UNIVERSITY OF OSLO
Department of Informatics

Pedagogical Framework and Network Laboratory Management Tool for Practical Network Administration Education

Zahid Nawaz

Network and System Administration

Oslo University College

May 27, 2013

Abstract

System Administration is an emerging academic discipline, yet there remains no commonly-accepted approach to teaching the core skill of troubleshooting system problems. Troubleshooting is considered by many in the field to be the single most important skill for a working professional. The lack of common teaching methodology for this skill has resulted in an on-the-job-training approach to teaching that is not scalable or measurable. This research addresses this gap with an approach that may be developed into a reliable, repeatable, scalable, and, most importantly, measurable approach to teaching the critical system administration troubleshooting skill. The research produced two core results, a proposed Pedagogical Framework to guide the teaching process, and a Network Laboratory Management Tool that is a proof of concept and basis for a more general purpose, practical implementation of the Framework. In this thesis the Framework and Tool are presented in detail and future work is described.

Acknowledgements

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my supervisor Andrew Seely, whose contribution in stimulating suggestions and encouragement helped me to coordinate my project and navigate the learning process of this master thesis.

Furthermore I would also like to acknowledge with much appreciation of Hrek Haugerud, in the program and over the years for the heated discussions and motivation to pursue the master degree. I also wish to draw attention to Dr. Aeleen Frisch, Kyrre Begnum, and Aamir Maqbool Ahmed, as they have been greatly influential for my work over the years and a true source of inspiration.

I will also thank my friends and family for the support. Their contributions and patience was highly appreciated and needed. I would like to extend special thanks to my dear brothers and sisters for their cooperation in my life. Last but not least, I would like to say THANKS for those students having MS010A subject in this year because they helped me to figure out the problems list.

Contents

List of Figures

Chapter 1

Introduction

This chapter will introduce the problem statement, objectives, and highlight the motivation to conduct this project.

The effectiveness of using virtual networks to train students in computer network troubleshooting skills is well known. The purpose of this study is to develop a process to improve the gaining of troubleshooting skills learned in a virtual network environment. In recent years the complexity of technologies for an organization has increased. Especially in educational field the virtualization and cloud computer play a vital role to overcome the cost and provide an experimental environment to the students.

Nowadays the performance of Virtual Machine (VM) is increasing and it improved the value of troubleshooting skills. User complaints of performance in virtual machine or real machine can be stressful for system administrators. Virtualization helps consolidate server workloads, and it is clear that virtualization technology also has a hidden complexity and it increased the potential configuration issues in virtual machines such as DNS problems, disk storage configuration, packages installation issues, and port forwarding problems [11].

In the teaching of basic and advanced computer networks concepts, such as routing, mobility, and load balancing, students not only have to acquire the theoretical knowledge but also the practical skills to configure and manage real computer network scenarios[27]. In engineering education, these practical skills are acquired in a standardized laboratory environment.

Crash and omission failures are common in service level management. During the process different services can break down or a link can fail anytime. In addition, when the number of machines increases than the probability of a machine failure increases [18]. To repair or troubleshoot a computer is the core skill for any network system administrator. Network and system administrators are expected to sharpen their troubleshooting skills to repair the computers [29].

A system administrator has a duty to over comes such kind of service failures. In order to avoid failures or repair the broken state machines, efficient troubleshoot skills must deal with virtual machine complexity. Troubleshooting skills can take place from practice and experience, which are difficult to learn from a book [9].

Furthermore, virtualization is becoming more and more popular and widely used in many organizations, enabling access to required learning information. With the help of virtual environments, learning is not limited in time and space while expensive learning resources can be circulated to achieve the goals in education proliferation [9]. Virtualization is becoming more popular in educational field due to cost savings, leading universities and colleges start to implement the virtualization in their premises [1].

1.1 Motivation

According to Kyrre Begnum, Karst Koymans and Arjen Krap network and system administration educators are facing significant challenges to provide the practical and theoretical environment to students. In labs students need systems to install and configure them for learning experience[3]. One answer to this challenge is Problem-Based Learning (PBL) is an approach in student-centered pedagogy [24] in which the students meet in small groups to discuss a particular problem situation, note lectures, exercises or assignments under the supervision of an instructors [35].

There are many tools available in the market to make experiments with virtual networks. These tools are very costly and may not be practical to use in many situations. The virtualization environment in education field is getting popular because it reduces the cost of physical resources[40].My research will develop a pedagogical framework and a teaching tool in which set of virtual machines will be prepared in working configurations. Now assign these virtual machines or virtual networks as an assignment to all students in the class, they have to fix them. Through this students will get knowledge how to troubleshoot the machines.

During the study period there are many assignments in system administration course to build knowledge. Troubleshooting is very tough job because deploying new services, testing protocol interactions, or validating network configurations are hard problems in todays Internet [15]. With rapidly changing environments and software, theres no easy way to teach skills that matter. Our teaching tools are not versatile and agile enough.

This pedagogical framework approach will overcome the hurdles to teaching troubleshooting and it may lead to a decrease in the burden of teaching. The teacher may easily build a virtual network having different number of virtual machines and they will easily restore these VM into broken state. At the end of a course, the network administrator or teacher may easily rebuild the whole virtual teaching network.

" I hear and I forget

I see and I remember

I do and I understand. "

Chinese Proverb

Experimentation is very important for systems-oriented courses in computer science curricula, which can give students hands on opportunity to understand the existing systems and their initial design tradeoffs, to answer many what if questions for emerging scenarios, and to potentially improve these systems in both functionality and performance. Experimentation has been credited as a key to an active, engaging learning experience [32] [28].

1.2 Problem Statement

Troubleshooting takes practice and experience, which are difficult to learn from a book. It is recommended that you experiment and test on broken state virtual machines to develop your troubleshooting skills [33]. It would be difficult to motivate the students to learn the computer networking skills because many students dont like dry and boring concepts [38].

During the literature review we found many textbooks having computer networking knowledge and each of them have extensively material about computer networking [31] [17], and local area network design strategies may also be found online[16] [6].

In the market there are number of network simulators to build a network but these are not sufficient to learn the networking skills because when students configure the actual network they gain first hand valuable experience.

Through network simulators or models students do not gain actual network configuration experience [5] [37].Thats why we want to create an environment where students can gain actual network experience to repair the networks and virtual machines.

One of the particular problems that system administration students face is that they do not have realistic troubleshooting experience, leading to difficulties finding real-world jobs. The true core of system administration is troubleshooting. Most of students studying networks and system administration have skills just to install and run different services, but they dont have any skill how to troubleshoot systems in different scenarios. In educational premises it is very hard to dedicate a single system to each student, so with the help of virtualization environments we can create VMs and practice on it to manage this issue as well. Teachers should provide an environment to build troubleshooting skills.

The high demand of virtualization environments in educational sector is increasing and this demonstrates a need of troubleshooting and recovery solutions in it because troubleshooting is main duty of any network administrator.

This thesis will develop a pedagogical framework and management tool for realistic system administrator education teaching. Teaching system administration is an emerging academic discipline, but there is no standard methodology for teaching the system administration core competency of system troubleshooting.

- a. This is partly due to the immaturity of the discipline.
- b. Partly due to the lack of standard tools and approaches[38].
- More mature disciplines have standard tools. In Biology, everyone dissects frogs the same way; in Physics: everyone does the same Newtonian experiments the same way.

The real center point of this project is to develop a pedagogical framework and network lab management tool that allows effective teaching (and learning) of system administration education.

The true core of system administration is troubleshooting; this framework and tool is intended to fill a gap in this field. Troubleshooting skills can take place from practice and experience, which are difficult to learn from a book [9]. TThere is no general-purpose tool today that provides realistic troubleshooting in the classroom environment for network system administration students.

Many system administration college courses currently leverage virtualization and other issues. Because the students start out with functioning systems, assignments can focus less on troubleshooting, the core of System Administration. Most of them focus how to deploying new services, testing protocol interactions, or validating network configurations but not on troubleshooting [36].

System Administratorss manage entire network and are responsible for performance and responding to system failures. Even if we could perfectly predict failures, we still need to apply some solution. Cloud computing and virtualization have opened a new window in failure management through using, resuming, and migrating VMs[36] [12].

1.3 Objectives

The main purpose of this research is to create a pedagogical framework and network lab management tool to teach network troubleshooting skills; on the base of the suggested framework, a prototype tool will be demonstrate the applicability of the framework and validate if it will provide a real environment for network troubleshooting sills.

With the help of common virtualization tools like XEN and MLN, virtual networks will be created to demonstrate the pedagogical framework. After creating virtual networks, thetool will convert them into pre-defined broken state. These networks will be assigned to the students to fix them into working condition. So the objectives show like this,

1. Give students and researchers a way to create and use a protected and versatile virtual network environment where they can practice to find the problems and solve them related to the subjects of system and network administration.
2. To create a virtual networks that helps the students to learn the network troubleshooting skills.
3. With the help of this framework and network lab management tool the network teachers will easy create lab assignments for new students to practice troubleshooting skills.
4. No need to create such environment manually again and again, only first time create virtual networks, next time suggested tool will reset it easily.

5. The tool also allows the teacher to easily and dynamically add new problems for students to solve.
6. Teacher will gives feedback, assessment, and verifies automation results by verifying students work using the repaired virtual machines.

Computer networking is the major subject in the Information Technology field and institutes try to provide the networking environment for students. Computer networking is becoming more popular because many organizations have their own computer networks so they need skilled persons having computer-networking knowledge. In general analysis the students learn more effectively from those subjects having practical work and practical exercises. For example if computer-networking course has practical work like, to configure the switch, assigning the root privileges to new user, installing and configuring the servers, and manage all networks activities through software [8] [34] [7] [21]. According to literature review this will be our objective to provide an environment where students can get actual network troubleshooting skills. And it will also helpful to the teacher to manage the virtual networks for students assignments.

1.4 Educational Context

As the use of the Internet expands into more areas, the eld of system and network administration is becoming more important as well. As this grows, the education of new administrators grows in momentum too. Several colleges and universities offer courses on system administration and a few, like the University College of Oslo, offer full degree plans. [39]

Network labs are set up to provide environments where the students can learn and solve problems. Entire rooms are often allocated for this purpose and contain different network hardware as well as a number of machines [39].It depends on the specic course and the students experience whether the lab has been set up ahead of time or if they have to do it themselves.

- Course support teams need a way to remember congurations in order to repeat the procedure for new students
- The labs need to scale so that the assignments work for many students at a time [39].

Although staff works hard to set up labs, which meet these criteria, it is difcult to achieve. These labs are usually difcult to manage successfully. The different courses and the number of machines they might need lead to conicting requirements. [39]

When semesters or courses end, then staff need to demolish the networks and create new networks for coming students for assignments. This can be resource-intensive and may not be practical for institutions with limited resources.

1.5 Approach

Enabling virtual machines (VMs) into broken state and again convert into their functioning state by tool is time-intensive work. A literature review does not find any tool or framework in network administration. For this we make a pedagogical framework first as described in figure 4.1. On the basis of this framework we developed a prototype to demonstrate the applicability of the framework to the stated problem.

First of all we build a physical server with Debian 6.0.7 as an operating system. XEN and MLN tools installed on server to create the virtualization environment. Before going to develop the prototype tool the manual commands run one by one to check their accuracy and reliability. These commands take place to build a prototype, which will install XEN and MLN automatically. The prototype has ability to create a new virtual network and assign different problems to particular virtual machine just to convert into broken-state. After converting the virtual machine into broken-state, the tool is also able to repair it as well and bring into functioning state.

1.6 Thesis Outline

The thesis is divided into nine chapters and each of them has following outlines.

The first chapter has introduction part; apart of this it highlights the motivation, problem statement, objectives, and thesis outline.

In the second chapter you can find background, literature review, and technology used throughout this thesis.

The third chapter highlights the practical approach with full details Of tools used for the project.

The fourth and fifth chapters have the results of this projectdescribing the pedagogical framework and its consequences.

The fifth chapter introduces the Network Lab Management Tool and demonstrates its function.

The sixth chapter is the analysis and discussion. Here the results are explained in brief with pros and cons. Means the findings and behavior of the "Network Lab Management Tool" is discussed and reviewed from flexibility and its performance. In the last part of this chapter the future work are suggested in few points.

The seventh chapter has full summary of this thesis with the name of "Conclusion." It concludes the thesis and summarizes the whole work.

The eighth chapter is the Appendix where you can find the Perl script used to develop a prototype Network and Lab Management Tool with full detail.

The nineth chapter has bibliography or list of references used to conduct this thesis for relevant study in literature review.

Chapter 2

Background and literature

This chapter will introduce the background literature and previous relevant research work. It will provide a short introduction about those topics that are used in this project, which relates to experiment and pedagogical framework. It will highlight that why we used these kinds of software and tools.

This chapter has information about the tools and technologies those we used in this project. Focusing on our main idea, to develop a pedagogical framework and management tool for network administration education, we accomplish one tool with different technologies. Here you can easily find full detailed information regarding to tools and terms.

2.1 Troubleshooting

Troubleshooting is one of the most important skills in system administration. Troubleshooting skills are considered crucial for a system administrator. Complex technical environments cannot be built one hundred percent correct, and are expected to include problems such as hardware, software, network devices, cables, room temperature as so on.

T troubleshooting is difficult to learn from a book due to the many different technologies, interfaces, and requirements. Troubleshooting is recommended to be taken by practice and own experiences or experience from others who have previous experience with the same environment, due to the rapidly-changing technology and customized implementations found in the industry.

2.2 Virtualization

Virtualization is useful in computer science education because it allows a better availability and management of resources to set up laboratories for the students. Up to now, virtualization has been used mainly for teaching administration of operating systems and the basic concepts of computer networks [27].

Nowadays the virtualization technology plays a vital role in the field of computer industry. Many companies and educational institutes create virtual environment in their premises to take the advantage of it. In the market there are different kinds of software to provide the virtual platform like KVM, XEN, MLN, Proxmox, and VMware. Virtualization provides a complete environment where user can install operating system just like on real machines. On a single server user can creates many VMs and on each VM a single operating system can be install. Through virtualization user can get maximum utilization of the hardware resources.

2.2.1 Hypervisor

In virtualization many VMs can be run at a time, and no one interrupts another VM. Hypervisor control all these VMs, the hypervisor is a tool that allows different operating systems to be installed on a single physical machine known as host for the sake of sharing hardware resources. Hypervisor manage each VM so each machine behaves independently because the hypervisor keeps its assigned resources, such as processor, storage, and memory to separate from other VMs. As we mentioned that virtualization has many VMs and each VM has its own operating system so they request to utilize the resources, hypervisor has duty to manage these requests and allocate these resources in a proper way. So hypervisor works between different operating systems and system resources to manage all of them.

2.2.2 Host

Host is the system that directly attached to the physical resources and capable to run many VMs at a time, and manage all of them without any problem is called virtualization host. It provides facility to all VMs just maximum utilize the physical resources.

2.2.3 Guest/Guest Operating System

All virtual machines those are running on host called guest. Each virtual machine behaves independently, operating system installed on it called guest operating system. So there are two types of operating system in virtualization, one of them is called guest operating system and second is host operating system.

Virtualization is getting more popular in educational field because of its benefits. There are few benefits to use the virtualization in education, Fo example, To ensure all students having same type of environment where they can find same specifications and they can easily access virtual machines from anywhere. Students can installed and configure software on these virtual machines very easily. Virtualization reduce the cost of resources and especially it reduce the number of rooms allocate for different labs. During the experiments if VMs get crash than you can rebuild in a short interval of time. Virtualization technology is very useful to make network assignments for students. Thats why we will use this technology in our project.

2.3 Virtualization Tools

There are many Virtualization tools [30] [41], which allow the emulation of network infrastructures in a highly realistic way. Moreover, the integration of computer technology can help students to perceive the improvement of their learning process [20]. For this reason, in computer science education, these tools have traditionally assisted the teaching of different subjects [10] [25].

Currently, some well-known virtualization tools are: Qemu, XEN, WMware server, virtual box, and User Model Linux (UML). Nevertheless, these tools do not provide a specific mechanism to facilitate the deployment of virtual network scenarios [27]. For this reason, in the latest years, some initiatives oriented to simplify the deployment and configuration of virtual network infrastructures have appeared [27].

Students need a practical environment that can be easily accessible and scalable. For this I suggest a framework (Figure 4.1) to build a virtual environment that would be helpful for teachers to make a network assignments and as well as it would be helpful for students to develop their troubleshoot skills.

