Teaching with the Emerging GENI Network

James Griffioen, Zongming Fei, Hussamuddin Nasir, Xiongqi Wu, Jeremy Reed, and Charles Carpenter
Laboratory for Advanced Networking, University of Kentucky
301 Rose Street, Lexington, KY 40506, USA

Abstract— *Over the last few years the National Science Foundation (NSF) has been investing in and developing a new network called* GENI, *a wide-area testbed network for at-scale experimentation with future internet designs. The GENI network has recently become available for use and is beginning to attract users.*

In this paper, we take a closer look at GENI with a particular focus on how GENI can be used to enhance education in the areas of computer science and computer engineering. We describe what GENI is, the resources available in GENI, and how instructors might use GENI in their classes. Being early adopters, we describe our experience using GENI in our classes, and we point out various features and challenges of using GENI. Finally, we provide tips and pointers to instructors who are interested in incorporating GENI into their own classes.

Keywords: hands-on experiment, GENI, network testbed, network monitoring, network lab assignment

1. Introduction

Although the Internet has been extremely successful and has transformed essentially every aspect of our culture, there is wide-spread agreement that the existing Internet architecture suffers from various problems and limitations that are constraining the development of new and innovative services. To address this issue the National Science Foundation (NSF) has launched several efforts designed to investigate new network architectures capable of supporting the types of applications and services desired in a "future internet" (e.g., the Future Internet Design (FIND) program [1] and the Future Internet Architecture (FIA) program [2]). In addition, NSF noted that the only way to truly evaluate new network architectures and designs is to construct an at-scale (i.e., Internet-scale) testbed network on which researchers can experiment with their ideas. To that end, NSF created and charged the GENI Project Office (GPO) with the task of implementing a large-scale testbed network that could be used for the development of next generation internet architectures.

After several years of development, the *Global Environment for Network Innovations (GENI) [3]* has recently become available for use and is rapidly attracting researchers working on novel network architectures and applications. Although GENI is also available for educational use, the vast majority of the GENI effort to-date has been focused on, and involved, the network *research* community.

The goal of this paper is to present ways in which GENI can be used to enhance *education*, particularly in the areas of computer science and computer engineering. Section 2 begins by giving a brief overview of GENI and the types of resources available in GENI. Section 3 then provides an example of one of the possible ways in which a user might access and control/use the GENI infrastructure. Having provided an introduction to GENI, we then offer some potential educational uses for GENI in Section 4. Section 5 then describes projects that we have used in our networking classes at the University of Kentucky. We then briefly compare GENI to alternative approaches in Section 6 and offer some concluding remarks and tips to instructors in Section 7.

2. GENI Overview

The goal of GENI is to enable users to build and operate their own Internet-scale networks. To that end, GENI has *resources* (e.g., computers, routers, network links) all across the nation (and to some extent internationally via federation) that are available for users to reserve and use in their own (private) network. For example, a user working on new data center services might build a testbed network by reserving desktop style computers (physically located) in Utah, Wisconsin, Kentucky, and Georgia for use as "clients" in their network, server machines in Salt Lake City to mimic a "data center", and routers in Atlanta, Washington DC, Kansas City, and Salt Lake city, with links that interconnect the "clients" and "data center" into a complete network. In the past, building such a network would be impossible for the normal user, and would even be challenging, if not impossible, for ISPs that own and operate network routers and data centers. However, with GENI, it becomes relatively easy to create such a network; in fact, the above network could be created for the user in a matter of minutes.

Alternatively, consider a user that wanted to test out new services run out of a data center supporting mobile (wireless) users. In this case, the user might build a test network by reserving a set of mobile (wireless) nodes in New Jersey and in Wisconsin connected via routers in Washington DC and Atlanta to a set of "data center" machines in Georgia and Kentucky.

