Developing a flexible virtual networking laboratory platform for education

Catalina Álvarez
Universidad de Chile
catalin@uchile.cl

1 Abstract

Giving hands-on networking experiences to engineering students is important as it strengthens knowledge and it gives a better idea of the challenges they will find in practice; it is, however, expensive and impractical to have physical equipment for the students to practice with. Hence, the idea to simulate complete physical networking laboratories using virtualization technology for use in networking education. There are a number of existing virtual laboratory alternatives, but all of them are limited in the machine images they can use. This work presents the main aspects of the design and implementation of a more flexible virtual networking laboratory platform.

2 Introduction

One of the biggest challenges in teaching networking is how to bridge theory and practice. It is usual for students to feel that both aspects of the area are disconnected: on one side, they see protocols and algorithms, on the other, machines and links that they simply use. Hence, hands-on experience is valuable, and desirable, to produce well qualified professionals that will maintain and develop technologies in the future. Moreover, even professionals in related areas, such as software engineering, can benefit from experience in practical networking, allowing them to see better how the software they develop communicates.

Laboratories are the most common way to teach hands-on networking, either using physical equipment or virtual versions. Both alternatives have their pros and cons: physical laboratories allow students to touch and interact with the same equipment they will see on field, while virtual ones can be considered more abstract, and are harder to visualize. On the other hand, constructing and maintaining a physical laboratory up to date is expensive, and, depending on the number of students and the required courses, the time each student can interact with the equipment is limited.

Virtual networking laboratories come in two flavors: proprietary and open source. Proprietary laboratories, such as Cisco's, are usually paid, and mostly focused on teaching their own technology stack, showing configurations, but without theorical background (as most protocols are proprietary). Open source laboratories are varied, with a number of objectives.

Among famous open source networking laboratories, we can name Marionnet[1], Netkit[2], Mininet[3], GNS-3[4], among others. Each of them, however, with its own limitations; evaluating each platform is out of the scope of this document, but it suffices to say that all current virtual laboratories are limited to a set of operating system (OS) images, mostly Linux-based and some networking systems such as Cisco's IOS or some open source alternatives.

This fact is important because of two reasons: First, it limits the uses of the laboratories to those of a physical testbed, meaning, they allow the creation of topologies and use of the protocols already installed in the operating system, but does not allow changing those protocols and recompiling the kernels. Second, as the images are fixed, one either relies on the community to keep the images up to date, or takes the matter into their own hands and creates those images, which is possible, but could be hard depending on one's knowledge and the laboratories documentation.

Considering the points established previously, we decide to design and implement an open source networking laboratory that is flexible enough for the use of any kernel, even custom ones, and allows the easy inclusion of new operating system images.

3 Related work

Most of the existing networking laboratories can be divided in two categories:

In: Proceedings of the IV School of Systems and Networks (SSN 2018), Valdivia, Chile, October 29-31, 2018. Published at http://ceur-ws.org

- Simulators, which model network behavior but do not keep the internal functionalities of the hardware; among these we can name GNS-3 and OMNeT++. In networking, simulators mostly use mathematical models of traffic, channels and protocols to predict network behavior. As they only mimic and are unable to faithfully represent all aspects of networking, they are not interesting for our ends.

- Emulators, which differ from simulators in the fact that they do appear, and act as, a real network; emulators use software to duplicate the conditions of the original system, fact that make them slower, but more realistic, than simulators.

We focus our investigation on emulators, as they give a more realistic approach to networking. We present a summary of the evaluated emulators, but first we present a small comment on virtualization technologies, relevant to virtual laboratories. There are several virtualization platforms and techniques, but they all can be roughly divided in two categories: full-virtualization platforms and para-virtualization ones. Para-virtualization allows the guest machine to use portions of the host machine's kernel, including I/O, thread and memory management, among others, instead of emulating these operations via software; on the other hand, full-virtualization engines emulate the entirety of the guest machine kernel, including costly operations such as the previously mentioned, making this technique considerably more resource intensive and the guests, slower. There are, however, several advantages to full-virtualization, such as the fact that it can virtualize all OS, with no modifications whatsoever; on the other hand to paravirtualize an OS it must be explicitly ported to the para-API, which makes standard OS unable to run on top of para-virtualization platforms. Moreover, paravirtualization, as it uses portions of the host kernel, is not able to emulate a different hardware architecture; this point is critical as a number of networking equipment, such as routers, switches and firewalls, do not use the x86 architecture commonly found in personal computers. Hence, we decide to use full-virtualization, because we desire to be able to emulate as much types of network equipment as possible, without being restricted by the architecture or the need to port the para-virtualization API.

