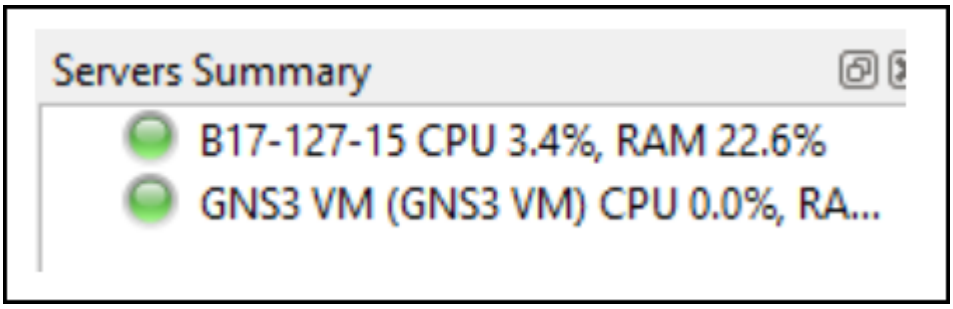


Figure 3 – GNS3 Working Environment is ready to go

NOTE: This can take some time depending on the hardware being used. It will be gray before it turns green. If the VM does not start automatically, restart GNS3 and click ***Help*** > ***Setup Wizard*** to reestablish the link to VirtualBox as outlined in Chapter 2. Do not manually start GNS3 VM before launching GNS3.

4. Connect the OpenWrt router to your simulated ISP

    4.1. Complete Chapter 4

    4.2. On the left side of the GNS3 environment, you can see various tools. Click on ***Browse all devices*** as shown here 

    4.3. A sub-menu will appear. Click the picture of a cloud with the word NAT next to it and drag it to the workspace

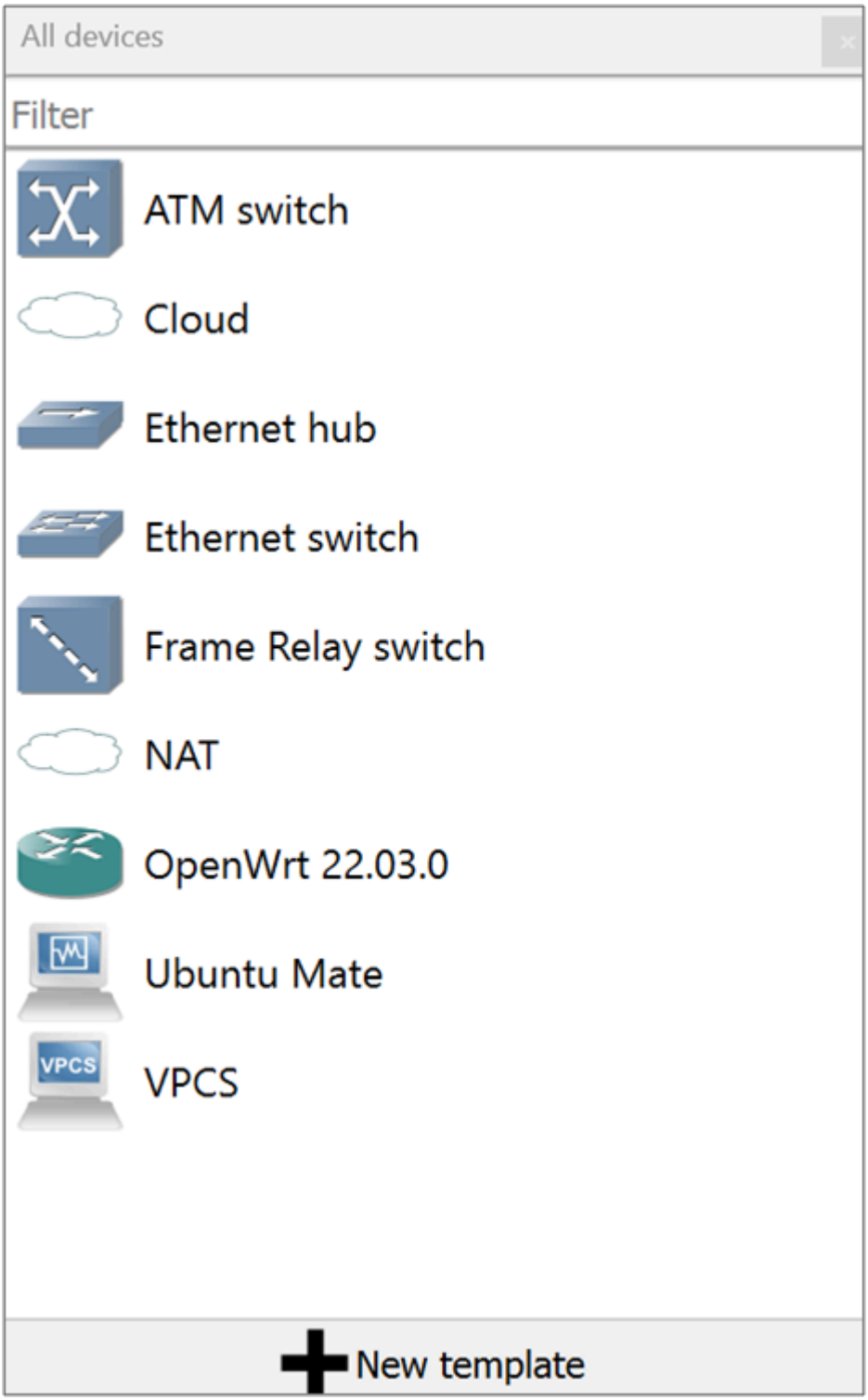


Figure 4 – Device icons

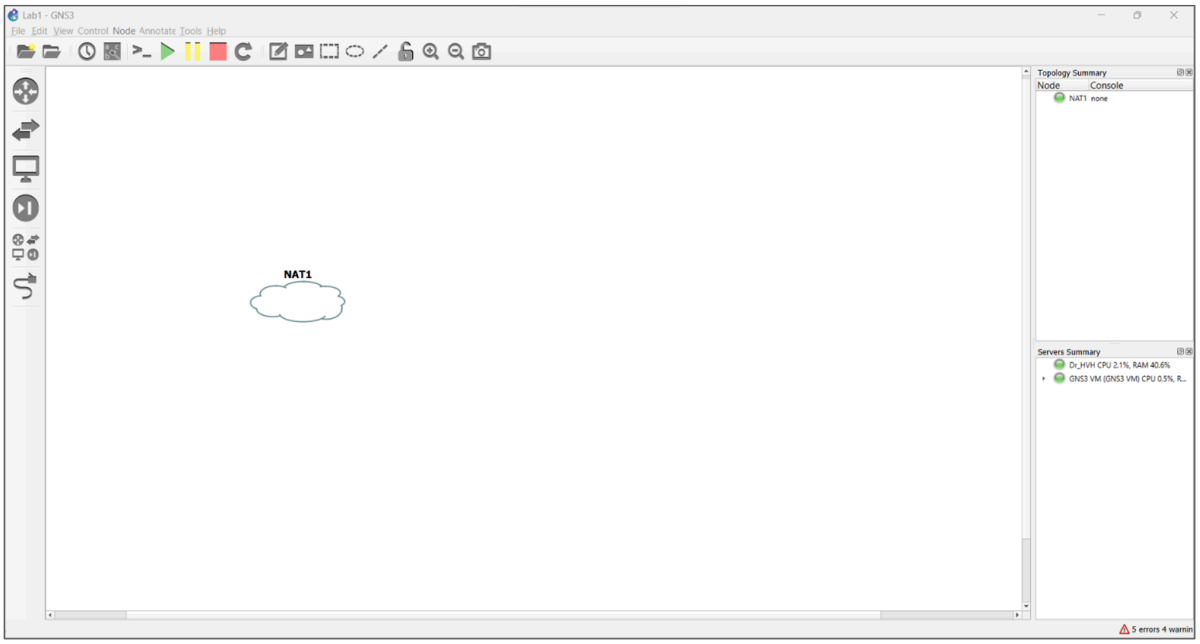


Figure 5 – Add a NAT cloud to GNS3

> **NOTE:** Whenever you are asked to choose a server, use the dropdown menu to select the ***GNS3 VM (GNS3 VM)*** option.

4.4. Change the name from “NAT1” to “ISP” by double-clicking on the name and pressing ***OK***

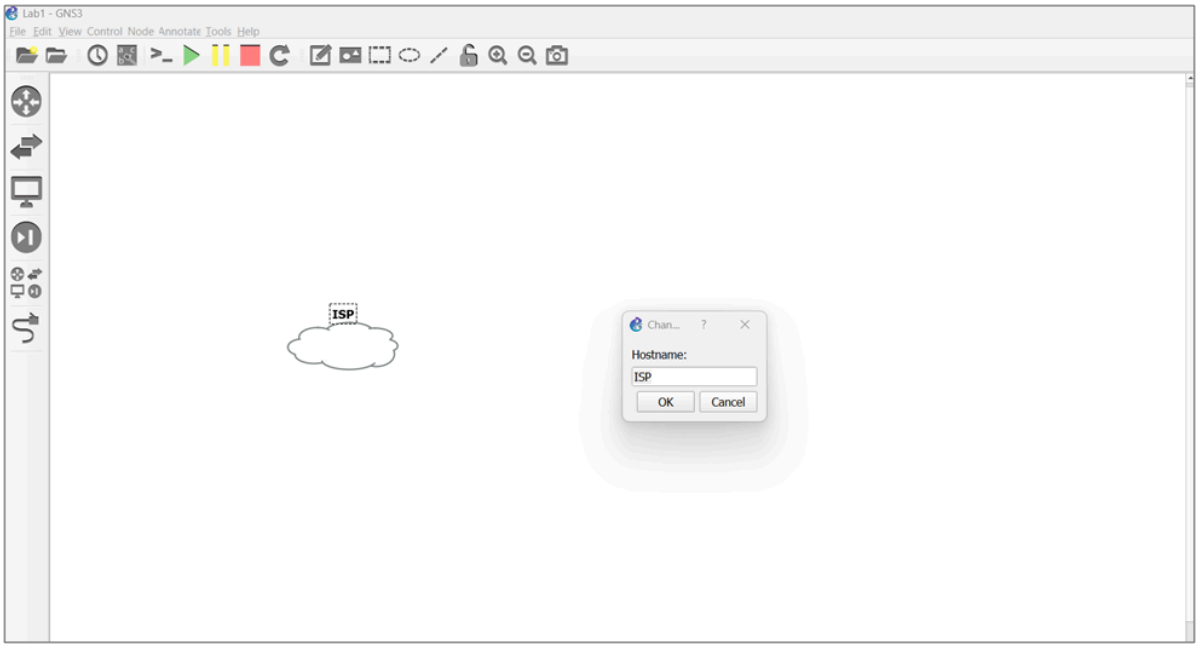


Figure 6 – Change NAT cloud name to ISP

4.5. Again, click on the ***Browse all devices***

4.6. Drag the device called ***OpenWrt*** to the workplace

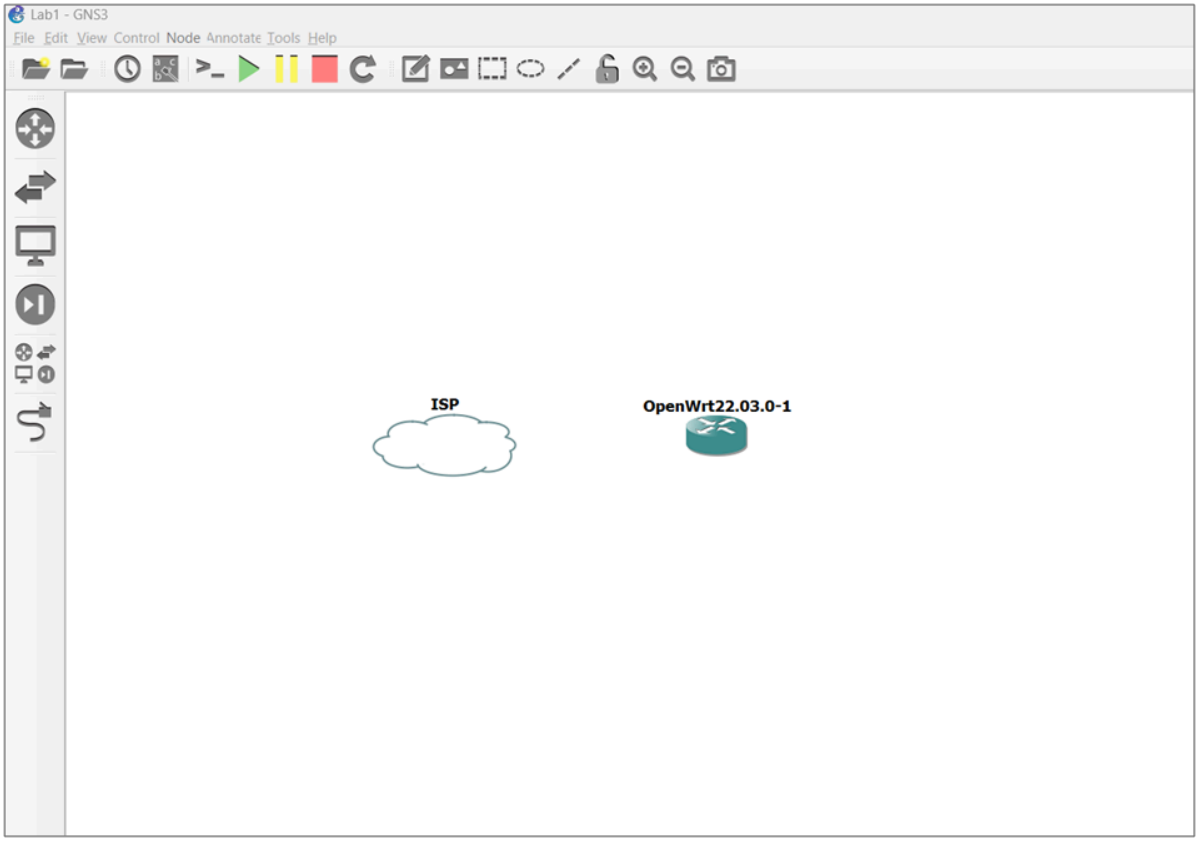


Figure 7 – add OpenWRT router

4.7. The default symbol for the OpenWrt router is the universal symbol for routers, but it can be hard to see at times. Change the symbol by right-clicking on the router and then selecting ***Change Symbol*** on the menu

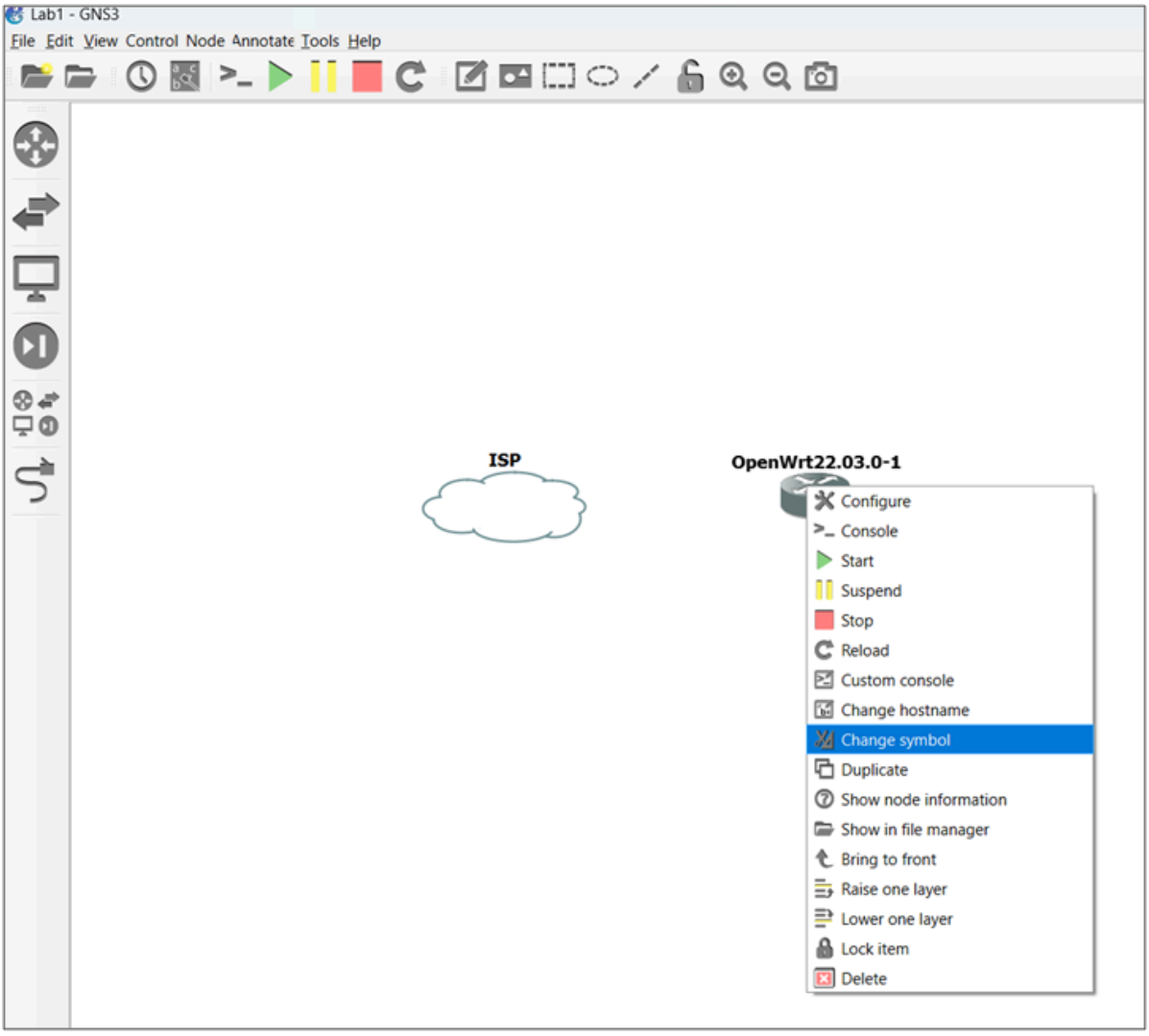


Figure 8 – Changing the Router Symbol

4.8. Select the ***Affinity-square-red*** option and look for the router symbol, then click on ***OK***

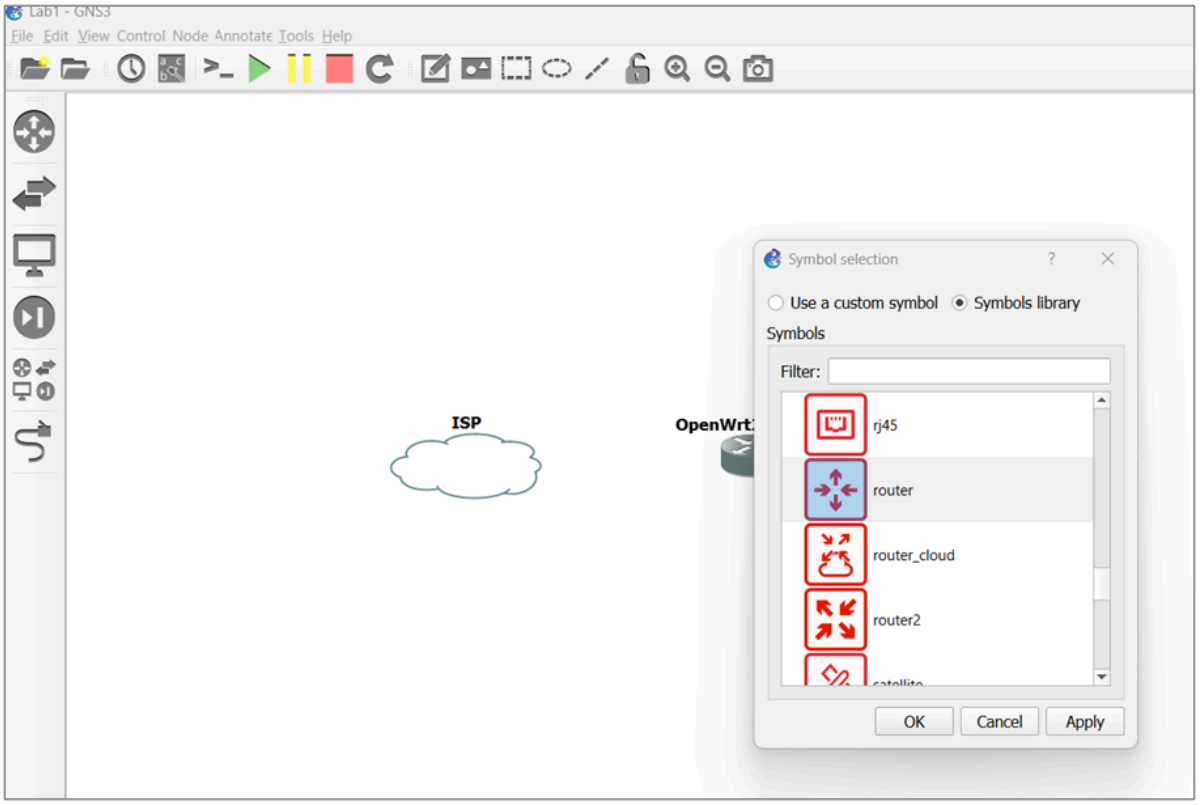


Figure 9 – Select the red router symbol

4.9. Double-click on the router name and change it from “OpenWrt-1” to “Router“

5. Add a network switch

5.1. Click on the ***Browse all devices*** button again

5.2. This time drag the ***Ethernet switch*** to the workspace

5.3. Use the GNS3 VM to host the switch

5.4. Rename the device to “Switch”

5.5. Change the symbol to the***Affinity-square-red*** option for the switch, by right-clicking on the switch and selecting the change symbol option. The workspace should now look similar to the following