2.4 XEN

XEN is open source software, which used to create the virtualization environment. It can execute the multiple guest operating systems at a time with strong performance and resource isolation. [2] XEN is also using hypervisor to manage the guest operating systems and allocate the physical resources to all VMs. It uses the Parallel virtualization concept [13] [26] to monitor the VMs. In XEN the virtual machine known as domain and it has low overhead but it behaves as a real machine. A nice feature of XEN is live migration of VM during it proceedings, mean virtual machine can be migrate across the physical server without down time and it increase it importance [14]

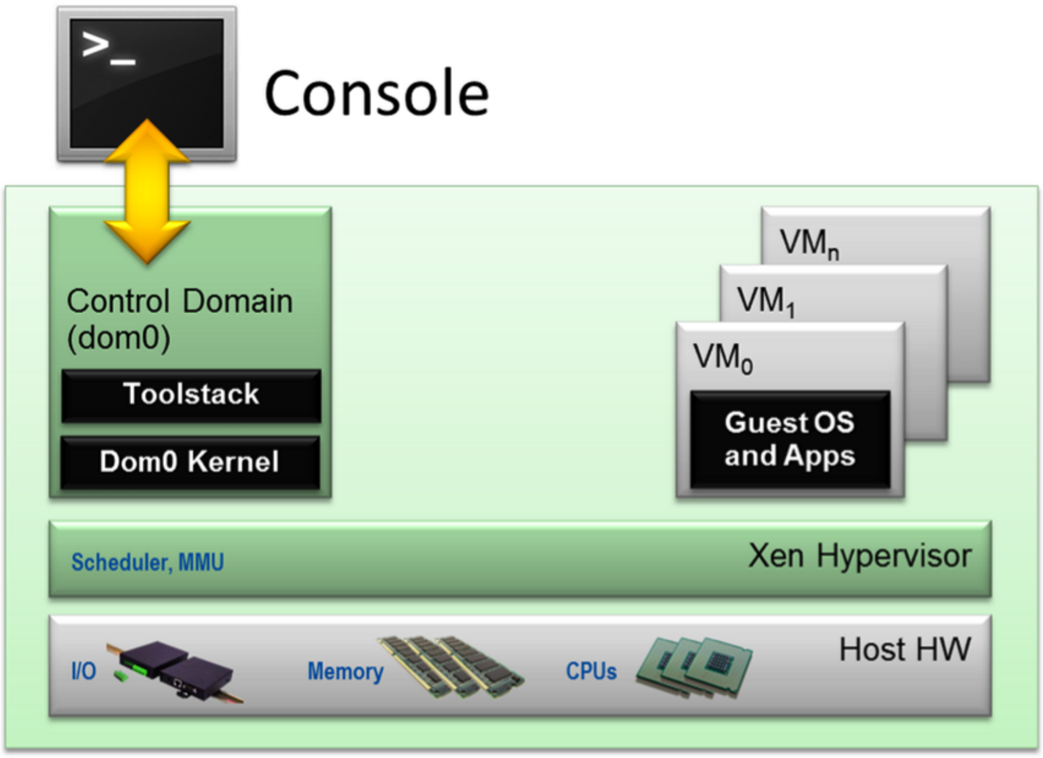


Figure 2.1: Conceptual implementation of the Pedagogical Framework in XEN

The XEN hypervisor runs directly on the hardware and is responsible for handling CPU, Memory, and interrupts. It is the first program running after exiting the boot loader. On top of XEN run a number of virtual machines? A running instance of a virtual machine in XEN is called a domain or guest is shows in figure 2.1. A special domain, called domain 0 contains the drivers for all the devices in the system [14].

XEN domain defines a file where it configured the virtual machine features like memory, CPU, and storage image. XEN create the VMs through xm tool and it based on domain configuration file. [14]

2.5 Manage Large Networks (MLN)

MLN is a tool, which manages the virtual machines; it is developed in Perl language and has very easy coding to build virtual networks. It works with XEN platform as well. It is getting more popular in education because it is ideal to create a number of virtual networks for students. University of Oslo is using this tool in different subjects to create virtual network lab for students. It is template-based tool having different templates for different file systems [2]. User can create and manage the virtual network through MLN, it is very powerful tool works with XEN and UML. A good thing that it is freely available on MLN website.

When MLN configured than it create three folders in /opt/mln/ directory. One of them is /opt/mln/templates/ where user can save different templates and can be useable as a file system for virtual machines. The second folder /opt/mln/files/ is use to save the MLN script files for execution. The third folder /opt/mln/projects/saves the projects those create in MLN script. In chapter three heading 3.4.4 explain how to install the MLN and how to build a virtual network with manual commands. MLN (Manage Large Networks) [42] was first used in 2004 as a tool for providing a virtual firewall lab running User-Mode Linux for students [22]. It has since then been expanded to support XEN as a virtualization platform and to include a plugin framework [22].

MLN reads configuration files and builds the hosts file systems based on templates. Start and stop scripts are also generated. MLN has commands for building, starting, stopping, upgrading and monitoring virtual networks. Hosts can be rebooted individually and their console can be accessed from the physical machine, meaning that virtual machine without properly functioning network interface can be accessed nonetheless by students. This removes the need for supervised access to a machines console [22].

2.5.1 Virtual Hosts

Virtual host depend on file system and it run software in it. XEN or UML use as a virtual host for MLN tool. User can run different guest operating system in

MLN. Host makes a connection between MLN and physical server hardware or OS. MLN has functionality to create virtual switch or it can be use the XEN default switch [23].

2.5.2 File System Templates

MLN is using the templates as a file system for host. There are different file systems for virtual hosts. For the ease of user different templates can be found on MLN home page. Templates are vary regarding to Linux distributions [23].

2.5.3 Virtual Switch

MLN has the capability to create the virtual switch by UML or XEN. It can also use the XEN default switch called eth0. But the XEN has bridge as will like xenbr0 to connect different network interfaces together. Virtual hosts connect to these switches to transfer the packets visa these switches [23].

2.5.4 Virtual Networks

MLN is tool deal with virtual networks, through MLN script user can creates many virtual networks through one script. Each network has different numbers of virtual machines. Virtual switches and virtual machines make pretty virtual networks in MLN. You can make the virtual networks to be a part of real networks and both networks can communicate with each other. When virtual hosts and real hosts connect with switch than there is no difference and they can connect with same LAN as well [22].

MLN virtual networks can be created easily and used in scenarios in the field of configuration management, intrusion detection and packet filtering. Administration of the networks is possible through a simplified interface that can start and stop the entire network [22]. Using the configuration specification files, scenarios can be taken down and reproduced quickly for the next class or student group. From the perspective of a student, there are also some benefits. MLN also supports building and running systems as a different user then root, which allows students to create their own test networks. MLN also supports upgrading the configuration of a running virtual network, like adding more machines, where only the affected machines are rebuilt. This is practical for networks that are already in use in class [22].

2.5.5 UML

User-mode Linux (UML)[71] is a project where the GNU/Linux kernel is modified to run in user space, like a regular process. It runs and behaves like a kernel as compared to instruction set emulators, like VMware or Virtual PC, which can run any kernel. User-mode Linux provides several ways to connect virtual machines together and to the real world. Part of the available tools is a switch emulator called the UML_switch, used for routing packages and virtual networks. These can be used to create large networks of arbitrary topology [70].

Chapter 3

Approach

In this chapter the approach will be described. It will explain the basic design of the experimental environment. How we conducted the experiments and what kind of tools we used to build the infrastructure.

This chapter will describe approach and concepts used to achieve the goal stated in first chapter. We are going to build our pedagogical framework and tool to manage the virtual networks to create the pedagogical environment, resulting in a practical demonstration of the pedagogical framework. Approach and methodology play a vital role to build a framework or prototype.

3.1 Concept

According to the problem statement troubleshooting skills are very important for network administrator but traditionally very difficult to teach effectively. Because if you havent a good grip to troubleshoot the virtual machine or system than it will make a trouble for you in your practical field [4]. In education sector the numbers of computer systems are increasing so many of them start to deploy the virtualization tools, in this way virtual machines have to manage in different departments. As the numbers of virtual platforms are increasing into the market so students should have troubleshooting skills to manage any virtual network. Teachers need easy way to deploy virtual networks for students and provide broken state virtual machines as an assignment to fix them. Through this way students will learn how to fix the virtual machines.

This tool presents virtual machines with specific problems, in oder to encourage students to focus on a problem and to handle diversity of problems. If the students will not able to fix the broken state machines than suggested tool will fix only those problems which are created through this tool. If students create other problems than this tool will not fix them. In this way you should create new networks through this tool. May be this will helpful to the students and as well as to the teachers, because no need to create virtual network again and again for new comers. First time you have to build a virtual networks than tool will convert this virtual network(s) into broken state and again reverse the virtual machine into functioning state for new class. Teacher feedback and assessment will play a good role to check the performance of VMs.

3.2 Challenges Faced by Network Administrators

In virtual networks may be there are many VMs so it needs to manage all of them in a proper way. Creating and managing a VM also an inevitably overlaps with same specification and configuration. If you assign a task to network administrator to build a virtual networks and each VM has unique specifications than it is hard to do this. When a network administrator is going to build such kind of virtual networks than there are a few challenges he can face, like as follows [4].

Virtual machine management software supports only a single virtual machine platform.

Free management tools are often intended for running few virtual machines. No support for grouping or the design of larger networks.

Commercial tools offer better support for larger networks but are proprietary and impossible to modify.

Host configuration is usually not a part of the management software. [4]

To demonstrate the pedagogical framework, a problems list was developed and implemented in virtual machine templates. to produce a set of broken state machines. We listed many problems here and few of them we used into proposed tool. We try our best to show a simple solution of our proposed framework.

We design a framework according to our problem statement. In which we describes how our proposed tool will work. In figure 4.1 we used to show the holistic picture. In figure 3.2 we show the flow of processes implemented by the tool.

Students need a practical environment that can be easily accessible and scalable. For this we suggest a framework (Figure 4.1) to build a virtual environment that would be helpful for teachers to make a network assignment and as well as it would be helpful for students to develop their troubleshooting skills.

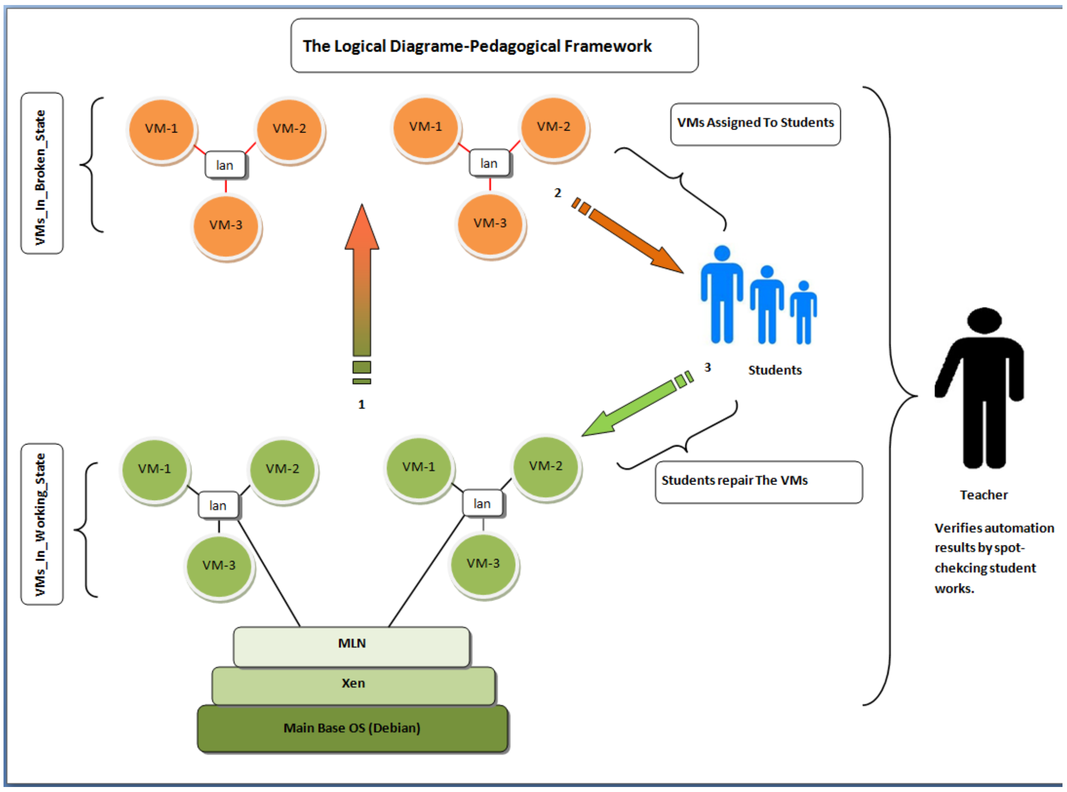


Figure 3.1: Logical Diagram-Pedagogical Framework for effective teaching

3.3 Process Flow Diagram

A proof of concept tool is required to support the Framework described in Figure 4.1, For this purpose we present a tool which implements the Frameworks mechanism. First the tool will be built with Base operating system (OS), XEN, MLN, and Perl. After this tool will create virtual networks according to the demand.

When all virtual networks will be created then the Tool will convert all virtual machines into broken state. Now teacher can assign these networks as an

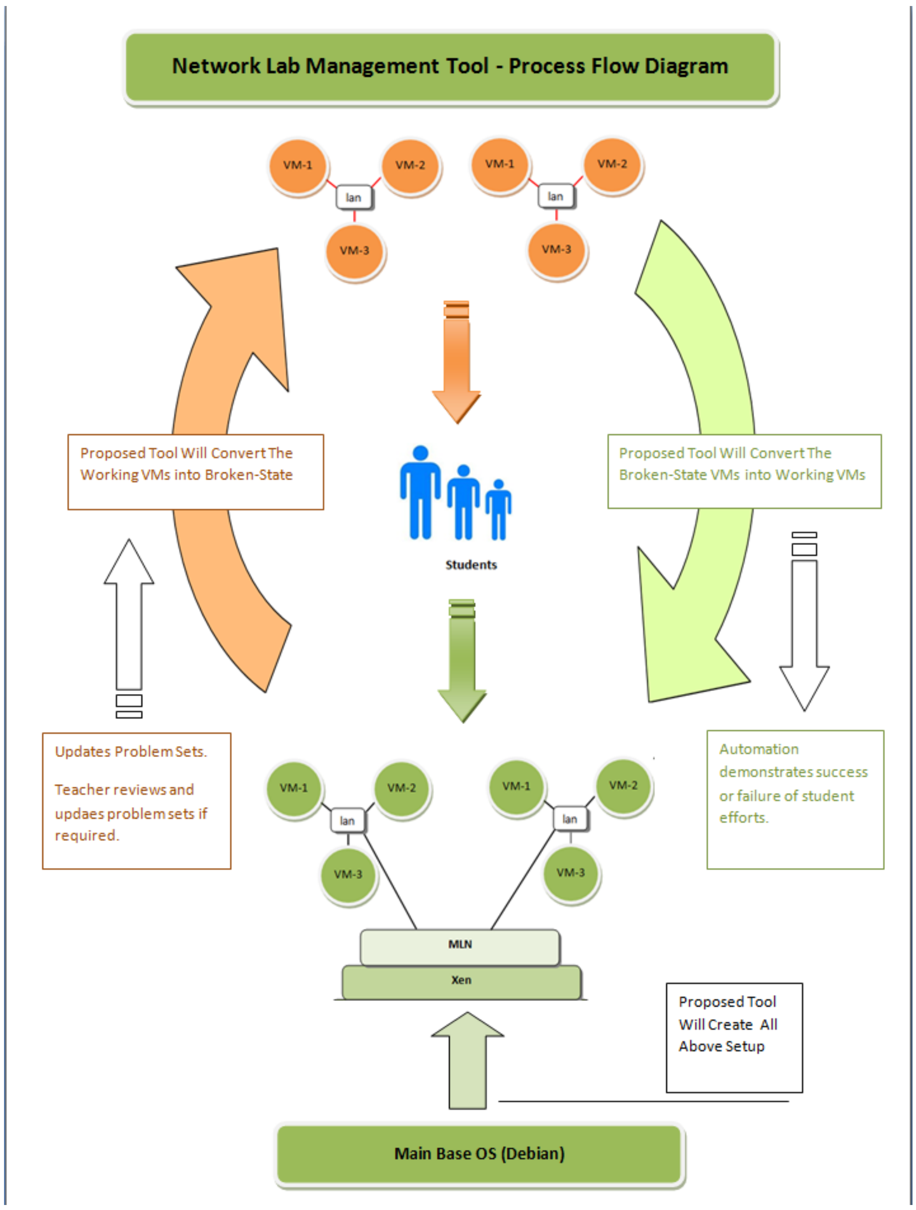

Network Lab Management Tool - Process Flow Diagram
VM-1
VM-2
lan
VM-3
VM-1
VM-2
lan
VM-3
Proposed Tool Will Convert The Working VMs into Broken-State
Students
Proposed Tool Will Convert The Broken-State VMs into Working VMs
Updates Problem Sets.
Teacher reviews and updaes problem sets if required.
VM-1
VM-2
lan
VM-3
VM-1
VM-2
lan
VM-3
Automation demonstrates success or failure of student efforts.
MLN
Xen
Proposed Tool Will Create All Above Setup
Main Base OS (Debian)

Figure 3.2: Process flow for the Pedagogical Framework tool.

assignment to the students for troubleshooting. This will provides the environment for practice and getting troubleshooting skills. Students have to fix virtual networks and bring the virtual machines on working state. Afterwards, the teacher will reset all virtual networks through suggested tool for new new students or for new attempts by students having trouble solving a particular problem.

3.4 To Build Physical Server

We select an operating system and supporting software to build the physical server. Theoperating system we used Debian Squeeze 6.0.7v. After installing an operating system we installed XEN and MLN through the command line. Before making an automatic installation through tool we want to make sure that each command is working properly, so for this we used command line to execute each command one by one.

3.4.1 Physical Server Specification

To configure the server for experiment the following specification.

Operating System	Debian 6.0.7 (Squeeze)
Model Name	Intel(R) Xeon(TM)
Processor	CPU 3.00GHz
Cache Size	1024 KB
CPU Cores	1
Memory	2090MB
HDD	129GB

Table 3.1: Server's Specification

3.4.2 Debian Squeeze

We used Debain as an operating system but there are many others operating systems in the market. We select the Debain because it has good integration with XEN

3.4.3 XEN

It is freeware and easily available. There are other virtualization software offerings, but some of them are not freely available. VMware is one of them and it is partly free but not full. First of all we would like to say that there is no best solution because it all comes down on what yours demands are, needs to fulfill your practical work as you wish. During our search we found plentify documentation of XEN, making it very supportable. There are few more reasons to select it like, XEN is free, enterprise ready, and almost have same features as KVM, Hyper-V, and VMware esx server. Users can easily build a quite impressive virtual infrastructure through XEN.

3.4.4 XEN Installation Steps

The following steps demonstrate the specific commands used to build the XEN server environment.

- apt-get -y install openssh-server
- aptitude -y install xen-hypervisor-4.0-amd64 xen-linux-system-2.6-xen-amd64 linux-image-xen-amd64 xen-qemu-dm-4.0 xen-tools xen-utils xen-watch xenstore-utils
- mv /etc/grub.d/10_linux /etc/grub.d/21_linux
- nano /etc/default/grub # Make a changes like this
  - GRUB_DISABLE_OS_PROBER=true
  - GRUB_CMDLINE_XEN="dom0_mem=1024M"
- update-grub
- nano /etc/default/xendomains
  - XENDOMAINS_RESTORE=false
  - XENDOMAINS_SAVE=""
- nano /etc/xen/xend-config.sxp
  - enable-dom0-ballooning no
  - network-script 'network-bridge antispoof=yes'
- reboot

3.4.5 MLN

MLN is new technology to build virtual networks according to your own wish. It has very interesting features like you can build many virtual networks on single server with MLNs strong scripting language.