In short, GENI offers a wide range of resources all across the nation that users can quickly and easily compose into a network of their own design. Moreover, GENI can support multiple user networks simultaneously – each being independent of the others. In other words, each user gets their own network composed of a set of resources that have been reserved for that network. In GENI terminology, a user's network is called a *slice* of GENI (a subset of the GENI resources allocated for the user's network). Each individual resource in the network is called a *sliver* of the slice. The list of resources that comprise a slice are often represented by a *Resource Specification (RSPEC)*.

The set of resources that make up GENI are owned and operated by different entities. The set of resources that are operated by a particular entity is called an *Aggregate*. When a user reserves resources, GENI contacts the various aggregate operators where those resources are located to verify that the resource is available and that it can be reserved. Consequently, each aggregate operator has the ability to allow or deny requests for its resources.

Many resources are programmable, implying that the user is free to design, implement, deploy, and control the software that runs on the resource. In particular, users are able to develop their own network protocol stacks and are not required to use the standard TCP/IP protocol stack. Ideally users could build their own network from the ground up, starting with their own physical and data link layers and all the layers up to the application layer. However, for pragmatic reasons the current version of GENI only allows users to select from a set of conventional technologies at the physical and data link layers (e.g., Ethernet, 802.11, etc). However, beyond that users are free to build/use any network layers they desire. For example, if a user wanted to write and test their own network layer protocol (instead of using IP), GENI would support it. If they wanted to add new services or processing into routers in the middle of the network (things that might be viewed as layering violations in the current Internet architecture), GENI would support it. In short, GENI allows users to throw out the current Internet (TCP/IP) model and start all over should they want to.

2.1 GENI Resources

Unlike other wide-area testbed networks such as Planetlab [4] and Emulab [5] which offer one (or very few) types of resources (e.g., a raw PC or Vserver virtual machine), GENI was designed to offer a wide range of different types of resources ranging from PCs, to (truly) mobile nodes, to network processors, to high-speed packet capture devices, to high-end compute servers.

GENI resources can be allocated to a single user for exclusive use. For example, a raw PC can be included in a slice, and the creator of the slice can request that a specific version and distribution of Linux operating system (e.g., Ubuntu 10) be installed and loaded on the PC. The slice creator has complete control over the machine. On the other hand, GENI resources can be virtualized and shared by multiple users. The virtualization technology makes it possible for each user to get a piece of the resource, such as an OpenVZ container or a virtual router, without interferences from other users. The user can pretty much do whatever she/he want within her/his slice.

Three categories of resources are provided by GENI. The first category of GENI resources are end hosts, which include raw PCs, virtual PCs such as OpenVZ and Vserver, and virtual platforms such as python sandbox environments provided by the million node GENI [6]. As the GENI project installs GENI server racks on more campuses, server class computers with fast processors and large storage space will become available as high-performance end hosts to be included in the user slice.

The second category of GENI resources are routers. When setting up a network experiment, PCs and virtual PCs can be configured to act as routers. GENI also provides physical routers (such as Juniper routers installed at Atlanta, Washington DC, Kansas City and Salt Lake City connected to the Internet2 backbone). The user can request Juniper logical routers (virtualized routers) from the GENI ShadowNet aggregate [7]. In addition, wireless access points, open flow switches, network processors (NetFPGAs) and mobile nodes can be allocated to set up a customized network testbed for users.

The third category of GENI resources are links, both wired (such as copper and optical links) and wireless. GENI provides layer 2 connections between different aggregates. The user can request that a layer 2 physical link be reserved between routers or end hosts for a given slice. To study the behavior of residential users, one can even include cables to the home as a part of slice, in order to collect the usage data [8]. Virtual links can also be established among nodes in an experiment. They can be VLANs set up among nodes within an aggregate or GRE tunnels/TCP-UDP tunnels set up between nodes from different aggregates.