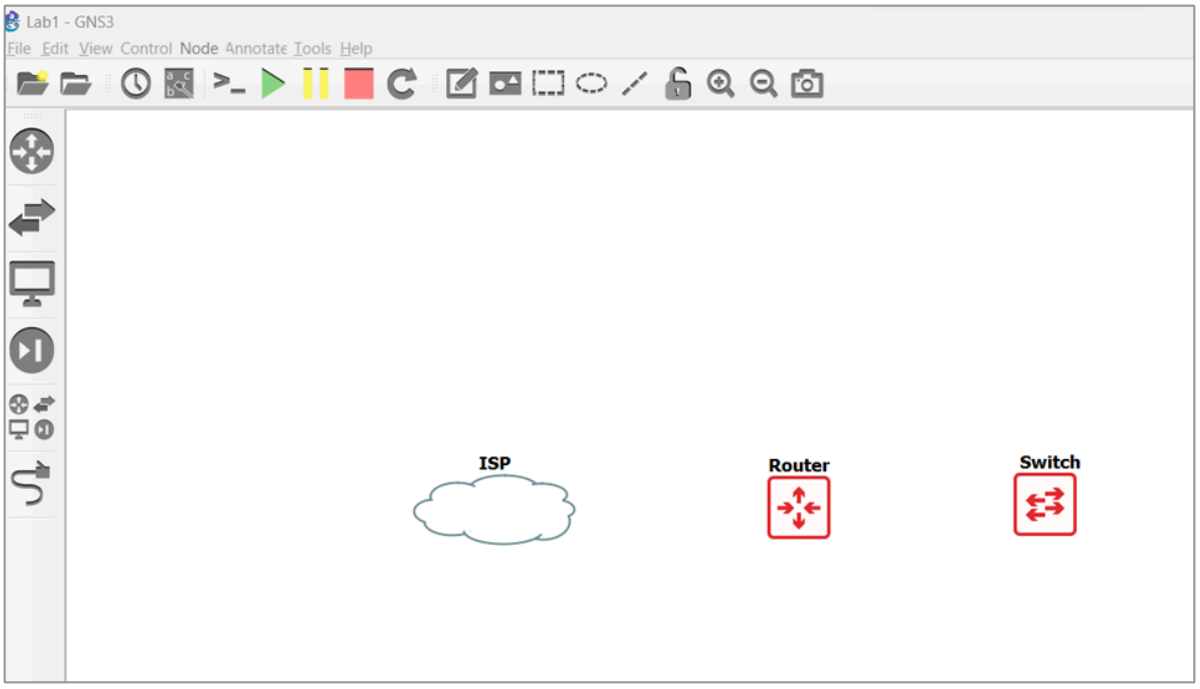


Figure 10 – add a switch

6. Add a VM with an Internet browser

    6.1. Add a VM to GNS3 (we are using Ubuntu Mate, but any VM with a browser should work)

    6.2. Allow GNS3 to configure VirtualBox network settings

        6.2.1. Go to ***File*** > ***Preferences***

        6.2.2. Navigate to the **VirtualBox VMs** section

        6.2.3. Double click on your imported VM

        6.2.4. Under the **Network** tab, check the box that says ***Allow GNS3 to use any configured any VirtualBox adapters***

    6.3. Rename the device to “PC” and your workspace should look like the following

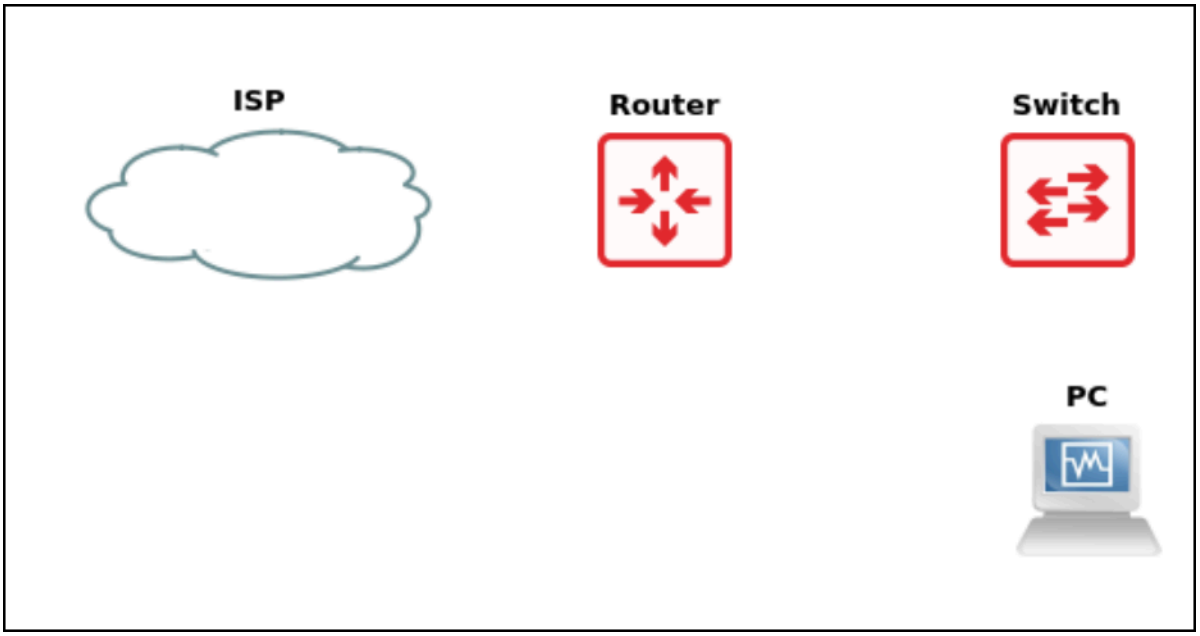


Figure 11 – the network built so far

> **NOTE:** We generally use GNS3's built-in virtual personal computer simulators (VPCS) because using too many VirtualBox VMs can drag down your system's performance. However, the VPCS does not have a browser to use the GUI interface of the OpenWrt router.

7. Link the devices together

    7.1. Return to the GNS3 workspace

    7.2. On the left side of the GNS3 workspace click on the ***Add a link button*** 

    7.3. Click on the ISP cloud and then click on the ***nat0*** port in the sub-menu

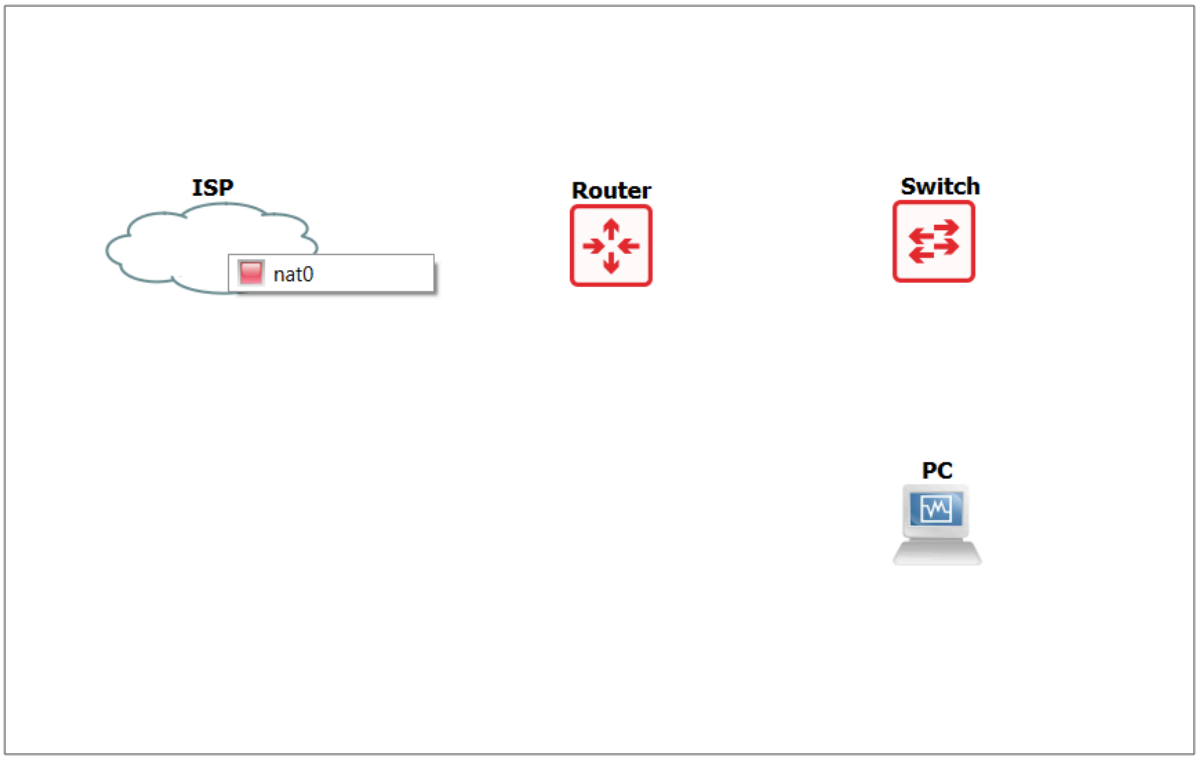


Figure 12 – Select NAT interface

7.4. Click on the Router and then select the ***Ethernet1*** port in the sub-menu. It is **VERY IMPORTANT** that this cable is connected to port Ethernet1. This is like plugging (screwing) in a cable from the ISP to your home router/switch/modem port that is commonly labeled "Internet"

7.5. Click on the Router and then click on the ***Ethernet0*** port in the sub-menu. This is like plugging a cable into your home router/switch/modem port that is commonly labeled "1" or "PC"

7.6. Click on the switch and select any red (unused) port

7.7. Now click on the switch and select another unused port and attach it to the PC's ***Ethernet0*** port

7.8. Show the interface labels by clicking on the ***Show/Hide interface labels*** button 

8. Your workspace should now look like the following

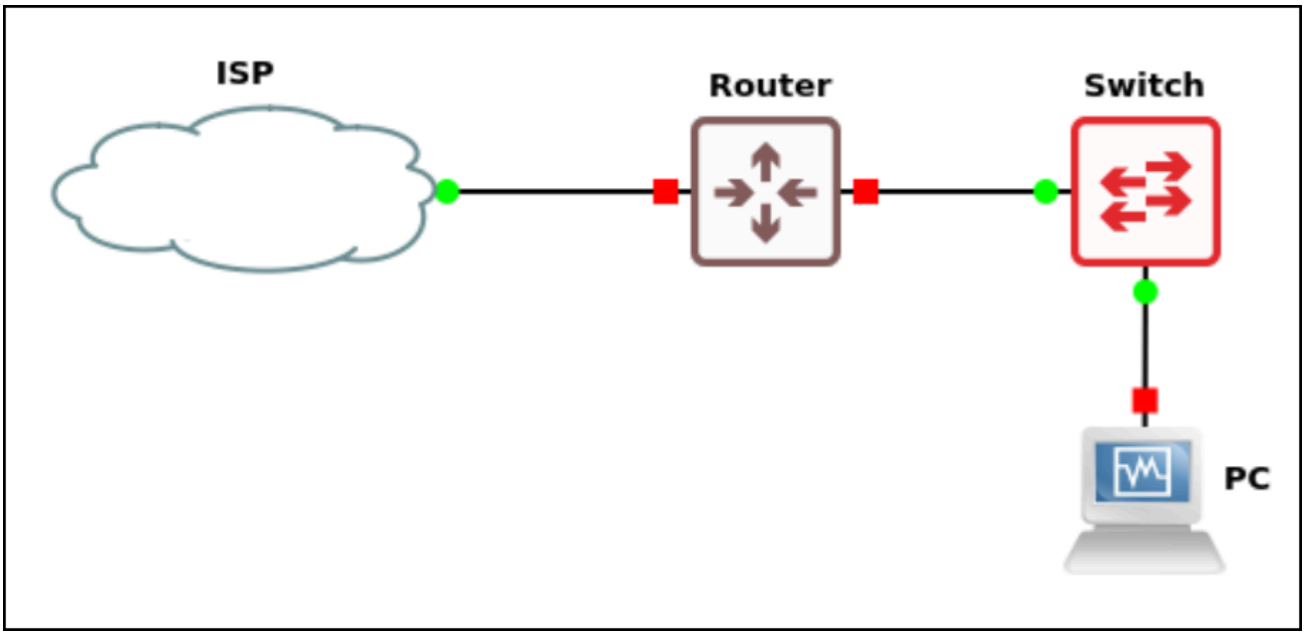


Figure 13 – All devices connected

NOTE: Notice that some of the ends show red, this means that the device is turned off or the interface has been disabled.

9. Now press the big green arrow to start all the devices  . All cable points should eventually turn green as all the devices boot up

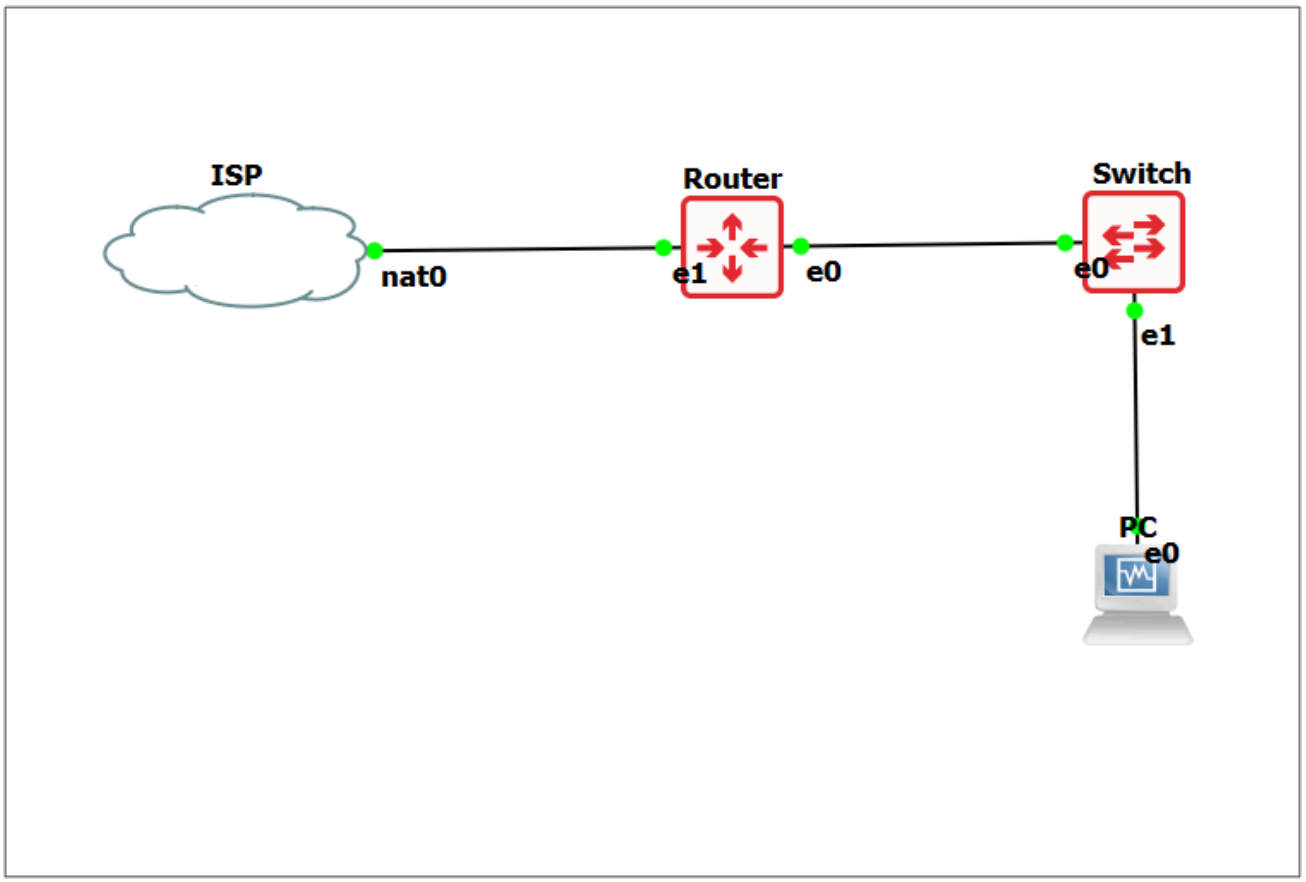


Figure 14 – all connections are green

Phase III – Interface with the Home Router

Most home routers are configured by using a PC or laptop. We are going to use our virtualized PC to do the same thing.

