI network testbed in the rest of this paper, in order to differentiate from the original MyVINI.

GpENI-VINI runs a private instance of public VINI and extends a few features from the public VINI to provide more flexible resources provisioning. The public VINI enables virtual topology creation on top of the physical networking infrastructure, and it is essentially a flavor of PlanetLab with a set of enhancements to the PlanetLab kernel and tools called *Trellis* [39]. Trellis allows users to create their own virtual topology in their slices, either automatically using the IIAS (Internet In A Slice) toolkit or manually designating links between *slivers*. GpENI-VINI extended the features of the IIAS tool to allow a researcher to create an arbitrary virtual topology in a slice, and automatically create the virtual links between the virtual nodes in the virtual topology.

The Trellis [39] software system combines both host and network virtualization in a single system. Trellis allows a GpENI-VINI node to be sliced into multiple virtual slivers that can be configured as Layer-3 virtual programmable routers by hosting the routing software suites such as Quagga [27] and XORP [8]. Those routing softwares support a wide range of existing routing protocols. For example, Quagga supports RIPv1, RIPv2, OSPFv2, BGP-4, RIPng, and OSPFv3. However, these programmable routers have very limited processing power and can only handle moderate size forwarding tables compared with realistic routers in backbone networks since they are running in commodity PCs. GpENI-VINI is initially running Quagga and XORP. Details on GpENI-VINI architecture and implementation will be described in Section 5.

4.4. DCN sub-aggregate

GpENI uses DCN for control of VLAN interconnections among L2TPv3 tunneled node clusters as well as optical switches connected directly to the core backbone.

In recent years, the Internet2 network has evolved from a pure IP-based packet-switching network into an advanced hybrid optical and packet network. Apart from the traditional IP service, the new Internet2 network offers a virtual circuit service to provision dedicated bandwidth across the network, called the Internet2 Interoperable On-demand Network (ION) [16]. The ION service is dynamic and can be used to set up short term connections by a requestor or an application through a web interface. The control plane software that automates the set up and tear down of the circuits was developed for the Internet2

dynamic circuit network (DCN) research prototype and leverages technology developed by DRAGON (USC/ISI East, MAX, and George Mason University), GÉANT2, and the DOE ESnet (OSCARS project).

We made necessary changes on the current network infrastructure of the GpENI to establish DCN across GpENI. This enables the creation of on-demand circuits at the required bandwidth for specified durations using the DCN software suite. Deploying DCN across GpENI will also facilitate setting up VLAN circuits across the Ciena CoreDirectors located at various locations in Internet2. The CoreDirector Component Manager Interface [36] describes the use of the CoreDirector in the GpENI testbed. As additional GpENI optical switches are deployed, a common GpENI–wide DCN testbed will emerge over a multidomain network with CoreDirectors forming the optical domain and Netgear switches forming the Ethernet VLAN domain at each GpENI institution. More details about DCN deployment in the GpENI network testbed will be discussed in Section 6.

5. GpENI-VINI: Architecture and implementation

In previous sections, we have discussed the GpENI network infrastructure from the backbone topology to node cluster architecture. In this section, we focus on details about network layer programmability in the GpENI network testbed through GpENI-VINI. In short, GpENI-VINI is a virtual network resource provisioning testbed to support programmable routing experiments. The core architecture of GpENI-VINI is a customized private instance of VINI [29] by extending the flexibility of conducting experiments in a virtualized network environment, from a user's perspective.

5.1. GpENI-VINI core architecture

The GpENI-VINI testbed is a geographical-distributed network infrastructure, where all physical GpENI-VINI nodes are under control of a central server. Fig. 5 depicts the GpENI-VINI core architecture on how GpENI-VINI nodes and the researchers interact with the GpENI-VINI Central Server. The major components of the GpENI-VINI architecture are MyPLC [53] and the IIAS (Internet In A Slice) tool [15]. MyPLC is portable PlanetLab central (PLC) software; this acts as a resource manager on the GpENI-VINI testbed. It has both a web interface and an API interface. The web interface facilitates easy access and management of user accounts. With the API interface, researchers can access data with a command line interface through XML-RPC. The IIAS tool helps researchers to create virtual interfaces and virtual links inside a slice, and it includes a set of programs consisting of two parts: server side programs and client side programs. In the following subsections, we explain the architecture of the GpENI-VINI central server and the GpENI-VINI node in detail.

5.1.1. GpENI-VINI central server

The GpENI-VINI central server is responsible for the testbed management from all aspects, such as sites (i.e., university), resources (nodes), users, and slices. The key components of GpENI-VINI are illustrated below:

- MyPLC: MyPLC is portable PlanetLab software; by using this, we can create a private PlanetLab. It acts as the manager of GpENI-VINI resources. From a management point of view, it is a combination of four components: a web server, an API server, a database server, and a boot server.
  - *Web Server* provides the web interface to researchers and the administrator. By using this interface, researchers can create accounts, create slices, and select resources from the GpENI-VINI testbed. An administrator can enable, disable or delete users, sites, and nodes. An administrator can also modify the data of sites, users, nodes, or can add content to the GpENI-VINI Server.
  - *API Server* is an interface between the database server (PostgreSQL) and other components of GpENI-VINI. MyPLC provides a few API methods to allow accessing data by using these methods. The API server listens on a port for incoming XML-RPC calls. Based on the incoming request method, first it authenticates the requestor, then it sends the request to the database server to get the data from the database and returns the result to the requested component.
  - *Database Server*, based on PostgreSQL, is the primary storage space of GpENI-VINI resource data. Its function is to process the API server requests and send the results to the API server.
  - *Boot Server* provides the required software for GpENI-VINI nodes. Software includes the boot OS and Node manager.
- IIAS Tool: Server-side IIAS Programs are used to create a virtual infrastructure on the GpENI testbed. They assist researchers in selecting a virtual topology and in creating a virtual topology inside a slice. Server side programs create topology resource specifications of virtual links inside a slice. Once these topology resource specifications are created, they are stored in the GpENI-VINI database. We have extended IIAS server-side programs to support arbitrary topology creation with general users' privileges, which provide more flexibility of experiment design within a slice. We will explain IIAS features from the implementation perspective in Section 5.3.3.

5.1.2. GpENI-VINI nodes: Programmable routers

GpENI-VINI Nodes are the physical machines to host multiple slivers as software-based routers by running the routing software suite like Quagga [27] or XORP [8], and they are geo-distributed and are available globally to experimenters. Each physical node is running a customized software system called Trellis [39] that combines two types of virtualization technologies. The client-side IIAS programs are responsible to create virtual interfaces with an assigned virtual IP address for the slivers, which are added in a virtual topology in a slice. An overview about the major components of a GpENI-VINI node is enumerated as follows:

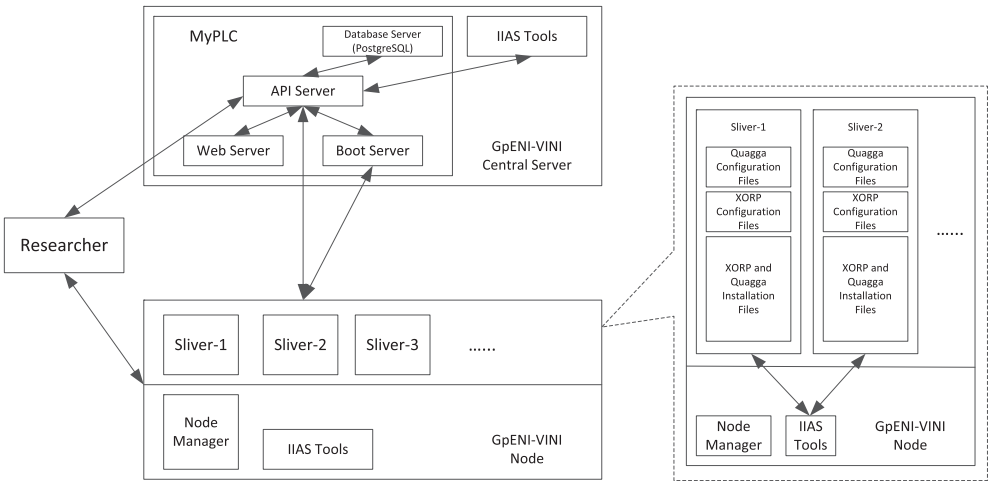


Fig. 5. GpENI-VINI architecture overview.

- Trellis Node: Trellis [39] is a customized software system for nodes in the GpENI-VINI testbed. It is a combination of two virtualization technologies, Linux VServer [19] and NetNS [22], to support virtual nodes, virtual interfaces, and virtual links inside a node.
- Node Manager: Node Manager is a daemon on the node that manages the node. It polls the data from the server at regular intervals and makes changes accordingly. Examples of changes include creating slivers and deleting slivers. It provides API for remote access and calls IIAS tools that are responsible for creating virtual interfaces and virtual links.
- IIAS Tool: Client (GpENI-VINI node) side programs are started by the node manager. These programs get the topology resource specifications from the database. Based on the topology resource specifications of a slice, these programs create virtual interfaces inside the sliver (virtual host context) and tunnel interfaces in the node (root context). By using tunnel interfaces, virtual links between slivers are constructed.
  Apart from the job of creating virtual and tunnel interfaces, these programs also provide support for Quagga and XORP to conduct routing studies, as shown in Fig. 5. To support Quagga functionality on a virtual network, the IIAS tools write Quagga installation and Quagga routing configuration files for routing protocols such as RIP and OSPF into each slice file system. In the same manner, to support XORP on a virtual network, IIAS tools write XORP installation and XORP configuration files for a routing protocol into each slice's file system (more details on XORP implementation in Section 5.3.2). These configuration and installation files can be used by researchers.

5.2. GpENI-VINI resources

As we mentioned in the beginning of this section, GpENI-VINI is responsible for the provisioning of virtual network resources that can be accessed through a slice interface. The virtual network resources inside a slice are available as long as the slice remains valid. By using virtual network resources we can build programmable virtual networks inside a slice. The virtual network consists of two components: virtual hosts and virtual links. The virtual networks resemble the real routable networks and provide a high degree of control to a researcher. The virtual network resources are built by using Trellis, a customized software system [39]. All GpENI-VINI nodes run Trellis software, available from a GpENI-VINI server as a boot image. Details of the Trellis design can be found at [39].

5.2.1. Trellis overview

Originally, the Trellis software platform was designed to support multiple programmable virtual networks on a single hardware system (a VINI node) and was designed to run on VINI nodes with the following properties:

- Speed – Packets should be forwarded at high speed in the virtual network.
- Isolation – It should provide isolation between virtual networks, i.e., one virtual network on one slice does not interfere with other virtual networks in different slices; it should provide isolation at the system level and the network level.
- Flexibility – It should provide the flexibility to researchers to select their routing protocols (including any modification) in a virtual network environment.
- Scalability – It should be able to simultaneously support a greater number of programmable virtual networks.
- Low cost – Because it can run on a normal system, it should decrease the cost of hosting virtual networks.

The Trellis software system combines both host and network virtualization in a single system to meet the above listed desired properties. For host virtualization, Trellis uses a container based virtualization technology called the Linux VServer [19]. The main advantage of the Linux VServer is that it provides OS-level virtualization instead of full virtualization. It also gives acceptable speed and scalability with reasonable isolation and flexibility that are

critically required properties. To provide network stack virtualization, Trellis uses NetNS (Network Name Space) [22]. Network Name Space virtualizes all access to network resources from the root system to the container system. It gives network containers with its in-kernel virtual devices, IP table settings, FIB, and so on. Fig. 6 presents an illustration of Trellis architecture.

We note that although isolation is an important property of Trellis, it is not completely possible to achieve isolation at the performance level due to each virtual network competing for resources from a common physical node.

5.2.2. Virtual node

Virtual nodes are built inside a GpENI-VINI node that runs Trellis software. It allows researchers to host multiple virtual networks on shared hardware (GpENI-VINI nodes). Trellis allows researchers to program their virtual topology based on their requirements such as a star, a mesh, or a fully-connected network. It also allows the researchers to select routing protocols such as RIP, OSPF, or BGP and allows them to define their own forwarding tables. Besides Linux Vserver and NetNS, the Trellis system implements a new tunneling mechanism called EGRE (Ethernet over Generic Routing Encapsulation (GRE) [51] tunnel) that allows the node to support virtual hosts and virtual links.