2.2 GENI Measurement Infrastructure

In addition to resources, GENI provides a rich set of instrumentation and measurement tools for monitoring user experiments. They enable users to collect measurement data easily and make sure that experiments behave as expected. For example, GENI instrumentation and monitoring tools (INSTOOLS) has been developing *slice-specific monitoring* infrastructure to capture, record, and display information about a user slice, based on the consideration that experimenters are primarily interested in the behavior and performance of their experiment [9]. After the user instructs the system to instrument a slice, INSTOOLS will automatically deploy, configure, and run monitoring software or services on the resources that comprise the slice. It will also set up servers to collect the information captured at these resources, process the information, and make it available to the user via a graphical user interface.

Three categories of data are collected. The first one is network tables such as ARP tables, IP address tables, and routing tables. The second one is traffic information, such as IP, ICMP, TCP, or UDP traffic over time. The last one is operating system information, such as CPU load, memory load, and loaded modules. All these data are presented via a portal, which provides a “one-stop shop" to access all the data collected from the experiment. The traffic and load information are presented as graphs showing how it changes over time, while other information is presented as tables. They are accessible from any web browser through the Internet. The user can quickly identify abnormal behavior by observing these graphs and tables.

3. Using GENI

The first step to use GENI is to get an account from a GENI aggregate. After answering several questions to provide the identity of the applicant, the user’s account will be approved by a GENI administrator. As GENI is a federated system, a user having an account with any aggregate can request resources from other aggregates to be included in her/his slice, subject to the policies of the aggregates involved.

GENI provides users with a graphical user interface called Flack, which is a Flash application accessible from a web browser [9]. It was developed as a part of the ProtoGENI project [10]. After logging into Flack, the user can find resources available across the GENI aggregates. To create a new experiment, the user only needs to drag the icons of the corresponding resources (raw or virtual nodes) from the desired aggregate onto the canvas, as shown in Figure 1. The user can specify which operating system image will be used in these nodes. Links can be set up between these nodes. Typically, a VLAN will be created for nodes from the same aggregate, while a GRE tunnel (IP-in-IP encapsulation) [11] will be created for nodes from different aggregates. The user can also drag a delay node from an aggregate to be used as a part of the link. The delay node will be used to emulate different characteristics, such as latency, bandwidth and loss rate, of the link.

After the “submit” button is clicked by the user, the GENI system will automatically allocate the resources from the corresponding aggregates and set up the experiment. This may take several minutes to finish because the machines will be booted and configured according to the specification provided by the user.

Once an experiment has been created, the slice can be instrumentized as shown in Figure 2. The first step is to click on the plugins tab. The “instrumentize” button will show up. The second step is to click the “Instrumentize” button. The instrumentation tool will distribute measurement software to experimental nodes and configure them to capture traffic

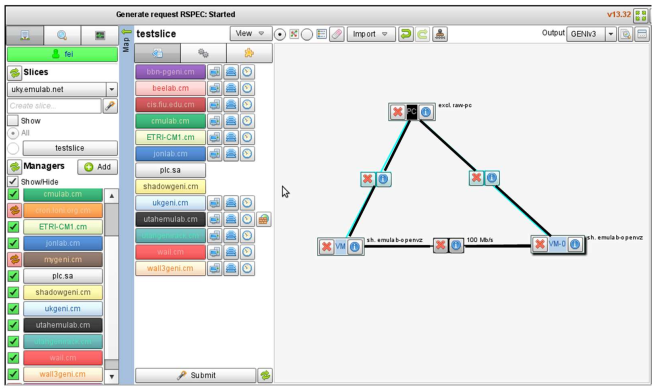

Fig. 1: Flack interface.

and load information on these nodes. One measurement controller will be added for the slice at each aggregate. It will be used to collect and process data captured at experimental nodes, and act as a server to present the measurement data to the user. When the instrumentation process is finished, clicking on the “Go to portal” button (step 3 in the figure) opens a browser window to the INSTOOLS portal for the experiment.

