Performance Analysis of Network Emulator Based On The Use Of Resources In Virtual Laboratory

Yuri Ariyanto [1], Yan Watequlis Syaifudin [2], Budi Harijanto [3]
Information Technology Department
State Polytechnic of Malang
Malang, East Java, Indonesia
yuri.bjn@gmail.com [1], qulis@polinema.ac.id [2], budi_hijet@yahoo.com [3]

***Abstract*— A network emulator is a software in its environment mimicking the functions and habits of the original network. Problems often encountered in doing research on computer networks is a problem of design and trial scenarios that require software and hardware enough to implement such a network scenario similar to a real scenario. Using virtual machine technology is expected to solve the problem. Netkit supports experiments with various network technologies and can support other technologies necessary for certain network experiments. This study uses a network emulator with netkit, where it is freely available and built based on Linux User Mode. Netkit allows users to experiment with complex computer network scenarios that can be easily implemented using linux. The experimental research is done by making the design first. The design is used to implement the routing process, to connect 4 different network addresses. On the Netkit implementation of the design, it is implemented by creating a virtual routing lab based on data from computer network topologies. The virtual laboratory implementation of this research routing is based on a network topology scenario. The results of these virtual routing lab experiments, the routing process runs well on the network based on predetermined test scenarios, with the occurrence of connections between different network addresses..In the end the virtual routing lab implementation works well, with performance analysis results from the netkit emulator on the use of computer resources showing Usage CPU usage of 0% and Memory Usage averaging 20942.8k test time for 12 minutes. The results of this trial is very helpful for administrators before the implementation on the real network**

Keywords*— *User Mode Linux; Linux; Netkit; Routing.

I. Introduction

The emulation of networks is rapidly gaining the interest of network administrators, teachers and researchers in the networking area due to ease of setup, adherence to the behavior of real networks, and low cost [1]. With a network emulation environment, administrators can quickly set up testbeds to check that particular configurations work as expected before deploying them on production networks, teachers can let students configure their own network for practice or examination by using their PC, and researchers can validate theoretical models with practical experimentation in environments that behave very much as the real ones [1].

In this study will explain about Netkit, which is an open source network emulator [3]. Netkit supports experiments with various network technologies and can support other technologies necessary for certain network experiments. To test the netkit emulator implementation required a laboratory with the configuration and implementation of the network routing, which copied the network devices run on the netkit emulator, then analyzed the virtual lab performance of the netkit emulator implementation.

Some research has been done using the Netkit emulator. The reference explains that Netkit a lightweight emulator is built based on User-Mode Linux in addition to being an effective instrument to support teaching computer networks [2].

II. Literature Review

A. *Emulator*

An emulator is a software or hardware environment capable of reproducing functionality better than a real system. Emulators, especially if implemented in software, are very useful in conducting experiments. Particularly for computer networks, emulators are useful for testing network device configurations before they are implemented. [2].

B. *User Mode Linux (UML)*

User Mode Linux (UML) is a Linux virtual machine running on Linux. Technically, UML is a port connection from Linux to Linux. Linux has connected ports to many different processors, including X86, Sun's SPARC, IBM and Motorola's PowerPC, DEC's (Compaq and HP) Alpha and various other processors. [3]

UML has been widely used for network system administrators, network system developers and users. UML is different from other virtualization technologies because it is a Virtual Operating System (OS), but to call UML can be done virtually. UML technologies such as VMWare are truly virtual machines by copying physical platforms, from the CPU to the device, even though the running OS on the physical platform also runs on the emulator platform provided by VMWare. In VMWare any OS that runs on a platform can be booted under VMWare, otherwise UML can only be a guest OS on Linux [3].

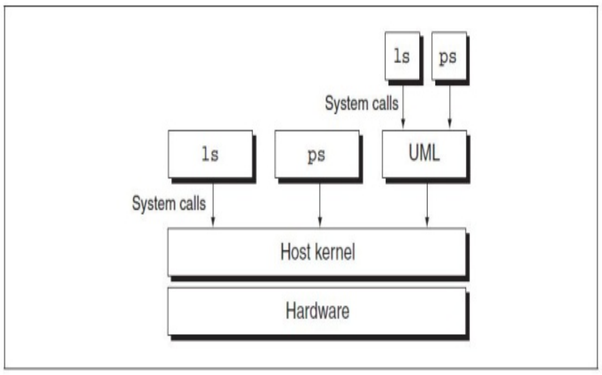


Figure 1. UML as Process and Kernel [3]

C. Netkit

Netkit is the place to set up and conduct network experiments at low cost and with little effort. The Netkit open-source network simulator was created by a group of university professors who use it as a tool in their teaching. It allows to create multiple virtual network devices (routers, switches, computers, etc.), which can be easily connected to form a network on one PC. The network equipment is virtual but has many original characteristics including the configuration interface [4].

