Some Experiences in Using Virtual Machines for Teaching Computer Networks

Ricardo Nabhen and Carlos Maziero

Pontifical Catholic University of Paraná
Exact Sciences and Technology Center
Curitiba PR, Brazil
{rcnabhen,maziero}@ppgia.pucpr.br

Abstract. Laboratory practice is a fundamental aspect of computer network learning. Experiments tend to be very specific, frequently demanding changes in the local network topology and privileged access to the operating system configuration. These features impose a specific and exclusive laboratory for network teaching experiments. However, it is not always possible to provide such laboratory; the reality in most institutions is to have shared laboratories, used by different students and disciplines. This problem can be alleviated by the use of virtual machines, allowing each student to build his/her own network experiment, using the appropriate topology, and thus not disturbing the other activities running in the lab. This paper presents some experiences in using virtual machines to teach advanced aspects of computer networks, such as IPSec, firewalls and network services. Also, some key points are highlighted in order to show the benefits of virtual machines for pedagogical practice.

1 Introduction

Laboratory practice is a fundamental aspect of learning computer networks. Experiments that are more complex usually demand changes in the local network topology and privileged access to the operating system configuration on each host. For instance, in an experiment aimed at implementing a firewall architecture, students need to a) organize the laboratory network topology, putting computers on the external, internal, and DMZ (Demilitarized Zone) networks, b) install additional network interfaces on the machines that will route and filter packets between the networks, c) configure their respective routing and filtering tables, and d) configure the network addresses and other attributes for the machines in the three regions.

This unconventional characteristic of computer networks laboratory classes implies a specific and exclusive laboratory for teaching network experiments. This

problem also appears in other system-related disciplines, such as operating systems. However, it is not always possible to provide such a laboratory; the reality in most teaching institutions is to have computer laboratories shared among several students and disciplines. This limitation makes it impractical to have complex experiments that demand laboratory exclusiveness. This problem can be alleviated by the use of virtual machines. The VM technology allows each student to build her own virtual computer network, connected according to the topology demanded by each experiment, without interfering in the laboratory physical structure and thus not disturbing the other activities running in the laboratory.

There are several ways to use virtual machines to teach system-related disciplines; this subject has been studied since the 80s [7]. However, the recent advances in virtual machine technology [17] explain the growing interest in using it as an important tool in the learning process. Today, most students have enough computing power in their home computers to execute several virtual machines simultaneously, allowing them to reproduce the class experiments at home.

This paper presents some experiences in using virtual machines to teach advanced aspects of computer networks, such as IPSec, firewalls and network services. The text is structured as follows: section 2 presents the virtual machine technology, enumerating its possibilities, advantages, and limitations; section 3 discusses the possibilities of using virtual machines to teach computer networks; section 4 presents the virtual machine monitor User-Mode Linux, which was used in this work; section 5 presents the context in which the experiments presented here were developed; section 6 shows and analyzes some of the experiments realized; section 7 discusses relevant related work; and finally, section 8 concludes the paper.

2 Virtual machines

A virtual machine (VM) is defined in [16] as an efficient and isolated duplicate of a real machine. A virtual machine environment is created by a *Virtual Machine Monitor* (VMM), also called an "operating system for operating systems" [8]. The monitor creates one or more virtual machines on a single real machine. Each VM provides facilities for an application or a "guest system" that believes itself to be executing on a standard hardware environment. VM monitors build some properties that are useful in system security, such as isolation (a software running in a VM cannot access or modify the monitor or other VM), inspection (the monitor can access the entire VM state), and interposition (the monitor can intercept and modify operations issued by a VM) [8, 16]. Typical uses for virtual machine systems include the development and testing of new operating systems, simultaneously running distinct operating systems on the same hardware, and server consolidation [17].

