Assignment 3

s316620

4018NSA

**PARTI: DHCP and DNS in Virtual Lab**

1. Start up the network that was created in the previous assignments.

2. Exploring the config files of your Quagga router configuration.

a. First configuring the adapter 1, connect it to Internal network “GW\_external” and adapter 2 to “GW\_SW1\_cable”.

When I set up the quagga interface in Assignment two I removed the ip address configuration line seen below from the “/opt/bootlocal.sh” file.

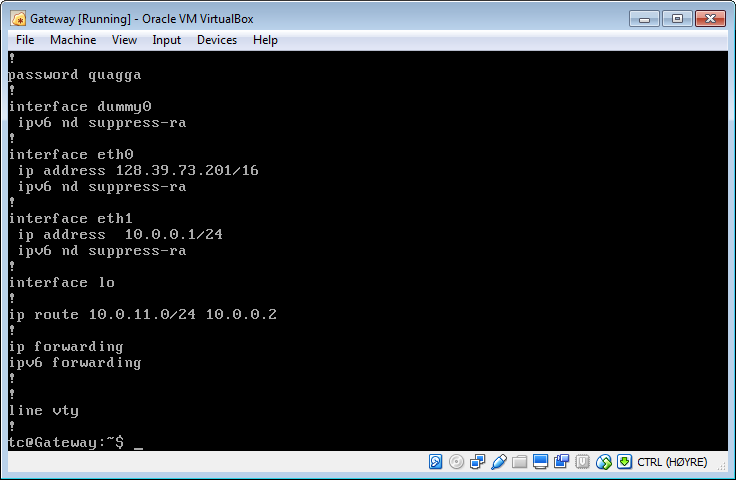
“*ifconfig eth0 10.0.0.1 netmask 255.255.255.0”*

b. After changing the internal networks the adapters are connected to, I change the ip address assigned to each interface. This can be done using multiple ways. We can either configure it through the quagga terminal or edit the “*/usr/local/etc/quagga/zebra.conf*” file.

Use one of the following method :

1. Editting the *zebra.conf*  file.

Open the file “*/usr/local/etc/quagga/zebra.conf*” and the edit the IP address to the new IP addresses for each of the interface.



1. Using the quagga terminal :

Use the commands:

*sudo vtysh // Open the quagga interface*

*config t // Configure*

*interface eth0 // Open interface eth0*

*no ip address 10.0.0.1/24 // Remove ip address from the interface*

*ip address 128.39.73.201/16 // Add new ip address to the interface*

*exit*

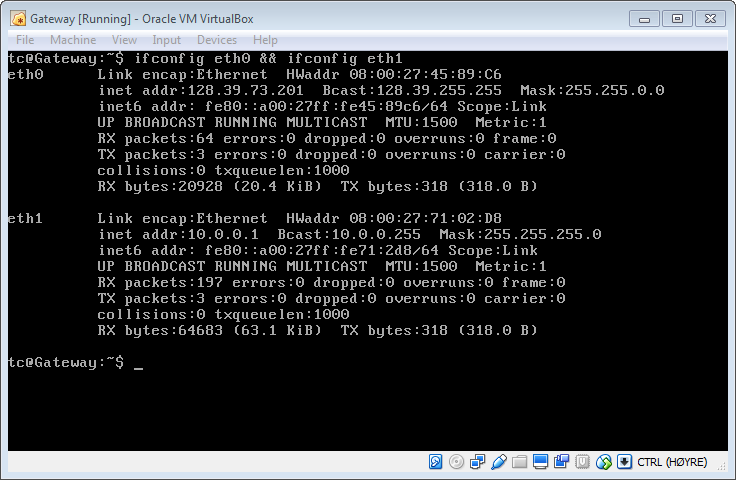
*exit*

*write // Write the changes to /usr/local/etc/quagga/zebra.conf*

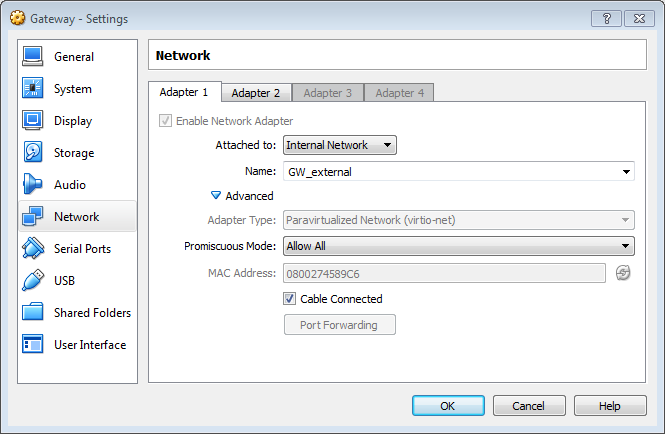
*exit*

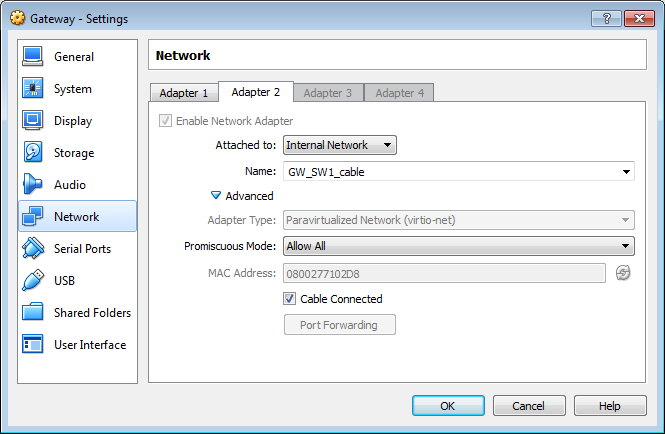
*filetool.sh –b // Save persistently*

Do the same for interface eth1 as well, and then reboot. Then we can see that the new IP addresses are assigned to the interfaces.