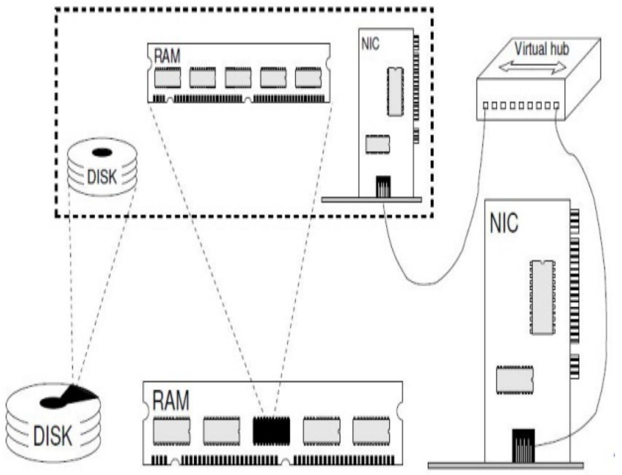


Figure 2. Netkit Virtual Machine Resources [5]

The approach adopted by Netkit is very simple, shown in Figure 2. Basically every device that forms the network is implemented inside Netkit as a virtual machine. Figure 2 shows how this mapping occurs. The virtual machine comes with a disk, where the raw image data file is stored on the computer disk. For memory can be set at startup and can be configured on a virtual network interface that is connected to the virtual hub.

Implementation of Netkit virtual machines in the network can be done by using a virtual hub, in practice a hub works as a kind of cable that connects multiple virtual machines. The virtual machine must always be connected to the virtual hub and can not directly connect to other virtual machines.

Figure 3 shows an example of a Netkit network emulator, where VM1, VM2 and VM3 are virtual machines and a simple topology consists of two collision domains (one included in VM1 and VM2 and the other entered in VM2 and VM3). In the VM2 topology, we can run a virtual hub connected to the virtual hub VM3 and on VM2 has two NICs. In its implementation, VM2 acts as a router that connects NICs in VM1 and NICs in VM3.

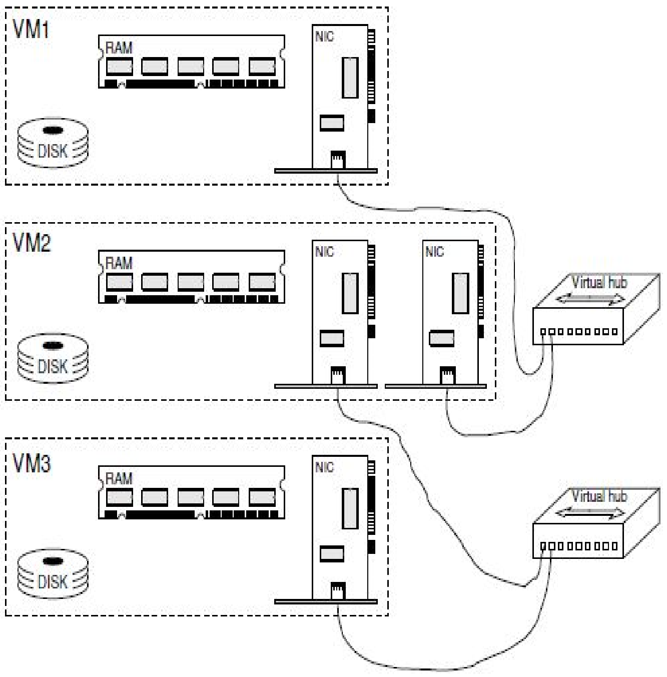


Figure 3. Sample of Netkit Network Emulator [5]

D. Routing

A router is one component of a computer network capable of passing data through a network or internet to its target, through a process known as routing. The routing process can be done by entering information of a network address manually into the routing table or with the help of routing protocol.

A router is capable of sending data / information from one network to another, the router is almost the same as the bridge, but the router is smarter than the bridge, because the router is able to connect two or more different networks, while the bridge is only able to connect the same network. Today, in the development of router devices have reached even beyond the limit of expected technological demands. The router will search for the best path to send a message based on the destination address and the original address.

Routers have a function to connect two or more different networks. Routers have a routing table that is used as a basis in path finding to the destination network of packets. If there is more than one path to the destination network, then the router will look for the best path according to the rules of "best path" it has, where the pathways are judged equally well [6].

