Practical activities in network courses for MOOCs, SPOCs and eLearning with Marionnet

Camille Coti†‡ Jean-Vincent Loddo†‡ Emmanuel Viennet†♮
camille.coti@univ-paris13.fr jean-vincent.loddo@univ-paris13.fr emmanuel.viennet@univ-paris13.fr
† : *IUT de Villetaneuse, Université Paris 13, Sorbonne Paris Cité*
‡ : *LIPN, CNRS UMR 7030, Institut Galilée*
♮ : *L2TI, Institut Galilée, Université Paris 13, Sorbonne Paris Cité*

***Abstract*—In this paper, we present Marionnet, a virtual network laboratory, and how it can be used in distance education (MOOC, SPOC and eLearning classes) to implement practical activities in network classes that should normally require students to have access to specific hardware. Marionnet provides virtual network equipment such as routers, switches, computers and cables, and allows users to design a whole network on a single computer. The hardware in Marionnet is virtualized and can therefore be configured like real devices, making the practical activities that are using it very realistic.**

I. Introduction

Technical learning is generally two-fold: theory, that presents the new notions, and practical activities, where skills are trained. Distance education (MOOC, SPOC and eLearning classes) benefits from a very large variety of formats to deliver theoretical contents: videos, interactive animations, text... However, some practical activities require specific hardware that may not be reasonably owned by students neither by university structures.

In this paper, we are focusing on computer network courses. Practical activity include installing, configuring and exploiting computer networks that feature several computers, routers, switches and a potentially large number of cables. Students that follow online courses cannot reasonably be obliged to own all this hardware, whereas laboratories and practical activities are essential for such technical classes.

Marionnet is a virtual network laboratory that emulates physical networks of computers and devices such as cables, hubs, switches and routers. It features an intuitive graphical user interface, making it really easy to use without requiring any specific training. It accurately reproduces the behavior of a real network, and gives the user access to each device's terminal in order to allow him/her to configure the said device.

Marionnet uses lightweight components to emulate each device of the network, such as specifically patched User-Mode Linux virtual machines[3] and the VDE communication layer[4]. Hence, a Marionnet virtual network as a small memory footprint, so that non-trivial networks can be emulated on reasonably sized, not-so-recent computers without any memory usage issue.

978-1-4799-1756-3/15/$31.00

Moreover, the networks built with Marionnet and the configuration of the machines and devices can be saved for further usage or to be transmitted to an instructor. A network and its configuration is called a *project*. A project is saved in a single file which is actually made of two parts: a representation of the network, and the state of the machines. The state of the machine is actually the modifications that were made from the initial state of its filesystem; this is a *copy-on-write* file, which takes typically a few megabytes of disk space.

As a consequence, Marionnet files can also be exchanged between students and instructors, which is a really interesting feature for eLearning courses. Instructors can send examples or partially configured networks to students, and students can send networks made as assignments or questions to their instructors.

In this paper, we are presenting Marionnet, a virtual network laboratory, and how it can be used in distance education to implement practical activities in network classes that should normally require students to have access to specific hardware. Marionnet provides virtual network equipment such as routers, switches, computers and cables, and allows users to design a whole network on a single computer. The hardware in Marionnet is virtualized and can therefore be configured like real devices, making the practical activities that are using it very realistic.

Section II presents the Marionnet virtual laboratory environment itself. Section III presents how it can be used to allow lab activities in a context of distant learning. Section IV presents how it can be used in the context of a French technical curriculum in particular.

II. A quick overview of Marionnet

Marionnet is graphical application based on GTK+ ([5]) that offers the usual project-oriented metaphor: in order to start using the application the user can create a new empty project or open an existing one. In both cases a dialog window pops up, asking to choose a file name. Once a project is created or opened the user can freely navigate through the functionalities offered by the interface: the *device palette*, the graphical network representation (henceforth the *network*

graph) and the panes providing advanced functionalities (*Interfaces*, *Anomalies*, *Filesystem history*) become active.

A. Hardware devices palette

Network devices can be created, modified and controlled from the device palette within the *Hardware* pane. The user is freed from the burden of physically placing device icons in the network graph two-dimensional space: placement is automatic, but several parameters (for example the length of edges and the size of icons) can be tuned if needed. The network graph is automatically updated at each device state change (such as startup, pause or resume) to reflect the current state.

