NT21 Assignment 6

DHCP/NAT/DNS

Introduction

In this 6th assignment you’re going to exercise DHCP, NAT and DNS protocols.

In most home LAN networks the IP addresses of all devices connected to this network are configured by using the DHCP protocol. Normally, the router which connects the home network to the Internet has also built-in functionality of the DHCP server that distributes IP addresses.

The other common feature of the home routers is to provide NAT functionality to map the private IP addresses of the home network to the public IP addresses.

Finally, we’re going to have a short look to DNS, an indispensable protocol in current networking that provides directory services for IP addresses to host names mappings.

Task 1: Unpack and study Netkit Lab

Copy provided nat-dhcp-ng.zip file in your Linux environment.

Unzip this file. You’ll see a directory structure used to start up a Netkit lab.

There are following important files in a Netkit lab:

1. *lab.conf* – configuration of the network. This file defines which nodes are to be started and how they are interconnected through collision domains (in our case LANs)
2. *<node>.Startup* files – here you can define any commands that you wish to be executed before the node is start-up. These files are already preconfigured for this lab.
3. *<node>* directory files. Each node has its own directory, which you can use to pass information from the host Ubuntu system to the Netkit node to be started. Here you can put for example network configuration files.

*Example:* Imagine you want to configure network interfaces for the node *PC1.* On an Ubuntu system you can do this in the */etc/network/interfaces* file. In the Netkit lab environment you can put the corresponding *interfaces* file in the *<netkit\_lab\_dir>/PC1/etc/network*. This file will be then used to configure the network interfaces during the automatic startup of *PC1* node.

For this week’s assignment a small network is being created in the lab:

1. *LANA* **:** *PC1*, *PC2* and *Router* represent a home network and are connected to *LANA*. They use private IP addresses from the range 192.168.1.x. The *Router* uses interface 192.168.1.254.
2. *LANB* : *Router* which is connected to *LANA* is also connected to *LANB*. *LANB* represents an external network to which the node *Server* is connected. The IP addresses for *LANB* are statically preconfigured in the lab.

Task 2: Configure DHCP and startup Netkit lab

Do the following steps:

As explained in task 1, we can pass configuration files from the lab directory structure to the Netkit node. To startup DHCP server on the *Router* node, a *dnsmasq.conf* file is used.

This file can be found in *<netkit\_lab>/Router/etc* directory.

Have a look at this file, and answer the following questions:

* describe the meaning of the *dhcp-range*
* which option(s) are used in this file.

**Dit is de range waarin een ip kan worden gebruikt voor de setup.**

**192.168.1.100,192.168.1.199,4**

One of the settings in this file is the range of the DHCP IP addresses that the server will distribute. Change this range to your own values which are different from the preconfigured values.

