

Jaakko Nurmi

Implementation of Nested Virtual Laboratory System

Bachelor thesis
Information Technology

March 2016



KYAMK
University of Applied Sciences

Tekijä/Tekijät	Tutkinto	Aika
Jaakko Nurmi	Insinööri (AMK)	maaliskuu 2016
Opinnäytetyön nimi		
Implementation of nested Virtual Laboratory System		39 sivua 3 liitesivua
Toimeksiantaja		
Kymenlaakson ammattikorkeakoulu		
Ohjaaja		
Lehtori Vesa Kankare		
Tiivistelmä		
<p>Tämän opinnäytetyön tarkoituksena oli luoda helppokäyttöinen virtualisoitu laboratorioalusta tietoverkkotekniikan ja kyberturvallisuuden käytännön harjoituksiin. Työssä hyödynnettiin paljon jo olemassa olevaa teknologiaa ja lisäksi luotiin muutamia omia teknologioita. Yksi niistä on virtuaalinen kytkentäkaapeli, joka kykenee simuloimaan kaapelissa tapahtuvia häiriöitä.</p> <p>Työssä käsitellään sisäkkäistä virtualisointia (Nested virtualization) ja muita käytettyjä teknologioita sekä niiden ominaisuuksia. Käytännön osuudessa käsitellään virtuaalisen laboratorioympäristön toimintaa, ominaisuuksia ja etuja verrattuna fyysisiin laitteisiin sekä muutamia kehitysideoita tulevaisuuden varalle.</p> <p>Lopputuloksena syntyi helppokäyttöinen virtualisoitu tietoverkkolaboratorioympäristö, joita käytettiin ja kokeiltiin muun muassa Advanced routing-, Network security equipment ja Service provider networks -kursseilla pääsääntöisesti hyvin lopputuloksin. Sivutuotteena syntyi kaksi virtuaalista demoskenaariota kyberturvallisuuden opettamiseen. Näitä skenaarioita kokeiltiin penetraatiotestaus-kurssilla.</p> <p>Työn loppupuolella toteutettiin pieni käyttäjäkysely, jolla mitattiin loppukäyttäjien kokemuksia virtuaalisen laboratorioympäristön toiminnasta verrattuna oikeaan laboratorioympäristöön. Lisäksi muutama suorituskykytesti tehtiin työn aikana syntyneen virtuaalisen kytkentäkaapelin ja vastaavan tekniikan välillä.</p> <p>Työ onnistui hyvin, vaikka paljon kehitettävää ja parannettavaa vielä jäikin. Työn tulokset ja kokemukset antavat myös hyvän lähtökohdan laajentaa virtuaalista laboratorioympäristöä muille kursseille.</p>		
Asiasanat		
virtualisointi, suunnittelu, linux, kvm		

Author (authors)	Degree	Time
Jaakko Nurmi	Bachelor of Information Technology	March 2016
Thesis Title		
Implementation of Nested Virtual Laboratory System		39 pages 3 pages of appendices
Commissioned by		
Kymenlaakson Ammattikorkeakoulu Oy		
Supervisor		
Vesa Kankare, Senior Lecturer		
Abstract		
<p>The objective of this bachelor's thesis was to implement a virtual laboratory environment for needs of networking related study units and cyber security. A lot of existing technologies were utilized in the project and, in addition, a few applications were created during the project. One of them is a virtual cable, which has capability to do link interference simulation.</p> <p>Nested virtualization and other used technologies are discussed in theory part of this bachelor thesis. The practical part discusses the operation of the system and project result. Also, some comparison between the virtual laboratory environment and the hardware laboratories were made.</p> <p>As a result of the project, a virtual networking laboratory environment was created. It was compressively used in advanced routing, network security equipment, and service provider network with good results. As a side product, two demo cyber security scenarios were emerged. These cyber security scenarios were used and tested in the penetration testing study unit.</p> <p>In the late part of the project, a small user survey was implemented for the end users of the Virtual Laboratory System. The survey measured experiences and usefulness of the system in view of end user. In addition, a few performance benchmark tests were performed between the virtual cable and similar technology.</p> <p>The project succeeded well, even though a lot of developing and improvements are still required. The results of the project give good basis to expand virtualized laboratory environments also for other study units.</p>		
Keywords		
virtualization, implementation, linux, kvm		

SISÄLLYS

1	INTRODUCTION	6
1.1	Background and motivation	6
1.2	Related products.....	7
2	WHAT IS VIRTUALIZATION	7
2.1	Nested virtualization	8
2.2	Nested virtualization use cases	9
3	USED SOFTWARE IN THE PROJECT	9
3.1	Qemu/KVM hypervisor.....	10
3.1.1	Networking in Qemu.....	10
3.1.2	Qemu display devices and protocols.....	11
3.2	Guacamole remote desktop gateway	13
4	CREATED SOFTWARE FOR THE PROJECT	14
4.1	Ebrd, Ethernet Bridge Daemon.....	14
4.1.1	Ebrd technical desing.....	14
4.1.2	Controlling ebrd process	17
4.2	TCP-Console Multiplexer.....	18
5	TECHNICAL DESIGN OF VIRTUAL LABORATORY	19
5.1	NestCore	19
5.2	Scenario Disk.....	20
5.3	Operation of Virtual Laboratory.....	20
5.4	Virtual device installation and configuration	22
5.5	Virtual laboratory networking	23
5.6	User interface	25
6	USE CASES OF VIRTUAL LABORATORY.....	27
6.1	Virtual Networking Laboratories.....	27
6.2	Cyber security training scenarios.....	27
6.3	Future possible use cases	28
7	PROJECT RESULTS AND BENCHMARK TESTS.....	29
7.1	Benefits of Virtual Laboratories.....	29

7.2	<i>Ebrd performance test against Linux bridges</i>	29
8	<i>FUTURE POSSIBLE DEVELOPMENT IDEAS</i>	35
8.1	<i>Virtual Laboratory control protocol</i>	35
8.2	<i>Implementation of laboratory reservation system</i>	36
8.3	<i>Multi user support</i>	37
8.4	<i>Implementing solution to use SPICE-protocol through web browser.</i>	38
9	<i>CONCLUSIONS</i>	39
10	<i>REFERENCES</i>	40
<i>APPENDICES</i>		
	<i>Appendix 1. Result of user survey and open feedback and development ideas from end users</i>	

1 INTRODUCTION

Virtualization has raised significance a lot today, because computer and network hardware has improved fast in last ten years. Virtualizing devices such as servers or network devices give many benefits such as flexibility, redundancy and cost effectiveness comparing to have them as physical hardware.

A growing number of students and lack of devices in a physical networking laboratory in KyUAS, Kymenlaakso University of applied sciences is an upcoming problem, where virtualized laboratory exercises would give a solution. Also hardware network devices in networking laboratory in KyUAS are aging fast and oldest of them are over 5 years old.

1.1 Background and motivation

A project originally started in November 2015 in a work practicing period in KyUAS. The project main goal was to implement a Qemu-KVM hypervisor platform for needs of Cyber Game project. During the project, multiple problems occurred and solving those required special programming. Also many new ideas were found during developing this platform. One idea was to implement an interactive view of network topology and implementation of a configurable virtual cable. In January of the 2016, it was expanded as a bachelor thesis.

For while, a virtualized network laboratory platform, SPNET has been available for a service provider networks study unit, which was done by Vesa Kankare as his master thesis in 2015. This platform has given many benefits over doing the same laboratory exercises with physical devices. One of them is that every student has their own set of virtualized equipment. By virtualizing laboratory exercises, larger and realistic exercises can be implemented without large investments for expensive devices. (Kankare 2016.)

Feedback of Kankare's SPNET was quite positive. Most of the students in the Service Provider Networks course in spring of 2015 liked the idea to have

each device themselves without sharing them with co students. (Kankare 2015.)

This project is somewhat based to original SPNET version. It has some similarities with it, but most part of the whole system has been redesigned to be more expandable and able to run a different type of scenarios. Also gamification of network study units has been taking into account.

1.2 Related products

GNS3 is an open source real time network simulator. It is very versatile and can run a wide range of different network device. It has graphical user interface for easy designing of network topologies. GNS3 is powered with Qemu hypervisor, which provides much possibilities for virtualizing different types of network devices. (GNS3 website 2016.)

Cisco VIRL is a commercial product of Cisco Systems. VIRL is Virtual Internet Routing Lab. VIRL allows user to design and simulate network in real time. VirI has a wide range of features and it is meant for network professionals. VirI utilizes Qemu/KVM hypervisor with OpenStack for controlling virtual devices. (Cisco 2016.)

GNS3 and Cisco VIRL are both good products for network learning purposes, but both of them are somewhat difficult to use, especially for non-advanced users. VIRL is not ideally suitable for gamification purposes because of the licenses needed to use, and it has complex backed system for controlling and accessing to devices. Both of them are not well suitable for needs of cyber security training scenarios.

2 WHAT IS VIRTUALIZATION

Virtualization is technology which allows running multiple virtual machines in one computer. Virtualization gives many benefits over hardware solutions, and it is a very cost-effective solution. It allows to have multiple virtual devices running on same hardware which reduces need of hardware.

Virtual machine host is called a hypervisor. Hypervisor is a software which manages host system resources such as CPU and RAM and allocates them to virtual machines. Hypervisors are defined in two types.

Type-1 Hypervisors means a bare metal hypervisors, which run on top of hardware. Type-2 Hypervisors are hosted hypervisors which operate as application top of an operating system. Type-1 hypervisor provides better flexibility and performance for the virtual machines, because it is designed to expose resources to the virtual machine. (Techtarget. 2010.)

2.1 Nested virtualization

Nested virtualization refers to running hypervisor inside another hypervisor. This allows to have more depth in virtualization systems.

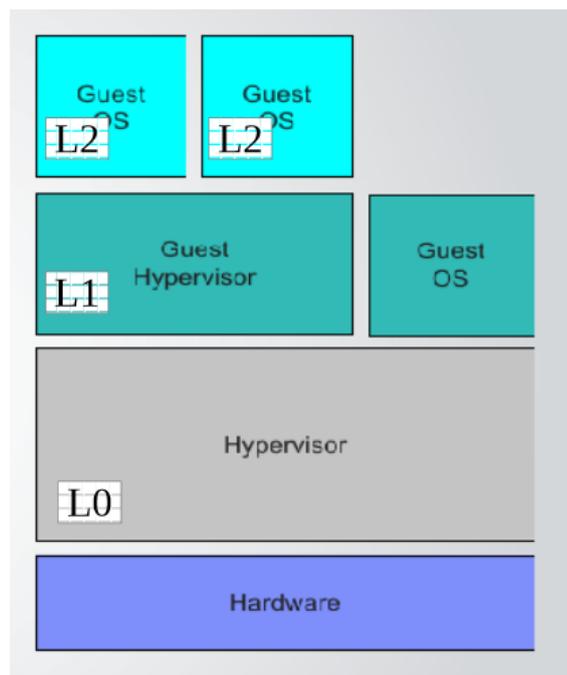


Figure 1: Virtualization layers (Wasserman 2013)

Referring to figure 1, nested virtualization can be defined as four levels. Hardware level which contains normal workstation or server hardware such as RAM memory and CPUs. Layer 0 contains Hypervisor such as VMware ESXi virtualization server or VMware workstation. Layer 1 contains guest hypervisor which is running in L0 hypervisor. Layer 2 is defined as guest operating systems which are running in L1 hypervisor. Nested guest virtual machine does

not have to be homogenous with host virtual machine. So, for example, application virtualization can be done in virtual machine created by using hardware virtualization. (Wasserman 2013.)