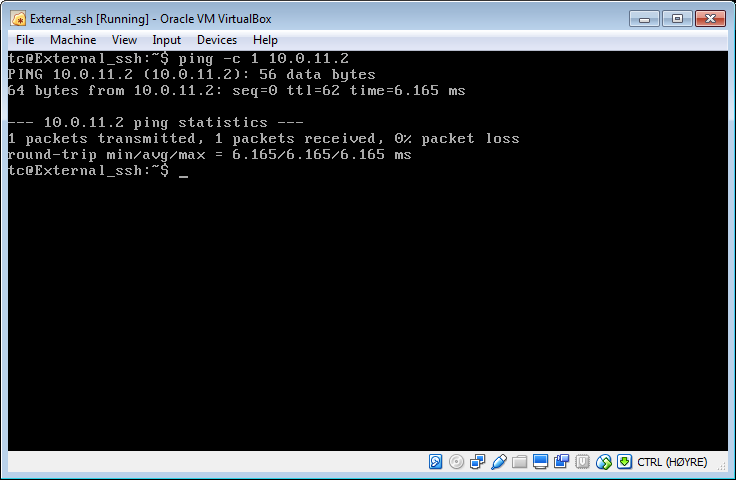


c. Making sure that Adapter 1 (eth0) is connected to “GW\_external” and Adapter 2 (eth1) is connected to “GW\_SW1\_cable”.





Then checking connectivity with ping from External\_ssh to Host 1 and it shows that it is connected.

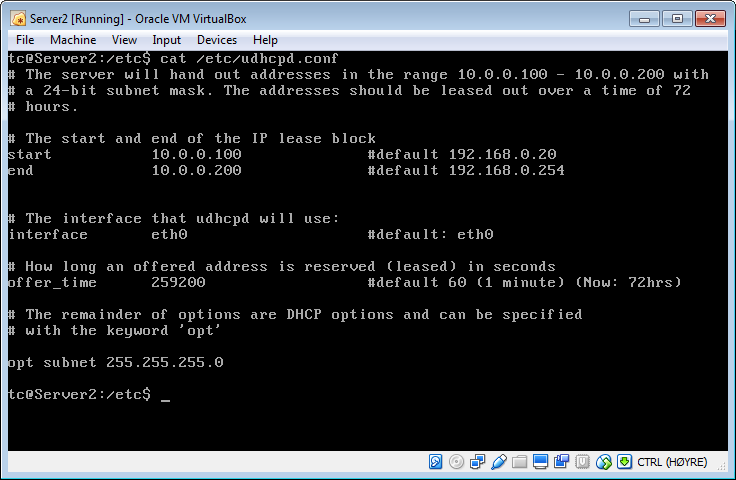


1. A minimal DHCP server/daemon (udhcpd)
2. Now I am going to install a virtual udhcp-server in the network. However, if there is a udhcp server running in the network, the udhcpc process would quickly obtain a new address from it and would override the manual IP address configuration I set up. So I am turning off the udhcp client processes at the nodes “Gateway”, “Choke” and “Server2” ( Where I will set up the udhcp-server).

I turn off the processes in each of the nodes by adding the command “*pkill udhcp”* in the *bootlocal.sh* file, before the assignment of IP addresses (*ifconfig* lines). [ *Note: Remember to save persistently ( filetool.sh –b) all the changes that is made.*]

1. Creating a persistent file *“/etc/udhcpd.conf”* to Server2, for the configuration of DCHP-server.

We get a default “*udhcpd.conf”* file from the internet (<https://udhcp.busybox.net/udhcpd.conf>). And use the format to configure the *udhcpd.conf* file in */etc/* .

My *udhcpd.conf* file looks like : 

1. Now starting the udhcpd-daemon at Server2.

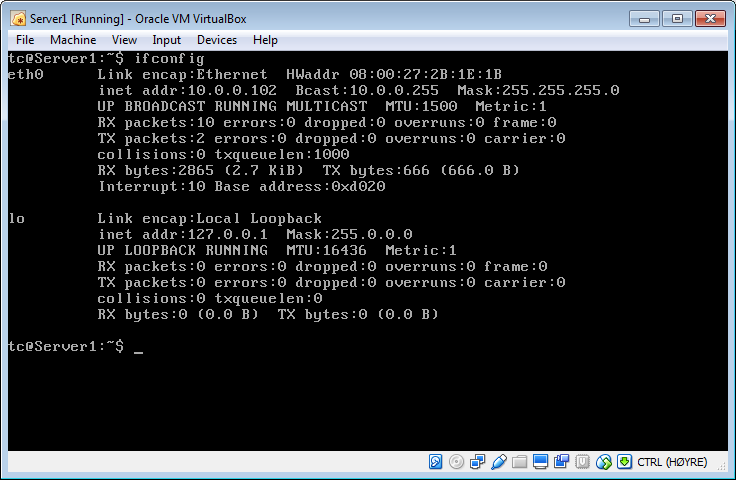
I use the command “ *sudo udhcpd /etc/udhcpd.conf &”*

Then restart “Gateway”, “Choke” and “Server1”.

Gateway has the IP address 10.0.0.1 on eth0 and 128.39.73.201 on eth1.