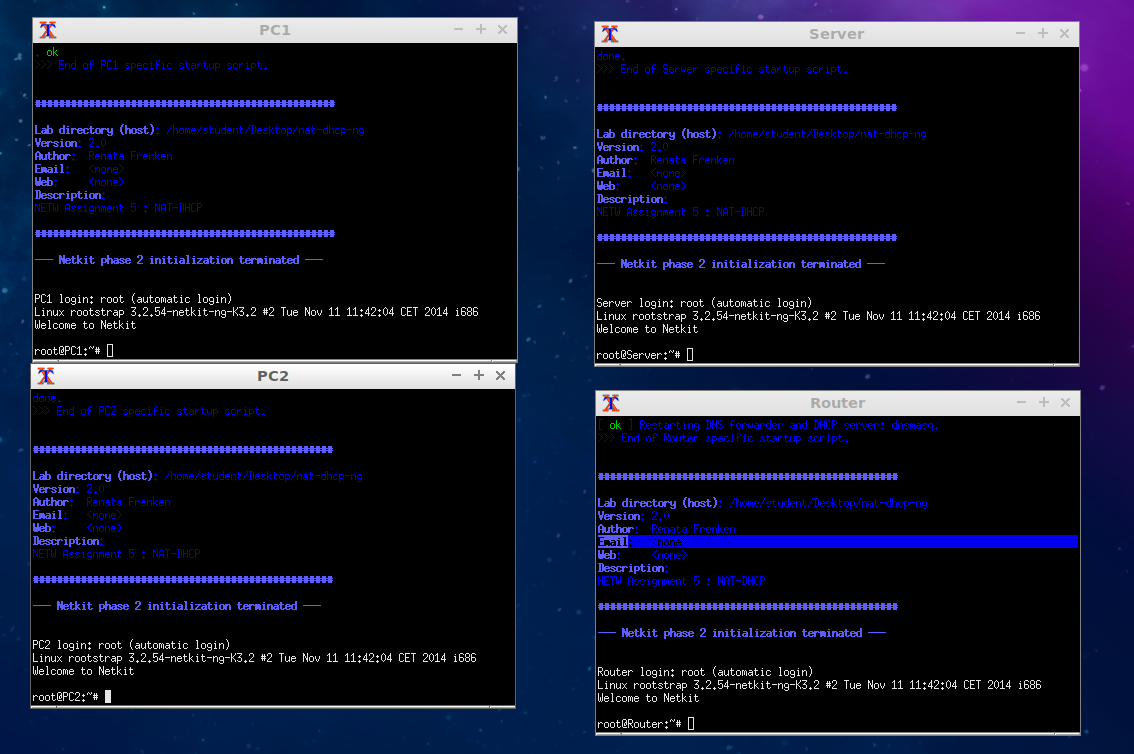
Startup the Netkit lab. To startup the Netkit lab you issue the following command in the lab root directory:

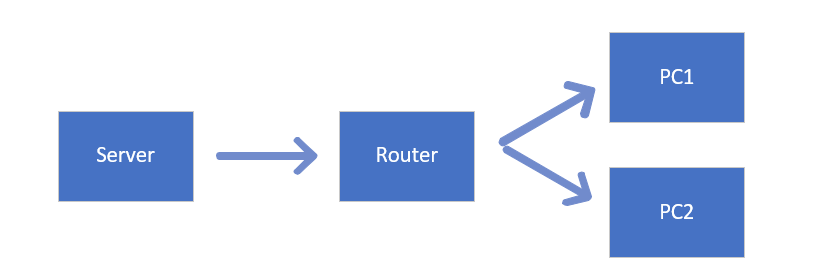
*lstart –f*

Option –f means that all nodes are started in parallel, so not one after another. It is important to do this for this exercise, as the nodes will need the DHCP server from the router to be able to startup.

Once your network is startup, inspect the configuration of your IP interfaces. The ranges defined in *dnsmasq.cnf* file should be used for *PC1* and *PC2*.

Create a screenshot from the PC1 and PC2 interface configuration and make your network drawing.





To stop a lab issue command:

*lcrash*

This will thoroughly clean up your Netkit environment.

Task 3: Add a new node to your network and follow the DHCP’s DORA process

In a home network, when a new device is added to your network, it will use DHCP protocol to acquire the IP configuration from the DHCP server. We’re going to simulate this situation in the following exercise.

Before we start a new node we are going to start *tcpdump* trace at the *Router* node, so that we can later analyze the DORA process (used to obtain IP address) of the newly added node in Wireshark.

To start the trace on the Router, issue the following command:

*tcpdump –w <filename>.pcap –s 0*

From now on, the tcpdump is going to redirect all its output to your <filename>.pcap file. The ‘–s 0’ option is there to get the entire packets in your trace.

Now that the sniffing on the *Router* is switched on, we can start a new node that is going to acquire an IP address from the DHCP server on the *Router*.