In theory there could be even a third hypervisor layer running in layer 2 guest. Performance in hypervisor located in third layer would be poor and in that case it would not be usable. Performance is poor can be caused by lack of hardware virtualization emulation in the guest hypervisor. Because of this, virtual machines running inside third hypervisor are very unresponsive.

Nested virtualization requires modern computer hardware with CPU that includes hardware-assisted virtualization extensions. Some hypervisors would also work without them, but performance is usually poor. Almost every modern cpu includes these extensions. Depending on what is being virtualized, plenty of disk space and RAM-memory is required

2.2 Nested virtualization use cases

Nested virtualization is very popular especially with VMware users around the World. It is used mostly to learn and test computer technologies which would be impossible to do it by reasonable price. Nested virtualization allows cost-effective solution for implementing testing platform instead of buying expensive devices only for learning and testing. (Vcritical 2012.)

It has been found that nested virtualization is good way to do the documentation of software in lab environment instead of doing it in production or other critical environment. It is also safe to test scripts and software in nested environments. (Vcritical 2012.)

3 SOFTWARE USED IN THE PROJECT

Many different type of software and technology were used in this project. Most of these are related to virtualization. The project relates heavily to nested virtualization technology. This chapter discuss only the most important software used.

3.1 Qemu/KVM hypervisor

Qemu, Quick Emulator is open-source hypervisor that performs hardware virtualization. KVM is a kernel module in Linux-kernel that allows Qemu to utilize the hardware virtualization extension features of modern CPUs. It is supported mostly in all recent X86 and x86_64 CPUs. Qemu can fully utilize hardware accelerated virtualization through KVM. (Qemu project web page, 2016)

Qemu is a very versatile hypervisor it can emulate many different types of hardware, for example network, mass storage and IO-devices. Qemu is also fairly simple to configure. Qemu also support other CPU architectures such as ARM, Sparc and PowerPC alongside X86 and X86_64.

3.1.1 Networking in Qemu

Qemu virtual machines have multiple ways to be connected to each other or to network. These network devices are separated in two parts, host devices which are called backend device and virtualized guest devices that are called frontend device.

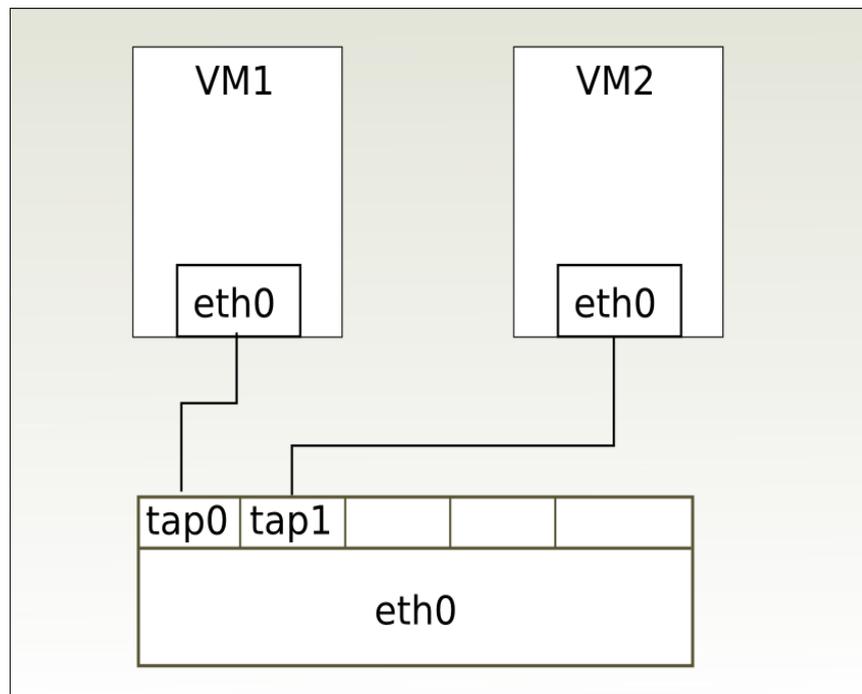


Figure 2: Qemu network connections using tunnel adapter interfaces Qemu Project 2016).

Qemu has some options as backend network device: SLIRP, tunnel adapters, sockets and VDE (Qemu-project, 2016). Most commonly used are tunnel adapters. Tunnel adapters are also used in this project. The Sockets is also interesting way to connecting up two Qemu virtual machine interfaces. It allows to have direct connection between two VM's, but some stability issues occurred and it was ignored in this project.

Qemu has few possible frontend virtual network devices. Most commonly used are e1000, rtl8139 and virtio. e1000 emulates Intel Gigabit Ethernet network adapter which are by default used in Qemu virtual machines. It is supported by most of modern operating systems such as Windows 7 and modern Linux distributions which both have built in driver.

rtl8139 emulates Realtek rt8139 100Mbit/s network adapter. This is supported by default older operating systems such as windows 2000 and XP. Driver for rtl8139 is included in most of the operating systems which have been available in this century.

Virtio has the best performance when comparing rtl8139 or e1000. It emulates ten gigabit Ethernet network adapter. However special driver is required on guest operating systems in order to utilize it.

3.1.2 Qemu display devices and protocols

Qemu has versatile choices to be used access to virtual machines. Most commonly used are virtual displays, but also text based connection is available via virtual serial port. As backend solution there are available SDL, VNC, spice and virtual serial ports.

SDL is used by default in Qemu, but it requires graphical user environment to be used. VNC is another commonly used way to access to virtual machine console or desktop which allows also remote support. Qemu has a built-in VNC-server which can be configured with Qemu parameters (Gentoo wiki, 2015). Qemu VNC-server has some minor performance issues which can be noticed sometimes as unresponsive desktop. It is usually recommended to install additional VNC-server in the guest virtual machine in order to achieve more responsive remote desktop connection into Qemu virtual machine.

SPICE is fresh remote-display protocol for Qemu/KVM virtual machines. It supports currently only 2D-operations, but also 3D-operations are an upcoming feature. SPICE is very efficient remote desktop protocol and it allows properly responsive for virtual machines. It is used by default in Gnome Boxes and Virt-Manager applications which both utilizes libvirt for controlling Qemu/KVM virtual machines (Spice-project 2016.)

Every back-end solution requires a front-end device, a virtual graphic processing unit. In Qemu there are some possibilities which are, cirrus, std, vmware and qxl. Each of them have some special features and requirements.

Cirrus emulates Cirrus Logic GPU device, it is supported in the most operating systems, but it does not have a wide range of supported screen resolutions or color depth options. Cirrus has some 2D graphic acceleration which makes desktop more responsive. (Qemu-project, 2016.)

Std is emulating standard VESA, Video Electronics Standards Association graphics. It supports wide range of screen resolutions but it does not have any graphic acceleration for 2D or 3D graphics. Qemu utilizes this by default as virtual graphic card. (Qemu-project, 2016.)

Qemu can also emulate VMware's virtual graphic card which is very useful to achieve responsive desktop over VNC. This virtual graphic card allows a wide range of screen resolutions and color depths and 2D-acceleration, however it requires OS driver in order to work (Qemu-project, 2016). Most Linux distributions include a driver but Microsoft Windows operating system does not. However this driver can be installed as after installation.

Qxl is a new virtual graphic card for Qemu/KVM guest. By using it, better performance can be achieved when comparing other Qemu virtual graphic cards, which makes the desktop more responsive. Spice remote display protocol is required in order to use QXL driver in the guest virtual machine. It has full 2D-acceleration and support for 3D-acceleration is a planned feature in the future. (Spice-project Website 2016.)

3.2 Guacamole remote desktop gateway

Guacamole is a free and open source client-less remote desktop gateway. It supports VNC and RDP protocols but also SSH and telnet. It is called client-less because no plugins or client software for the end user are required. Only thing what end user needs is a modern web browser with HTML5 capabilities. Guacamole works in normal desktop PCs but also in mobile devices such as smartphones and tables. Guacamole web-application utilizes a tomcat web server to operate. (Guacamole Documentation 2016.)

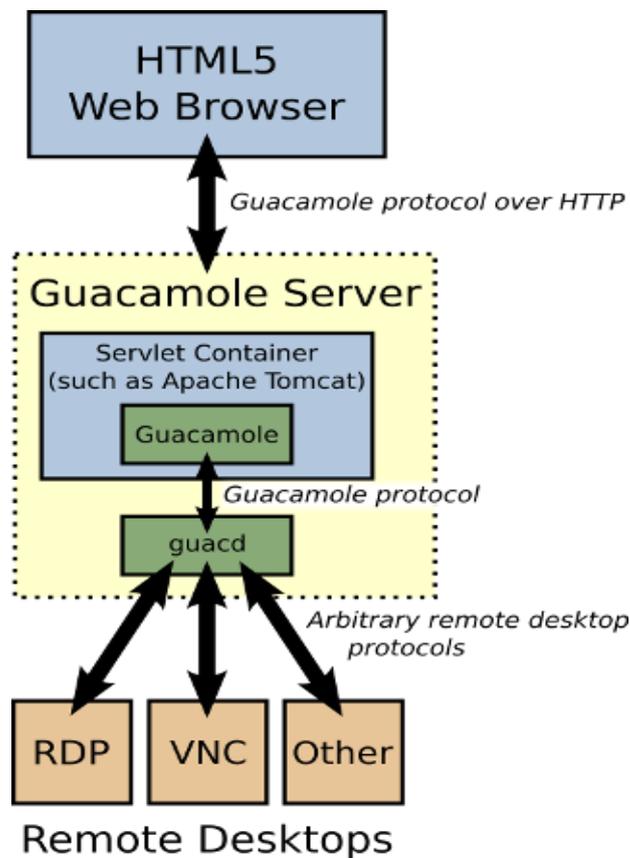


Figure 3: Guacamole architecture (Guacamole Documentation, 2016)

As seen in the figure 3, the Guacamole server operates as middleware between the web browser and remote desktops. (Guacamole Documentation 2016.) This operation allows flexible access into remote desktops via HTML5 client which is running in server.

4 SOFTWARE CREATED FOR THE PROJECT

A number of scripts and applications were made during the project. The scripts were made to controlling and installing laboratory and its devices more easily. Applications were made as workaround solution in order to solve some problems during the project.

4.1 Ebrd, Ethernet Bridge Daemon

Ebrd, Ethernet Bridge Daemon is a multi-threaded packet retransmit software which connects two virtual or physical interfaces with “virtual cable”. Ebrd is written in C-programming language and it utilizes libpcap for read/write operations. Libpcap allows high efficiency packet capturing from the network interfaces.

Ebrd was created to be an alternative solution for Linux bridges, which is the usual way to interconnect two interfaces in Linux environments. During the project it was noticed that at least Fedora 23 and Ubuntu 15.10, Linux bridges had some kind of stability problem when setting bridge up. Bridge might inform it is up and running, but still any data were not going through of it. This problem could be caused by high amount of bridges in the guest hypervisor. Linux bridges also dropped by default the packets which carried dot1q header.

Ebrd has multiple features. In addition to transmissions data packets, it has capabilities to simulate link interference: delay, jitter, packet loss and bit error rate. Another feature is capability control the interference in real time without restarting of the process.