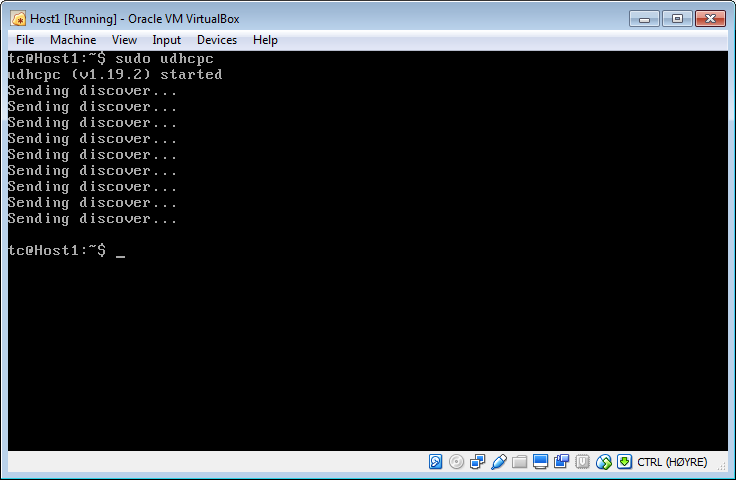
Choke has the IP address 10.0.0.2 on eth0 and 10.0.11.1 on eth1.

Whereas “Server1” has the IP address 10.0.0.102 on eth0.



The reason “Server1” gets an IP address in the range 10.0.0.100 – 10.0.0.200 as configured in “Server2” (DHCP-server) is because when “Server1” reboots the udhcp process runs in Server1 and gets the IP address from “Server2”.

1. Running “*sudo udhcpc”* at Host1, to check what is happening.



Host1 does not discover any IP address, here Host1 is the dhcp-client and “Server2” is the dhcp-server. There is no DHCP-relay agent between the server and the client so the client does not get configured.

1. DHCP-Relay

The purpose of the DHCP- relay is to pass packets containing information, from DHCP-clients to the DHCP-server and vice-versa. The client requests are forwarded to the server and then the server can configure the IP-address and lease it to the clients. The DHCP relay agent is relied upon when the DHCP-server and DHCP-client are not on the same subnet. In the assignment above, “Host1” (DHCP-client) is on a different subnet than “Server1” (DHCP-server). Without a DHCP-relay agent, which would be “Choke” in this case, the “Host1” is not assigned an IP address. In “Choke” the *udhcp* process has been terminated so this is the region that “Host1” does not have an IP.

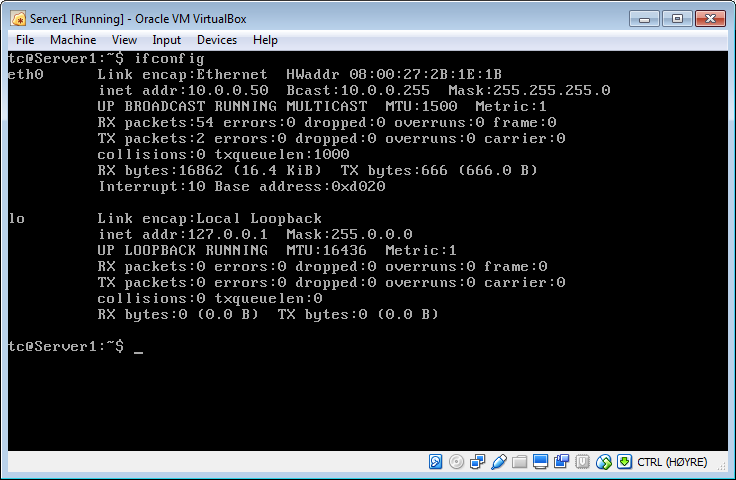
1. To assign a static IP address to “Server1” we will run the *ifconfig eth0* command on “Server1”, to get the HW address of “Server1”. Then add the line to the *udhcpd.conf* file in “Server2” :

*static\_lease <Server1 MAC address>*

Where <Server1 MAC address> is the actual MAC address of “Server1”. The MAC address of the interface *eth0* is referred to as “HWaddr”.

After adding the ‘*static\_lease’* to the udhcpd.conf file save persistently.

Now to check run the *udhcpd* process with the new *udhcpd.conf* file and then restart “Server1”.



[*NOTE: The “sudo udhcpd /etc/udhcpd.conf &” command is not added to bootlocal.sh as instructed in the assignment. Here we are just checking if what we did worked.*]

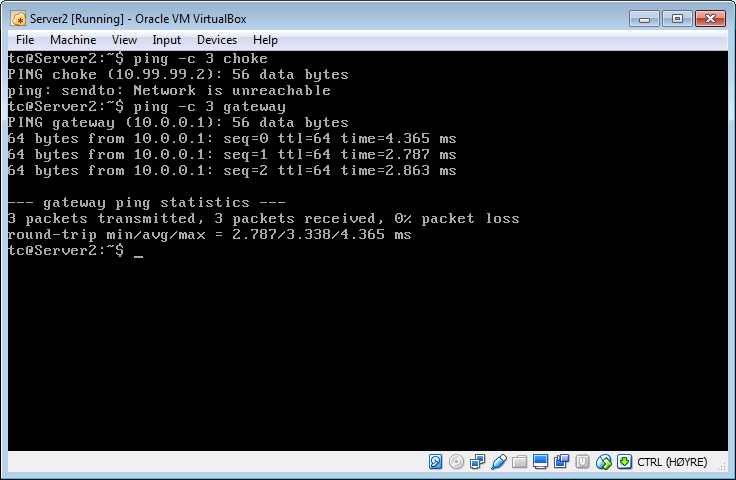
1. The /etc/hosts file.
2. At Server2, issuing the following command “*ping gateway”*. We get the error message :