Start a Netkit node connected to **LANA** (don’t make the mistake of using a network name called **A**, use **LANA**). When the node is started up it has no IP configuration yet. To acquire the configuration from the DHCP server, issue the following command on this node:

*dhclient eth0*

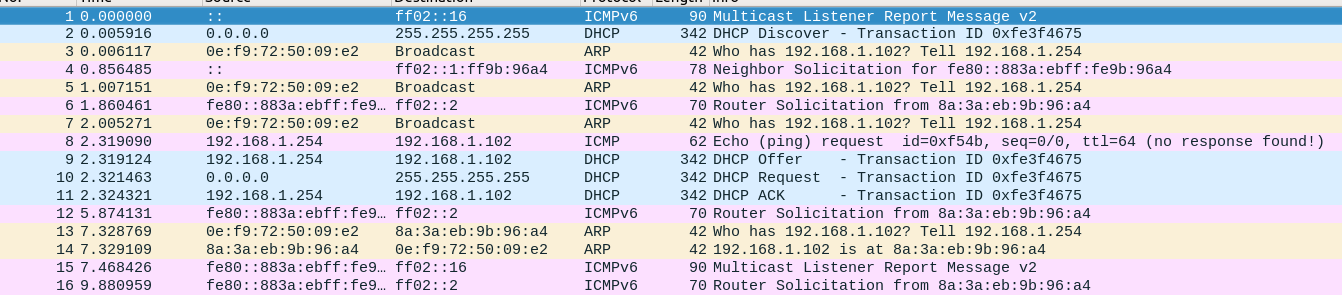
This should start DHCP DORA process on your eth0 interface. Once you acquired the IP configuration (check this first), you can stop *tcpdump* on the *Router*.

To be able to read your pcap file generated by tcpdump on the *Router* node, you have to copy this file in the Router’s */hosthome* directory. This will make this file available in your Ubuntu home directory so that you can open it with Wireshark.

Open the pcap file in Wireshark and find the trace of the DHCP DORA process.

Provide a screenshot of the Wireshark trace where the DORA process can be seen.

Provide a network drawing of your network.

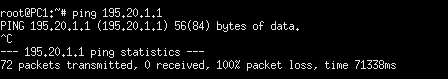


Task 4: Configure NAT table on the Router and experiment with ping

We are now going to look closer at the NAT protocol. To configure NAT protocol, we’ll use Linux *iptables* command functionality. *Iptables* is usually used to set up firewall on Linux system but can also be used to set up NAT table.

For this exercise we’ll reuse the network and networking drawing created in the previous tasks.

Go to *PC1* and try to ping your *Server* node. Is the ping successful? Explain why yes or no.



**Ping was not successful. Packets worden niet ontvangen.**

Now, go to your *Router* node and issue the following command:

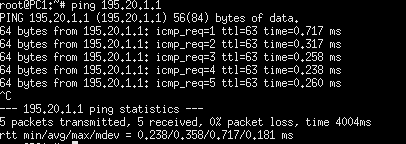
iptables –t nat –A POSTROUTING –o eth1 –j MASQUERADE

This command created a POSTROUTING entry of a MASQUERADE type in the NAT table (if you’re interested in detailed specification of this command, you can read the manual pages of *iptables*). This means that all packets after they have been routed by the *Router* will replace their source IP address (from the private network) by the IP address of the *Router*’s outgoing interface. The *Router*’s outgoing interface to *Server* is *eth1*.

Once you’ve added the above entry in the iptables NAT table, you can check the table by issuing the following command:

*iptables –t nat –vL*

After adding this entry, go to *PC1* again and try to ping your *Server* node. Is the ping successful? Explain why yes or no.



**Ja, want de router stuurt het nu door naar de server.**

# Go to your *Server* and ping *PC1*. Is the ping successful? Explain why yes or no.

# Network is unreachable nee. De server weet niet waar hij naartoe moet.

# In Task 3 you’ve learned how to create a trace with *tcpdump*. Redo your ping from node *PC1* to *Server* while the trace on the *Router* is set by using *tcpdump*.

# *Tcpdump* is by default using your first network interface (mostly *eth0*). To see the change of source IP address you have to specify the right network interfaces with tcpdump command.

# To use another interface with tcpdump issue:

# *tcpdump –i <interface>*

# or to use all interfaces issue:

# *tcpdump –i any.*

# *Tcpdump –i* any is preferable, as by issuing this command you can see both traffic coming to the router and leaving the *Router*. This makes it easy to see the change of IP address.

