NVLab, a Networking Virtual Web-Based Laboratory that Implements Virtualization and Virtual Network Computing Technologies

Muhammad Wannous, *Student Member*, *IEEE*, and Hiroshi Nakano

Abstract—In various ICT courses, it is necessary to establish a proper space where each learner has access to a set of network devices with which he/she can build and test networks with different structures and components. This will enable the learners to practice working on multiple devices as in the case of real world context, and freely apply modifications to the network structure to solve any issue that may arise with the current scenario. While on-campus facilities are designed to meet this criterion, current Web-based laboratories either have fixed network designs, or do not offer a number of real devices to work on. Taking a step forward toward building a Web-based laboratory, we used open source *Virtualization* and *Virtual Network Computing* technologies in NVLab, a system that offers the learners tools to draw, configure, test, and troubleshoot network designs using real Operating System instances running in the virtual mode on a host machine. The functionality of the system has been experimented by introducing a case study exercise to a group of remote learners as part of *Computer Networks* course. Evaluating the participants' advancement was done by comparing their answers to a level test prior and after completing the exercise. The evaluation showed that the learners achieved better results in the level test after completing the exercise and were able to use the system efficiently.

Index Terms—Computer and information science education, distance learning, virtual network computing, virtualization.

◆

1 INTRODUCTION

IN a previous work [4], we tested the possibility of using *Virtualization* (Xen) [5] and *Virtual Network Computing* (VNC) [6] technologies to build a remotely accessible laboratory system that is suitable for *Computer Networks* courses introducing concepts related to network addressing, routing, traffic flow-controlling, etc. *Virtualization* technology was used to create a network comprising a number of *Virtual Machines* (VM) on a single server as shown in Fig. 1, and in these VMs run real Operating System (OS) instances. With the capabilities modern OSs have, these instances can perform the functions of all network devices such as Routers, IDSs, IPSs, etc. The machines have been set up to enable remote access and learners were able to connect to their consoles through the VNC modules installed on the host machine and on the VMs in order to complete some practical exercises.

Manually creating the VMs and setting them up for remote access involved lot of effort and was a time-consuming process that instructors were requested to carry out whenever the network was to be rearranged to adopt modifications necessary for introducing new concepts. Furthermore, learners were not able to modify the network structure in order to try scenarios different from the ones the instructors have created. And because certain settings

were required for enabling remote access to the VMs, these settings were prohibited from being modified. All these issues brought forth the idea of having a tool for automating the creation of the network, and finding a way to allow learners work on the devices with no restrictions.

1.1 Research Contribution

In this paper, we demonstrate how we used Xen and VNC technologies to build a modular Web-based laboratory (NVLab) suitable for ICT courses that require the individual learner to work on multiple networked devices. Learners will use the tools NVLab includes to produce a network design and then construct it on the server. After this phase is completed, learners will be able to connect to the server and work on their networks. In our laboratory, the VMs need not be preconfigured for remote access, and learners are able to work freely and with full control over them. NVLab also enables the instructors to produce properly/improperly configured network designs and ask the learners to verify/troubleshoot the settings of the devices included in these networks.

The rest of the paper is organized in sections as follows: Section 2 provides brief background information, and Section 3 presents description of related work in the field of ICT Web-based courses. Then, in Section 4, we introduce the core technologies used in NVLab. A description of how the system was implemented is taken up in Section 5. Next, we introduce a case study in Section 6. Finally, we conclude with our results in Section 7.

• *M. Wannous is with the Graduate School of Science and Technology, Kumamoto University, Kurokami 2-39-1, Kumamoto-City 860-8555, Japan. E-mail: muhammad.wannous@ieee.org.*
• *H. Nakano is with the Center for Multimedia and Information Technologies, Kumamoto University, Kurokami 2-39-1, Kumamoto-City 860-8555, Japan. E-mail: nakano@cc.kumamoto-u.ac.jp.*

2 BACKGROUND

As in all engineering curricula, laboratory activities play a major role in ICT courses. These activities not only help
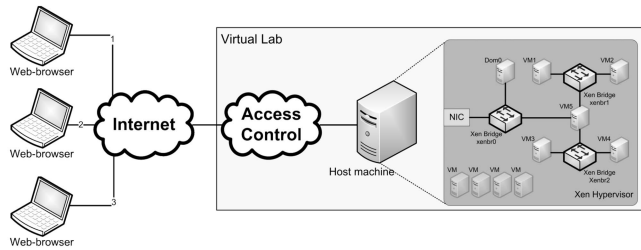
Fig. 1. A network constructed of a number of VMs on one machine.

learners get better understanding of what they are being taught, but also provide them with the problem-solving skills and experience necessary for adopting their knowledge in real-life contexts [1], [2]. Beyond this, laboratory activities became the main components of the course rather than a separate supporting part, and it is now common to find laboratory-based courses delivered at increasing number of universities and training centers [3].

Among ICT courses that require the learner to work on multiple devices is the *Computer Networks* course. These days, computer networks have widened to cover the whole world facilitating communication between individuals, access to various resources, e-commerce, etc. That is a reason why well-trained and qualified staff personnel are required at increasing number of companies not only specialized in installation and maintenance, but also in other fields. It is now common to find Network Experts working at stores, government offices, transport stations, and at every place where information is to be exchanged. Consequently, the need for training centers/laboratories utilizing proper infrastructure necessary for qualifying individuals to be Network Experts arises.

Configuring devices operating at different levels of the OSI standard, maintaining connectivity between these devices on one hand and with the external world on the other hand, traffic and load balancing, and upholding carefully designed security policy are among the network administration tasks. Hence, a laboratory addressing *Computer Networks* issues should contain enough machines of various types, computers, switches, routers, firewalls, IDSs, IPSs, etc., to cover all the above mentioned tasks, among others. Besides, every individual learner should have access to a set of these devices in order to try different possible network scenarios that meet the requirements and pick one of them to be implemented.