*ping : bad address ‘gateway’*

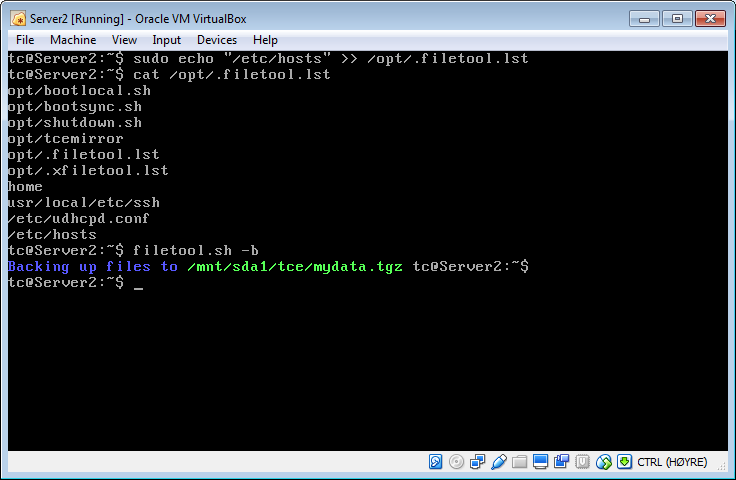
1. Updating the */etc/hosts* – file at Server2, by adding the two lines :
2. *10.0.0.1 gateway*
3. *10.99.99.2 choke (*We intentionally have set an incorrect address for testing purpose)

Now we issue the commands “*ping choke”* and “*ping gateway”* on Server2.

It can be seen that Server2 cannot “*ping”* Choke, an error message is returned as seen in the image. But now it is able to ping gateway by issuing the command “*ping gateway”*. This is because we assigned the correct IP address of Gateway but not of Choke in the *“/etc/hosts”-*file.

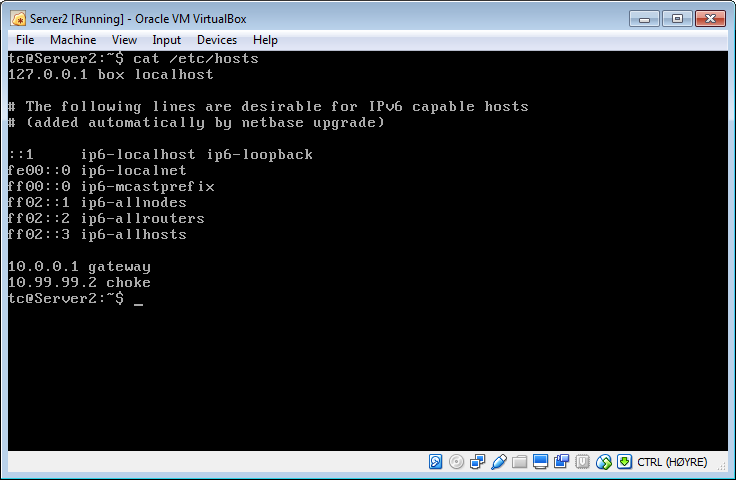


1. The change in the *“/etc/hosts”* file is made persistent by adding the file name to the “*/opt/.filetool.lst”*. Then running the command *“filetool.sh –b”*, which saves the changes in the *hosts* file persistently.



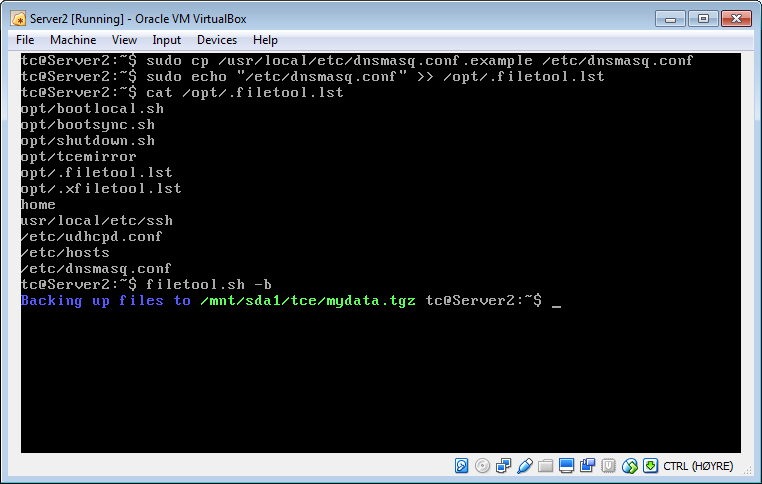
1. The line we added in the above assignment is overwritten by a process after reboot, but before *bootlocal.sh* is run. To fix this problem, I add the line “*echo 10.0.0.1 gateway >> /etc/hosts”*, *“ echo 10.99.99.2 choke >> /etc/hosts”* in the *bootlocal.sh* – file on Server2.

Then save the changes persistently and reboot the host. Check that the Gateway can be pinged from Server2 with the command “*ping gateway”* after reboot, and that the */etc/hosts* – file is OK.



1. Using dnsmasq for local DNS
2. Installing dnsmasq (Using the command *“ab dnsmasq.tcz”*) at Server 2. To accomplish this, Server2 has to be connected to the internet. The adapter of Server2 should be connected to NAT, and the hosts should be started. We have the “*pkill udhcp”* command running in the *bootlocal.sh* so we have to start the *udhcpc*  process first. This can be done by issuing the command “*sudo udhcpc”*. Then when an internet connection is established download and install “dnsmasq.tcz” packet using *ab* command.

Then copy the configuration file example from “*/usr/local/etc/dnsmasq.conf.example”* to *“etc/dnsmasq.conf”*, then save it persistently.