E. Routing Internet Protocol (RIP)

Routing Information Protocol (RIP) is a protocol that utilizes the Bellman-Ford algorithm (distance-vector protocol group) in its best route selection. Compared with the OSPF protocol, the RIP protocol has a lower level of computational complexity, resulting in lower memory consumption. However, the consequence is that the use of RIP is limited to the medium to lower network with a relatively small number of hosts.

III. System Design

A. Hardware and Software Specification

At this stage will explain the needs of computer hardware specifications used to design and implement routing laboratories using netkit. Hardware and software requirements are as follows:

1. OS Debian 7
2. RAM 2 GB, Hardisk 320 GB
3. Processor Intel Dual Core
4. Emulator Netkit (Netkit-2.8.tar.bz2, Netkit-filesystem-i386-F.2.tar.bz2, Netkit-kernel-i386-K2.8.tar.bz2)

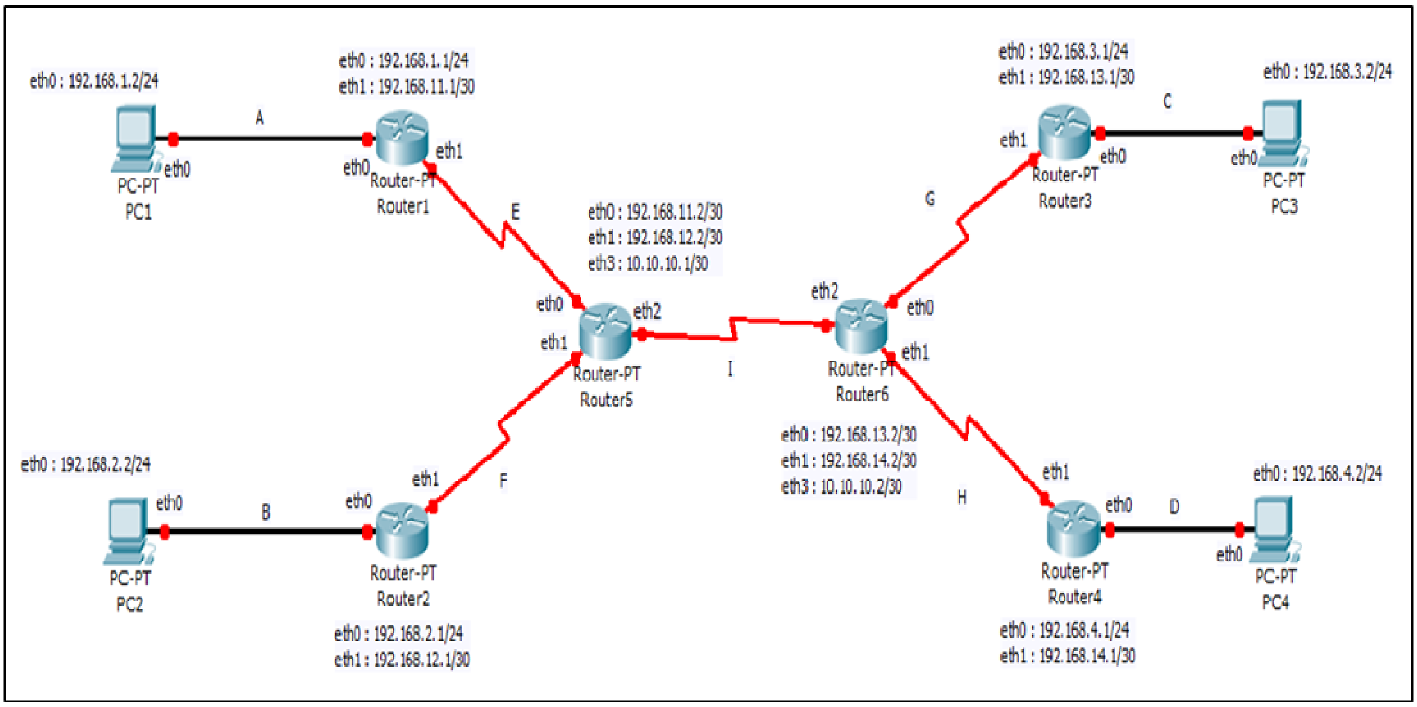


Figure 4. Routing Labs Topology

Please note that RIP does not adopt distance-vector protocols for granted, but by doing some additions to its algorithms so that routing can be minimized. Split horizon is used on RIP to minimize the effects of the hull (bouncing). To prevent counting to infinity cases, RIP uses the Triggered Update method. RIP has a timer to know when the router should re-provide routing information. If there is a change in the network, while the timer is not exhausted, the router must still send routing information as triggered by the change (triggered update). Thus, routers within the network can quickly figure out the changes that occur and minimize the possibility of routing loops occur.