1. Navigate to your VM instance and log in. Remember, in this example, we are using Ubuntu Mate

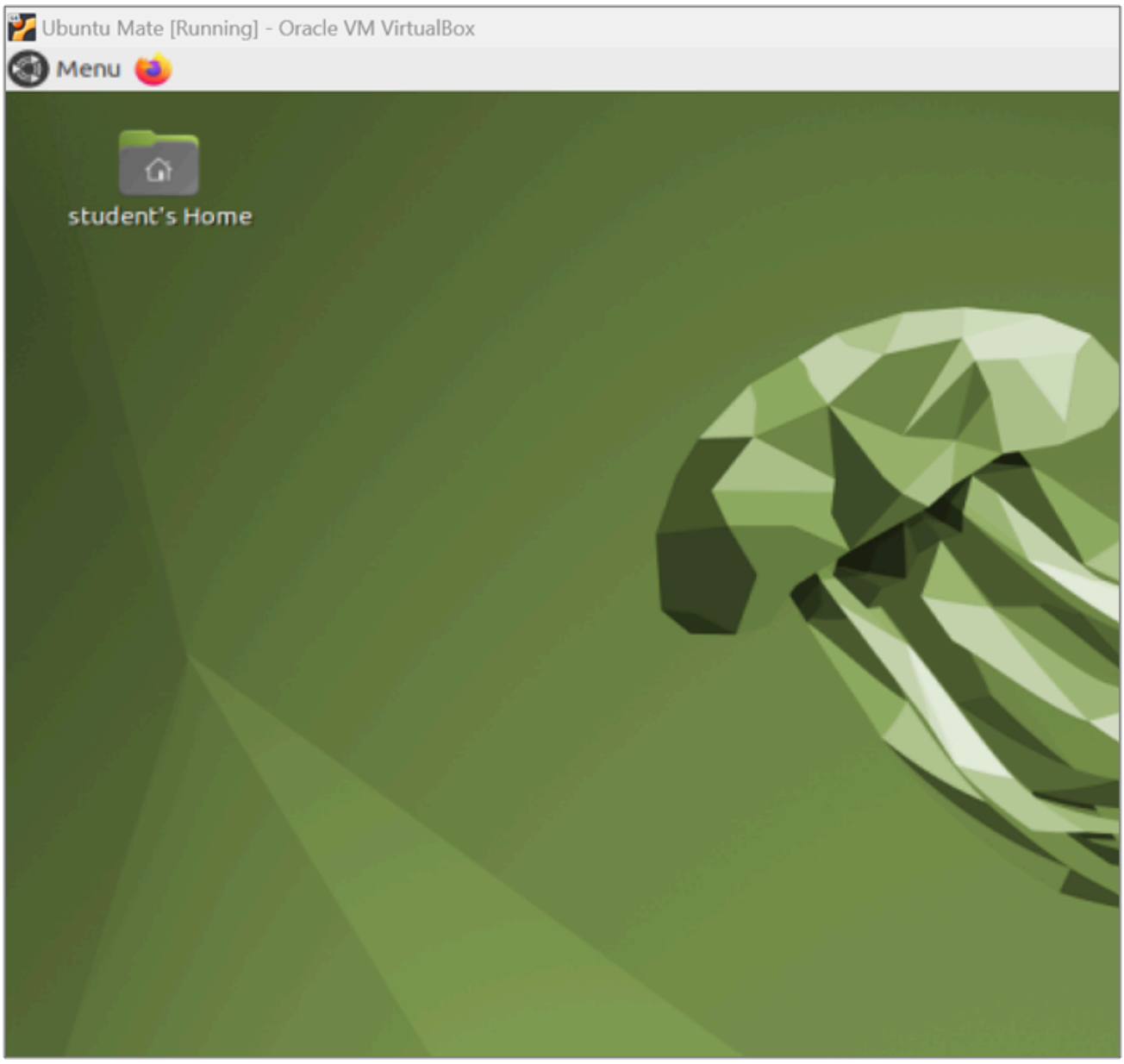


Figure 15 – Ubuntu Mate

2. Access the OpenWrt router management webpage

    2.1. Open a browser application (ex. Firefox)

    2.2. In the navigation bar type 192.168.1.1 and press ***Enter***

    2.3. You should be at the OpenWrt GUI interface

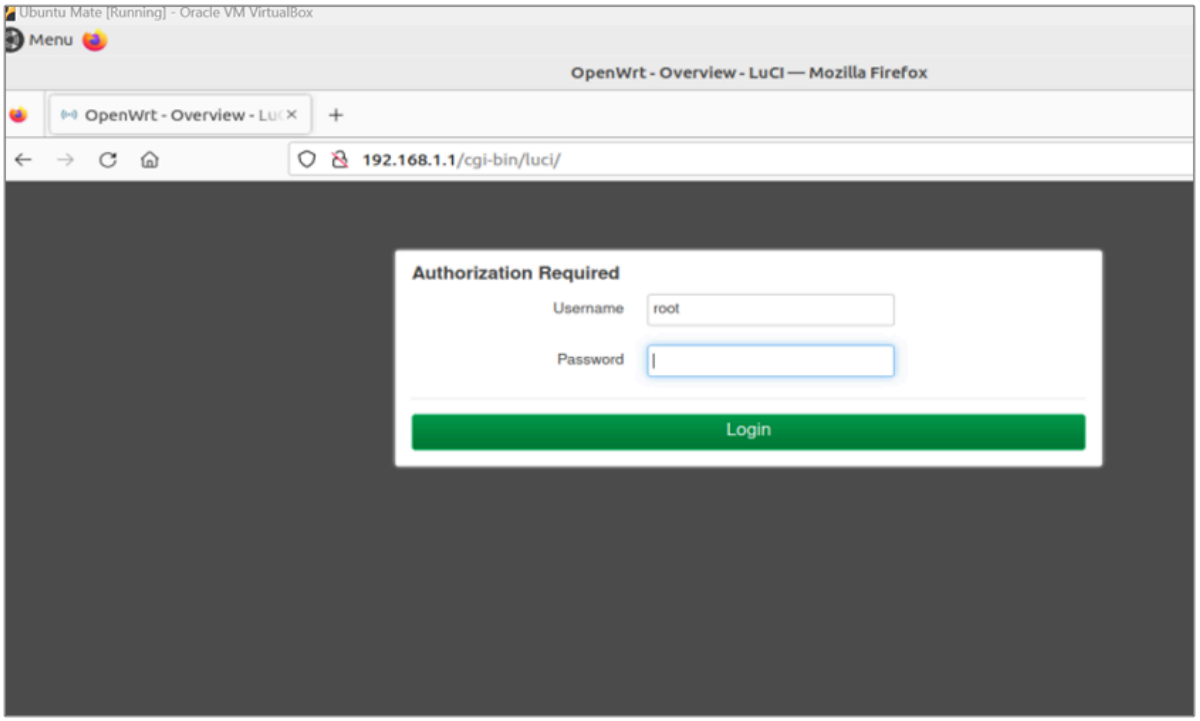


Figure 16 – Web interface for OpenWRT

2.4. The username is “root” and there is no password, so just click ***Login***

3. It will take you to the status overview section. You can scroll through this information and see that devices are connected to the router. The network information is the ISP and the DHCP leases show the VM you are using now

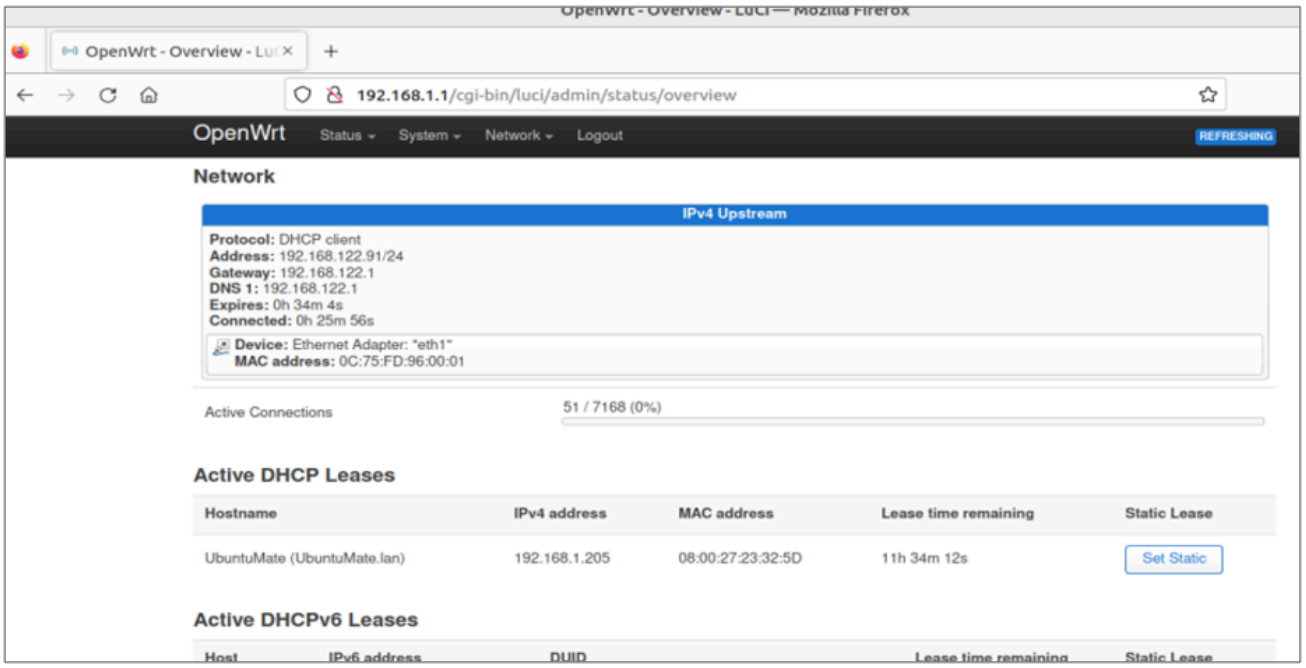


Figure 17 – OpenWRT network leases

4. If you click around on the OpenWrt router, you can see it has many of the same settings as a Linksys or TP-Link router. Don't change any settings at this time. We want to add more devices

Phase IV – Add More Devices to the Home Network

Rarely do home networks have only a single device. So we are going to add a few more.

1. Click on the ***Browse all devices*** button and drag a VPCS device to the workspace

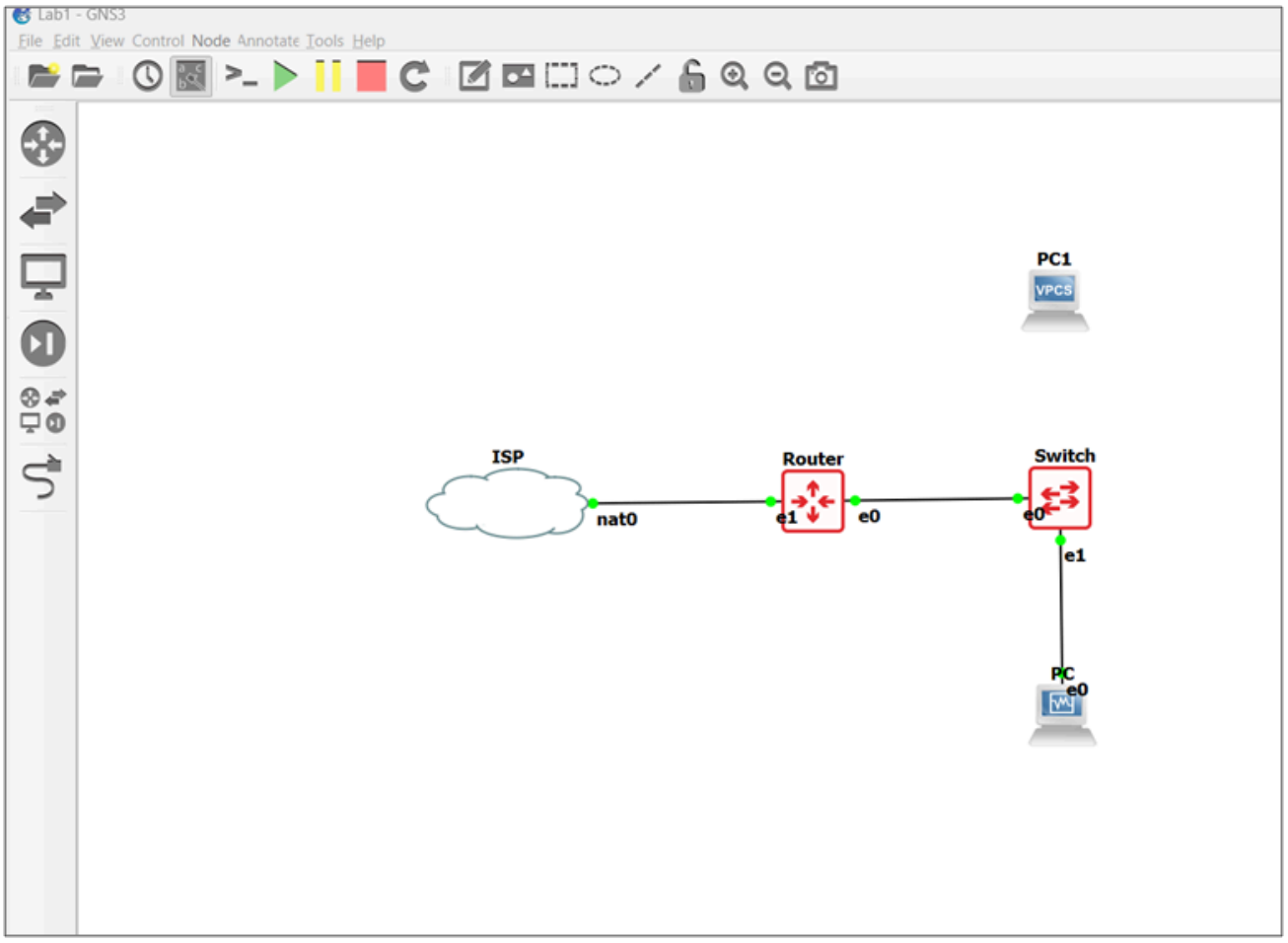


Figure 18 – Add a VPCS

NOTE: This device is very lightweight and does not require nearly as many resources as a VirtualBox VM. Don't forget to use the GNS3 VM when asked

2. Use the techniques learned earlier and make the following changes

    2.1. Change the VPCS symbol to a laptop

    2.2. Change the name to "Laptop"

    2.3. Connect a cable from the laptop to the switch

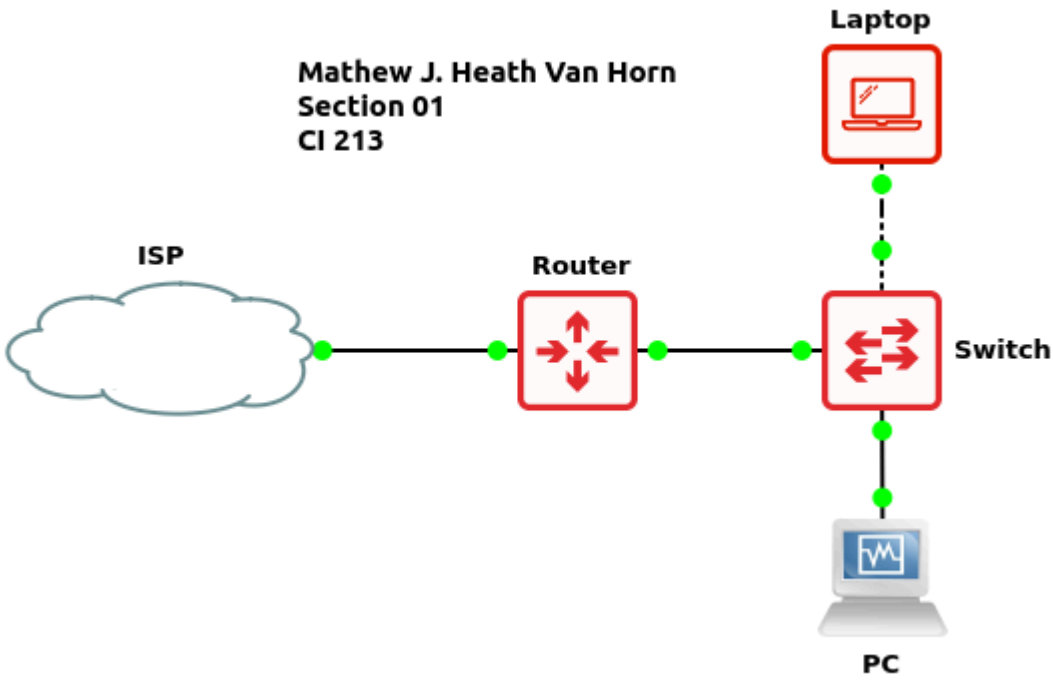


Figure 19 – Changing to laptop

3. Laptops are typically wireless, so let's change the connecting line to reflect this

    3.1. Right-click on the cable and select ***Style***

    3.2. On the submenu change the border style to ***Dash Dot Dot***.

    3.3. Press ***Apply*** and ***OK*** to close the submenu

4. Turn on the laptop by right-clicking and selecting ***Start***

5. Add the laptop to the network

    5.1. Right-click on the laptop and click on ***Console***

    5.2. Read the opening statements. Note that if you get lost, you can always enter the question mark to get assistance

    5.3. At the prompt type the following command to request an IPv4 address and press ***Enter***

```
> ip dhcp
```

5.4. After a few seconds, you will get a message reporting which IP address was assigned

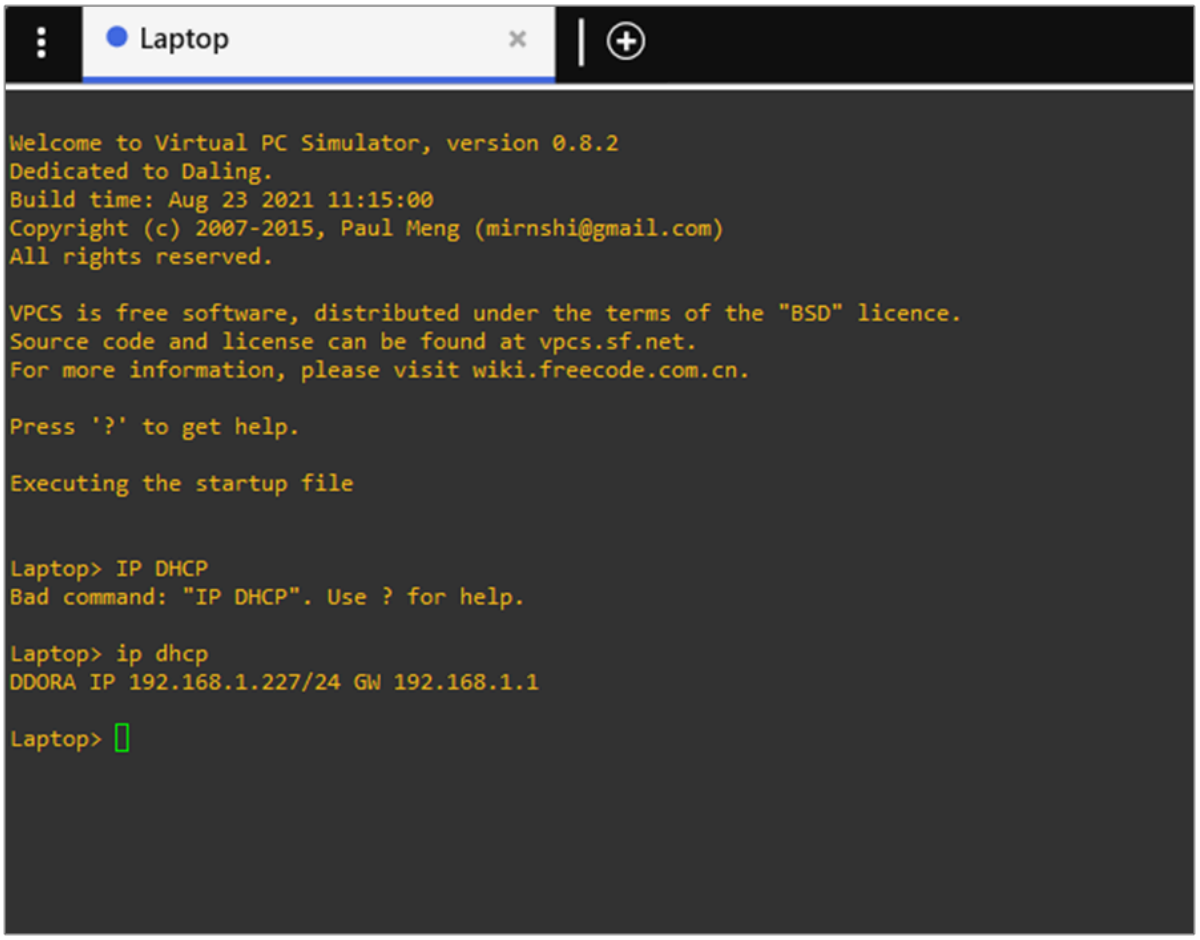


Figure 20 – VPCS (laptop) showing DHCP connection

> **NOTE:** If you get lost, you can always enter a question mark ***[?]*** to get assistance such as available commands and their syntax.