1. Now after the installation of the dnsmasq and then having a default configuration file at “*/etc/dnsmasq.conf”*. We configure the following things in *dnsmasq.conf*.
2. Enable “domain-needed” and “bogus-priv” ( probably line 14 – 16)
3. Set “dhcp-range =10.0.0.3, 10.0.0.100, 12h” (line 136)
4. Set “dhcp-leasefile=/tmp/dnsmasq.lease” (line 413)
5. Set “dhcp-option=option:router 10.0.0.1” (line 250)
6. Configure static routers(gateway and choke) by adding two line in the conf-script (line 195)

* “dhcp-host=<Gateway HWaddr (eth1)>,ignore”
* “dhcp-host=<Choke HWaddr (eth0)>,ignore”

Then start the dnsmasq in default mode by running the command :

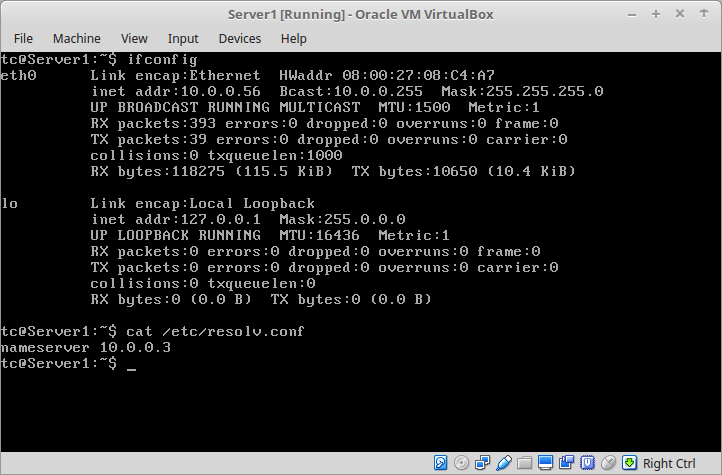
*Sudo dnsmasq –d*

Then to save the command persistently we added the command to *bootlocal.sh. Using the command :*

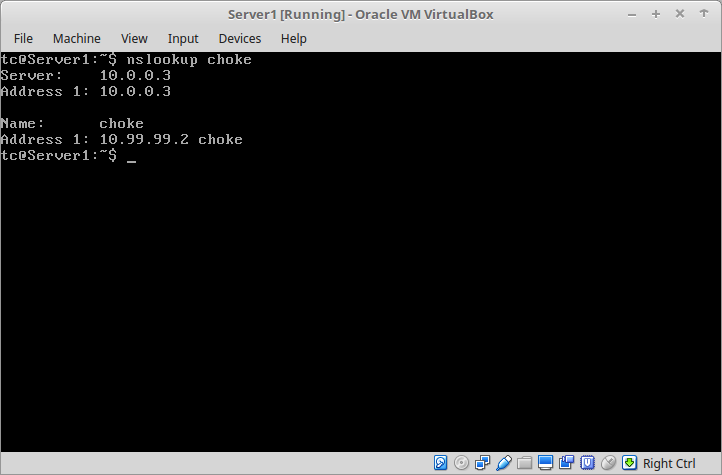
*sudo echo “dnsmasq” >> /opt/bootlocal.sh*

and then save persistently using *filetool.sh*. This runs the dnsmasq after every reboot of the host.

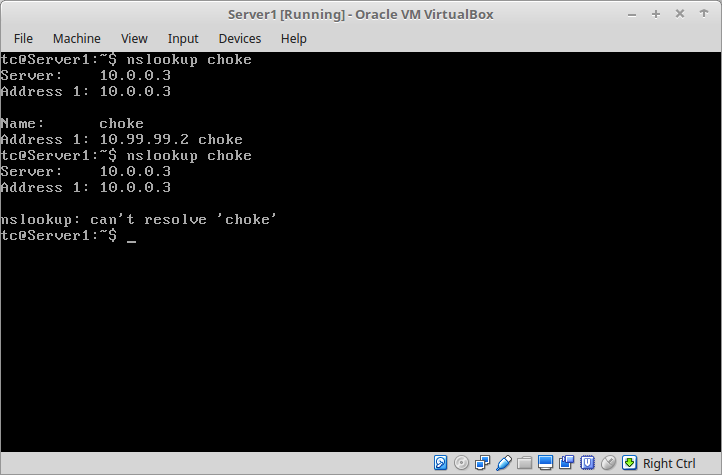
c. Running *udhcpc* at Server1 we get the IP address 10.0.0.56 .The local dns-server has the IP address 10.0.0.3



d. Now when running the command *nslookup choke* from Server1. It can be observed that the IP address 10.99.99.2 ip associated with Choke.



The nameserver (Server2) has the IP address 10.99.99.2 associated with the name choke in the *“/etc/hosts”* file. As Server2 (10.0.0.3) is the nameserver it looks up its hosts names and assigns the name to the network.

e. If I turn off the dnsmasq at Server2 and then run nslookup we get the message “can’t resolve ‘choke’”. This is because Server1 has Server2 as the nameserver

where it gets its IP address from, but as the dnsmasq is down Server2 does not provide any IP addresses. Thus we get the above error message.

f. Manually configuring the Gateway and Host to use Server2 as a local dns server, by adding the line :