B. Network Topology

For the experiments the design of laboratory network topology design is implemented in the netkit emulator, shown in Figure 4.

Description of the design of the network topology of the routing laboratory shown in Table 1.

Table 1. Node Details

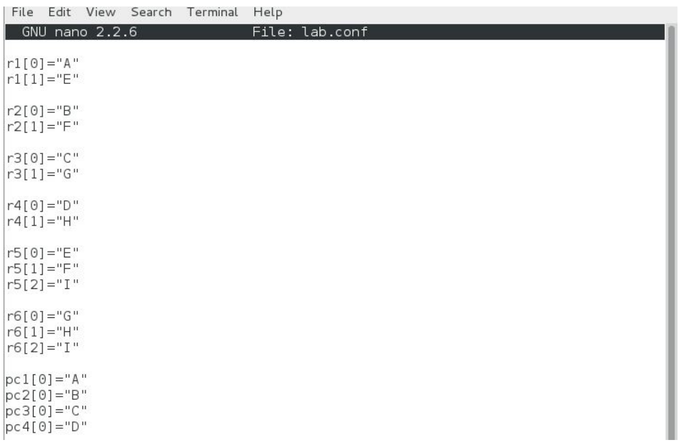
No	Node Name	Lan Card	Ip Address
1	PC 1	eth0	192.168.1.2/24
2	PC 2	eth0	192.168.2.2/24
3	PC 3	eth0	192.168.3.2/24
4	PC 4	eth0	192.168.4.2/24
5	Router 1	eth0	192.168.1.1./24
	Router 1	eth1	192.168.11.1/30
6	Router 2	eth0	192.1682.1./24
	Router 2	eth1	192.168.12.1/30
7	Router 3	eth0	192.168.3.1/24
	Router 3	eth1	192.168.13.1/30
8	Router 4	eth0	192.168.4.1/24
	Router 4	eth1	192.168.14.1/30
9	Router 5	eth0	192.168.11.2/30
	Router 5	eth1	192.168.12.2/30
	Router 5	eth1	10.10.10.1/30
10	Router 6	eth0	192.168.13.2/30
	Router 6	eth1	192.168.14.2/30
	Router 6	eth2	10.10.10.2/30

IV. Labs Scenario

At this stage, a virtual routing lab is created using a netkit emulator. The laboratory is implemented based on the network topology design in Figure 4. The testing scenario if laboratory development is shown in the following stages:

1. Create a lab.conf file

Created a lab.conf file to set the network scenario according to the routing lab topology design, in accordance with the configuration syntax of the Netkit emulator. The implementation of the lab.conf file is shown in Figure 5.



```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.2.6              File: lab.conf

r1[0]="A"
r1[1]="E"

r2[0]="B"
r2[1]="F"

r3[0]="C"
r3[1]="G"

r4[0]="D"
r4[1]="H"

r5[0]="E"
r5[1]="F"
r5[2]="I"

r6[0]="G"
r6[1]="H"
r6[2]="I"

pc1[0]="A"
pc2[0]="B"
pc3[0]="C"
pc4[0]="D"
```

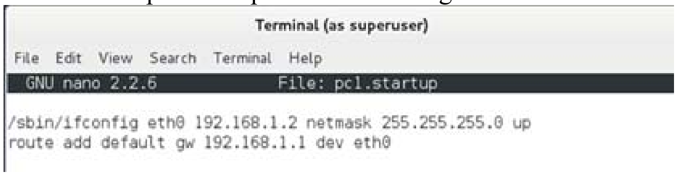
Figure 5. A screenshot of lab.conf file

Scenarios organized in the lab.conf file represent connections between NICs that are connected with virtual hubs on each device. If a NIC has the same letter as another NIC, then the two NICs are connected to the virtual hub, the lab.cont file is made up of six router computers and four client computers.

2. Configuring VM in lab.conf

At this stage configuration file is done .startup netkit. The file is used to run the configuration and service on the virtual machine at run time.

- Configuring pc1
  Create a pc1.startup file shown in Figure 6.

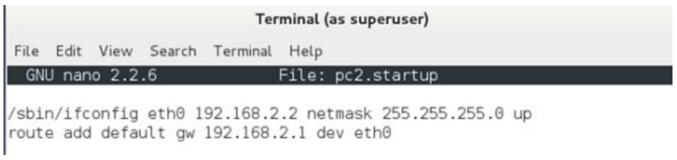


```
Terminal (as superuser)
File  Edit  View  Search  Terminal  Help
  GNU nano 2.2.6              File: pc1.startup

/sbin/ifconfig eth0 192.168.1.2 netmask 255.255.255.0 up
route add default gw 192.168.1.1 dev eth0
```

Figure 6. A screenshot of pc.1 startup

Figure 6 shows the file and configuration for the pc1 virtual machine with ip address 192.168.1.2/24 and adds ip gateway 192.168.1.1 so that the pc1 virtual machine to connect between virtual machines must use the ip gateway.