The device palette offers two kinds of functionalities:

1) virtual network *editing*, including *definition*, *modification* and *removal* of individual devices.
2) virtual network *control* features: each device can be *started-up*, *paused*, *resumed*, *shutdown* and *powered-off*.

We briefly review the eight types of virtual network components which are currently provided.

1) Virtual computer: The *virtual computer* device represents a computer running a GNU/Linux operating system. Just like a physical computer a virtual computer can be *off* or *running*. As an "extension" to what it is possible in a physical network we also provide the possibility of *pausing* computers: in the paused state computers do not react to incoming messages. Pausing allows to experiment with dynamic routing protocols, making a machine temporarily unreachable. The user can set some machine specific parameters, as the *amount of RAM* reserved by the host system to the this guest system, the *number of ethernet cards* (1 by default), the particular *GNU/Linux distribution* (chosen among the ones provided for guest systems by the Marionnet installation), and possibly a *variant*. A *variant* represents a modification to the filesystem of the selected distribution[1]. The user can also select a particular guest *kernel*, chosen from several versions of Linux compiled with different features enabled.

2) Virtual hub: A *hub* is a very simple electronic device reproducing the signals it receives from one of its port into all the other ports. All the network nodes connected to a hub belong to the same Ethernet collision domain. Despite nowadays being mostly disregarded as obsolete variants of switches, hubs are often convenient for intercepting and analyzing network traffic: this can be easily realized by running a *sniffer* application such as *wireshark* or *tcpdump* on a virtual computer directly connected to a hub. The user can choose the *number of ethernet ports* (4 by default). This kind of devices is simulated with the VDE technology (see [4]).

3) Virtual switch: An Ethernet *switch* also allows to relay several Ethernet frames through a network, but differently from a hub it outputs data only to the intended receiving node. Like the virtual hub, the user can select the *number of ethernet ports* (4 by default) and the simulation is again implemented using VDE.

4) Virtual router: An IP *router* is a device directing packets from a local network to another local network. The main purpose of a router is to find the next node of a network through which a packet should be sent to reach its final destination in the minimum time. It is worth to emphasize that, differently from hubs and switches which operate at the *link* layer, routers work at the *network layer*, and their behavior is considerably more complex. *Routing tables* can be set either *statically* or *dynamically*. Router interfaces can be configured from the *Interfaces* pane or, once it is started, accessing it with the *telnet* protocol. In this case, the user access the router through the port 0 pre-configured to a known IP number (by default $192.168.0.254/24$). The other parameters are close to the ones for hubs and switches. Virtual routers are implemented with the *Quagga* software (see [6]) running on a UML virtual machine. Quagga allows both static and dynamic routing (in the latest case supporting different protocols such as RIP, OSF, BGP and ISIS).

5) Virtual cable: Ethernet cables allow to physically connect nodes in the network. Marionnet simulates the most common sort of cables deployed today, twisted pair with "RJ45" connectors, but abstracting over such low-level details (which are typically not relevant at the high-level where the user works). For pedagogical reasons (perhaps pedantic in 2015), the distinction between "straight" and crossover cables is relevant: when an "incorrect" cable is used, for example a crossover cable to connect a computer with a switch, Marionnet simply does not transmit any frames. The control actions provided by the devices palette are limited to *connect* and *disconnect* in the case of cables. Cables are always connected by default, but they can be disconnected and reconnected at will by the user, as it is common while making tests on the network at several different levels. Cables are represented in the network graph as *solid* lines when connected, and as *dashed* lines when disconnected.

6) Virtual Ethernet "cloud": An *cloud* represents an Ethernet (level 2) network composed of hubs, switches and cables, with exactly two endpoints an unspecified internal structure. The only externally observable effects of a cloud consist in delays and other anomalies in the relaying of frames from one endpoint to the other. This "device" is particularly useful for the simulation of dynamic routing. Anomalies as delays can be set from the *Defects* pane with a very fine level of detail.