There are two classical approaches to organize virtual machine systems. In Type I systems, the virtual machine monitor is implemented between the hardware and the guest system(s); the Xen [3] and VMWare ESX Server [21] virtual environments are good examples of this approach. In Type II systems, the monitor is implemented as a normal process of an underlying real operating system, called the host system. The

VMWare Workstation [21] and User-Mode Linux [6] virtual machine systems adopt this architecture. Both approaches are depicted in Figure 1.

Standard PC processors provide no adequate support for hardware virtualization. Consequently, virtualization overhead can be as high as 50% of total computing time [4]. However, recent research significantly reduced such costs to under 10%, as shown in [12, 17]. For instance, the VMware system [21] uses an on-the-fly rewriting technique in which the binary code loaded by the virtual machine is dynamically modified to better adjust it to the virtual environment, improving its performance. After completely rewriting the logical interface between the monitor and the guest kernels, the Xen project [3] obtained average computing costs under 3% for virtualizing Linux, FreeBSD, and Windows XP. These research results open many perspectives on the use of virtual machines in production environments.

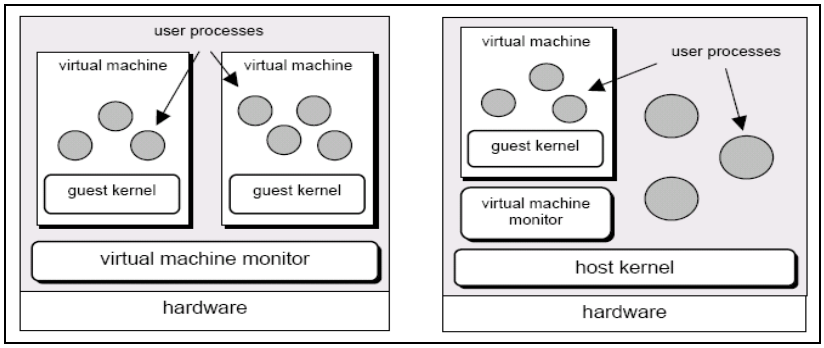

Figure 1. Type I (left) and type II (right) virtual machine monitors

3 Using virtual machines to teach computer networks

There are several ways to use virtual machines to teach system-related disciplines; this subject has been studied since the 80s [7]. However, the recent advances in virtual machine technology [17] explain the growing interest on using it as an important tool in the learning processes. Today, most students have enough computing power in their home computers to execute several virtual machines simultaneously, allowing them to easily reproduce the class experiments at home. Therefore, the virtual machine technology can be adopted with concrete benefits in system-related laboratory classes, such as computer networks and operating systems. Among those benefits, one can mention [4, 12]:

- it is possible to create more virtual hosts than the number of physical computers available in the laboratory, allowing each student to create complex scenarios involving several hosts;
- the number of network interfaces in each host and their interconnection are defined in the virtual context, with no restrictions related to the laboratory physical structure or the local computer's hardware configuration;

- the student is the administrator of her virtual hosts, allowing her to change their configuration and to install the software required by each experiment;
- the student can save on the real machine the configuration and state of each virtual host; allowing her to develop more complex, longer, or incremental experiments;
- finally, the student can reproduce the experiments at home.

A first approach for using virtual machines in computer network teaching would be to provide a specific laboratory, in which each computer would have locally installed a virtual machine monitor, like UML [6] or VMWare Workstation [21]. Such a system could be configured to allow virtual hosts to interact with the real hosts in the lab or with virtual hosts running on other real hosts. This approach, used in [1, 13, 18], is simple to implement, but is not flexible, because it demands a specific (yet not exclusive) laboratory. Furthermore, when deploying longer or incremental experiments, the student remains "tied" to the same computer on which she started it. Also, this structure is harder to reproduce at home and makes it impracticable to work remotely on the experiments.

Another approach consists of installing the virtual machine monitor on a central server, accessible to the students through the network, as proposed in [5]. In this case, the student connects to the server, starts the virtual machines needed for the experiment and interacts with them locally (in the server) or using real hosts on the local network. Although this approach is more flexible, it demands a central server for the virtual machine's execution, which can be very demanding in processing, memory and disk space.