While on-campus facilities are designed to meet the previously mentioned criterion, Web-based laboratories are evolving to become equal. This has been reinforced by the introduction of emerging technologies and the increased demand for them to be used within the e-learning environments. Systems implementing emerging *Virtualization* and Remote-Access technologies to deliver *Computer Networks* training have been successfully tested and verified to be adequate for reaching the goal [23], [24].

3   RELATED WORK

Work dealing with building a Web-based laboratory environment is of wide range and is not limited to courses related to ICT. But, the design of the laboratory differs according to the field of application and the availability of resources. In this regard, a number of methods have been investigated.

The first approach suggests equipping the on-campus laboratory facilities with remote access network. In this way, learners can access and work on real devices while monitoring the results of the modifications they do to the system. This arrangement has been adopted in courses of wide range such as: electronics engineering [14], robotics [15], besides ICT courses [16], [17]. A drawback of this method is the requirement to have a suitable preinstalled laboratory facility and more resources for the remote access devices. Moreover, changing the configuration of the laboratory requires the presence of some staff on the laboratory's premises or complex switching arrays [18].

Other researches proposed to use Web-based software simulation tools within the learning environment. Software simulators are rich with animations and variable controls that help learners produce a more meaningful mental model for the introduced concepts. A combination of simulators designed in this way and static materials (text, images, etc.) have proved to be very effective in reaching a better learning performance with medical science students [25]. But, according to a recent research, special care should be given to the complexity of the contents these tools introduce [26]. In other cases, these tools are designed to imitate a hardware device and provide an easy and flexible way of exploring it. In engineering, Web-based laboratories using tools designed in this way are of wide range, [19] in electronics engineering, and [20], [21] in ICT-related courses. In the software simulators, the laboratory structure can be easily altered and the cost is significantly reduced especially when implementing Open Source code programs. However, a disadvantage of such systems is that work is done on a simulated device which puts limitation on what can be tested since simulators are not designed to cover all aspects of the real device.

The introduction of new technologies for implementing and remotely accessing various devices evoked a new way in designing Web-based laboratories especially in the field of *Computer Science*. *Virtualization* technology, for example, is now being used to create several OS instances and run them on one host machine [22]. In parallel, *Remote Desktop*, *Secure Shell*, *Virtual Network Computing* technologies, etc., serve in facilitating access to the laboratory through the Web. Systems implementing these technologies to deliver *Computer Networks* training have been successfully tested and verified to be adequate to for reaching the goal [23], [24]. They offer the learners real instances of the laboratory devices to practice on, and feature flexibility and mobility. However, in most cases, these laboratories served to provide training on a narrow range of computer networks concepts such as *Network Security*, for example.

4   TECHNOLOGIES

4.1   Virtualization

Introduction of software virtualization technology goes back to the sixties of the last century. Since that time, it passed through periods when its importance faded out; however, new fields of application have emerged and the technology is strongly back to the scene [7].

Fig. 2. *Virtual Machine Manager.*

Inserted as a software layer on top of the host machine's hardware, the *Virtual Machine Monitor* (VMM) allows transparently sharing the resources among multiple guest OS instances which run while isolated from each other, but they still can communicate via the different software networking scenarios the VMM offers.

Software vendors introduced a number of modes in which the VMM builds and controls the VM environment [8]. For example, VMs can be run in the *full-virtualization* mode in which they have no access to the host machine's resources; instead, the VMM simulates the functions of all these resources. On the other hand, VMs running in the *paravirtualization* mode can access some hardware parts in the host machines. Differences between these techniques come in terms of overhead imposed by inserting the new layer between the hardware and the guest OSs, and the simplicity of implementation. In this regard, the *full-virtualization* mode is easier to implement but has a less performance indicator; while the *paravirtualization* mode has a better performance figure but requires inserting special modifications in the guest OS, i.e., it is difficult to implement and with no support in legacy OSs.

Open Source Xen [9] is one software VMM (called *Virtualization Hypervisor* in Xen terminology) offering *full-virtualization* and *paravirtualization* techniques. Its free version has been included along with a set of supporting tools in a number of OS distributions, CentOS, for example. In Xen terminology, the guest-OSs are called *Domains* with a special privilege given to *Domain-0* in which runs the hypervisor. This privileged domain will handle the resources provided to the running VMs.

Xen introduces a number of networking scenarios in which the VMs can exchange traffic among each other, on one hand, and with the external world (including *Domain-0*), on the other [10]. These scenarios include: *bridging*, *routing*, and *Virtual Network*. In all these scenarios, a *Virtual Interface* will be bounded to each NIC the VM has (including *Domain-0*) and will be used to pass the traffic to/from this network interface. The pair consisting of the VM's interface and the *Virtual Interface* bounded to it can be thought of as two Ethernet interfaces connected by an internal crossover cable. *Domain-0* will use standard OS mechanisms for bridging, routing, rate limiting, etc., to handle the traffic to/from this *Virtual Interface*.

When a *bridged network* mode is selected, a VM can be connected internally to any software switch, or more than one software switch that Xen creates. The software switches work

```
name = "R"
maxmem = 64
memory = 64
vcpus = 1
builder = "hvm"
kernel = "/usr/lib/xen/boot/hvmloader"
boot = "c"
pae = 1
acpi = 1
apic = 1
on_poweroff = "destroy"
on_reboot = "restart"
on_crash = "restart"
device_model = "/usr/lib64/xen/bin/qemu-dm"
sdl = 0
vnc = 1
vncunused = 1
keymap = "en-us"
disk = ["file:/var/lib/xen/images/R.img,hda,w", ]
vif=["mac=00:16:3e:78:d9:01,
     bridge=xenbr1,type=ioemu" ,
    "mac=00:16:3e:78:d9:02,
     bridge=xenbr2,type=ioemu"  ]
serial = "pty"
```

Fig. 3. *Virtual Machine's* configuration file.

in the *Network* layer (L2) of the OSI model directing traffic based on the MAC address assigned by Xen. *Routed network*, in contrast, creates routes between *Domain-0* and the other Domain(s) in the system allowing traffic to be routed based on the IP address. Another possible networking configuration in Xen is the *Virtual Network*, but it is not standard.