7) Real world access: Using the components presented up to this moment it is possible to build a virtual network made of computers, hubs, switches, routers and clouds connected by "straight" and crossover Ethernet cables. Such a virtual network is a possibly interesting and useful but completely *closed* system, isolated by the outside world. For this reason, there are also two *real world access* components, the *gateway* and the *bridge*. The bridge represents a "female Ethernet wall socket", opening a breach in this apparent closure: when connected to an external socket other compoenents can access

[1]The *COW* (*Copy On Write*) technology supported by UML allows a very efficient implementation of this feature, involving a single *sparse* file containing only the blocks which are different from the unmodified distribution. A typical COW file takes only a few megabytes of disk space on the host.

the *same* (non-virtual) network to which the *host belongs* (supposedly Internet). The gateway has instead an associated IP number and simply acts as a router to the host network. The external access provide several useful opportunities as connecting virtual computers to the Internet and easily installing additional software on virtual machines, for example using `apt-get install` on a *debian* distribution or downloading and compiling sources. The implementation of the bridge component depends on the *bridging* functionality in Linux, while the gateway is supported once again by VDE.

B. Network graph

When any virtual device changes state or a Marionnet project is opened the *network graph* image is regenerated to reflect the current situation. The configuration parameters settable by the user from the palette next to the graph image in the *Hardware* pane allow the user to tune the node icons' *size*, to randomly rearranges nodes or to resets the nodes arrangement to their initial state. Concerning edges, the user can regenerate the image so that its main spine is drawn horizontally or vertically, he can set the *minimum edge length* or swap the ends of an edge in the image. A slider allows to set the *distance* between a label and the icon representing the node it describes. A distinct slider allows to set the *size of the canvas* containing the whole image.

C. Advanced usage

Marionnet provides several advanced functionalities through three additional GUI panes.

1) Interfaces: this pane allows the user to configure the network interfaces of virtual computers and routers by setting parameters like the MAC address, the IP address and gateway. Although all these parameters can also be set after startup by logging in the virtual computer or router and invoking `ifconfig`, this interface provides a conventient shortcut.

2) Defects: with this pane the user can introduce some artificial "faulty" behaviors like frame loss, frame duplication, flipped bits and trasmission delay. Defects can be set up with the granularity of the single *electric line*, i.e. the *direction* (*in-to-out* or *out-to-in* for ports and *left-to-right* or *right-to-left* for cables). For each direction of each cable or port of each device the user can individually set any defect, by entering a probability or, in the case of delays, a time in milliseconds. Note also that defects settings can be updated "hot", i.e. while the network is running: the behavior is immediately affected.

3) Filesystem history: For each virtual computer or router, a complete history of the disk states is available: each state is saved just before startup. A machine or router can be started up in the most recent state (which is the default behavior), or in any previously saved state. This allows users to freely experiment with potentially "dangerous" filesystem modifications, as each change is reversible. For each machine or router the *filesystem history* displays a tree structure keeping track of the "parent-child" derivation relation of states. States can also be *deleted* or exported as *variants*, to be used for new machines or routers in the same or even in different projects.

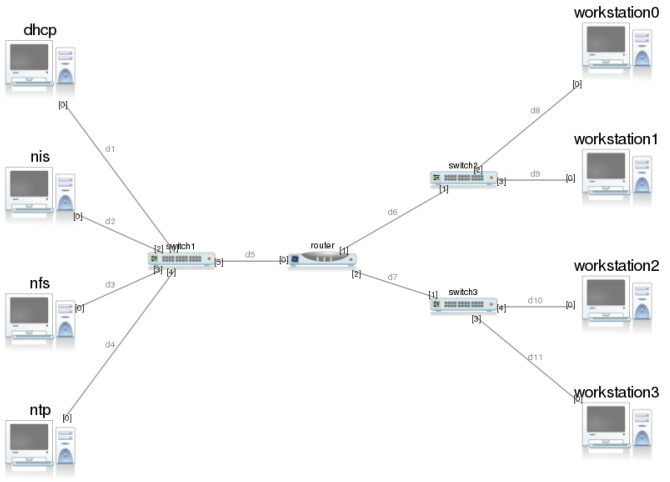

Figure 1. A Marionnet network for a lab project on network servers and clients configuration.