An important aspect to discuss here is the adequacy of the distinct types of virtual machine monitors for the teaching environment needs. A type I monitor executes directly on top of the hardware (or is embedded in the host operating system). In this case, creating a new virtual machine is a privileged operation, only accessible to the system administrator, making it impracticable to create virtual machines on demand. Furthermore, access to low-level resources (such as drivers) and network configuration are also privileged, limiting the configuration possibilities. Examples of using type I monitors as a teaching tool are presented in [16, 20]. On the other hand, a type II monitor is seen by the host system as a user process. Thus, the creation of virtual machines based on user demand is restricted only by the availability of resources such as memory and disk space on the host system. Moreover, each student has full control over the hardware and software configuration of each virtual machine for the deployment of more complex experiments. The papers [1, 5, 13, 18] use type II monitors, exploring the on-demand creation of virtual machines.

4 The User-Mode Linux monitor

The experiments described in this work were accomplished using UML - *User-Mode Linux* [6], a type II virtual machine monitor that allows Linux guest systems to execute on top of a Linux host. This monitor was chosen due to its main features, which are:

- it is open source software, which can be adapted and customized to user-specific needs; being free, it allows any student to reproduce the experiment environment at home;
- the monitor is embedded in the guest operating system kernel code; monitor and kernel are seen by the host system as a single executable file;
- the guest kernel executes unmodified native Linux applications, allowing the user to install software packages already available to current Linux systems such as RedHat, Debian, etc.;
- the number of disks and network interfaces of the virtual machine can be defined at the command prompt, when the virtual machine is launched;
- it allows the use of predefined disk images, shared through a copy-on-write policy (COW), which results in efficient host disk space usage;
- several network interconnection mechanisms are available, from virtual isolated hubs to virtual switches connected to a host interface, allowing the virtual machines to interact with other real hosts.

5 The teaching environment

In our institution, computer network teaching laboratory activities are carried out mainly in a specific Network Laboratory. This laboratory is composed of 25 computers, distributed on five rows and interconnected by two local networks. There are also hubs, switches, routers, wireless adapters, patch panels and other connectivity equipment available to students. In addition, they have full access to the machines' configurations. The motivation for using virtual machines in this environment emerged from these observations:

- the students should come to the university to do their extra-class work; as the laboratory usage is intensive, they have some difficulty in having the lab free enough time for their experiments;
- the frequent changes in the machines' configuration needed by each experiment are a source of headaches and difficulties, mainly for beginner students;
- very frequently, an ongoing experiment was deleted by someone else.

Based on such evidence, we started to devise a solution using virtual machines, which allowed us to solve the problems found and offered other benefits, such as the possibility to carry out experiments remotely, even when graphical clients were needed (browsers and e-mail clients). A major concern was the privacy of the experiments: we would prevent one student from observing, interfering with, or even deleting another student's work. The solution found consisted of offering a type II virtual machine system on a medium-size server that can be remotely accessed by the students using text terminals (SSH) or graphical terminals (X11 or VNC). Each student has an account on the server and can launch on-demand virtual machines,

configured to meet the experiment requirements. Figure 2 depicts the architecture of the deployed teaching environment.

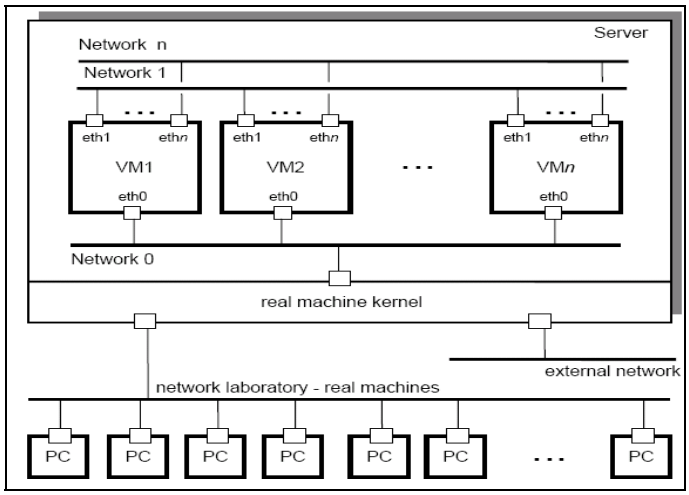

Figure 2. The teaching environment

This environment allows students to create on-demand, user-configured virtual machines, offering also the following benefits:

- as the server is integrated into the laboratory network, the virtual machines can interact with the laboratory computers, allowing a more real and convincing environment, if needed;
- the server offers remote access to user accounts through text terminals (SSH) and graphic terminals (X11 and VNC); the graphical interface is important when making experiments using popular network services like WWW and E-mail;
- the virtual disk images and other configurations are stored in the student disk area on the server, allowing the student to deploy long duration or incremental experiments;
- predefined images containing specific configurations and/or software are available in a public directory;
- graphical applications such as protocol analyzers and web clients can be run inside each virtual machine (their windows will be sent to the user graphical desktop session).

The current server configuration consists of a dual PIII Dell PowerEdge 1500 server (1.1 GHz), 2 GBytes RAM and a 100 GByte SCSI disk. The host system runs Fedora Core 3 Linux and the guest machines run RedHat 9 Linux. This server normally supports up to 40 active users; during a usage peak, up to 55 virtual machines simultaneously active were observed (monitoring statistics can be found at *http://espec.ppgia.pucpr.br*).

6 Sample practical experiments

The use of virtual machines provides a great flexibility for activities related to computer networks teaching. It allows easy implementation of experiments that would be much harder to do without an exclusive lab (or virtual machine), going from simple host configuration tasks to more complex ones such as network services installation, configuration and operation. Students can be challenged with scenarios and problems typically observed in computer networking. Such scenarios can be deployed individually or by groups of students, allowing them to interact, collaborate, and better understand. This section presents two examples of experiments that represent practical situations in network security.

6.1 Firewall architecture - DMZ scenario

The DMZ (De-Militarized Zone) network is a usual solution for firewall architecture. It is based on the isolation of an internal network from an intermediate network to which the externally-accessible hosts are connected. During this experiment, students propose a solution for a typical scenario where the use of a DMZ network is required, for instance:

> A business company provides a web-based application to its customers. This application runs in one of its web servers. In addition, the company has an internal network from which external web access is allowed. Finally, the company maintains a DNS server to keep its name service.

From the above scenario, students perform the network topology planning, which is followed by procedures related to IP addressing and routing schemes. Finally, they start to implement their proposed solutions using the infrastructure provided by virtual machines. Figure 3 shows a typical network topology to implement this scenario. It shows the use of six virtual machines (VM 1 to VM 6). From the point of view of the learning process, two aspects justify the importance of using virtual machines in this context. First, this scenario could be implemented by one student or by a group, giving conditions to all of them to exercise the practical procedures. This is a key point for enhancing the practical skill that is being taught, which is sometimes limited by the resources available. Second, the implementation of this complex scenario would be impractical without the use of virtual machines, because it would require a large number of hosts and network devices to be available to students. Also, these equipments should have a suitable configuration for this particular practice, sometimes a difficult task in shared and frequently busy teaching laboratories.