3.5 MLN Installation Steps

The following steps demonstrate the specific commands used to build the MLN environment.

- apt-get install -y uml-utilities bridge-utils screen sudo
- wget https://www.dropbox.com/s/iudlu6ztmyo2qgx/mln
- chmod +x mln
- ./mln setup # Follow the Default Values Just Enter at each option
- cp mln /usr/local/bin/
- chmod +x /usr/local/bin/mln
- nano /etc/mln/mln.conf # Made Following changes into the file
  - default_memory 512M
- apt-get install -y user-mode-linux uml-utilities
- wget https://www.dropbox.com/s/gzu0gwjrfj0nup9/l enny_small_backports.ext3
- cp lenny_small_backports.ext3 /opt/mln/templates/
- export PATH=PATH:/usr/local/bin
- mkdir /opt/mln/projects
- echo ”projects /opt/mln/projects” ¿ /.mln
- echo ”templates /opt/mln/templates” ¿ /.mln
- echo ”files /opt/mln/files” ¿ /.mln
- mln register_template -m ”The dummy template” -t /opt/mln/templates/lenny_small_back
- mln list_templates

3.5.1 MLN Script

Using an MLN script we built a virtual network with 4 virtual machines. Through MLN network administrator can create different virtual networks with different virtual machines on single server or on distributed servers. MLN script is simple and easy to understand.

Figure 3.3 is the view of MLN script, which we used in this project to build a virtual network with one gateway machine and three virtual machines. There are two switches in this network. When we installed XEN on physical server than it create the XEN switch with the name switch eth0. Second switch in the network create by MLN with the name switch lan. In this network port forwarding to allow the Internet traffic inside and outside of the virtual network.

```
global {
project VNetwork3
}
superclass common {
xen
template lenny_small_backports.ext3
term screen
memory 512M
nameserver 128.39.89.8
network eth0 {
netmask 255.255.255.0
}
users {
root IySSSbVsUUmlI # user root has encrypted password that is 123
}
}
```

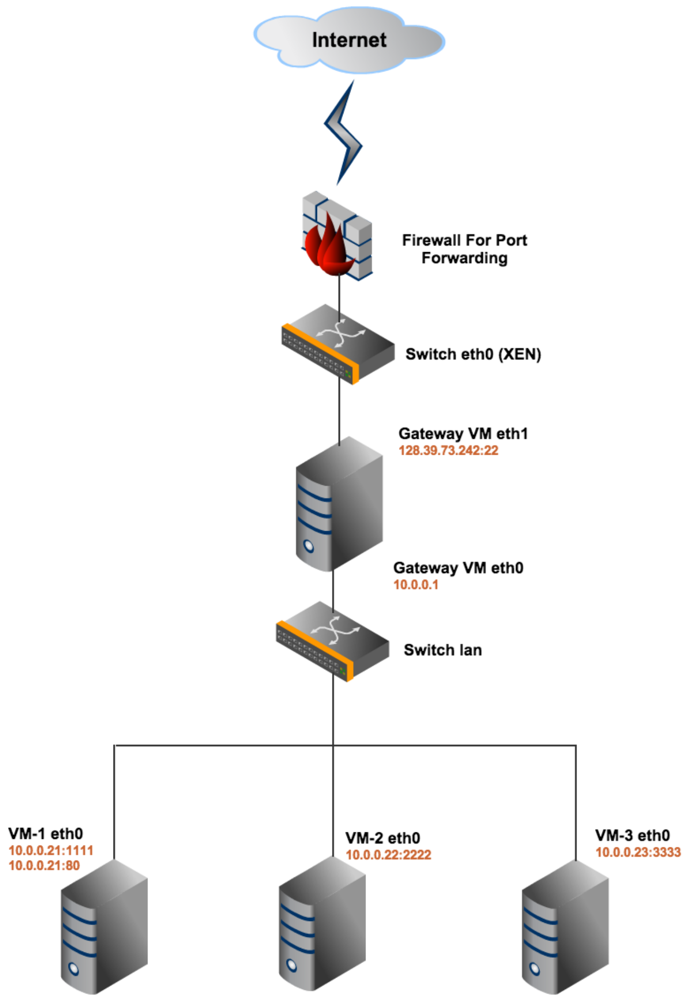


Figure 3.3: Virtual network created through MLN script.

```
host gateway {
superclass common
network eth1 {
address 128.39.73.242 # Public IP address to gateway machine
netmask 255.255.255.0
gateway 128.39.73.1
switch eth0
}
network eth0 {
address 10.0.0.1
switch lan
}
startup {
depmod
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth1 -p tcp –dport 1111 -j DNAT –to-destination 10.0.0.21:22
iptables -t nat -A PREROUTING -i eth1 -p tcp –dport 2222 -j DNAT –to-destination 10.0.0.22:22
iptables -t nat -A PREROUTING -i eth1 -p tcp –dport 3333 -j DNAT –to-destination 10.0.0.23:22
iptables -t nat -A PREROUTING -i eth1 -p tcp –dport 80 -j DNAT –to-destination 10.0.0.21:80
depmod
echo 1 ¿ /proc/sys/net/ipv4/ip_forward
}
```

```
}
host VM-1 {
superclass common
network eth0 {
address 10.0.0.21
gateway 10.0.0.1
switch lan
}
}
host VM-2 {
superclass common
network eth0 {
address 10.0.0.22
gateway 10.0.0.1
switch lan
}
}
host VM-3 {
superclass common
network eth0 {
address 10.0.0.23
gateway 10.0.0.1
switch lan
}
```

}

switch lan {}

Source: http://mln.sourceforge.net/doc/mln-manual.pdf

Basically we use the template of MLN script from MLN manual and add other features according to our requirements. In this script we create one gateway with public IP address (128.39.73.242) and three virtual machines with static IP address. All these virtual machines are connected through virtual switch lan. On gateway we implement port forwarding because to provide an Internet access to all virtual machines. If you want to increase the number of virtual machine than you can do it through copy the one virtual machine configuration with another name and assign new IP address to it.

Create the MLN script with .mln extension in /opt/mln/files/ folder. The name of the project is VNetwrok3. MLN has the manual commands to build the virtual network, to stop and to check the virtual network status as follow.

To build the project use

```
mln build f VNetwork3.mln
```

To start the project use

```
mln start p VNetwork3
```

To check the status of project than use

```
mln status p VNetwork3
```

To stop the project use

```
mln stop p VNetwork3
```

3.5.2 Programing Language

Programming languages like C, C++, and Java allow the programmer to write detailed code with good execution speed. But if one would like to write a code on high level just to develop an application like text manipulation applications, the basic unit in C and C++ is character but in the Perl language the basic unit is general text. C and C++ have better speed but if speed is not matter than programmer like to convenience of scripting language to save the time and get high performance. To develop a prototype we have to select one programing language. We mentioned in introduction chapter that we would develop a prototype so we select Perl Scripting to do this.

3.5.3 Perl Scripting

Perl is very feature-rich language [19]. There are few reasons to select the Perl scripting language in our thesis are as follows. The goal is to develop a rapid prototype network lab management tool for teaching and learning troubleshooting skills in the labs.

- Used often for system administration and rapid prototyping.
- Very casual with regard to typing of variables, e.g. no distinction between integer, oating-point or string variables. Functions can return nonscalars, e.g. arrays, nonscalars can be used as loop indexes, etc.
- Lots of high-level operations intrinsic to the language, e.g. stack push/pop.
- Interpreted, rather than being compiled to the instruction set of the host machine. [19]

Chapter 4

Results: Pedagogical Framework

This chapter describes the output of this thesis. We divide the results into two chapters. In this chapter we focus on our suggested pedagogical framework. There are three parts of this chapter: The first has information about Teaching and Learning Cycle, the second shows the scenario of Teaching and Learning Cycle, and the third section gives the brief description of tool processes.

According to the problem statement our results depend on two parts. In first part we show pedagogical framework. Second part we describe the output of our tools described from framework. For ease of understanding first we write this chapter to build a holistic picture of our results.

4.1 A Pedagogical Framework

Pedagogical framework will help the student to do actual network-troubleshooting practices. It will also highlight the how to build virtual networks and get automation feedback from the network lab management tool. Through this the teacher provides the amount of support necessary to ensure that new learning occurs. Teachers want to teach latest skills to their students, because they want that their students can solve the critical problems related to network troubleshooting without facing any difficulty. This will require additional work to create new and relevant problem templates.

Our whole thesis depends on the pedagogical framework in which we describe different processes to provide a learning environment. Each process is interconnected with another process. This framework is the student learning cycle, developed to ensure students learn how to manage and repair systems. Through this students will get troubleshooting skills and feedback from the teacher, while also enabling the teacher to focus on students individual problems.

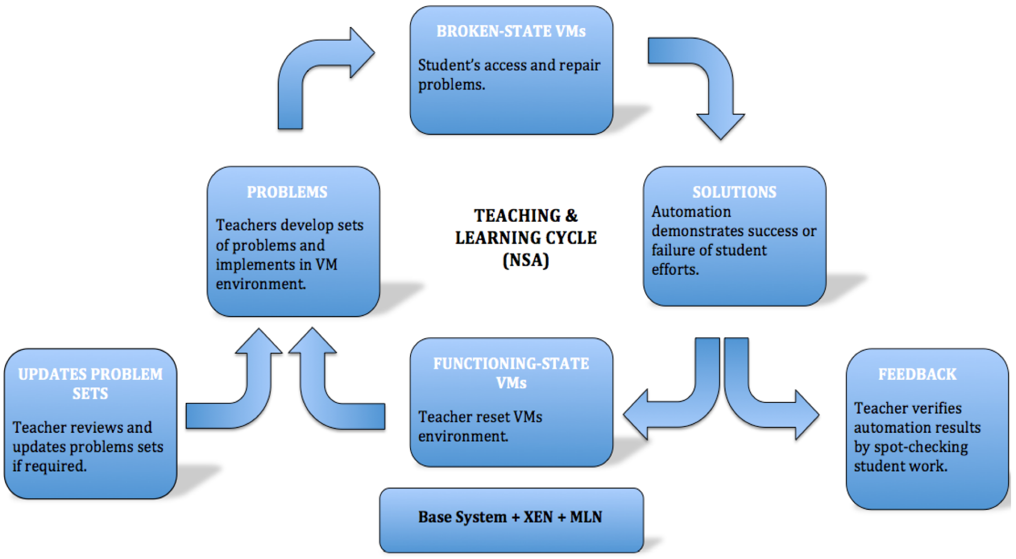

BROKEN-STATE VMs
Student's access and repair problems.
PROBLEMS
Teachers develop sets of problems and implements in VM environment.
TEACHING & LEARNING CYCLE (NSA)
SOLUTIONS
Automation demonstrates success or failure of student efforts.
UPDATES PROBLEM SETS
Teacher reviews and updates problems sets if required.
FUNCTIONING-STATE VMs
Teacher reset VMs environment.
FEEDBACK
Teacher verifies automation results by spot-checking student work.
Base System + XEN + MLN

Figure 4.1: Teaching and Learning Cycle Diagram for Networks and Sys Admin

The teaching and learning cycle describes the process by which teachers use configured virtual networks to implement the pedagogical framework to produce new learning.

4.2 Pedagogical Framework Processes

This learning cycle has 7 stages. Each process is interconnected with another.

1. Base System (e.g. Debian) with XEN and MLN
2. Functioning State of VMs (Teacher reset VMs environment)
3. Problems (Teachers develop sets of problems and implements in VM environment)
4. Broken State of VMs (Students access and repair problems)
5. Solutions (Automation demonstration success or failure of student efforts)
6. Feedback (Teacher verifies student and advises students)
7. Updates the problems sets (Teacher reviews and update problem sets if required)

4.2.1 stage 1: Base System + XEN + MLN

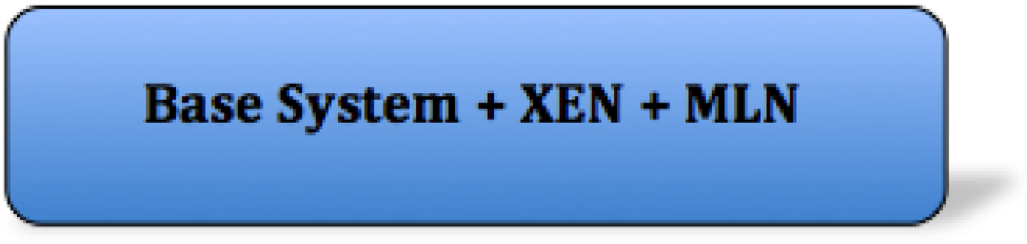


Figure 4.2: First stage of pedagogical framework for teaching and learning cycle

Figure 4.2 shows the first stage to build an automation process for the teaching and learning cycle. First install the base system (Operation System) with Perl package. After this install XEN and MLN to build a virtualization environment and associated virtual networks. We used MLN script to create virtual machines because this is very handsome and efficient tool. For ease of user the tool should install XEN and MLN automatically or on user demand.

4.2.2 Stage 2: Functioning State of VMs

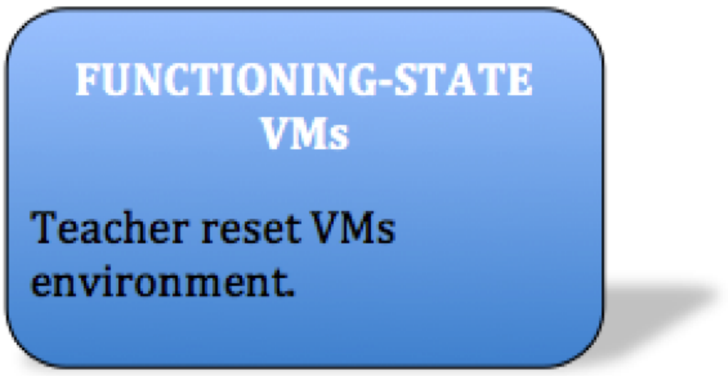


Figure 4.3: Second Stage Functioning State of VMs.

After building the base systems with XEN and then create virtual networks. The management tool will create the virtual networks automatically or the user can create the virtual networks by command line.

Teachers have two steps here, the first time they will create the networks, and the second time they will only reset the virtual networks through management tool, which we designed.

4.2.3 Stage 3: Problems

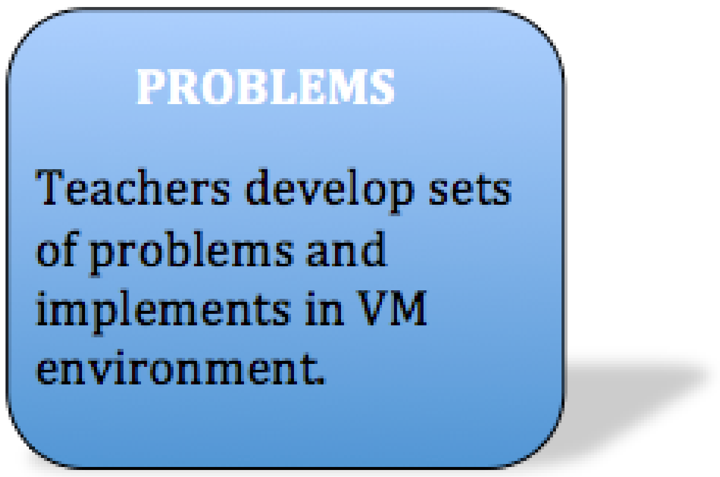

PROBLEMS
Teachers develop sets of problems and implements in VM environment.

Figure 4.4: Third Stage Assign Problems to the VMs

Convert the functioning virtual machines into broken state. Here teacher will select a particular virtual machine and assign different problems to it, changing from functioning to broken state. Now assign these networks to the students as an assignment to troubleshoot and repair.

4.2.4 Stage 4: Broken State of VMs

When virtual machines are converted from functioning state to broken state, they are ready to be exposed to students for assessment and repair. This practice helps to the students to explore and develop critical troubleshooting skills. Learning is our main goal and in this phase students will learn how to repair the virtual machines. This practice will help the students in their practical life, because as a system administrator he will face many systems having different kinds of problems.

BROKEN-STATE VMs
Student's access and repair problems.

Figure 4.5: Fourth Stage Broken state of VMs

4.2.5 Stage 5: Solutions

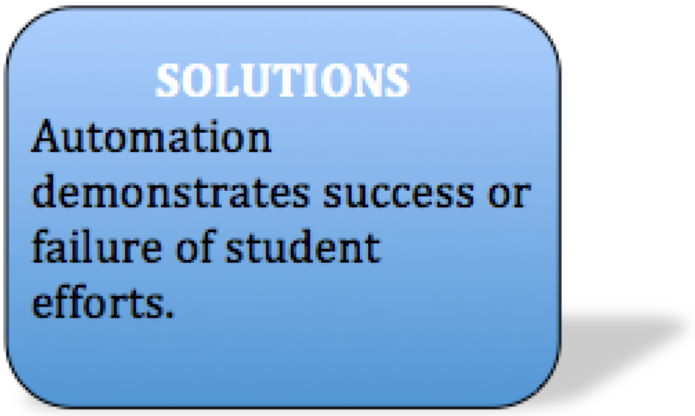

SOLUTIONS
Automation demonstrates success or failure of student efforts.

Figure 4.6: Fifth Stage - Solutions

In this phase teachers get the reports from students, either they get success to repair the virtual machines or not. Management tools should automatically demonstrate success or failure of student efforts as a check and balance. It should give full details with feedback of students reports. This will reduce the teacher processing time and it makes the whole process efficient. The automation allows increased teacher-student time.