Providing an easy to use graphical user interface (Fig. 2), *Virtual Machine Manager* (or *virt-manager* [11]) is a free open-source support tool for managing the VMs in Xen.

Using this tool, tasks like creating new VMs, modifying the hardware resources attached to them, booting/shutting down VMs, and monitoring their performance and resource utilization can be carried out.

Virt-manager creates for every VM installed on the host machine a configuration file that contains entries of hardware resources assigned to it such as the number of virtual CPUs, the size of memory, storage type and size, network interfaces, etc. By modifying the contents of this file using any simple text editor, it is possible to alter some of the resources assigned to the VM to which this file belongs. Sample configuration file of a VM named "R" is provided in Fig. 3.

In this file, the resources assigned to the VM are:

- One virtual CPU (vcpus = 1).
- 64 MB of memory (memory = 64).
- A regular file on the hard disk to be used as a storage medium

  (disk=["file:/var/lib/xen/images/R.img, hda,w",]).

- Two network interfaces:

  (vif = ["mac=00:16:3e:78:d9:01,
              bridge=xenbr1,type = ioemu",
         "mac = 00:16:3e:78:d9:02,
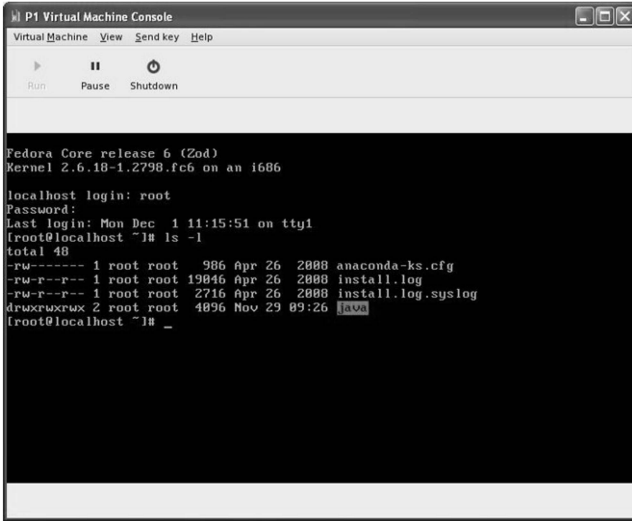   bridge = xenbr2, type=ioemu"]).

Fig. 4. The *Virtual Machine's* console window in virt-manager.

Other information required for the proper handling of the VM's console and booting options are also included in this file.

Virt-manager provides a graphical console window (Fig. 4) as a replacement for the VM's monitor. This console can be brought to view by double clicking on the desired VM's name in *virt-manager* window, then pushing the "Run" button from the console's window. The user will use the console window and the host machine's input devices to boot, pause, resume, shutdown, and interact with the guest OS.

4.2   Virtual Network Computing

VNC technology, originally introduced by "Olivetti and Oracle Research Laboratory, Cambridge Univ." enables users to access centralized resources remotely from widely available devices [12]. Its architecture is based on a server-client model (Fig. 5) where the server is the VNC-Server that resides on the machine hosting the resources to be accessed, and the VNC-Client (called vncviewer) is an ultrathin or generic software.

Underlying VNC is the *Remote Frame Buffer* (RFB) protocol: a simple protocol for remotely accessing a GUI [13]. Using frame buffer resulted in the applicability of this protocol in all OSs implementing windowing such as Windows, Linux, and MAC OS. This protocol defines subprotocols for exchanging the data between the server and the client sides, which are: the *Display*, the *Update*, and the *Input* subprotocols. The *Display* portion of the protocol is
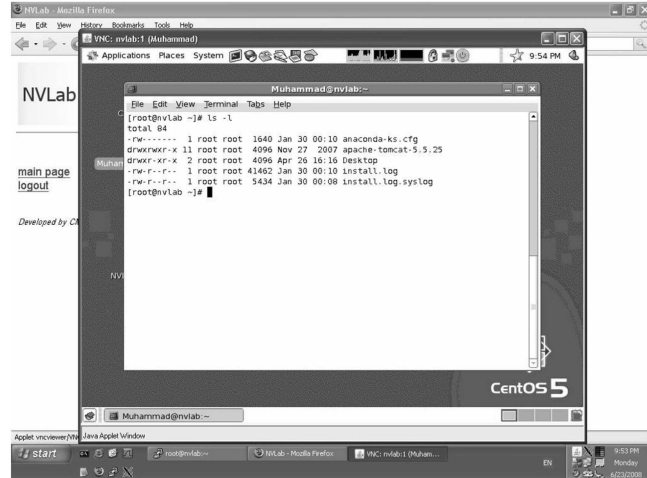

Fig. 6. VNC-viewer applet.

used to update a rectangle area located at (x, y) point in the client machine's frame buffer. The *Update* protocol is client-event-driven, i.e., update data are sent from the server in response to a request from the user. On the other hand, within the *Input* portion of the protocol, the viewer sends input events to the server whenever a keyboard or pointing device event occurs on the client's side while the viewer window is active. The VNC-server will create a display identified by a unique number for each user allowed to connect via VNC and the connection will, by default, be over TCP ports 5800+Display# and 5900+Display # (Display# is interpreted as the display number through which the remote user will login).