6. While there is no browser application to open, you can still test Internet connectivity via the ping command

```
> ping www.google.com
```

```
VPCS is free software, distributed under the terms of the "BSD" licence.
Source code and license can be found at vpcs.sf.net.
For more information, please visit wiki.freecode.com.cn.

Press '?' to get help.

Executing the startup file


Laptop> ip dhcp
DDORA IP 192.168.1.227/24 GW 192.168.1.1

Laptop> ping www.google.com
www.google.com resolved to 192.178.48.228

84 bytes from 192.178.48.228 icmp_seq=1 ttl=59 time=41.511 ms
84 bytes from 192.178.48.228 icmp_seq=2 ttl=59 time=29.067 ms
84 bytes from 192.178.48.228 icmp_seq=3 ttl=59 time=29.096 ms
84 bytes from 192.178.48.228 icmp_seq=4 ttl=59 time=29.509 ms
84 bytes from 192.178.48.228 icmp_seq=5 ttl=59 time=26.174 ms

Laptop>
```

Figure 21 – Pinging devices

CONGRATULATIONS!
YOU HAVE BUILT YOUR FIRST NETWORK IN GNS3!

End of Lab

Deliverables

One screenshot is needed of your GNS3 environment:

- Click on the ***Add a note*** button at the top of the workspace and put your name, class, and section on your workspace 
- Select the***Take a screenshot*** button to save a of your workspace and submit it per the instructor's instructions to receive credit for completing this activity 
- Example output…

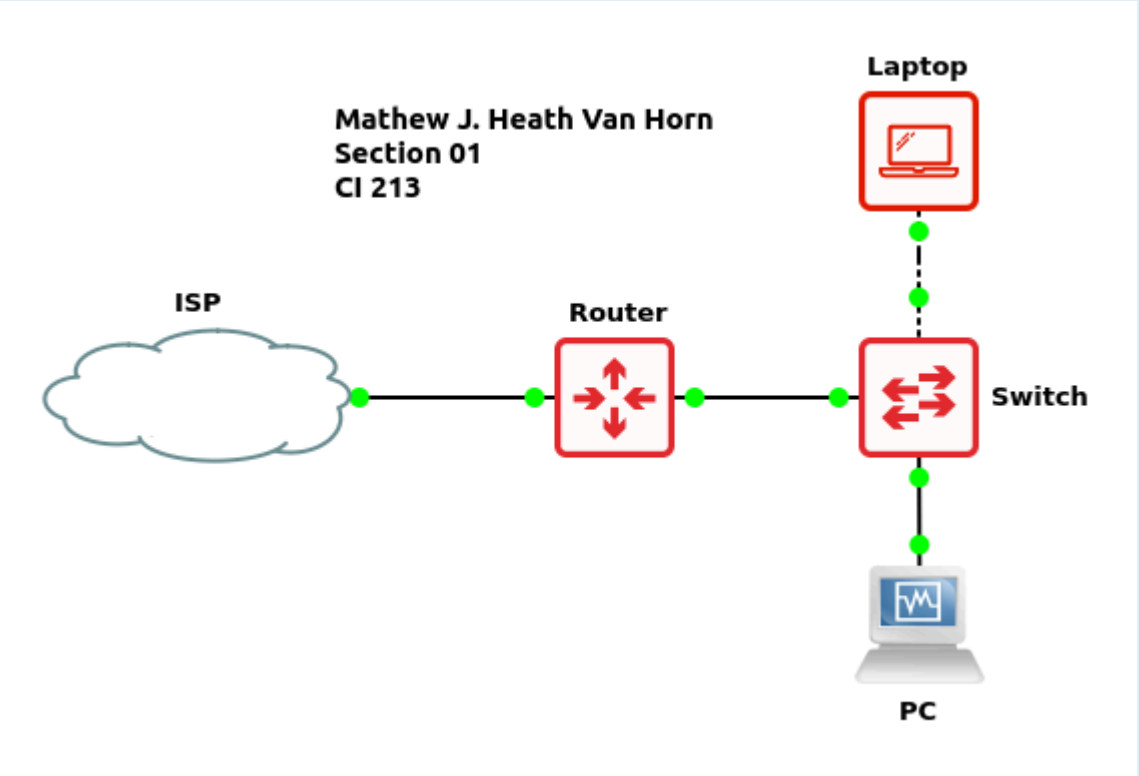


Figure 22 – Example

Homeworks

Assignment 1 – Add more devices

- Add another VPCS and change the label connection and picture to resemble a Cell Phone
- Add another VPCS and change the device's label and picture to resemble a desktop printer
- RECOMMENDED GRADING CRITERIA
    - Screenshot of the new GNS3 working environment with everything labeled

CHAPTER 15

Hubs and Switches

RAECHEL FERGUSON

Hubs are not often used in modern network environments, but learning how they operate still provides a solid foundation to enterprise networks. Alternatively, switches are ubiquitous in network implementations. Finally, learners can use this exercise to gain more experience with GNS3 and VirtualBox.

Estimated time for completion: 15 minutes

LEARNING OBJECTIVES

- Understand how hubs function in a network
- Understand how switches function in a network
- Gain further experience using GNS3
- Introduce the use of Wireshark

PREREQUISITES

- Chapter 2 – Setting Up a GNS3 Environment
- Chapter 14 – Your First Network

DELIVERABLES

Six screenshots are required:

- GNS3 workspace
    - With Hub
    - With Switch
- Wireshark capture between PC3 and the specified network device
    - PC1 pinging PC2 (hub)
    - PC2 pinging PC3 (hub)
    - PC1 pinging PC2 (switch)

- PC4 pinging PC3 (switch)

RESOURCES

- **NOTE: Each source will referenced with its corresponding number in superscript (EX: [1]) at the end of a step**
- 1.“Wireshark Introduction.” Chapter 1. introduction. Accessed May 27, 2024. https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html.
- 2. Bombal, David. “GNS3 Tips: Using the GNS3 Hub and Switch with Wireshark (Part 1).” YouTube, November 28, 2016. https://www.youtube.com/watch?v=lHcF6KXMPJ0.
- 3. Bombal, David. “GNS3 Tips: Using the GNS3 Hub and Switch with Wireshark (Part 2).” YouTube, November 28, 2016. https://www.youtube.com/watch?v=27KdkT0yIxg&t=2s.
- 4. Bombal, David. “GNS3 Tips: Using the GNS3 Hub and Switch with Wireshark (Part 3).” YouTube, November 28, 2016. https://www.youtube.com/watch?v=kgFxGM9E3tI&t=1s.

CONTRIBUTORS AND TESTERS

- Sawyer T. Hansen, Cybersecurity Student, ERAU Prescott
- Quinton D. Heath Van Horn, 7th grade
- David Reese, Mathematics Major, SUNY Bridgeport
- Stephen Torres, 9th grade
- Jacob M. Christensen, Cybersecurity Student, ERAU-Prescott
- Dante Rocca, Cybersecurity Student, ERAU-Prescott

Phase I – Setup

We complete the setup first so that we can focus on the learning activities later on. Your network should look like the following image:

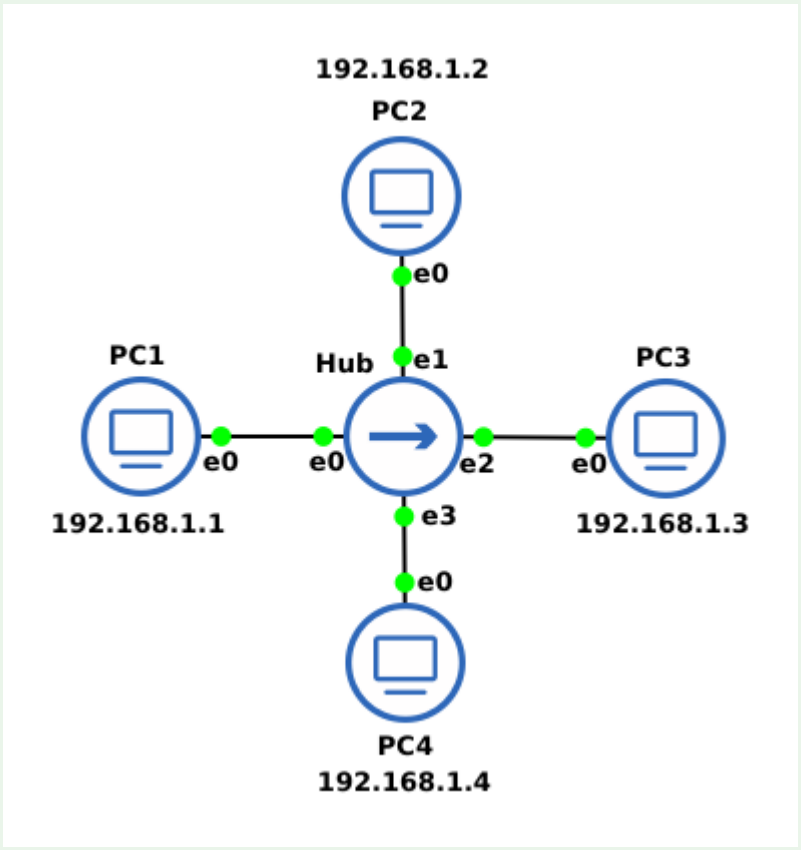


Figure 1 – Final Network Topology

1. Start GNS3

    1.1. Create a new blank project: **LAB_02**

    1.2. Verify the GNS3 VM is running by looking for the green light under the **Servers Summary** section

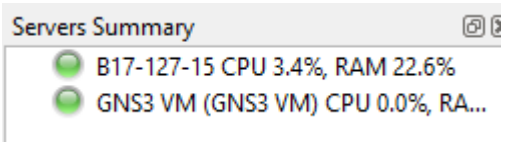


Figure 2 – GNS3 server summary

2. Add a hub to the GNS3 workspace [2]

    2.1. Click on *Browse all devices* 

2.2. Drag the ***Ethernet hub*** device into the workspace

> **NOTE:** When given the option, remember to always choose ***GNS3 VM*** as the server to host devices.

3. Add the PCs to the network

    3.1. Click on ***Browse all devices*** [2]

    3.2. Drag a ***VPCS*** device to the main workspace [2]

    3.3. Repeat this three more times so you have four VPCS devices in the workspace [2]

4. Add labels and change device symbols as you see fit

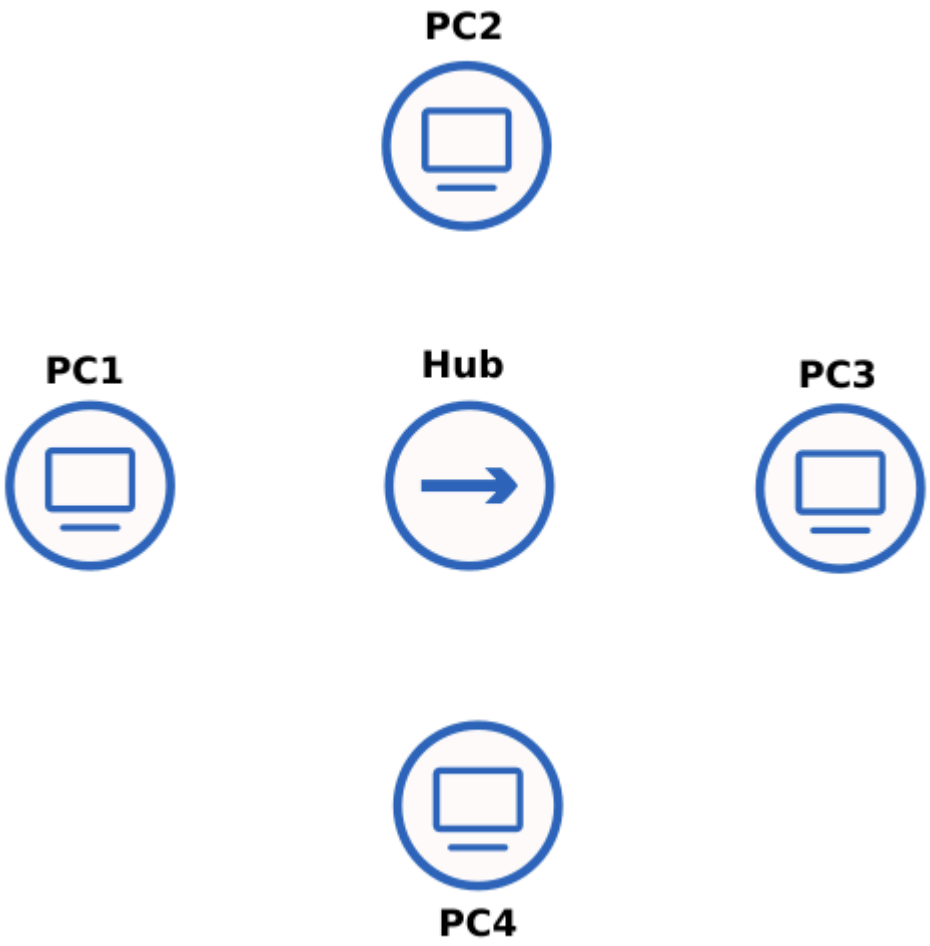


Figure 3 – GNS3 Workplace

Phase II – Assign an IP address for each of the PCs

We will learn more about networking addresses later on. However, for now, just know that an IP address is like a mailing address or a person's name. It is used to identify "This Device" amongst a sea of other IT devices. We are going to name our devices, but instead of using names like "Todd" or "Main Street", we are giving the device a name like "192.168.10.56".

1. Our network space for this lab is **192.168.1.0/24**

2. Click ***Start/Resume all nodes*** to power on all devices 

3. Assign an IP address to PC1 [2]

   3.1. ***Right-click*** on PC1 and select ***Console*** [2]

   3.2. At the PC1 prompt, assign it an IP address with a Class C subnet mask [2]

```
> ip 192.168.1.1/24
```

```
PC1> ip 192.168.1.1/24
Checking for duplicate address...
PC1 : 192.168.1.1 255.255.255.0

PC1>
```

Figure 4 – VPCS assign IP address

   3.3. At the PC1 prompt type "save" to keep the IP configuration the next time the PC is rebooted

```
> save
```

   3.4. After a few seconds, the newly entered IP address will be assigned to PC1

```
> show ip
```

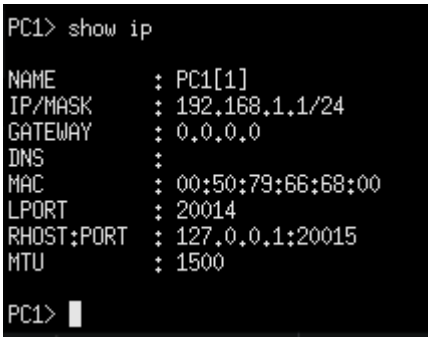


Figure 5 – Display currently assigned IP address

3.5. In the GNS3 workspace add a text label of "192.168.1.1" next to PC1

4. Repeat the step 3 to configure IP addresses for the remaining PCs (Figure 6)

Device	IPv4 Address
PC1	192.168.1.1
PC2	192.168.1.2
PC3	192.168.1.3
PC4	192.168.1.4

Phase III – Connect the devices

We have to connect the devices to the hub in order for the devices to 'talk' to each other. Even wireless devices have a connection, we can't see it with our eyes, but there is a connection.

1. Connect PC1

1.1. On the left side of the GNS3 workspace click the ***Add a link*** option [2] 

1.2. Click on PC1 and select ***Ethernet0*** port in the sub-menu [2]

1.3. Click on the hub and select any open Ethernet port available [2]

1.4. Click the ***Show/Hide interface labels*** to view the connections 

2. Repeat step 1 for the remaining PCs

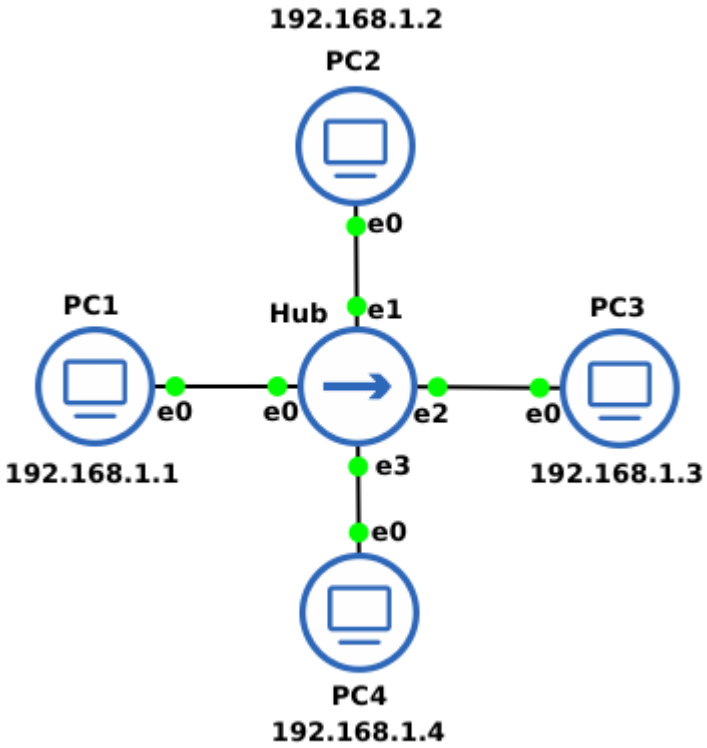


Figure 7 – Finalized GNS3 network

> **NOTE:** You can right-click anywhere in the workspace to de-select the cable.

Phase IV – Observe the network communications

We can use a common network monitoring tool called Wireshark to view how the networked devices 'talk' to each other. Think of electricity in a home. In order to 'see' the electricity, we use a tool called a voltmeter. Wireshark is a great tool to watch live network packets as they are transmitted.

1. Start a Wireshark capture between PC2 and the hub

    1.1. Hover your mouse over the connection between the two devices [3]

    1.2. ***Right-click*** the cable between the two devices and select ***Start capture***[3](Figure 8)

    1.3. A new Wireshark window will open [3] (Figure 9)

    1.4. A magnifying glass will appear on the wire to show you where the network sniffing is

taking place 

2. Now open the PC1 console and ping the IP address for PC2

```
> ping 192.168.1.2
```

```
PC1> ping 192.168.1.2

84 bytes from 192.168.1.2 icmp_seq=1 ttl=64 time=0.385 ms
84 bytes from 192.168.1.2 icmp_seq=2 ttl=64 time=0.338 ms
84 bytes from 192.168.1.2 icmp_seq=3 ttl=64 time=0.503 ms
84 bytes from 192.168.1.2 icmp_seq=4 ttl=64 time=0.539 ms
84 bytes from 192.168.1.2 icmp_seq=5 ttl=64 time=0.540 ms

PC1>
```

Figure 10 – PC1 pinging PC2

3. Now watch for the ICMP ping requests and replies in the Wireshark window [1,3]

192.168.1.1	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.1	ICMP	Echo (ping) reply
192.168.1.1	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.1	ICMP	Echo (ping) reply
192.168.1.1	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.1	ICMP	Echo (ping) reply
192.168.1.1	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.1	ICMP	Echo (ping) reply
192.168.1.1	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.1	ICMP	Echo (ping) reply

Figure 11 – Captured ICMP packets in Wireshark

NOTE: Don't be overwhelmed by the amount of information and options available within Wireshark. Right now, we are only looking at a small fraction of all possible network traffic. One benefit of using these simulated network environments is to focus only on information that is important, so that we can build up to observing more complicated packet streams in the future.