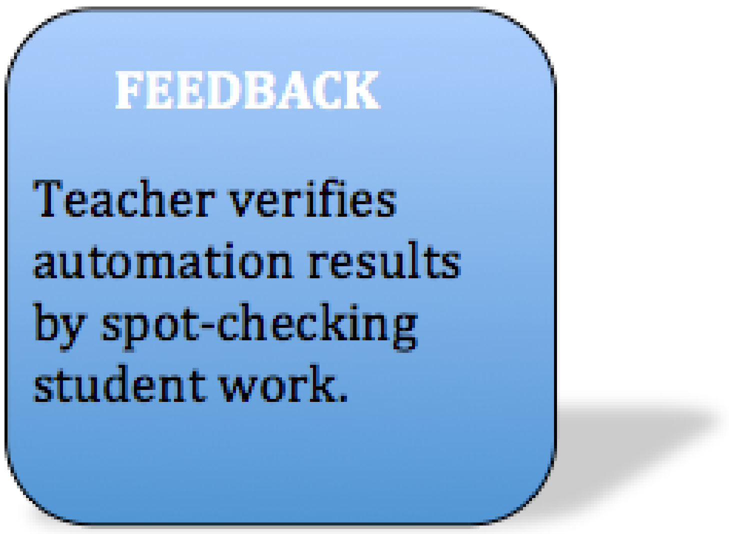

FEEDBACK
Teacher verifies automation results by spot-checking student work.

Figure 4.7: Sixth Stage Teachers feedback

4.2.6 Stage 6: Teachers Feedback

After submitting the reports teachers verify the automation results by spot-checking student work. In this way teachers give the feedback to the students and point out their mistakes as well. Teachers should double-check each virtual machine manually and their status. Learning processes complete on this level. Now teachers will reset all virtual machines into their functioning state where the first process started.

4.2.7 Stage 7: Updates problem sets

Our intention is to build a pedagogical framework that will fulfill the requirement for a predictable teaching tool. In the future there will be lot of other problem sets. So our framework expects that teachers will be able to add new and latest problem sets in management tool. Teaching and Learning Cycle is not designed just for recent requirements and currently-known problems. In the future if teachers want to add more problems than they would do it.

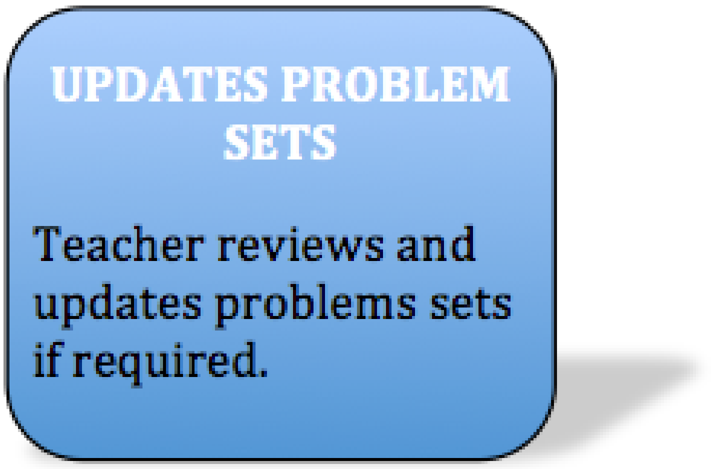


Figure 4.8: Seventh Stage - Updates problem sets

4.3 Logical Diagram-Pedagogical Framework

Figure 4.9 is derived from Teaching and Learning Cycle. To show the scenario in which Teaching and Learning Cycle can be implemented. On the base system the XEN and MLN will be configured to provide virtualization environment. With the help of MLN script you can create virtual networks. Each network has different number of virtual machines to communicate with each other to create a complex environment.

Reference figure 4.9, in the MLN environment you can find two networks and each network has its own switch. Green color shows that these networks are in proper working condition. Now management tool should make some problems to convert into broken state. When virtual machines are converted into broken state than their color has changed from green to orange in the flow diagram. This color shows the broken state of virtual machines. Now virtual machines are ready to be assigned to the students. Students have assignment to repair the virtual machines. In this way some of them successfully or all fix the virtual machines but may be some cannot do this. So teacher will verifies all virtual machines using automation and manual assessment and gives his feedback or assessment. The teacher has three options here, first he will give the feedback regarding to students work. Second if he can add new problem sets into the management tool according to the current requirements. In third option he will reset all virtual machines into functioning state.

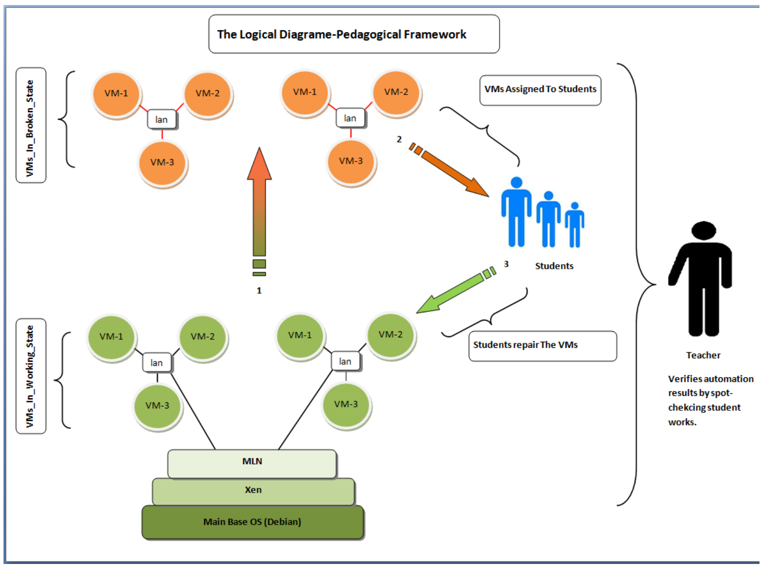


Figure 4.9: Logical Diagram-Pedagogical for administration skills.

4.4 Management tool flow processes

We have mentioned in chapter first to build a prototype according to pedagogical framework. Prototype is good way to improve either suggested framework is applicable or not. The procedure of prototype is as follow.

From first step the tool should install the XEN and MLN tools automatically and after that it should build a virtual networks on MLN platform. When virtual network will be created than teachers can assign the set of problems to the selected virtual machine, and this will convert the VM into broken state. Now teachers can assign each network to the students to repair the network and VMs because this is assignment for students. Some of them will be able to do this and may be some not.

Through management tool teacher can fix all those problems that were created in second step. There are two more things, network lab management tool should capable to add new problem sets in current problem list and second one it should give automatically demonstrates success and failure of student efforts.

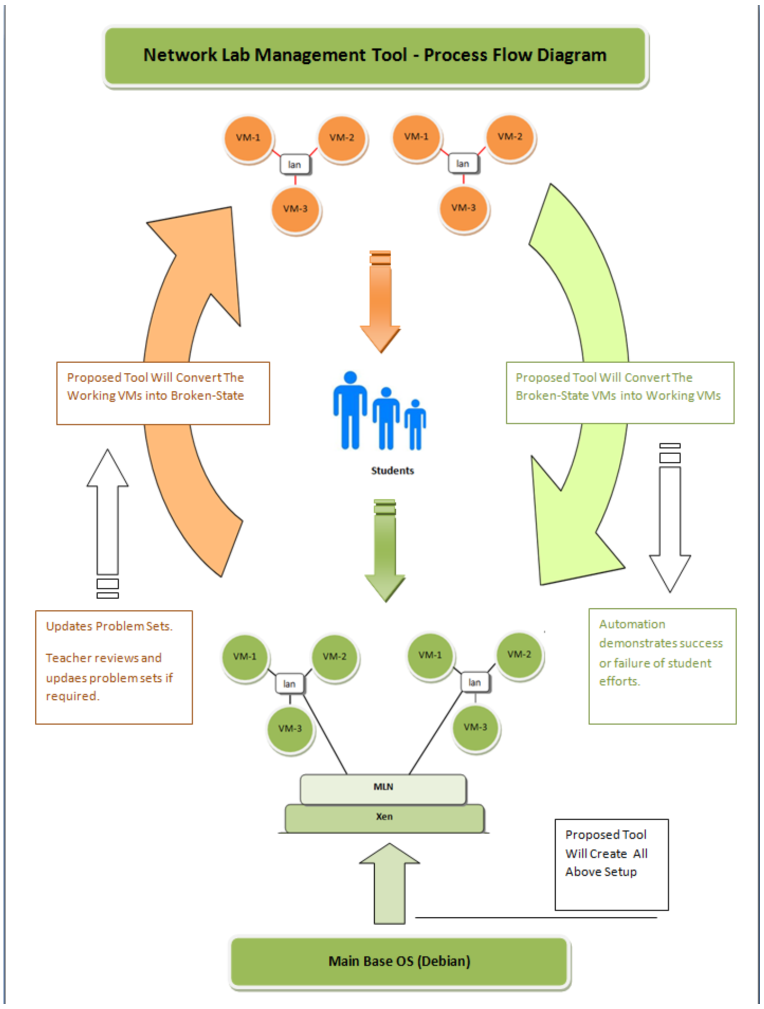

Network Lab Management Tool - Process Flow Diagram
VM-1
VM-2
lan
VM-3
VM-1
VM-2
lan
VM-3
Proposed Tool Will Convert The Working VMs into Broken-State
Students
Proposed Tool Will Convert The Broken-State VMs into Working VMs
Updates Problem Sets.
Teacher reviews and updaes problem sets if required.
VM-1
VM-2
lan
VM-3
VM-1
VM-2
lan
VM-3
Automation demonstrates success or failure of student efforts.
MLN
Xen
Proposed Tool Will Create All Above Setup
Main Base OS (Debian)

Figure 4.10: Procedure of our prototype Management Tool.

Chapter 5

Results: Network Lab Management Tool

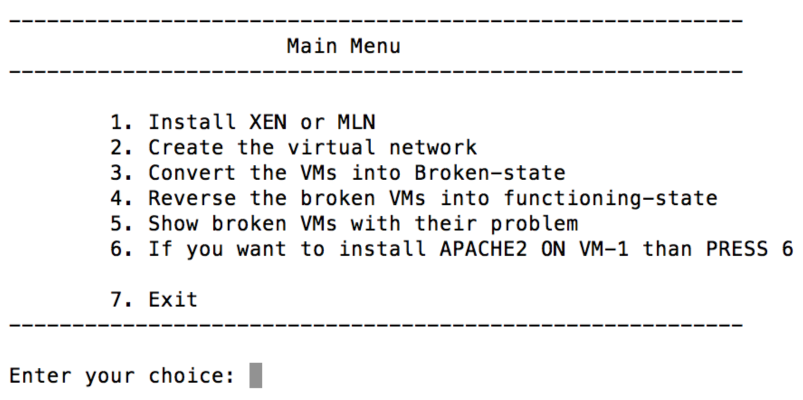
Here we describe the network lab management tool with a practical view of how the tool works and what kind of options it has. In this chapter we list few problems with solution to make clear picture to our readers.

This chapter contains the outputs of our tool that we developed. According to the problem statement we develop a prototype tool in which we implement our idea, how to build a physical server having Operating System with XEN, MLN, and Perl. In our tool we involved the user input as well and he can choose different options at different levels.

Our suggested tool may be useful to different kind of users; especially it may helpful to the network teachers to develop a virtual network on a single server. Teachers should check the status of virtual machine before creating any problem in it. After broking the virtual machines network administrator can assign these virtual machines to the students as an assignment. Students have to fix all virtual machine and check the virtual machine performance as well. This tool would be useful to a person who has limited knowledge about XEN or MLN; they can easily build a virtual network easily without any trouble.

5.1 Main Menu

Now this is a time to demonstrate how our network lab management tool functions. Figure 5.1 is the main menu in which seven options are there. Option 1, 2, 3, and 4 has further sub options.



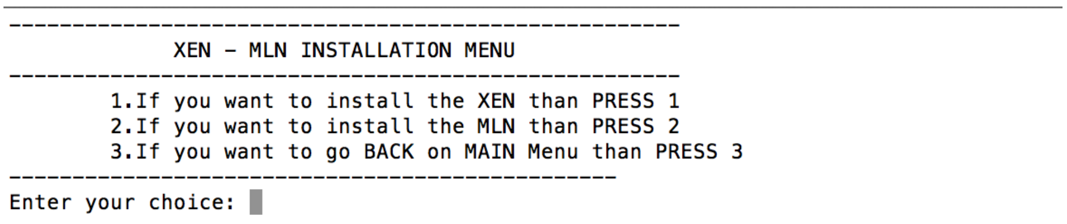
```
----------------------------------------------------------
                      Main Menu
----------------------------------------------------------

        1. Install XEN or MLN
        2. Create the virtual network
        3. Convert the VMs into Broken-state
        4. Reverse the broken VMs into functioning-state
        5. Show broken VMs with their problem
        6. If you want to install APACHE2 ON VM-1 than PRESS 6

        7. Exit
----------------------------------------------------------

Enter your choice: █
```

Figure 5.1: Main Menu of Proposed Tool

Only user select the option then next page will open for further selection. We try to make it simple and easy for better understanding.

5.2 Install XEN or MLN automatically

Now we will explain each option and its sub-option here. When the user selects the first option (Install XEN or MLN) then it opens new page with three options. Figure 5.2 is the sub-option of first option. If you want to install the XEN tool than PRESS 1, or if you want to install the MLN tool than PRESS 2, otherwise PRESS 3 to go back on Main Menu. It will install both (XEN and MLN) on base system (only operating system with Perl). Now users can install these tools properly within 2 minutes. In Apendex you can find the perl script which is use to develop the Network Lab Management Tool.



```
-----------------------------------------------------
             XEN - MLN INSTALLATION MENU
-----------------------------------------------------
        1.If you want to install the XEN than PRESS 1
        2.If you want to install the MLN than PRESS 2
        3.If you want to go BACK on MAIN Menu than PRESS 3
------------------------------------------------
Enter your choice: █
```

Figure 5.2: Sub-Menu, Selection To Install XEN or MLN

5.3 Create the Virtual Network

When XEN and MLN will be installed than user has option to build a virtual network(s) according to the provided options. Figure 5.3 shows four options for selection. In this tool we only active the option first Create a virtual network with 3 VMs when user will select the first option than it will automatically build a virtual network with 3 VM and one gateway within 2 minutes. If you dont want to build a virtual network than you can press 4 to go back to the Main menu.

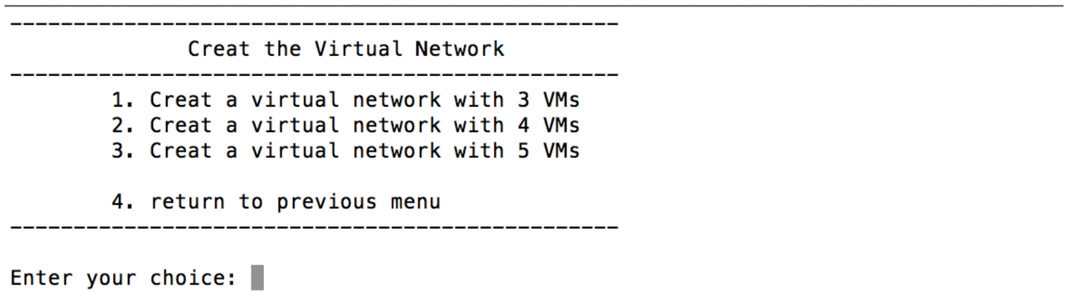

```
------------------------------------------------
              Creat the Virtual Network
------------------------------------------------
        1. Creat a virtual network with 3 VMs
        2. Creat a virtual network with 4 VMs
        3. Creat a virtual network with 5 VMs

        4. return to previous menu
------------------------------------------------

Enter your choice:
```

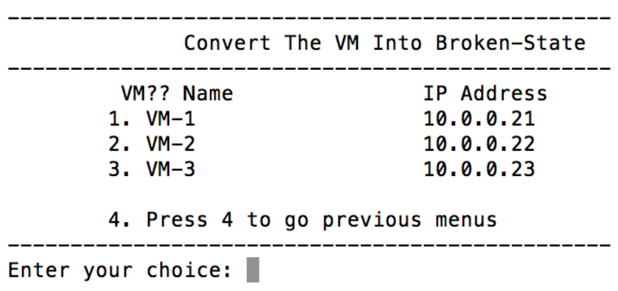
Figure 5.3: Sub-Menu, Selection To Create Virtual Networks

5.4 Convert the VM into Broken-State

According to the pedagogical framework we mentioned in it that suggested tool should have the capability to convert the functioning virtual machines into broken state. Here the user will select the particular virtual machine to assign different kind of problems, which are shown in the next menu level. In this proof of concept tool, the user can select only one option here. Figure 5.4 shows the list of virtual machines to be selection for problems.

5.5 Select Problem(s) For Virtual Machine

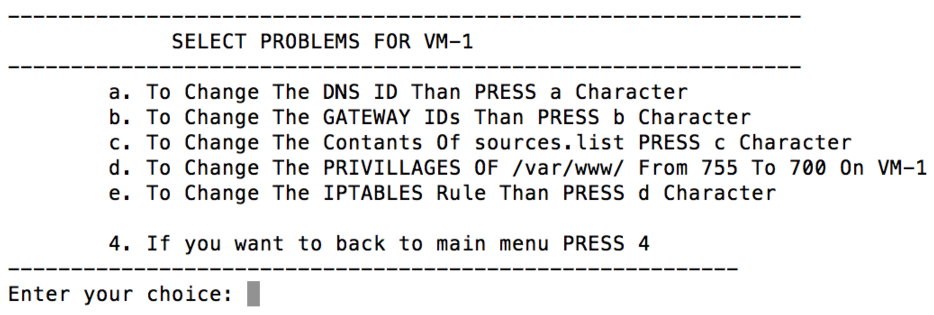
Figure 5.5 shows the list of problems; user can assign more than one problem to any virtual machine from previous list. If user selects any problem than on



```
-------------------------------------------------
              Convert The VM Into Broken-State
-------------------------------------------------
         VM?? Name                IP Address
        1. VM-1                   10.0.0.21
        2. VM-2                   10.0.0.22
        3. VM-3                   10.0.0.23

        4. Press 4 to go previous menus
-------------------------------------------------
Enter your choice: 
```

Figure 5.4: Sub-Menu, Selection To Convert The VM into Broken-State

the spot it will assign to the selected machine. On the screen users can see the progress how to assign the problems and confirmation about assigning the problem successful held.