4.1.1 Ebrd technical desing

Ebrd is designed to be lightweight and simple to use. It consists of a packet forwarding engine and a control engine. The packet forwarding engine forwards packets between network interfaces. The Control engine controls some runtime variables that can be modified in real time. It is also designed to be start up with one command, which allows easy use in different scripts, bash scripts for example.

In a start of Ebrd process, it initiates two network interfaces into promiscuous mode and set their value and init two forwarding engines for each interface into different threads. Using different threads for each interfaces was noticed to allow low latency and full-duplex connection.

At the start of forwarding engines, Ebrd allocates memory for a circular buffer, which contains a field for packet data and another field for send time variable, which allows jitter and delay simulation. A circular buffer is very efficient buffer type for asynchronous communication.

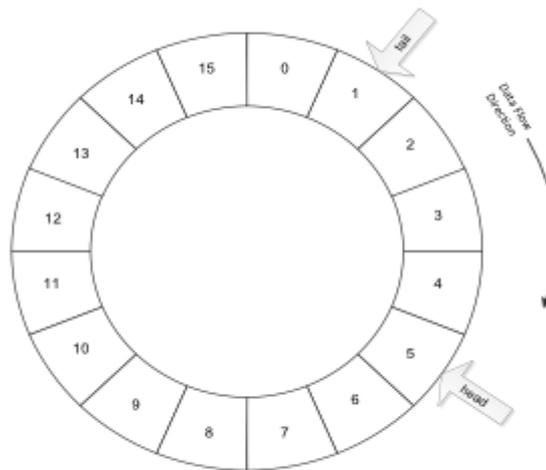


Figure 4: Circular buffer operation (Wada 2013).

Circular buffer is also known as a ring buffer or cyclic buffer. Operation type of this buffer is simple: what comes first in comes first out. Referring to figure 4, when data is pushed into the buffer, the head will move to the next slot, and when the data is read from buffer, the tail will move to the next slot. If data write rate is faster than data read rate buffer head will be equal to tail which mean buffer is full. If buffers are full, there are two possibilities what can happen: all new data frames are dropped or all oldest frames in the buffer are replaced with fresh ones. Both behaviors cause packet loss. (Wada 2013).

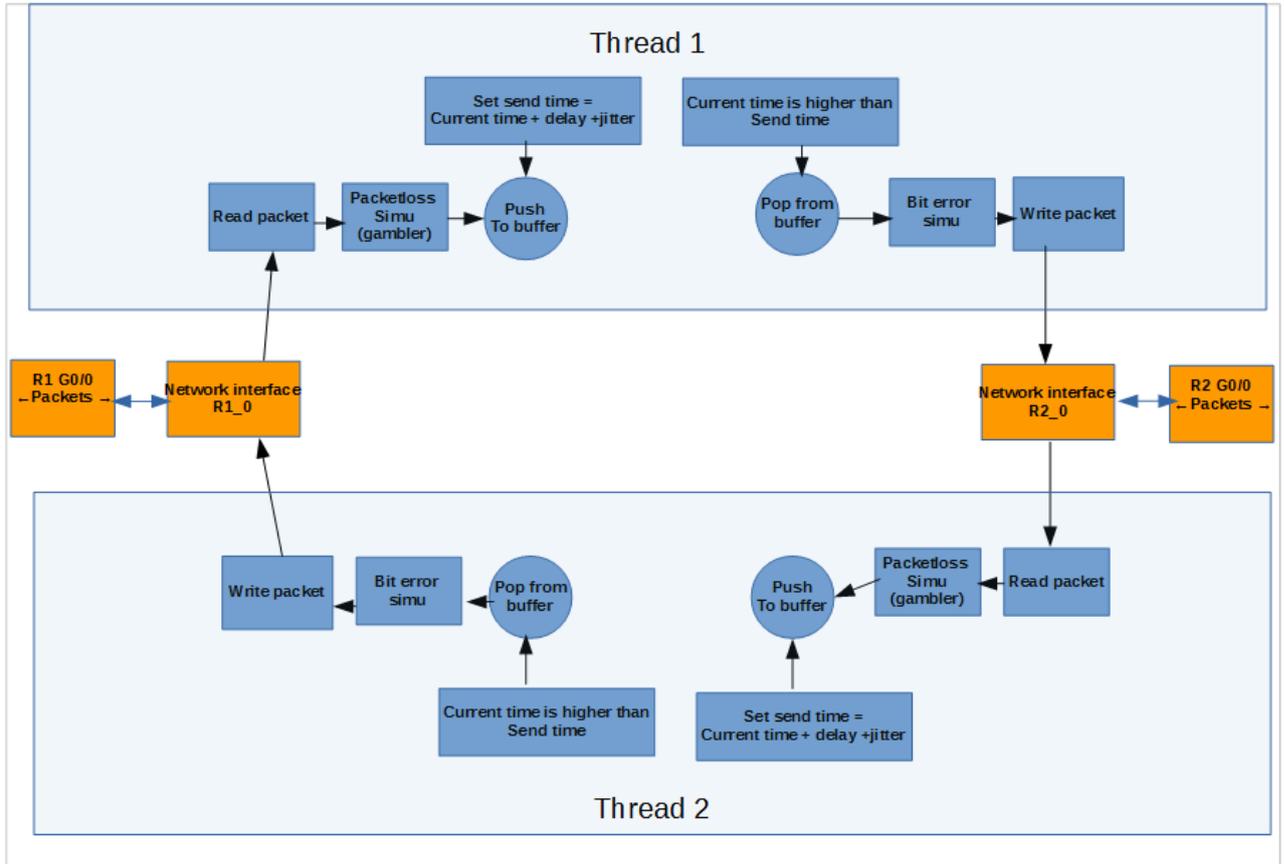


Figure 5: Ebrd operation diagram

Ebrd operation is designed to be simple and efficient as can be seen in Figure 5. After the initiation of a process, it will start forwarding data packets between two interfaces. Once it receives a packet, it will gamble packet loss if packet loss simulation was enabled. Packet loss simulation drop randomly a packet in the change of given a packet loss percentage variable. Resolution is X:1 Million where X refers the change of dropped packets in a million packets.

After the packet loss simulation packet will be pushed into the ring buffer. Ebrd circular buffer has a special variable, send time, which defines time when packet will be popped out from the buffer in microseconds. This variable is set once data frame is pushed into buffer in formula: $\text{current time} + \text{delay} + \text{jitter}$ all values in microseconds. If delay or jitter are defined as zero, a/the frame will be popped out from the buffer immediately. Otherwise it stays in the buffer until current time is equal to send time.

Once packet pops out from the circular buffer it will be written into the outgoing interface. If bit error simulation was enabled, it will modify frame content randomly by defined value.

4.1.2 Controlling ebrd process

To get the link interference simulator usable, a real time controller application for the Ebrd links were also programmed. It was done by utilizing Unix Sockets which is hosted by every Ebrd process in startup of the process. Unix sockets are a common way in Unix environments to use to get different processes communicating each other. The operation of controller application is described in Figure 6.

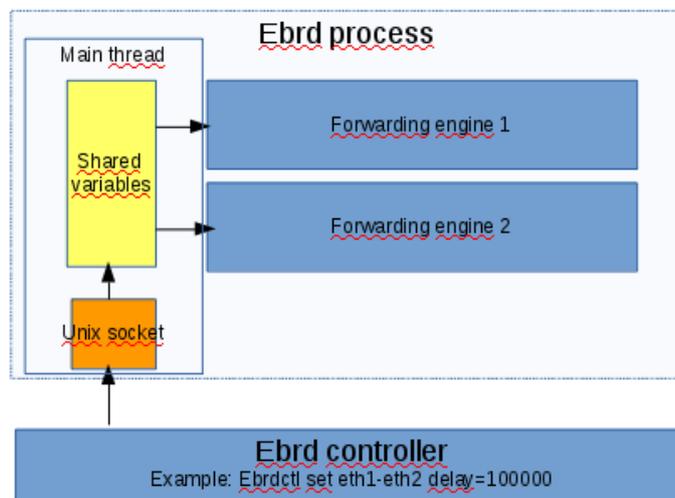


Figure 6: Operation of Ebrd controller.

In Ebrd, Unix socket descriptor is named in simple format: [Interface 1]-[Interface 2], for example eth1-eth2. This naming system allows easy managing of large amount of cables. Unix socket is used to give possibility to change some Shared variables between two threads. Example command to use Ebrd controller is:

```
ebrdctl set eth1-eth2 delay=100000
```

This will set delay in the Ebrd link to 100000 µsecs. Other possible keywords are 'packetloss', 'jitter' and 'ber'.

4.2 TCP-Console Multiplexer

The TCP-Console multiplexer, `tcpmux` is an application which multiplexes one terminal connection into multiple connections. This allows multiple console-sessions for remote console port. It was created as workaround solution to jamming serial console ports in Qemu/KVM.

TCP-Multiplexer is written in C-programming language as simple workaround solution for console port jamming in devices which has only console user access. It is aimed to be simple to use. It takes several command line arguments in order to run.

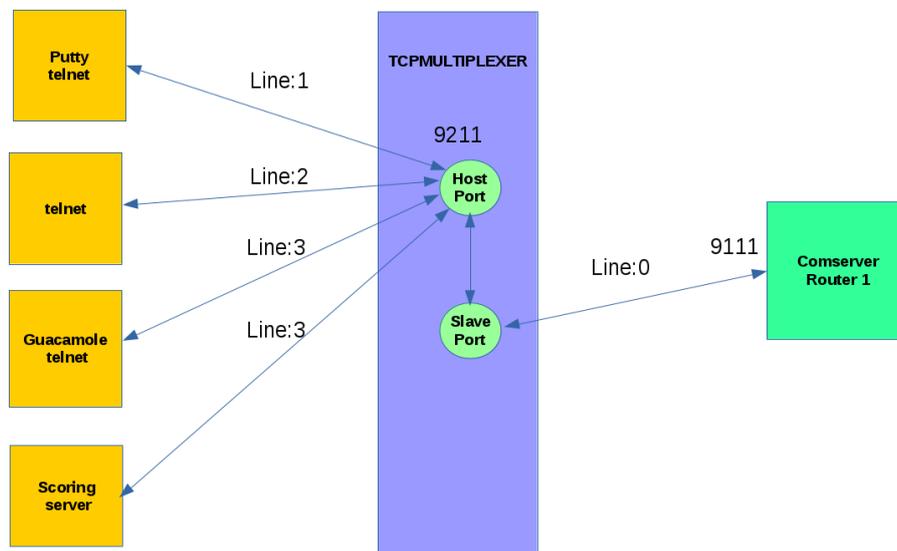


Figure 7: Tcp multiplexer operation diagram

Figure 7 shows how `tcpmux` operates. When devices with a console interface are starting up, a TCP-multiplexer instance starts for every device and it will connect into virtual device console port, for example 9111. Meanwhile Tcp-multiplexer starts TCP listener at chosen host port, for example 9211.

At the initiation of a terminal session with putty or some other telnet client into port 9211, connection will be assigned to a free connection slot in the TCP console multiplexer and it will send 6-bytes: "FF FB 01, FF FB 03", which will put telnet terminal into character mode.

Referring to RFC 854 document, these bytes can be read as IAC, WILL, echo, IAC, WILL, suppress go ahead. IAC means Interpret as a command. WILL means sender want terminal to do something (Postel & Reynolds 1983.) If these bytes are not sent from tcpmux, terminal will not work correctly. Because telnet terminal does not have direct access into real destination, there will not happen protocol initiation.