Phase V – Observe Hub Operations

Hubs are unique in that when any ethernet packet is sent to the hub, the hub will retransmit the packet to every device connected to the hub. Any PC on the network can see the ethernet packets of every device using the

network. This eavesdropping is one of the reasons that hubs are rarely used anymore. We are going to watch how PC1 can eavesdrop on the packets being sent between PC2 and PC3.

1. Ensure you have an active Wireshark capture on the **PC1-Hub** connection [3]

2. Open the console on PC2 and ping PC3

```
> ping 192.168.1.3
```

3. In Wireshark, observe the packets between PC2 and PC3 [1,3]

192.168.1.3	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.3	ICMP	Echo (ping) reply
192.168.1.3	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.3	ICMP	Echo (ping) reply
192.168.1.3	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.3	ICMP	Echo (ping) reply
192.168.1.3	192.168.1.2	ICMP	Echo (ping) request
192.168.1.2	192.168.1.3	ICMP	Echo (ping) reply

Figure 12 – Captured ICMP packets in Wireshark

4. Feel free to experiment; try capturing packets between any PC and the hub and then ping different PCs

Phase VI – Replace the hub with a switch

Switches vary widely in function and purpose. However, the one thing they have in common is what makes them different from hubs. Switches only forward Ethernet packets from the source to the destination. We are going to watch what our packets do when we replace a hub with a switch

1. Replace the hub with an ***Ethernet switch***

    1.1. ***Right-click*** on the hub and delete it from the workspace [4]

    1.2. Click on ***Browse all devices 4***

    1.3. Drag the device labeled ***Ethernet switch*** to the workspace [4]

2. Reconnect the computers to the switch by attaching cables

NOTE: Again, don't worry about what interfaces to use. I personally use interface 1 for PC1, and interface 2 for PC2, etc., but it doesn't matter for this lab.

3. Start a Wireshark capture between the PC3-Switch connection [4]

4. From the PC1 console, ping PC3

```
> ping 192.168.1.3
```

   4.1. On Wireshark, you should see the same ICMP request and reply packets as you did earlier [1,4]

5. From the PC1 console, ping PC2

```
> ping 192.168.1.2
```

   5.1. Notice how, unlike the hub, you will not see the ping conversation between PC1 and PC3 in the Wireshark capture [1,4]

6. Feel free to experiment by watching different connections and pinging different devices

End of Lab

Deliverables

Six screenshots are required to receive credit for this exercise:

- Two GNS3 networks with device connections green and neatly labeled
  - Ethernet hub being used
  - Ethernet switch being used
- Wireshark capture between PC3 and the associated network device
  - PC1 pinging PC2 [hub]
  - PC2 pinging PC3 [hub]
  - PC1 pinging PC2 [switch]
  - PC4 pinging PC3 [switch]

Homeworks

Assignment 1 – Build a GNS3 network using 4 PCs and 1 hub

Device	IP Address	Network Mask
PC 1	56.121.149.10	255.255.255.0
PC 2	56.121.149.20	255.255.255.0
PC 3	56.121.149.30	255.255.255.0
PC 4	56.121.149.40	255.255.255.0

Assignment 2 – Build a GNS3 network using 4 PCs and 2 hubs. Connect hub 1 to hub 2 and follow the connection table below.

Hub Number	Device	IP Address	Network Mask
1	PC1	120.107.148.50	255.255.255.0
1	PC2	120.107.148.75	255.255.255.0
2	PC3	120.107.148.200	255.255.255.0
2	PC4	120.107.148.240	255.255.255.0

Recommended binary grading criteria:

- Screenshot of GNS3 Working environment where:
  - All connections are made according to instructions
  - All connections are properly labeled with the correct IP address
  - Interface labels are turned on
- Screenshot of Wireshark packet captures taken from the PC3-Hub link:
  - PC1 successfully pinging PC2
  - PC1 successfully pinging PC4

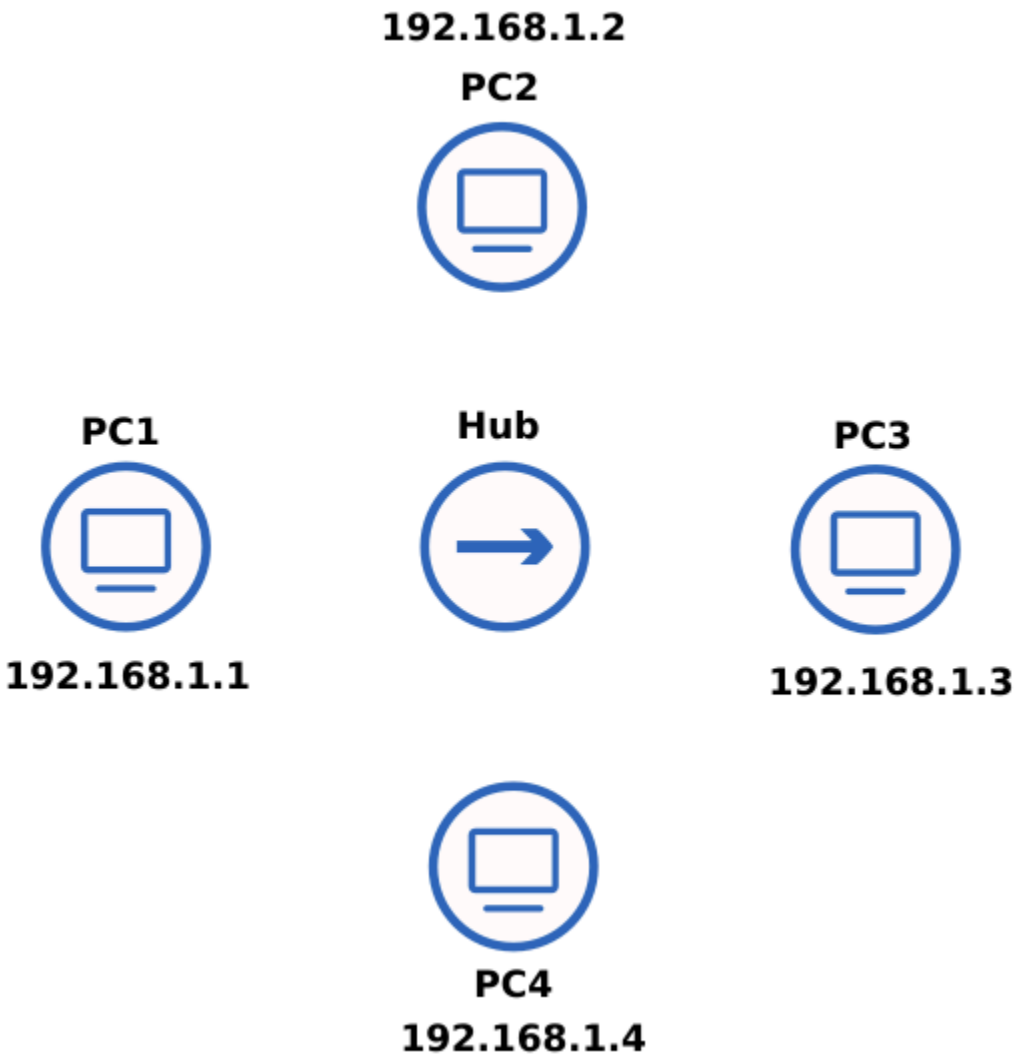


Figure 6 – Assign IP addresses to each VPC

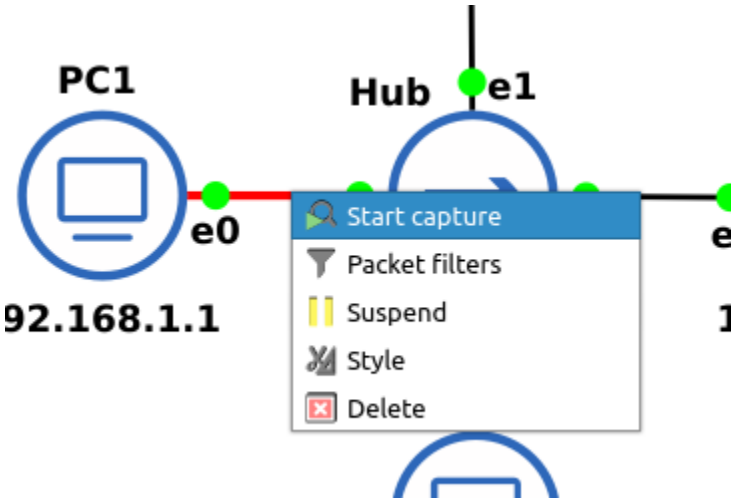


Figure 8 – Start a Wireshark capture

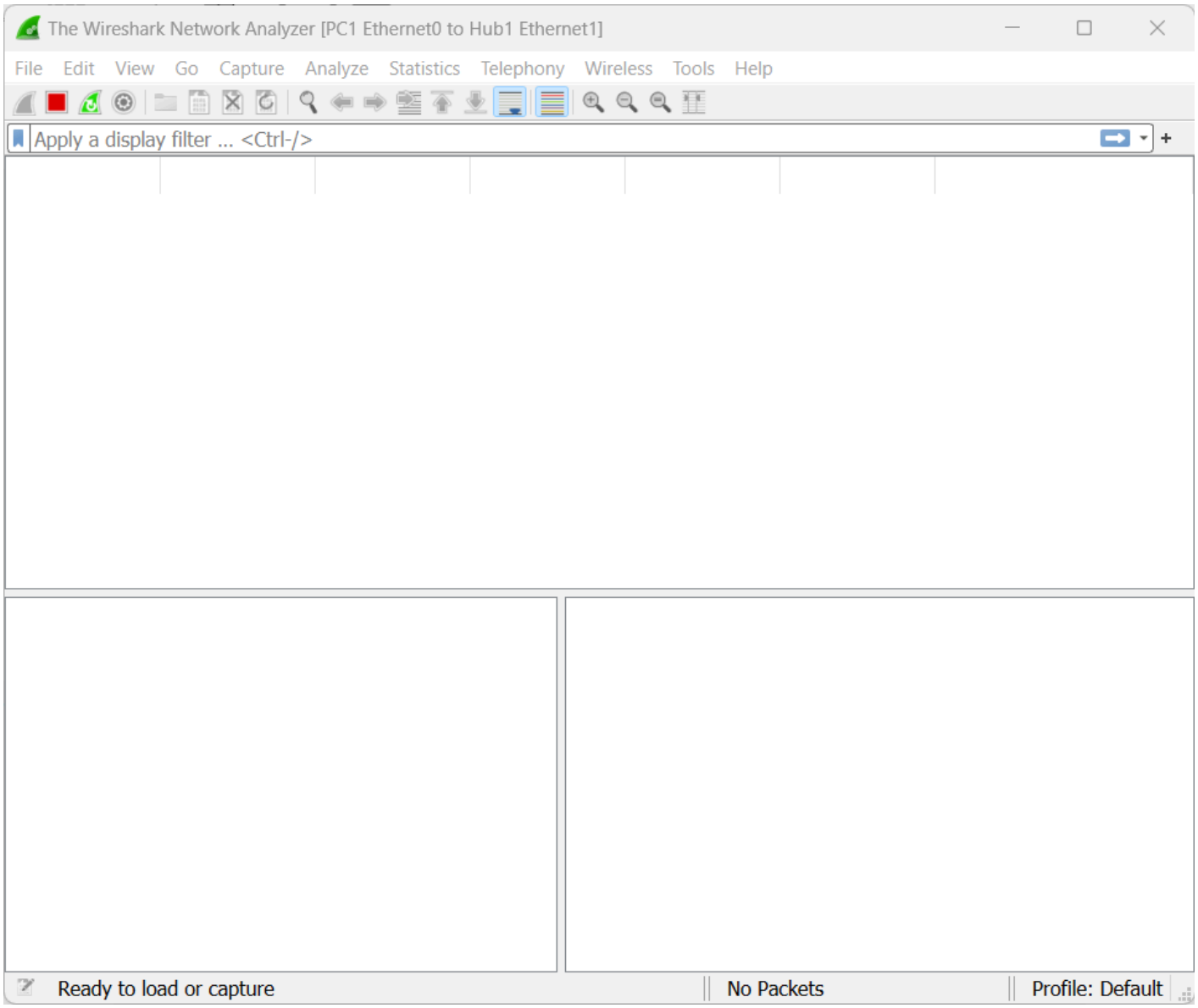


Figure 9 – A new Wireshark window

CHAPTER 16

Introduction to Routers

JACOB CHRISTENSEN

Simply stated, where switches and hubs connect end devices (desktops, laptops, smartphones, etc.), routers connect switches and hubs to each other. Routers are the devices that enable the internet to function. This is an elementary introduction to using routers for first-time learners. This lab will build two LANs and connect them using a MikroTik router.

Estimated time for completion: 20 minutes

LEARNING OBJECTIVES

- Demonstrate successful router configuration using two or more local area networks
- Increase experience in utilizing virtual environments in learning enterprise networks
- Analyze Wireshark results to identify ethernet packets and frames

PREREQUISITES

- Chapter 3 – Adding a MikroTik Appliance in GNS3
- Chapter 15 – Hubs and Switches

DELIVERABLES

Four screenshots are required:

- PC1 console successfully pinging PC4
- Wireshark results (ICMP packets) of PC2 successfully pinging PC3
- Neatly labeled and organized GNS3 Workspace
- Configuration settings of the MikroTik router console (interface print, ip address print)

RESOURCES

- MikroTik RouterOS Documentation, https://help.mikrotik.com/docs/display/ROS/RouterOS

CONTRIBUTORS AND TESTERS

- Quinton D. Heath Van Horn, 7th grade
- David Reese, Mathematics Student, SUNY Bridgeport
- Julian Romano, Cybersecurity Student, ERAU-Prescott
- Dante Rocca, Cybersecurity Student, ERAU-Prescott
- Sawyer Hansen, Cybersecurity Student, ERAU-Prescott

NOTE: Students should be familiar enough with GNS3 by now to understand the commands. This lab starts reducing the number of screenshots since these things were documented in previous chapters.

Overview

You are going to use your LAN knowledge and build two networks. You will then use a router to connect these networks together. Your final product should look similar to this.

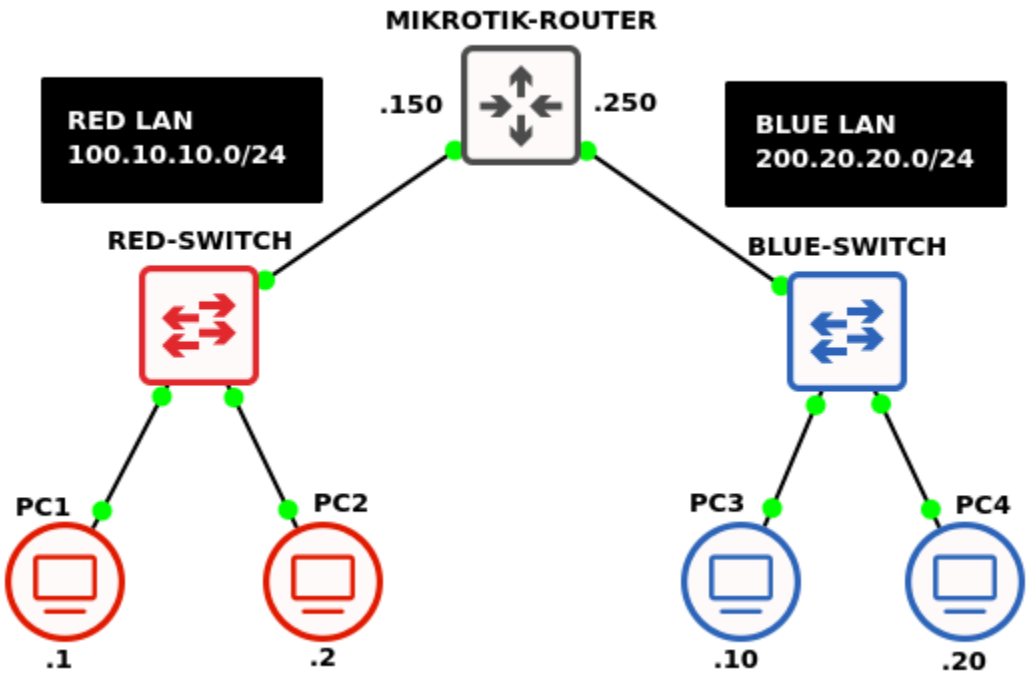


Figure 1 – Final network topology

Phase I – Configure the Red and Blue Networks

Most networks have a specific purpose. The network can be centered around a function (marketing) or

geography (front building). In our examples, we generally use colors to abstract the learners from the specific functions. This way learners won't be locked into a certain configuration for an "Accounting" department.

1. Start GNS3

    1.1. Create a new project: **LAB_03**

2. Build the Red LAN with the network address **100.10.10.0** and netmask **255.255.255.0**

NOTE: /24 is CIDR notation for the subnet mask of 255.255.255.0, which is itself a decimal representation of the binary octets 11111111.11111111.11111111.00000000 used to name network interfaces (the /24 represents the number of 1's used). These PC's happen to accept CIDR notation and subnet mask notation, but not all end devices do. So get familiar with using both ways to assign IP addresses.

    2.1. Use two ***VPCS*** devices for PC1 and PC2

    2.2. Add an ***Ethernet switch***

    2.3. Connect the PCs to the switch

    2.4. Start the PCs and assign them appropriate host addresses

        2.4.1. Configure PC1 to have a host address of **100.10.10.1**

```
> ip 100.10.10.1/24
```

```
> save
```

        2.4.2. Configure PC2 to have a host address of **100.10.10.2**

```
> ip 100.10.10.2/24
```

```
> save
```

    2.5. Organize your network

        2.5.1. Label the machines with both their IP address and hostname for clarity

2.5.2. Add a textbox with the subnet network address using either CIDR notation or traditional subnet masks

2.5.3. To help visually differentiate the subnets, change the device symbols or use the ***Draw a rectangle*** button to encapsulate the LAN with its associated color

> **NOTE:** You may have to send the square to the background by right-clicking on it and then selecting the ***Lower one-layer*** option.