“*echo nameserver 10.0.0.3 >> /etc/resolv.con”*

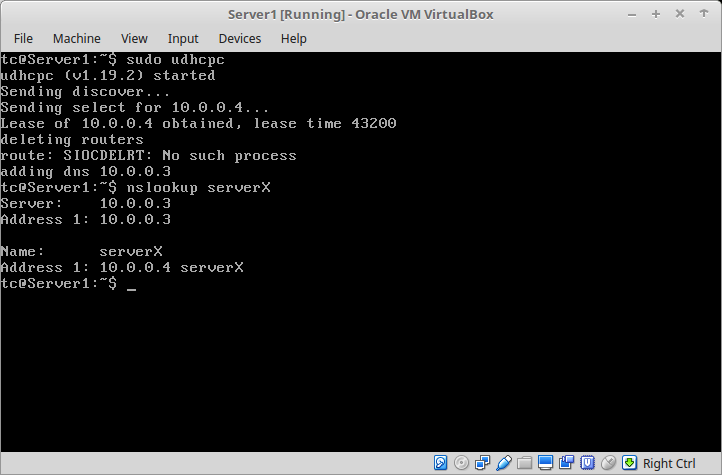
in the *bootlocal.sh* file at Gateway, Choke and Host1.

g. Now, configuring Server1 so that it gets a fixed address by adding the line:

“*dhcp-host=<Server1 MAC address>,serverX,10.0.0.4,12h” (line 164)*

where the <Server1 MAC address> is the actual Hwaddr of the adapter (eth0) at Server1. This line is added in the “*/etc/dnsmasq.conf”* -file at Server2.

Then running *udhcpc* at server1.



Running the udhcpc command and nslookup from Server1 (show above) and Server2 works.

h. Now configuring dnsmasq for all the nodes on the subnet 10.0.0.0/24.

The *dnsmasq.conf* file is edited and Server1 is assigned 10.0.0.4 by changing the line to :

“*dhcp-host=<Server1 MAC address>,server1,10.0.0.4,12h” (line 164)*

The correct IP address is used for choke in */etc/hosts* at Server2 and Server1, Server2, External\_ssh and Host1 is also added to the file.

6. Obtaining Internet access from the virtual network

a. Setting up eth0 of Gateway to connect to the internet. This is done by :

I. Powering off gateway

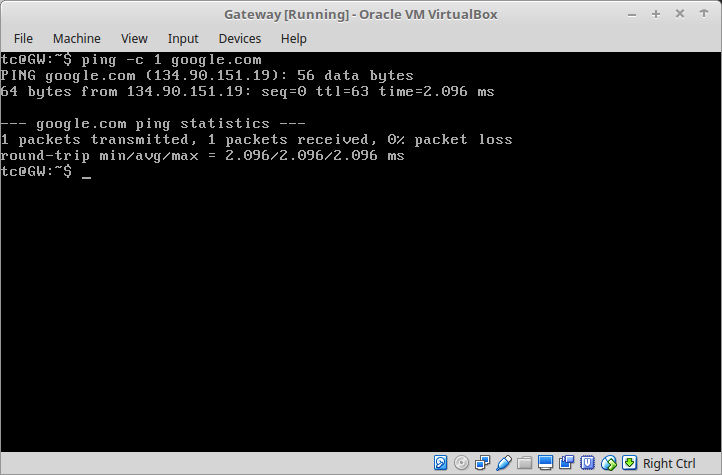
ii. Assign an adapter (Adapter 3) configured to a NAT interface of VirtualBox.

Iii. Disable the Adapter1. Now the VirtualBox assigns Adapter3 as eth0.

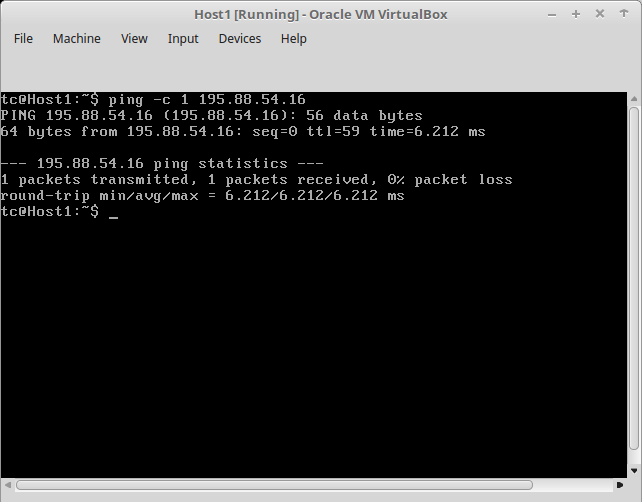
The line :

*sysctl -w net.ipv4.conf.eth0.proxy\_arp=1*

is added to the *bootlocal.sh* file. Then we *ping* google.com from Gateway and check that we have internet connection. As seen below, it was sucessful.

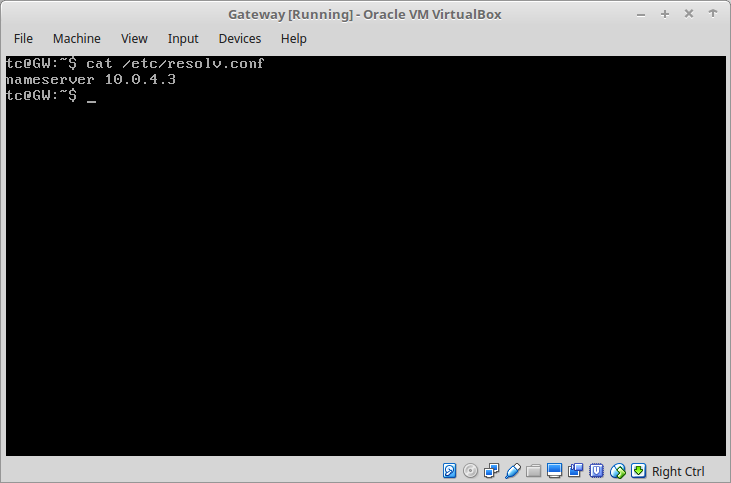


b. Connecting to the internet from Host1. I ping the IP address of *vg.no* (195.88.54.16). Host1 can ping the IP address but not the domain name as the dns server is Server2(10.0.0.3) which does not have the internet nameserver.