Continuing with our related work investigation; first, we explore Xen Worlds[5] and NVLab[7], both based on Xen server, the technology used in Amazon Web Service; all emulators based on Xen are discarded due to the fact that Xen is a bare-metal hyper-visor, meaning, it runs directly on hardware, with no host operating system, which makes virtual machine administration harder. Xen is both a para-virtualization and full-virtualization platform, which is ideal since it functions with para-virtualization with compatible kernels and full-virtualization with those that are not; however, running Xen requires a compatible kernel (which not all Linux versions are) or the use of a commercial version, such as Citrix's Xen Server. Moreover, both projects based on Xen were found to be quite old and with no continued development or even available code.

Next, we consider Netkit, a popular teaching laboratory based on UML (User Mode Linux); it has a number of pre-made laboratories, which are considered as a base for our own, and active community participation. However, it is not useful in protocol experimentation, as UML uses the same kernel as the host machine, which, naturally, means that all laboratories are bounded to the networking implementations found in Linux kernels, which, as mentioned before, is not what we are looking for.

Third, we explore Mininet, a network emulator focused on SDN and Open Flow learning with an active community and a number of fork projects. It uses network namespaces (a containerization mechanism of the Linux kernel that provides a way to copy the network stack of the Linux kernel) and process based emulation, so it only has as many tools as the Linux kernel on top of which it is running, without support for any stack based on Windows, BSD, or any other OS; Mininet is a good alternative, because of its community support, but if we want a laboratory able to emulate all different equipment found in real networks, we need an alternative more flexible in what operating systems it supports.

Finally, we find Nemu[6], based on QEMU and with mobile simulation capabilities; it is evaluated favorably, but presented two problems that we find pivotal in the decision of developing a new networking laboratory: First, its development was halted midway and much of the functionality is unstable or poorly documented; moreover, it is basically impossible to run, and even its website was put down during our development, implying that the work has been halted or canceled. Second, as the previous emulators, uses fixed pre-configured virtual machine images, which do not have the flexibility desired.

We conclude from our investigation that, in order to allow our students to experiment with all types of network equipment (including those which operating system is not based on Unix platforms) we should develop our own virtual laboratory platform.

4 Development of the platform

The platform is open-source and available for download in `https://github.com/niclabs/VirtualLabs`.

We decide to use QEMU/KVM for virtualization for a number of reasons: first, in combination, they are a full-virtualization platform that allows custom kernels and can emulate different architectures. Second, differently from Xen Server, QEMU/KVM work on top of a Linux-based operating system, which allows for easier administration of the virtual machines. This way, the virtual laboratory platform can be run on almost any host machine (the laboratory "server"), as long as it has a Linux-based OS and supports virtualization.

The most extensive part of the design stage was deciding how the different elements in a network topology would be modeled; in particular, we decide to model the most common elements find in a network: terminals (end users), switches, routers and the links that connect them. At this stage of the development, we focus on open-source solutions, so we choose Linux-based OS for the terminals, LISA (Linux Switching Appliance[8]) for switches, VyOS (a fork of the Vyatta project) for routing, and standard Linux bridges for links.

It is important to mention that we are extremely conscious throughout the modeling stage, and later the implementation, that we must avoid the main pitfall found in current virtual laboratories, their restrictiveness, so we design the system so that it is easy to include new networking elements such as load balancers, firewalls, NAT servers, etc.