It is possible with VNC to simultaneously connect a number of clients to the VNC-server while keeping them stateless, i.e., closing the connection can be done from either side without undesired consequences; still, upon reconnection, the user will be able to complete the session from where it was closed. Furthermore, VNC does not require running the server and the client applications on the same Operating System, i.e., users accessing the server need not be running the same Operating System installed on the server on their machines.

With the introduction of Java as a major programming language for Internet applications, VNC-server was implemented to accept HTTP connections, and VNC-client software (vncviewer) was implemented as a *Java Applet* (Fig. 6). In this way, any Java-capable Web browser can be used as a VNC-viewer.

5   IMPLEMENTATION

The need to supply each learner with a Web-based space offering a set of devices to try and test different network scenarios was the main reason behind introducing NVLab. Further than this, it is required to have the proper tools that enable learners construct and access the networks they want to test. In the next part, we describe how Xen and VNC were implemented, and the different tools provided to the learners in the suggested system.
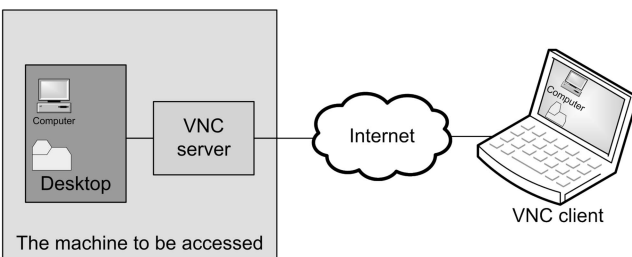

Fig. 5. VNC architecture.

```
#!/bin/sh
dir=$(dirname "$0")
"$dir/network-bridge" "$@" vifnum=0
"$dir/network-bridge" "$@" vifnum=1 netdev=''
"$dir/network-bridge" "$@" vifnum=2 netdev=''
"$dir/network-bridge" "$@" vifnum=3 netdev=''
```

Fig. 7. The script used to create four software switches (xenbr$\{0 \sim 3\}$) in Xen.

5.1 Host Machine Specifications

During test phase, NVLab has been hosted on a rack-mounted server with Pentium D CPU running at 3.0 GHz, 224 GB of SCSI storage, and 8 GB RAM module. The Operating System running on the server was CentOS5 and the Web hosting application was apache tomcat 5.5.25.

5.2 Xen

For ease of installation and less need for high performance, we set up prototype VMs for a Router and PC (the currently supported device-types) machines to run in the *full-virtualization* mode. From these prototypes, all machines in the network diagram will be copied. A VM is assigned one virtual CPU, 64 MB of RAM, and a number of network interfaces depending on its type (one interface for the PC machine and up to three interfaces for the Router machine). This information will be saved in the configuration file of each VM created in the system. As for inter-networking, a *bridged network* configuration has been adopted in NVLab for simplicity. We created a number of software switches in the system using a script similar to the one provided in Fig. 7 and made a number of them available for each learner to use. The user will not have the privilege to modify the settings of these switches, but will be able to connect other devices to them. This will not have any impact on the exercises since Switches can normally be operated with their factory settings. For security reasons, no external traffic was allowed in the network the learners create.

5.3 *VNC*

The VNC-server is configured to accept incoming HTTP connections initiated from Web browsers over VNC default TCP ports and provide the learner with an 800×600-size applet showing the host machine's desktop upon successful
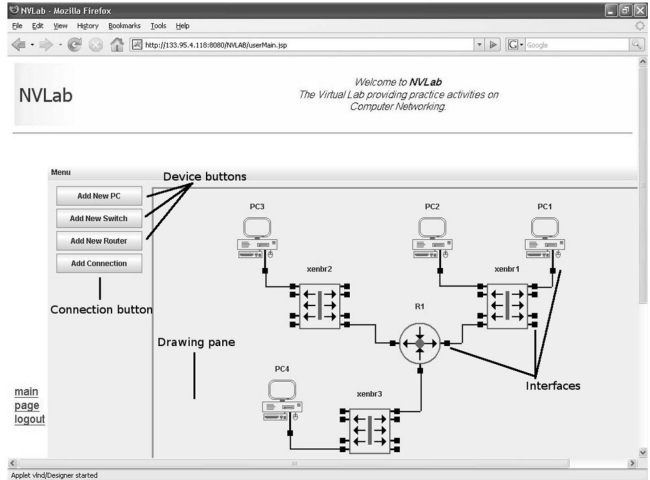

Fig. 9. *Designer* applet.

verification of the login information as shown in Fig. 6. During test stage, one user account has been created on the host machines for users connecting via VNC-server.

5.4 NVLab

NVLab is realized as a Web-based environment offering tools for: producing a small-scale computer network design, creating the network by instantiating one VM for every router and PC used, and configuring the different VMs by connecting to the machine hosting them. A diagram of the system structure is shown in Fig. 8.

In this figure, the new tool called *Designer* is a simple *Java Applet* with a GUI interface that we designed to allow users to draw a network. Its interface (Fig. 9) includes a number of buttons for adding devices to the drawing and inter-connecting them. The maximum number of devices that can be used depends on the instructor's preferences and can be accordingly changed by the system programmer. For time being, this number is set to 16 which we consider to be suitable for small-scale networks.

Designer will ensure that connections made in the network diagram adhere to a rule imposed by the Xen-*bridged networking* model which allows connections to be made between one VM's interface on one end and an interface on a switch on the other end only.

Through the menu *Designer* has (Fig. 10), the learners will be able to:

Fig. 8. NVLab structure.