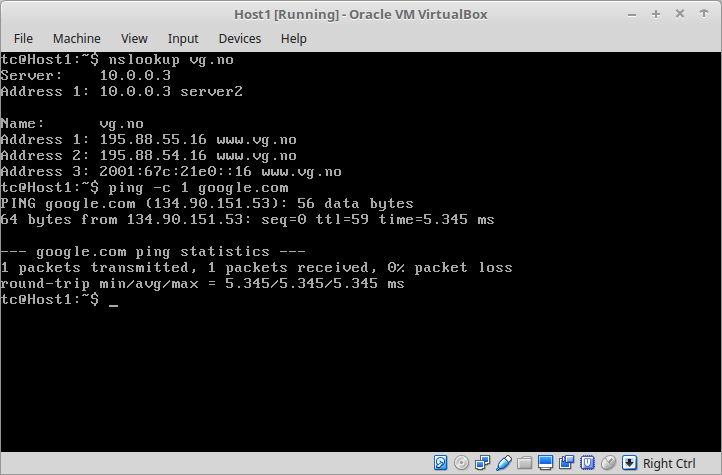
c. Now looking for the nameserver at *Gateway* in the *“/etc/resolv.conf”* file.

The address was 10.0.4.3 .



Now adding the line : *echo* «nameserver 10.0.4.3» >> /etc/resolv.conf

in the *bootlocal.sh* file at Server2. This provides Server2 with an external nameserver that dnsmasq can read. We use *nslookup*  to check for *vg.no* ,Then pinging from Host1 to google.com we can see that it has internet connection as well.

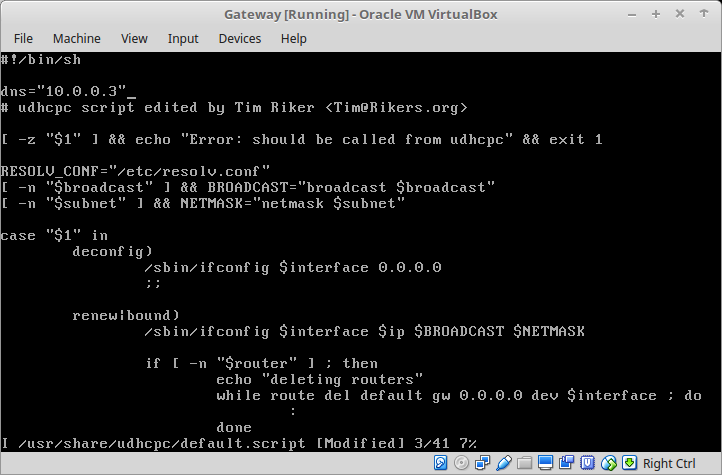


d.

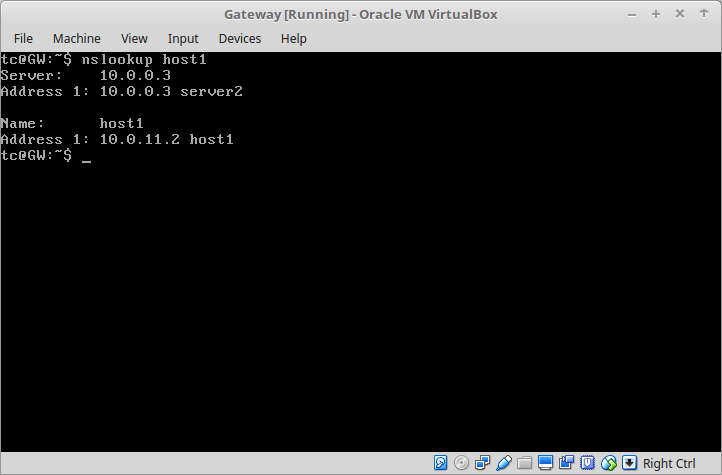
According to content of */etc/resolv.conf* of Gateway we can not look for local names (like Host1). To configure this I added the line:

*echo nameserver 10.0.0.3 >> /etc/resolv.conf*

to the *bootlocal.sh* file at the Gateway. This worked on my personal computer but did not work in the computers at the university. So I configured the *«/usr/share/udhcpc/default.script»* file. I added the line *«dns=10.0.0.3»* to the start of the file as seen below.



Then after saving persistently and then rebooting, I had connection to the nameserver at Server2(10.0.0.3).



7. Documenting the final dnsmasq configuration.

a. For proper the dnsmasq configuration the settings we have enabled are:

i. domain-needed

Tells the dnsmasq never to give plain names (without dots or domain parts) to upstream nameservers. A «not found» is returned if the name in not known from */etc/hosts.*

ii. bogus-priv

It reverse lookups the private Ips which are not found in /etc/hosts, «no such domain» is answered to the DHCP lease file if not found.

iii. Dhcp-range

This is the range in which the IP addresses will be handed out. The format is <start-address>,<end-address>,<lease time>. The lease time is the amount of time the IP will be linked to the host.

iv. dhcp-host

This leases an IP address to the host with the given Hwaddress for the specified amount of time.

v. dhcp-option with router

This tells the hosts which gateway to use when a hosts requests an IP address.