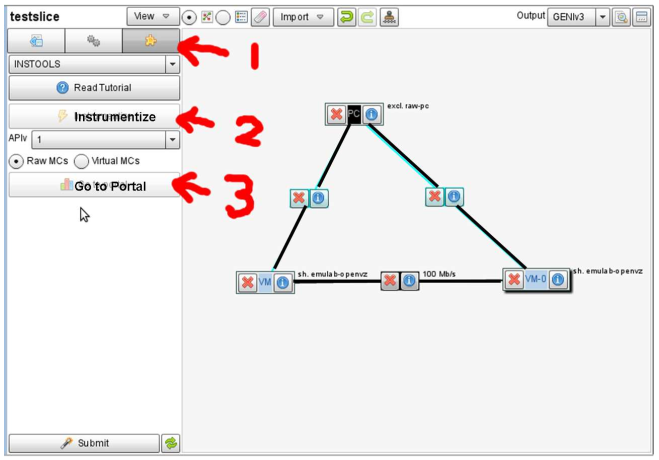

Fig. 2: Instrumentize an experiment

The portal presents the user with the collected information. Figure 3 shows the physical topology of the experiment to the user. It is a map view overlaid with the resources and network topology used in the slice. The user can also choose to look at the logical topology, which ignores the location information and can be read more easily. The user can click on any node or any link and a dialog box will appear, from which the user can choose what performance information to observe. Whatever is picked by the user will show up on the left-hand side of the interface.

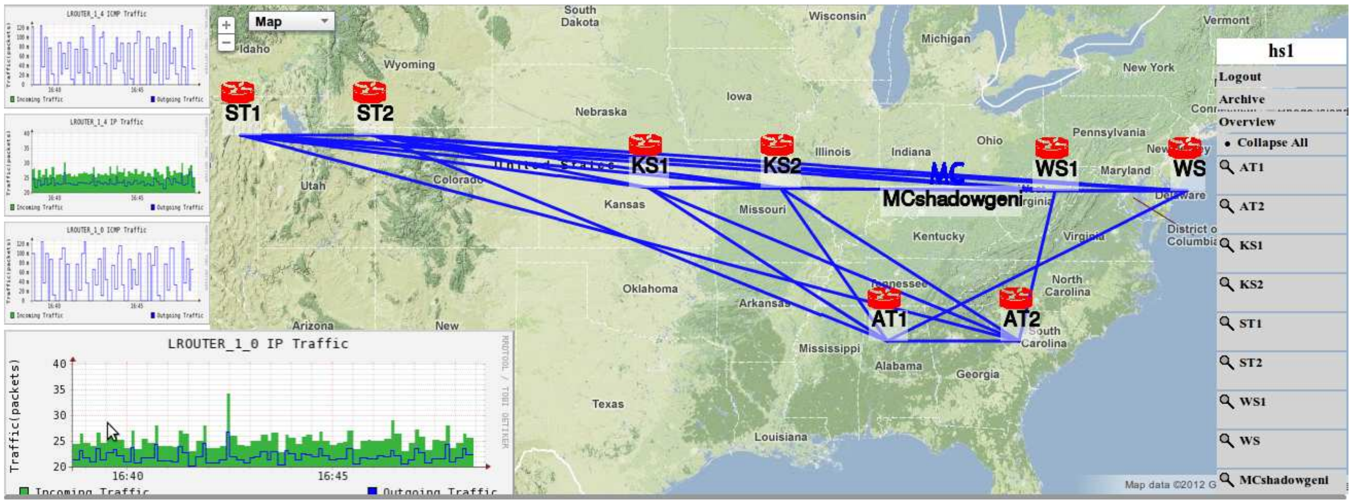

Fig. 3: The monitoring portal

4. Educational Uses for GENI

Networking classes have long used a variety of different "lab" environments to give students hands-on experience. Examples range from a single PC executing a network of virtual machines (e.g., VMware, Virtual Box, OpenVZ, etc.), to custom-built networking lab facilities [12], to emulation facilities like Emulab [5], to wide-area overlays like Planet-lab [4]. This raises the obvious question "What can GENI do that these others cannot?" or stated another way, "Why should I use GENI?"