Fig. 10. Designer's menu options.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE diagram SYSTEM "resources/VLND.DTD">
<diagram>
 <devices>
  <device type="SW">
   <net_name>xenbr1</net_name>
   <location>
    <x>300</x>
    <y>120</y>
   </location>
  </device>
  <device type="PC">
   <net_name>PC2</net_name>
   <location>
    <x>220</x>
    <y>20</y>
   </location>
  </device>
  <device type="PC">
   <net_name>PC1</net_name>
   <location>
    <x>360</x>
    <y>20</y>
   </location>
  </device>
 </devices>
 <connections>
  <connection>
   <start name="PC1" type="PC">eth0</start>
   <end name="xenbr1" type="SW">eth4</end>
   <segments>
    <segment dx="423" dy="160" sx="423" sy="136"/>
    <segment dx="393" dy="160" sx="423" sy="160"/>
   </segments>
  </connection>
  <connection>
   <start name="PC2" type="PC">eth0</start>
   <end name="xenbr1" type="SW">eth0</end>
   <segments>
    <segment dx="282" dy="159" sx="282" sy="135"/>
    <segment dx="307" dy="159" sx="282" sy="159"/>
   </segments>
  </connection>
 </connections>
</diagram>
```


PC2
PC1
xenbr1

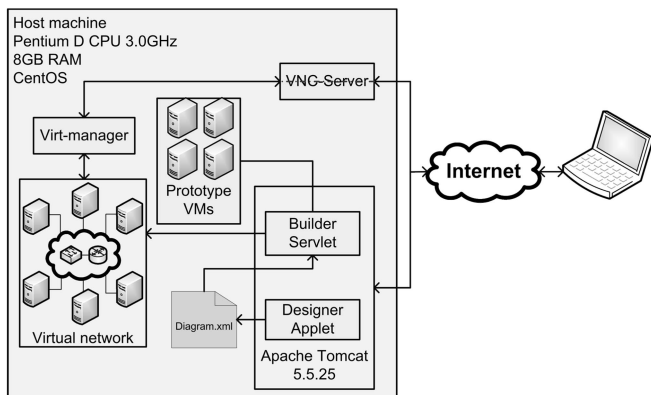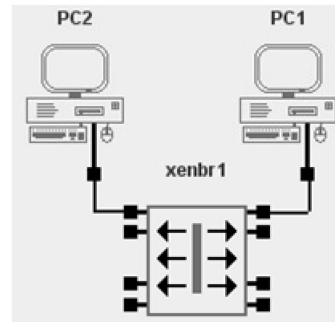

Fig. 11. The XML document created for a simple network diagram.

- Send the network diagram's information to be saved on the server side as an XML document. (*Send to server.*)
- Invoke the so-called Builder tool, which will use the XML document, for building the network on the server. (*Create the diagram on the server.*)
- Clear the drawing (*Discard and Open New Design*) and start over from the beginning.
- Print the diagram on the learner's local printer. (*Print Current Design on Local Printer.*)

When the learner has finished drawing the network diagram and wants to send its information to be saved on the server, he/she can click on *Send to Server* option from the menu. This will instruct *Designer* to create an XML document containing all information related to the design and initiate an HTTP connection to the remote side over which the document will be sent. The XML document contains information about the devices included in the diagram and the interconnections between them, and is structured as shown in Fig. 11. This figure has been generated for the simple network drawing shown to the right of the XML document.

Selecting the second option in *Designer*'s menu (*create the diagram on the server*) invokes the new tool *Builder* which we designed to take care of creating the devices included in learner's network on the server. *Builder* will try to load the network diagram XML file the learner has already saved on the server, confirm the success of parsing its content, and then start creating the network by instantiating copies of the prototype VMs and connecting them to the designated software switches.

For a VM to be ready, two files need to be prepared: the storage image file and the configuration file. The storage image file will be exported as a hard disk where the guest OS instance resides. This file will be copied from a default location on the server's file system to the directory where Xen stores all VMs' image files. Copying this file takes around 4-5 minutes depending on the prototype machine image file's size which is determined by the instructor. The configuration file will (as mentioned before) include a description of all resources assigned to the VM to be used by Xen when creating it, and will be generated by Builder. Generating this file does not consume much time since its size is usually within hundreds of bytes.
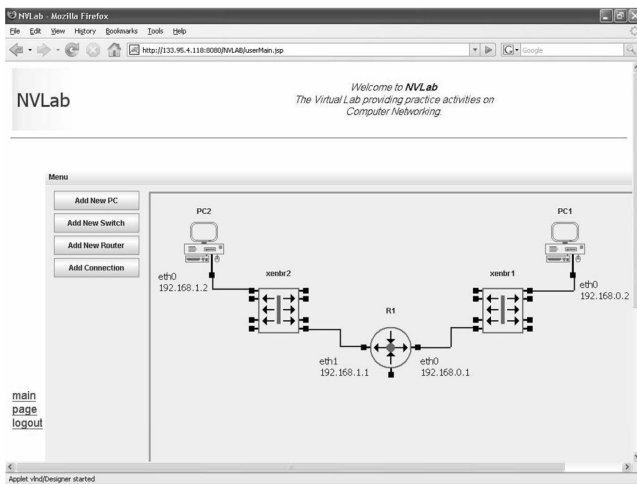
Fig. 12. A case study network design.

6 CASE STUDY

To test the basic functionality of the system and how useful it can be, a group of 15 learners with different levels of IT backgrounds were asked to try an exercise that is part of *Computer Networks* course from remote locations.

All of the participants were university students who either had previously studied or attended *Computer Networks* classes at the time when the exercise was proposed to them. Two individuals had previously been certified as *Cisco-Certified Network Associate*.

The exercise is based on a sample network comprising two PCs (PC1 and PC2), two software switches (xenbr1 and xenbr2), and a router (R1), as shown in Fig. 12. This simple network has two broadcast domains and one router device that, when properly configured, can pass traffic between them. It can be used to develop the learner's skills in determining and configuring IP addresses in a network, enabling traffic forwarding between devices connected to different switches, and troubleshooting connectivity problems.

Two prototype VMs for a PC and a Router were created in the system, and we selected Linux (Fedora Core 6) to run as guest OS in both of them. A minimum number of packages were installed in order to shorten the VM's creation time. The X window system was not among these packages and the learners were requested to use the command-line prompt to perform each step in the exercise. Furthermore, SELinux and the iptables services were disabled since the task is not designed to provide training on network security concepts and to allow learners to test whatever they may see necessary without having to consider the firewall rules. Learners were able to log into their machines using the "root" account information and this enabled them to have full control over the devices.