- Configuring pc2
  Create a pc2.startup file shown in Figure 7.

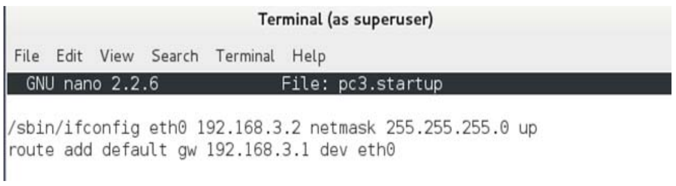


```
Terminal (as superuser)
File  Edit  View  Search  Terminal  Help
  GNU nano 2.2.6              File: pc2.startup

/sbin/ifconfig eth0 192.168.2.2 netmask 255.255.255.0 up
route add default gw 192.168.2.1 dev eth0
```

Figure 7. A screenshot of pc.2 startup

Figure 7 shows the file and configuration for the pc2 virtual machine with ip address 192.168.2.2/24 and adds ip gateway 192.168.2.1 so that the pc2 virtual machine to connect between virtual machines must use the ip gateway.

- Configuring pc3
  Create a pc3.startup file shown in Figure 8.

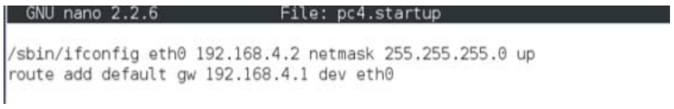


```
Terminal (as superuser)
File  Edit  View  Search  Terminal  Help
  GNU nano 2.2.6              File: pc3.startup

/sbin/ifconfig eth0 192.168.3.2 netmask 255.255.255.0 up
route add default gw 192.168.3.1 dev eth0
```

Figure 8. A screenshot of pc.3 startup

Figure 8 shows the file and configuration for the pc3 virtual machine with ip address 192.168.3.2/24 and adds ip gateway 192.168.3.1 so that the pc3 virtual machine to connect between virtual machines must use the ip gateway.

- Configuring pc4
  Create a pc4.startup file shown in Figure 9.

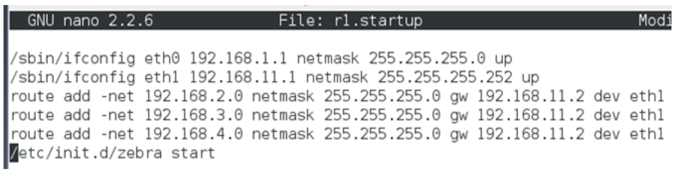


```
 GNU nano 2.2.6              File: pc4.startup

/sbin/ifconfig eth0 192.168.4.2 netmask 255.255.255.0 up
route add default gw 192.168.4.1 dev eth0
```

Figure 9. A screenshot of pc.4 startup

Figure 9 shows the file and configuration for the pc4 virtual machine with ip address 192.168.4.2/24 and adds ip gateway 192.168.4.1 so that the pc4 virtual machine to connect between virtual machines must use the ip gateway.

- Configuring r1
  Create a r1.startup file shown in Figure 10.

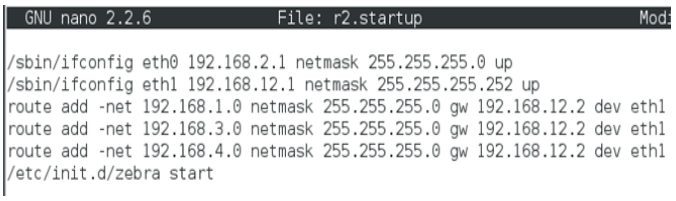


```
 GNU nano 2.2.6              File: r1.startup                        Modi

/sbin/ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
/sbin/ifconfig eth1 192.168.11.1 netmask 255.255.255.252 up
route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.11.2 dev eth1
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.11.2 dev eth1
route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.11.2 dev eth1
/etc/init.d/zebra start
```

Figure 10. A screenshot of r1 startup

Figure 10. Show file and configuration for virtual machine router r1 with ip address eth0 192.168.1.1/24 and eth1 192.168.11.1 and add network address 192.168.2.0, 192.168.3.0 and 192.168.3.0 with ip gateway 192.168.11.2 to connect between virtual machine With different network address. Service zebra is also enabled so that dynamic routing configuration with quagga software can be done.