TCP-Multiplexer could also be utilized in future projects, especially in case someone is going to make some kind of laboratory scoring server or similar. This system could just make a TCP-connection from Scoring system to TCP-multiplexer and start listening incoming data and comparing it to another, for example router's routing tables with correct ones.

5 TECHNICAL DESIGN OF VIRTUAL LABORATORY SYSTEM

Virtual laboratory system is designed to be a stand-alone virtual machine which has pre-configured services and configurations to run virtualized scenarios in it. Virtual devices in different scenarios are running in hypervisor in guest hypervisor.

Linux was chosen to be the guest host operating platform for a guest virtual machines because it is good features especially for networking and virtualization. It also has good programming and scripting tools available. Qemu/KVM was chosen to be guest host hypervisor. It is very versatile hypervisor and as an open source project it is cost effective solution.

The system consists of two parts, a core operating system and scenario. The Core operating system is located in a Nest Core disk and all scenario data is located in a scenario disk.

5.1 NestCore

The NestCore virtual machine is the main core for all nested virtual machines in the laboratory. It is designed to run on VMware hypervisors, but there are no hardware or platform limitations, if it is desired to run on another platform.

Basically, it is just computer or virtual machine which run other virtual machines.

It contains a few services for user interaction and access. Guacamole HTML5 VNC Client is used for accessing the virtual devices running in nested virtual machine. Lighttpd-server with php support for creating Web-based user interfaces for the scenarios.

NestCore is built on Ubuntu Linux-distribution, because it had the most fresh hardware drivers and support for VMware Software Toolkit. Originally it was built on Fedora 23 Linux-distribution, but for the lack of official support in VMware ESXi, it could not be used part of Cyber Game project which utilizes VMware software toolkits for customizing virtual machine.

5.2 Scenario Disk

The Scenario disk is the second part of Virtual Laboratory system. It is secondary VMware virtual disk which called Scenario Disk. During the boot of Virtual Laboratory, if the disk exist in the system in location `/dev/sdb1` it will be mounted automatically to directory `/lab`.

The base file system for a scenario disk contains all the data which each scenario needs, virtual machines and linked clones of them. Also virtual machine controlling scripts, startup scripts and connection scripts are located in this disk. Scenario disk is recommended format to use btrfs file system, which reduces bloating of similar virtual machine clone images

5.3 Operation of Virtual Laboratory

The Virtual laboratory System contains 3 main blocks which are: Virtual Hardware and Disks which is provided by Layer0 host hypervisor and access services. Virtual hardware contains normal computer hardware such as RAM, CPUs, “physical” network interfaces and virtual network interfaces. NestCore disk contains core operating system of laboratory and all pre-configured services, which are used to access and maintain virtual laboratory.

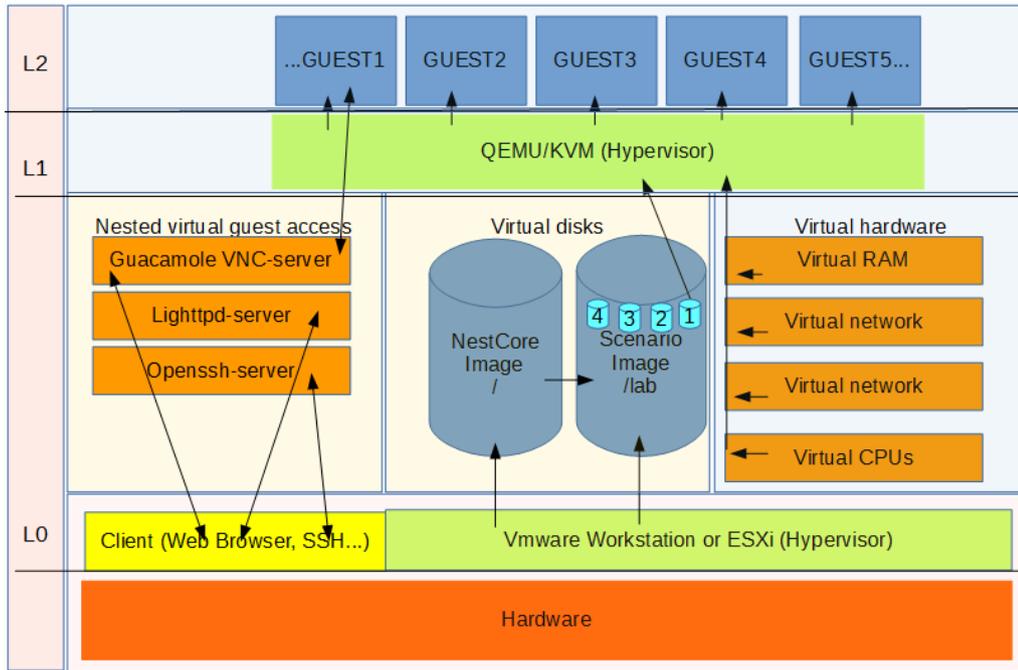


Figure 8: Virtual Laboratory System technical design

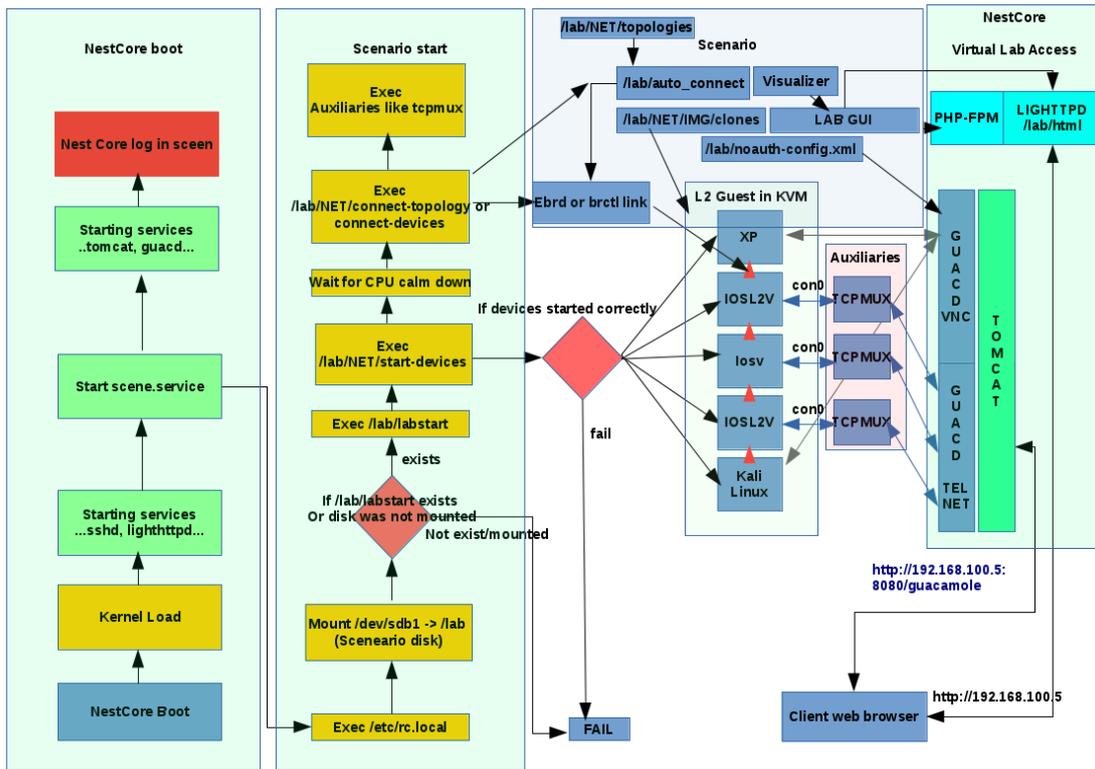


Figure 9: Virtual Laboratory operation diagram

Referring to figure 9, during the boot of NestCore, it mounts the scenario disk to /lab directory. The scenario disk contains all data for scenario which will be

run in. If the scenario disk is attached in to host hypervisor scenario start service in NestCore will execute labstart bash-script which is located at root of the Scenario disk, /lab. Labstart bash-script is a scenario based script and scenario author can modify it to meet needs of each scenario.

5.4 Virtual device installation and configuration

Installing a virtual machine into the laboratory was designed to be simple. In the first versions of the Virtual Laboratory system, virtual machine start and connection scripts were done by hand, which required a lot of time. Lab installer script was implemented to solve this problem. It creates all needed bash scripts automatically from given parameters. By using scripts, it is only needed to do is copy the generic virtual machine base images of virtual machines to the base image directory and create an installation file with text editor, such as GNU/Nano or notepad.

The installation file describes a name of virtual machine name, a base virtual machine image, amount of memory, cpu cores and network interfaces for the virtual machines. Virtual machines name is used as identification which means that there cannot be two devices with the same names. These interfaces are named in NestCore -system with syntax [virtual machine name]_[interface number] without.

Installation file syntax:

```
DEVICE [NAME] [IMAGE] [MEMORY] [CPU COUNT] [INTERFACE COUNT]
[ACCESS PROTOCOL] [PORT] [Graphic driver][Snapshot mode]
```

Example

```
DEVICE R1 iosv155m.vmdk 348 1 4 telnet 9111 - 0
DEVICE PC1 tc-host.vmdk 256 1 1 vnc 5901 vmware 1
DEVICE PC2 labxp.img 256 1 1 vnc 5902 cirrus 1
```

The Installation file is loaded into lab installation script, which will create linked clones from given base images into the system. Linked clones are very space efficient virtual machine clones. Those allow multiple similar virtual machines to be stored in relatively small disk space instead of using full clone. Linked clones

Once script is executed, it creates following scripts automatically: start-devices, start-multiplex and guacamole access configuration file noauth-config.xml.

5.5 Virtual laboratory networking

Virtual Networks in Virtual Laboratory system was designed to be flexible and simple to create. It has possibility interconnect virtual machines together through tunnel interfaces, which Qemu/KVM has created at VM startup. Tunnel interfaces were chosen to be backend solution for virtual machine network connections, because those are stable and simple to utilize.

Tunnel interfaces are named in format [Device]_[Interface number], for example Router R2 interface G0/1 is named as R2_1. This naming format method allow simple scripts and topology files to be utilized because of no need to keep any database about the names of tunnel adapters or virtual machines.

Interconnections can be done with Linux bridges or with Ebrd, which was programmed to serve needs of this project. Linux bridges can be created by utilizing brctl and ifconfig -command.

For example:

```
brctl addbr br0  
brctl addif br0 R2_1  
brctl addif br0 eth4  
ifconfig br0 promisc up
```

First command setups bridge name br0, and to next commands add both interfaces into bridge. Last command sets bridge to promiscuous mode and up. When interface is in promiscuous mode, every transmitted data packet can be received by a network interface (Techtarget 2016).

When utilizing ebrd, connection can be set up with single command. For example:

```
ebrd R2_1 eth4 1 500 0 0 0 9226
```

For example, upper command connects up interfaces R2_1 and eth4 and set the controlling mode on. Ring buffer size is set to 500 entries and all interference simulations are set to 0. Tunnel adapter's MTU are set to 9226 Bytes. High MTU in tunnel adapters is required to get protocols such as MPLS to operate correctly through the link.

Virtual networks are meant to be configured with topology files, which contains virtual devices with X and Y coordinates and Cables. Only cables are required to define in topology files in order to get the virtual network operational. Device definitions are on required for an interactive topology map in Web user interface. More about the web user interface in Chapter 5.6.

Syntax of topology file is designed to be simple.