To help the learners get used to the system before taking the exercise, we prepared a tutorial with sample movies to illustrate how the different tools work. And because most of the participants had little experience using Linux, we included the commands mostly encountered in networked environments in the tutorial.

The importance of providing *static* contents (a text containing explanation of the concepts to be introduced)

along with the system tools has been indicated in recent researches [25]. With such an arrangement, learners were able to get more out of what they are presented. Accordingly, we handed a paper that included simple explanation of the targeted skills to the learners to read before or while practicing.

Due to the availability of one server for testing phase, access to the system has been granted to one user at a time, and a period of two hours for each user to complete the task was scheduled. Users were also encouraged to try any idea related to this task that comes into their minds and even try to use more devices if needed.

Evaluating the progress made by taking the exercise was done by introducing one level-test to all participating learners and asking them to complete it prior to starting the task and later after finishing it. This level-test included multiple-choice, yes/no, and fill-in-blank questions on the various concepts included in the exercise. Examples of these questions are shown in Table 1.

Furthermore, learners were requested to fill a survey that helps us identify any weak points or improperness of the task/laboratory design. The survey was designed to get general and detailed feedback from learners about what they think of the different tools in NVLab. While the general questions were put to help compare the learners' performance in a similar laboratory environment with the one in NVLab, the detailed fields of the survey queried about the technical aspects of the tools and any difficulties that learners faced using them.

6.1 Task

To cover concepts of basic IP planning, connectivity check, and simple routing in Linux machines, a sample task was proposed for the students to complete. Steps to be followed are:

- Use the two C class IP subnets 192.168.0.x and 192.168.1.x in this task.
- Assign IP addresses from one subnet to the interfaces connected to xenbr1 (eth0 from the router and eth0 from PC1).
- Assign IP addresses from the other subnet to the interfaces connected to xenbr2 (eth1 from the router and eth0 from PC2).
- Check the connectivity between each PC and the router.
- Configure the router to allow IP forwarding and check the connectivity between PC1 and PC2.

6.2 Exercise Result

All participating learners were able to complete the task successfully (snapshots of one learner's machine is provided in Fig. 13). Comparison of the level-test results for the learners before and after taking the exercise gave two positive indicators as shown in Table 2.

This table shows that the increase in the correct answers was 8 percent and that this was also accompanied by a decrease of 5 percent in the number of wrong answers. The difference between the two percentages arose because learners could select right and wrong answers in the multichoice questions.

Participants also provided their feedback and comments on the survey prepared for this experiment. Analysis of the

TABLE 1
Sample Questions from the Level Test

Question type	Question text
multiple-choice	Which of the following groups defines two IP addresses that belong to the same subnet? ☐ 192.168.0.1/25, 192.168.0.125/25 ☐ 192.168.0.1/25, 192.168.0.126/25 ☐ 192.168.0.1/25, 192.168.0.129/25 ☐ 192.168.0.1/25, 192.168.0.130/25
yes/no	In a network comprising three PCs (PC1, PC2, and PC3) that are connected to one layer-2 switch and properly setup with suitable IPs. The administrator on PC1 wishes to check the connectivity with PC3, and therefore uses the command "ping" with the proper arguments to send some packets to PC3 and then receives a reply indicating success. Does PC2 receive a copy of the exchanged packets? ☐ Yes ☐ No
fill-in-blank	PC1 has the following IP address: 192.168.11.25/25. What are the network and the broadcast IP addresses on the network PC1 belongs to? Network: Broadcast:

feedback information contained in the survey which was handed to the learners provided the results shown in Fig. 14. These can be summarized as follows:

1. On a scale between 0 and 9, where 0 indicates "very bad" and 9 indicates "excellent":

   • *Designer* tool rating was in the range of 7-9. Detailed feedback related to this tool showed that some learners found difficulty in drawing the connection between two devices.This is due to the fact that *Designer* is not intended to perform high-level drawing functions.