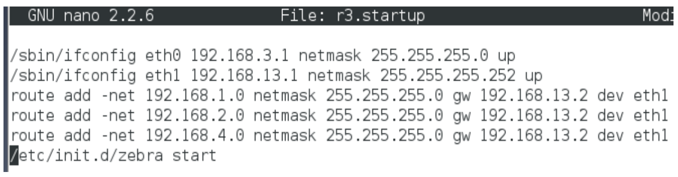
- Configuring r2
  Create a r2.startup file shown in Figure 11.



```
 GNU nano 2.2.6              File: r2.startup                        Mod:

/sbin/ifconfig eth0 192.168.2.1 netmask 255.255.255.0 up
/sbin/ifconfig eth1 192.168.12.1 netmask 255.255.255.252 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.12.2 dev eth1
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.12.2 dev eth1
route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.12.2 dev eth1
/etc/init.d/zebra start
```

Figure 11. A screenshot of r2 startup

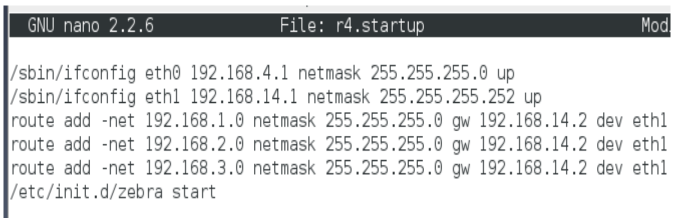
- Configuring r3
  Create a r3.startup file shown in Figure 12.



```
 GNU nano 2.2.6              File: r3.startup                        Mod:

/sbin/ifconfig eth0 192.168.3.1 netmask 255.255.255.0 up
/sbin/ifconfig eth1 192.168.13.1 netmask 255.255.255.252 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.13.2 dev eth1
route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.13.2 dev eth1
route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.13.2 dev eth1
/etc/init.d/zebra start
```

Figure 12. A screenshot of r3 startup

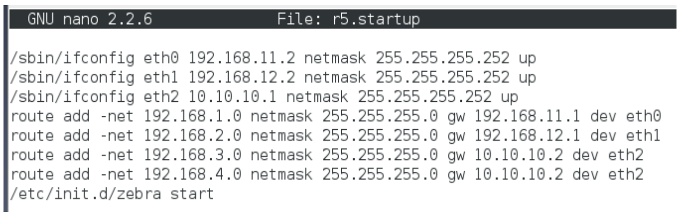
- Configuring r4
  Create a r4.startup file shown in Figure 13.



```
 GNU nano 2.2.6              File: r4.startup                        Mod

/sbin/ifconfig eth0 192.168.4.1 netmask 255.255.255.0 up
/sbin/ifconfig eth1 192.168.14.1 netmask 255.255.255.252 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.14.2 dev eth1
route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.14.2 dev eth1
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.14.2 dev eth1
/etc/init.d/zebra start
```

Figure 13. A screenshot of r4 startup

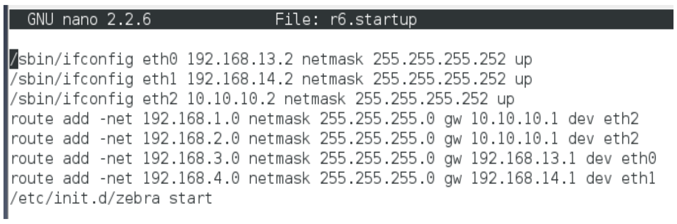
- Configuring r5
  Create a r5.startup file shown in Figure 14.



```
 GNU nano 2.2.6              File: r5.startup

/sbin/ifconfig eth0 192.168.11.2 netmask 255.255.255.252 up
/sbin/ifconfig eth1 192.168.12.2 netmask 255.255.255.252 up
/sbin/ifconfig eth2 10.10.10.1 netmask 255.255.255.252 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.11.1 dev eth0
route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.12.1 dev eth1
route add -net 192.168.3.0 netmask 255.255.255.0 gw 10.10.10.2 dev eth2
route add -net 192.168.4.0 netmask 255.255.255.0 gw 10.10.10.2 dev eth2
/etc/init.d/zebra start
```

Figure 14. A screenshot of r5 startup

- Configuring r6
  Create a r6.startup file shown in Figure 15.