For devices:

```
[Device type] [Name/ID] [X] [Y]
```

For cables:

```
CABLE [SRC DEV] [Port] [DST DEV] [Port] [Desc]
```

Example content of topology file:

```
ROUTER R1 300 200
ROUTER R2 500 200
ROUTER R3 700 200
WORKS PC1 50 200
WORKS PC2 1000 200
ETHX eth4 500 600
CABLE PC1 0 R1 0 192.168.0.0/24
CABLE R1 1 R2 0 10.0.1.0/24
CABLE R2 1 eth4 -1 10.0.2.0/24
CABLE R2 2 R3 0 10.0.3.0/24
CABLE R3 1 PC2 0 192.168.3.0/24
```

Referring to example topology file content, content of each topology goes into small size and it is in simple readable format. Readable format allow easy creation and modifying of topology files. Example topology uses only a few type of devices, a full list of different device keywords is documented in a user manual of the Virtual Laboratory System. Figure 10 shows visually the connections and how those looks in the guest host operating system.

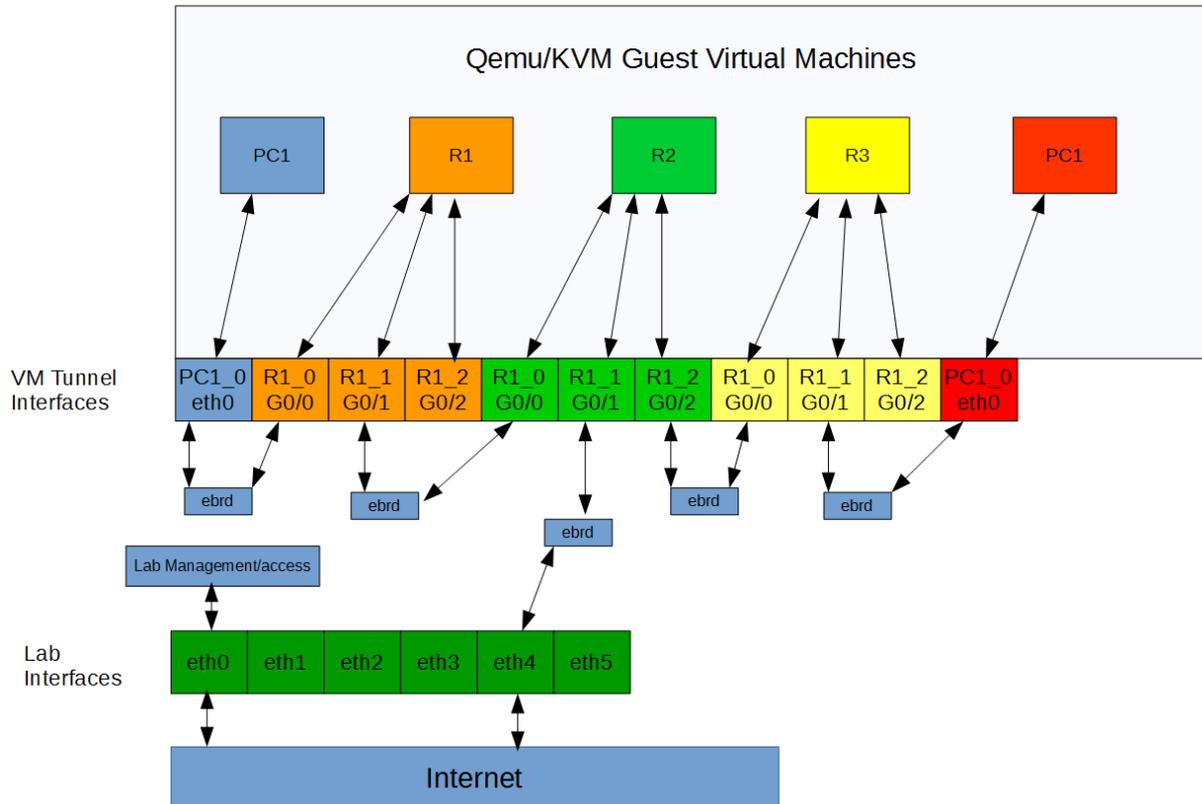


Figure 10: Virtual Laboratory System interconnections diagram when using Ebrd.

5.6 User interface

A simple user interface for accessing into the devices was a very important thing to implement, because it increases the usability of the whole system. It allows user to see a network topology in visual format and access to devices easily.

Web user interface was developed in HTML5, php and Javascript programming languages. These programming language were chosen because they are standardized way to create web pages. User can access to it easily by accessing to [http://\[Lab IP Address\]/](http://[Lab IP Address]/) with modern Web-browser with HTML5 support.

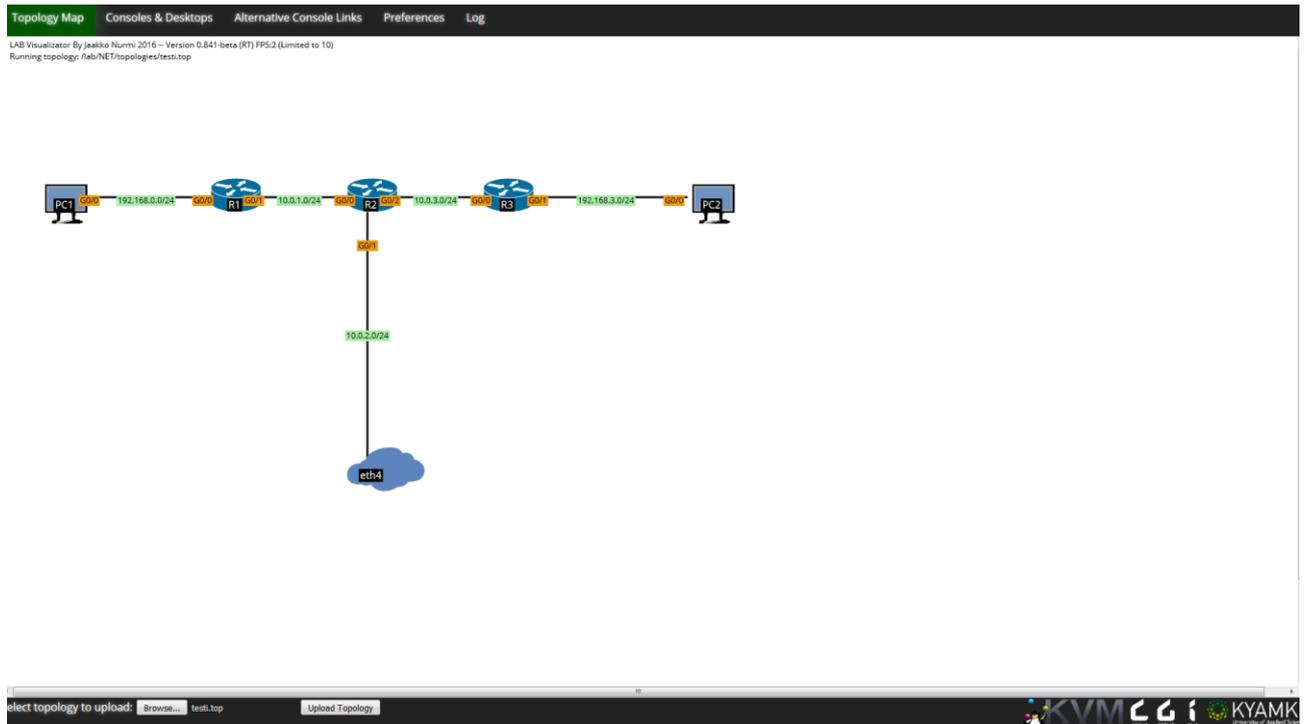


Figure 11: Screenshot from web user interface showing same example topology that was mentioned in chapter 5.5

As can be seen in figure 11, user interface has been designed to be simple. User can click a certain device to open its desktop or console. These will be opened in new tab in the web browser. It may require user to allow pop-up windows in the lab user interface. Consoles and Desktop tab show a list of consoles and desktops that are available in in current installation. In the preferences tab, a user can find options to manage the laboratory. These options include lab settings, cable settings and virtual device manager.

At lab options user can also set topology or change laboratory installation. Cable options allow access to change Ebrd link interference parameters. In virtual device manager user can modify, add, delete, install, stop and start devices.

To set topology, user can easily upload it via form at the bottom left corner of the user interface. Due to the small size of the topology files, each lab exercise could be easily shared by a teacher for example via Moodle eLearning platform.

6 USE CASES OF VIRTUAL LABORATORY SYSTEM

Virtual Laboratory has multiple use cases. Originally its primary use case was to be core KVM Host to Cyber Game -project. Later it got more uses cases as being a platform for Routing and Switching laboratory, RSLAB and Vesa Kankare's service provider network laboratory, SPNET. Later all networking related laboratories were packed into the same packet.

6.1 Virtual Networking Laboratories

Virtual Laboratory the main use case is to virtualize networking related study units. RSLAB and SPNET were combined into one packet in order to give better maintenance possibilities compared to have two similar virtual laboratories in different packages. At a start of the laboratory, user can select laboratory installation, which are SPNET and different variations of RSLAB.

As Virtual Laboratory System is utilizing Cisco System's virtualized network device images, IOSv, IOSvL2, IOS-XRv and ASA-v, realistic networking simulation can be achieved. These laboratories were used and tested in networking related study units: Advanced Routing, Data Security Equipment and Service Provider Network.

6.2 Cyber security training scenarios

Two cybersecurity hacking scenarios were made as proof of concept during the project. Both scenarios were built on a view of hacker. Building scenario in a view of defender is also theoretically possible, but any proof of concept were not done. These scenarios allowed a student to step into the boots of hacker. The main goal in these scenarios is to solve predefined hacking task with real life tools.

In technical perspective, these cyber security scenarios are identical to virtual networking laboratory scenarios. Only differences is that only hacker's workstation is accessible via user interface. Also there is no interactive view of network topology.

In each scenario, user get for use one Kali-Linux, which is a very versatile operating system for professional it security testing. The scenarios contains many hidden network devices, servers and workstations where user must get access.

First scenario was only a simple test. There was a fictive Internet Service Provider with a small network, workstation and web server. Student goal was to get access to the secret server at ISP's core network by getting information about target and its network.

Second scenario was more complex testing for the platform, it contained more workstations, network devices and servers. The main goal for the student was to get access to the blocked website, which was blocked by firewall at Campus Edge.

Results of these test cyber security scenarios were impressive, because high amount of different types of devices were installed in relatively small disk space. A second scenario was included four Ubuntu-Linux servers, four Windows XP workstations, one Kali Linux workstation and 3 routers and 2 switches. The total size of the scenario was only 17 GB. This was possible because of the use of the btrfs file system in the scenario disk and linked VM-clones. Even more virtualized devices could have been added without increasing much scenario disk size.

6.3 Future use cases

Virtual Laboratory can be used easily also in different types of scenarios. One of them is a data center scenario, where Virtual Lab contains completely virtualized data center equipment, such as network devices, application delivery controllers and servers.

This type of scenario could be used as practical exercise environment in study units which are related to data centers. Having full data center equipment as hardware for each student are very expensive investment.

7 PROJECT RESULTS AND BENCHMARK TESTS

This chapter discusses tests that were performed to measure the capabilities of Virtual Laboratory System, especially the performance of the virtual network using two technologies. This chapter also discusses the benefits and results of the project and user feedback given by end users.