Once we define how the different elements involved are modeled, we design how these models are to be implemented; in particular, how we will create the elements of the network, meaning, the virtual machines that are to be terminals, switches and routers. A first approach would be to simply keep an iso image for each operating system, but installing a virtual machine from scratch each time one needs a terminal is simply too time consuming. Luckily, KVM provides "templates", virtual machines with an installed operating system which can be "cloned" as many times as necessary; machines created from the same template do not share configurations nor disk, so they are, in essence, different machines. Using these templates, we can provide a number of ready to use base terminals, switches and routers, which can be copied as many times as necessary; moreover, including new elements to our laboratories is just a matter of creating new template images. The virtual machines communicate with each other using Linux bridges defined in the host machine, virtual network interfaces which the machines associate to using the bridged networking mode included in KVM. Finally, to interact with the virtual machines, we take advantage of KVM, which provides a VNC server to all virtualized machines.

Besides the networking elements themselves, we include the concept of a "laboratory", which is a network topology plus all the virtual machines with some configurations. The network topology is represented by an XML file, which details the network elements, including name, type of element (terminal, router, etc.), template the machine is based on, number of network interface cards (which can either be named or referred by a numerical index). The XML file also details the links between the elements, using the network interface cards defined in each element as the two endpoints; it is possible to add shaping properties to each of the links, such as delay, jitter, limited bandwidth and loss, which are added to the bridge that models the link. To add connection to the internet, a special type of link is included, called an "external link", with only one endpoint, which, in turn, connects to a bridge that is also associated to the physical interface of the host machine that has internet connection.

The machines of a laboratory are copied from a template, but can be latter accessed to and modified; they have an explanatory name (a combination of the laboratory name and the name of the element itself) and are kept in the hard drive of the host machine, so the laboratory can be started and paused several times. A possible extension to the platform is the inclusion of start-up scripts with the machines configuration, which would make keeping the machines unnecessary, as each time one starts the laboratory new machines would be created from a template, and then configured as required.

5 Conclusions

We present the problem of hands-on experience in networking teaching, particularly related to the bridge between theory and practice, and then briefly describe why current solutions do not suffice for all ends. Then, we present some of the current alternatives, further detailing their characteristics and why they are discarded; exploring this alternatives has the double purpose of focusing our objectives and serve as inspiration in the design and implementation of our virtual laboratory. We present a summary of the steps taken during development, detailing the tools used, how we choose to model the problem and some details of the implementation, including a summary of the most relevant concepts require understanding the use of the virtual laboratory platform.

6 Future work

In the future, we have three concrete objectives:

- Design a number of pre-made laboratories with different teaching objectives in mind. We think

Netkit's approach to laboratories is interesting, and expect to follow similar guidelines.

- Test the virtual laboratory, using the previous laboratory experiences, in a networking course, and continue refining it depending on the student's feedback. We plan to focus in an undergrad course since the students usually come with little or no networking background; hence we theorize that the students would most benefit from practical laboratory experience when compared with graduate students.

- Design and implement a GUI for the creation, administration and interaction with the laboratories, since, as commented previously, for now we rely on VNC to interact with the virtual machines.

References

[1] Loddo J, Saiu L. Marionnet: a virtual network laboratory and simulation tool. First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems. 2008.

[2] Pizzonia M, Rimondini M. Netkit: network emulation for education. Software: Practice and Experience, 46(2), 133-165. 2016.

[3] Huang T, Jeyakumar V, Lantz B, Feamster N, Winstein K, Sivaraman A. Teaching computer networking with mininet. ACM SIGCOMM. 2014.

[4] Peng C, Liu B. Application of GNS3 at Computer Network Teaching. Theory Research, 20, 136. 2016.

[5] Anderson B, Joines A, Daniels T. Xen worlds: leveraging virtualization in distance education. ACM SIGCSE Bulletin (Vol. 41, No. 3, pp. 293-297). 2009.

[6] Autefage V, Magoni D. Network emulator: a network virtualization testbed for overlay experimentations. Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on (pp. 266-270). 2012.

[7] Wannous M, Nakano H. NVLab, a networking virtual web-based laboratory that implements virtualization and virtual network computing technologies. IEEE Transactions on Learning Technologies, 3(2), 129-138. 2010.

[8] Rendec R, Nicu I, Purdila O. Linux multilayer switching with LiSA. Proceedings of the 5th RoEduNet IEEE International Conference. 2006.