2.6. Test network connectivity by opening the console for PC1 and pinging PC2

```
> ping 100.10.10.2
```

3. Repeat to build Blue LAN with the network space of **200.20.20.0/24**

    3.1. Configure PC3 with the host address **200.20.20.10**

    3.2. Configure PC4 with the host address **200.20.20.20**

4. Verify your network looks similar to the following

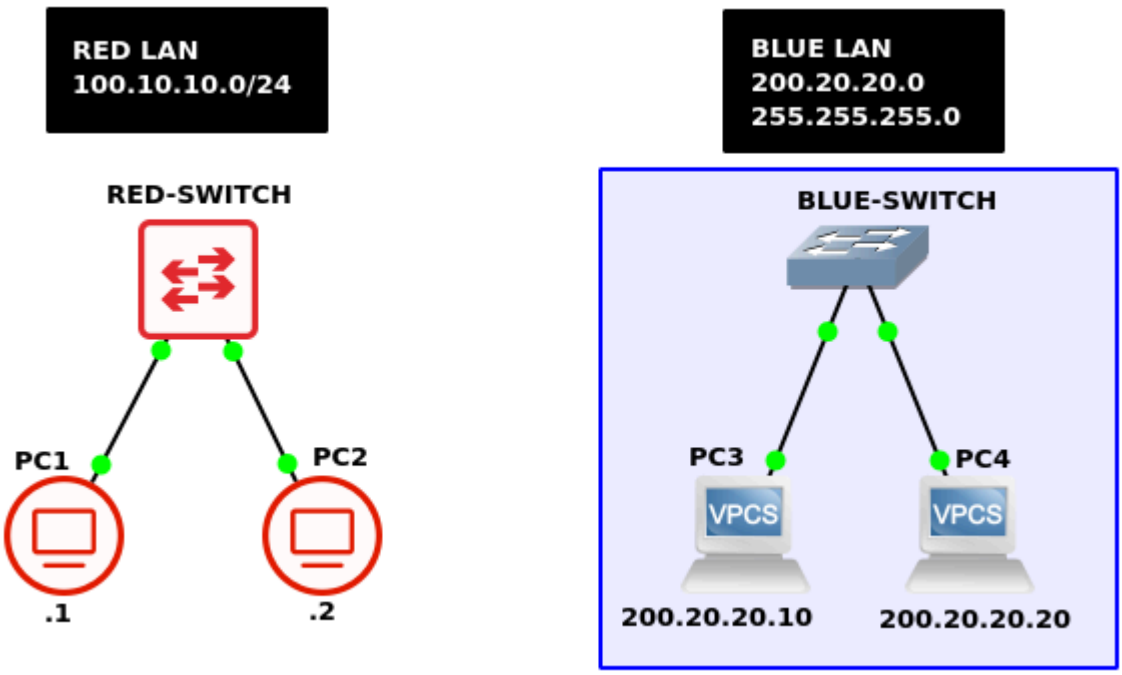


Figure 2 – Basic GNS3 network

NOTE: In the figures, you can see different methods of labeling and visual organization:

- The Red LAN posts the network ID and just the host part is next to the device
- The Blue LAN uses the complete network ID next to each device

Phase II – Connect the LANs to a Router

As discussed earlier, hubs and switches connect end devices to create LANs (an over simplistic explanation, but it suffices in this instance). Now we are going to use a router to connect the individual LANs to form an enterprise network.

1. Import the MikroTik appliance from GNS3 marketplace
2. Connect the Red and Blue LANs to a router
    2.1. Drag a MikroTik router to the workplace
    2.2. Connect a cable from the Red switch to the ***ether1*** router interface
    2.3. Connect a cable from the Blue switch to the ***ether2*** router interface

> **NOTE:** Unlike the switches, taking note of which router ports are in use is **VERY IMPORTANT**!

3. Start the router and open its console (Figure 3)

    3.1. Login: **admin**

    3.2. Password: (nothing just hit ***Enter***)

    3.3. Set a new password (ex. **Security1**)

> **NOTE:** You can change the hostname of the router for clarity. This will be useful when we build more complicated and interconnected networks later.

```
> system identity set name=new_name
```

```
[admin@MikroTik] > system identity set name=Router
[admin@Router] >
```

Figure 4 – Set new router hostname

MikroTik Router Troubleshooting

If the router is stuck in an infinite boot-loop (throttling), ensure that KVM acceleration is **DISABLED** in the GNS3 VM configuration file.

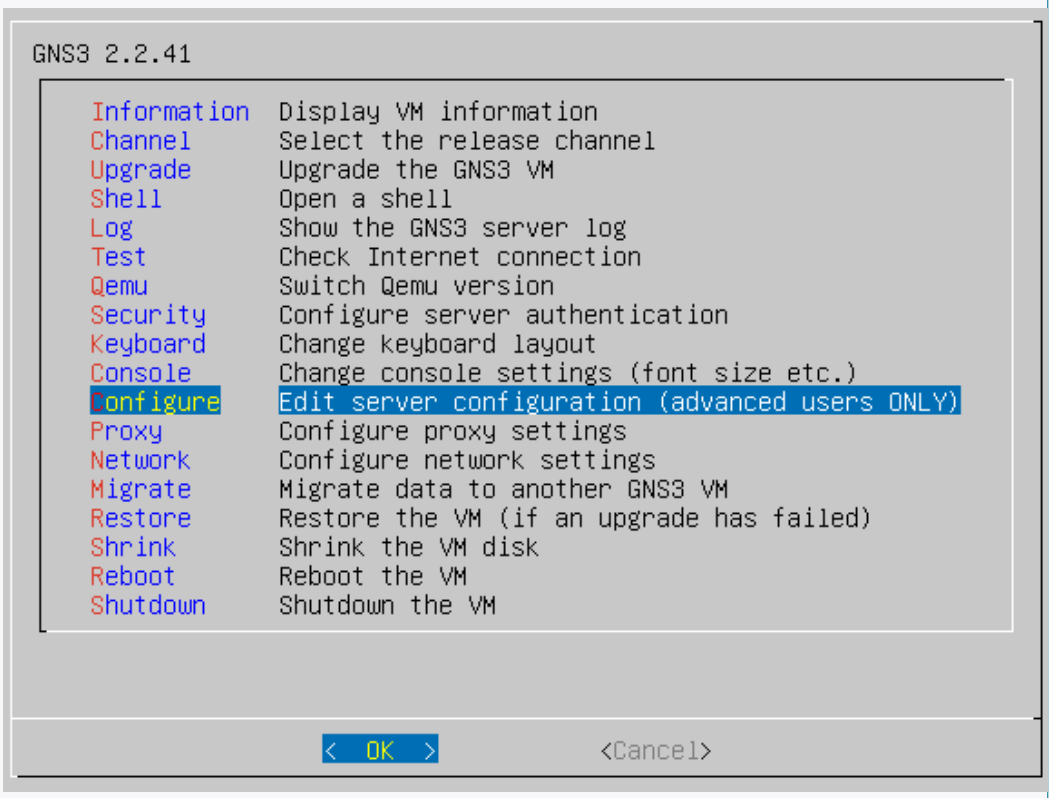


Figure 5 – GNS3 VM main menu

```
[Server]
host = 0.0.0.0
port = 80
images_path = /opt/gns3/images
projects_path = /opt/gns3/projects
report_errors = True
[Qemu]
enable_kvm = false
```

Figure 6 – GNS3 VM configuration

4. Configure ***ether1*** with a Red IP address (Figure 7)

4.1. View the list of ethernet ports on the router

```
> interface print
```

4.2. Assign the IP address of **100.10.10.150** to ether1

```
> ip address add address=100.10.10.150/24 interface=ether1
```

4.3. Ensure the IP address has been taken

```
> ip address print
```

4.4. From the MikroTik router, try pinging one of the Red PCs from its console

```
[admin@Router] > ping 100.10.10.1
  SEQ HOST                                     SIZE TTL TIME       STATUS
    0 100.10.10.1                                56  64 2ms333us
    1 100.10.10.1                                56  64 1ms603us
    2 100.10.10.1                                56  64 1ms652us
    3 100.10.10.1                                56  64 1ms726us
    4 100.10.10.1                                56  64 1ms621us
    5 100.10.10.1                                56  64 884us
    sent=6 received=6 packet-loss=0% min-rtt=884us avg-rtt=1ms636us
   max-rtt=2ms333us

[admin@Router] >
```

Figure 8 – VPCS connectivity test

> **NOTE:** Routers will ping indefinitely whereas end devices usually only ping 4 to 5 times. This continuous pinging offered by routers aids in troubleshooting efforts. To stop the pinging, make sure the MikroTik console is active and press ***Ctrl+C***.

4.5. Troubleshoot as necessary until there's connectivity between the Red PCs and ether1 on the router

5. Configure ***ether2*** with the Blue IP address of **200.20.20.250** by following step 4

6. Verify that your network looks similar to the following

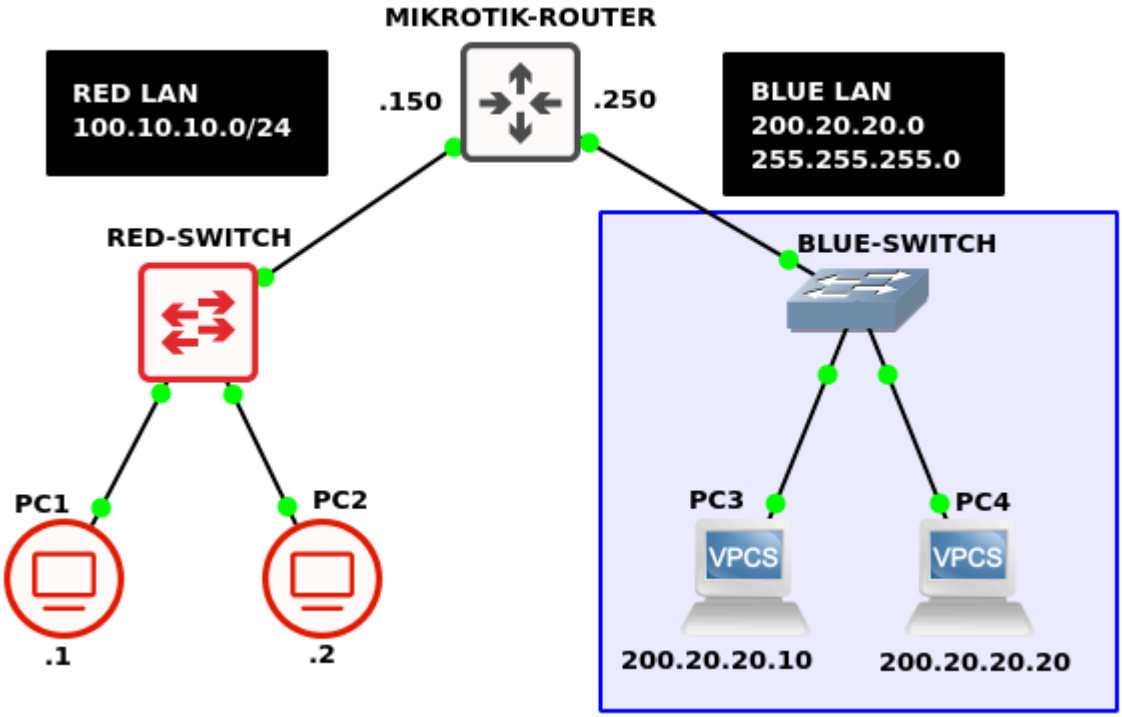


Figure 9 – Finalized GNS3 network

7. From PC4, try to ping PC1

NOTE: You will get a ***No gateway found*** error. This is expected. At the time we built our LANs, we didn't have a router. Now we need to configure our PCs to include the gateway address for their respective networks. The gateway address is the address the endpoint (ex PC1) needs to send its packets when the destination is outside of the LAN.

```
PC3> ping 100.10.10.1

No gateway found

PC3>
```

Figure 10 – No gateway found

8. Use the VPCS IP command to assign each PC a gateway address

    8.1. The gateway address for the Red network is **100.10.10.150** (router port ***ether1***)

    8.2. Configure PC1's gateway addresses

```
> ip 100.10.10.1/24 100.10.10.150
```

8.3. Configure PC2's gateway address

```
> ip 100.10.10.2/24 100.10.10.150
```

8.4. Repeat these steps to add the appropriate gateway address (**200.20.20.250**) to PC3 and PC4

Phase III – Testing your network

Testing – Testing – Testing. It never ends.

1. Open Wireshark packet capture between PC1 and the Red Switch
2. Use PC1's console to ping PC4

```
> ping 200.20.20,20
```

```
PC1> ping 200.20.20.20

84 bytes from 200.20.20.20 icmp_seq=1 ttl=63 time=1.447 ms
84 bytes from 200.20.20.20 icmp_seq=2 ttl=63 time=1.379 ms
84 bytes from 200.20.20.20 icmp_seq=3 ttl=63 time=1.563 ms
84 bytes from 200.20.20.20 icmp_seq=4 ttl=63 time=1.394 ms
84 bytes from 200.20.20.20 icmp_seq=5 ttl=63 time=1.557 ms
```

Figure 11 – PC1 pinging PC4

3. Watch Wireshark for the ARP and ICMP packets

```
100.10.10.1       200.20.20.20    ICMP   Echo (ping) request
200.20.20.20      100.10.10.1     ICMP   Echo (ping) reply
100.10.10.1       200.20.20.20    ICMP   Echo (ping) request
200.20.20.20      100.10.10.1     ICMP   Echo (ping) reply
100.10.10.1       200.20.20.20    ICMP   Echo (ping) request
200.20.20.20      100.10.10.1     ICMP   Echo (ping) reply
100.10.10.1       200.20.20.20    ICMP   Echo (ping) request
200.20.20.20      100.10.10.1     ICMP   Echo (ping) reply
0c:69:be:3b:00:00 00:50:79:66:6…  ARP    Who has 100.10.10.1?
00:50:79:66:68:00 0c:69:be:3b:0…  ARP    100.10.10.1 is at 00
```

Figure 12 – Wireshark packet capture

4. Verify that you can do this between any two points in the network

End of Lab

Deliverables

Four screenshots are required to receive credit for this exercise:

- PC1 console successfully pinging PC4
- Wireshark results (ICMP packets) of PC2 successfully pinging PC3
- Neatly labeled and organized GNS3 Workspace
- Configuration settings of the MikroTik router console (interface print, ip address print)

Homeworks

Assignment 1 – Add Green LAN to the network.

DEVICE	IP ADDRESS	NETWORK MASK	GATEWAY ADDRESS
PC5	177.50.0.1	255.255.255.0	177.50.0.250
PC6	177.50.0.2	255.255.255.0	177.50.0.250
Router Interface	177.50.0.250	255.255.255.0	none-this **is** the gateway!

Assignment 2 – Add Purple LAN to the network.

DEVICE	IP ADDRESS	NETWORK MASK	GATEWAY ADDRESS
PC7	10.10.0.1	255.255.0.0	10.10.255.250
PC8	10.10.0.2	255.255.0.0	10.10.255.250
PC9	10.10.0.3	255.255.0.0	10.10.255.250
PC10	10.10.100.1	255.255.0.0	10.10.255.250
Router Interface	10.10.255.250	255.255.0.0	none-this is the gateway!

Recommended binary grading criteria:

- Screenshot of the GNS3 Working environment where:
    - All connections are made according to instructions
    - All connections are labeled with the correct IP Address
    - Interface labels are turned on
    - Everything is organized neatly
- Screenshot of Wireshark packet captures taken from the PC5-Switch link
    - PC5 from the Green subnet can successfully ping PC2 in the Red subnet

List of Figures for Print Copy

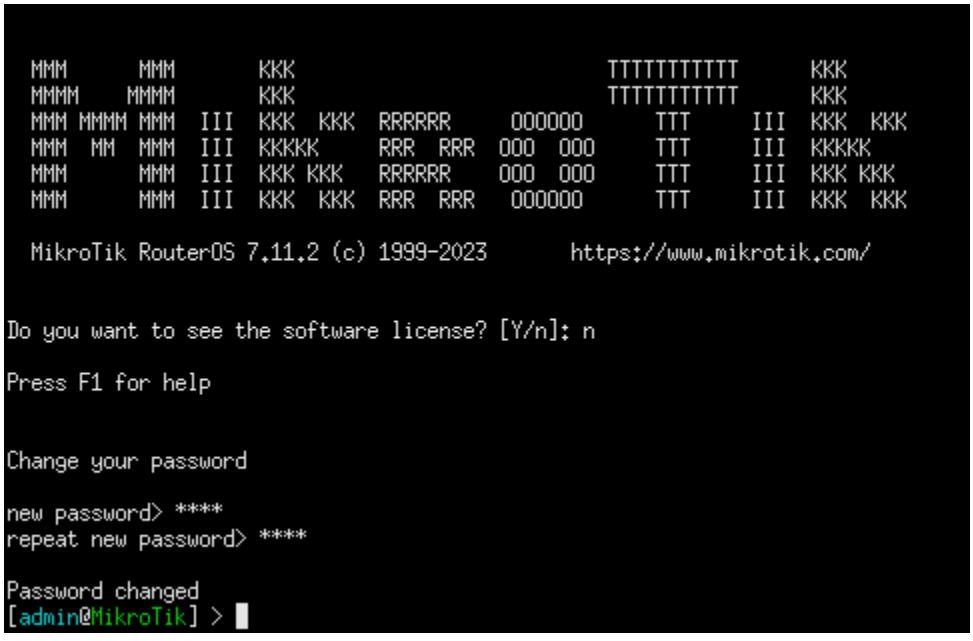


Figure 3 – MikroTik router login screen

```
[admin@Router] > interface print
Flags: R - RUNNING
Columns: NAME, TYPE, ACTUAL-MTU, MAC-ADDRESS
#   NAME    TYPE   ACTUAL-MTU  MAC-ADDRESS
0 R ether1  ether        1500  0C:69:BE:3B:00:00
1 R ether2  ether        1500  0C:69:BE:3B:00:01
2   ether3  ether        1500  0C:69:BE:3B:00:02
3   ether4  ether        1500  0C:69:BE:3B:00:03
4   ether5  ether        1500  0C:69:BE:3B:00:04
5   ether6  ether        1500  0C:69:BE:3B:00:05
6   ether7  ether        1500  0C:69:BE:3B:00:06
7   ether8  ether        1500  0C:69:BE:3B:00:07
[admin@Router] > ip address add address=100.10.10.150/24 interface=ether1
[admin@Router] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS           NETWORK      INTERFACE
0 100.10.10.150/24  100.10.10.0  ether1
```

Figure 7 – Ether1 configured with red network IP address

CHAPTER 17

IPv4 Addressing - A Very Brief Review

MATHEW J. HEATH VAN HORN, PHD

IPv4 subnetting can be confusing to many. These exercises are intended to emphasize the concept that IPv4 numbers are not numbers at all, but rather symbols of identification (i.e. names. e.g. Eileen =192.168.1.4, Hasan = 201.4.56.12, McDonalds = 10.14.67.12, etc.). When you see IP addresses, do not think numbers, think names.