7.1 Benefits of Virtual Laboratories

Virtualized laboratories have many benefits over hardware laboratories. One and the most important is cost-effectiveness, which is great compared to hardware laboratories. As the Virtual Laboratory System was created mostly by using open source technologies, there are no license fees or additional hardware cost. The only cost when using virtualized laboratories is the hardware cost of the workstations or the servers, which run the laboratories. In hardware laboratories, network devices are expensive and one device with proper licenses can cost more than one powerful workstation which can simulate the whole set of the devices.

In virtual laboratories, each students get their own set of devices, which means that it is not needed to share devices with co students. This may increase learning level of the student, because they will have the whole network under their own control. Virtual laboratories also allow a teacher to do the same exercises same time with students. In the hardware network laboratory, this is not normally possible, because of lack of the devices.

7.2 Ebrd performance test against Linux bridges

As a proof of concept, a few benchmark tests were carried out against Linux bridges. These benchmarks were done between two virtual machines running Tiny Core Linux with one e1000 virtual network adapter. Both virtual machines had 1 logical CPU and 1024 Megabyte of RAM-memory. Host CPU clock frequency was 3.6 GHz. Tested things were latency between two VMs when

idling and in high load. Another test was data rate test which were done with iperf3 benchmark tool which is versatile tool for testing network performance.

Test result are not absolutely reliable, because host machine had other unnecessary processes running and each test were executed only 10 seconds. Result of the test are still approximate.

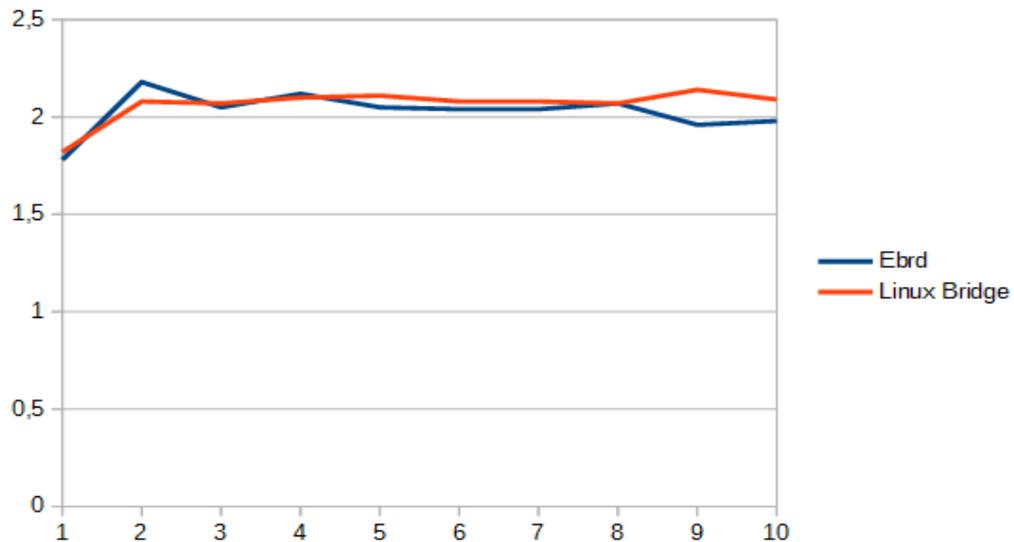


Figure 12: Iperf3 TCP performance test: Ebrd vs Linux bridges Y-axis are Gbit/s and X-axis is seconds. Higher is better.

There is almost no difference in performance between Ebrd and Linux bridges. During the test Host CPU usage was 40.1% when using Ebrd and 37.2% when using Linux Bridge. This small difference could be a problem when using high data rates through multiple devices. However data rates over 10Mbit/s are not usually used in network simulations or in future Cyber Game Scenarios.

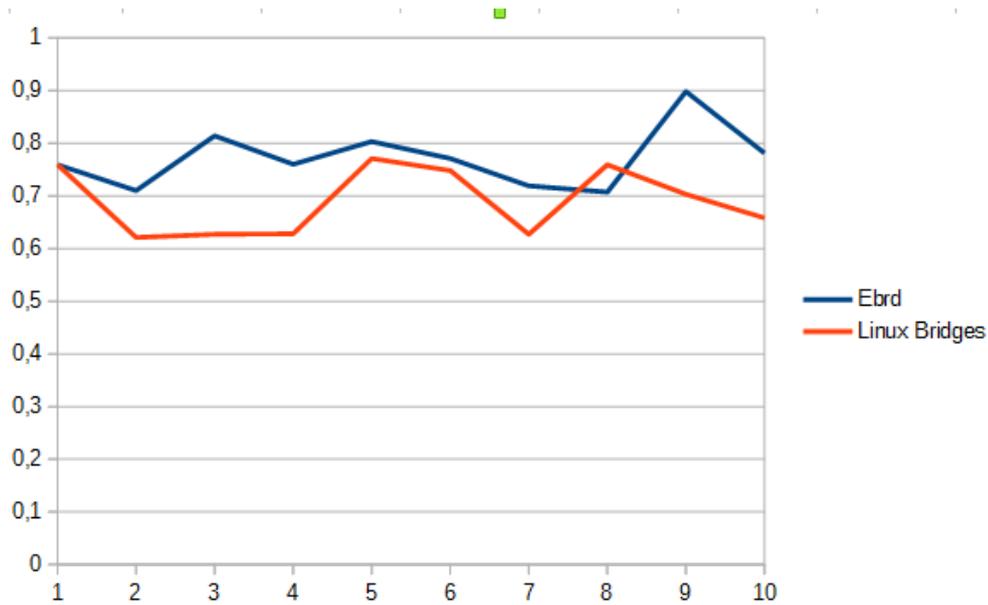


Figure 13: Link latency test (low load): Ebrd vs Linux bridges Y-axis are milliseconds is X-axis is second. Lower is better.

Link latency test measured latency in the link. It was tested with ICMP echo-method packet size being 1000 Bytes. Average latency while using Ebrd was 770 μ sec and by using Linux bridges 690 μ sec.

Linux bridges got 80 μ sec advantage over Ebrd in this test. This difference could be caused by Ebrd data polling with 2 msec timeout and buffering which in theory increases latency.

Originally, Ebrd had possibility for direct pass through for data frames without buffering, but it was replaced with buffering method in newer versions. Buffering the data frames requires more memory to be allocated per process than direct pass-through method, but without buffering, delay or jitter simulations would not be possible to implement.

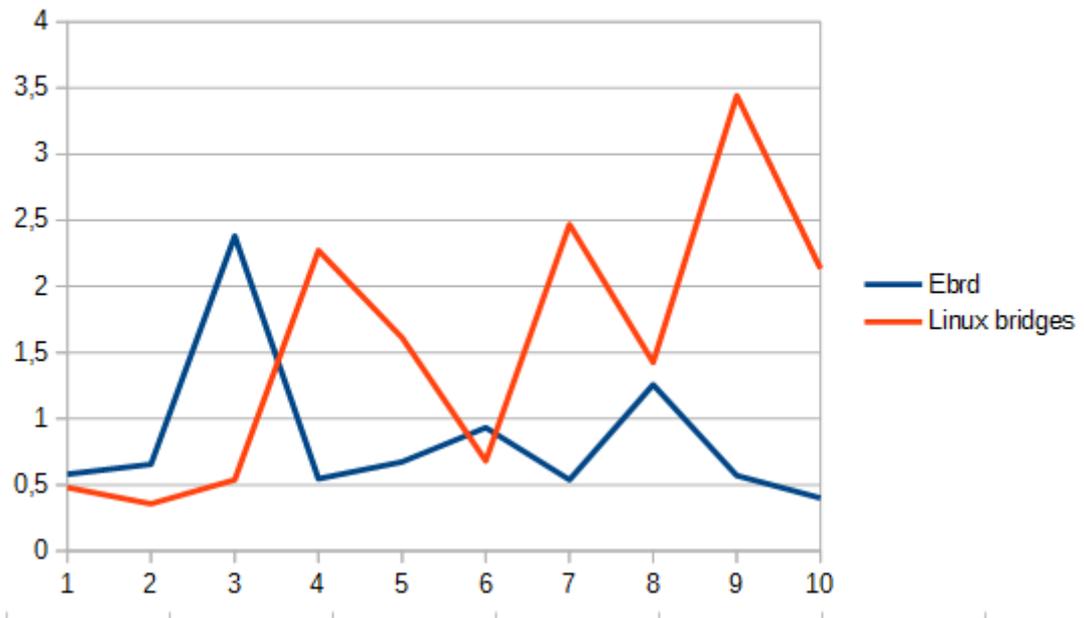


Figure 14: Link latency test (high load): Ebrd vs. Linux bridges Y-axis are milliseconds is X-axis is second. Lower is better.

The link latency test under high load is a similar test than previous. In addition it has high load in the link generated with iperf3. Referring to figure 14, in this case Ebrd took advantage over Linux bridges. Ebrd average latency was $853,7\mu\text{Sec}$ and $1541,1\mu\text{sec}$ for Linux bridges. This behavior can be explained by how Ebrd read data. Polling rate in Ebrd is increased by high data flow through it which means, that poll is returning data almost immediately instead of waiting for timeout.

7.3 User Feedback

In the end of the project, user feedback was collected from end users. Questions what were used was the same than in Vesa Kankare's Master Thesis because comparable results were wanted. There could be more questions, but in this case anonymous open feedback and development ideas are more important.

The questions measured how Virtual Laboratory System was working in view of end user and how it helped to learn compared to hardware labs. Total amount of answers was 31 from 42 users. Figures 15, 16 and 17 has graphical diagram of answers of questions. Raw data of answers can be found from Appendix 1.

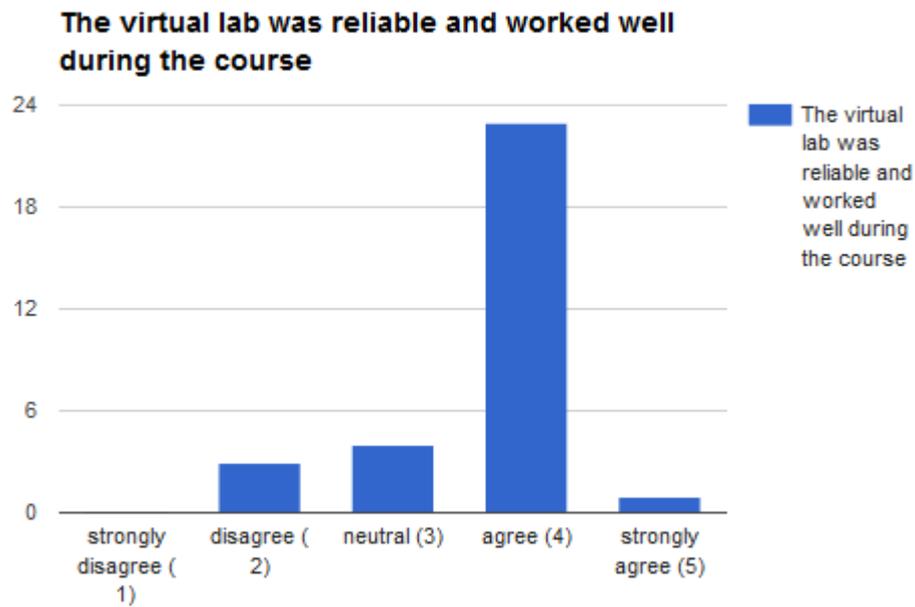


Figure 15: The virtual lab was reliable and worked well during the course. (1-5).