D. File format

The Marionnet project file format is a `tar` archive containing some OCaml marshalled objects and UML *cow* files. The format has been very carefully designed to be back- *and forward*-compatible: newer versions of Marionnet can read project saved by older versions and vice-versa: when Marionnet finds some information which it doesn't "understand", the system simply ignores it. If instead some needed field is lacking then a default value is generated. We hope this to become a conventional exchange format for people who desidred to share projects and "prepackaged" networks.

III. Marionnet as a tool for virtual lab activities

As mentioned in the introduction, practical activities are very important for technical learning. In this section, we detail how Marionnet can make it possible in the context of distant learning and how it can help leading students being more and more autonomous.

Configuring a complete, working network requires a large set of skills, ranging from wiring to service configuration and route settings. Students acquire these skills one by one and therefore, practical activities must focus on each one of them. Asking students to install and configure a full network from scratch at the beginning of their training would be too hard and would put them in front of challenges and waste time on issues that are not the core focus of a given class.

On the other hand, teachers can provide students with a pre-configured network and ask them to work on this partly-configured environment. As a consequence, a practical activity here focuses on one topic in particular, and students do not need to spend time and risk to stumble or even fail and become discouraged by configuration steps that would not be in the scope of the notions they are working on.

A. Activity: configure network services

One example of practical activity would consist in configuring network services, such as DHCP, NTP, NIS, NFS...

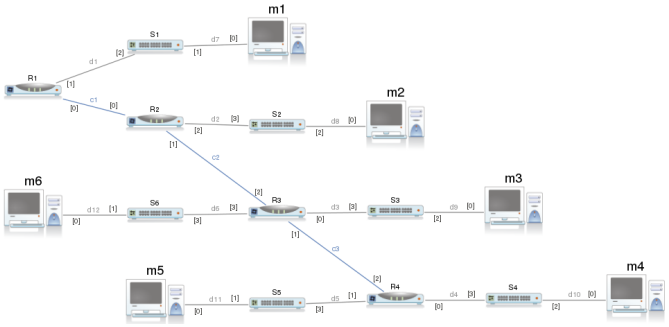

Figure 2. A Marionnet network for a lab project on the configuration of 6 subnetworks.

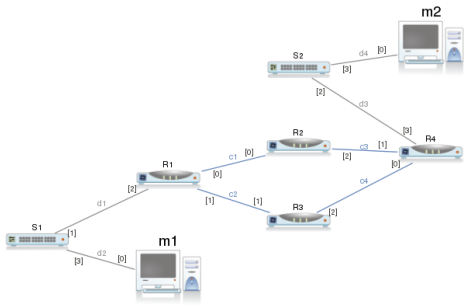

Figure 3. A Marionnet network for a lab project on the configuration of a network backbone between two subnetworks.

The focus of this lab is neither on the installation nor on the configuration of the network, but on the configuration of services running on machines of the network. Therefore, it is not necessary to ask the students to wire and configure the network.

For instance, we can consider a lab project in which students must configure various network services and client machines on three networks (one dedicated to the servers, the other ones for clients). The graphical display presented by Marionnet is depicted in figure 1.

The students can be provided with a `.mar` file (section II-D) that contains the pre-configured network (configuration of the virtual hardware components, wiring) and network interfaces for the router and the servers. As a consequence, they can start the network from a working configuration and focus on the core topic of the lab project. The file for this particular network weights about 10 kB. It can therefore be reasonably uploaded by the instructor on a server and downloaded by the students from their home Internet access.

B. Activity: network configuration

Marionnet is also used for labs on configuring the network elements themselves. In this situation, the instructor can provide the virtual hardware configuration and ask the students to configure the network interfaces of the elements.

For instance, students can work on subnetworks: a network IP address is divided into several logical networks, and one or several routers link them together and forwards network packets between subnetworks. Marionnet uses virtual routers that can be configured like real-life ones (section II-A4). As a consequence, students can work on non-trivial networks made of several subnetworks and several routers.

Figure 2 is the graphical display presented by Marionnet for a network made of six subnetworks and four routers. Each subnetwork has its own switch and contains one machine. Routers are linked together in a chain topology by point-to-point connections using cross cables.

Using the graphical representation, students can have a concrete view of the network topology and make a preliminary reflection exercise on the path followed by packets transmitted between any two machines of the network. Then they can configure the network interfaces of the hardware components (routers and machines) and their routing tables.