Estimated time for completion: 10 minutes

LEARNING OBJECTIVES

- Increase familiarity with using the GNS3 Environment
- Identify Ethernet packet traffic on Wireshark
- Convert Decimal to Binary
- Convert Binary to Decimal
- Given an IP address and netmask, determine:
    - Network ID
    - First Usable IPv4 Address
    - Last Usable IPv4 Address
    - Broadcast IPv4 Address

PREREQUISITES

- Chapter 16 – Introduction to Routers

DELIVERABLES

- Complete the IPv4 Worksheet

RESOURCES

- IP Subnet Calculator – https://www.calculator.net/ip-subnet-calculator.html
- Heath Van Horn, Mathew, "IP Refresh", https://www.youtube.com/watch?v=Sr9gIYNpT4I

- Heath Van Horn, Mathew, "Calculating Subnet Mask", https://www.youtube.com/watch?v=wIS8SLvAGkM
- Watchguard Articles:
    - Nachreiner, Corey, "Understanding IP Addressing and Binary", https://www.watchguard.com/wgrd-resource-center/security-fundamentals/understanding-ip-addresses-and-binary
    - Farrow, Rik, "Understanding IPv4 Subnetting (Part 1)", https://www.watchguard.com/wgrd-resource-center/security-fundamentals/understanding-ipv4-subnetting-part-one
    - Farrow, Nachreiner, and Pinzon, "Understanding IPv4 Subnetting (Part 2)", https://www.watchguard.com/wgrd-resource-center/security-fundamentals/understanding-ipv4-subnetting-part-two

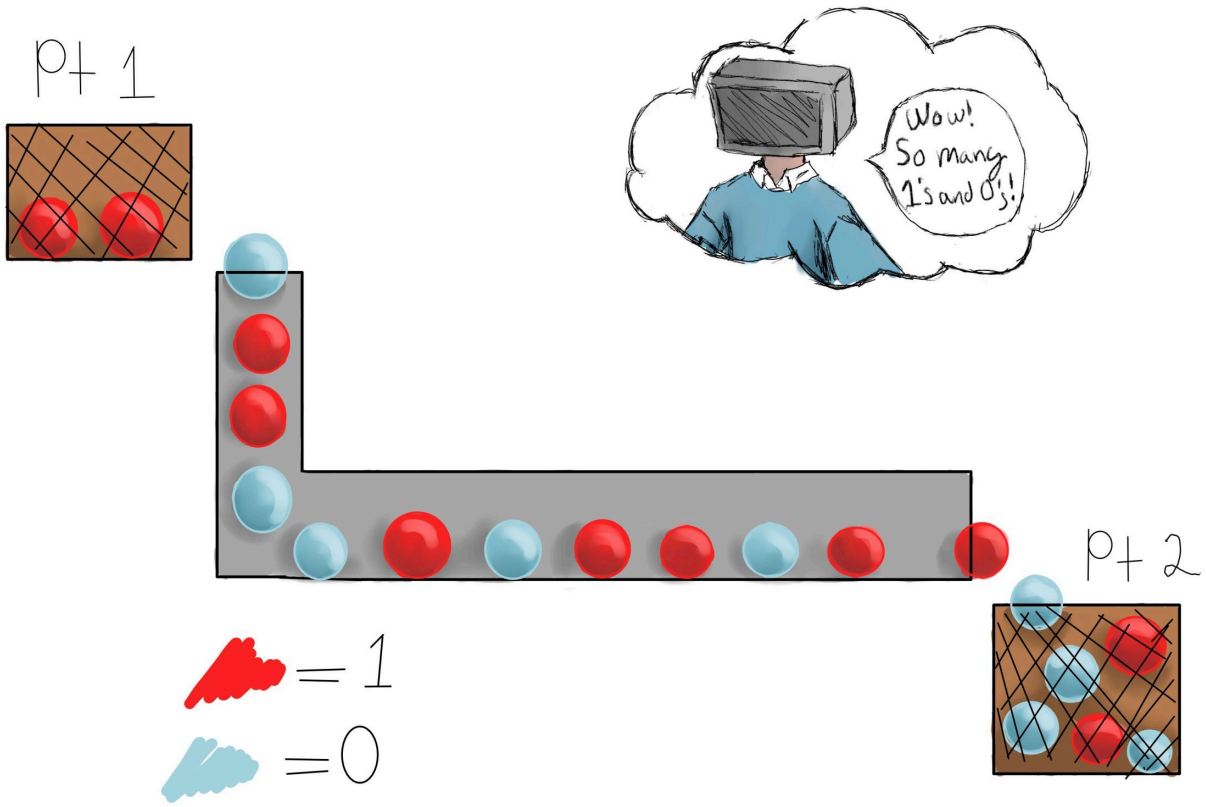
CONTRIBUTORS AND TESTERS

- Jacob M. Christensen, Cybersecurity Student, ERAU-Prescott
- Dante Rocca, Cybersecurity Student, ERAU-Prescott
- Sawyer Hansen, Cybersecurity Student, ERAU-Prescott

Phase I – A Very Brief Review

This instructional material is not designed to replace people's favorite learning materials. We want to simply augment what already exists. However, it was pointed out by some of our testers that a very abbreviated review would be a helpful inclusion within the textbook.

Computers view the world in 1s and 0s. At the most fundamental level, Network Interface Cards (NICs) produce and receive streams of 1s and 0's like ping-pong balls moving through a glass tube.



Used with permission from the artist – Romana A. Heath Van Horn

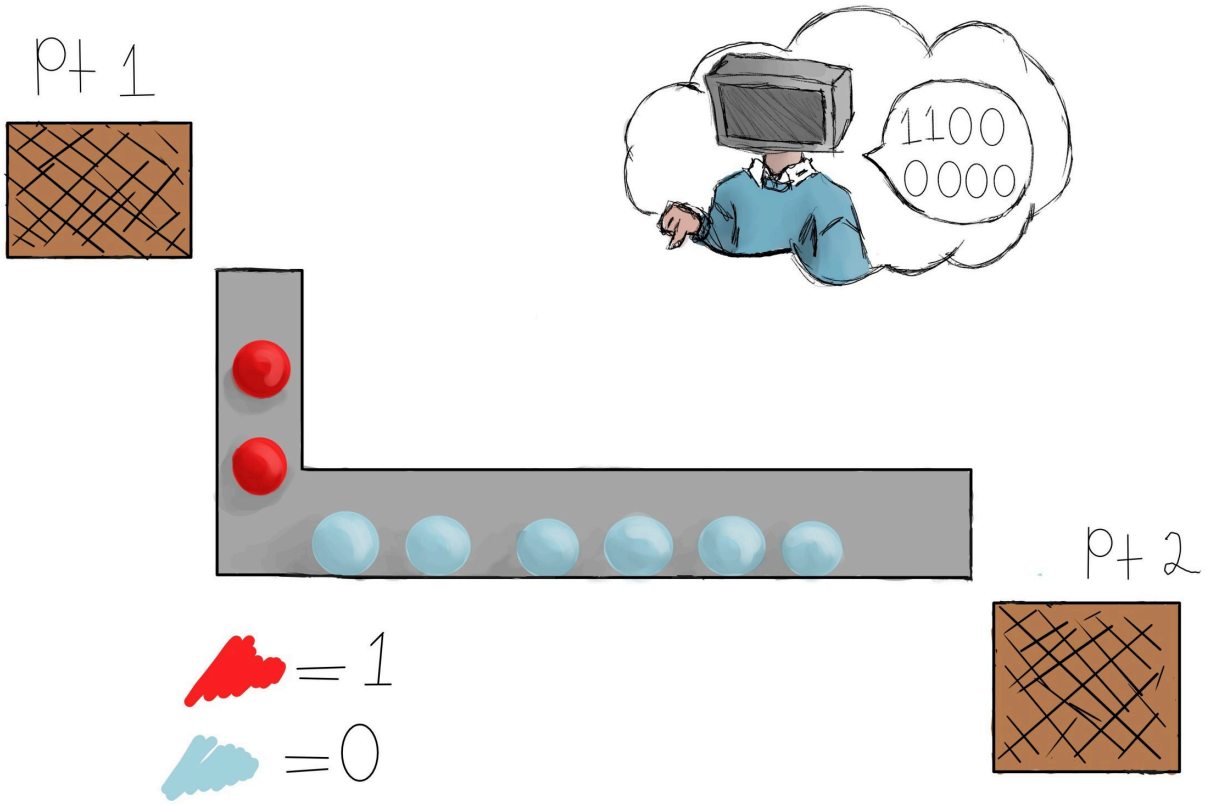
The NIC cannot see what data is coming, it can only see what has been received and it has to make sense of the information. The Ethernet Protocol defines which combinations of 1s and 0s will result in commands for action. If the combination gets mixed up, the command is garbled, and the NIC has no idea what to do and will throw the data away.

IPv4 networks use two human-readable notations of 32 bits each to provide identification of a NIC, such as 192.168.1.14 255.255.255.0. Back in the day, the first part, 192.168.1.14, was a sufficient identifier. Sort of like a house number in a town. However, as the internet grew, just like towns, further identification was needed to handle the new NICs (new houses) and yet still use the existing identification system. Subnet masks were introduced (255.255.255.0) that act sort of like street names.

Each human-readable notation of 32 bits is broken into 4 octets (8 bits). This is for human readability, the NIC doesn't care. Remember, the NIC only looks at 1s and 0s. We can look at our example (192.168.1.14) using these principles. Generally, spaces are inserted every 4 bits to make things easier for humans to read.

First Octet	Second Octet	Third Octet	Forth Octet
192	168	1	14
1100 0000	1010 1000	0000 0001	0000 1110

The 192 is a decimal, human-readable notation of the binary value 1100 0000. This means if this was sent from one NIC to another, we would send a ping-pong ball sequence of red, red, blue, blue, blue, blue, blue, blue.



Used with permission by the artist – Romana A. Heath Van Horn

Converting between binary and decimal notation is beyond the scope of this lesson. Suffice it to say, most scientific calculators can make the conversion.

Looking at this, we can quickly run out of notational identifiers. Remember, the largest decimal notation we can have is 255 because the largest binary notation is 1111 1111.

Decimal	Binary
255	1111 1111
999	0011 1110 0111 (NO! larger than 8 bits)

Subnet masks use a continuous string of 1s to identify which part of the IPv4 address is the "street name" and which part is the "house number". A subnet mask means that the sting of 1s is never interrupted. For example, if our subnet mask is 255.255.255.0 it represents a binary representation of 1111 1111.1111 1111.1111 1111.0000 0000. Notice there is no interruption of the sequence of 1s. It is impossible to have a subnet mask of 255.192.16.14 because the sequence of 1s is interrupted.

255	255	255	0	No interruption of bits
1111 1111	1111 1111	1111 1111	0000 0000	
255	192	16	14	Series of 1's bits interrupted
1111 1111	1100 0000	0001 0000	0000 1110	

The final piece of the puzzle is called Logical AND addition. In this type of addition, any binary 1 added to another binary 1 will produce a binary 1 (1+1=1). Any use of zero will result in a zero. e.g. (1+0=0, 0+1=0, 0+0=0)

When we perform a Logical AND addition of our IP address with its associated network mask it reveals both the "street name" and the "house number" of our device.

In this example, our PC has an IP address of 192.168.1.65 with a network mask of 255.255.255.0. We will apply Logical AND addition and look at the results.

First, we convert the IP address and netmask to binary.

- 192.168.1.65 → 1100 0000 . 1010 1000 . 0000 0001 . 0100 0001
- 255.255.255.0 → 1111 1111 . 1111 1111 . 1111 1111 . 0000 0000

Second, apply Logical AND addition to the first octet of both

> NOTE: the first octet in all instances is highlighted in red for illustrative purposes

192	1	1	0	0	0	0	0	0
255	+1	+1	+1	+1	+1	+1	+1	+1
Logical AND Addition (1+1=1 with all other results being 0)								
Result	1	1	0	0	0	0	0	0

Third, apply Logical AND addition to the remaining three octets

168.1.65	1010 1000	0000 0001	0100 0001
255.255.0	+1111 1111	+1111 1111	+0000 0000
Logical AND Addition (1+1=1 with all other results being 0)			
Result	1010 1000	0000 0001	0000 0000

Fourth rewrite the results into 32-bits

Results from Step 2	Results from Step 3		
1100 0000	1010 1000	0000 0001	0000 0000

> NOTE: As you get more experienced, steps two and three would be done at the same time so there would be no need for a 4th step.

Finally, convert the 32-bits into decimal, human-readable form.

1100 0000 . 1010 1000 . 0000 0001 . 0000 0000 = 192.168.1.0

The result tells us that our "Street Address" is 192.168.1.0 and that we can use any number from 1-254 in place of the 0 as our "House Humber". Yes, I see you in the back. "What happened to 255?" In this case, 255 is used as a short cut meaning "every house on the street". Think junk mail. If a company wants to send snail mail to everyone on a street, it doesn't look up every address, it just sends it to every house on the street.

Used with artist's permission: Romana A. Heath Van Horn

At this time we are going to abandon the street name and house number metaphor and use the proper names:

- Street Name = Network ID
- House Number = Host ID
- Every House = Broadcast ID

There are three more important identifiers we need to know:

- First Usable IP Address = Add 1 to the Network ID (e.g. 192.168.1.0 + 1 = 192.168.1.1)
- Last Usable IP Address = Subtract 1 from the Broadcast ID (e.g. 192.168.1.255 – 1 = 192.168.1.254)
- IP Address Range = The number of real numbers between the First Usable and Last Usable IP addresses, inclusive (e.g. 192.168.1.1 – 192.168.1.254 = 0.0.0.253 then include the 1 to increase the result to 0.0.0.254). This means 254 hosts can be joined to the network.

This is a lot of information, so let's clean it up in a summary:

Given a PC with an IP address of 192.168.16.14 and a subnet mask of 255.255.255.0 find the resulting information:

Given				
IP Address	192	168	16	14
Subnet Mask	255	255	255	0
Convert to binary				
IP Address	1100 0000	1010 1000	0001 0000	0000 1110
Subnet mask	1111 1111	1111 1111	1111 1111	0000 0000
Logical AND addition result	1100 0000	1010 1000	0001 0000	0000 0000
Convert back to human readable decimal form				
Network ID	192	168	16	0

- Network ID = 192.168.16.0
- Host ID = 192.168.16.0
- Broadcast ID = 192.168.16.255
- First Usable IP = 192.168.16.1
- Last Usable IP = 192.168.16.254
- Address Range = 254 hosts

Some will look at this and make some shortcut inferences. And some shortcuts can be made, but only under specific conditions. This is how cyber folks can impress laymen with their mental math skills. However, so long as you follow the above procedures every time, you will always get the correct answers and your network will function because it is using the correct IP addresses.

A quick discussion on Classless Inter-Domain Routing (CIDR). Using network masks of 255.0.0.0, 255.255.0.0, or 255.255.255.0 leaves many unused Host IDs that could be used by other networks. Take our example earlier, 255.255.255.0 resulted in 254 Host IP addresses. However, if we are only connecting two devices (say 192.168.16.1 and 192.168.16.2), we are preventing the use of the remaining 253 Host IP addresses by other parts of the network.

CIDR allows us more flexibility. Using a netmask of 255.255.255.252 gives us 2 usable Host IDs and frees the remaining Host ID's for other purposes. CIDR notation provides us with some shortcut tools because it just lists the number of continuous binary 1s in the netmask.

	Network Mask (netmask)				Number of 1's	CIDR notation
IP	255	255	255	0		
Binary	1111 1111	1111 1111	1111 1111	0000 0000	24 binary 1's	/24
IP	255	255	255	252		
Binary	1111 1111	1111 1111	1111 1111	1111 1100	30 binary 1's	/30

Phase II – Practice Subnetting Principles

The following problems are self-graded. You can take as many attempts as you need to ensure you understand the information.

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=244#h5p-1

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=244#h5p-2

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=244#h5p-3

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=244#h5p-4

End of Lab

Deliverables

Complete this worksheet and turn it in to receive credit for this exercise: Worksheet

Homeworks

Assignment 1 – Create your own GNS3 network

Generate a random IPv4 network ID and use it to build a GNS3 network using 4 PCs and a switch. Use a CIDR that would allow for the addition of 1,000 more hosts.

Assignment 2 – Create a GNS3 that doesn't waste any IP addresses

Generate a random IPv4 network ID and use it to build a GNS3 network using 5 PCs and a switch. Use a CIDR that wastes as few IP addresses as possible.

Recommended binary grading criteria:

- Screenshot of the GNS3 Working environment where:
  - All connections are made according to instructions
  - All connections are labeled with the correct IP Address
  - Correct CIDR is used
  - Interface labels are turned on
- A screenshot of Wireshark packet captures taken from the PC1-switch link
  - Any PC can successfully ping PC1
  - A second PC can successfully ping PC2

CHAPTER 18

Dynamic Host Configuration Protocol – Linux

JACOB CHRISTENSEN AND MATHEW J. HEATH VAN HORN, PHD

Dynamic Host Configuration Protocol (DHCP) is an interacting client-server protocol that automatically provides a host (PC, Laptop, Phone, etc) with an Internet Protocol (IP) address upon request. The purpose of this activity is for learners to see DHCP in action instead of just reading about the DHCP handshake theory. A secondary purpose is for learners to experience frustration when hosts do not get a DHCP IP address. Learners can see how the packets move and where they may get hung up while working on making DHCP function correctly on their network. Finally, learners will be able to see Address Resolution Protocol (ARP) in action. While DHCP occurs when a host requests an IP address for an existing MAC address, ARP is when a host has an IP address, but is unsure what MAC address it belongs to.

Estimated time for completion: 25 minutes

LEARNING OBJECTIVES

- Successfully deploy a DHCP solution using Linux on an enterprise network
- Capture and Observe DHCP packets using Wireshark
- Capture and Observe ARP packets using Wireshark
- Successfully add hosts to an enterprise network and receive IP addresses automatically

PREREQUISITES

- Chapter 7 – Create a Linux Server
- Chapter 15 – Hubs and Switches
- Chapter 17 – IPv4 Addressing

DELIVERABLES

- 5 Screenshots:
    - Wireshark – DHCP Packets for PC1
    - Wireshark – DHCP Packets for PC2
    - Wireshark – ARP Packets for PC1/PC2

- GNS3 Workspace
- Configuration of DHCP Daemon

RESOURCES

- Internet Systems Consortium – "ISC DHCP" – https://www.isc.org/dhcp/

CONTRIBUTORS AND TESTERS

- Julian Romano, Cybersecurity Student, ERAU-Prescott
- Dante Rocca, Cybersecurity Student, ERAU-Prescott

Phase I -Build the Network Topology

The following are steps to set up the learning environment to better understand DHCP. Since learners have performed many of these tasks in other labs, we have taken liberties to reduce the number of steps and screenshots for repeated material. If you are confused about what you've been asked to do, please review the appropriate chapter of this book.

Your final network will look like the following:

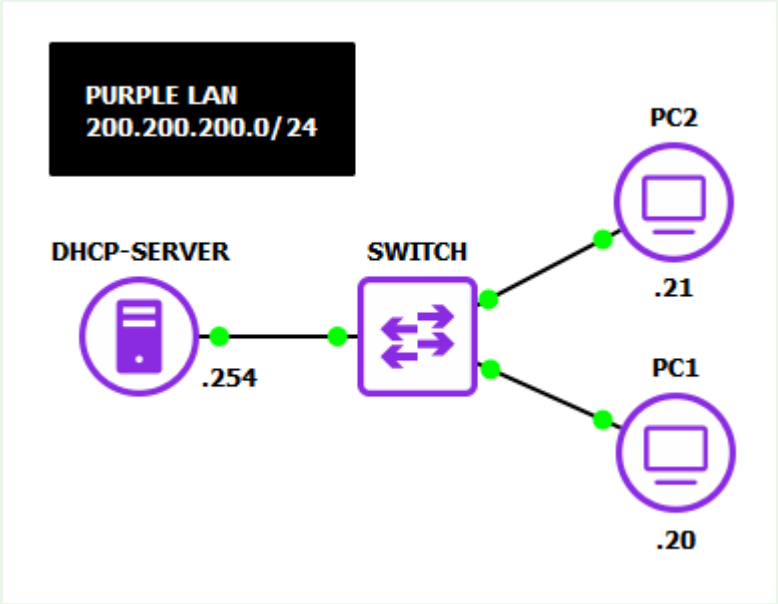


Figure 1 – Final GNS3 network environment

1. Start GNS3
    1.1. Create a new project: **LAB_04**
2. Build a new LAN with a network address of **200.200.200.0/24**

2.1. Use two ***VPCS*** devices for PC1 and PC2

2.2. Add an ***Ethernet switch***

2.3. Add an ***Ubuntu Server*** VM and rename it to “DHCP-Server”

2.4. Connect the server and PCs to the switch

2.5. Label and organize your network as necessary

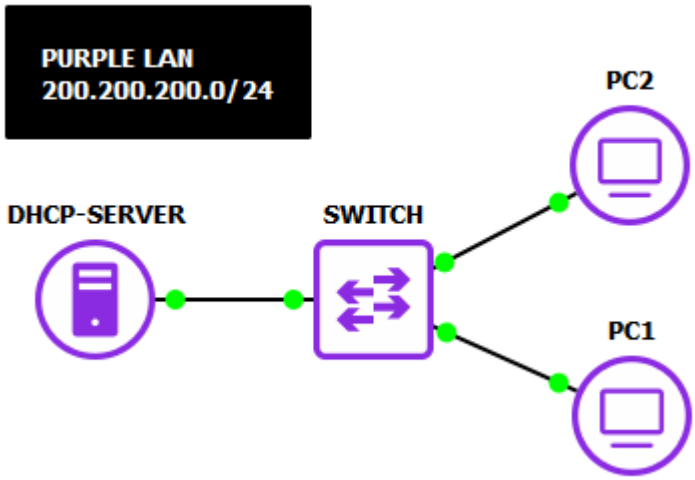


Figure 2 – Basic network topology

Phase II – Configure the DHCP Server

This lab requires that **isc-dhcp-server** is installed to your VM. Verify this with the following command. Refer to Chapter 7 for installing necessary packages from the APT repository. If you are already familiar with DHCP configuration, please feel free to use and explore other solutions too.

```
> apt list isc-dhcp-server
```

```
iako@dhcp-server:~$ apt list isc-dhcp-server
Listing... Done
isc-dhcp-server/jammy-updates,now 4.4.1-2.3ubuntu2.4 amd64 [installed]
N: There are 2 additional versions. Please use the '-a' switch to see them.
iako@dhcp-server:~$ _
```

Figure 3 – Verifying DHCP server package is installed

Ensure that yours also says “[installed]” next to the output.