The short answer to that question is that GENI is in many ways a superset of these other approaches. Because GENI is based on Emulab, Planetlab, etc, most anything that can be done in one of these other environments can also be done in GENI; *but* GENI also offers features not available in these other environments. In particular, GENI is designed to support "at-scale" networks that can scale up in terms of the number of resources offered, the types of resources offered, and the speed/performance of resources offered. As a result, a variety of new types of experiments and students projects are enabled by GENI–projects that are difficult, if not impossible to implement using past approaches. The following briefly outlines various types of projects that are enabled by the new GENI infrastructure.

4.1 Types of Projects Enabled by GENI

The following is not intended to be a comprehensive list of the types of projects possible on GENI, but rather is presented to give an idea of the potential uses for GENI in computer science and computer engineering courses. The first two types of projects mentioned below are possible using past systems and have been widely used in operating system and networking classes. GENI not only supports these, but enables a variety of new types of network experiments:

1) *Conventional OS/Networking Assignments*: These projects ask the student to make modifications to existing OS and networking code to create their own (routing) protocols or network services. Systems like Emulab, virtual machine networks, and custom lab facilities have long been used to support these types of assignments. Many GENI resources also allow students to have complete control over the software that runs on GENI resources.
2) *Network Monitoring Assignments:* These projects ask the student to write active (intrusive) and passive monitoring code to measure the performance of the Internet. GENI, like PlanetLab, offers geographically distributed resources that have been particularly useful as a basis for these types of assignments.
3) *Data Center/Cloud Assignments:* With the emergence of cloud computing, GENI, unlike past systems, offers high-performance clusters that can be used to implement "data center" services using custom or conventional (e.g., hadoop) data center software.
4) *Mobile Networking Assignments:* Because GENI includes a variety of (virtualized) mobile resources, it is an excellent platform on which to do assignments involving mobile users and code.
5) *Wireless Assignments:* GENI supports a variety of wireless network technologies including both mobile and fixed infrastructure that allows students to do assignments that must deal with the realities of wireless networks (e.g., variable loss rates).
6) *Home Networking Assignments:* While there are limited home network resources (e.g., cable modem/DSL) available, the ability to write code that utilizes the resources of opt-in home users is a unique feature of GENI.

7) *High-performance Networking Assignments:* Most student assignments are designed to build something that works, with performance as an afterthought. However, GENI offers high-performance servers, programmable network processors, and optical networks that enable assignments that test scalability of performance.
8) *Application-level Monitoring Assignments:* GENI includes some unique application-level resources such as low-power radar sensors and web cameras that are virtualized and accessible for use by users. Moreover, the high-performance network links available in GENI make it possible to move data off of these devices to network servers in real-time.
9) *Complete Network Assignments:* Because GENI supports so many different resources, it is possible for students to work on projects that involve every aspect of a complex/complete network ranging from (mobile) client nodes connected via wireless links to an optical backbone networks with advanced services built-into the network structure, as well as data center computing power offering cloud services.

5. Experience of Teaching with GENI

We have been using GENI to teach network courses at the University of Kentucky, including hands-on projects that use GENI. Students have enjoyed many features provided by GENI. The controlled environment makes it worry-free for unexpected changes to operating systems and network connections. The widely distributed GENI aggregates enable experiments to be deployed over machines distributed across the Internet. The easy setup and teardown let students make mistakes and modify and improve their experiments within minutes. In this section we describe in some detail three networking experiments that are helped by the availability of GENI. We have given the first two projects in our networking classes. The third project is one that we are considering giving in the future.

5.1 Reliability Protocols

Reliability is one of the most important concepts in computer networks. Traditional mechanisms for implementing reliability include Stop and Wait, Go Back N, and Selective Repeat. TCP uses a slightly different reliability protocol by using cumulative acknowledgment and buffering out-of-order packets at the receiver.