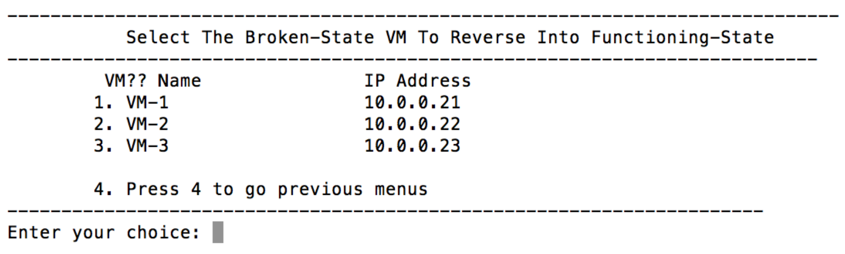
```
---------------------------------------------------------------
             SELECT PROBLEMS FOR VM-1
---------------------------------------------------------------
        a. To Change The DNS ID Than PRESS a Character
        b. To Change The GATEWAY IDs Than PRESS b Character
        c. To Change The Contants Of sources.list PRESS c Character
        d. To Change The PRIVILLAGES OF /var/www/ From 755 To 700 On VM-1
        e. To Change The IPTABLES Rule Than PRESS d Character

        4. If you want to back to main menu PRESS 4
----------------------------------------------------------
Enter your choice: 
```

Figure 5.5: Sub-Menu, Select Problems To Assign the VM-1

5.6 Select the Broken-State VM To Reverse Into Functioning-State

Figure 5.6 is sub option from Main Menu where user select 4th option Reverse the Broken VMs into Functioning state. Here user has option to select one broken VM to convert into functioning state. After this selection virtual machine comes into functioning state.

In the Main Menu there are two more options (5th and 6th), through option 5 you can print out the log file where you can find virtual machine name and



```
-----------------------------------------------------------------------------
           Select The Broken-State VM To Reverse Into Functioning-State
---------------------------------------------------------------------------
         VM?? Name                IP Address
        1. VM-1                   10.0.0.21
        2. VM-2                   10.0.0.22
        3. VM-3                   10.0.0.23

        4. Press 4 to go previous menus
----------------------------------------------------------------------
Enter your choice: █
```

Figure 5.6: Sub-Menu, Select The VM To Reverse Into Functioning-State

its problems with date. Here you can see the problems solution of particular virtual machine with date. Option 6 in Main Menu has functionality to install the APACHE2 on VM-1.

5.7 Problem List

We can make many problems into the virtual machine, but here we are mentioning few problems to demonstrate our framework procedure.

5.7.1 Problem 1: Plug off network cable

Background:

As a system administrator, we are used to access our system remotely by remote desktop or VNC. So it is quite easy for us to ignore the physical level issues. This problem is trying to simulate a network failure by plug off the cable from our server. One needs to fix this one before any further steps.

Problem approach:

Plug off network cable from server.

Problem results:

User will unable to connect to the entire system.

Repair method:

Plug in network cable back to server.

5.7.2 Problem 2 DNS problem

Background:

The problem 1 where the user is reporting trouble-accessing Internet from the new Debian desktop is a DNS problem we created. The problem was really easy to create and is a common problem that occurs often in real life. This is the reason we created the problem so it would be more close to a real life situation. This might be difficult to catch at first sight because the system administrator can ping the machine from the Debian squeeze server. DNS is a common networking issue and as a Sysadmin it is important to have a basic understanding of what it does and how it works. In order to create this problem we changed the nameserver(s) in /etc/resolv.conf file on virtual machine.

The problem was created with opening the /etc/network/interfaces and removing the dnsnameservers from the file. After this was done we restarted the service and the problems started to occur with the Internet.

Solution:

To fix the problem again is basically changing back what we removed from the interfaces file. First enter the file with the command:

jed /etc/network/interfaces
The add the following line to the bottom of the file:
dnsnameservers 128.39.89.8 128.39.74.66
Now save the /etc/network/interfaces and restart the network through following command.
/etc/init.d/networking restart Now the machine should have Internet access one again.

5.7.3 Problem 3: Change in source.list file

Background:

When system install new packages than It use the source list where it fetch the packages. Each country has its own source list paths. Where users can download all packages. If there is no source path in the source.list file than system will not download the packages for installation. Made a change in source.list file than system will show the failure message to download the package. Make a comment to all paths in source.list file, so system will not able to find the exit path. It creates a problem to download any package from internet.

Solution:

Uncomment all lines in source.list and refresh the system. Now system will find the exact path to download the particular package to install it.

5.7.4 Problem 4:Change the File Permissions for /var/www/

Background:

In an organizations web server is very important to launch the web site. Apache is web server to install on Linux platform as well. Mostly the web traffic uses the port 80 on Internet. If the apache folder /var/www/ has changed privileges the web server service will not start.

Problem:

In the Linux machines the Apache web server installed in /var/www/ directory. By default the privileges of this directory is 755 set. Now change the privileges of this directory from 755 to 700 then the httpd user cannot access the apache server from internet. Following command is used to change the privileges.

```
chmod 700 /var/www/
```

Solution:

According to the problem the solution is to reset the privileges of /var/www/ directory onto default privileges with following commands

```
chmod 755 /var/www/
```

5.7.5 Problem 5: Nagios File Error

Background:

Nagios system is a client-server structure. The server side is usually more stable compare with client side. Since the access privilege to server is quite limited to few users. But client server has less control compare with center server. It is quite normal that the Nagios report experiences failure caused by client administrators operations. We are trying to simulate a typo which may be made by a client administrator, resulting in the Nagios administrator having unknown status for that client.

Problem approach:

Modify nrpe.cfg, vi /usr/local/nagios/etc/nrpe.cfg
Change command[check_load]=/usr/local/nagios/libexec/check_load -w 15,10,5 -c 30,25,20
To command[check_load]=/usr/local/nagios/libexec/check_lood -w 15,10,5 -c 30,25,20

Problem result:

Nagios administrator will spot unknown status for test-client from web console.

Repair method:

Modify nrpe.cfg, vi /usr/local/nagios/etc/nrpe.cfg
Change command[check_load]=/usr/local/nagios/libexec/check_lood -w 15,10,5 -c 30,25,20
To command[check_load]=/usr/local/nagios/libexec/check_load -w 15,10,5 -c 30,25,20

5.7.6 Problem 6 Disk usage

Background:

Using a command to fill up the remaining disk space on the Debian server created this problem. The main purpose of this problem is to simulate that there is no more disk space to work on which can happen in real life if you download a large file or a script malfunctions.

First we checked available disk space with df h and we saw that it was about 23 GB available. To create the file that consumes disk space we used the command:
fallocate l 60G /var/bigassfile.file where 60G = 60GByte.
This creates a file that fills up the remaining pace on the disk. You are now unable to create new or edit files in your system, which is a problem for everybody.

Solution:

The first thing to finger out which disk is full. This is done with the command:
df h

The next step is to find the specific file or files that is taking up large space on the system. To find the file we have created this command to search through the system and displaying file over 1000000 Kb. The command can be altered to search specific part of the system or the size files. Since our system is not that large it works searching the whole system and you get an answer within 20 seconds.

find type f size 1000000k exec ls lh {}

If the file has been found in the system it can be removed. This is done with the command:
rm /var/cruptfile.file
Now the file is removed and it is possible to return to normal business.

5.7.7 Problem 7: Change Firewall IPtables Rules

Problem creation:

We wanted to make the apache server unavailable, so it was decided to implement errors in the firewalls. The first one is to change the forwarding port into firewall rule, which redirects all HTTP traffic to and from the server. The second one to changes the IP address from 10.0.0.21 to 10.0.0.35 which makes the firewall forward the packets to the wrong IP and makes the web server inaccessible.

The manifestation of this problem will make it unable to connect to web server as well as the other services. Since the local Iptables firewall will block all traffic.

Problem fix:

The first problem can be fixed by correcting the port forwarding from 8080 to 80 to the apache server with correct IP address of the apache server 10.0.0.21 in the firewall rules. This will allow the web traffic from Internet to web apache 10.0.0.21. The second part of the problem can be solved either by changing the default policy to accept or rebooting the machine. The default policy can be changed in the command line with root permissions by typing the command:
iptables -P inputs ACCEPT.
The machine should now be able to receive traffic.

5.7.8 Problem 8: Change Nagios File Privileges

Problem creation:

The problem concerns the Nagios monitoring system. We decided to change permissions on the index.php file to make it inaccessible, but not unreachable. The default permissions on the index.php file are set to 644 but are now changed to 000. To do this, use the command chmod 000 index.php in the nagios3 directory.

Problem manifestation:

This problem will cause the main page of the Nagios interface to crash with the HTTP Error 500. The front page of the interface will not be able to view before permissions are reset. The problem will only affect the front page of the Nagios server, nothing else.

Problem fix:

To fix this problem the permissions need to be changed back to 644. Running the Nagios test configuration will not detect this, but by checking the /var/log/apache2/error.log file Apache will detect errors with the permissions and point to the correct file, which is the index.php.

5.7.9 Problem 9: Bash

Background:

This problem is very much connected with the users knowledge of Bash. On server 3 the user will experience a problem with navigating the directories. He will see himself always getting back to his home directory. This is due to a Bash variable PROMPT COMMAND that is set to cd.

Whatever is in the PROMPT COMMAND variable is run after the users command. To set up this problem:
echo ”PROMPT_COMMAND=cd” $\gg$ *.bashrc*

Solution:

Remove the last line of .bashrc

5.7.10 Problem 10: Wiki and .htaccess

Background:

Server is running the web server serving the main page with links to the different tools. This one will now present a basic authentication box asking for username and password. This is done through .htaccess.

Solution:

Create a .htaccess file in /var/www. Username/password: lab/labROcks Changed /etc/apache2/sites-available/default Set AllowOverride None to AllowOverride AuthConfig on /var/www To fix it: Remove .htaccess

The main reason to show some problems is to clear the picture about what we want to develop an environment in which students get troubleshooting skills to solve different problems.

Chapter 6

Analysis and Discussion

This chapter describes the results and discusseslimitations and difficulties encountered. Future work relevant to this thesis is also discussed. The data obtained from the previous chapters will be briefly described here. All results will be analyzed for their validity and significance. This will end up with discussion on what types of difficulties there were during this thesis and further future works related to pedagogical framework and network lab management tool. Two types of results produce from problem statement. The first part describes to analyze the pedagogical framework and second part describes to analyze the network lab management tool.

6.1 Pedagogical Framework VS Problem Statement:

The problem statement is to develop a pedagogical framework that helps to build an environment in which teachers and students participate. The main target of the problem statement is learning troubleshooting skills in networks and system administration. To accomplish problem statement a pedagogical framework created with some examples.

The pedagogical framework focuses how to learn the troubleshooting skills in network lab environment. Creating virtual networks take significant time to configure them. Our focus is to create a framework in education sector. Students will get experience and teachers will manage it. Through this framework teachers will save their time and give efficient feedback to the students. There are seven stages in the pedagogical framework and each has full specification.

6.1.1 Stage 1: Base System + XEN + MLN

The first step of pedagogical framework to build a base system through name you can understand it meaning. On physical server install operating system (Debian 6.0.7) with XEN and MLN, this makes a virtualization environment for virtual networks. For practical work following specifications used to build a physical server.

Operating System : Debian 6.0.7 (Squeeze)

Model name : Intel(R) Xeon(TM)

Processor : CPU 3.00GHz

Cache size : 1024 KB

CPU cores : 1

Memory : 2090MB

HDD : 129GB

Practical work held on above specification but not the other specification just because the purpose is to develop the prototype on framework. One can do the same practical on other hardware and software specification. Only Linux platforms were used but not Windows platforms due to time constraints.

XEN and MLN provide the virtualization platform, to create virtual machines and virtual switches. There are other tools available in the market to provide the virtualization environment like KVM, VMware, and Proxmox. These tools are open source and easily available on Internet. While configuring the MLN lack of documentation and some configuration commands missed. KVM, VMware, and Proxmox were not use because of limited time to finish the practical work in thesis.

6.1.2 Functioning State of VMs (Teacher reset VMs environment)

When the base system ready with XEN and MLN than teacher can create the virtual networks and all of them should work properly. In the teaching and learning cycle this is first step because when the virtual machines will repair than virtual networks come again on this step again. On the step two options can avail, one of them is to repair all VMs and second one to build new virtual networks again. Second options would be good option because when teacher assign the broken virtual machines to students for repairing, may be they will crash the whole machine during repairing. So it would be better to create new networks and it takes less than 2 minutes to create 4 virtual machines.

6.1.3 Problems (Teacher's defined problems in VM environment

After creating new virtual networks teachers will assign the problem sets to VMs just to convert the state of functioning VMs into broken state. You can assign one or more than one problem to any particular machine. All these processes will be held automatically so no need to be worried how you can do this. Virtual machines has Linux operating system so all problems are related to Linux. To check either this framework is applicable for Windows need future work as well.

6.1.4 Broken State of VMs (Students access and repair problems)

Now virtual machines are get broken state and ready for repairing. Students have assignments to repair the virtual machines. Through this exercise students can get the troubleshooting skills. Pedagogical framework designed for both environments (Linux and Windows). Because of short time only one platform (Linux) used in practical work.

6.1.5 Solutions(Automation success/failure of student)

According to the figure 4.1, when students submit their reports or finished to repair the VMs than prototype should generate the automatic demonstration reports of success or failure of student efforts. Through this the teachers time will be saved and probability of error will be decreased because system will do it automatically. Network lab management tool will count the marks or grades as well.

6.1.6 Feedback(Teacher verifies results checking student work)

Teachers verifies automation results by spot-checking student work. They should double check the results because it decrease the students complain regarding to their results. After verifying the results teacher has to option, one he can repair all the virtual machines through network lab management tool, and second he can create new networks again. Creating new virtual network is best options because of new virtual machines. But it doesnt mean each time he create new networks. The tool should have functionality to repair all virtual machines. The teacher should check all virtual machines by manual and make sure all VM are working properly.

6.1.7 Updating(Teacher can update problem sets,if required)

Teacher reviews and update problem sets if required because with the passage of time new software and technologies will come into the market so teacher will be able to add new problems to the tool. When the teacher will be able to update the tool with latest problems then the effectiveness of tool will be increased over time.

6.2 Network Lab Management Tool VS Problem Statement

Creating the pedagogical framework is the first step of this thesis, and the second is to develop its prototype. Network Lab Management Tool is prototype of the pedagogical framework. Chapter 5 briefly describes how it works. Here we analyze the amplification of the tool in real examples.

6.2.1 Main Menu

Network Lab Management Tool is developed in Perl language. Figure 5.1 is the first look of Network Lab Management Tool. Users have 7 options for selection. This is first struggle to build like this; it needs more development to enable feedback automation, and should be useful for Windows platforms as well as Linux.

6.2.2 Install XEN or MLN automatically

From the Main Menu select the first option to install XEN or MLN automatically. Figure 5.2 shows the next page where users have three options 1) Install XEN 2) Install MLN or 3) Exit.

Now users have no need to bother how to install the XEN or MLN manually because this tool has ability to install automatically on the server. If one want to install another virtualization tool than this tool will not help him. As mentioned before that shortage of time this tool only consider few tools (e g. XEN, MLN, and Apache) but not all of them (KVM, Proxmox, and Open stack). It needs more development to add new tools and technologies in this tool to make it smart.

6.2.3 Create the Virtual Network

Network Lab Management Tool builds a virtual network with 4 virtual machines (one gateway and three VMs). Gateway has public IP to provide the Internet access to all VM in the network. Figure 5.3 shows the options where user can select one of them. If user select option 1 than it will build automatically virtual network with one gateway and three virtual machines. Only first option is active in this prototype now because we have not too much time to active other options as well. It should be smart to take the input from user in which he should give number of virtual machines to build. MLN script is uses in which all virtual machines have Linux as an operating system. Future development would include a MLN script in which virtual machines should have Windows as an operating system.

6.2.4 Convert the VM into Broken-State

In the Main Menu, option 4 uses to convert the VM into broke-state. Figure 5.4 shows to select any particular VM to assign problem sets. As we mentioned in 6.2.3 heading, only one virtual network can be create in this prototype. In option 4th user can create problems in three VMs because during 3rd option only one network created having three VMs.

6.2.5 Select Problem(s) For Virtual Machine

Figure 5.5 present the list of problems those can assign to the VMs. Only one problem can select in single time. When one problem is assigned than again user can able to selects another problem for that particular VM. In prototype only 5 problems are listed to assign the VMs. If you want to increase the number of problems than you can add in Perl script. But the good way to add new problems in the list is to create a plugins. Through this way user can easy add the problems otherwise he has to go in the script to explore the particular line to add new problem there.

6.2.6 Select The Broken VM To Reverse Into Functioning-State

According to the prototype teacher can create only one virtual network with three numbers of VMs. When all VMs or one of them get broken because of problems than teacher can repair one by one. Figure 5.6 shows to select the VM for repairing. Three VMs are listed there because there is one virtual network with three VMs. These VMs can be repair. This is limitation of prototype it should be dynamic according to the virtual networks.

Prototype will repair only those problems, which are selected during 3rd opting Select Problem(s) For Virtual Machine. If students create another problem into the machine than prototype will not repair that particular problem. This is limitation of the prototype needs more development in this area.

6.2.7 Install Packages on VMs (Apache web server at VM-1)

Prototype has functionality to install the Apache web server on VM-1. If teacher need to install apache web server than he should select this option and it will automatically install apache without any trouble on VM-1. This is one example to install any package.

Apache web server is fixed with VM-1 so if teacher want to install it at another VM than he has to make changes in the prototype script lines. It seems static behavior; it should be dynamic with other options.

6.2.8 Create Log File

To create a log file, which stores all the information regarding to VMs, mean which VM has assigned the problems at what date. In this way teacher can check the log file to know what kind of problems assigned to VM. In Main Menu 5th option designed for this purpose.

In only prints the VM name and it problems with date. You can find one more things in Log file than just the solution of particular problem with date. Prototype should be smart here as well, mean when problem is get solved than it should be canceled from log file as well. Through this way teacher will get to know how many problems are still in VM.

