GINI: A User-Level Toolkit for Creating Micro Internets for Teaching & Learning Computer Networking

Muthucumaru Maheswaran*, Alexis Malozemoff†, Daniel Ng†, Sheng Liao†, Song Gu†, Balasubramaniyam Maniymaran†, Julie Raymond†, Reehan Shaikh†, Yuanyuan Gao†
Advanced Network Research Lab
School of Computer Science
McGill University
*{firstname}.{lastname}@mcgill.ca †{firstname}.{lastname}@mail.mcgill.ca

ABSTRACT

GINI (GINI Is Not Internet) is an open-source toolkit for creating virtual micro Internets for teaching and learning computer networking. It provides lightweight virtual elements for machines, routers, switches, and wireless devices that can be interconnected to create virtual networks. The virtual elements run as unprivileged user-level processes. All processes implementing a virtual network can run within a single machine or can be distributed across a set of machines. The GINI provides a user-friendly GUI-based tool for designing, starting, inspecting, and stopping virtual network topologies. This paper describes the different components of GINI, briefly discusses ways of using the toolkit in a computer networking course, and reports on user feedback on an early (incomplete) version of the toolkit.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses In Education; D.2.0 [**Software Engineering**]: General

General Terms

Design, Experimentation, Human Factors

Keywords

computer networking, education, training, emulation, networking hardware, virtualization

1. INTRODUCTION

It is widely accepted that experimentation is the most effective way of learning computer networking. Several approaches [8, 12, 13, 16, 19, 20] have been proposed to experiment with computer networks and they can be grouped into two major classes: (i) dedicated laboratory-based approaches and (ii) discrete-event simulation-based approaches. While dedicated laboratories faithfully mimic portions of the Internet, they are costly, need continuous maintenance, inflexible, and provide limited capacity. Although many discrete event simulation based toolkit have been developed for computer networking, they are ideally suited for evaluating and studying performance trade-offs and less suited for learning the intricacies of networking. This is because simulation frameworks by design abstract away low level details that are often important in the learning process and do not provide the same operational model as the real network.

GINI (GINI Is Not Internet) is an open-source toolkit designed and implemented by the authors that provides an entirely software-based approach containing many of the features found in more expensive laboratory-based solutions. The GINI provides lightweight but IP (Internet protocol) compatible virtual elements for machines, routers, switches, wireless access points, and mobile devices. The virtual elements can be interconnected to create virtual networks for experimentation purposes. The GINI provides a tool with a GUI (graphical user interface) called *gBuilder* (see Figure 1) to design, start, inspect, and stop virtual networks. The processes that are created as part of the elements of a virtual network such as virtual machines can all run within a single machine or be distributed across multiple machines. The GINI is designed such that it can install and run without special privileges (e.g., super user access). This allows students to use GINI toolkit on machines provided in university computing centers or on their personal computers.

One of GINI's strengths is its realism. One of the central components of GINI is the virtual element representing a machine. A custom version of user-mode Linux is used for implementing the virtual machine. This allows Internet-based applications such as email, web servers/clients, and file transfer to run unmodified on GINI. Further, it means the skills acquired through interactions with GINI is applicable in real-world situations. GINI is also quite extensible. It provides a framework that can be enhanced by the addition of new networking protocols and elements. For instance, functionality can be added to the virtual routers provided with GINI to support multicast protocols. Similarly, new network elements such as IP version 6 to IP version 4 translators can be added by modifying new elements or writing them from scratch.

Another benefit of GINI over similar tools is its simplicity. If an industrial strength IP router (e.g., one provided inside Linux kernel) is used for experiments, the sheer complexity of the component is going overwhelm the students and digesting the software architecture to modify or enhance the component is simply impossible within a single semester. With GINI, students are provided with a very lightweight and simple router, easy enough to fully understand and modify over the course of a semester. This is one of the strengths of GINI: it is rare for students to delve into enhancing a router in an introduction to computer networking course, but GINI makes it possible.