5.2.3. Virtual link

Virtual links give an illusion of a direct physical link between two systems, although they may be situated at multiple hops away. In GpENI-VINI, the virtual links are created between the slivers inside a slice. By using this virtual link analogy, we can build a virtual topology between nodes inside a slice. Fig. 7 illustrates a virtual link between slivers in a slice. Since, in reality, virtual links may be built over the commodity Internet (such as using L2TP tunnel), it is not entirely possible to avoid performance impact. Recently, it was reported in [56], which used GpENI for experimentation, that it is important to conduct multiple runs of an experiment over the created virtual network to avoid any artifacts due to the underlay network.

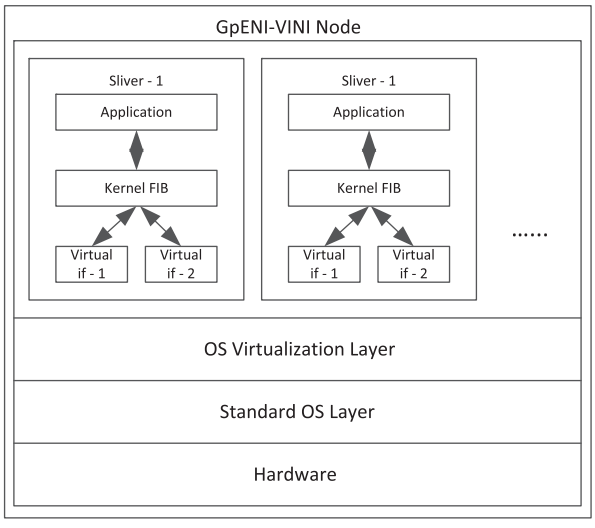


Fig. 6. Trellis design architecture (adapted from [39]).

5.2.4. Packet flow in virtual link

Once a virtual link is created, it is helpful to see the packet flow in it. In Fig. 7, the packets flow on a virtual link between slivers inside a slice with an EGRE (Ethernet over GRE) tunneling mechanism. First, the data packet comes out of the virtual interface that is an Ethernet frame in the context of a virtual host. This becomes the payload in the context of the root. At the tunnel interface (root context), this payload is encapsulated with a GRE header [51] and a four byte-key to demultiplex the packet to the correct tunnel interface. Then the IPv4 delivery header is added. The reverse process is carried out at the other end. First, the IPv4 header is removed, and then the GRE header is checked to determine the correct tunnel interface. Finally, the payload (Ethernet Frame) is delivered to the correct virtual interface of the sliver.

5.3. Flexible resource provisioning by GpENI-VINI

The major components of the core architecture of the GpENI-VINI inherits the resource management framework and the technologies applied to the network virtualization. We have made three specific contributions towards exploring a user-friendly interface that supports a more flexible resource provisioning manner: (1) Successfully deploy XORP 1.7 on GpENI-VINI slivers, and script the XORP installation process into one program; (2) Extend IIAS features on both server and client sides, so that there are less restrictions on virtual links when creating virtual topology. On the client side, XORP configuration files customized for the virtual topology are automatically generated. (3) Automate the routing software installation and startup process. In this section, we present technical details on how these have been accomplished.

5.3.1. Issues and challenges

By investigating resource provisioning with the VINI Veritas testbed [29,37], we noticed a few issues and challenges.

- Routing tools such as XORP were not supported in the current VINI testbed. Having such additional tools would allow researchers to choose from multiple programmable routing systems such as Quagga and XORP in their experimentation.
- With the auto virtual topology creation, originally the IIAS tools limited the link creation to be between physically adjacent nodes as a design choice. Regular users are not provided the permission to manually create arbitrary virtual topology in a slice. While this made sense for the VINI Veritas testbed, we wanted to extend the functionality of the IIAS tools to allow regular users to create arbitrary virtual topologies with the GpENI-VINI testbed.
- With the VINI Veritas testbed, since the auto virtual topology creation for the regular users' framework is based on the default physical topology between VINI nodes, there was no graphical user interface (GUI) to

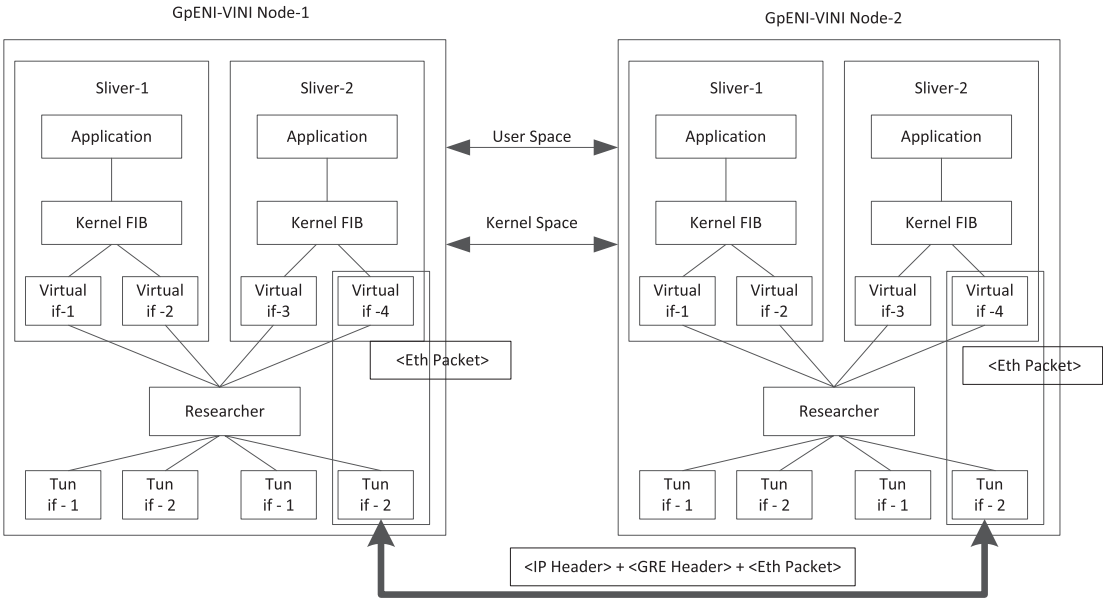


Fig. 7. Virtual link between slivers.

create a virtual network inside a slice. This limited researchers from a graphical view of the virtual network and they were required to use manual configuration through a file.

To provide GpENI-VINI with additional services and make it user friendly to create a virtual network, our goal was to extend the functionality of the IIAS tools and design a prototype for the GUI. Thus, we faced the following challenges:

1. How to integrate a XORP routing application support with routing protocol (OSPF) in GpENI-VINI nodes (this is discussed in Section 5.3.2).
2. How to make a virtual network inside a slice to be a routable network allowing an arbitrary virtual topology, with the XORP routing application running (this is discussed in Section 5.3.3).
3. How to run/start a routing application inside all slivers of a slice simultaneously (this is discussed in Section 5.3.4).

5.3.2. XORP integration with GpENI-VINI

XORP [8] supports IPv4 and IPv6 routing protocols such as RIP, OSPF, BGP, and PIM-SM. It supports unicast routing policies and SNMP. The architecture of XORP consists of two subsystems. The first one is at a higher-level (“user-space”) that consists of the routing protocols and management mechanisms. The second one is at a lower-level (“kernel”) that provides a forwarding path and API for the higher-level to access.

XORP uses a multi-process architecture at the user-level with one process per routing protocol. It also uses XORP Resource Locators (XRLs) that are novel inter-process communication mechanisms to communicate with other processes. The lower-level subsystem can use traditional UNIX or Linux forwarding, the Click modular router [3], or Windows kernel forwarding. Currently, we use Linux kernel forwarding in the GpENI-VINI testbed. There are several different important XORP processes. For more details, refer to the XORP documentation [8].

Before incorporating XORP on the GpENI-VINI testbed, we first tested it on one of our internal lab demo platforms. Initially, we downloaded XORP version 1.6 from [8]. We configured and tested XORP for its routing functionality by using the OSPF routing protocol. We created two virtual interfaces on the Ethernet interface and configured those with OSPF and then tested them. After compiling XORP 1.6, we found that this occupied around 2.2 GB of disk space, where the XORP directory contains configuration, makefiles, different routing protocol daemons, documents, and other related files.

In order to integrate XORP in the GpENI-VINI testbed, we needed to create a slice with a virtual topology, and XORP 1.6 in every sliver in a slice by building and installation. We created a three-sliver virtual topology in a test slice. While XORP 1.6 requires approximately 2.2 GB for each node, it turned out to be a time consuming process taking approximately 30 min to configure and build XORP for all three slivers. Thus, we realized that XORP 1.6 was not a scalable option for the end users due to such a large setup time.

XORP’s new version 1.7 was made available as XORP.CT-1.7 under Candela Technologies [30] through a GNU General Public License based on the official version of the XORP 1.7 SVN repository. We decided to try XORP implementation with this new version on our test slice. By using the new version, the biggest advantage that we observed was that it created an executable binary file in compressed form after building XORP. This was advantageous to us to build the XORP’s latest version on the GpENI-VINI Server and install this in our GpENI-VINI local

software repository so that it can be distributed as an executable binary file to the nodes whenever a researcher requests it for an experiment. This was found to be useful for GpENI-VINI nodes as this significantly saves configuration time and memory space.

Implementing XORP has created its own challenges. This required immense understanding of the underlying architecture of both XORP and Trellis. Issues and experience working with XORP for GpENI have been described in [48].

5.3.3. Extending IIAS features

Before getting into details of extended functionalities of IIAS tools on the GpENI-VINI testbed, we start with several features of the original IIAS tools.

IIAS tools are based on VINI-Veritas [37] and were written in Python. On the server side, two software components, which are `topology` and `create-topo-attributes`, are referred to as the *Topology Resource Specifications* (“topo_rspec”) generator. The `topology` module contains the list of physical links between adjacent sites in the GpENI-VINI testbed. These adjacencies must be manually added by the GpENI-VINI administrators, a laborious process. Running as a cron job every 15 min, the `create-topo-attributes` takes the list of physical links information in the module `topology` as input to generate virtual topology links in the slice, if the slice tag “topo_links” is set as “iias”. In other words, the functions of `create-topo-attributes` will create *topology resource specifications* (“topo_rspec”), hosts, and virtual topology links. The topology resource specifications (“topo_rspec”) originally represented only unidirectional links. The IIAS tools created “topo_rspec” in the following formats: “node-id”, “IP address”, “Link rate”, “my virtual tip IP”, “remote virtual tip IP”, and “virtual network.”

On the client side, the IIAS tools contain two Python modules: `optin` and `topo`; these together are called the topology manager. They can be accessed in the root context of the GpENI-VINI node. This is done by extending the original node manager from the PlanetLab with plug-ins. Optin generates open VPN configuration files that support the injection of external traffic into the virtual network topology. The node manager polls the latest sliver information every 15 min from the GpENI-VINI server and passes a copy of this to topo. The topo program interprets the “topo_rspec” attribute values and performs the following basic functions: (1) Creates virtual and tunnel interface names based on the EGRE key and node ids, (2) Creates new and clears old virtual interfaces based on topo rspecs, (3) Creates new and clears old tunnel interfaces based on topo rspecs, (4) Creates and deletes NAT interfaces in both the root context and the sliver context, (5) Sets up and tears down EGRE links and NAT.

We have extended the features of IIAS from both the server and client slide, which is presented as follows:

- Extended Features of IIAS Tools on Server Side: Based on the original IIAS tools, we have extended its features to support a virtual topology creation through a GUI.
  - Web-based GUI Design: We have created a web-based GUI to create and view virtual topology in a user’s slice. With the GUI, a list of nodes within the slice will be shown, together with their node ids. Meanwhile, we can view the current virtual topology link information in the form of $[(a,b),(b,c),\ldots,(m,n)]$, where the letters represent the node ids of nodes in the slice. To create the topology, the user can just enter the topo links information in the same format.
  - To support the GUI functionality, we have modified the IIAS tools on the server side. When the user has confirmed the topology creation, the topo link information will be written into `gui_topo` module together with the relevant slice’s name. Then the extended IIAS software component `gui-create-topo-attributes` will import information of both the slice name and topo links from `gui_topo` and generate *topology resources specification* (“topo_rspec”). It will create a “topo_links” tag value, which represents all the links in a virtual topology network. For example, if node A has id 1 and node B has id 2, “topo_links” consists of the value $[(1,2),(2,1)]$, where each one represents a unidirectional link.
- Extended Features of IIAS Tools on Server Side: On the client side, we would like to achieve two goals with extended IIAS features.
  - Integrating XORP to IIAS Tools on Client Side: We added the XORP supporting functionality to create the XORP configuration based on sliver interfaces and to write this configuration in the corresponding sliver file system. The integration component is useful to researchers to run routing applications with a XORP configuration file on each sliver of a slice in the GpENI-VINI testbed. This removes the burden from researchers to write the configuration information for each and every interface and protocol. For example, if a researcher has an N nodes fully connected network in his/her slice for a project, she would need to write N configuration files with each $N-1$ interfaces, which can be time consuming. Our integration tool has automated this phase.
  - Integrating automation with routing softwares: We have also extended the IIAS functionality to automate the routing process in virtual network topologies on GpENI-VINI resources. When the client side IIAS program is running, it will create the XORP and Quagga installation and startup programs, and write them into the sliver file system. This is helpful to researchers by making it easy to install and start XORP or Quagga for their experimentation by running our automation toolkit. We will explain the automation process in the subsequential section.

