AC 2009-2223: A VIRTUALIZED NETWORK TEACHING LABORATORY

Eric Freudenthal, University of Texas, El Paso
Eric Freudenthal is an Assistant Professor of computer science at the University of Texas at El Paso.

Frederick Kautz, University of Texas, El Paso
Frederick Kautz earned a B.S. in computer science at the University of Texas at El Paso.

Bivas Das, University of Texas, El Paso
Bivas Das is a M.S. candidate at the University of Texas at El Paso.

Luc Longpre, University of Texas, El Paso
Luc Longpre is an Associate Professor of computer science at the University of Texas at El Paso.

© American Society for Engineering Education, 2009

A Virtualized Network Teaching Laboratory

Abstract

Since for most students, learning dramatically improves with hands-on experience, a good networking lab is an asset for teaching networks. However, building such a lab is usually a challenge. It requires costly equipment and flexible configurations that are often not compatible with the campus network. In this paper, we describe how we designed a network teaching lab based on virtual machines connected on a virtual network. An instructor can create a virtual network and make it available to students. Students can configure the network and run experiments as instructed. When the task is complete, the students can submit the result of their work.

Traditional networking laboratories

A good network teaching laboratory is essential to support student learning in a Networks course. A traditional networking lab, in addition to the computers, requires networking equipment such as routers, switches and appropriate connections. The equipment needs to be updated regularly for the students to be able to apply the skills they learn in the lab directly in the work force.

Unfortunately, traditional networking labs are a fairly scarce resource. In addition to the cost of equipment and updates, it is a challenge to design the lab to allow flexible configurations. These network configurations are often not compatible with the campus network. Class assignments may be restricted to those that can be performed using the capabilities of the lab. For institutions with limited budgets, it may be impractical to purchase all the devices necessary for each student. Equipment that students have access to are often shared with other students. Usually, students are put into teams and the team must spend time allocated time slots when the lab is available. Once a team has worked in the lab, it may be time consuming and sometimes not obvious how to reset the lab to an appropriate configuration. This effectively restricts the kind of assignments given to the students.

Approaches to use virtualization

Although not specifically designed for a teaching lab, an approach to simplify the management of networks was proposed by Chandra, Zeldovich, Sapuntzakis and Lam.[1] They proposed the concept of a computational kiosk architecture, called *The Collective*. They utilize virtual machines as the execution environment for all user workstations (henceforth generically called ”computational kiosks” or ”kiosks”). Under The Collective’s model, user ”installations” are dynamically transferred to kiosks upon user login. By exploiting the computational power and the generic VMWare machine’s virtual architecture provided by each kiosk, users are provided high performance. User environments become seamlessly transportable among kiosks, and the kiosks themselves become virtually stateless, and thus quickly and inexpensively replaceable. The

virtual machines and their configurations are stored at a central repository. A computer can check for updates to the virtual machine regularly. This type of virtual machine is known as a LivePC. LivePCs can also be derivatives of other LivePCs. A system that has become inoperable can be reverted back to its original state easily.

A more recent approach to implement a virtual network laboratory was proposed by Loddo and Saiu.[4] They designed Marionnet, a system of virtual machines that is easy to set up and maintain. Marionnet contains a GUI that is designed to allow students to make virtual connections between virtual machines. This shifts the challenge of network configuration into a tool for teaching. This provides a very clean way to create containers running Linux. Marionnet was designed using a modified User Mode Linux.[5] However, this also limits the operating system of the guest to Linux. For networking, User Mode Linux was modified to use VDE instead of the standard kernel API calls. VDE allows for very flexible virtual distributed Ethernet networks to be created on the fly.[2] Various simple networking devices were simulated such as hubs and switches. Virtual routers are virtual machines with Quagga running on them. A graphical tool was implemented to allow simple, yet flexible reconfiguration of the virtual network.

The Reconfigurable Computer and Network Lab (ReCoN)

In the design of our lab, we extended the concepts proposed by the two approaches described above. Students in networking labs are able to conduct experiments on multiple networked virtual machines running commodity operating systems such as Linux and Windows. In detail (see Figure 1), a set of virtual network buses are created. Each virtual network represents a physical network between two or more nodes. Virtual machines are connected to each bus creating a network layout to be experimented on by the student. This is equivalent to connecting hosts to a hub. The virtual machines are loaded with commodity operating systems such as Windows or Linux. A virtual machine connected to two or more virtual networks with an operating system capable of routing packets can be used as a router or switch on the network. Interesting network configurations can be created using these components as building blocks.