It is often helpful for students learning computer networking to experiment and observe traffic. Due to security concerns, this is not possible with live Internet even under lab settings. GINI allows students to create their own virtual network that can run standard applications and can be observed using standard packet visualization tools. Designing new protocols and networking elements is another focus of computer networking, but one that is often taught in theory and not practice. GINI provides a platform in which new devices and protocols can be developed and tested without fear of security or other issues.

The estimated user familiarization time for learning GINI and all of its components is 2-3 weeks, which is only a small portion of a semester, leaving enough time for more in depth learning. Through *gBuilder*, GINI provides a user-friendly interface that automates many tedious tasks but provides sufficient feedback on the state of the network. Experimental versions of GINI have been used in the Computer Networks course at McGill since 2004. The feedback from the students have been used improve various aspects of GINI. It is currently available for public use [1].

Section 2 of this paper describes each element in GINI together with the steps to create a virtual network. Section 3 discusses the implementation details of the various components. Related work is discussed and compared with GINI in Section 5. Section 4 illustrates past and future development of GINI.

2. BUILDING NETWORKS WITH GINI

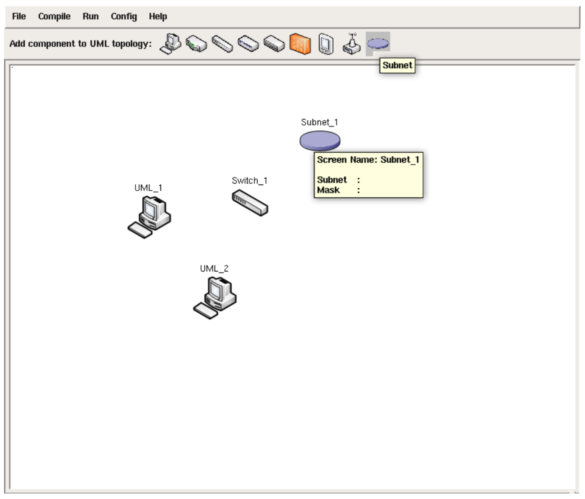
The *gBuilder* is the main interface used by the users to interact with GINI. The The *gBuilder* is simple enough such that a novice user can become familiar and sufficiently productive in very little time while an experienced user can design complex topologies with ease. As shown in Figure 1, the *gBuilder* includes a palette of network elements such as switches, routers, mobile devices, wireless access points, and wired hosts the user could use in composing the virtual network. The selects the desired devices and places them on the canvas and interconnects them using sub-nets and connectors. Currently, work is underway to add hubs, bridges, firewalls, and custom-networking devices to the *gBuilder*'s device palette.

The end machines are represented by custom-configured user-mode Linux (UML) [11] instances. The UML is a facility to run an instance of Linux as an application on a Linux machine (like using a virtual machine without installing any special software on the host machine). Several previous efforts have used UML for teaching system adminstration and networking concepts [9, 14]. Again mobile devices are represented as UML instances, however, the wireless channel is emulated by server called *gwCenter* that was developed as part of the GINI project. In addition to emulating the wireless channel, the *gwCenter* acts as an access point and wireless router.

Once the topology of a network is set, a network element's properties such as IP address, subnet mask, MAC address can be set using *gBuilder* (see Figure 2 for one example). For complex and relatively large topologies, manually setting up network elements' parameters and routes can be a tedious process. To alleviate this tedium, GINI provides an auto-generate option so that users only have to input subnet addresses, while everything else is generated at by *gBuilder*.

Once the network is configured, the user must compile the network. If the auto-generate option is set, the compilation process generates missing parameters, verifies that the specification complies with the rules built into *gBuilder*, and generates an XML topology file as output. The output file is used by *gLoader* to instantiate the appropriate network elements. The separation of *gLoader* from *gBuilder* allows an advanced user to by pass the GUI and use a command line tool to start and stop instances of virtual networks. The *gBuilder* has a built-in task management facility called the task manager which allows the user to inspect and kill processes that implement the different network elements (see Figure 3).

GINI provides several ways to visualize the performance of a running virtual network. For example, *gRouter* continously outputs time averaged queue sizes and packet throughput rates that are graphed by the *gBuilder*. Similarly, the *gwCenter* provides the packet throughput for wireless channels that are managed by it. When mobile devices are connected to through the *gwCenter*, the packet throughput information is displayed in a pop-up windows besides the mobile device as shown in Figure 4. The movement of a mobile device can be simulated by moving the icon of the corresponding mobile device on the canvas.