A programming assignment can let students implement one of the reliability mechanisms. The sender and receiver can be implemented as a UDP client and a UDP server. A timer must be implemented in order to deal with the cases when data or acknowledgment packets are lost. The goal is to observe the behavior of the programs in a wide variety of loss scenarios.

The problem with testing these programs in a general purpose computer lab is that we typically do not observe much loss in normal networking conditions. Therefore, all data packets to the receiver and all acknowledgment packets back to the sender get through without loss. Consequently the timeout and retransmission mechanisms will not be tested. Ideally, we would like to have a controlled environment in which we can get whatever loss rate we would like for any links. The scenarios should be repeatable. This is hard, if not impossible, to achieve in a general purpose lab.

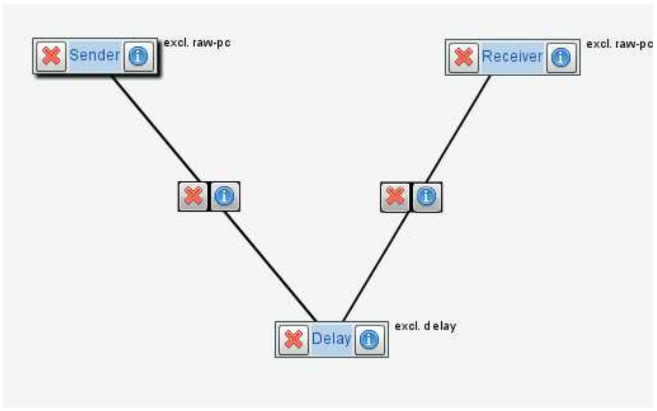

Fig. 4: Sender and receiver connected with a delay node as the link between them.

In contrast, GENI allows students to set up a network topology with a sender node, a receiver node and a link between them as in Figure 4. The link can be implemented by a delay node. GENI allows the delay node to control the link characteristics, especially the loss rate in this case. All these can be easily set up using the GUI provided by GENI and the experiment can be up running in a few minutes. Students can run their programs on the sender node and the receiver node, which typically are loaded with standard Linux OS. The loss rate can be specified to whatever values we want. We can do stress test of the protocol implemented to see whether it can handle all kinds of extreme scenarios.

5.2 Network Configuration and Automatic Route Control

In this project, students first create a network topology and then manually configure the network routing using conventional network administration software. In the second phase, students implement, deploy, and test a new/emerging type of router that separates routing from forwarding. In particular, they will write a Forward Information Base (FIB) controller process that allows remote control of the forwarding operation of a router. This creates the potential to run Routing Decision (RD) services remotely - for example, on a centralized controller node. By sending commands to a FIB Controller, an RD server is able to modify the Forwarding Information Base in a router.

The first step of this project is to create a network topology using the ProtoGENI Flack interface. For example, students can create a 4-node topology as shown in Figure 5, where the two nodes on the top are shared nodes from the utahemulab component manager and the two nodes at the bottom are from the ukgeni component manager.

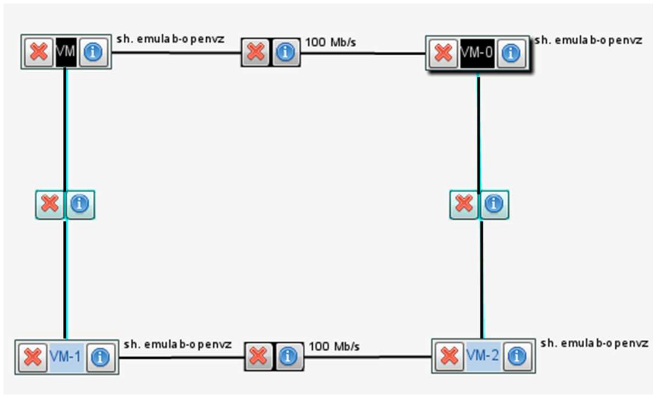

Fig. 5: A sample network topology.

Students login to these machines and use Unix commands such as *ifconfig*, *netstat*, *arp* and *route*, to observe and set up routing so that one can ping from any node to any other node. This manual configuration process helps students gain experience with the basics of IP, ARP and routing.