In a similar way as with the previous activity, a `.mar` file can be provided with all the hardware components of the network wired together, so that the students can focus on the configuration itself.

It can also be noted that this lab project involves four routers, six switches and six machines. Requiring students to have this equipment at home is hardly possible, especially in a context of distant learning. This equipment can be found in most network lab classroom, but not in many homes. As a consequence, the virtualization feature of Marionnet is exploited here to make this non-trivially sized network possible on a single personal computer.

This idea can be followed to work on various network topologies and configuration methods. For instance, figure 3 depicts a network with a backbone infrastructure made of four routers connected in a ring topology.

The students can focus on the configuration of the backbone itself. This topology is particularly interesting for a lab with Marionnet because dynamic modifications can be made at execution-time. For instance, virtual network cables can be hot plugged and unplugged. Students can observe the effects of unexpected hardware disconnections.

An example of lab activity using these features is dynamic routing configuration, using protocols such as RIP, OSPF or BGP. Students can start only three routers, forming a chain between the two machines. They must configure a dynamic routing protocol on the routers and observe the formation of their routing tables. Then they turn on the fourth router, configure it and observe the modifications on the routing tables and the paths followed by packets that are sent between the two machines. Last, they can unplug a cable and observe the modifications that follow.

This lab focuses on the configuration of the equipment that form a network that has a given topology. The topology can be provided to the students in a `.mar` file in order to make them focus on how they will make it work.

C. Complete configuration

If the students are not provided with any file to start from, they need to configure their Marionnet network from scratch.

They must select the appropriate hardware, wire the devices with each other using the correct cables (cross or straight), and proceed with their configuration.

This hardware installation is quite realistic in a sense that, for instance, the port a cable is plugged in must be selected from a list of existing, available ones. Moreover, if the wrong cable type is selected (cross instead of straight or vice versa), connectivity will not work.

As a consequence, Marionnet allows students to set up a complete network from scratch without requiring any specific hardware in addition to their personal computer. Here again, this tool is particularly suitable for distant learning courses such as MOOCs or eLearning, since it allows practical activities without requiring any specific lab room.

D. Feedback and evaluation

As presented in section II-D, the whole configuration of the network, including the hardware components, the connectivity between them and a COW image of their state is saved in a `.mar` file.

As a consequence, students can send their `.mar` files to their instructor in order to ask a question or in an valuation purpose. For instance, the instructor can evaluate whether the interfaces are properly configured, if some configuration files on the virtual machines are correct, if the devices are wired correctly (using proper wires)...

Therefore, the instructor can evaluate students on the resulting network and perform practical tests on it, which cannot be done on real hardware if students are remote.

IV. Use in a French IUT curricula

A. Context: french's IUT

The 113 French University Institutes of Technology (IUT — *Institut Universitaire de Technologie*[2]) provide technical education spanning over four semesters of studies (DUT — *Diplôme Universitaire de Technologie*), as well as technical bachelor with two additional semesters (*Licence Professionnelle*). IUTs are a major player in France's superior educational system. IUTs provide technical university education, preparing students to careers in the industry and services. The main diploma is called DUT, *Diplôme Universitaire de Technologie* [7]. They are organised in teaching departments covering 23 different subjects ranging from humanities to sciences.

Designed to train mid-level technical staff in 2 years, IUT programmes also allow graduated students to pursue their studies with a more advanced degree, such as a *Licence Professionnelle* [8].

IUT of Villetaneuse, University Paris 13, has been proposing a *Licence Professionnelle* specialized in the Security of Information Systems and Computer Networks for more than ten years. In order to address the growing demand of firms and adult professional, the IUT is currently working on the design and implementation of a new modality, suitable for distant learning. The targeted audience is either employed professional already working in the field, or unemployed adults with a background in computer science.

1) Focus of the diploma: The objective of the LP Networks and Telecommunications option ASUR is to meet the growing demand from companies in the areas of administration and security of computer networks and telecommunications.

Corporate computer network administrators are now facing major evolutions of communication technologies resulting in important changes in working methods. Among the consequences, the companies are facing:

- Needs for greater skills in the administration of application servers (open source and commercial software, virtual servers, etc.).
- Critical issues related to secure backup and storage of data.
- Securing Internet communications, in the context of growing demand for mobility: encryption, authentication to access company's data.