   • *Builder* rating (6-9) was slightly lower than *Designer*. In their comments, learners mentioned that they found the time *Builder* takes for creating the network on the server to be long.

   • The connection to the server was granted rating in the range of 7-9, but learners commented that the size of the VNC applet window showing the host server's desktop was relatively small which caused inconveniency while taking the exercise.
   • The size of the VMs' window was also small but had less improper impact on the learners than the impact of the small size of the server's VNC window.

2. On a scale between 0 and 9, where 0 indicates "very difficult" and 9 indicates "extremely easy," most learners found the exercise somewhere between easy and difficult since they were exposed to similar exercise as reflected in the results also (Fig. 14). Only 20 percent of the sample group found the exercise very easy because they had previous networking experience and further study of advanced networking concepts. Working on Linux machines rather than MS Windows machines made the exercise slightly difficult for most learners.

3. In the exercise text, suggestions to perform more tests on the network, try more than one plan for IP planning, and even try a different network were

Fig. 13. Completed as appeared on the learner side.

TABLE 2
Number of Correct and Wrong Answers Before
and After the Exercise for All Learners

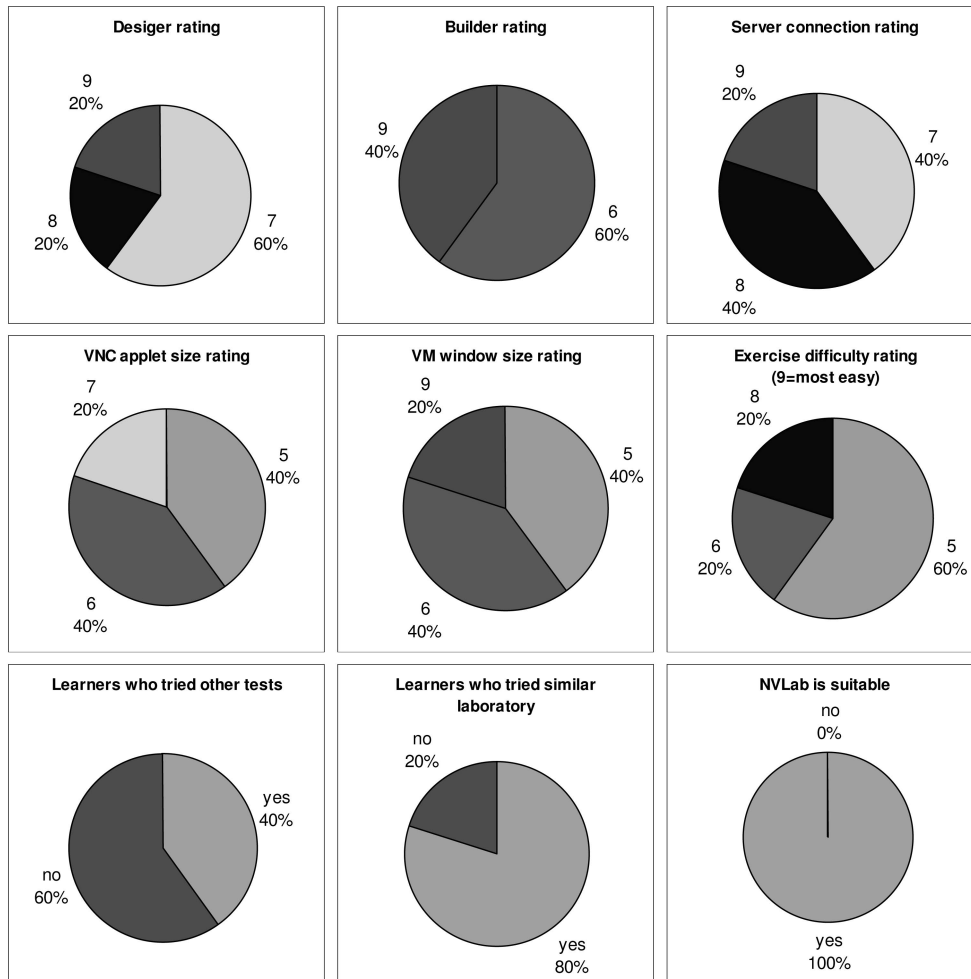|	Before the exercise	After the exercise
Correct answers	582/855 (68%)	653/855 (76%)
Wrong answers	80/555 (14%)	50/555 (9%)

Fig. 14. Feedback analysis results.

included. While 60 percent of the learners completed the exercise as handed, 40 percent of them went beyond the exercise text and tried some of the suggestions and other advanced scenarios for the network.

4. All learners who tried NVLab mentioned that they were able to complete the exercise successfully and that they find the system suitable for practicing *Computer Networks* and that they believe that it will be possible to take more complex exercises within it.

7 CONCLUSION

In this paper, we presented our approach toward building a Web-based *Computer Networks* laboratory (NVLab) where students can freely practice on designing, configuring, and troubleshooting a network scenario.

To build NVLab, we used *Virtualization* and VNC as core technologies. *Virtualization* enables creating a number of networked VMs on one server, and VNC facilitates remote access to the server hosting these VMs in order to configure them and test their work.

The laboratory introduces two new tools for the learners: *Designer* and *Builder.* The learners use *Designer* for producing a network drawing and sending its information as an XML document to be saved on the server side. Then, they will invoke *Builder* which is responsible for creating *Virtual Machines* that correspond to those included in the XML document. In the next step, learners will be able to use their Web browsers to connect to the VNC module installed on the host server, start *virt-manager* (the application through which they can control the VMs), and then start configuring and testing their network.

Responding to our request, a selected group of students with different levels of networking background tried a sample task covering concepts on IP subnetting, connectivity check, and basic routing within NVLab from remote locations.

All the participants could complete the exercise successfully, and the feedback received from them showed that they were satisfied with the way they worked on their designs. The feedback also indicated some notes on the laboratory interface and we will consider working on them in newer versions of the laboratory system.

8 FUTURE WORK

Having verified the appropriateness of the system to be used in computer networks course, we are working on:

1. Modifying the laboratory structure to accept more than one user logging to the system.

2. Improving Builder to make the creation of the Virtual Machines faster (currently one VM takes 4-5 minutes to be created).

3. Implementing the laboratory in real environment to test it quantitatively.

REFERENCES

[1] N.I. Sarkar, "Teaching Computer Networking Fundamentals Using Practical Laboratory Exercises," *IEEE Trans. Education,* vol. 49, no. 2, pp. 285-291, May 2006.

[2] N. Linge and D. Parsons, "Problem-Based Learning as an Effective Tool for Teaching Computer Network Design," *IEEE Trans. Education,* vol. 49, no. 1, pp. 5-10, Feb. 2006.

[3] P. Mateti, "A Laboratory-Based Course on Internet Security," *Proc. 34th ACM SIGCSE Technical Symp. Computer Science Education,* pp. 252-256, 2003.

[4] M. Wannous, H. Nakano, T. Kita, and K. Sugitani, "A Core System for a Web-Based Virtual Computer Laboratory," *Proc. Eighth Int'l Conf. Information Technology in Higher Education and Training,* pp. 196-199, July 2007.