Portable USB storage devices are becoming faster and holding larger quantities of data (for our prototype, we used an eight gigabyte Corsair Flash Voyager GT). Copy-on-write (COW) images provide a mechanism for efficiently storing file systems that are very similar (see Figure 2.) We instantiate multiple virtual machines using the same base image, with any changes to each virtual machine's file system being stored to a separate file. Copy-on-write semantics can be implemented either within the virtualization system or within the host operating system's filesystem. Our implementation utilizes virtual disk-image COW facilities provided by VMWare. With this, students are free to perform experiments using most computers that support booting from a USB flash drive.

Our initial implementatino used Following Monica Lam's collective model,[1] the virtual machines and their configurations were stored at a central repository which was transferrred via HTTP..[1]

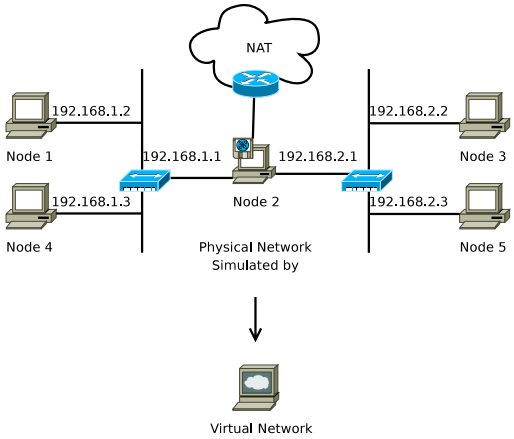

Figure 1: Virtualizing a Network Lab

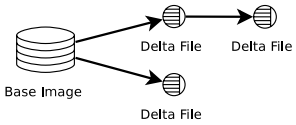

Figure 2: Cow File as Deltas of a Base Image

LivePCs can also be derivatives of other LivePCs. All all changes stored using COW images overlayed upon the base image. Furthemore, should a student makes an error that renders a virtualized system inoperable, the system can be reverted back to its original state easily.

Since our virtual machines are running commodity operating systems, we are able to utilize software commonly available to those operating systems including DNS, firewalls, and other critical network services. Students are also able to utilize software used specifically for routing packets. For example, students can use the click modular router to easily create routers with advanced functionality including packet classification, queuing, and scheduling.[3]

For our prototype, we used Linux for the host. VMWare Workstation was used to create the virtual machine images. VMWare Player was used by the students to execute and interact with the virtual machines. The virtualized systems are running Linux. Windows, BSD, and Solaris could easily fit in. Moka5 was used for packaging, transferring, and managing the Virtual Machines. Moka5 also provided a nice graphical interface to switch between running virtual machines. The entire system with virtual machines were stored on a bootable USB flash drive. The students checked out the flash drive and initialized them in class. Due to the current complexity of modifying Moka5 to support multiple host-only networks, and modifying the virtual machine's configuration file, the networking lab should be pre-designed by the instructor.

Combining the LivePC and the Marionnet concepts allowed us to make a unified lab creation, deployment, and student submission model that would help facilitate teaching network classes. A

LiveLab consists of both a group of LivePCs and a network configuration describing how each of these systems are interconnected. This description file is stored on the server which contains the location of each LivePC to be used and what network it is to be configured to. To save space and download time, the same LivePC can be cloned multiple times. Any changes to each LivePC will be stored in a separate file using a COW that overlays the base file system.

Our tool to create the lab configuration was inspired from Marionnet's network configuration tool, manipulating LivePCs instead of Virtual Machines. The tool is also able to publish the LiveLab. When students are working on the LiveLab, the changes to that particular LiveLab are stored in its own COW separate from other assignments that may use the same lab configuration. Students publish their own changes back to a central repository. Since each lab is kept separately, students can easily switch forward and back between each assignment. A syndication technique such as RSS or atom can be used to identify when new labs become available. The student's client should provide a mechanism to perform and submit multiple assignments using the same LiveLab or derivates of previous labs.

Using a repository with access control lists (ACLs), the type of submission to the repository can be determined without modifying the LiveLab itself. Both students and instructor can post the lab to a central repository. The instructor's account allows a lab to be published to a class. Since our entire student lab configurations are stored within USB flash drives, once the assignment is complete, students hand the drives in along with appropriate instructions. Instructors grade the lab by booting each student's virtual networked environment and examining its configuration.