5.3.4. Routing software auto-initialization in GpENI

To automate the routing process, we also added the following features to the IIAS tools: (1) With our customized IIAS tools at the client side, a XORP installation and Quagga installation program were written into the sliver’s file system. (2) we created the program `automation` that takes user choices, such as credentials, slice name, routing

software, version, and protocol. This triggers routing daemons in all the slivers of a slice at a time with the help of `codeploy` [4].

To make the routing processes automated, we have created XORP and Quagga local repository files at the GpENI-VINI server. Hence, these installation programs point the GpENI-VINI server to get XORP and Quagga software. The `experiment` code is now made available from the GpENI-VINI server [13] so that researchers can readily download and use it for their experimental work. The callout box in Fig. 5 shows what the extended IIAS tools and the automation tool of the GpENI-VINI node contains.

5.4. Measurements and validation

We evaluated the robustness of the customized IIAS programs for XORP integration on the client side, by reporting the measurement time taken using two XORP programs. These two programs generate a XORP configuration file for the OSPF routing protocol and a XORP installation script for the automation process.

To study the robustness of the client-side IIAS feature, in terms of the XORP configuration and installation script generation time, we varied the number of virtual links to an individual sliver from 1 to 21, and measured the XORP integration time on three slivers that reside on different physical nodes. In this study, we aimed to find out the dominant factor to the generating time, which could be either the number of virtual links or the number of slices.

- Case I – Single Slice Case: We measured the XORP configuration file generating time on three different slivers from KSU, UMKC, and ETH Zürich (ETHZ). On each sliver, we varied the number of virtual links from 1 to 21. Fig. 8 depicts the average generating time. We observed that the time increased linearly with the increment of the number of the virtual links to the sliver. On the other hand, the physical machine's hardware configuration was another factor to the overall performance.
- Case II – Multiple Slice Case: In this case, we chose one physical GpENI-VINI node from KSU, and measured the XORP configuration file generating time on one sliver when its physical host has 1, 3, 5, and 11 slivers. Fig. 9 shows that the number of slices has less influence on the XORP configuration file generating time than the number of virtual links.

We also created a routing study automation program to make it easy to start a routing application simultaneously without logging onto each sliver. The researcher can use this program from their system or laptop to start the routing application on each sliver of the slice with their SSH key. To measure the total running time of this application, we took some sample slices with a different number of nodes.

Table 2 shows the average time and the standard deviation to start the XORP routing application on each slice (measured over five instances). Note that the variation can be attributed to network conditions such as link speeds and the physical location of the virtual nodes (slivers) from the GpENI-VINI server as the nodes need to download the XORP software from the GpENI-VINI server.

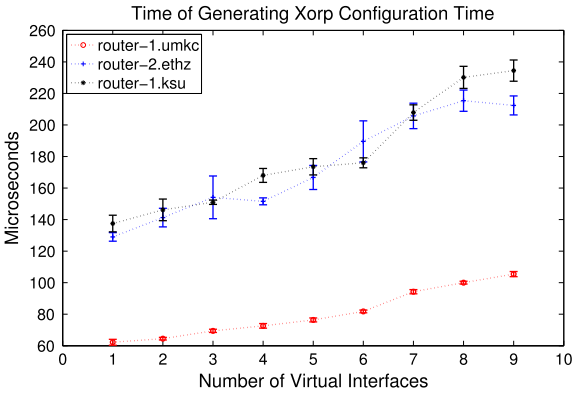


Fig. 8. XORP configuration file generating time with only one slice.

Our sample nodes were located in physically diverse regions. We took four nodes from the Midwest region of the USA and two additional nodes from the European region. Results up to four nodes were based on the fours nodes in the Midwest region; with these nodes the average automation time was less because of physical proximity. Results beyond four nodes include European nodes. With these added, the average automation time increases, which is affected due to the physical distance between the regions.

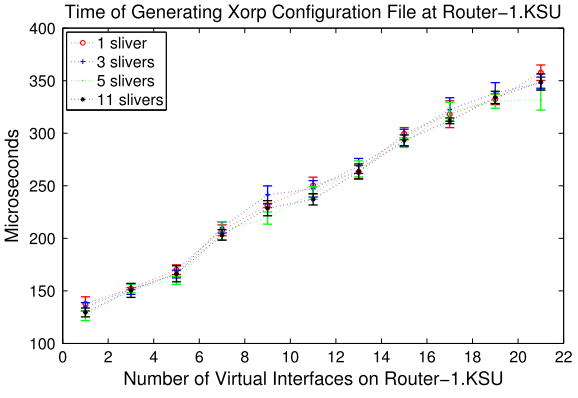


Fig. 9. XORP configuration file generating time with multiple slices.

Table 2
Run time of routing automation.

# Nodes	Average (s)	StdDev
3	204.6	22.075
4	468.0	37.543
5	890.0	39.592
6	1034.2	69.941

6. Dynamic circuit creation in the regional network testbed

The GpENI testbed also supports dynamic circuit creation at the optical layer. To enable dynamic circuit network (DCN) in a regional network testbed, we need to make necessary changes to the current infrastructure. In this section, we first introduce some background knowledge on the DCN, then discuss how to establish DCN across the GpENI testbed and how to establish DCN across two testbed domains.

6.1. Background

We first give a brief description on the background knowledge on Dynamic Circuit Network's relevant technologies before presenting our effort on deploying DCN on GpENI testbed.

6.1.1. Dynamic circuits

Dynamic Circuit Network (DCN)/Interoperable On-demand Network (ION) [17] is a networking service in Internet2 that provides researchers the ability to create short-term circuits of large bandwidth across the network. These circuits are created for bandwidth-intensive applications that are run over the Internet2 backbone network. This service uses both the software components of the OSCARS [7] and DRAGON [5] projects to create dynamic circuits across various domains and across various network technologies. The circuits are created and deleted using the Web User Interface provided by the OSCARS software components. The Inter Domain Controller (IDC) is basically the entity managing the circuit creation and deletion along with user authentication and authorization mechanisms in an Autonomous System (AS) or local domain. Internet2 uses the ION service to transfer large scientific data for projects such as Large Hadron Collider (LHC) and Compact Muon Solenoid (CMS).

6.1.2. DRAGON

Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) [5] was a NSF funded project to dynamically provision network resources across various domains and across heterogeneous networking technologies. GMPLS [38] is the key protocol used to create circuits spanning across both optical and Ethernet domains and hence, DRAGON creates a Layer 1 virtual circuit. A set of software components has been developed to leverage this capability across a testbed over the Washington, D.C. area. The major components of DRAGON software are VLSR (Virtual Label Switched Router), NARB (Network Aware Resource Broker), ASTB (Application Specific Topology Builder), and RCE (Resource Computation Engine). As DRAGON provides the capability to create circuits that span across various domains the NARB acts as the entity that represents a local domain or Autonomous System (AS). In each domain each switch needs to be configured separately for creating a circuit and hence, VLSRs act as the entity controlling the switches. The RCE and ASTB are used for computing the resources required for creating circuits. Hence a particular DRAGON domain will have a NARB and one or more VLSR, depending upon the number of switches in the domain.

6.1.3. OSCARS

On-Demand Secure Service and Advance Reservation System (OSCARS) is a networking service deployed in the DoE ESnet to create dynamic, deterministic, and secure circuits across the ESnet network. MPLS [55] and RSVP [43] are the key protocols used to create advance reservations of bandwidth using the software components developed as part of the OSCARS project. The Label Switched Path (LSP) s are created using MPLS both in Layer-2 and Layer-3 using OSCARS software. The circuits are created and deleted using a web interface provided by OSCARS and hence, this method is adopted in the DCN/ION project as the interface for managing virtual circuits. The major software components of the OSCARS are Reservation Manager (RM), Path Setup Subsystem (PSS) and Bandwidth Scheduler Subsystem (BSS), Authentication, Authorization, and Audition Subsystem (AAA). The RM, PSS, BSS are used for reserving resources and creation and deletion of actual circuits in the network, and AAA is used to provide authentication mechanisms using X.509 certificates.

6.1.4. VLAN

Virtual LAN is a networking technology used to provide secure and reliable transport between hosts that are not physically connected to each other. IEEE 802.1Q is the most commonly used standard for VLANs and it has been implemented in most of the commercially available switches. VLAN tag is a 32-bit field added to the Ethernet frame, which has a 12-bit field called VLAN ID specifying the VLAN number of the packet transmitted over the network. The VLAN number is the entity which differentiates packets of different virtual circuits over a network. There are two approaches to assign VLAN membership, static VLANs and dynamic VLANs. An Ethernet packet carrying a (non-default) VLAN tag is said to be a "tagged" packet and the one carrying a default VLAN tag is said to be an "untagged" packet. The static vlans are created by assigning ports to VLAN and dynamic vlans are created using software such as CiscoWorks 2000 in Cisco managed switches, and it can also be created using SNMP. VLANs are mostly used by corporate networks to separate traffic of various applications that share the same network infrastructure. The VLAN tags are added when the packet enters the corporate network and they are removed when they leave the network.

6.1.5. Q-in-Q

Q-in-Q or Double tagging is a method to add one more outer VLAN tag to already tagged packets. This is used by Internet Service Providers to separate network traffic between different user groups so that one user group will be isolated from another group. However, each member in a group can have their packets tagged differently so that they can protect their packets from other members of the same group. IEEE 802.1ad is the standard specifying this method of double tagging the packets sent over the network. Similar to VLAN tagging, the outer tags are added once the packets with an inner tag enters the corporate

network and the outer tag alone is removed when the packets leave the network.

6.2. DCN in GpENI

In the following section, we first explain the current network infrastructure of the GpENI and the changes needed in the current infrastructure to establish DCN across GpENI.

6.2.1. Current GpENI network configuration

GpENI's basic connectivity (see Fig. 10) is designed as a single Ethernet broadcast domain capable of transporting arbitrary VLANs. All the four GpENI universities (UNL, KSU, KU, and UMKC) are connected to their own interface at the GPN Cisco 6509 Ethernet switch (GPN Switch) in Kansas City PoP and all these interfaces are configured to the same VLAN number 125. UNL has a direct fiber connection of capacity 1 GigE to the GPN switch transported through Ekinops DWDM equipment. UMKC connects to the GPN switch using L2TP tunneling through MOREnet infrastructure. KSU and KU form a single MPLS domain in the KanREN network infrastructure and are connected to the GPN switch through VPLS. The CoreDirector CI switch, which connects to Internet2 at the Kansas City PoP is also connected to the GPN switch using a 10 GigE link. Each university has a Netgear GSM7224 switch and a node cluster connected to the switch. We have modified the DCN software suite to support the Netgear GSM7224 switch. Hence, we can create dynamic circuits between the universities with DCN software running over these switches if the infrastructure in Kansas City PoP supports it. The limitation is, that as these Netgear switches do not have the per-VLAN bandwidth policing feature, they do not have the capability to create circuits of a specific bandwidth as requested by the user.

6.2.2. Option 1: GpENI network connectivity with DCN (using GPN switch)

The GPN switch is a production switch that carries traffic between the four GpENI universities and is also not supported by the current DCN/ION software suite. This option (see Fig. 11) analyzes the possibility of using the GPN switch for creating DCN circuits between GpENI universities. There are two ways in which the GPN switch can be configured to enable DCN circuits between GpENI universities:

- The first way is to configure static VLANs over the GPN switch so that DCN circuits can be created between universities only with the pre-configured VLAN tags. Hence this involves creating a table of VLAN tags for all possible sources and destinations of DCN circuits between GpENI universities and configuring them appropriately in the GPN switch. In the case of the IDC and two VLSRs, one VLSR is for controlling the Ciena CoreDirector CI switch in UNL and the other one is for controlling all the Netgear GSM7224 switches in all the GpENI universities. The IDC and the two VLSRs will be located in the UNL. Hence we will be able to create, delete and modify dynamic circuits over the web interface provided by the DCN/ION software suite between these universities.
- The second way is to configure a Q-in-Q cloud in the GPN 6509 switch with VLAN 125 so that it acts as a pass through for packets of any VLAN tag generated by any of the GpENI universities. Hence, in this case also, the IDC and two VLSRs will be placed at UNL. The only difference is that we will be able to create circuits of arbitrary VLAN tags between the GpENI universities. Though the advantage of this method over the previous one is the freedom of choice of VLAN tags, the drawback is that, because of the Q-in-Q cloud, the packet transmitted by any university will be broadcasted to all four GpENI universities.