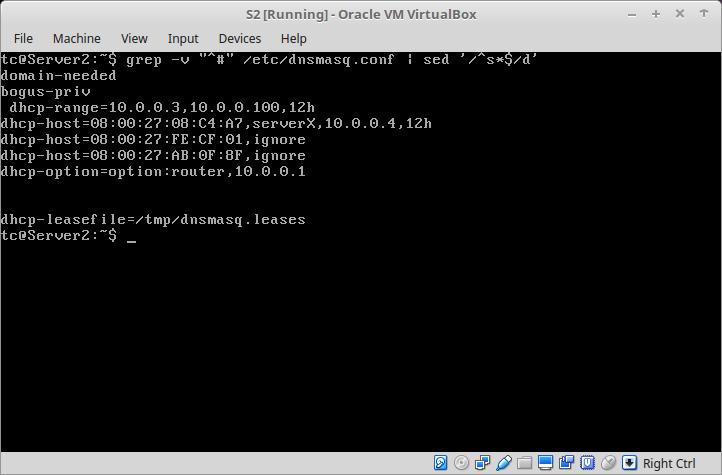
vi. dhcp-leasefile

This is where the DHCP lease information is stored.

This information about the dnsmasq was gathered from the dnsmasq-man page[[1]](#footnote-2).

To see the *dnsmasq.conf* file I use the command:

*grep -v “^#” /etc/dnsmasq.conf | sed ‘/^s\*$/d’*



Here we can see the *dnsmasq.conf* file with our configurations.

**Part II : Using TCPDump to capture TCP connections**

a. Checking that we have connectivity between Host1 and Server2. First we ping Server2 from Host1, and make sure there is connection (which there was). Then the *openssh* server is started on both Server2 and Host1, if not already running. This can be done by issuing the following command at both the hosts;

*“sudo /usr/local/etc/init.d/openssh start”*

I have already set up the passwords for the hosts, if this is not done, it can be done with the command : *“sudo passwd tc”* where *tc* is the username.

Then I have also installed *tcpdump* at choke from before using the “*ab* *tcpdump”* command.

b. Now we run the TCPdump to capture traffic in interface which is connected to the Host1, that is eth1 at Choke. The command that is used is :

*“sudo tcpdump -i eth1 -w ssh.dump not port 67”*

This command is run at Choke. This captures all the traffic at the adapter *eth1* at Choke and stores it in the file *ssh.dump*. The *not port 67* tells the tcpdump not to capture traffic on port 67.

c. Running the command “*ssh 10.0.0.3”* from Host1, connects Host1 to Server2 (10.0.0.3). The TCPDump at choke captures the SSH TCP connection.

d. Now it is possible to simulate the packet loss in this TCP connection by shutting down the adapter at Server2. This is done with the command “*sudo ifconfig eth0 down”* at Server2.

e.Now we can see the tcpdump file where we saved all the packets that were captured. When the *ssh* connection between Host1 and Server2 is established, TCP uses a 3 way handshake:

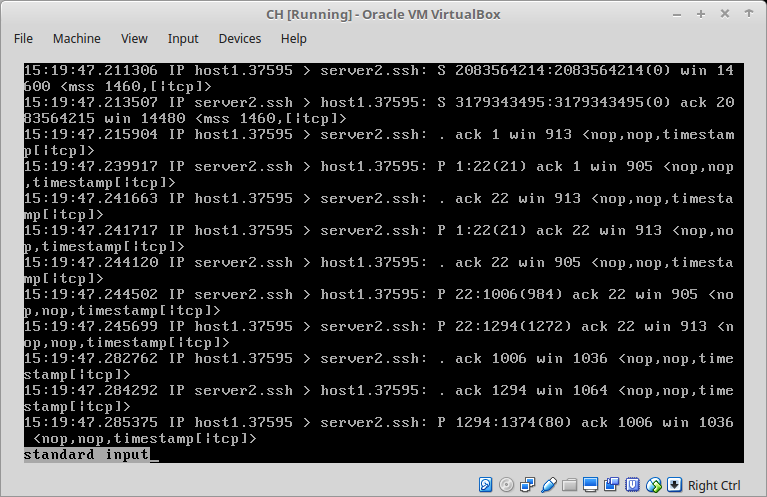
I. Host1 sends SYN

ii. Server2 responds with SYN,ACK

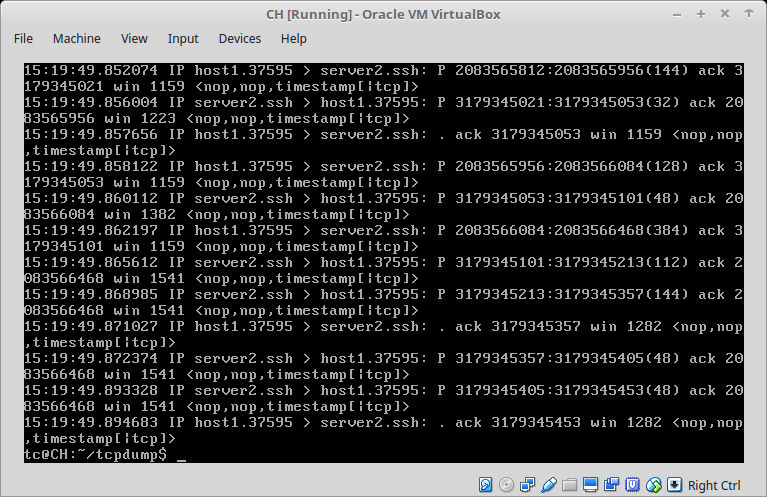
iii. Host1 sends ACK

We can see this in the following image as well. I have used the following line to filter the packages scene:

*tcpdump -r ssh.dump “tcp[tcpflags] & (tcp-syn|tcp-ack) != 0*

**

When the adapter (eth0) at Server2 is down ( *ifconfig eth0 down)* the Host1 freezes, the tcpdump captures a packet where Host1 sends a packet but Server2 does not respond.



Reference

I. <http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html>; September 7, 2016

ii. <http://www.tcpdump.org/tcpdump_man.html>; September 7, 2016

iii.

1. http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html [↑](#footnote-ref-2)