The second part involves writing programs to act as a FIB controller at each node and the RD server to control the routing through the FIB controllers.

The FIB controller is a process that runs on every router. It accepts commands from RD servers and then inserts or deletes the appropriate routes in the FIB (routing tables). To insert or delete entries in the FIB, the FIB Controller uses the Unix "system()" system call or the UNIX "exec()" system call to run the Unix *route* program. The FIB Controller and the RD Server speak a FIB Controller Protocol (FCP) that students design and implement. Students are free to design any FCP protocol. The only requirement is that the FCP be run over the UDP protocol (i.e., carried in the payload of UDP packets). The FIB Controller should receive UDP packets (from the RD servers) on a predefined UDP port number.

The RD server runs on a separate node - a node that is not acting like a router. The RD needs to be able to send IP packets to every one of the FIB Controllers it is controlling. This implies that there exists a direct link from the RD node to each router, or there exists an IP path from the RD node to every router. Students need to modify the above topology to meet these requirements.

As mentioned above, the RD server will speak the FCP protocol with the FIB controllers that it manages. The RD server will not make any routing decisions itself. Instead, the RD server will accept commands from a computer terminal (command-line interface), translate the commands into the FCP protocol and send them to the appropriate FIB Controller to be install/deleted. The command-line interface can include commands such as creating a path to a destination address, deleting a path, showing the routes to an IP address.

5.3 Path Characteristics of WANs

Most of our network classes use the Internet as a basis to teach principles of network design. To get a better understanding of the Internet, it is essential to understand the characteristics (such as delay, bandwidth, loss rate) of the links/paths in the Internet. We can ask the following common questions about the Internet. What is the typical delay of a path from the east coast to the west coast? How do delay, bandwidth, and loss rate differ between a local link and a wide area path, or among different wide area paths? Do they change a lot over time? The goal is to understand these characteristics and how they affect the design of Internet protocols. For example, we need to track round trip time in TCP because it changes over time. TCP congestion control protocol needs to dynamically adjust the congestion window because available bandwidth over a link will vary a lot depending on the competing traffic.

This project requires students to collect measurement data from the current Internet. They have to compare the characteristics of the different paths chosen, and then observe how the performance changes over time for each path. A simple tool such as ping can be used to measure the round trip time. We can send ping traffic to some popular websites, such as www.google.com. This has two problems. The first problem is related to the high frequency pings performed automatically by programs. Sometimes they may be misunderstood as denial of service attacks. We did get emails requesting explanation or termination of tests from those sites in the early days. The second problem is that these tests are one-way operations. We can send ping to these well-known websites, but the responses are pre-determined by the ICMP protocol. We can use other tools, but have similar problems that we cannot program the target site to reply in the way we want.

The alternative approach is to get guest accounts from friends in other universities or corporations. Typically, the number of these guest accounts is very limited, partially because various security concerns lead organizations to impose strict policies on guest accounts allowed.

GENI provides an easy way to have machines located around the country and even globally. We can define a topology consisting of machines from selected aggregates. We can send traffic between these nodes either through the dedicated links, or through the normal Internet paths via their public interfaces. We have control over both ends of the path we are interested in. With this capability, students can write/run their programs or use existing tools such as pathrate, pathchar, on the experimental machines allocated by GENI. These programs can collect path information over the wide area networks over time.

6. Related Work

Virtualization (vmware, virtualbox, etc) [13]–[15] has been widely used in Operating System/Networking courses for projects because it is a simple and cost effective way to give students complete access over a computer. Logical links can also be set up among VMs to establish the topology of a network experiment. However, VMs have limitations on the flexibility of communication with other machines and do not offer the performance of real hardware. More importantly, they do not offer the ability to control the delay/bandwidth of the network links, or the geographic location/distribution we get in GENI.