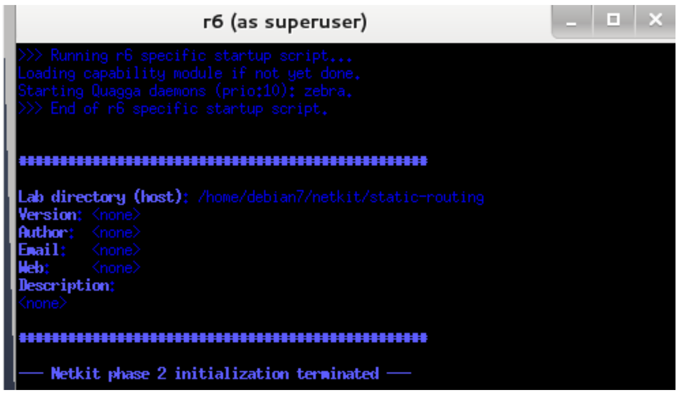


```
 GNU nano 2.2.6              File: r6.startup

/sbin/ifconfig eth0 192.168.13.2 netmask 255.255.255.252 up
/sbin/ifconfig eth1 192.168.14.2 netmask 255.255.255.252 up
/sbin/ifconfig eth2 10.10.10.2 netmask 255.255.255.252 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw 10.10.10.1 dev eth2
route add -net 192.168.2.0 netmask 255.255.255.0 gw 10.10.10.1 dev eth2
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.13.1 dev eth0
route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.14.1 dev eth1
/etc/init.d/zebra start
```

Figure 15. A screenshot of r6 startup

V. LABS IMPLEMENTATION

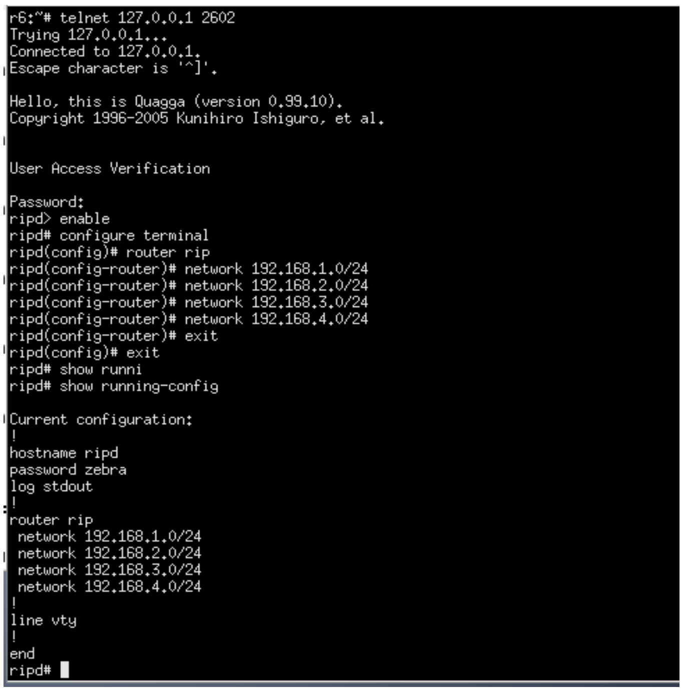
The following lab routing display in the netkit emulator is shown in figure 16.



```
>>> Running r6 specific startup script...
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra.
>>> End of r6 specific startup script.


##################################################

Lab directory (host): /home/debian7/netkit/static-routing
Version: <none>
Author:  <none>
Email:   <none>
Web:     <none>
Description:
<none>

##################################################

--- Netkit phase 2 initialization terminated ---
```

Figure 16. Virtual Routing Labs Emulator

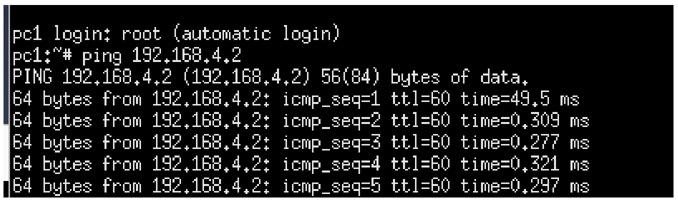
The quagga configuration for the dynamic router ripd on router r6, shown in figure 17.



```
r6:~# telnet 127.0.0.1 2602
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.10).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
ripd> enable
ripd# configure terminal
ripd(config)# router rip
ripd(config-router)# network 192.168.1.0/24
ripd(config-router)# network 192.168.2.0/24
ripd(config-router)# network 192.168.3.0/24
ripd(config-router)# network 192.168.4.0/24
ripd(config-router)# exit
ripd(config)# exit
ripd# show runni
ripd# show running-config

Current configuration:
!
hostname ripd
password zebra
log stdout
!
router rip
 network 192.168.1.0/24
 network 192.168.2.0/24
 network 192.168.3.0/24
 network 192.168.4.0/24
!
line vty
!
end
ripd# █
```