6.3 Future Work

The Pedagogical Framework is not specific to a technology or implementation. The proof of concept can be used for Linux and Windows platforms, but the prototype is applicable only for a Linux environment. It should be able to support Windows environment. There is limited time to develop a smart prototype so it needs more development in different points like as follows:

1. Regarding to feedback, Management tool should automatically demonstrate success or failure of student efforts. It should give full details with feedback of students reports. This will reduce the teacher processing time and it makes the whole process very smart.
2. Prototype should support both environments like Linux and Windows.
3. Now prototype only creates one virtual network, but it should depend on user input for the size and scope of the created network.
4. When user enters 1 or more than 1 VM to create than the list of all virtual machines comes in options 3 at Main Menu To convert the VMs into Broken-State.
5. When user selects 4th option from Main Menu than all VMs those are created in 3rd option should show in the list.
6. In the problem list only 5 numbers of problems are there for practice. It should be dynamic and smart, means user can able to add new problem sets in the future. Through plugins it can be happened, to develop a plugin that can add new problem sets. Otherwise you have to add the problems in prototypes script file.

7. MLN has ability to create many virtual networks in single MLN script. So develop a MLN script which can create many virtual networks and all of them should be accessible in Main Menu options.

Chapter 7

Conclusion

This thesis takes the approach from an overall idea of a framework based method to learn system administration. To be a system administrator takes time, both theoretical and practice work. Science shows that practice work is a more helpful way of learning. Based on this thesis framework I have created a tool to give better knowledge to student that will become experienced system administrators.

One of the particular problems that system administration students face is that they do not have realistic troubleshooting experience, leading to lot of difficulties in real world jobs. The true core of system administration is troubleshooting. Most of students studying networks and system administration have skills just to install and run different services, but they dont have any skill how to troubleshoot systems in different scenarios. In educational premises it is very hard to dedicate a single system to each student. So with the help of virtualization environment we can create VMs and practice on it to manage this issue as well. Teachers should provide an environment to build troubleshooting skills. In universities the teacher needs to build a virtual networks for students assignments regarding to diagnose the systems problems.

During the study period of network system administration there are many assignments in system administration course to build the knowledge. The purpose of network administration courses is to get handsome knowledge and practice.

The real center point of this project is to develop a pedagogical framework and network lab management tool that allows effective teaching (and learning) of system administration education.

This project may be helpful to the network teachers those have purpose to build a virtual networks for students. Use these virtual networks for students assignments to get troubleshooting skills. Creating the bulk of virtual networks through manual commands take too much time. Network lab management tool has the ability to create the virtual networks with in few minutes (5 to 10 minutes). It will save the teachers time and as well as it will do automatic demonstration success or failure of student efforts.

The Pedagogical Framework has seven stages and each of them has its own value to build teaching and learning environment. A prototype is the way to show either framework is applicable in real life or not. So after building a framework we develop a prototype called Network Lab Management Tool. The Network Lab Management Tool is the practical example of the Pedagogical Framework. According to the Pedagogical Framework processes prototype should do each processes automatically or with respect to user input. Our tool has ability to install the XEN, MLN, and Apache server as well. It can build virtual network with one gateway and three virtual machines. The teacher will use the tool to break the virtual machines and assign them to the students. If teachers feel to add some problem set into existing problem list than he can do it, to update the tools functionality.

The Pedagogical Framework is designed for both environments (Linux and Windows). Because of short time only one platform (Linux) used in practical work. The focus of this thesis is to develop a general purpose-troubleshooting framework and demonstrate a practical tool that implements the framework. My research ended up on a framework in which sets of virtual machines are there and those are in working positions.

The prototype will repair only those problems which are selected during 3rd option in Main Menu, Select Problem(s) For Virtual Machine. If students create other problems into the machine than prototype will not repair that particular problem. This is a limitation of the prototype it need more development in this area.

Using the management tool the teacher can fix all those problems that were created in the second step. There are two more things that remain to be done, the network lab management tool should capable to add new problem sets in current problem list and secondlyit should give automatically demonstrates success and failure of student efforts. To conduct this project we have limited time so there are few tasks left, you can find the list of tasks those should be completed in future work under heading 6.3.

Bibliography

[1] Clara Gerhardt Mary Sue Baldwin. Problem based learning. *The Journey. Paper presented at the International Problem Based Learning Symposium*, .:12–16, 2007.

[2] Kyrre Begnum. The mln project. Available at `http://mln.sourceforge.net/`.

[3] Kyrre Begnum and Karst Koymans Arjen Krap. Using virtual machines in system administration education. *Journal of Computing Sciences in Colleges*, 20 Issue 2:287–292, December 2004.

[4] Kyrre M Begnum. Managing large networks of virtual machines. *Proceedings of LISA '06: 20th Large Installation System Administration Conference. Washington, DC: USENIX Association*, .:205–214, December 3-8, 2006.

[5] X. Chang. Network simulations with opnet. *Proc. 1999 Winter Simulation Conf.: SimulationA Bridge to the Future*, .:307314, 1999.

[6] A. Chianese and M. De Santo. Methodology for lan design. *Comput. Commun*, vol. 9, no. 4:177185, 1986.

[7] J. Cigas. An introductory course in network administration. *Proc. Special Interest Group Computer Science Education (SIGCSE) , Reno, NV*, .:113116, Feb. 1923, 2003.

[8] D. E. Comer. Hands-on networking with internet technologies. englewood cliffs. *NJ: Prentice-Hall*, .:3, 2002.

[9] Christopher Curran. Introduction to troubleshooting and problem solving. Available at `http://docs.fedoraproject.org/en-US/Fedora/13/html/Virtualization_Guide/chap-Vi`

[10] D. Dobrilovic and Z. Stojanov. Using virtualization software in operating systems course. *Proc. Intl Conf. Information Technol- ogy: Research and Education*, ITRE 06:222–226, May 2006.

[11] Daniel Eason. Troubleshooting vm performance capacity management woes. Available at `http://www.computerweekly.com/tip/Troubleshooting-VM-performance-capacity-manag`

[12] A. Nagarajan F. Mueller C. Engelmann and S. Scott. Proactive fault tolerance for hpc with xen virtualization. *21st annual international conference on Supercomputing Seattl WA USA*, .:2332, June 1620 2007.

[13] Barham P. et al. Xen and the art of virtualization. *SOSP 03*, .:2, 2003.

[14] RackSpace's Fanatical. What is xen? Available at `http://wiki.xen.org/wiki/Xen_Overview`.

[15] Andreas Wundsam Amir Mehmood Anja Feldmann and Olaf Maennel. Improving network troubleshooting using virtualization. *.*, .:2, 2009.

[16] J. Fitzgerald and A. Dennis. Business data communications and networking. *7th ed. New York: Wiley*, .:5, 2002.

[17] B. A. Forouzan. Data communications and networking. *3rd ed. New York: McGraw-Hill*, .:5, 2004.

[18] Inigo Goiri Ferran Juli Jordi Guitart and Jordi Torres. Checkpoint-based fault-tolerant infrastructure for virtualized service providers. *.*, .:3–6, 2010.

[19] JeffDike. A user mode port of the linux kernel. *Proceedings of the 4th Annual Linux Showcase & Conference Atlanta*, .:2–6, 2000.

[20] J. Keengwe and L.O. Anyanwu. Computer technology-infused learning enhancement. *J. Science Education and Technology*, 16. no. 5:387–393, June 2007.

[21] K. Abe T. Tateoka M. Suzuki Y. Maeda K. Kono and T. Watanabe. An integrated laboratory for processor organization compiler design and computer networking. *IEEE Trans. Educ*, vol. 47. no. 2:311320, May 2004.

[22] Begnum K. K. Koymans A. Krap and J. Sechrest. Using virtual machines in system and network administration education. *Proceedings of the System Administration and Network Engineering Conference*, SANE:1–7, 2004.

[23] Begnum K. K. Koymans A. Krap and J. Sechrest. Using virtual machines in system and network administration education. *Proceedings of the System Administration and Network Engineering Conference*, .:3, 2004.

[24] Janus S. Liang. Education and training for troubleshooting in automotive chassis. *Yung-Ta Institute of Technology and Commerce*, .:1–3, 2007.

[25] M. Pollitt K. Nance B. Hay R.C. Dodge P. Craiger P. Burke C. Marberry and B. Brubaker. Virtualization and digital forensics: A research and education agenda. *J. Digital Forensic Practice*, 2:62–73, Apr. 2008.

[26] Michael Markl. Complete cost-effective datacenter and server virtualization. Available at `http://www.xensource.com`, Sept. 2006.

[27] Antonio RuizMartnez Fernando PerenguezGarca Rafael MarnLopez. Teaching advanced concepts in computer networks: Vnuml-um virtualization tool. *Learning Technologies IEEE Transactions on*, 6 . Issue: 1:85 96, 2013.

[28] J. Matthews. Computer networking: Internet protocols in action. *Wiley*, .:1–4, 2005.

[29] Tom McNamara. Computer troubleshooting skills. Available at `http://www.ehow.com/way_5449314_computer-troubleshooting-skills.html`.

[30] S. Nanda and T.C. Chiueh. A survey of virtualization technologies. *Technical Report TR-179 http://citeseerx.ist.psu.eduviewdoc summary-doi=10.1.1.74.371*, .:7–9, 2005.

[31] M. Palmer and R. B. Sinclair. Guide to designing and implementing local and wide area networks. *2nd ed. Canada: Course Technology*, .:5–9, 2003.

[32] Jianping Pan. Teaching computer networks in a real network. *University of Victoria Victoria BC Canada*, .:1, .

[33] Fedora Project. how to install and use the fedora operating system and the software packaged. Available at `http://docs.fedoraproject.org/enUS/Fedora/13/html/Virtualization_Guide/chap-Vi`

[34] J. R. Anderson L. M. Reder and H. A. Simon. Situated learning and education. *Educ. Res*, vol. 25. no. 4:511, 1996.

[35] Xin ZHAO Ya-Wei LI Yi-Yun ZHOU Liang REN and Gui-Zhang LU. Virtual process: Concept, problems and implementation framework. *Inst. of Robotics Information Automatic System Nankai University Tianjin China*, 300071:1–5, 2003. Jun.

[36] M. Rosenblum and T. Garfinkel. Virtual machine monitors current technology and future trends. *Computer*, 38. no. 5:2005, 3947.

[37] N. I. Sarkar and J. H. Lian. Lan-designer: a software tool for teaching and learning lan design. *Proc. 3rd IEEE Int. Conf. Advanced Learning Technologies, Athens, Greece*, .:260261, Jul. 911. 2003.

[38] Nurul I. Sarkar. Teaching computer networking fundamentals using practical laboratory exercises. *IEEE TRANSACTIONS ON EDUCATION*, 49. NO. 2:49–50, MAY 2006.

[39] Karst Koymans Arjen Krap John Sechrest and Kyrre Begnum. Using virtual machines in system administration education. ., .:6, 2005.

[40] Gert Jan Verhoog and Jeroen van der Ham. Virtual environments for networking experiments. ., .:6–8, 2004.

[41] J. White and A. Pilbeam. A survey of virtualization technologies with performance testing. *The Computing Research Repository http://arxiv.org/abs/1010.3233*, .:1–7, Oct. 2010.

[42] Xen XenSource. The xen virtual machine monitor. Available at `http://www.cl.cam.ac.uk/research/srg/netos/xen/`.