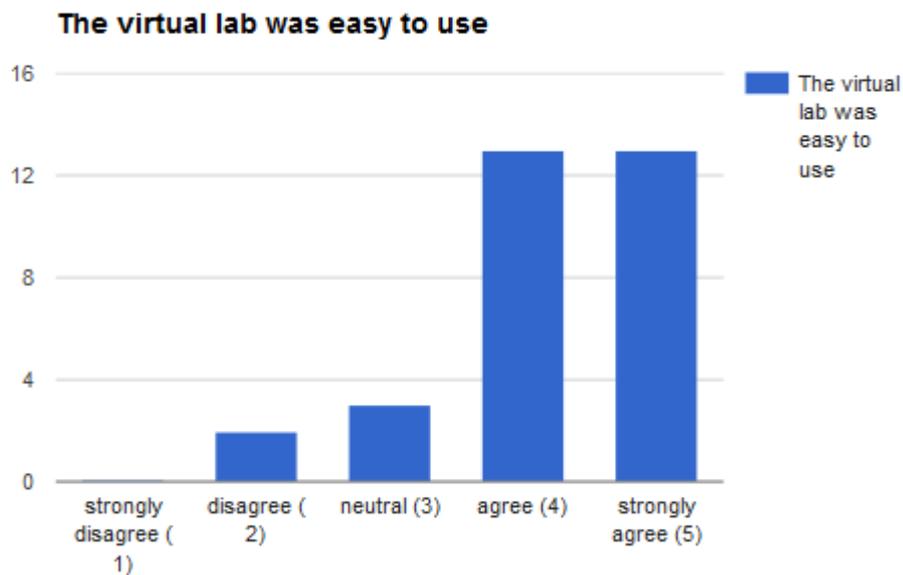


Figure 16: The virtual lab was easy to use. (1-5)

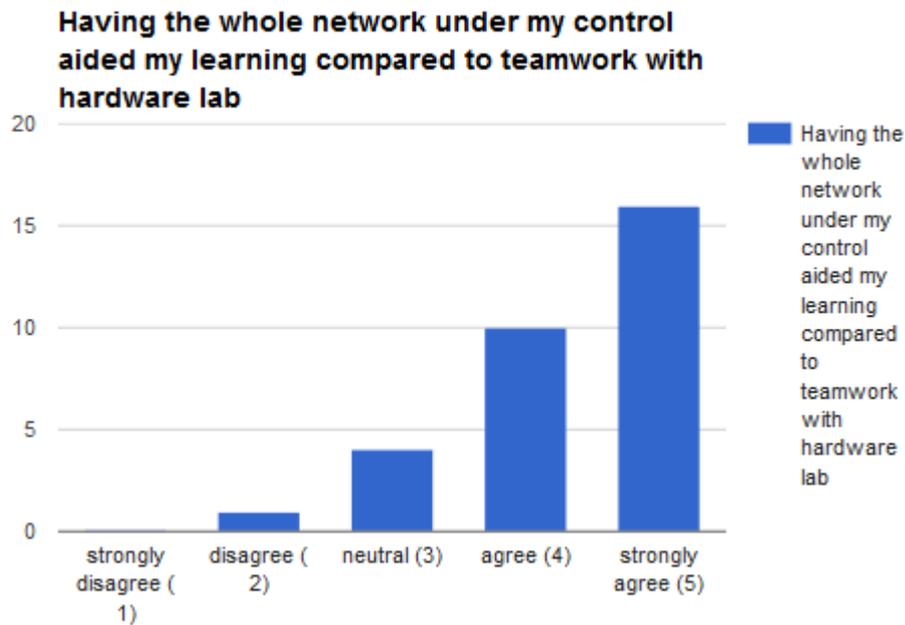


Figure 17: Having the whole network under my control aided my learning compared to teamwork with hardware lab. (1-5)

Overall of the answers of open feedback from the end users were mostly positive but also some negative feedback was given. Negative feedback comments were mostly given because of low reliability in some cases. The criticism on low reliability in some cases is justifiable, because the whole system was under heavy development as it was simultaneously used in lessons. Also a few problems in the host platform which caused sometimes connectivity issues to Virtual Laboratory. Also open feedback and development ideas were asked from the end users. A few of them are mentioned under and the rest of them can be found in Appendix 1.

“The idea of this lab is quite amazing. It frees you from many obstacles during studying such as connecting the cables or teamwork. It's much easier to work if you control all the network. I myself don't like teamwork much. You can work in virtual lab with any pace you want and you don't have to ask anyone if you want to try out things. Of course you still can discuss your work with others but you're not dependent on them anymore. Virtual lab had some issues but most of them were either introduced by VMWare or by misuse. I had a chance to see how it evolves. And it improved a lot. I also had a chance to use VIRL and I can say that this lab works perfectly compared to VIRL. It doesn't have unnecessary complexity, it has pretty web-based interface. It feels pretty much

like real lab but you can even see when traffic is going through the link. There are still things that can be improved (or more like features added) like one-click configs load. dynamic adding and removing of devices. But even now it's an amazing experience of reliable and sophisticated software."

"The overall feeling about the Virtual labs (for me atleast) is negative, because we had so many issues etc. However when everything was working the virtual labs were great addition to our learning experiences. I think we had the most problems as we were kind of the "test" group so I believe that later students will have better experience with the virtual labs."

"The virtual lab worked quite well during the courses, and I can't wait to see where the lab will be used in the future. Of course there were minor bugs but we were test users so that is normal. I hope that in the future ICT-lab gets better computers so that we can run more complex simulations in virtual lab too."

These answers show that most of the users had no major problems with the virtual laboratories. Some minor problems in the laboratory related in reliability can be agreed because most of the features were not carefully tested before. Most of the problems were related to VMware authentication service and Windows networking features.

8 FUTURE POSSIBLE DEVELOPMENT IDEAS

Many ideas has come into mind during the project. This chapter presents a few of them as future development ideas for the Virtual Laboratory System and how they could be designed in simplified format.

8.1 Virtual Laboratory control protocol

The Cyber Game project stand up to VMware's Software Development Kit when cloning and controlling as. This would lock the host hypervisor platform

only to one vendor, VMware. If someday the platform is changed to another, incompatibility issues can occur.

As workaround for this possible problem, there could be simple control protocol running in Virtual Laboratory which operates in a link layer. When operating in the link layer, communication is done with MAC-addresses. This could be useful to configure Network layer addressing into a number of Virtual Laboratory instances. The only limiting factor is that this control protocol and virtual laboratories must be located in the same broadcast domain in order to work. Figure 18 shows the idealistic operation of the controlling protocol.

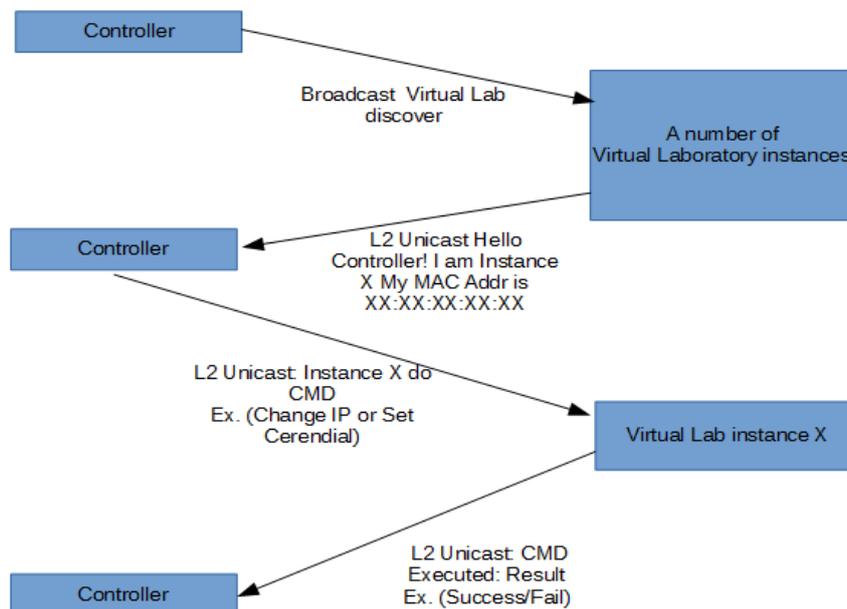


Figure 18: Possible operation of control protocol running at link-layer in simple format

8.2 Implementation of laboratory reservation system

In future, if Virtual Laboratories installed into server cluster to serve needs of remote students, there should be possibility to reserve laboratory time by user. There are number possibilities to implement it. One possibility is by using F5 ADC, Application Delivery Controller. Another solution is to implement standalone solution for it.

As standalone solution, it could be simpler than doing it with ADC. For example, there could be X amount of Virtual Laboratory instances idling in Esxi cluster and user could grab one of them via web user interface.

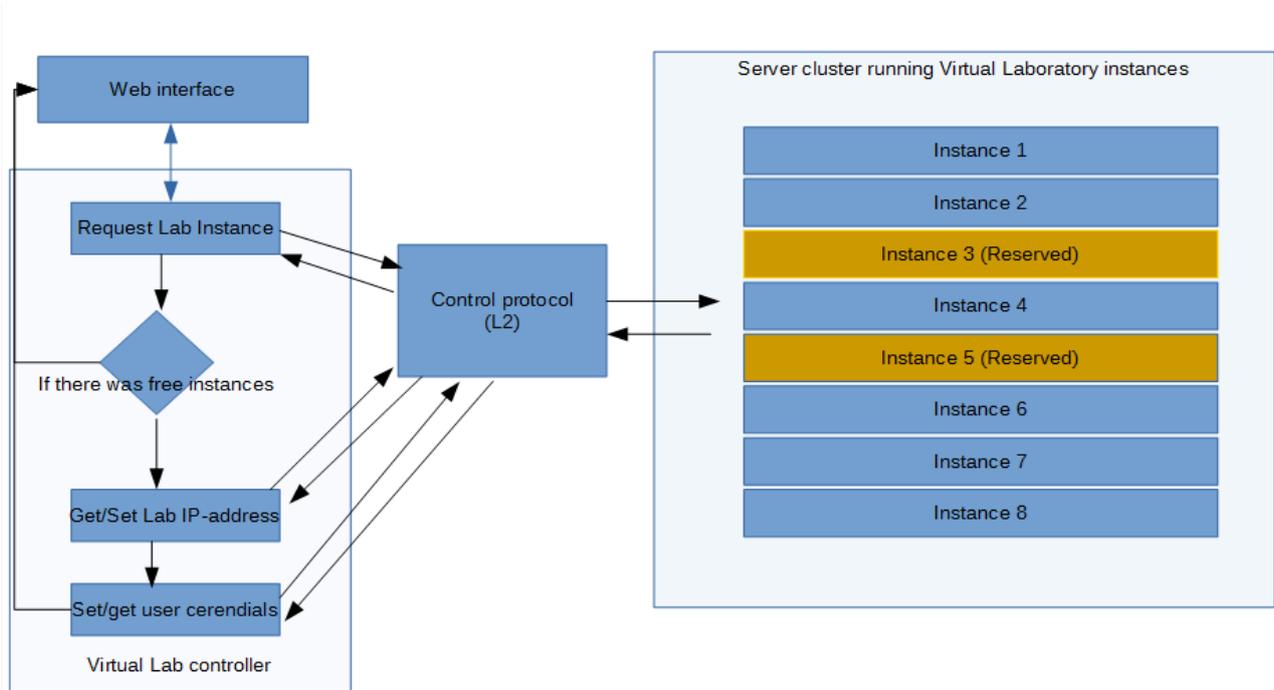


Figure 19: Simplified idealistic operation diagram of possible standalone solution.