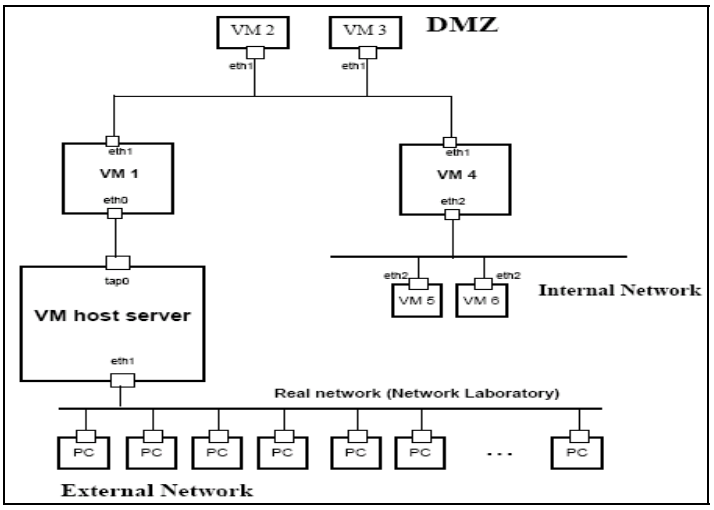

Figure 3. Typical **D**MZ network topology using virtual machines

In this scenario, the following network services should be installed:

- WEB Server, on VM 2: students usually adopt the open-source http server maintained by the Apache Http Server Project (httpd.apache.org).
- DNS Server, on VM 3: the Bind DNS server is used (www.bind.org);
- Packet Filtering on VM 1 and VM 4, using the IPTables package, from the NetFilter Project (www.netfilter.org).

After installing and configuring the services, students should define and apply the packet filtering rules on virtual machines VM 1 and VM 4. Below, some rules are presented in order to illustrate this scenario:

- Rule 1 - VM 1: Accept all traffic destined to VM 2 on port 80/TCP entering on interface eth0;
- Rule 2 - VM 1: Accept all traffic destined to VM 3 on port 53/UDP entering on interface eth0;
- Rule 3 - VM 4: Accept all traffic on interface eth2 destined to external hosts or to VM 2 on port 80/TCP coming from hosts which IP addresses belong to the internal network IP addressing range;
- Rule 4 - VM 4: Accept all traffic on interface eth2 destined to VM 3 on port 53/UDP.

Some other interesting experiments can be performed in this scenario. For instance, real machines located at the real network laboratory can be configured to access the virtual machine network. Doing so, students can simulate access to DNS and Web servers from external customers, allowing the investigation of possible attacks and vulnerabilities due to their configurations. In addition, browsers can be launched on VM 5 and VM 6, in order to access the Web server on VM 2 or even external Web servers.

6.2 IPSec Virtual Private Network scenario

A Virtual Private Network (VPN) consists of using a public communication infrastructure, such as the Internet, to connect securely to private networks. It provides a cost-effective solution for private network interconnection. The private network interconnection is created using techniques based on packet tunneling, encryption and authentication procedures.

The IPSec (IP Security) framework [11] is one of the most common solutions for VPN implementation. It is available on the majority of modern operating system distributions. IPSec security policies are defined based on rules providing a list of conditions and the corresponding list of actions to be performed. These actions are typically specified in terms of parameters that will be used by the communication of peering entities during the VPN session. For example, two sample rules are presented below in order to illustrate policy definition:

- Rule 1: If traffic packets use the TCP protocol and are destined to the WEB Server on port 80, then data encryption is required, otherwise reject the VPN connection;
- Rule 2: If traffic packets are coming from the 10.0.0.0/8 network then packet authentication is required, otherwise reject the VPN connection.

The IPSec framework provides two protocols for use during a VPN session. First, the AH (Authentication Header) protocol [9] defines a mechanism for packet authentication, while the second, the ESP (Encapsulating Security Payload) protocol [10], allows data encryption aiming to provide information confidentiality. A typical scenario adopted in practical experiments is presented in Figure 4. This scenario is a net-to-net VPN, where students should create a secure channel through to the Internet for network interconnection of a hypothetical company that has two computer networks located remotely one from the other, respectively, Local Area Network 1 (LAN 1) and Local Area Network 2 (LAN 2).