# Analyze your trace in Wireshark. The source address of the ping packet going from *PC1* to *Server* should be changed.

# Give a proof of this by providing Wireshark screenshot of the packet.

# 

# Het is geen wireshark, maar je ziet wel de verandering

# When you pinged a node from *PC1* to *Server*, NAT changed your source address. Ping answers the ping request by a ping reply from Server to *PC1*. The destination address of this ping will be the address of the router where NAT translation is done.

# What do you think: how does ping (ICMP) know to which internal node it should send the answer?

# ICMP gaat de nodes af welke node een response geeft is de goede.

Task 5: NAT on TCP connections

This last task is similar to task 4 but instead of looking at NAT by using ping (ICMP) we will now use TCP connections.

A simple way to establish a TCP connection is using *netcat* command (abbreviated to nc) as explained in the previous assignment. You can start this command either in a ‘server’ mode when the node is waiting (listening) for a connection or in a ‘client’ mode when the node attempts to create TCP connection. To check whether our NAT configuration is working, we’re going to establish TCP connection between *PC1* and *Server*.

Based on the knowledge from the previous assignment, use nc command to establish a connection between *PC1* (client) and *Server* (server).

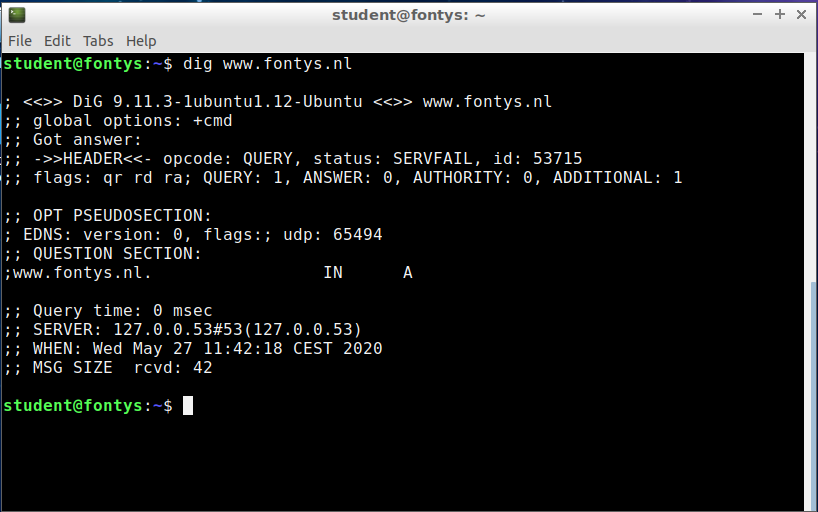
Again, make a Wireshark trace of TCP connection establishment and data exchange and provide a Wireshark screenshot of a packet where we can see that the IP address of the outgoing packet was changed by NAT.

What do you think: When we establish a TCP connection from *PC1* to *Server* and send data through this connection, how does the *Router* know to which of the two PCs (*PC1* and *PC2*) should it communicate on the way back from *Server* to *PC1*?

Task 6: DNS

Choose your own web address (e.g. [www.fontys.nl](http://www.fontys.nl)) and use *dig* command from the Linux command line to find out its IP address.

What is the output of this dig command? Can you explain what you see in the output of the dig command?