Screen Name: Subnet_1

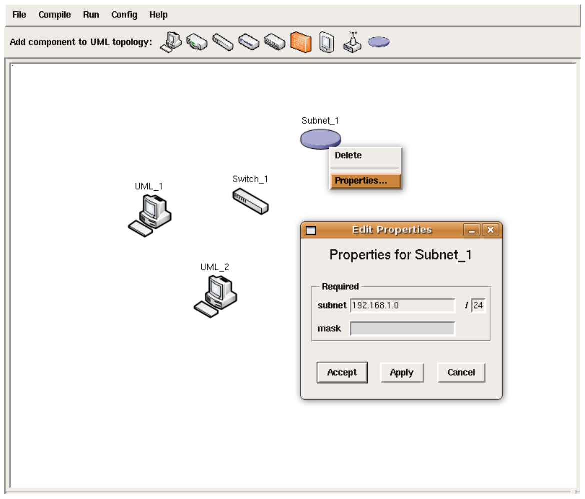
Subnet :
Mask :

Figure 1: Building a simple network of two machines connected by a switch using gBuilder.

3. IMPLEMENTATION DETAILS

The virtual elements of GINI that correspond to the machines, switches, routers, wireless access points, and mobile devices are coded in C. The glue infrastructure such as gBuilder, task manager, and gLoader are coded in Python. The entire code base is open source, licensed under the GPL. This provides students the ability to read and learn from the source code.

As discussed above, *gBuilder* is the main interface between



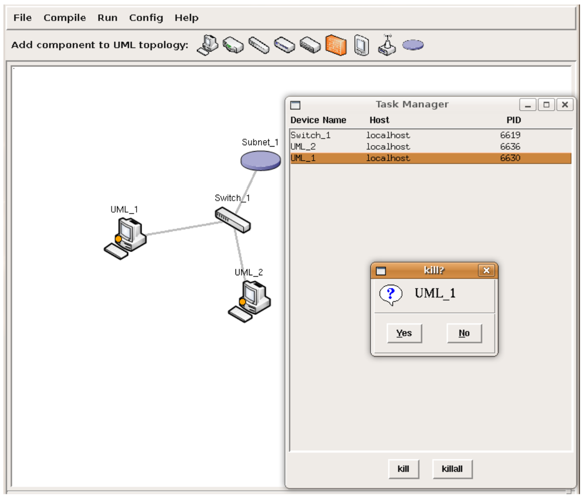
Properties for Subnet_1

Required

subnet 192.168.1.0 / 24

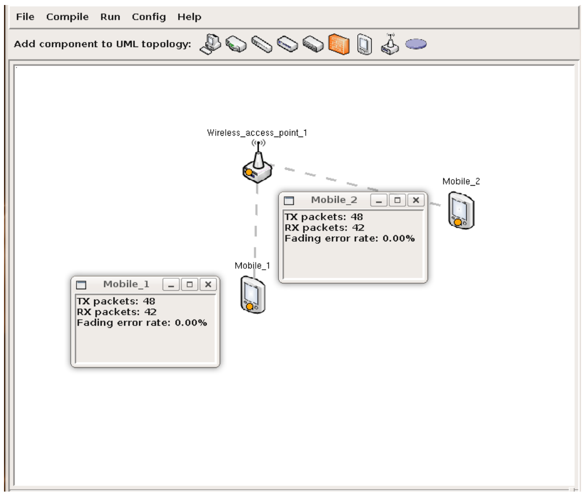
mask

Figure 2: Configuring the subnet properties.



Device Name	Host	PID
Switch_1	localhost	6619
UML_2	localhost	6636
UML_1	localhost	6630

Figure 3: The task manager for killing selected elements of a network.



```
TX packets: 48
RX packets: 42
Fading error rate: 0.00%
```

```
TX packets: 48
RX packets: 42
Fading error rate: 0.00%
```

Figure 4: Statistics (e.g., packet throughput) of a mobile node as it is moved in space.

the user and GINI. *gBuilder* handles saving and compiling the topology into an XML file, which can be parsed by *gLoader* to run the actual components and configuration as a whole. After *gLoader* launches the topology, *gBuilder* can be used as a tool for managing the running components. With a simple double-click of the mouse, *gBuilder* provides command shells for command-driven devices, such as machines, routers and wireless access points.