6.2.3. Option 2: GpENI network connectivity with DCN (using GpENI switch)

This option (see Fig. 12) requires acquiring a new Ethernet switch (GpENI switch), which is already supported by the DCN/ION software suite and in replacing the existing GPN switch with the GpENI switch. Hence this option requires all four GpENI universities to have a Layer-2 connectivity to the GpENI switch and one interface of the GpENI switch will be connected to the GPN switch. For this option, the dedicated switch in Kansas City could also be a switch placed by the ProtoGENI group at this location. The IDC of the DCN/ION can be placed at UNL or Kansas City PoP and two VLSRs, one for controlling the CoreDirector CI and another for controlling the Netgear switches, can be placed in UNL or Kansas City PoP. Hence, we will be able to create dynamic circuits of desired bandwidth of arbitrary VLAN tags between any of the GpENI universities with this network infrastructure. The Ciena CoreDirector in Kansas City is shown as connected to the GpENI switch because this will be the Connector for GpENI universities to connect to the Internet2 infrastructure.

6.2.4. Decision

We have chosen option 1 mentioned in Section 6.2.2 after considering the cost factors involved in Option 2.

6.3. DCN between GpENI and MAX

We explored different options to connect the GpENI network with MAX across the Internet2 backbone at Layer-2.

6.3.1. Option 1: GpENI network connectivity with MAX (using GPN switch)

This option (see Fig. 13) discusses using the existing GPN switch for creating the DCN network within GpENI. It further discusses the two ways, static VLAN and Q-in-Q, to achieve this goal.

- In the first method, static VLANs are configured in the GPN switch so that DCN circuits can be created between universities with predefined VLAN tags. Hence this involves creating a table of VLAN tags for all possible sources and destinations of DCN circuits between GpENI universities and configuring them appropriately at the GPN switch. Hence, in this method, to connect to MAX we need to configure one more VLAN tag for each university or configure a VLAN tag so that all uni-

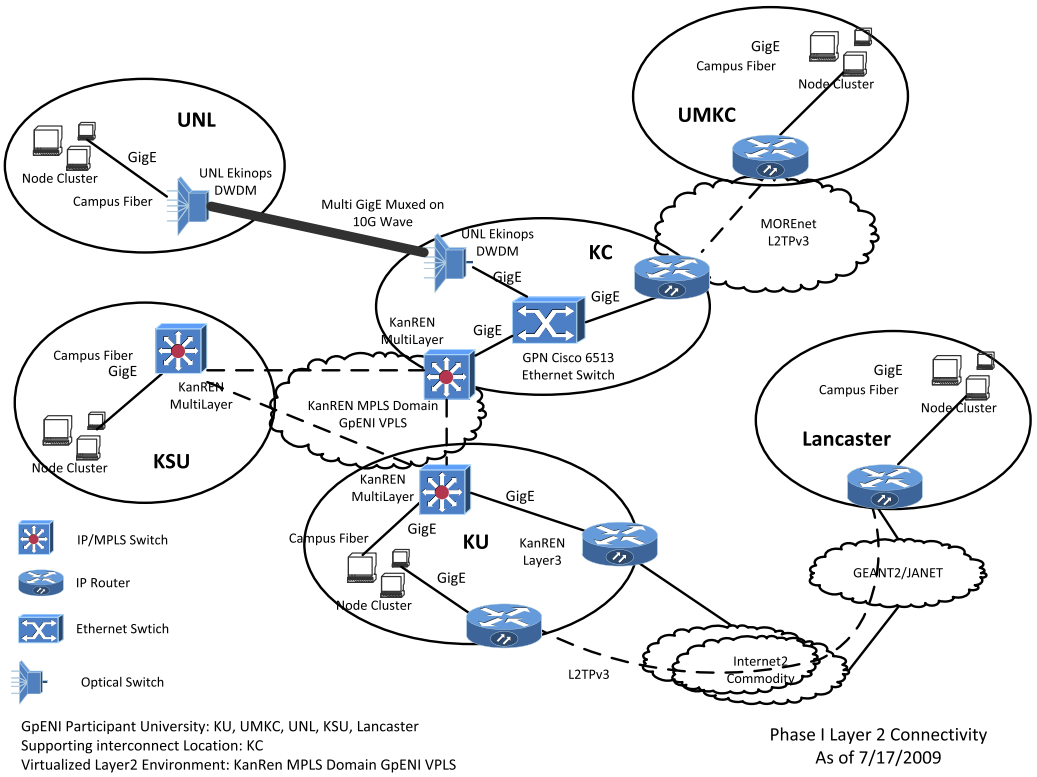


Fig. 10. GpENI current network connectivity. *Source:* https://wiki.ittc.ku.edu/gpeni/Image:GpENI-L2.png.

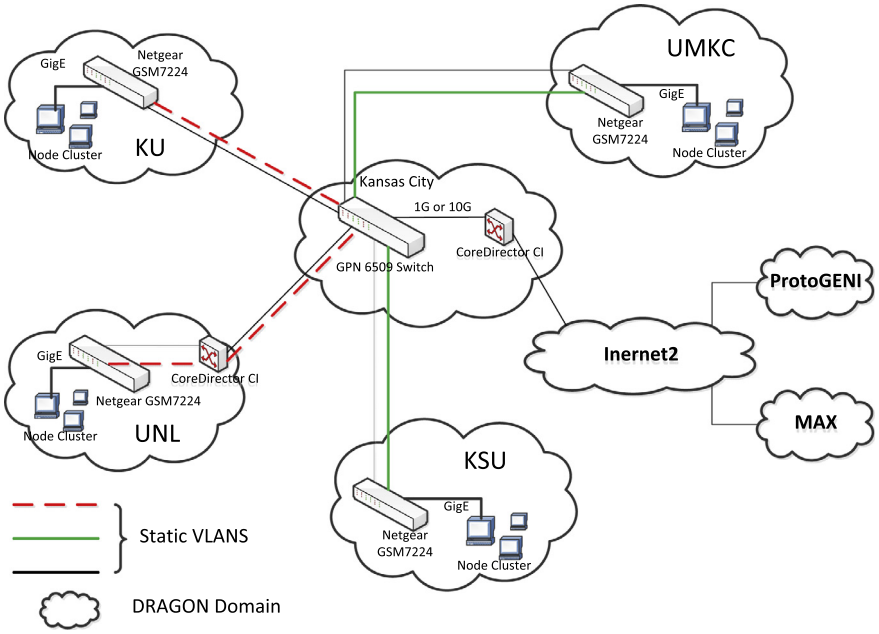


Fig. 11. Option 1: DCN in GpENI (using GPN switch).

versities use DCN to MAX only with this VLAN tag. This is primarily a choice between individual circuits from each university to MAX or to have a single broadcast domain to MAX. These VLAN tags need to be configured to the interface of GPN in which the CoreDirector of Kansas City PoP is connected to the GPN switch. In this method, we need to have an IDC controlling the creation of circuits with the predefined tags from each

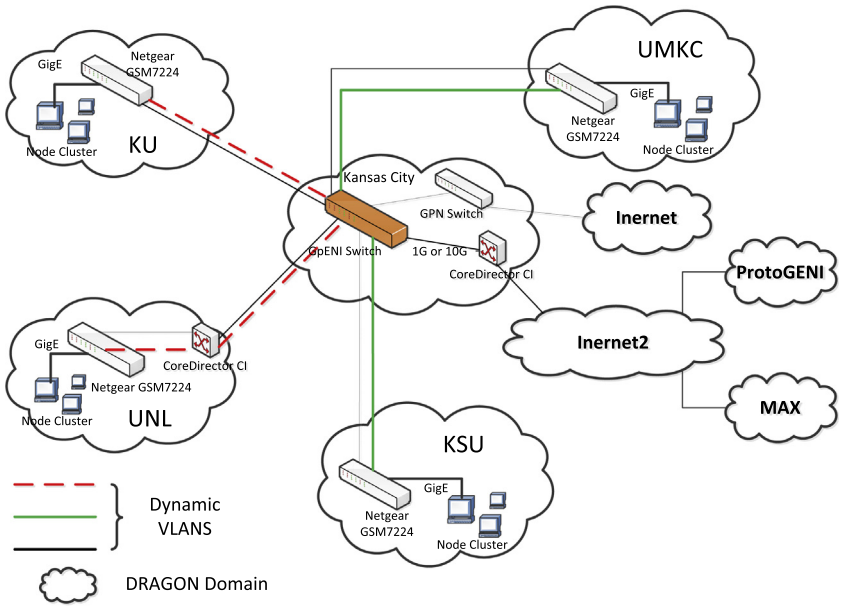


Fig. 12. Option 2: DCN in GpENI (using GpENI switch).

university. Thus, the IDC could be placed in UNL and we need a VLSR for controlling all the Netgear switches of all the GpENI universities and a VLSR for controlling the CoreDirector. We can configure one of the PCs to act as a web server and configure the OSCARS software in it so that each university can create and delete circuits using the web interface of OSCARS software.

- In the second method, Q-in-Q is configured in the GPN switch so that it acts as a pass through for packets of any VLAN tag generated by any of the GpENI universities. Hence to connect to MAX in this method we need to just include the interface of the GPN switch, which is connected to the CoreDirector of the Kansas City PoP in the Q-in-Q cloud. In this manner, we can create circuits of any VLAN tag from any of the GpENI universities to the MAX. Also, the IDC placement and VLSR placement are similar to the method above but the only difference is that users can create circuits with an arbitrary VLAN tag to MAX.

In both of the above methods we can also have a separate VLSR in each GpENI university creating individual

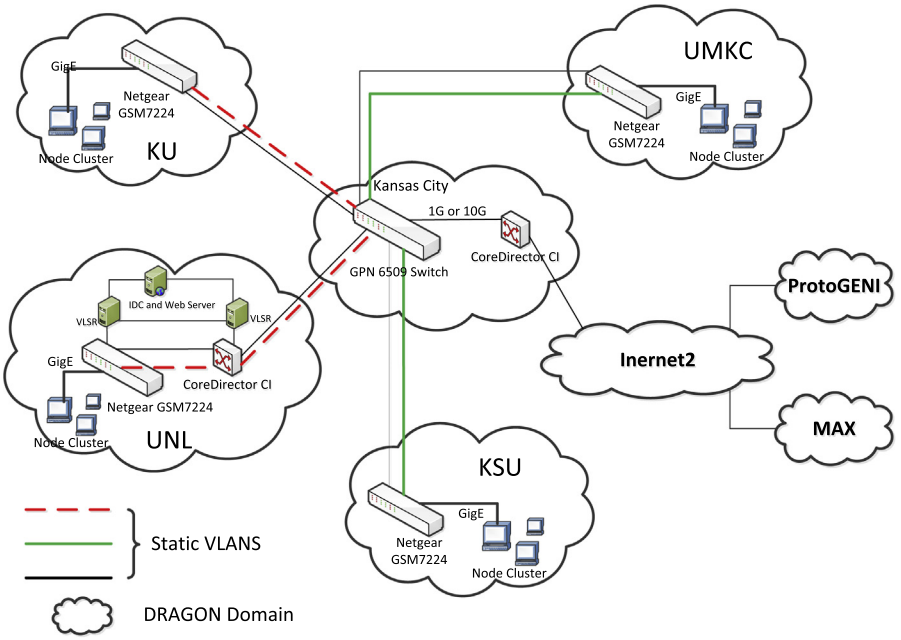


Fig. 13. Connection to MAX using GPN Switch and Internet2.

DRAGON domains in each GpENI university. Instead, if we have a VLSR for controlling all the Netgear Switches then we have only one DRAGON domain representing all the GpENI universities.