**Zelf weet ik hier niet veel van, maar wat ik hieruit kan opmaken is dat je de**

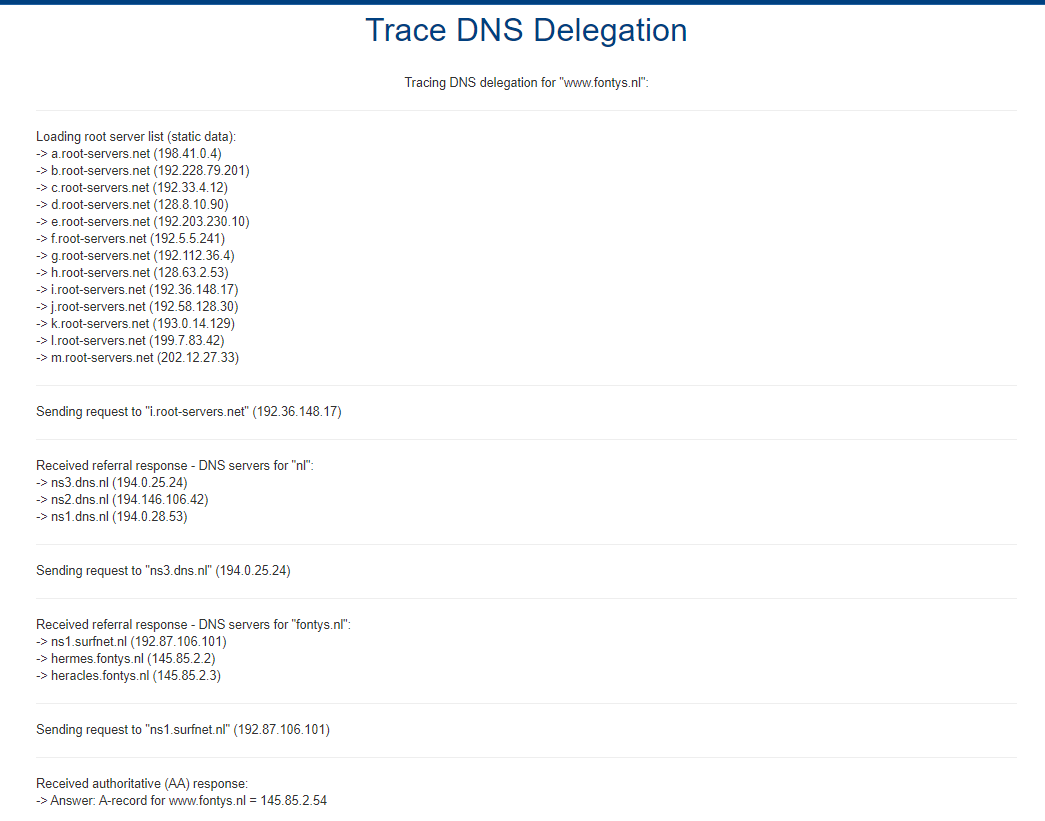
**header van de website opvraagt samen met de server waar hij vandaan komt.**

What IP Address belongs to your chosen host?

**127.0.0.53**

Try also DNS delegation for your chosen host. Go to <http://simpledns.com/lookup-dg> page and find out how the DNS resolution is being done.

Provide a screenshot of the output. Which root server is used?



Gebruikte root server staat er niet bij.

Task 7 Canvas Quiz:

Fill-in Canvas quiz for week 6.

Task 8 (Optional): Port Forwarding

Tasks 4 and 5 dealt with working of NAT. These tasks were about client being a part of the private network and server having an external address.

In this task we’re going to simulate a situation, where server is being run on a private network and it is being accessed by a client from the outside world – an external address (as an example imagine you want to run your own web or game server in your private home network and make it available on the Internet). To achieve this, port forwarding has to be added.

To simulate port forwarding in the Netkit lab of this assignment, we are going to use a web server that is started on *PC1* node. The available web server is *apache2* in /etc/init.d/ directory.

To check whether this web server is correctly started you can issue the following command:

/etc/init.d/apache2 status.

The default apache2 setup offers a test html page located in /var/www/index.html.

Use *iptables* command to specify port forwarding in such a way, that we are able to access web server on internal network (e.g. *PC1*).

*TIP : use Internet to find out how to do this, e.g. this page :* [*https://www.systutorials.com/816/port-forwarding-using-iptables/*](https://www.systutorials.com/816/port-forwarding-using-iptables/)

*You don’t have to specify router forwarding, this is already done in Netkit lab.*

*One command should be enough to specify required port forwarding.*

Provide here the port forwarding command that you used.

Now we are going to test a connection from the *Server* node to the web server on *PC1* node. For this, we’re going to use a very simple “links” browser located in the root directory of Server.

Start the *links* browser from your *Server* node and connect it to your web server on *PC1* node (use “g” for entering URL/IP address). You should now be able to see the default page from the *apache2* webserver on *PC1.*

Which IP address have you specified in the links browser?

Provide a screenshot of what you see in links after successful connection to the *PC1’s* webserver