Appendix A

Perl Script for NLM Tool

```

%% Paste the script here


\#!/usr/bin/perl

\# our needed packages
use strict ”vars”;
use Getopt::Std;
use warnings;
use Switch;

\# Global Variables
my \$VERBOSE = 0;
my \$DEBUG = 0;
my(\$day, \$month, \$year)=(localtime)[3,4,5];
\# commandline options
my \$opt_string = ”vdh”;
getopts( ”\$opt_string”, \my %opt ) or usage() and exit(1);

\$VERBOSE = 1 if \$opt\lbrace 'v'\rbrace;
\$DEBUG = 1 if \$opt\lbrace 'd'\rbrace;

if ( \$opt\lbrace 'h'\rbrace ) \lbrace
 usage();
 exit 0;
\rbrace
\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#
my \$input = '';
while (\$input ne '7')
\lbrace
 clear_screen();
 print ”\n\n –\n”.
   ”  Main Menu\n”.
   ” –\n\n”.
   ”\t1. Install XEN or MLN\n”.
   ”\t2. Create the virtual network\n”.
   ”\t3. Convert the VMs into Broken−state\n”.
   ”\t4. Reverse the broken VMs into functioning−state\n”.
   ”\t5. Show broken VMs with their problem\n”.
   ”\t6. If you want to install APACHE2 ON VM−1 than PRESS 6\n\n”.

   ”\t7. Exit\n”;
 print ” –\n\n”;
 print ”Enter your choice: ”;
 \$input = <STDIN>;
 chomp(\$input);

 switch (\$input)
 \lbrace
        case '1'
        \lbrace
         \$input = '';

         while (\$input ne '3')
         \lbrace
                clear_screen();
        print ” — — — — — — — — — — — — ——\n”.
 ”  XEN − MLN INSTALLATION MENU\n”.
 ” — — — — — — — — — — — — ——\n”.
 ”\t1.If you want to install the XEN than PRESS 1\n”.
 "\t2.If you want to install the MLN than PRESS 2\n".
 "\t3.If you want to go BACK on MAIN Menu than PRESS 3\n";
                print " — — — — — — — — — — — — —\n";

                print "Enter your choice: ";
                \$input = <STDIN>;
                chomp(\$input);

                switch (\$input)
                \lbrace
                 case '1'
                 \lbrace

 print " — — — — — — — — — — — — — —\n".
   "  System Is Going To Inastall Xen\n".
   " — — — — — — — — — — — — — —\n";


 print "Installing Process ...........\n\n";
   system 5;
 system ("apt-get -y install openssh-server");
 sleep 5;
system ("aptitude -y install xen-hypervisor-4.0-amd64 xen-linux-system-2.6-xen-amd64 linux-image-xen-amd64
xen-qemu-dm-4.0 xen-tools xen-utils xenwatch xenstore-utils");
sleep 5;
system ("mv /etc/grub.d/10_linux /etc/grub.d/21_linux");
sleep 5;

   open (MYFILE, '>>/etc/default/grub');
   print MYFILE "GRUB_DISABLE_OS_PROBER=true\n";
   print MYFILE "GRUB_CMDLINE_XEN=\"dom0_mem=1024M\"";
   close (MYFILE);
   sleep 3;
   \# system ("sed -i bak -e s/Bob/Sameer/g /opt/mln/files/file ");

   system ("update-grub");
   sleep 5;

   system ("sed -i 's%XENDOMAINS_RESTORE=true%XENDOMAINS_RESTORE=false%g' /etc/default/xendomains");
 print "\n\n GIVEN FILE HAS BEEN MODIFIED\n\n\n";
 system("sed -i 's%XENDOMAINS_SAVE=/var/lib/xen/save%XENDOMAINS_SAVE=\"\"%g' /etc/default/xendomains");
\# system ("sed -i bak -e s/XENDOMAINS_SAVE=\/var\/lib\/xen\/save/XENDOMAINS_SAVE=""/g /etc/default/xendomains");
 print "\n\n GIVEN FILE HAS BEEN MODIFIED\n\n\n";
 sleep 3;

 system ("sed -i '174d' /etc/xen/xend-config.sxp");
   system ("sed -i '194d' /etc/xen/xend-config.sxp");
 print "\n\n This Files has been MODIFIED /etc/xen/xend-config.sxp\n\n";

   \# system("sed -i 's%XENDOMAINS_SAVE=/var/lib/xen/save%XENDOMAINS_SAVE=\"\"%g' /opt/mln/files/file");
   \#print "\n\n GIVEN FILE HAS BEEN MODIFIED\n";
   \#sleep 5;

   open (MYFILE, '>>/etc/xen/xend-config.sxp');
   print MYFILE "(enable-dom0-ballooning no)\n";
   print MYFILE "(network-script 'network-bridge antispoof=yes')\n\n";
   close (MYFILE);
     print "\n\n This Files has been MODIFIED /etc/xen/xend-config.sxp\n\n";

    print "   -\n".
   "  XEN CONFIGURATION HAS BEEN SUCCESSFULLY COMPLETED \n".
   "   -\n";

    print "\n\nNOW properly EXIT from the SCRIPT And REBOOT The System\n";
   print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\n\n";
    sleep 10;

                 \rbrace

                 case '2'
                 \lbrace

 print "   -\n".
   "  WELCOME TO YOU - MLN INSTALLATION \n".
   "   -\n";

 print "\n\n PROCEEDING ......\n\n";
 sleep 3;

 print "   -\n".
   "  SOME PACKAGES ARE GOING TO BE INSTALL \n".
   "   -\n";

 system ("apt-get install -y uml-utilities bridge-utils screen sudo");

 print "\n\n PACKAGES HAVE BEEN INSTALLED SUCCESSFULLY.\n\n";

 \# system ("wget http\:\/\/mln.sourceforge.net\/files\/mln-latest\.tar\.gz ");
 \# system ("tar xzfv mln-latest.tar.gz");

 \# system ("cp \/mln-lates\/mln \.");

 print "\\ STEP 1-5\n";
 sleep 3;
 system ("wget https\:\/\/www\.dropbox\.com\/s\/iudlu6ztmyo2qgx\/mln");
 system ("chmod \+x mln");
 system ("\.\/mln setup");
 system ("cp mln \/usr\/local\/bin\/");
 system ("chmod \+x \/usr\/local\/bin\/mln");
 print "\n\n FILE Download Complete \n\n";
 print "\n STEP 2-5\n";
 sleep 3;

 system ("sed -i 's%default_memory 64M%default_memory 512M%g' /etc/mln/mln.conf");
 system ("apt-get install -y user-mode-linux uml-utilities");

 system ("wget https\:\/\/www\.dropbox\.com\/s\/gzu0gwjrfj0nup9\/lenny_small_backports\.ext3 ");
 print"\n\n TEMPLATE DOWNLOAD COMPLETED\n\n";
 sleep 3;
 system ("cp lenny_small_backports\.ext3 \/opt\/mln\/templates\/");
 print "\n\n Template Copy into /etc/mln/templates/ FOLDER\n\n";
 print"\n\n STEP 3-5\n";
 sleep 3;
 system ("export PATH=\\\$PATH\:\/usr\/local\/bin");
 system ("mkdir \/opt\/mln\/projects");
 system ("echo \"projects \/opt\/mln\/projects\" \> \~\/\.mln");
 system ("echo \"templates \/opt\/mln\/templates\" \> \~\/\.mln");
 system ("echo \"files \/opt\/mln\/files\" \> \~\/\.mln");
                print "\n\n STEP 4-5\n";
 sleep 3;
 \#system ("export LANGUAGE=en_US\.UTF-8");
 \#system ("export LANG=en_US\.UTF-8");
 \#system ("export LC_ALL=en_US\.UTF-8");
 system ("locale-gen en_US\.UTF-8");
 print "\n\n STEP 5-5\n";
 sleep 3;

 system ("mln register_template -m \"The dummy template\" -t \/opt\/mln\/templates\/lenny_small_backports\.ext3");
 print "\n\n\n";
 system ("mln list_templates");


 print" — — — — — — — — — — — — —\n".
   "     CONGRATULATIONS - MLN INSTALLED SUCCESSFULLY\n".
   " — — — — — — — — — — — — —\n";
   print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\n\n";

 sleep 6;
 \rbrace
                \rbrace
         \rbrace

         \$input = '';
        \rbrace
        case '2'
        \lbrace
         \$input = '';

         while (\$input ne '4')
         \lbrace
                clear_screen();

                print" — — — — — — — — — — — — —\n".
 "     Creat the Virtual Network\n".
 " — — — — — — — — — — — — —\n".
 "\t1. Creat a virtual network with 3 VMs\n".
         "\t2. Creat a virtual network with 4 VMs\n".
 "\t3. Creat a virtual network with 5 VMs\n\n".
 "\t4. return to previous menu\n";
                print " — — — — — — — — — — — — —\n\n";
                print "Enter your choice: ";
                \$input = <STDIN>;
                chomp(\$input);

                switch (\$input)
                \lbrace
                 case '1'
                 \lbrace

   print"\n\n\n — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork3 IS GOING TO BUILD NOW\n".
   " — — — — — — — — — — — — —\n\n";

   print "\n\n PROCESSING .........\n\n";
   sleep 6;
 system ("wget https\:\/\/www\.dropbox\.com\/s\/6ffyknodlt1bsqz\/VNetwork3\.mln");
```

```
 system ("chmod \+x VNetwork3\.mln");
 system ("mln build -f VNetwork3.mln");
 sleep 6;

   print"\n\n — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork3 HAS COMPLETLY BUILD\n".
   " — — — — — — — — — — — — — —\n\n";

   sleep 6;
 system ("mln status -p VNetwork3");
   print "\n\n ALL VMs IN VNetwork3 ARE DOWN\n\n ";
 sleep 6;

   print"\n\n — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork3 IS GOING TO START OR UP NOW\n".
" — — — — — — — — — — — — — —\n\n";

 sleep 6;
 system ("mln start -p VNetwork3");
 sleep 6;

   print"\n\n — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork3 HAS STARTED OR UP NOW\n".
   " — — — — — — — — — — — — — —\n\n";

 sleep 6;
 system ("mln status -p VNetwork3");
 sleep 6;

                 print "\n\n NOW GENERATE THE SSH-KEYGEN FOR ALL VMs :\n\n";
                 system ("ssh-keygen");
                 print "\n\n Enter The VM's Root Password For 9 Times:\n\n";


   system ("ssh 128\.39\.73\.242 \-p 1111 \"mkdir \/root\/.ssh\/ \"");
   system ("scp \-P 1111 \/root\/.ssh\/id_rsa\.pub  root\@128\.39\.73\.242\:\/root\/.ssh\/");
   system ("ssh 128\.39\.73\.242 \-p 1111 \"mv \/root\/.ssh\/id_rsa\.pub \/root\/.ssh\/authorized_keys\"");
   print "\n\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n";

   system ("ssh 128\.39\.73\.242 \-p 2222 \"mkdir \/root\/.ssh\/ \"");
   system ("scp \-P 2222 \/root\/.ssh\/id_rsa\.pub  root\@128\.39\.73\.242\:\/root\/.ssh\/");
   system ("ssh 128\.39\.73\.242 \-p 2222 \"mv \/root\/.ssh\/id_rsa\.pub \/root\/.ssh\/authorized_keys\"");
   print "\n\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
     system ("ssh 128\.39\.73\.242 \-p 3333 \"mkdir \/root\/.ssh\/ \"");
   system ("scp \-P 3333 \/root\/.ssh\/id_rsa\.pub  root\@128\.39\.73\.242\:\/root\/.ssh\/");
   system ("ssh 128\.39\.73\.242 \-p 3333 \"mv \/root\/.ssh\/id_rsa\.pub \/root\/.ssh\/authorized_keys\"");

system ("ssh 128\.39\.73\.242 \-p 1111 \"sed \-i \'\\\\\$adeb\ http\:\/\/ftp\.no\.debian\.org\/debian\/\ squeeze\ main\ contrib\
non\-free\' \/etc\/apt\/sources\.list\"");
system ("ssh 128\.39\.73\.242 \-p 1111 \"sed \-i \'\\\\\$adeb\-src\ http\:\/\/ftp\.no\.debian\.org\/debian\/\ squeeze\ main\ contrib\
non\-free\' \/etc\/apt\/sources\.list\"");

system ("ssh 128\.39\.73\.242 \-p 2222 \"sed \-i \'\\\\\$adeb\ http\:\/\/ftp\.no\.debian\.org\/debian\/\ squeeze\ main\ contrib\
non\-free\' \/etc\/apt\/sources\.list\"");
system ("ssh 128\.39\.73\.242 \-p 2222 \"sed \-i \'\\\\\$adeb\-src\ http\:\/\/ftp\.no\.debian\.org\/debian\/\ squeeze\ main\ contrib\
non\-free\' \/etc\/apt\/sources\.list\"");

system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \'\\\\\$adeb\ http\:\/\/ftp\.no\.debian\.org\/debian\/\ squeeze\ main\ contrib\
non\-free\' \/etc\/apt\/sources\.list\"");
system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \'\\\\\$adeb\-src\ http\:\/\/ftp\.no\.debian\.org\/debian\/\ squeeze\ main\ contrib\
non\-free\' \/etc\/apt\/sources\.list\"");



   print"\n\n  — — -\n".
   "  CONGRATULATIONS TO YOU FOR COMPLETING NEW VIRTUAL NETWORK WITH 3VMs. \n".
   "  — — -\n\n";
  print"\n\n PLEASE WAIT TO FINISH THE PROCESS ........................................\n\n";

 sleep 10;

                 \rbrace
   case '2'
                 \lbrace
 print "\n You Pressed Sub-Option-2\n\n";
 system ("ls");
 sleep 7;

 print"\n\n\n — — — — — — — — — — — — — ——\n".
   "  PROJECT VNetwork4 IS GOING TO BUILD NOW\n".
   " — — — — — — — — — — — — — ——\n\n";

   print "\n\n PROCESSING .........\n\n";
   sleep 6;
   system ("wget ");
   system ("chmod \+x VNetwork4\.mln");
   system ("mln build -f VNetwork4\.mln");
   sleep 6;

   print"\n\n — — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork4 HAS COMPLETLY BUILD\n".
   " — — — — — — — — — — — — — —\n\n";

   sleep 6;
   system ("mln status −p VNetwork4");
   print "\n\n ALL VMs IN VNetwork4 ARE DOWN\n\n ";
   sleep 6;

   print"\n\n — — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork4 IS GOING TO START OR UP NOW\n".
   " — — — — — — — — — — — — — —\n\n";

   sleep 6;
   system ("mln start −p VNetwork4");
  sleep 6;

   print"\n\n — — — — — — — — — — — — — —\n".
   "  PROJECT VNetwork4 HAS STARTED OR UP NOW\n".
   " — — — — — — — — — — — — — —\n\n";

 sleep 6;
   system ("mln status −p VNetwork4");
   sleep 6;

   print"\n\n  — —  −\n".
   "  CONGRATULATIONS TO YOU FOR COMPLETING NEW VIRTUAL NETWORK WITH 4 VMS. \n".
   "  — —  −\n\n";
   print"\n\n PLEASE WAIT TO FINISH THE PROCESS ......................................\ n\n";
   sleep 10;
 \# system("sed −i 's%XENDOMAINS_SAVE=/var/lib/xen/save%XENDOMAINS_SAVE=\"\"%g' /opt/mln/files/file");

 \# system ("sed −i '16d' /opt/mln/files/file");
 \# system ("sed −i '18d' /opt/mln/files/file");
 \# system("sed −i 's%XENDOMAINS_SAVE=/var/lib/xen/save%XENDOMAINS_SAVE=\"\"%g' /opt/mln/files/file");
   \# print "\n\n GIVEN FILE HAS BEEN MODIFIED\n";
   \# sleep 5;

\# open (MYFILE, '>>/opt/mln/files/file');
 \# print MYFILE "(enable−dom0−ballooning no)\n";
 \#     print MYFILE "(network−script 'network−bridge antispoof=yes')\n\n";
 \# print MYFILE "GRUB_CMDLINE_XEN=\"dom0_mem=1024M\"";
 \# close (MYFILE);
 \# print "\n\n File Successfully Modified";
 \#     sleep 3;
 \# system ("sed −i bak −e s/Bob/Sameer/g /opt/mln/files/file ");
\# system ("sed −i 's%XENDOMAINS_RESTORE=true%XENDOMAINS_RESTORE=false%g' /opt/mln/files/file");
 \# print "\n\n GIVEN FILE HAS BEEN MODIFIED\n\n\n";
 \# system("sed −i 's%XENDOMAINS_SAVE=/var/lib/xen/save%XENDOMAINS_SAVE=\"\"%g' /opt/mln/files/file");
 \# system ("sed −i bak −e s/XENDOMAINS_SAVE=\/var\/lib\/xen\/save/XENDOMAINS_SAVE=""/g /etc/default/xendomains");
 \# print "\n\n GIVEN FILE HAS BEEN MODIFIED\n\n\n";
 \# sleep 3;
                  \rbrace
                  case '3'
   \lbrace
   print "\n You Pressed Sub−Option−3\n\n";
   system ("ls");
   sleep 7;
print"\n\n\n — — — — — — — — — — — — — ——\n".
   "  PROJECT VNetwork5 IS GOING TO BUILD NOW\n".
   " −\n\n";
   print "\n\n PROCESSING .........\ n\n";
   sleep 6;
   system ("wget ");
   system ("chmod \+x VNetwork5\.mln");
   system ("mln build −f VNetwork5\.mln");
   sleep 6;
   print"\n\n — — — — — — — — — — — — — ——\n".
   "  PROJECT VNetwork5 HAS COMPLETLY BUILD\n".
   " — — — — — — — — — — — — —  −\n\n";
   sleep 6;
   system ("mln status −p VNetwork5");
   print "\n\n ALL VMs IN VNetwork5 ARE DOWN\n\n ";
   sleep 6;
   print"\n\n — — — — — — — — — — — — — ——\n".
   "  PROJECT VNetwork5 IS GOING TO START OR UP NOW\n".
   " — — — — — — — — — — — — — ——\n\n";
   sleep 6;
   system ("mln start −p VNetwork5");
   sleep 6;
 print"\n\n — — — — — — — — — — — — — ——\n".
   "  PROJECT VNetwork5 HAS STARTED OR UP NOW\n".
   " — — — — — — — — — — — — — ——\n\n";
   sleep 6;
 system ("mln status −p VNetwork5");
 sleep 6;
 print"\n\n  — —  −\n".
```

```
      "   CONGRATULATIONS TO YOU FOR COMPLETING NEW VIRTUAL NETWORK WITH 5 VMS. \n".
      "   — —   -\n\n";
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS ....................................... \ n\n";
 sleep 10;
                    \rbrace
                   \rbrace
           \rbrace
           \$input = '';
          \rbrace
          case '3'
       \lbrace
 \$input = '';
 while (\$input ne '4')
 \lbrace
  clear_screen();
  print " — — — — — — — — — — — ——\n".
 "      Convert The VM Into Broken−State\n".
 " — — — — — — — — — — — — ——\n".
 "\t  VM    Name \t\t  IP Address\n".
 "\t1. VM−1 \t\t  10.0.0.21\n".
 "\t2. VM−2 \t\t  10.0.0.22\n".
"\t3. VM−3 \t\t  10.0.0.23\n\n".
 "\t4. Press 4 to go previous menus\n";
  print " — — — — — — — — — — — — ——\n";
  print "Enter your choice: ";
  \$input = <STDIN>;
  chomp(\$input);
  switch (\$input)
  \lbrace
                    case '1'
    \lbrace
\$input = '';
while (\$input ne '4')
 \lbrace
  clear_screen();
  print "   -\n".
     "  SELECT PROBLEMS FOR VM−1 \n".
     "   -\n".
     "\ta. To Change The DNS ID Than PRESS a Character\n".
     "\tb. To Change The GATEWAY IDs Than PRESS b Character\n".
     "\tc. To Change The Contants Of sources.list PRESS c Character\n".
     "\td. To Change The PRIVILLAGES OF /var/www/ From 755 To 700 On VM−1\n".
     "\te. To Change The IPTABLES Rule Than PRESS d Character\n\n".
     "\t4. If you want to back to main menu PRESS 4\n";
  print " -\n";
  print "Enter your choice: ";
  \$input = <STDIN>;
  chomp(\$input);
  switch (\$input)
  \lbrace
          case 'a'
          \lbrace

           print "\n\n — — — — — — — — — — — — ——\n".
             "      Change The DNS ID of VM−1 Into Broken−State\n".
             " — — — — — — — — — — — — ——\n";
             \#  system ("ssh 128\.39\.73\.242 \−p 1111 \"scp /etc/resolv\.conf \. \"");
           \# Important command  \#\#\#\#\#\#\#\#\#\#\# system ("scp  \−P 1111 root\@128\.39\.73\.242\:\/etc\/resolv\.conf \.");
           system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i\.save \'s\/128\.39\.89\.8\/128\.39\.99\.8\/\' \/etc\/resolv\.conf\"");
           print"\n\n VM−1 HAS BROKEN NOW\n\n";
                  open (MYFILE, '>>data.txt');
                  print MYFILE "\n======================================\n";
                  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
                  print MYFILE "VM−1 Has DNS Name Changed\n";
                  print MYFILE "======================================\n\n";
                  close (MYFILE);
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS ....................................... \ n\n";
sleep 3;
           \#system ("ssh user@10.0.0.11 \"cat \/etc\/resolv\.conf\"");
\rbrace
          case 'b'
          \lbrace
           print "\n\n   -\n".
             "      Change The GATEWAY IDs Of VM−1 Into Broken−State\n".
             "   -\n";
           system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'8d\' \/etc\/network\/interfaces\"");
            \# system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'\\\\\$anetmask 255\.255\.255\.255\' \/etc\/network\/interfaces\"");
           system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'\\\\\$agateway 10\.0\.3\.1\' \/etc\/network\/interfaces\"");
           \# system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'7inetmask 255\.255\.255\.0\' \/etc\/network\/interfaces\"");
           \#   system ("ssh 128\.39\.73\.242 \−p 1111 \"\/etc\/init\.d\/networking\ restart\"");
  print "\n\nProcess Completed Now . . . . . . . \n";
                  open (MYFILE, '>>data.txt');
 print MYFILE "\n======================================\n";
 print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
 print MYFILE "VM−1 Has Been Changed The GATEWAY ID\n";
 print MYFILE "======================================\n\n";
 close (MYFILE);
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS ....................................... \ n\n";
```

```
            sleep 5;

          \rbrace
          case 'c'
          \lbrace
            print "\n\n  — — — — –\n".
             "      To Change The Contants Of sources.list PRESS c Character VM–1 Into Broken–State\n".
             "  — — — — — — ——\n";
                  system ("ssh 128\.3\9.73\.242 \–p 1111 \"sed –i \'s\/\^\/\\#\=\/\' \/etc\/apt\/sources\.list\"");
                  print "\n\n Process Completed sources.list Modified\n";
 open (MYFILE, '>>data.txt');
 print MYFILE "\n=========================================================\n";
 print MYFILE "\$day–".(\$month+1)."–".(\$year+1900)."\n";
 print MYFILE "The Contants Of sources.list Modified Successfully On VM–1\n";
 print MYFILE "=========================================================\n\n";
 close (MYFILE);
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\ n\n";
      sleep 5;
          \rbrace
case 'd'
   \lbrace
      print "\n\n  — — — — — —\n".
 "      To Change The PRIVILLAGES OF /var/www/ From 755 To 700 On VM–1\n".
 "  — — — — — — —  –\n";
 system ("ssh 128\.39\.73\.242 \–p 1111 \"chmod 700 \/var\/www\/\"");
 print "\n\n Process Completed: /var/www/ Privillages Have Changed\n";
 open (MYFILE, '>>data.txt');
 print MYFILE "\n=========================================================\n";
 print MYFILE "\$day–".(\$month+1)."–".(\$year+1900)."\n";
 print MYFILE "PRIVILLAGES HAVE CHANGED From 755 To 700 At VM–1\n";
 print MYFILE "=========================================================\n\n";
 close (MYFILE);
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\ n\n";
      sleep 5;
   \rbrace
 \rbrace
                  \rbrace
 \$input = '';
          \rbrace
          case '2'
          \lbrace
          \$input = '';
          while (\$input ne '4')
          \lbrace
                  clear_screen();

                  print " — — — — — — — — — — — — ——\n".
                  "  SELECT PROBLEMS FOR VM–2 \n".
                  " — — — — — — — — — — — — ——\n".
     "\ta. To Change The DNS ID Than PRESS a Character\n".
     "\tb. To Change The GATEWAY IDs Than PRESS b Character\n".
     "\tc. To Change The Contants Of sources.list PRESS c Character\n".
     "\td. To Change The IPTABLES Rule Than PRESS d Character\n".
     "\te. To Shutdown The Given VMachine Than PRESS e Character\n\n".
     "\t4. If you want to back to main menu PRESS 4\n";
                  print " — — — — — — — — — — — — —\n";

                  print "Enter your choice: ";
                  \$input = <STDIN>;
                  chomp(\$input);

                  switch (\$input)
\lbrace

                  case 'a'
                  \lbrace

                  print "\n\n — — — — — — — — — — — — ——\n".
 "      Change The DNS ID of VM–2 Into Broken–State\n".
 " — — — — — — — — — — — — ——\n";

 \# system ("scp  \–P 1111 root\@128\.39\.73\.242\:\/etc\/resolv\.conf \.");
  system ("ssh 128\.39\.73\.242 \–p 2222 \"sed \–i\.save \'s\/128\.39\.89\.8\/128\.39\.99\.8\/\' \/etc\/resolv\.conf\"");
 print"\n\n VM–1 HAS BROKEN NOW\n\n";

  open (MYFILE, '>>data.txt');
  print MYFILE "\n=====================================\n";
  print MYFILE "\$day–".(\$month+1)."–".(\$year+1900)."\n";
  print MYFILE "VM–2 Has DNS Name Changed\n";
  print MYFILE "=====================================\n\n";
  close (MYFILE);
print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\ n\n";

  sleep 3;



                  \rbrace

            case 'b'

                    \lbrace

                    print "\n\n    -\n".
 "      Change The GATEWAYIDs Of VM-2 Into Broken-State\n".
 "    -\n";


              system ("ssh 128\.39\.73\.242 \-p 2222 \"sed \-i \'8d\' \/etc\/network\/interfaces\"");
 \# system ("ssh 128\.39\.73\.242 \-p 2222 \"sed \-i \'\\\\\$anetmask 255\.255\.255\.255\' \/etc\/network\/interfaces\"");
  system ("ssh 128\.39\.73\.242 \-p 2222 \"sed \-i \'\\\\\$agateway 10\.0\.3\.1\' \/etc\/network\/interfaces\"");

              \# system ("ssh 128\.39\.73\.242 \-p 2222 \"\/etc\/init\.d\/networking\ restart\"");
  print "\n\nProcess Completed Now . . . . . . \n";
 open (MYFILE, '>>data.txt');
 print MYFILE "\n======================================\n";
 print MYFILE "\$day-".(\$month+1)."-".(\$year+1900)."\n";
 print MYFILE "VM-2 Has Been Changed The GATEWAY ID\n";
 print MYFILE "======================================\n\n";
 close (MYFILE);
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\n\n";

             sleep 3;

           \rbrace

 case 'c'

           \lbrace

       print "\n\n  — — — — -\n".
 "      To Change The Contants Of sources.list PRESS c Character VM-2 Into Broken-State\n".
 "  — — — — — — ——\n";

 system ("ssh 128\.3\9.73\.242 \-p 2222 \"sed -i \'s\/\^\/\\#\=\/\' \/etc\/apt\/sources\.list\"");
 print "\n\n Process Completed sources.list Modified\n";

 open (MYFILE, '>>data.txt');
 print MYFILE "\n=============================================================\n";
 print MYFILE "\$day-".(\$month+1)."-".(\$year+1900)."\n";
 print MYFILE "The Contants Of sources.list Modified Successfully On VM-2\n";
 print MYFILE "=============================================================\n\n";
 close (MYFILE);

print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\n\n";
      sleep 5;
           \rbrace
       \rbrace
 \rbrace

           \$input = '';
\rbrace
          case '3'
          \lbrace

                    \$input = '';

                    while (\$input ne '4')
                    \lbrace
                    clear_screen();

                    print " — — — — — — — — — — — — ——\n".
                    "  SELECT PROBLEMS FOR VM-3 \n".
                    " — — — — — — — — — — — — ——\n".
     "\ta. To Change The DNS ID Than PRESS a Character\n".
     "\tb. To Change The GATEWAY IDs Than PRESS b Character\n".
     "\tc. To Change The Contants Of sources.list PRESS c Character\n".
     "\td. To Change The IPTABLES Rule Than PRESS d Character\n".
     "\te. To Shutdown The Given VMachine Than PRESS e Character\n\n".
     "\t4. If you want to back to main menu PRESS 4\n";
                    print " — — — — — — — — — — — — —\n";

                    print "Enter your choice: ";
                    \$input = <STDIN>;
                    chomp(\$input);

                    switch (\$input)
                    \lbrace

            case 'a'
            \lbrace

                    print "\n\n — — — — — — — — — — — — ——\n".
 "      Change The DNS ID of VM-3 Into Broken-State\n".
 " — — — — — — — — — — — — ——\n";