[5] http://xen.org, 2009.

[6] http://www.realvnc.com, 2009.

[7] W.I. Bullers, Jr., S. Burd, and A.F. Seazzu, "Virtual Machines—An Idea Whose Time Has Returned: Application to Network, Security, and Database Courses," *ACM SIGCSE Bull.,* vol. 38, no. 1, pp. 102-106, Mar. 2006.

[8] D.A. Menasć, "Virtualization: Concepts, Applications, and Performance Modeling," *Proc. 31th Int'l Computer Measurement Group Conf.,* pp. 407-414, 2005.

[9] XenSource, "Xen: Enterprise Grade Open Source Virtualization," white paper, http://www.xen.org/files/xenWhitePaper3.2.pdf, 2006.

[10] XenSource, "Xen Networking," http://wiki.xensource.com/xen wiki/XenNetworking. 2009.

[11] http://virt-manager.et.redhat.com, 2009.

[12] T. Richardson, Q. Stafford-Fraser, K.R. Wood, and A. Hopper, "Virtual Network Computing," *IEEE Internet Computing,* vol. 2, no. 1 pp. 33-38, Jan./Feb. 1998.

[13] T. Richardson, "The RFB Protocol," http://www.realvnc.com/docs/rfbproto.pdf, 2009.

[14] T. Zimmer, D. Geoffroy, and M. Bilaud, "Best Practice of On-Line Labs in Electrical Engineering Education: A Ten Years Experience at the University Bordeaux," *Proc. Eighth Int'l Conf. Information Technology in Higher Education and Training,* pp. 162-167, July 2007.

[15] C.S. Tzafestas and M. Alifragis, "Virtual and Remote Robotic Laboratory: Comparative Experimental Evolution," *IEEE Trans. Education,* vol. 49, no. 3, pp. 360-369, Aug. 2006.

[16] H.A. Lahoud and X. Tang, "Information Security Labs in IDS/IPS for Distance Education," *Proc. Seventh Conf. Information Technology Education,* pp. 47-52, 2006.

[17] W.C. Summers, W.C. Bhagyavati, and C. Martin, "Using a Virtual Lab to Teach an Online Information Assurance Program," *Proc. Second Ann. Conf. Information Security Curriculum Development,* pp. 84-87, 2005.

[18] R.T. Abler, D. Contis, J.B. Grizzard, and H.L. Owen, "Georgia Tech Information Security Center Hands-On Network Security Laboratory," *IEEE Trans. Education,* vol. 49, no. 1, pp. 82-87, Feb. 2006.

[19] M. Duarte, B.P. Butz, S.M. Miller, and A. Mahalingam, "An Intelligent Universal Virtual Laboratory (UVL)," *IEEE Trans. Education,* vol. 51, no. 1, pp. 2-9, Feb. 2008.

[20] M. Grigoriadou, E. Kanidis, and A. Gogoulou, "A Web-Based Educational Environment for Teaching the Computer Cache Memory," *IEEE Trans. Education,* vol. 49, no. 1, pp. 147-156, Feb. 2006.

[21] N.I. Sarkar and J.H. Lian, "LAN-Designer: A Software Tool for Teaching and Learning LAN Design," *Proc. Third IEEE Int'l Conf. Advanced Learning Technologies (ICALT '03),* pp. 260-261, 2003.

[22] J. Nieh and C. Vaill, "Experiences Teaching Operating Systems Using Virtual Platforms and Linux," *ACM SIGOPS Operating Systems Rev.,* vol. 40, no. 2, pp. 100-104, Apr. 2006.

[23] C. Border, "The Development and Deployment of Multi-User, Remote Access Virtualization System for Networking, Security, and System Administration Classes," *ACM SIGCSE Bull.,* vol. 39, no. 1, pp. 576-580, Mar. 2007.

[24] M. Anisetti, V. Bellandi, A. Colombo, M. Cremonini, E. Damiani, F. Frati, J.T. Hounson, and D. Rebeccani, "Learning Computer Networking on Open Paravirtual Laboratories," *IEEE Trans. Education,* vol. 50, no. 4, pp. 302-311, Nov. 2007.

[25] A. Holzinger, M. Kickmeier-Rust, S. Wassertheurer, and M. Hessinger, "Learning Performance with Interactive Simulations in Medical Education: Lessons Learned from Results of Learning Complex Physiological Models with the HAEMOdynamics SIMulator," *Computers & Education,* vol. 52, no. 2, pp. 292-301, Feb. 2009.

[26] A. Holzinger, M. Kickmeier-Rust, and D. Albert, "Dynamic Media in Computer Science Education; Content Complexity and Learning Performance: Is Less More?" *Educational Technology & Soc.,* vol. 11, no. 1, pp. 279-290, 2008.

Muhammad Wannous received the MEng degree in computer science and electrical engineering in 2009 from Kumamoto University, Japan, where he is currently working toward the PhD degree in the Graduate School of Science and Technology. His main research topic is the design of Web-based ICT laboratories and integrating them into the available courses. He has published a number of papers on this topic in international and Japanese conferences. He is a student member of the IEEE and the IEEE Computer Society and is also a member of the IEEE Education Society.

Hiroshi Nakano received the PhD (Doctor of Science) degree from Kyushu University. He is currently a professor in the Center for Multimedia and Information Technology, Kumamoto University. His university teaching experiences are in physics and information science at Nagoya University, and information education and instructional systems (ICT field) at Kumamoto University, where he introduced university-wide e-learning environments and played an important role in founding the Graduated School of Instructional Systems and the Institute of e-Learning Development. His research work focuses on the development of the virtual learning environments, e-laboratory systems and virtual reality for e-learning, and scientific visualization.