6.3.2. Option 2: GpENI network connectivity with ProtoGENI (using GPN switch)

ProtoGENI has a 10 Gbps backbone in the Internet2 network and it has already deployed its nodes (which includes HP 5400 switches, NetFPGA cards, and 2 PCs) at three Internet2 sites (Kansas City, Salt Lake City and Washington, D.C.). Currently, in the HP Procurve switch deployed at Kansas City, there are no free 10 GigE ports available. Hence, in this option (see Fig. 14), we created a 1 GigE connection between the GPN switch at Kansas City and the HP procurve switch in the ProtoGENI node and used this connection to connect to MAX.

Currently, in the GpENI network, all the universities deliver untagged packets to the GPN switch. Since ProtoGENI requires the packets to be tagged with a specific VLAN, the Q-in-Q cloud needs to be setup in the GPN switch and each university is required to transmit packets to the GPN switch with a predefined outer VLAN tag that is agreed with ProtoGENI. Q-in-Q would be used on the GPN switch to alleviate the need for the VLAN number coordination within GPN. The component manager of the ProtoGENI needs to be setup so that we could request dynamic circuits from GpENI to any node of ProtoGENI. In this case, we will be using the client software of ProtoGENI instead of DCN to create dynamic virtual circuits. However the circuits can only be created from and to the ProtoGENI nodes and hence, the traffic inside GpENI will remain as a broadcast domain.

6.3.3. Option 3: GpENI network connectivity with MAX (using GpENI switch)

This option (see Fig. 15) discusses acquiring a new Ethernet switch (GpENI switch), which is already supported by the DCN/ION software suite and replacing the existing GPN switch with the GpENI switch. The MAX network can be connected in this method by just connecting the CoreDirector of Kansas City PoP to the GpENI switch. We can have a dedicated IDC controlling the GpENI switch and 1 VLSR controlling all the Netgear switches and a VLSR controlling the CoreDirector at the UNL campus. In this case, only the dynamic circuit from or to the UNL need to be in the order of 50 Mbps as CoreDirector is in the path. Otherwise, dynamic circuits between other universities could be of any bandwidth capacity supported by the interface of the GpENI switch. Having the CoreDirector participate in the dynamic circuit is just a choice, and hence, if we want to create dynamic circuits of any bandwidth to the UNL we could remove the VLSR controlling the CoreDirector switch and make it as a pass through switch, simply passing the traffic to another end, irrespective of the packets VLAN tag. In this method, the IDC needs to be located in the Kansas City PoP and the VLSRs could be in each university or we could have one VLSR controlling all the Netgear Switches located in the UNL. The GpENI switch can be connected to Internet2 with the DCN or it can also be connected to the ProtoGENI backbone if we have a connection between the GpENI switch and the HP Procurve switch of the ProtoGENI node.

6.3.4. Decision

We have chosen both options 1 and 2, and so far, we adopted the second method of option 1 configured in the Kansas City PoP. We have shown demos of options 1 & 2 at GENI conferences, and we plan to pursue option 3 in the future.

7. GpENI federation deployment

The GpENI testbed has achieved federation on three sub-aggregates: the MyPLC sub-aggregate, the GpENI-VINI sub-aggregate, and the DCN sub-aggregate.

Currently, the federation on the MyPLC sub-aggregate and the GpENI-VINI sub-aggregate are running the PlanetLab implementation of the slice facility architecture (SFA). There are three interfaces: registry, slice manager (SM), and aggregate manager (AM). These GENI interfaces are accessible via the slice facility interface (SFI) implementing functions to get slice details, node details, and user accounts.

- MyPLC federation: MyPLC has *federated* with the public PlanetLab, which means that GpENI resources are available for authorized PlanetLab researchers using the public PlanetLab interface. The SFA deployment on MyPLC follows the tutorial available on the GENI website.
- GpENI-VINI federation: GpENI-VINI has *federated* with both public PlanetLab and public VINI, and it needed additional configurations on the SFA at the server side to provision network resources like virtual links information, besides the regular procedures. According to our survey, we are the first to configure SFA on MyVINI to support federation. Therefore, we communicated with Princeton researchers who have done federation on the public VINI for technical support and debug issues. In general, there are two steps to configure SFA on MyVINI:
  - Create a copy of the VINI schema at the GpENI-VINI server. This VINI schema is the same as the copy from the public VINI server, and it is an XML-based file that is a resource specification for the VINI-based testbed.
  - Edit two items in the SFA configuration: (1) *SFA_AGGREGATE_TYPE* and (2) *SFA_AGGREGATE_RSPEC_SCHEMA*. The first item is set as *vini* to add GpENI-VINI as an aggregate in the SFA. The second item is to set the file path of the VINI schema at the GpENI-VINI server.

A Layer-2 federation deployment was done with the DCN software suite. Currently, the GpENI has been able to implement dynamic circuits with the MAX testbed to establish an inter-domain DCN. Section 6.3 provides detail descriptions on the inter-domain DCN between the GpENI and MAX.

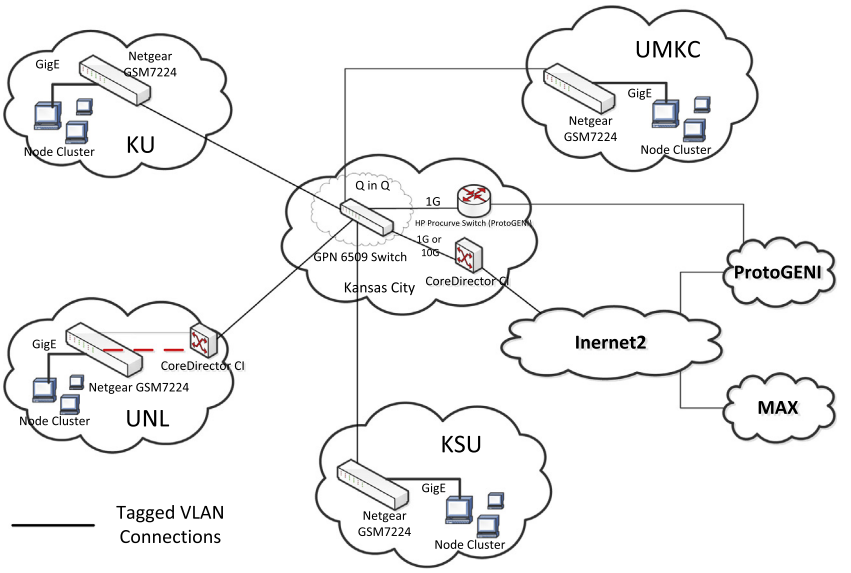


Fig. 14. Connection to MAX using GPN switch,ProtoGENI and Internet2.

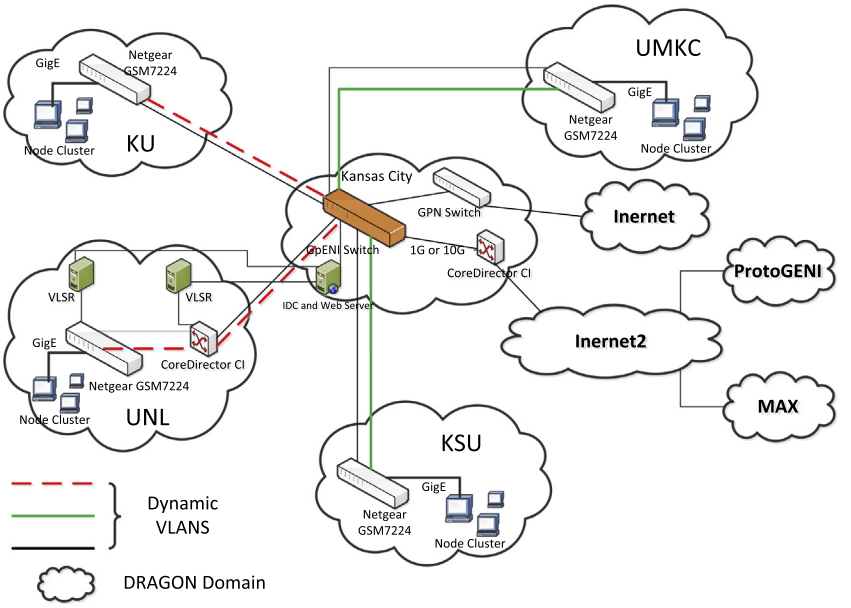


Fig. 15. Connection to MAX Using GpENI Switch.

For our long-term goals, there are plans for federations among GENI, as well as for other Future Internet testbeds such as OneLab [24].

8. Experimentations on GpENI Testbed

The GpENI infrastructure [68] is in the process of expanding to 38 to 40 clusters with 200 nodes worldwide, federated with the larger GENI PlanetLab control framework and interconnected to several ProtoGENI facilities. This enables users to perform resilience and survivability experiments at scale, both in terms of node count and with the geographic scope needed to emulate area-based challenges such as large-scale disasters.

8.1. Resilience research with GpENI testbed

In our own research efforts, we are using these facilities to enable experiments that cross-verify the analytical and simulation-based resilience research currently underway at The University of Kansas [66]. It is leveraging topology and challenge generation tools (KU-LoCGen [67] and KU-CSM [45]) developed for this purpose, with emphasis on

resilience metrics [57] and multi-path, multi-realm diverse transport (ResTP) [64,63] developed as part of our NSF FIND research in the PostModern Internet Architecture project [40].

Resilient topologies generated by KU-LoCGen and analyzed by KU-CSM are used to generate Layer-2 topologies that configure the topology of GpENI experiments. We evaluated performance when slice topologies are challenged by correlated failures of nodes and links, measuring connectivity, packet delivery ratio, goodput, and delay, when subject to CBR, bulk data transfer, and transactional (HTTP) traffic [60,61]. Large scale resilience experiments are run over interconnected aggregates using the DCN (within the GpENI) and OpenFlow configured paths, with VINI/PlanetLab Layer-3 topologies, to emulate both the existing ISP and synthetic topologies. Over these topologies, we ran our multipath-aware transport protocol ResTP to evaluate its performance under varying application and traffic loads. Based on the output of our challenge generation simulations, we selectively disabled node slivers and links to emulate correlated network failures and attacks. In the future, we plan to use the wireless emulator under the ProtoGENI framework to emulate jamming attacks to wireless access networks. Each challenge set is classified as a single scenario and each scenario is run multiple times to establish reasonable confidence in the results.

Another project on resilience research with autonomic management is described later in Section 8.4.

8.2. Graph algorithm evaluation on GpENI

We develop a heuristic algorithm that improves the connectivity of a graph in terms of the algebraic connectivity metric by adding links [35]. Algebraic connectivity is defined as the second smallest eigenvalue of the Laplacian matrix and it is widely used for topological optimization. A secondary objective of our algorithm is to select the links that improve the algebraic connectivity of the graph in the least costly fashion in which we capture the cost of network as the total link length. The heuristic to increase algebraic connectivity in a graph is based on adding links to the nodes that have the fewest incident links (i.e., minimal degree nodes).

Large scale resilience experiments are run over interconnected PlanetLab clusters using tinc VPN tunneling software [32]. The tinc project allows creation of arbitrary topologies while preventing broadcast storms. We create sample topologies consisting of five GpENI PlanetLab nodes (i.e., KSU, KU, Cambridge, KIT, Bern) as shown in Fig. 16. The sample binary-tree topology as shown in Fig. 16(a) has the root node in Cambridge. The KU node is the highest-degree node in the partial-mesh topology shown in Fig. 16(b).

We measure the network performance in terms of flow robustness, which quantifies resilience as the fraction of node pairs that remain connected in a network after it has been subjected to a number of node failures. Simultaneous ping traffic between every pair of node in each topology is generated. We pause tinc processes to emulate challenges against critical nodes in each scenario topology. Flow robustness is measured on the sample topologies with and without our optimization algorithm being applied as shown in Fig. 17.

We plot the flow robustness of the binary-tree scenario as shown in Fig. 17(a). The scenario represents an attack against the highest betweenness node (Cambridge) in this tree topology as shown in Fig. 16(a). The optimized topology performs better since additional link (between KSU and Bern) provide alternate path between node pairs. Flow robustness of the partial-mesh scenario is shown in Fig. 17(b). In this scenario the highest degree node (KU) is attacked in a partial-mesh topology as shown in Fig. 16(b). The optimized topology (with additional link between KSU and KIT) has a flow robustness of 0.6, where as non-optimized topology has a flow robustness of 0.3. The flow robustness of non-optimized partial-mesh topology is better than the non-optimized binary-tree topology when critical nodes are attacked because nodes are more connected in the partial-mesh topology. This resilience experiment demonstrates creation of arbitrary topologies and application of our heuristic algorithm on the GpENI testbed.