The graduate is able to understand and master the modern techniques of administration and security in computer networks of companies.

The trades covered are:

- Network Administrator.
- Network Assistant engineer.
- Safety and Quality systems.
- Head of IT.
- Network Architect, Project Manager deployment networks

Table I gives a list of all modules, with associated amount of hours in the face-to-face modality. Classes are organized on approximately 6 months, summing to 550 hours. After that, the students must complete a training internship in a company.

Table I
Modules of LP ASUR in the classical (face-to-face) modality

Code	Module	Hours
M01	UNIX Operating Systems (intro)	24
M02	Introduction to programming (Python)	30
M03	Basic concepts of IT Security	12
M3i	Networking	35
M5u	UNIX Administration	36
M6p	Cryptography	30
M3r	Routing	22
M3a	Network Services	30
M3w	Wifi	30
M7	Protection and Monitoring of Networks	30
M3v6	Introduction to IPv6	12
M4	QoS and VoIP	30
M5a	Windows Administration	36
M6s	Attacks' Techniques	22
M8p	Client/Server Programming	25
M1e	Oral & Written Communication	30
M1a	English	30
M2g	Project Management	18
M2d	Laws and Norms	20
M9	Project	30
Mx	Misc (CISCO, Conferences)	20
	Total	550

[2]http://www.campusfrance.org/en/page/short-programs

2) eLearning training plan: The new proposed training plan is based on a mix of SPOCs (*Small Online Private Courses*) and conventional face-to-face teaching. Face-to-face teaching is required or more efficient for modules centered on human interactions (communication, projects in groups) and for modules requiring practice on real devices (such that IPBX, routers, and so on). This is where Marionnet is really useful: its usage allows us to reduce the volume of face-to-face activities, by allowing the students to train themselves autonomously on network configuration , supervision, and also on system administration (e.g. network services on UNIX servers).

Our goal is to propose 75% of the training online, keeping only 130 hours in face-to-face classes.

3) Organization of a module: In the eLearning modality, each module will be composed of a sequence of activities, like:

- Video lectures.
- Written tutorials;
- Quizz and online evaluation.
- Individual help: mentoring by chat, visioconference or e-mail.
- Group activities: forums, group chats sessions.

V. Conclusion

In this paper, we have presented Marionnet, a virtual lab for practical activities in network classes. Marionnet allows students to set up, install, wire, configure and use computer networks made of a non-trivial number of devices such as computers, routers, switches and hubs, without requiring any hardware in addition to a regular, personal computer.

This property makes it particularly suitable for distant learning, making it possible for remote students to have labs and practical activities without asking them to get the aforementioned network devices. In particular, we give examples of how these labs can be organized and how virtual networks can be shared between instructors and students, for grading purpose or as a starting point for some projects.

We give a specific implementation of these features in a technical program in France called Licence Professionnelle, and how Marionnet and distant learning can be used for an eLearning version of this program.

References

[1] J.-V. Loddo, L. Saiu: *Marionnet: a virtual network laboratory and simulation tool*, 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (Simulation-Works'2008), Marseille, France, 2008.

[2] J.-V. Loddo, L. Saiu, *Status report: marionnet or "how to implement a virtual network laboratory in six months and be happy"*, ACM SIGPLAN Workshop on ML (ML'2007), Freiburg (Germany), 2007.

[3] J. Dike: *User mode linux (Vol. 2)*, Englewood Cliffs: Prentice Hall, ISBN 0-13-186505-6, 2006.

[4] R. Davoli: *VDE: Virtual Distributed Ethernet*, First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, (TRIDENTCOM 2005), pp. 213-220, IEEE, 2005.

[5] Owen Taylor and others: *GTK+ - GNU toolkit for X windows development*, http://www.gtk.org

[6] Kunihiro Ishiguro and others: *Quagga Project*, http://www.quagga.net

[7] Assemblée des directeurs d'IUT (ADIUT) et Union nationale des présidents de conseils d'IUT. Livre blanc sur le système IUT, 2007. http://www.iut-fr.net/files/fck/File/documents/publications/livre_blanc_iut_2007.pdf.

[8] Thierry Malan. Implementing the Bologna process in France. *European Journal of Education*, 39(3):289–297, 2004.