```

```

         \# system ("scp  \-P 3333 root\@128\.39\.73\.242\:\/etc\/resolv\.conf \.");
                 system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i\.save \'s\/128\.39\.89\.8\/128\.39\.99\.8\/\' \/etc\/resolv\.conf\"");
   print"\n\n VM-1 HAS BROKEN NOW\n\n";

  open (MYFILE, '>>data.txt');
 print MYFILE "\n======================================\n";
                 print MYFILE "\$day-".(\$month+1)."-".(\$year+1900)."\n";
  print MYFILE "VM-3 Has DNS Name Changed\n";
  print MYFILE "======================================\n\n";
  close (MYFILE);
  print"\n\n PLEASE WAIT TO FINISH THE PROCESS ....................................... \ n\n";

   sleep 3;




         \rbrace

         case 'b'

         \lbrace

                 print "\n\n   -\n".
  "      Change The GATEWAYIDs Of VM-3 Into Broken-State\n".
  "   -\n";



         system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \8d\' \/etc\/network\/interfaces\"");
   \# system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \'\\\\\$anetmask 255\.255\.255\.255\' \/etc\/network\/interfaces\"");
   system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \'\\\\\$agateway 10\.0\.3\.1\' \/etc\/network\/interfaces\"");

     \#  system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \'7igateway 10\.0\.3\.1\' \/etc\/network\/interfaces\"");
     \#  system ("ssh 128\.39\.73\.242 \-p 3333 \"sed \-i \'\\\\\$agateway 10\.0\.3\.1\' \/etc\/network\/interfaces\"");

  \# system ("ssh 128\.39\.73\.242 \-p 3333 \"\/etc\/init\.d\/networking\ restart\"");
   print "\n\nProcess Completed Now . . . . . . \n";

  open (MYFILE, '>>data.txt');
  print MYFILE "\n======================================\n";
  print MYFILE "\$day-".(\$month+1)."-".(\$year+1900)."\n";
  print MYFILE "VM-3 Has Been Changed The GATEWAY ID\n";
  print MYFILE "======================================\n\n";
  close (MYFILE);
  print"\n\n PLEASE WAIT TO FINISH THE PROCESS ....................................... \ n\n";

         sleep 3;
\rbrace

                         case 'c'

          \lbrace
 print "\n\n  — — — — -\n".
  "      To Change The Contants Of sources.list PRESS c Character VM-3 Into Broken-State\n".
  "   — — — — — — ——\n";

  system ("ssh 128\.3\9.73\.242 \-p 3333 \"sed -i \'s\/\^\/\\#\=\/\' \/etc\/apt\/sources\.list\"");
  print "\n\n Process Completed sources.list Modified\n";

  open (MYFILE, '>>data.txt');
  print MYFILE "\n============================================================\n";
  print MYFILE "\$day-".(\$month+1)."-".(\$year+1900)."\n";
  print MYFILE "The Contants Of sources.list Modified Successfully On VM-3\n";
  print MYFILE "============================================================\n\n";
  close (MYFILE);

  print"\n\n PLEASE WAIT TO FINISH THE PROCESS ....................................... \ n\n";

      sleep 5;


          \rbrace

        \rbrace

   \rbrace

\$input = '';
 \rbrace

    \rbrace
 \rbrace
         \$input = '';
\rbrace

        case '4'
```

```
          \lbrace
          \$input = '';

          while (\$input ne '4')
          \lbrace
                  clear_screen();

                  print "  — — — — —\n".
                  "      Select The Broken−State VM To Reverse Into Functioning−State \n".
                  "  — — —  −\n".
                  "\t VM?? Name \t\t IP Address\n".
                  "\t1. VM−1 \t\t 10.0.0.21\n".
                  "\t2. VM−2 \t\t 10.0.0.22\n".
                  "\t3. VM−3 \t\t 10.0.0.23\n\n".
                  "\t4. Press 4 to go previous menus\n";


                  print "  — — ——\n";

                  print "Enter your choice: ";
                  \$input = <STDIN>;
                  chomp(\$input);

          switch (\$input)
          \lbrace

           case '1'
           \lbrace
                  print " — — — — — — — — — — — — — —\n".
  "      Reverse the VM−1 Into Functioning−State\n".
  " — — — — — — — — — — — — — —\n";

                  \#  Important Command \#\#\#\#\#\#\#\#\#\# system ("scp  \−P 1111 resolv\.conf root\@128\.39\.73\.242\:\/etc\/");


                  system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'1inameserver 128\.39\.89\.8\' \/etc\/resolv\.conf\"");

                  system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'2d\' \/etc\/resolv\.conf\"");

                  \# system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'\\\\\$anameserver 128\.39\.89\.8\' \/etc\/resolv\.conf\"");

                  print "\n\nOn VM−1 The DNS Problem Has SOLVED Now :\n\n";


                  open (MYFILE, '>>data.txt');
  print MYFILE "\n======================================\n";
                  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";

  print MYFILE "On VM−1 The DNS Problem Has SOLVED\n";
                  print MYFILE "======================================\n\n";
  close (MYFILE);

                  system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'8d\' \/etc\/network\/interfaces\"");
  \# system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'\\\\\$anetmask 255\.255\.255\.0\' \/etc\/network\/interfaces\"");
   system ("ssh 128\.39\.73\.242 \−p 1111 \"sed \−i \'\\\\\$agateway 10\.0\.0\.1\' \/etc\/network\/interfaces\"");


 open (MYFILE, '>>data.txt');
  print MYFILE "\n==============================================\n";
  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "GATEWAY Problem Has SOLVED On VM−1\n";
  print MYFILE "==============================================\n\n";
  close (MYFILE);
                  print"\nGATEWAY Problem Has SOLVED On VM−1\n";
   system ("ssh 128\.3\9.73\.242 \−p 1111 \"sed −i \'s\/\\#\=\/\/\' \/etc\/apt\/sources\.list\"");
  open (MYFILE, '>>data.txt');
  print MYFILE "\n=========================================================\n";
  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "The Contants Of sources.list Revised Successfully On VM−1\n";
  print MYFILE "=========================================================\n\n";
  close (MYFILE);
 print "The Contants Of sources.list Revised Successfully On VM−1";
      print "\n\n  — — — — — —\n".
  "      To Change The PRIVILLAGES OF /var/www/ From 755 To 700 On VM−1\n".
  "  — — — — — — —  −\n";
   system ("ssh 128\.39\.73\.242 \−p 1111 \"chmod 755 \/var\/www\/\"");
   print "\n\n Process Completed: /var/www/ Privillages Have Changed\n";
  open (MYFILE, '>>data.txt');
  print MYFILE "\n=====================================================\n";
  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "PRIVILLAGES HAVE CHANGED From 700 To 755 At VM−1\n";
  print MYFILE "=====================================================\n\n";
  close (MYFILE);
 print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\n\n";
      sleep 5;

                  \rbrace

```

```
            case '2'
 \lbrace



                  \# system ("scp  \−P 2222 resolv\.conf root\@128\.39\.73\.242\:\/etc\/");
   print " — — — — — — — — — — — — — —\n".
 "      Reverse the VM−2 Into Functioning−State\n".
 " — — — — — — — — — — — — — — —\n";


   system ("ssh 128\.39\.73\.242 \−p 2222 \"sed \−i \'1inameserver 128\.39\.89\.8\' \/etc\/resolv\.conf\"");

   system ("ssh 128\.39\.73\.242 \−p 2222 \"sed \−i \'2d\' \/etc\/resolv\.conf\"");

                  print "\n\nOn VM−2 The DNS Problem Has SOLVED Now :\n\n";

                  open (MYFILE, '>>data.txt');
  print MYFILE "\n=====================================\n";
                  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "On VM−2 The DNS Problem Has SOLVED\n";
  print MYFILE "=====================================\n\n";
  close (MYFILE);

                  system ("ssh 128\.39\.73\.242 \−p 2222 \"sed \−i \'8d\' \/etc\/network\/interfaces\"");
   system ("ssh 128\.39\.73\.242 \−p 2222 \"sed \−i \'\\\\\$agateway 10\.0\.0\.1\' \/etc\/network\/interfaces\"");

  open (MYFILE, '>>data.txt');
  print MYFILE "\n===============================================\n";
  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "GATEWAY Problem Has SOLVED On VM−2\n";
  print MYFILE "===============================================\n\n";
  close (MYFILE);

                  print"\nGATEWAY Problem Has SOLVED On VM−2\n";
   system ("ssh 128\.3\9.73\.242 \−p 2222 \"sed −i \'s\/\\#\=\/\/\' \/etc\/apt\/sources\.list\"");

  open (MYFILE, '>>data.txt');
  print MYFILE "\n============================================================\n";
  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "The Contants Of sources.list Revised Successfully On VM−2\n";
  print MYFILE "============================================================\n\n";
  close (MYFILE);

                  print "\n\nThe Contants Of sources.list Revised Successfully On VM−2";

   print"\n\n PLEASE WAIT TO FINISH THE PROCESS .......................................\n\n";
                  sleep 5;


 \rbrace

case '3'
 \lbrace



                  \# system ("scp  \−P 3333 resolv\.conf root\@128\.39\.73\.242\:\/etc\/");
   print " — — — — — — — — — — — — — —\n".
 "      Reverse the VM−3 Into Functioning−State\n".
 " — — — — — — — — — — — — — — —\n";

   system ("ssh 128\.39\.73\.242 \−p 3333 \"sed \−i \'1inameserver 128\.39\.89\.8\' \/etc\/resolv\.conf\"");

   system ("ssh 128\.39\.73\.242 \−p 3333 \"sed \−i \'2d\' \/etc\/resolv\.conf\"");
                  print "\n\nOn VM−3 The DNS Problem Has SOLVED Now :\n\n";

    open (MYFILE, '>>data.txt');
  print MYFILE "\n=====================================\n";
                  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "On VM−3 The DNS Problem Has SOLVED\n";
  print MYFILE "=====================================\n\n";
  close (MYFILE);

          system ("ssh 128\.39\.73\.242 \−p 3333 \"sed \−i \'8d\' \/etc\/network\/interfaces\"");
   system ("ssh 128\.39\.73\.242 \−p 3333 \"sed \−i \'\\\\\$agateway 10\.0\.0\.1\' \/etc\/network\/interfaces\"");

  open (MYFILE, '>>data.txt');
  print MYFILE "\n===============================================\n";
  print MYFILE "\$day−".(\$month+1)."−".(\$year+1900)."\n";
  print MYFILE "GATEWAY Problem Has SOLVED On VM−3\n";
  print MYFILE "===============================================\n\n";
  close (MYFILE);

                  print"\nGATEWAY AND NETMASK Problems Has SOLVED On VM−3\n\n";
   system ("ssh 128\.3\9.73\.242 \−p 3333 \"sed −i \'s\/\\#\=\/\/\' \/etc\/apt\/sources\.list\"");
   print "\n\n Process Completed sources.list Modified\n";

  open (MYFILE, '>>data.txt');
  print MYFILE "\n============================================================\n";
  print MYFILE "\$day-".(\$month+1)."-".(\$year+1900)."\n";
  print MYFILE "The Contants Of sources.list Revised Successfully On VM-3\n";
  print MYFILE "==========================================================\n\n";
  close (MYFILE);
print "\n\n The Contants Of suorces.list Revised Successfuly On VM-3";

   print"\n\n PLEASE WAIT TO FINISH THE PROCESS ......................................\ n\n";
                  sleep 5;


  \rbrace



         \rbrace

  \rbrace

  \$input = '';

\rbrace



case '5'
\lbrace

                  print "\n\n ---------------------------------------------\n".
  "     You Have Selected Optioin 5 From Main Menu\n".
  " ---------------------------------------------\n";



                  print "\n\n===========> INFORMATION ABOUT VIRTUAL MAHCHINES  <=================\n\n\n";
   system ("cat data.txt |more");

                  \# system ("ssh 128\.39\.73\.242 \-p 3333 \"mkdir \/root\/.ssh\/ \"");
                  \# system ("scp \-P 3333 \/root\/.ssh\/id_rsa\.pub  root\@128\.39\.73\.242\:\/root\/.ssh\/");
                  \# system ("ssh 128\.39\.73\.242 \-p 3333 \"mv \/root\/.ssh\/id_rsa\.pub \/root\/.ssh\/authorized_keys\"");
                  print "\n\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n";
                  sleep 10;


\rbrace
case '6'

                  \lbrace


   print "\n\n  ---------------\n".
  "     You Have Selected Optioin 6 From Main Menu To Install APACHE2 On VM-1\n".
    "  ---------------\n";
print "\n\n Virtual Machine Is Going To Upgrade The apt-get Package\n\n";

                  system ("ssh 128\.39\.73\.242 \-p 1111 \"apt\-get\ upgrade \"");
                  system ("ssh 128\.39\.73\.242 \-p 1111 \"apt\-get\ update  \"");
                  system ("ssh 128\.39\.73\.242 \-p 1111 \"apt\-get\ \-y\ install\ apache2 \"");

                  print "\n\n APACHE2 SERVER INSTALLED SUCCESSFULLY\n";


                  sleep 5;

                  \rbrace



exit(0);


sub clear_screen
\lbrace
  system("clear");
\rbrace



exit 0;


\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#
\# subroutines


sub usage \lbrace

  print "Usage:\n";
  print "-h for help\n";
  print "-v for verbose (more output ) \n";
  print "-d for debug (even more output)\n";
\rbrace


sub verbose \lbrace
  print "VERBOSE: " . \$_[0] if ( \$VERBOSE);
\rbrace


sub debug \lbrace
  print "DEBUG: " . \$_[0] if ( \$DEBUG );
\rbrace
sub menue \lbrace
  system ("clear");
  if (\$_[0] == 1)\lbrace
         print "Submenu 1\n";
         print "1- select1\n";
         print "2- select1\n";
         print "3- select1\n";
\rbrace
  if (\$_[0] == 2)\lbrace
         print "Submenu 2\n";
         print "1- select21\n";
         print "2- select2\n";
         print "3- select2\n";
  \rbrace
  if (\$_[0] == 3)\lbrace
         goto main;
  \rbrace
\rbrace

\rbrace


\rbrace


}



%%%% Script end here
```