The attachable command shells are supported by an open-source program called *screen* [2]. The *screen* can handle multiple virtual terminals which can be attached or detached and can receive signals from the physical terminal. This program also provides a listing of the running screens and their state, which is handled by *gBuilder* through the task manager.

The management of host machines and mobiles devices is controlled by a program called *uml_mconsole*, provided by the UML project [3]. It is responsible for sending restart and shutdown signals to the UML devices. The other devices' (router, switch, etc.) restart and shutdown sequences are handled directly by *gBuilder*.

GINI uses a custom built software-based router called *gRouter*. While simple enough (around 6000 lines of code) to allow a student to fully understand the code over a semester, *gRouter* also provides full compliance with the Linux TCP/IP stack. The simplicity of the router allows projects to be assigned in which students can modify and extend the router's functionality.

Another benefit of *gRouter* is the flexible approach taken in developing the packet scheduling algorithm. It is easy for students to develop new algorithms and test them using features provided within *gRouter* and *gBuilder*. This facilitates exploration into different packet scheduling algorithms, an aspect that significantly impacts the performance delivered to the eventual traffic streams.

gRouter also includes a facility for traffic to be classified by incoming IP address, outgoing IP address, incoming port, outgoing port, and protocol. Once classified, the scheduling algorithms described above could be used to schedule the traffic, and different queuing disciplines, such as tail-drop and RED, can be assigned to manage the different queues.

The *gRouter* allows performance visualization through a real-time graphing tool provided by *gBuilder*. The packets flowing through *gRouter* can be visualized by connecting a Wireshark program through the plugin interface to the *gRouter*. The *gBuilder* provides a convenient drop-down menu for performing this connection.

GINI also includes basic wireless support, which emulates an ad-hoc wireless local area network [15]. This interacts with the mobile device components provided by GINI. The central controlling instance is called the wireless GINI center (*gwCenter*) which is also used as the wireless access point (WAP) for GINI. Mobiles connected to the WAP are considered nodes for *gwCenter*, where each node is identified through the MAC addresses provided in *gBuilder*. On top of efficient medium access control, *gwCenter* provides configuration for propagation channels, concrete frame corruption and node mobility. *gwCenter* has a command line interface which accepts configurations based on each of these elements for each node. To simplify this process, the configurations can be done from *gBuilder*, which sends the corresponding commands to *gwCenter*.

One of the strengths of GINI is the use of a custom Linux distribution, called *GiniLinux*. By using Linux-from-scratch [4] as the base of the distribution, we have been able to create a lean (80 MB in size) and extremely fast (5 seconds to startup and shutdown on average) operating system with all the necessary tools for network experimentation. One of the main benefits of this small size is the ability to launch multiple instances of *GiniLinux* without hogging all the memory on a computer. To reduce the overall size of the distribution we have forgone including a graphical user interface. While a certain level (the knowledge to execute relevant commands from a command shell) of Linux knowledge is necessary to use *GiniLinux*, most students have past experience using Linux or Unix-like operating systems, and thus learning to use *GiniLinux* is not a great challenge.

4. USER FEEDBACK ON EARLY VERSIONS

Figure 5 shows the results of a survey taken in the computer networking class at the end of Fall 2004. Although the toolkit was very primitive at that time, the results indicate that the students felt that GINI provides better insights into networking. Current version of the GINI toolkit is much advanced. It can support experimentations on a variety of different aspects of computer networks.