In Figure 4, six virtual machines are shown. The {VM 1, VM 2} and {VM 5, VM 6} pairs act as internal hosts from both local area networks. VM 3 and VM 4 are VPN gateways. Students should install IPSec packages on these gateways (e.g., the Kame IPSec Tools package available in http://ipsec-tools.sourceforge.net) and configure required policy rules for establishing the secure interconnection between networks using both eth_1 interfaces. The following are usual policy rules for this scenario:

- Policy Rule 1 - VM 3: All traffic from LAN 1 destined to LAN 2 must go through the VPN connection established between peering VPN gateways VM 3 and VM 4. This VPN session must use the ESP protocol;
- Policy Rule 2 - VM 4: All traffic from LAN 2 destined to LAN 1 must go through the VPN connection established between peering VPN gateways VM 4 and VM 3. This VPN session must use the ESP protocol.

After this configuration, students evaluate its efficiency by analyzing the captured traffic between the networks, where it is possible to examine tunneling and

encryption techniques and the importance of virtual private networks for secure communication.

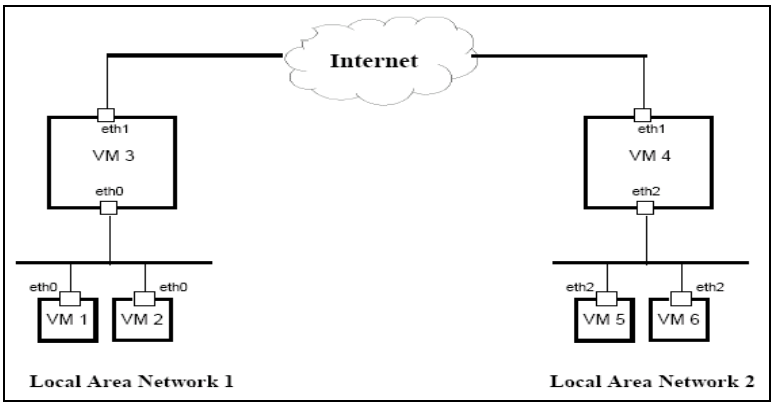

Figure 4. Example of IPSec VPN using Virtual Machines

7 Related Work

Several works present some experiences in the use of virtual machines for disciplines related to operating systems, computer networks and distributed systems learning. The work [5] presents some possibilities of the use of the UML environment for Unix server administration teaching, showing some examples for DNS service configuration on IPv4 and IPv6 environments, LDAP service configuration and routing configuration. A set of three servers hosts students' virtual machines, which can be created on demand. Real servers' access is made only through SSH (Secure Shell), which can limit the use of graphical clients by novice students. The paper [13] implements an environment, named *Velnet*, for computer network teaching. This environment is formed by a set of virtual machines using VMware and running on each student host. The communication between these virtual machines is done through virtual networks, without the possibility of communication between virtual and real machines. A graphical tool to support virtual machine creation based on pre-configured schemes was provided: routers, file servers, web servers, etc.

A different experience was presented in [15], where an IBM S/390 computer is used to run hundreds of Linux virtual servers, one for each student. These servers can be accessed from the Internet and are used for development of academic work related to operating systems and Web programming subjects. A similar solution was presented in [20]. The paper [1] shows a comparative study about the UML and VMWare environments. The configuration presented there differs from the one shown in our work because it is based on local installations on each network machine located at the laboratory, which requires exclusive access and cannot be accessed externally.

In [18], researchers propose a mixed approach, where virtual machines run locally on laboratory hosts using remote access technologies (e.g., VNC and Windows Remote Desktop) for 1) allowing teachers interact with students' desktops for supporting practical experiments, and 2) allowing students to view the teacher's desktop to get information about required configurations.

The main feature of monitors is the isolation of virtual machines from the real machine. This feature allows the configuration of each virtual machine as an autonomous host. However, another strategy, named virtualization, allows the creation of isolated and autonomous contexts under the same operating system. For example, Solaris Zones [19], FreeBSD Jails [14] and the Linux Virtual Server [22] adopt this strategy. The work presented in [2] explores the Jails functionality under the FreeBSD environment, creating an isolated replica of the user space in the operating system, each one having its own IP address. Such an approach has a better performance than virtual machines but restricts the students to user-space operations: they cannot configure or modify the network stack or the system drivers, because such structures belong to the real machine kernel and require root privileges to be managed. This restriction prevents experiments on package filtering and routing, for instance.