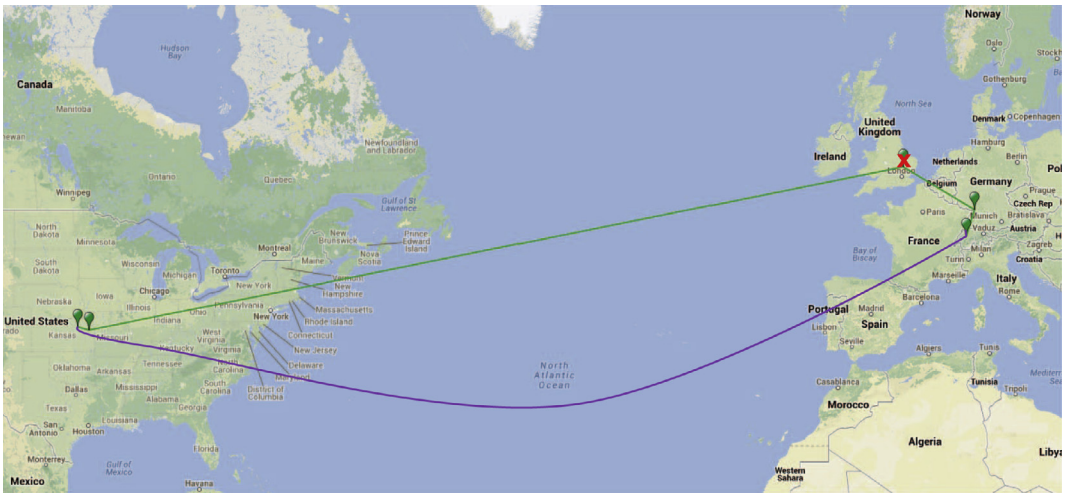
8.3. Protocol emulation on GpENI

We developed the ANTP (airborne network and transport protocols) suite that operates in this highly-dynamic environment while utilizing cross-layer optimizations between the physical, MAC, network, and transport layers [62]. We showed how each component in the ANTP suite outperforms the traditional TCP/IP and MANET protocols through simulation using ns-3 [46]. Having verified these protocols through simulation and analysis, the next step towards deployment of the ANTP suite is developing a cross-platform implementation of the protocols. Moreover, we emulated the ANTP suite implementation on GpENI PlanetLab nodes. Mobility of the nodes was emulated using the GPS emulator that uses a mobility model such as 3D Gauss-Markov, random waypoint, or random direction to generate the location and velocity of a given node [44]. The visualization system is added to ease the development and debugging phase of the implementation as well as to provide logging data for performance analysis [34]. It is implemented as a web-based interface with integration of the Google Maps API to show the nodal locations and velocity in real time [33]. Emulating the ANTP suite on the distributed GpENI PlanetLab nodes eases the development of the implementation code.

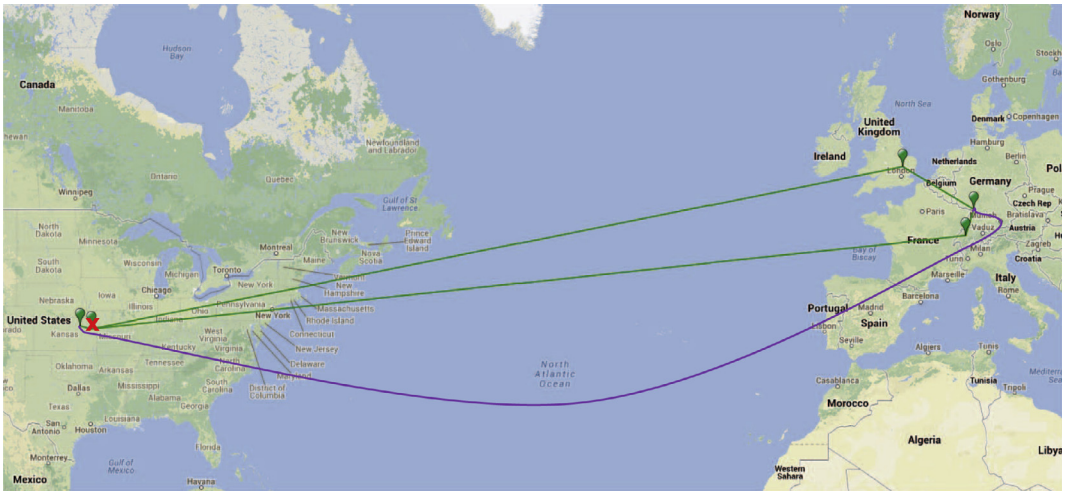
8.4. Autonomic management experiments on GpENI-VINI

The GpENI-VINI allows researchers to design experiments at the network layer. First, researchers can create multiple kinds of virtual topologies in their slices. Second, researchers can install either Quagga or XORP to the slivers to make them routers. Third, network events such as link failures or node failures could be triggered on the GpENI-VINI testbed. In other words, researchers have full control on the virtual routers in their slices to conduct research experiments.

Our recent work regarding dynamic network reconfiguration with autonomic management [56] was an experimental study conducted on the GpENI-VINI testbed. We



(a) Binary-tree sample topology



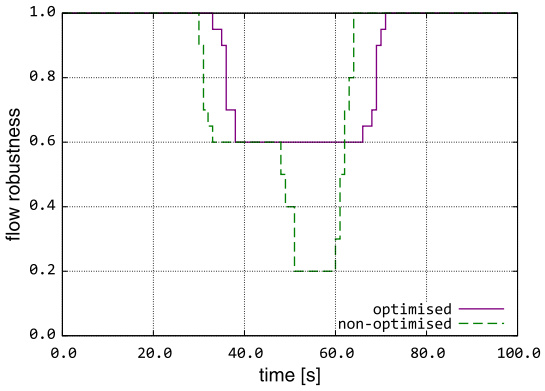
(b) Partial-mesh sample topology

Fig. 16. Visualisation of experimental scenario topologies.

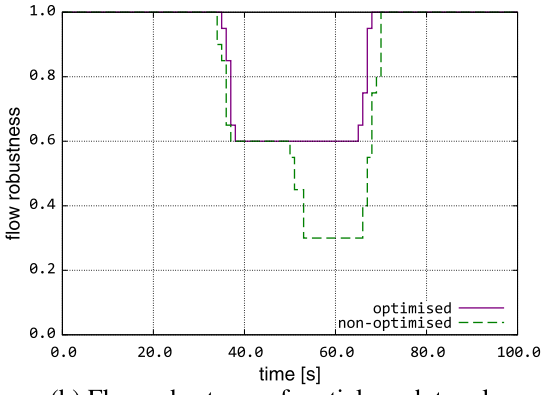
designed the experiments in a virtualized network environment, triggered router failures in the virtual topology and recovered the failure by replacing the failed router with a standby router. Since each virtual router ran XORP, we were able to collect the OSPF routing tables from each virtual router in the topology and relevant timestamps to evaluate the routing convergence time for all of the virtual networks. In turn, we evaluated the performance of the autonomic management method on the network reconfiguration. On the other hand, GpENI-VINI is a testbed with nodes distributed in different countries that provides a real-network infrastructure for researchers to analyze network performances, instead of running simulations on a simulator.

8.5. Demonstration of transferring CMS data with DCN

We showed a demo of transferring Compact Muon Solenoid (CMS) data using a dynamic circuit established from UNL to MAX at the GLOBECOM 2010 conference. In the UNL, we established a Layer-2 connection between the Prairiefire super computer and the Netgear switch [36]. The GPN switch in the Kansas City PoP was connected to the Internet2's Juniper switch via a static VLAN. We established a dynamic circuit of this VLAN from the Internet2 switch to a PlanetLab node in the MAX domain using MAX IDC. We used UNL's IDC to create the circuit of this VLAN id from Netgear to the GPN switch which completes the experimental setup for transferring CMS data to the PlanetLab node via these dynamic circuits. These configurations enabled the Prariefire node to have a Layer-2 connection to the PlanetLab node in the MAX and we transferred the CMS data over this circuit. A researcher in the MAX domain or anywhere in the Internet2's site, can use this service to perform experiments over the Prarifire node cluster and transfer the results to his place.



(a) Flow robustness of binary-tree topology



(b) Flow robustness of partial-mesh topology

Fig. 17. Performance of optimized and non-optimized topologies.

9. GpENI extension: KanREN-GENI deployment plans and topology

KanREN-GENI (Fig. 18) is a GENI mesoscale OpenFlow deployment underway in KanREN (the Kansas Research and Education Network) as well as selected deployment into GPN (the Great Plains Network). This deployment heavily leverages on the existing GpENI infrastructure. We are deploying Brocade OpenFlow-enabled switches co-located with the production KanREN switches that will provide full opt-in for any users accessing KanREN infrastructure at its PoPs (Kansas City, Lawrence, Manhattan, Ft. Hays, Wichita, Emporia, Pittsburg, Overland Park, and Internet2). Furthermore, OpenFlow switches are being deployed at selected GPN (Great Plains Network) institutions (such as UMKC).

The phase 1 deployment includes switches in the KU GpENI cluster and at the KanREN KU and Internet2 PoPs. The GpENI OpenFlow switch is located between the GpENI node cluster and KanREN backbone interconnection link. All other KanREN OpenFlow switches are collocated and directly connected via 1 Gb/s fiber to production KanREN Brocade switches. This permits arbitrary flow manipulation through KanREN and the KU GpENI cluster. The first OESS OpenFlow controller is being deployed within the GpENI node cluster. The plan for Phase 2 is to deploy additional switches throughout KanREN, a K-12 institution, and selected GPN GpENI institutions. We will additionally seek to interoperate the OpenFlow and PlanetLab/VINI subaggregates and integrate with other GpENI institutions that have OpenFlow capabilities.

10. Summary

The GpENI testbed is an international Future Internet research testbed centered in the Midwest region of the United States and in Europe. We are now making an effort to expand to Asian countries as well. The main goal of the GpENI testbed is to provide all-layer programmability in the network. In particular, as a part of the GENI control framework cluster B, GpENI runs a private instance of PlanetLab for application-layer and transport-layer programmability, and runs a customized private instance of VINI for network-layer programmability. Moreover, establishing the DCN across the GpENI testbed enables Layer-2 programmability.

In this paper, we presented an overall description on the network infrastructure and node cluster of the GpENI testbed and the status of federation deployment on the GpENI. We also discussed our recent effort on the Layer-3 (GpENI-VINI) and Layer-2 (DCN) programmability on the GpENI testbed. With the GpENI-VINI, researchers are allowed to create arbitrary virtual topologies in their slices and start the routing automation process to deploy the routing software (i.e., Quagga or XORP) to all the slivers in the virtual topology and run routing experiments. On the other hand, the creation of a dynamic circuit network enables researchers to transfer large scientific data for short durations of time reliably and quickly, without going through the current best effort traffic nature of the Internet. Enabling a regional network for dynamic circuit network needs changes to the configurations in the production switches used to connect participating institutions as well as design the control plane and data plane for their network domain. We also discussed the experimentations we have done so far to explain a variety of research directions that the GpENI testbed can support.