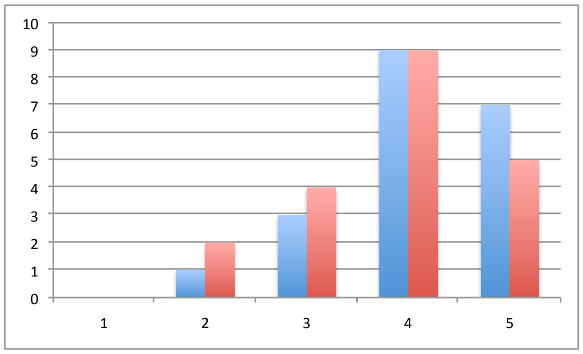


Figure 5: The results of a user study on GINI. The blue bar shows the response for the question "does GINI give additional insights into how Internet works?" and the red bar the response for the question "would you like to see GINI in future courses?." A strong yes is 5 and a strong no is 1.

5. RELATED WORK

As mentioned previously, systems related to GINI can be grouped into two classes: dedicated laboratories and simulators. The networking community has put significant effort in both classes and built several examples.

Computer networking research tends to focus heavily on simulators. This is because simulators provide an easy way of exercising specific aspects of networks for quickly answering the research questions [5, 17, 18]. Over the years, the familiarity of network simulators in university environments have made them attractive teaching tools for computer network courses [7]. GINI, on the other hand, is built first and foremost for educational purposes, specifically for teaching undergraduates or first-year graduates the basics of computer networking. Thus, GINI is simpler to understand than most of the tools mentioned above, an important requirement for students who are exposed to computer networking for the first time. Further, GINI is able to capture lot of low-level details and remain almost same as an actual implementation above the physical layer of the network. Due to the faithful reproduction of actual Internet conditions, students are likely to find the knowledge gained with GINI more valuable in real-world scenarios. Another problem with simulations from a teaching point-of-view is the difficulty of setting up meaningful simulation scenarios. For effective exposition of reality, simulators need to be fed with the correct parameter values which require advance knowledge of the real world and the simulation model. While this kind of knowledge can be assumed among research students, it is hardly true for non-research students.

There are numerous hardware lab-based approaches. Due to lack of space only two are mentioned. One of the recent approaches is Open Network Laboratory (ONL) [10]. While ONL is ideal for certain experiments, it is a dedicated laboratory and not suitable for large-scale adoption. Several approaches based on dedicated hardware are described in [12], including the use of virtual machines, the path taken by GINI. Although virtual machines are used in [12] for the end systems, dedicated routers and switches are used for other elements of the network. In contrast, GINI provides an all-software implementation.

Kneale and Box [13] raise many interesting issues with regard to current approaches for teaching computer networking, including cost issues and lack of access time to labs. They provide a system called Velnet based on a virtual machines-based approach to computer networking experiments. Unfortunately, Velnet uses the routing functionality embedded in Linux kernels instead of developing a lightweight router, leaving the students with a steep learning curve if they intend to delve into the router and add new functionality.

The Stanford Virtual Router [6] is a project designed to help students in router education. It provides a Virtual Router server that communicates with client routers, run by the students. The clients only see their own traffic, whereas the server manages the traffic of all the clients. While the Stanford Virtual Router provides a better alternative than the solutions that use Linux as the router, the GINI goes even further. By providing a distributed network of routers, the GINI provides an opportunity where GINI routers can be mapped onto machines far apart on the Internet to simulate wide-area linkages.

Two books on computer networking which take opposite approaches to network education are Comer's *Hands-On Networking with Internet Technologies* [8] and Matthews' *Computer Networks: Internet Protocols in Action* [16]. Comer relies on a dedicated laboratory for network experimentation. The book gives experiments that can be created and tested using the suggested laboratory. The use of a lab raises the issues of cost and access time for students, which in a large class could be too short to successfully learn the material.

Matthews' book, *Computer Networks: Internet Protocols in Action* [16], uses the novel idea of teaching networking by providing packet traces for many different networking situations, including SMTP, BGP, and RIP traces, to name a few. While definitely an interesting and noteworthy book, Matthews' approach fails to give students a hands-on approach, in the sense that they are unable to make changes to various networking settings or code and see the results. GINI

	GINI	Labs	Simulators
Dedicated Hardware		●	
Free	●		●
Hands-on	●	●	●
Level of Abstraction	Low	None	High
Open Source	●		●
Runs Locally	●		●
Software Complexity	Low	High	High

Table 1: Comparison of GINI with other popular networking education methods.

contains many of the benefits of Matthews' book without the negatives. By providing a Wireshark (previously known as Ethereal – an open-source industry standard packet visualizing tool) plugin to the router, students are able to visualize the packets as they flow through the virtual routers. Because everything is open source and accessible, students are able to alter the router and/or application code and instantly see the results through the Wireshark plugin.