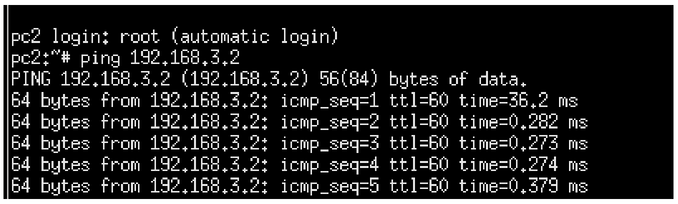
Figure 17. Ripd Quagga Configuration in r6

Figure 17 shows the configuration on router r6 that implements ripd dynamic routing using quagga applications. The routing test by pinging the pc1 to pc4 virtual machine is shown in figure 18 and pc2 to pc3 shown in figure 19.



```
pc1 login: root (automatic login)
pc1:~# ping 192.168.4.2
PING 192.168.4.2 (192.168.4.2) 56(84) bytes of data.
64 bytes from 192.168.4.2: icmp_seq=1 ttl=60 time=49.5 ms
64 bytes from 192.168.4.2: icmp_seq=2 ttl=60 time=0.309 ms
64 bytes from 192.168.4.2: icmp_seq=3 ttl=60 time=0.277 ms
64 bytes from 192.168.4.2: icmp_seq=4 ttl=60 time=0.321 ms
64 bytes from 192.168.4.2: icmp_seq=5 ttl=60 time=0.297 ms
```

Figure 18. ping from pc1 to pc4



```
pc2 login: root (automatic login)
pc2:~# ping 192.168.3.2
PING 192.168.3.2 (192.168.3.2) 56(84) bytes of data.
64 bytes from 192.168.3.2: icmp_seq=1 ttl=60 time=36.2 ms
64 bytes from 192.168.3.2: icmp_seq=2 ttl=60 time=0.282 ms
64 bytes from 192.168.3.2: icmp_seq=3 ttl=60 time=0.273 ms
64 bytes from 192.168.3.2: icmp_seq=4 ttl=60 time=0.274 ms
64 bytes from 192.168.3.2: icmp_seq=5 ttl=60 time=0.379 ms
```

Figure 19. ping from pc2 to pc3

Figures 18 and 19 show the connection test of the routing implementation successfully done with the ping command from the pc1 virtual machine to pc4 and the pc2 to pc3 vitality machine is connected.

VI. PERFORMANCE ANALYSIS

In this section we will analyze the performance of netkit emulator based on the use of computer resources. The analysis is done by looking at the virtual machine performance of the virtual routing lab implementation using the #top command in linux.

Table 2. Performance Statistics

VM	Time (minute)	App. Running	App. Sleeping	CPU Usage	Memory Usage
Pc1	12	1	21	0%	19780k
Pc2	12	1	21	0%	19828k
Pc3	12	1	21	0%	19784k
Pc4	12	1	21	0%	19784k
R1	12	1	22	0%	21708k
R2	12	1	22	0%	21696k
R3	12	1	22	0%	21700k
R4	12	1	22	0%	21704k
R5	12	1	22	0%	21708k
R6	12	1	22	0%	21736k

The test results from the successful connection of computer clients with different network addresses using the #ping command, shown in table 3.

Table 3. Connection Test with Ping Command

VM	Ip Address	VM	Ip Address	TTL	Time	Status
Pc1	192.168.1.2/24	Pc4	192.168.4.2/24	60	0.297 ms	connected
Pc2	192.168.2.2/24	Pc3	192.168.3.2/24	60	0.379 ms	conneted

VII. Conclusions

From the test results that have been passed can be concluded:

1. Using Netkit emulators can be created to create a laboratory based virtualization technology to teach the network concept.
2. The hardware requirement to make a netkit laboratory is not so high as to be able to run a virtualization based technology lab.
3. The use of computer hardware resources when the netkit laboratory runs is not too high indicated on CPU Usage 0% usage and Memory Usage averaging 20942.8k

References

[1] Pizzonia, Maurizio; Romandini, Massimo. *Netkit: Easy Emulation of Complex Networks on Inexpensive Hardware*. 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom 2008) Mar 18th, 2008.

[2] Rimondisi, Massimo. *Emulation of Computer Networks with Netkit*. Technical Report RT-DIA-113-2007, Roma Tre University, Jan 2007.

[3] Dike, Jeff. *User Mode Linux*. Prentice Hall, Apr 2006.

[4] University of Roma Tre Computer Networks Research Group. *Netkit*. http://www.netkit.org.

[5] Rimondini, Massimo. *Emulation of Computer Networks with Netkit*. Technical Report RT-DIA-113-2007, Roma Tre University, Jan 2007.