Emulab [5] is the original software the current ProtoGENI is based on. It offers many of the features of GENI, such as easy allocation of resources, easy setup of experiments and friendly graphical user interface. However, it is a single site facility and thus does not provide the geographic distribution of resources. Besides, it consists of a cluster of PCs and does not offer the wide range of resources (wireless/mobile, compute servers, juniper, openflow, optical routers, etc) as GENI does.

Planetlab allows users to set up experiments using Vserver of Planetlab nodes widely distributed across the Internet [4]. It offers geographic diversity, but does not offer the ability to control all layers of the network. The system does not set up links for communications between experimental nodes. Instead, the user has to create overlays links among nodes to set up the topology of an experiment. Using Vserver prohibits users from controlling the operating system in experimental nodes. It also results in poor performance compared to physical nodes. Although resources are distributed, there are very few resources at each location – typically 2 nodes at each location.

Special purpose network lab facilities (like the hands-on lab at Purdue, or the lab experiments by Jorg Liebeherr and Magda El Zarki) can be used for hands-on experiments [12], [16], [17]. However, these dedicated labs have cost and maintenance issues, are not easily (concurrently) shared by students, typically offer only a single configuration (e.g., network topology, operating system, application software) that can only be modified on the timescale of days or weeks (if at all possible), and can only be used when the lab is "open". It also requires a lab monitor to allocate lab resources to users. Moreover, because resources are limited, it is is impossible to experiment with large-scale systems consisting of many nodes separated by large geographic distances.

7. Conclusion

Hands-on experiments are an essential part of computer science courses for students to learn practical skills by doing. Many of these experiments are enabled by GENI because of its unique features, such as its easy-to-use graphical interface, quick setup and teardown of experiments, and a wide range of available resources. Though GENI can further improve the access methods of experimental nodes, it definitely makes a dramatic advance in the ways in which the hands-on experiments can be done. To explore further, we recommend the ProtoGENI website [10] and the GENI tutorial [9].

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under grant CNS-0834243. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of BBN Technologies Corp, the GENI Project Office, or the National Science Foundation.

References

[1] "NSF Future Internet Design Project," http://www.nets-find.net/.

[2] "NSF Future Internet Architecture Project," http://www.nets-fia.net/.

[3] The GENI Project Office, "GENI System Overview," http://www.geni.net/docs/GENISysOvrvw092908.pdf.

[4] "Planetlab: An Open Platform for Developing, Deploying, and Accessing Planetary-scale Services." http://www.planet-lab.org.

[5] "The Emulab," http://www.emulab.net.

[6] "Million Node GENI," http://groups.geni.net/geni/wiki/ GeniAggregate/MillionNodeGeni.

[7] "GENI ShadowNet Project," http://groups.geni.net/geni/wiki/Shadownet.

[8] "The HomeNet Project," http://homenet.hcii.cs.cmu.edu/.

[9] J. Duerig, R. Ricci, L. Stoller, M. Strum, G. Wong, C. Carpenter, Z. Fei, J. Griffioen, H. Nasir, and J. R. ans X. Wu, "Getting started with geni: A user tutorial," *ACM SIGCOMM Computer Communication Review (CCR)*, January 2012.

[10] "ProtoGENI." http://www.protogeni.net.

[11] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Gneric Routing Encapsulation (GRE)," RFC 2784, March 2000.

[12] D. E. Comer, *Hands-On Networking with Internet Technologies*. Upper Saddle River, New Jersey: Prentice Hall, 2004.

[13] "VMware." http://www.vmware.com.

[14] "Xen Source™." http://www.xensource.com.

[15] "The Linux Kernel-based Virtual Machine (KVM)," http://kvm.sourceforge.net.

[16] J. Liebeherr and M. E. Zarki, *Mastering Networks: An Internet Lab Manual*. Addison-Wesley, 2004.

[17] J. C. Adams and W. D. Laverell, "Configuring a multi-course lab for system-level projects," in *Proceedings of the 36th SIGCSE technical symposium on Computer Science Education*, February 2005, pp. 525–529.