This setup gives us a flexible laboratory that can be used to instruct students while minimizing cost and risks posed by these labs.

Example of networking lab assignments

The following is a set of labs that were built for our networking course. Since the LiveLab has not been implemented yet, we worked around this problem by populating a network of 5 systems with 2 networks.

1. This lab is an introduction to what a network is. It consists of connecting two virtual machines as seen in Figure 3. The student configures both computers so that they are capable of pinging each other.
2. Using the network built in lab 1, set up an http server and using a client to connect to it. If the networking lab includes a focus on programming network applications, this would be a good opportunity to write a very simple HTTP client.
3. The network in lab 1 is extended to include a second network and a third virtual machine that is part of both networks. See Figure 4. No routing is done between each network yet.

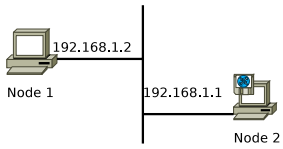

Figure 3: Two Node Lab

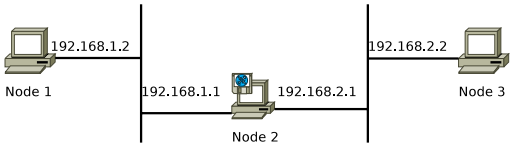

Figure 4: Trivial Routing Lab

4. Static routing is added to the network built in lab 3. If node 2 is a linux system, iptables can be used.

5. Add one computer to each of the two networks. This is shown in Figure 5. Set up simple networking services such as DNS.

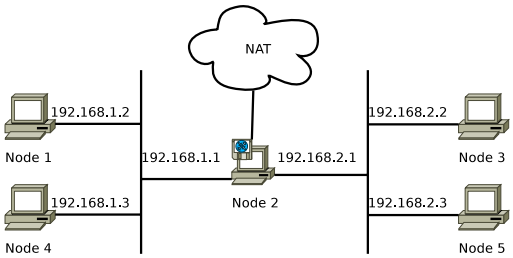

Figure 5: 5 System, 2 Network, and Network Address Translation

6. Make a query from node 3 to node 1. Capture the packets at node 2 on the virtual network containing node 2 and 3. Manually parse the packets to determine the contents of the packets. See Figure 6.

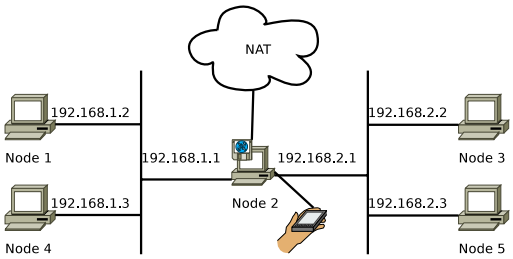

Figure 6: Example Packet Capture Location

For an advanced routing lab, students are able to implement routers using the click modular router. Using click, students are able to implement and test complex labs designed to teach them how routers work with a minimal amount of equipment and physical setup. Click itself can also be extended to include additional functionality such as routing protocols.

A research lab with enough computers can be used as a virtual cluster to experiment with both new routing protocols and peer to peer systems. For example, a lab with 20 physical systems each running 5 virtual machines would contain 100 independent virtual network nodes.

Further developments and directions

We have ported our virtual machines environment from VMWare to qemu running upon Ubuntu. Each client and router are a virtual PC running an Ubuntu Server 8.10 Linux distribution with a kernel optimized for virtualization. Virtualization is provided by the qemu cpu emulator. More details and configuration instructions are posted at the web site *http://wiki.utep.edu/display/robust/Labyrinth.*

Bibliography

[1] R. Chandra, N. Zeldovich, C. Sapuntzakis, and M. Lam. The Collective: A Cache-Based System Management Architecture. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05)*, 2005.

[2] R. Davoli. VDE: Virtual distributed ethernet. *Testbeds and Research Infrastructures for the Development of Networks and Communities, International Conference on*, 0:213–220, 2005.

[3] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297, 2000.

[4] J. Loddo and L. Saiu. Status report: Marionnet — How to implement a virtual network laboratory in six months and be happy. In *Proceedings of the ACM SIGPLAN Workshop on ML*, pages 59–70. ACM Press New York, NY, USA, 2007.

[5] J. Loddo and L. Saiu. Marionnet: a virtual network laboratory and simulation tool. In *SimulationWorks*, 2008.