There are a number of lessons learned building the GpENI testbed. First, dividing the key responsibilities among the initial partners along clear boundaries was helpful in executing the project; this way, UNL took the lead on DCN, UMKC on network layer programmability, and KSU on PlanetLab functionality at the application layer, with KU heading the overall coordination. Second, building such a wide-area testbed required tremendous knowledge and expertise below Layer-3 on physical or tunnel connectivity—this was possible because of dedicated support from the campus IT staff at each institution, KanREN, and MORENet. Third, taking software built by others (such as VINI and PlanetLab) and customizing for our purpose on GpENI turned out to be a non-trivial exercise. As an example, as we worked on GpENI network programmability, we were able to identify a number of issues using VINI in the GpENI environment that required a significant amount of troubleshooting and fixes. Last but not the least, as a whole, we gained knowledge and understanding about many issues that came up with the testbed deployment at

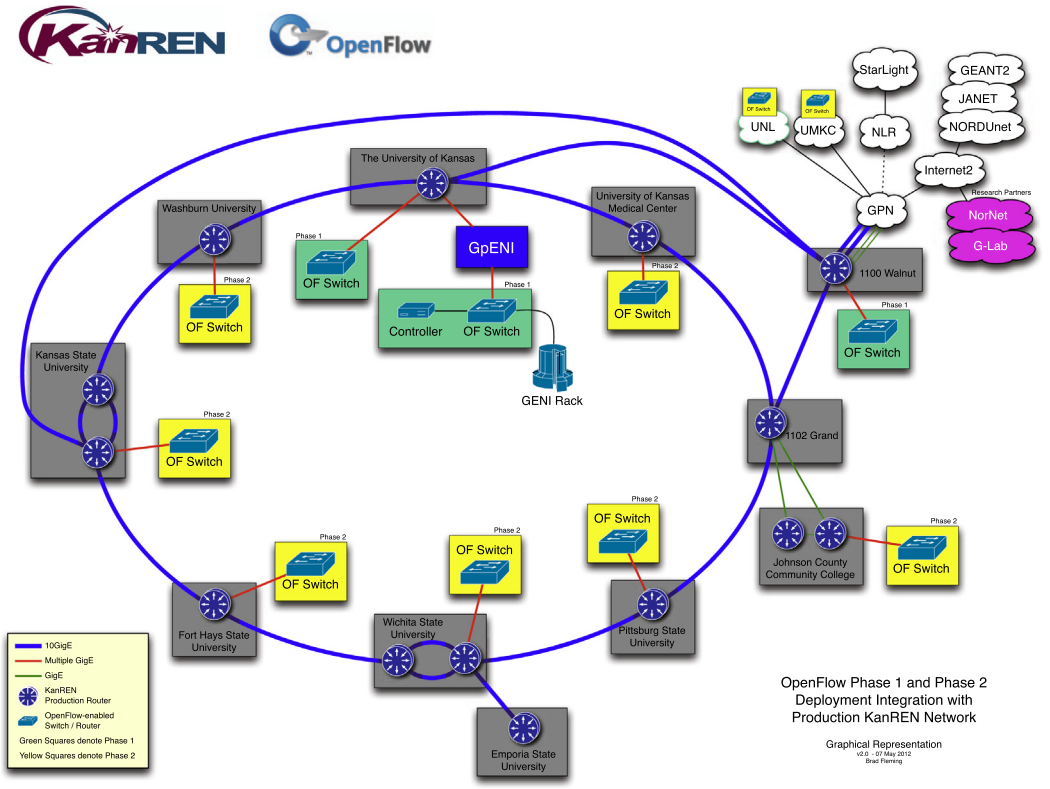


Fig. 18. KanREN-GENI.

a much deeper level that would not have been possible to know otherwise.

Acknowledgments

This work is funded in part by the US National Science Foundation GENI program (GPO Contract No. 9500009441), by the EU FP7 FIRE programme ResumeNet project (Grant Agreement No. 224619), and in large part, by the participating institutions. GpENI is made possible by the involvement of many people in the participating institutions. We particularly acknowledge the following people: Tricha Anjali, Torsten Braun, Richard Becker, Baek-Young Choi, Kent G. Christensen, Riddhiman Das, Joseph Evans, Robert Fines, Dale M. Finkelson, Brad Fleming, Don Gruenbacher, Ajita Gupta, Sam Hays, Mary Lou Hines Fritts, David Hutchison, Michael Hulet, Parikshit Juluri, Can Kanli, George Koffler, Sean Korzdorfer, Yunzhao Li, Lin Liu, Wesley Mason, Rick McMullen, Greg Monaco, Andrew Moore, Bernard Plattner, Adam Pullen, Haiyang Qian, A. Scott, James Schonemann II, John Sherrell, Mukesh Subedee, Ali Sydney, Tim Sylvester, Nidhi Tare, David Wolfinbarger, and Dongsheng Zhang.

Appendix A. Glossary

IIAS: Internet In A Slice. It is a tool kit to facilitate users to create virtual resources on GpENI-VINI testbed.

L2TP: Layer 2 Tunneling Protocol.
MyPLC: A portable PlanetLab central (PLC) software to make private PlaneLabs [53].
MyVINI: A private VINI implementation within the GpENI nodes, permitting full control of virtual topology.
Node: Any dedicated physical system that runs PlanetLab and VINI components in the GpENI testbed.
SFA: Slide Facility Architecture.
Site: Any geographical location (ex: a University or an Organization) where GpENI nodes are located.
Slice: It is a group of resources (nodes) allocated from distributed nodes across the GpENI testbed to a project. Each slice has a finite lifetime and must be renewed before it expires.
Sliver: Sliver is a slice running on a specific node. It is a virtual host on a node that is participating in the slice. A sliver (virtual host) is created with the slice name on participating nodes.
Trellis: A software system that combines host and network virtualization technologies. For host virtualization, Trellis uses a container based virtualization technology called the Linux VServer and NetNS to support virtual nodes, virtual interfaces, and virtual links inside a node.
Quagga: It is another open source routing software.
XORP (eXtensible Open Router Platform): It is an open source routing software.

References

[1] Active Networks Program. <http://www.darpa.mil/sto/strategic/an.html>.
[2] Cacti. <http://www.cacti.net/>.
[3] The Click Modular Router Project. <http://www.read.cs.ucla.edu/click/click>.
[4] CoDeploy. <http://codeen.cs.princeton.edu/codeploy/>.
[5] DRAGON – Dynamic Resource Allocation via GMPLS Optical Networks. <http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/Network>.
[6] Emulab: Network Emulation Testbed. <http://www.emulab.net/>.
[7] ESnet's On-Demand Secure Circuits and Advance Reservation System (OSCARS). <http://www.es.net/services/virtual-circuits-oscars/>.
[8] eXtensible Open Router Platform (XORP). <http://www.xorp.org/>.
[9] FIRE: Future Internet Research Experiment. <http://cordis.europa.eu/fp7/ict/fire/>.
[10] Firestarter. <http://www.fs-security.com/>.
[11] FP6 Situated Autonomic Communications. <http://cordis.europa.eu/fp7/ict/fire/future-internet-projects_en.html>.
[12] GENI: Global Environment for Network Innovations. <http://www.geni.net/>.
[13] GpENI-VINI. <http://geni-myvini.umkc.gpeni.net/>.
[14] GpENI wiki <http://www.gpeni.net/>.
[15] IIAS: Internet In A Slice. <http://svn.planet-lab.org/wiki/ViniInternetInASlice>.
[16] Internet2 DCN/ION Software Suite. <https://wiki.internet2.edu/confluence/display/DCNSS/DRAGON+Supported+Switches>.
[17] Internet2 ION. <http://www.internet2.edu/ion>.
[18] JGN2plus Testbed. <http://www.jgn.nict.go.jp>.
[19] Linux-VServer. <http://linux-vserver.org/Welcome_to_Linux-VServer.org>.
[20] Magic Gigabit Testbed. <http://www.magic.net/>.
[21] Nagios. <http://www.nagios.org/>.
[22] NetNS: Network NameSpace. <https://lists.linux-foundation.org/pipermail/containers/2007-September/007290.html>.
[23] NSF NeTS FIND Initiative. <http://www.nets-find.net>.
[24] OneLab: Future Internet test beds. <http://www.onelab.eu/>.
[25] OpenFlow Switch Consortium. <http://www.openflowswitch.org/>.
[26] PlanetLab. <http://www.planet-lab.org/>.
[27] Quagga Routing Suite Software. <http://www.nongnu.org/quagga/>.
[28] Supercharged Planetlab Platform (SPP) Hardware Components. <http://wiki.arl.wustl.edu/index.php/SPP_Hardware_Components>.
[29] VINI: Virtual Network Infrastructure. <http://vini-veritas.net/>.
[30] XORP.CT Branch. <http://www.candelatech.com/xorp.ct/>.
[31] Zenoss. <http://www.zenoss.com/>.
[32] Tinc wiki, 2010. <http://www.tinc-vpn.org/>.
[33] ANTP Visualizer, January 2011. <http://experiment-1.ku.gpeni.net/antp/aerorp/common/www/map.php>.
[34] M.J. Alenazi, E.K. Çetinkaya, J.P. Rohrer, J.P.G. Sterbenz. Implementation of the AeroRP and AeroNP protocols in python, in: Proceedings of the International Telemetering Conference (ITC), San Diego, CA, October 2012.
[35] M.J.F. Alenazi, E.K. Çetinkaya, J.P.G. Sterbenz. Network design and optimisation based on cost and algebraic connectivity, in: Proceedings of the 5th IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM), Almaty, September 2013.
[36] P. Angu, B. Ramamurthy, Experiences with dynamic circuit creation in a regional network testbed, in: High Speed Networking (HSN) Workshop, IEEE INFOCOM, 2011.
[37] A. Bavier, N. Feamster, M. Huang, L. Peterson, J. Rexford. In VINI Veritas: realistic and controlled network experimentation, in: SIGCOMM Comput. Commun. Rev., vol. 36, no. 4, 2006, pp. 3–14.
[38] L. Berger (Ed.), Generalized Multi-Protocol Label Switching (GMPLS): Signaling Functional Description, Internet RFC 3471, January 2003. <http://www.ietf.org/rfc/rfc3471.txt>.
[39] S. Bhatia, M. Motiwala, W. Mühlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, J. Rexford, Trellis: a platform for building flexible, fast virtual networks on commodity hardware, in: Proc. of 2008 ACM CoNEXT Conference, Madrid, Spain, 2008, pp. 72:1–72:6.
[40] B. Bhattacharjee, K. Calvert, J. Griffioen, N. Spring, J.P.G. Sterbenz, Postmodern Internetwork Architecture, Technical Report ITTC-FY2006-TR-45030-01, The University of Kansas, Lawrence, KS, February 2006.
[41] M. Bermann, J.S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Sesakar, A federated testbed for innovative network experiments. Comput. Netw. 61 (2014) 5–23.
[42] B. Braden, L. Ricciulli, A plan for a scalable abone – a modest proposal, in: Technical Report, USC – Information Science Institute, 1999.
[43] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, Internet RFC 2205, September 1997. <http://www.ietf.org/rfc/rfc2205.txt>.
[44] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wireless Commun. Mobile Comput. 2 (5) (2002) 483–502.
[45] E.K. Çetinkaya, D. Broyles, A. Dandekar, S. Srinivasan, J.P.G. Sterbenz, Modelling communication network challenges for future Internet resilience, survivability, and disruption tolerance: a simulation-based approach, Telecommun. Syst. 52 (2013) 751–766.
[46] E.K. Çetinkaya, J.P. Rohrer, A. Jabbar, M.J. Alenazi, D. Zhang, D.S. Broyles, K.S. Pathapati, H. Narra, K. Peters, S.A. Gogi, J.P.G. Sterbenz, Protocols for highly-dynamic airborne networks, in: Proceedings of the 18th ACM Annual International Conference on Mobile Computing and Networking (MobiCom), Istanbul, August 2012, pp. 411–413 (Extended Abstract).
[47] T. Chart, S. Schmid, M. Sifalakis, A.C. Scott, Active routers in action: evaluation of the LARA++, active router architecture in a real-life network, Lect. Notes Comput. Sci. 2982 (2004) 215–227.
[48] R. Cherukuri, X. Liu, A. Bavier, J.P.G. Sterbenz, D. Medhi, Network virtualization in GpENI: framework, implementation and integration experience, in: Proc. of 3rd IEEE/IFIP International Workshop on Management of the Future Internet (ManFI'2011), Dublin, Ireland, May 2011, pp. 1212–1219.
[49] CNRI. The Gigabit Testbed Initiative, 1996. <http://www.cnri.reston.va.us/gigafr/Gigabit_Final_Rpt.pdf>.
[50] DARPA Active Networks Security Working Group. Security Architecture for Active Nets, July 1998. <http://srg.cs.uiuc.edu/Security/seraphim/May2000/SecurityArchitecture.pdf>.
[51] D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, Generic Routing Encapsulation (GRE). Internet RFC 2784, March 2000. <http://www.ietf.org/rfc/rfc2784.txt>.
[52] A. Galis, B. Plattner, J.M. Smith, S.G. Denazis, E. Moeller, H. Guo, C. Klein, J. Serrat, J. Laarhuis, G.T. Karetsos, C. Todd, A flexible ip active networks architecture, in: Proceedings of the Second International Working Conference on Active Networks, IWAN '00, Springer-Verlag, London, UK, 2000, pp. 1–15.
[53] M. Huang, T. Parmentelat. MyPLC User's Guide. <https://svn.planet-lab.org/wiki/MyPLCUserGuide>.
[54] A. Jackson, J.P.G. Sterbenz, M. Condell, R. Hain, Active network monitoring and control: the SENCOMM architecture and implementation, in: DARPA Active NEtworks Conference and Exposition, 2002.
[55] K. Kompella, Y. Rekhter. Signalling Unnumbered Links in Resource ReSerVation Protocol – Traffic Engineering (RSVP-TE). Internet RFC 3477, January 2003. <http://www.ietf.org/rfc/rfc3477.txt>.
[56] X. Liu, P. Juluri, D. Medhi, An experimental study on dynamic network reconfiguration in a virtualized network environment using autonomic management, in: Proc. of IFIP/IEEE International Symposium On Integrated Network Management (IM'2013): Mini-Conference, Ghent, Belgium, May 2013, pp. 616–622.
[57] A.J. Mohammad, D. Hutchison, J.P.G. Sterbenz, Towards quantifying metrics for resilient and survivable networks, in: Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP), November 2006, pp. 17–18.
[58] C. Partridge, B. Davie, R. Campbell, C. Catlett, D. Clark, D. Feldmeier, R. McFarland, P. Messina, I. Richer, J. Smith, J.P.G. Sterbenz, J. Turner, D. Tennenhouse, J. Touch, Report of the ARPA/NSF Workshop on Research in Gigabit Networking, 1994. <http://www.isi.edu/touch/pubs/arpansf94.pdf>.
[59] L. Peterson, S. Sevinc, J. Lepreau, R. Ricci, J. Wroclawsk, S.S.T. Faber, Slice-Based Facility Architecture, 2007. <http://www.cs.princeton.edu/llp/arch_abridged.pdf>.
[60] J.P. Rohrer, E.K. Çetinkaya, J.P.G. Sterbenz, Progress and challenges in large-scale future internet experimentation using the GpENI programmable testbed, in: The 6th ACM International Conference on Future Internet Technologies (CFI), Seoul, June 2011, pp. 46–49.
[61] J.P. Rohrer, E.K. Çetinkaya, J.P.G. Sterbenz, Resilience experiments in the GpENI programmable future Internet testbed, in: Proceedings of the 11th Würzburg Workshop on IP: Joint ITG and Euro-NF