Table 1 sums up the information described above by comparing GINI to the two most popular methods of hands-on networking education: dedicated laboratories and simulators.

6. CONCLUSIONS

GINI provides an all-software toolkit that can be deployed on a typical Linux machine to create an experimentation platform suitable for teaching and learning computer networks at universities. Although undergraduates and junior graduate students are the intended audiences of GINI, it is conceivable that GINI might be interesting to people wanting to learn computer networks for fun. One of the unique aspects of GINI is the easy entry it provides into the world of network experimentation. By removing the requirements for dedicated laboratories and advanced domain-specific knowledge required by simulators, GINI effectively lowers the barrier for entry into the fascinating world of network experimentations.

The GINI project began in 2004. Many experimental versions of the toolkit were used in computer networking courses at McGill. A highly stable and usable Version 1.0 of the toolkit can be found at `http://www.cs.mcgill.ca/~anrl/gini`.

7. REFERENCES

[1] GINI Project. http://www.cs.mcgill.ca/ anrl/gini/.

[2] GNU Screen. http://www.gnu.org/software/screen/.

[3] User Mode Linux. http://user-mode-linux.sourceforge.net/.

[4] G. Beekmans. *Linux from Scratch*. IUniverse.com, Inc., 2000.

[5] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in Network Simulation. *Computer*, 33:59–67, May 2000.

[6] M. Casado, V. Vijayaraghavan, G. Appenzeller, and N. McKeown. The Stanford Virtual Router. *ACM SIGCOMM Computer Communication Review*, 32:26–26, July 2002.

[7] X. Chang. Network simulations with OPNET. In *Simulation Conference Proceedings*, volume 1, pages 307–314, 1999.

[8] D. Comer. *Hands-on Networking with Internet Technologies*. Prentice-Hall, 2002.

[9] R. Davoli. Teaching operating systems administration with user mode linux. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 112–116. ACM New York, NY, USA, 2004.

[10] J. DeHart, F. Kuhns, J. Parwatikar, J. Turner, C. Wiseman, and K. Wong. The open network laboratory. In *Proceedings of ACM SIGCSE*, 2006.

[11] J. Dike. User-mode Linux. *Proceedings of the 5th conference on 5th Annual Linux Showcase & Conference-Volume 5 table of contents*, pages 2–2, 2001.

[12] J. Gerdes and S. Tilley. A conceptual overview of the virtual networking laboratory. In *SIGITE '07: Proceedings of the 8th ACM SIGITE Conference on Information Technology Education*, pages 75–82. ACM, 2007.

[13] B. Kneale and I. Box. A Virtual Learning Environment for Real-World Networking. *Information Science*, 71, 2003.

[14] A. Krap. Setting up a virtual network Laboratory with User-Mode Linux. In *Proceedings of the 4th International System Administration and Network Engineering Conference*, 2004.

[15] S. Liao. Wireless GINI: An Emulator for Ad-hoc Wireless Local Area Networks. *Masters Thesis*, 2005.

[16] J. Matthews. *Computer Networks: Internet Protocols in Action*. John Wiley & Sons, 2005.

[17] S. McCanne and S. Floyd. ns Network Simulator, 1995.

[18] G. Riley. The Georgia Tech Network Simulator. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 5–12. ACM New York, NY, USA, 2003.

[19] J. Theunis, B. Van den Broeck, P. Leys, J. Potemans, E. Van Lil, and A. Van de Capelle. OPNET in Advanced Networking Education. In *OPNETWORK 2002*, August 2002.

[20] J. Wang, B. Peng, and W. Jia. Design and Implementation of Virtual Computer Network Lab Based on NS2 in the Internet. In *Advances in Web-Based Learning - ICWL 2004*, pages 346–353, 2004.