8 Conclusion

This paper presented some experiences in using virtual machines to teach advanced aspects of computer networks. It showed some important concepts of virtual machines and some scenarios for practical experiments, such as the setting of Virtual Private Networks and firewall architectures. Results showed several benefits of this use during the practical experiments, notably students' motivation and engagement, as well as a better utilization of the allocated time for classes. In addition, the attenuation of existing restrictions was accomplished due to the flexibility given by the use of virtual machines. This can be seen in the definition of several possible scenarios, such as network services and applications installation, configuration and operation. This rich set of possibilities allows a unique environment for computer network teaching in which students can maximize their learning experience.

References

1. J. Adams and W. Laverell, Configuring a multi-course lab for system-level projects, ACM Technical Symposium on Computer Science Education, (2005).
2. G. Armitage, Maximising student exposure to networking using FreeBSD virtual hosts. ACM SIGCOMM Computer Communications Review, 33(3), (2003).
3. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, L. Pratt and X. Warfield, Xen and the art of virtualization. Proceedings of the ACM Symposium on Operating Systems Principles – SOSP, (2003).
4. P. Chen and B. Nobel, When virtual is better than real, Proceedings of ACM Annual Conference on Innovation and Technology in Computer Science, (2001).
5. R. Davoli, Teaching operating system administration with User-Mode Linux. Proceedings ACM Annual Conference on Innovation and Technology in Computer Science, (2004).

6. J. Dike, A user-mode port of the Linux kernel, Proceedings of the 4th Annual Linux Showcase and Conference, (2000).
7. J. Donaldson, Teaching operating systems in a virtual machine environment, Proceedings of 18th SIGCSE Technical Symposium on Computer Science Education, (1987).
8. N. Kelem and R. Feiertag, R, A separation model for virtual machine monitors, Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, (1991).
9. S. Kent and R. Atkinson, IP Authentication Header. Request for Comments - RFC 2402, (1998).
10. S. Kent and R. Atkinson, IP Encapsulating Security Payload (ESP), Request for Comments - RFC 2406, (1998).
11. S. Kent and R. Atkinson, Security Architecture for the Internet Protocol, Request for Comments - RFC 2401, (1998).
12. S. King and P. Chen, Operating system extensions to support host based virtual machines, Technical Report CSE-TR-465-02, (2002).
13. B. Kneale, A. Horta, and I. Box, Velnet - virtual environment for learning networking. Proceedings of the 6th Australasian Computing Education Conference, (2004).
14. M. McKusick and G. Neville-Neil, The design and implementation of the FreeBSD operating system, Addison-Wesley professional, Addison-Wesley Professional, (2004).
15. R. Norton, Using virtual Linux servers, IEEE Computer, (2002).
16. G. Popek and R. Goldberg, Formal requirements for virtualizable third generation architectures. Communications of the ACM, 17(7), (1974).
17. M. Rosenblum and T. Garfinkel, Virtual machine monitors: Current technology and future trends, IEEE Computer, (2005).
18. M. Stockman, Creating remotely accessible virtual networks on a single PC to teach computer networking and operating systems. ACM Conference On Information Technology Education, (2003).
19. Tucker, and D. Comay, Solaris zones: operating system support for server consolidation, 3rd USENIX Virtual Machine Research Technology Symposium, (2004).
20. Villanueva and B. Cook, Providing students 24/7 virtual access and hands-on training using VMware GSX server, ACM SIG on University and College Computing Services Conference, (2005).
21. VMware, VMware technical white paper, VMware Inc., (1999).
22. VServer Project, Linux VServer Project, http://www.linux-vserver.org, (2004).