Workshop "Visions of Future Generation Networks" (EuroView2011), August 2011, pp. 29–30.
[62] J.P. Rohrer, A. Jabbar, E.K. Çetinkaya, E. Perrins, J.P.G. Sterbenz, Highly-dynamic cross-layered aeronautical network architecture, IEEE Trans. Aerospace Electron. Syst. 47 (4) (2011) 2742–2765.
[63] J.P. Rohrer, A. Jabbar, J.P.G. Sterbenz, Path diversification: a multipath resilience mechanism, in: Proceedings of the IEEE 7th International Workshop on the Design of Reliable Communication Networks (DRCN), Washington, DC, October 2009, pp. 343–351.
[64] J.P. Rohrer, R. Naidu, J.P.G. Sterbenz, Multipath at the transport layer: an end-to-end resilience mechanism, in: Proceedings of the IEEE/IFIP International Workshop on Reliable Networks Design and Modeling (RNDM), St. Petersburg, Russia, October 2009, pp. 1–7.
[65] N. Shalaby, L. Peterson, A. Bavier, Y. Gottlieb, S. Karlin, A. Nakao, X. Qie, T. Spalink, M. Wawrzoniak, Extensible routers for active networks, in: DARPA Active NEtworks Conference and Exposition, 2002.
[66] J.P.G. Sterbenz, E.K. Çetinkaya, M.A. Hameed, A. Jabbar, J.P. Rohrer, Modelling and analysis of network resilience, in: Proceedings of the Third IEEE International Conference on Communication Systems and Networks (COMSNETS), Bangalore, January 2011, pp. 1–10.
[67] J.P.G. Sterbenz, E.K. Çetinkaya, M.A. Hameed, A. Jabbar, Q. Shi, J.P. Rohrer, Evaluation of network resilience, survivability, and disruption tolerance: Analysis, topology generation, simulation, and experimentation, Telecommun. Syst. 52 (2013) 705–736.
[68] J.P.G. Sterbenz, D. Medhi, B. Ramamurthy, C. Scoglio, D. Hutchison, B. Plattner, T. Anjali, A. Scott, C. Buffington, G. Monaco, D. Gruenbacher, R. McMullen, J. Rohrer, J. Sherrell, P. Angu, R. Cherukuri, H. Qian, N. Tare, The great plains environment for network innovation (GpENI): a programmable testbed for future Internet architecture research, in: Proc. of 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom), Berlin, Germany, May 2010, pp. 428–441.

Deep Medhi is a Curators' Professor in the Department of Computer Science & Electrical Engineering at the University of Missouri–Kansas City, USA, and a honorary professor in the Department of Computer Science & Engineering at the Indian Institute of Technology–Guwahati, India. He received B.Sc. in Mathematics from Cotton College, Gauhati University, India, M.Sc. in Mathematics from the University of Delhi, India, and his Ph.D. in Computer Sciences from the University of Wisconsin–Madison, USA. Prior to joining UMKC in 1989, he was a member of the technical staff at AT&T Bell Laboratories. He served as an invited visiting professor at the Technical University of Denmark, a visiting research fellow at Lund Institute of Technology, Sweden, and State University of Campinas, Brazil. As a Fulbright Senior Specialist, he was a visitor at Bilkent University, Turkey, and Kurukshetra University, India. He is the Editor-in-Chief of Springer's *Journal of Network and Systems Management*, and is on the editorial board of *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Network and Service Management*, and *IEEE Communications Surveys & Tutorials*. He has published over 125 papers, and is co-author of the books, *Routing, Flow, and Capacity Design in Communication and Computer Networks* (2004) and *Network Routing: Algorithms, Protocols, and Architectures* (2007), both published by Morgan Kaufmann Publishers, an imprint of Elsevier Science. His research interests are multi-layer networking, network virtualization, data center optimization, and network routing, design, and survivability. His research has been funded by NSF, DARPA, and industries.

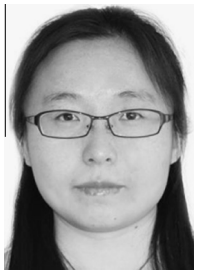
Byrav Ramamurthy is currently a Professor and Graduate Chair in the Department of Computer Science and Engineering at the University of Nebraska–Lincoln (UNL). He has held visiting positions at the Indian Institute of Technology–Madras (IITM), in Chennai, India and at the AT&T Labs-Research, New Jersey, U.S.A. He is author of the book *Design of Optical WDM Networks – LAN, MAN and WAN Architectures* and a co-author of the book *Secure Group Communications over Data Networks*, published by Springer in 2000 and 2004, respectively. He has authored over 125 peer-reviewed journal and conference publications. He serves as Editor-in-Chief for the Springer Photonic Network Communications journal. He was Chair of the IEEE ComSoc Optical Networking Technical Committee (ONTC) during 2009–2011. Dr. Ramamurthy served as the TPC Co-Chair for the IEEE INFOCOM 2011 conference to be held in Shanghai, China. He is a recipient of the College of Engineering Faculty Research Award for 2000 and the UNL CSE Dept. Student Choice Outstanding Teaching Award for Graduate-level Courses for 2002–2003 and 2006–2007. He has graduated 10 Ph.D. and 40 M.S. students under his research supervision. His research has been supported by the U.S. National Science Foundation (NSF), the U.S. Department of Energy (DOE), the U.S. Department of Agriculture (USDA), National Aeronautics and Space Administration (NASA), AT&T Corp., Agilent Tech., HP, OPNET Inc. and the University of Nebraska–Lincoln (UNL).

Caterina M. Scoglio is Professor of Electrical and Computer Engineering at Kansas State University. Her main research interests include modeling, analysis, and design of networked systems, with applications to epidemic spreading and power grids. Caterina received the Dr. Eng. degree from the "Sapienza" Rome University, Italy, in 1987. Before joining Kansas State University, she worked at the Fondazione Ugo Bordoni from 1987 to 2000, and at the Georgia Institute of Technology from 2000 to 2005.

Justin P. Rohrer is currently a Research Associate of Computer Science at the Naval Postgraduate School (NPS) and an Adjunct Assistant Professor of Electrical Engineering and Computer Science at the KU Information & Telecommunication Technology Center (ITTC). He received his Ph.D. in Electrical Engineering from the University of Kansas in 2011 with honors. He received his B.S. degree in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2004. From 1999 to 2004, he was with the Adirondack Area Network, Castleton, NY as a network engineer. He was also an ITTC Graduate Fellow from 2004 to 2006. He received the best paper award at the International Telemetering Conference in 2008 and the best graduate student paper award at the same conference in 2011. His research focus is on resilient and survivable transport and routing protocols. Interests also include highly-dynamic mobile networks, and simulating network disruptions. Previous research has included weather disruption-tolerant mesh networks and free-space optical metropolitan networks. He is a member of the IEEE Communications and Computer Societies, ACM SIGCOMM, Eta Kappa Nu, and was an officer of the Kansas City section of the IEEE Computer Society for several years.

Egemen K. Çetinkaya is Assistant Professor of Electrical and Computer Engineering at Missouri University of Science and Technology (formerly known as University of Missouri–Rolla). He received the B.S. degree in Electronics Engineering from Uludağ University (Bursa, Turkey) in 1999, the M.S. degree in Electrical Engineering from University of Missouri–Rolla in 2001, and Ph.D. degree in Electrical Engineering from the University of Kansas in 2013. He held various positions at Sprint as a support, system, and design engineer from 2001 until 2008. He is a graduate research assistant in the ResiliNets research group at the KU Information & Telecommunication Technology Center (ITTC). His research interests are in resilient networks. He is a member of the IEEE Communications Society, ACM SIGCOMM, and Sigma Xi.

Ramkumar Cherukuri is a software engineer at CGI. He designs software for living and specializes in the field of Network Systems. His Research interests include Network Routing, Protocol Development, Software Design and Cloud Computing. His outside interests include attending professional meetups, brainstorming on new ideas with peers and visiting foreign places. He obtained his M.S. in Computer Science from the University of Missouri–Kansas City and his B.E. in Electronics and Communication Engineering from Andhra University College of Engineering, Visakhapatnam, India.



Xuan Liu is a Ph.D. student at the University of Missouri–Kansas City. She received B.S. in Communication Engineering from China University of Geosciences (CUG) in June 2007 and M.S. in Computer Science from the University of Missouri–Kansas City in December 2010. Her research interests include network virtualization, information centric networking, computer networking modeling and optimization.

Pragatheeswaran Angu is currently an R&D Software Developer at Epic. He received his M.S. degree in Computer Science from the University of Nebraska–Lincoln in May 2011. His research interests include optical networking, scheduling and optimization.

Andy Bavier is a Research Scholar at Princeton University. He has been building research testbeds since 2002. He is a designer and core developer of the PlanetLab, VINI, and VICCI testbeds among others. He also actively participates in the NSF GENI project and serves on the GENI Architects board.

Cort Buffington is the Executive Director of KanREN, Inc., The high-speed research and education network in Kansas. Cort joined the KanREN team in 1999 and had served the organization in several different technical capacities before accepting the directorship in 2008. Cort was the principle architect and engineer of the current and previous generations of the KanREN network, and still takes an active role in engineering/architecture along with the administrative aspects of the organization. Cort is an active participant in the state, regional and national R&E networking community, participating actively in Internet2, The Great Plains Network, The Quilt, and Kan-ed.

James P.G. Sterbenz is Associate Professor of Electrical Engineering & Computer Science and on staff at the Information & Telecommunication Technology Center at The University of Kansas, and is a Visiting Professor of Computing in InfoLab 21 at Lancaster University in the UK. He received a doctorate in computer science from Washington University in St. Louis in 1991, with undergraduate degrees in electrical engineering, computer science, and economics. He is director of the ResiliNets research group at KU, PI for the NSF-funded FIND Postmodern Internet Architecture project, PI for the NSF Multilayer Network Resilience Analysis and Experimentation on GENI project, lead PI for the GpENI (Great Plains Environment for Network Innovation) international GENI and FIRE testbed, co-I in the EU-funded FIRE ResumeNet project, and PI for the US DoD-funded highly-mobile airborne networking project. He has previously held senior staff and research management positions at BBN Technologies, GTE Laboratories, and IBM Research, where he has lead DARPA- and internally-funded research in mobile, wireless, active, and high-speed networks. He has been program chair for IEEE GI, GBN, and HotI; IFIP IWSOS, PfHSN, and IWAN; and is on the editorial board of *IEEE Network*. He has been active in Science and Engineering Fair organization and judging in Massachusetts and Kansas for middle and high-school students. He is principal author of the book *High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*. He is a member of the IEEE, ACM, IET/IEE, and IEICE. His research interests include resilient, survivable, and disruption tolerant networking, future Internet architectures, active and programmable networks, and high-speed networking and systems.