Referring to figure 19, the operation of the reservation system could be very simple. Just need to use possible control protocol or another technology to check for free laboratory instances and do the reservation.

8.3 Multi user support

In future, there could be demand for multi user support for single Virtual Laboratory instance. Instead everyone having their own copy of Virtual Laboratory System, it could be installed only one time into every machine where it is needed and each user virtual device images could be saved in a server. Disk space would not be a problem, because linked clones do not take much space in normal situations. This method would allow user to use any computer and do exercises or projects in anywhere. Figure 20 shows the idealistic implementation of the multi user support.

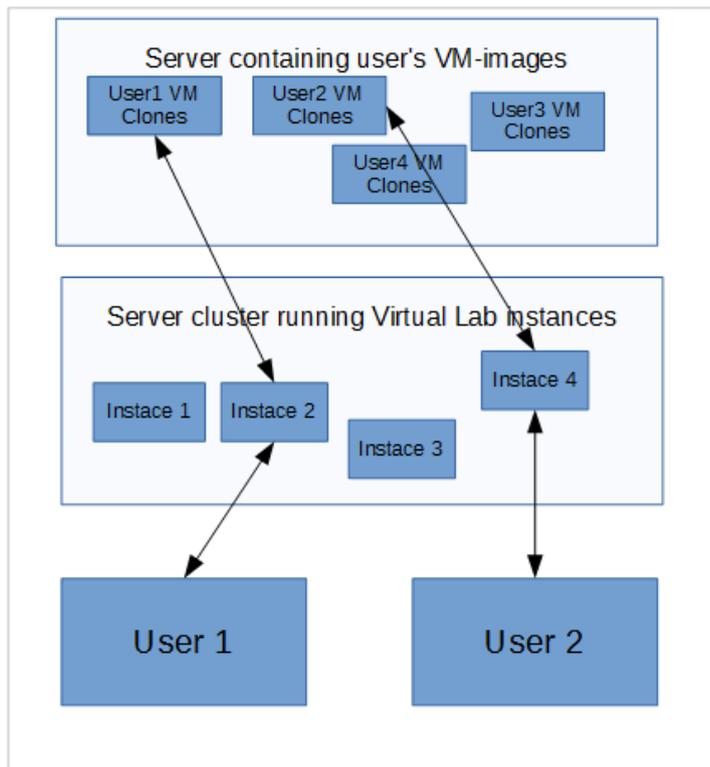


Figure 20: Possible way to implement multi user support.

Another possibility is to save user's cloned VM-images in local the folder in every Virtual Laboratory instance, but it would ruin the idea for having good support for remote connections.

8.4 Implementing solution to use SPICE-protocol through web browser.

Responsiveness of virtual desktop by using VNC remote display protocol is good but not excellent. It could be improved by using SPICE remote display protocol using QXL as a fronted driver. Using SPICE protocol via web browser would be usable in future especially in Cyber Game scenarios where user has only Kali Linux desktop. Well responsive desktop in these scenarios increases usability of them.

While writing this, SPICE has experimental HTML5 -based client available, which could be utilized. The current version of Guacamole (0.9.9) does not offer support for SPICE. Guacamole is open source software, anyone can program support for it.

9 CONCLUSIONS

The topic in this bachelor thesis has been very wide and it would be never be complete, because information technology and especially virtualization technologies are improving remarkably in a short time. The implementation process of the virtual cable, Ebrd was a complex project and it could have been also expanded into a full bachelor thesis topic.

The project has been very instructive and many technical and non-technical things were learnt. From the technical perspective operation of buffers, especially circular buffers and some link-layer protocols were learned. Also a lot has been learned about programming and scripting.

As non-technical thing, because Virtual Laboratory System, was used in few study units same time than it was developed, often critical bug was found which required to fix bug problem quickly. This was very instructive, because it required to delimit the problem and find the faulty thing what caused the problem.

Many new ideas were got during the project and most of them were implemented but time being the limiting factor, all ideas were not possible to implement. A few sub topics of the project was very wide and time consuming. In my opinion project main goals of the project were perfectly completed, and it result of the project is ready to be used also in future.

10 REFERENCES

Cisco Virl Website. 2016. Available: <http://virl.cisco.com/>. [Referred April 4, 2016].

Gentoo wiki. QEMU/Options. 2015. Available: <https://wiki.gentoo.org/wiki/QEMU/Options>. [Referred March 10, 2016].

GNS3 website. 2016. Available: <https://www.gns3.com> [Referred March 3, 2016].

Guacamole Documentation. 2016. Available: <http://guac-dev.org/doc/gug/guacamole-architecture.html>. [Referred March 19, 2016].

Postel J, & Reynolds, J. Telnet Protocol Specification. 1983. Available: <https://tools.ietf.org/html/rfc854>. [Referred February 2, 2016].

Kankare, V. 2015. Implementation of virtualized service provider study unit. Master Thesis. Kymenlaakso University of applied sciences. Available: https://www.theseus.fi/bitstream/handle/10024/91528/Kankare_Vesa.pdf?sequence=1. [Referred February 20, 2016]

Kankare, V. 2016. Tietoverkko-opetus virtuaalistuu. Koskinen 2/2016, s 1 Available: <http://www2.kyamk.fi/Koskinen/022016/virtualisointi.html>. [Referred March 23, 2016]

Qemu Project website. 2016. Available: <http://www.qemu.org>. [Referred February 5, 2016]

Qemu Project. 2016a. QEMU Emulator User Documentation. Available: <http://wiki.qemu.org/download/qemu-doc.html>. [Referred February 27, 2016]

Qemu Project. 2016b. Documentation/Networking.. Available: <http://wiki.qemu.org/Documentation/Networking>. [Referred March 12, 2016]

Spice-project Website. 2016. Available: <http://www.spice-space.org/>. [Referred March 19, 2016]

Techtarget. 2007. Definition of Promiscuous mode. Available: <http://searchsecurity.techtarget.com/definition/promiscuous-mode>. [Referred March 4, 2016]

Techtarget. 2010. Virtualization hypervisor comparison. Available: <http://searchservvirtualization.techtarget.com/tip/Virtualization-hypervisor-comparison-Type-1-vs-Type-2-hypervisors>. [Referred March 7, 2016]

Vcritical. 2016. What are you doing with nested virtualization?. Available: <http://www.vcritical.com/2012/02/what-are-you-doing-with-nested-virtualization/>. [Referred March 26, 2016]

Wada K. 2013. Ring buffer basics. Available: <http://www.embedded.com/electronics-blogs/embedded-round-table/4419407/The-ring-buffer>. [Referred March 13, 2016]

Wasserman Orit. Kvm Forum - Red Hat Inc. 2013. Available: <http://www.linux-kvm.org/images/e/e9/Kvm-forum-2013-nested-virtualization-shadow-turtles.pdf> [Referred February 1, 2016]

APPENDIX 1 - Result of user survey and open feedback and development ideas from end users

	Total answers	31	of	41	
	Questions	3	and Open	feedback	
Question	strongly disagree (1)	disagree (2)	neutral (3)	agree (4)	strongly agree (5)
The virtual lab was reliable and worked well during the course	0	3	4	23	1
The virtual lab was easy to use	0	2	3	13	13
Having the whole network under my control aided my learning compared to teamwork with hardware lab	0	1	4	10	16

Koska kurssissa käytettyä virtuaaliverkkoa ei oltu aikaisemmin testattu, hidasti se kurssilla tehtyä suurempaa työtä. Erittäin kätevä tapa järjestää kurseja, jotka sisältävät konfigurointia, kunhan ongelmat ovat ennen aloittamista korjattu. 4/5

It would be nice if the devices had an easier way to copy the configurations. It is really time consuming to copy the configs to each device individually.

The overall feeling about the Virtual labs (for me atleast) is negative, because we had so many issues etc.. However when everything was working the virtual labs were great addition to our learning experiences. I think we had the most problems as we were kind of the "test" group so I believe that later students will have better experience with the virtual labs.

Using virtual lab was easier to learn compared to teamwork with hardware lab

For first time user access links are available in the VMconsole but web UI access link could be emphasized more. In build guide for copy-paste tools would also make it easier to start working with the lab. Lab is convenient to use and overview is excellent.

Lab could include clear instructions for using putty connections

The virtual lab worked quite well during the courses, and I can't wait to see where the lab will be used in the future. Of course there were minor bugs but we were test users so that is normal. I hope that in the future ICT-lab gets better computers so that we can run more complex simulations in virtual lab too.

Needs some more development and some bug fixing, otherwise 5/5 setup.

The idea of this lab is quite amazing. It frees you from many obstacles during studying such as connecting the cables or teamwork. It's much easier to work if you control all the network. I myself don't like teamwork much. You can work in virtual lab with any pace you want and you don't have to ask anyone if you want to try out things. Of course you still can discuss your work with others but you're not dependent on them anymore. Virtual lab had some issues but most of them were either introduced by VMWare or by misuse. I had a chance to see how it evolves. And it improved a lot. I also had a chance to use VIRT and I can say that this lab works perfectly compared to VIRT. It doesn't have unnecessary complexity, it has pretty web-based interface. It feels pretty much like real lab but you can even see when traffic is going through the link. There are still things that can be improved (or more like features added) like one-click configs load, dynamic adding and removing of devices. But even now it's an amazing experience of reliable and sophisticated software.

Labs made a lot of troubles throughout the study, can't say that it was your mistake but some weird stuff, which were solved usually restarting, shut-down of interfaces. Usually took a while to troubleshoot. It would be nice to include some kind of config upload, reload, config erase, which would ease everything sooooo much

Virtual labs were excellent learning tools. On troubleshooting course there were quite a few problems but apart from that it was nice.

Loading new configurations is slow and tedious (troubleshooting networks, loading new ticket conf) since you have to do it for every device. Guacamole takes some time to learn all the little things (how to copy/paste), some kind of quick manual for it would be nice. From what I understood, you have to do some coding if you want to make your own topologies.

Cables could use something to show the way packets are going. If possible it would be cool if you could inject/load configurations automatically to the Cisco devices. If you decide to license this please use GNU General Public License for ethical reasons. And please continue to use non-proprietary software and consider using free and ethical routing and switching software.

Occasional bugs here and there: sometimes numpad does not work on consoles, changing window size inputs some weird letters and number to console... Copy&paste is a pain in the ass, but probably not under your control. Otherwise very stable, no reconnect issues for me.

Pieniä ongelmia mm. virtuaalikoneiden muisteissa ja disconnecteja lukuun ottamatta varsin hyvä toteutus. Keep it up!

Hyvin toimiva. Kaapeloinnin muokkaus olisi hyödyllinen ominaisuus.

great job with pentesting virtual sites

Joitain pieniä ongelmia oli. Ihan mukava oli configuroida myös virtuaaliympäristössä.

Satunnaisia disconnecteja lukuunottamatta virtuaalilabrat ovat toimineet hyvin, sen jälkeen kun niitä tottui käyttämään.

Topologian muokkauksen mahdollisuus itse lennosta olisi mainio lisä, mutta kaiken kaikkiaan hieno systeemi.

Copy paste things from other sources to the environment.

The virtual lab worked extremely well without any bugs. Any difficulties using the lab were the results of user errors. The GUI could have some visual improvements, but that's just for looks.