UPDATE: KEA is the new Linux standard for implementing DHCP, but at the time we developed these labs, KEA documentation was lacking. We will update the labs to use KEA in future editions.

NOTE: For those who have limited computer power/resources, consider using a TinyCore VM as your primary Linux server as covered in Chapter 9.

Moving forward, managing Linux boxes will become a staple in our networks. While this might be intimidating for those who are new to CLI environments, be assured that you will quickly learn. However, it is a good idea to refresh basic terminal navigation skills online before continuing.

1. Start the DHCP Server and give it a minute or two to boot

    1.1. Login: **student**

    1.2. Password: **Security1**

NOTE: Learners unfamiliar with Linux need to know that Linux CLI will NOT move the cursor as you type the password. This is a security countermeasure to prevent shoulder-surfing. Even if an observer can't see the characters being typed in the password, just knowing the number of characters in the password makes brute-force password hacking easier. Therefore, if the cursor doesn't move, nobody can count the number of characters used in the password by watching the screen.

Making Backups

Before editing ciritcal files (such as configuration files), it is always good practice to create backups. When needed, they will save you lots of time and headache, so make it a habit now so that you don't regret it later.

Make a new backup folder in your home directory.

```
> mkdir ~/backups
```

Verify it was successfully created.

```
> ls ~/ | grep backups
```

```
iako@dhcp-server:~$ ls ~/ | grep backups
backups
iako@dhcp-server:~$ _
```

Figure 4 – Backups file created

Make a copy of a file to the backups folder (sudo may be necessary depending on the security of the file).

```
> cp /path/to/file/example.conf ~/backups/
```

Restore the file if needed.

```
> cp ~/backups/example.conf /path/to/file/
```

NOTE: obviously "/path/to/file/" and "example.conf" are placeholders. Adjust them as necessary to your situation.

2. Configure the server's interface with the static IP of **200.200.200.254** on a **/24** network

2.1. Identify the network configuration file (Figure 5)

```
> ls /etc/netplan/
```

NOTE: The netplan configuration file will always be a .YAML in this directory; however, the name may change between releases or user modification. As of writing this textbook, I am using Ubuntu 22.04.X LTS. The default configuration file name for netplan is **00-installer-config.yaml**. Adjust as necessary.

2.2. Edit the YAML file to match the example provided below

```
> sudo vi /etc/netplan/00-installer-config.yaml
```

```
# This is the network config written by 'Jake M. Christensen'
# 2024.02.07
network:
  ethernets:
    enp0s3:
      optional: true
      dhcp4: false
      addresses:
        - 200.200.200.254/24
  version: 2
```

Figure 6 – Ubuntu netplan configuration

Command	Description
enp0s3	This is the name of interface you want to configure with options indented after. This can be checked with the command***ip address show***. Adjust as necessary.
optional	Determines whether the system should wait to boot until the specified interface is configured. If you are getting an error on boot concerning "waiting for network configuration", ensure that this value is set to 'true'.
dhcp4	Determines whether the specified interface can dynamically receive IP addresses from a DHCP server.
addresses	Set static IP address value(s) to the specified interface.

NOTE: Use any preferred text editor, but it is recommended that you become familiar with Vi since it is installed by default on even the most minimal of Linux distributions.
Basic commands:
– Press ***i*** to enter***Insert*** (editor) mode.
– Press ***Esc*** to return to ***Command*** mode.
– Type ***:wq*** in Command mode to save (write) and exit back to the terminal.
– Type ***:q!*** in Command mode to exit without saving.

2.3. Apply the changes (Figure 7)

```
> netplan try
```

NOTE: Proper spacing in this file is critical. If an error is thrown, ensure that your file matches the one provided. Double and triple-check for spelling errors.

2.4. Verify the IP address of the DHCP Server was taken and that the ethernet card has a state of ***UP*** (Figure 8)

```
> ip -c addr
```

NOTE: Some testers have reported that you might not get the above information when you type ***ip addr***. Some interfaces, even virtual ones, need to think a cable is plugged in. Ensure that a cable is connected from the server to the switch.

2.5. In GNS3, add a label next to DHCP Server with its new host address

3. Modify the DHCP Server configuration file as shown below

```
> sudo vi /etc/dhcp/dhcpd.conf
```

```
# Configuration written by 'Jake M. Christensen'
# 2024.02.07

# This is the main DHCP server on this subnet
authoritative;

# Global parameters
default-lease-time 600;
max-lease-time 7200;

# This is a very basic subnet delaration
subnet 200.200.200.0 netmask 255.255.255.0 {
  range 200.200.200.20 200.200.200.250;
  option broadcast-address 200.200.200.255;
}
```

Figure 9 – DHCP daemon configuration example

> **NOTE:** You can delete most of the lines in the file or you can append it as appropriate. Your choice.

4. When editing system services, it is good practice to reload the manager with updated configurations

```
> sudo systemctl daemon-reload
```

5. Verify that the server is ***enabled*** so that it starts on boot

```
> systemctl is-enabled isc-dhcp-server
```

    5.1. Enable the service if necessary

```
> sudo systemctl enable isc-dhcp-server
```

6. Start the DHCP daemon

```
> sudo systemctl start isc-dhcp-server
```

7. Verify the server is running (type ***Q*** to quit)

```
> sudo systemctl status isc-dhcp-server
```

```
student@ubuntuserver:~$ sudo systemctl status isc-dhcp-server
● isc-dhcp-server.service - ISC DHCP IPv4 server
     Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; disabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-02-08 02:35:05 UTC; 12min ago
       Docs: man:dhcpd(8)
   Main PID: 1602 (dhcpd)
      Tasks: 4 (limit: 2221)
     Memory: 4.5M
        CPU: 10ms
     CGroup: /system.slice/isc-dhcp-server.service
             └─1602 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhcpd.pid -cf /etc/dh

Feb 08 02:35:05 ubuntuserver sh[1602]: PID file: /run/dhcp-server/dhcpd.pid
Feb 08 02:35:05 ubuntuserver dhcpd[1602]: Wrote 0 leases to leases file.
Feb 08 02:35:05 ubuntuserver sh[1602]: Wrote 0 leases to leases file.
Feb 08 02:35:05 ubuntuserver dhcpd[1602]: Listening on LPF/enp0s3/08:00:27:8a:ab:0e/200.200.200.0/24
Feb 08 02:35:05 ubuntuserver sh[1602]: Listening on LPF/enp0s3/08:00:27:8a:ab:0e/200.200.200.0/24
Feb 08 02:35:05 ubuntuserver dhcpd[1602]: Sending on   LPF/enp0s3/08:00:27:8a:ab:0e/200.200.200.0/24
Feb 08 02:35:05 ubuntuserver sh[1602]: Sending on   LPF/enp0s3/08:00:27:8a:ab:0e/200.200.200.0/24
Feb 08 02:35:05 ubuntuserver dhcpd[1602]: Sending on   Socket/fallback/fallback-net
Feb 08 02:35:05 ubuntuserver sh[1602]: Sending on   Socket/fallback/fallback-net
Feb 08 02:35:05 ubuntuserver dhcpd[1602]: Server starting service.
lines 1-21/21 (END)
```

Figure 10 – DHCP server status

NOTE: If you get an error, it is likely that there was a syntax error in your configuration file or the unit file was not updated properly. You can try restarting the service with the following command:

```
> sudo systemctl restart isc-dhcp-server
```

NOTE: If restart doesn't work, you can view the details of the failure in the logs by typing the following command

```
> journalctl -xeu isc-dhcp-server
```

Switch	Description
-x / –catalog	Provide more verbose/explanatory log and error messages.
-e / –pager-end	Jump to the end of the log file.
-u / –unit	Specify the service to view the logs messages of.

8. Congratulations! This machine is now acting as the DHCP server for any end-device client (laptop, PC, phone, etc.) that connects to the network

Phase III – Watch DHCP in Action

Remember, DHCP is a network management protocol that allows hosts to obtain IP addresses automatically upon request. It uses the User Datagram Protocol (UDP) and the server listens on port number 67 and the client listens on port 68.

- The end device (also called a client) will send out a discovery request (e.g. “Is anyone out there handing out names?”)
- The server sees the discover request and sends out a DHCP offer (e.g. “Yes, I see you, try 20.20.20.25”).
- The client will then send a request packet (e.g. “Can I really use this name?”).
- Finally, the server will send an acknowledge packet (e.g. “20.20.20.25 is all yours man”).


Hello? Who am I? I don't have an IP address!
0.0.0.0
?
?
0.0.0.0
You need a name? I have one for you!
20.20.20.25
Here!
20.20.20.25
Wow! Thanks, I love it!
20.20.20.20
20.20.2
Now you're not some Nameless Schmo!
yay
20.20.20.20
20.20.20.25

Used with artist's permission: Romana A. Heath Van Horn

1. Initialize a new Wireshark session on the PC1-Switch link

2. Start PC1, open its console, and request a new IP address

```
> ip dhcp
```

NOTE: If successful, you should see an output that looks similar to the following:

```
PC2> ip dhcp
DDORA
PC2>
```

Figure 11 – DHCP request on VPCS

If you are unable to see the ***DORA*** (discover, offer, request, acknowledge) string in the VPCS console, either the Linux box is unreachable or the DHCP daemon is offline. Go back and troubleshoot before moving forward. If all else fails, try rebooting the server or restarting GNS3 entirely.

3. Now go back to Wireshark and look at the packets captured between PC1 and the switch (Figure 12)

3.1. Filter for only ***DHCP*** packets

dhcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover
3	1.000127	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover
4	1.001240	200.200.200.254	200.200.200.20	DHCP	342	DHCP Offer
5	4.000299	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request
6	4.031957	200.200.200.254	200.200.200.20	DHCP	342	DHCP ACK

Figure 13 – Wireshark DHCP handshake

3.2. **[DHCP Discover]**

Look at the Discover packets. There is at least 1, but there could be more depending on lag. In this example, we see on Wireshark that someone on the network basically says "HELLLOOOO! I don't know who I am (0.0.0.0). Is there anyone out there handing out IP addresses? Please speak to me!"

– Source: 0.0.0.0 (Who am I?)
– Destination: 255.255.255.255 (Who's out there?)
– MAC Address: 00:50:79:66:68:00 (This is what my face looks like)

3.3. **[DHCP Offer]**

> Now look at the DHCP offer packets. It's like our DHCP server (200.200.200.254) is saying, "Hey buddy I'm here, and if you need a name you can use this one (200.200.200.###)."

– Source: 200.200.200.254 (I'm here)
– Destination: 200.200.200.20 (Here's a name that's available)

3.4. **[DHCP Request]**

> Then the next packet should be our no-name PC saying "Oh, me me me, I like the name 200.200.200.###" in a DHCP request packet.

– Source: 0.0.0.0 (I still don't have an official name yet)
– Destination: 255.255.255.255 (If you're still out there I want that name)

3.5. **[DHCP ACK]**

> Finally, our server acknowledges PC1s eagerness and says, "Ya buddy you are now 200.200.200.### from now on and not just some schmo (0.0.0.0) who looks like 00:50:79:66:68:00."

– 200.200.200.254 (I'm still here)
– 200.200.200.20 (You're officially known as host .20 on this network)

4. Repeat the above steps to assign PC2 its own IP address

5. Return to GNS3 and update the VPCS labels with new IP addresses they were assigned

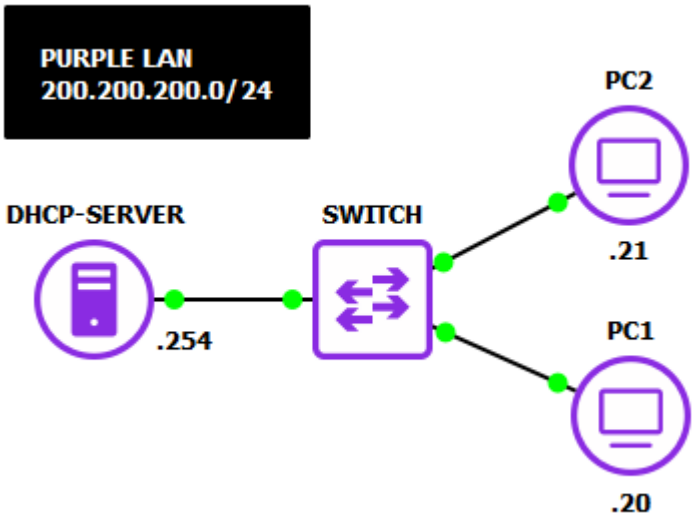


Figure 14 – GNS3 network labeled

Phase IV – Watch ARP in Use

Remember, network packets are passed by MAC addresses in Layer 2 of the OSI Model. Since we manually configured the static IP addresses on both the router and the server, the network doesn't know which IPv4 address goes to which NIC on the individual hosts. The networks now see each other and say, "Who are you?" to each other. In this example, we can see 200.200.200.254 (our DHCP server) asking who has an IPv4 address of 200.200.200.1 (our server) and vice versa. You can see that 200.200.200.254 responds and says "My Layer 2 name is 08:00:27:b2:50:bc". Don't worry about the other packets you see currently. Just get used to what ARP looks like so you can understand it when you see it again in the future.

5.1. Return to the PC1-Switch Wireshark capture window and filter for ***ARP*** packets

arp

Source	Destination	Protocol	Info
00:50:79:66:68:01	ff:ff:ff:ff:ff:ff	ARP	Gratuitous ARP for 200.200.200.20 (Request)
08:00:27:2a:17:11	00:50:79:66:68:01	ARP	Who has 200.200.200.20? Tell 200.200.200.254
00:50:79:66:68:01	08:00:27:2a:17:11	ARP	200.200.200.20 is at 00:50:79:66:68:01
00:50:79:66:68:01	ff:ff:ff:ff:ff:ff	ARP	Gratuitous ARP for 200.200.200.20 (Request)
00:50:79:66:68:01	ff:ff:ff:ff:ff:ff	ARP	Gratuitous ARP for 200.200.200.20 (Request)

Figure 15 – ARP Packet Capture using Wireshark

5.1.1. **[Gratuitous ARP]**

After a PC receives an IP address from the subnet's DHCP server, it may broadcast (ff:ff:ff:ff:ff:ff) to all devices on the network that its MAC address (00:50:79:66:68:01) now

belongs to a specific layer 3 address (200.200.200.20). This is known as a “gratuitous ARP response”, since it was sent unprompted by any specific ARP request.

– Source: 00:50:79:66:68:01 (This is my face)
– Destination: ff:ff:ff:ff:ff:ff (Hey everyone who can hear me!)
– IP Address: 200.200.200.20 (This is my name now!)

5.1.2. **[Who has?]**

If a device wants to know who currently owns an IP address, it may send an ARP request to specific MAC address or broadcast the message to all devices. In this example, the DCHP server (08:00:27:2a:17:11) wants to know if PC1 (00:50:79:66:68:01) still holds the IP address 200.200.200.20. This is known as an ARP request.

– Source: 08:00:27:2a:17:11 (This is my face)
– Destination: 00:50:79:66:68:01 (Hey you there)
– Sender IP Address: 200.200.200.254 (This is my name)
– Target IP Address: 200.200.200.20 (Is this your name?)

5.1.3. **[ARP Reply]**

In response to an ARP request packet, the device that hold the target IP will respond with an ARP relply.

– Sender IP Address: 200.200.200.20 (This is my name...)
– Source: 00:50:79:66:68:01 (... associated with this face)

5.2. Use this opportunity to analyze Wireshark packets and get comfortable with how DHCP and ARP work on a live network

End of Lab

Deliverables

5 screenshots are needed to receive credit for this exercise:

- Wireshark – DHCP Packets for PC1
- Wireshark – DHCP Packets for PC2
- Wireshark – ARP Packets for PC1/PC2

- GNS3 Workspace
- Configuration of DHCP Daemon

Homeworks

Assignment 1 – Combined network traffic watching

- Turn off all devices
- Replace the switch with a hub and reconnect all devices
- Monitor any of the PCs with Wireshark and capture ARP, DHCP, and ICMP packets for each PC's as you turn devices back on
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 environment with everything labeled
    - Screenshot of Server-PC1 ARP from PC2-Hub link
    - Screenshot of Server-PC1 DHCP from PC2-Hub link
    - Screenshot of Server-PC1 ICMPfrom PC2-Hub link

Assignment 2 – Reconfigure the DHCP server

- Figure out the number of devices that can be attached to the switch
- Generate a random IP address and choose a subnet that will allow the use of all the switch connections with as few wasted IP addresses as possible
- Reconfigure the network to use these new network addresses
- Reconfigure the DHCP settings to issue IPv4 address in this new space
- RECOMMENDED GRADING CRITERIA
    - Screenshot of the DHCP configuration file
    - Screenshot of the GNS3 workspace
    - Screenshot of DHCP of one PC
    - Screenshot of ICMP of one PC

List of Figures for Print Copy

```
student@ubuntuserver:~$ ls /etc/netplan/
00-installer-config.yaml
student@ubuntuserver:~$
```

Figure 5 – Identify network configuration file

```
student@ubuntuserver:~$ sudo netplan try
[sudo] password for student:
Do you want to keep these settings?


Press ENTER before the timeout to accept the new configuration


Changes will revert in 119 seconds
Configuration accepted.
student@ubuntuserver:~$ _
```

Figure 7 – Apply changes to the network configuration

```
student@ubuntuserver:~$ ip -c addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:8a:ab:0e brd ff:ff:ff:ff:ff:ff
    inet 200.200.200.254/24 brd 200.200.200.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe8a:ab0e/64 scope link
       valid_lft forever preferred_lft forever
student@ubuntuserver:~$ _
```

Figure 8 – Verify changes took place

14	130.891592	::	ff02::2	ICMPv6	62	Router Solicitation
15	157.486050	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover - Transaction ID 0x7bb51203
16	157.487812	PcsCompu_b2:50:bc	Broadcast	ARP	60	Who has 200.200.200.20? Tell 200.200.200.254
17	158.486894	0.0.0.0	255.255.255.255	DHCP	406	DHCP Discover - Transaction ID 0x7bb51203
18	158.489827	200.200.200.254	200.200.200.20	DHCP	342	DHCP Offer - Transaction ID 0x7bb51203
19	158.505682	PcsCompu_b2:50:bc	Broadcast	ARP	60	Who has 200.200.200.20? Tell 200.200.200.254
20	159.529598	PcsCompu_b2:50:bc	Broadcast	ARP	60	Who has 200.200.200.20? Tell 200.200.200.254
21	161.488586	0.0.0.0	255.255.255.255	DHCP	406	DHCP Request - Transaction ID 0x7bb51203
22	161.501419	200.200.200.254	200.200.200.20	DHCP	342	DHCP ACK - Transaction ID 0x7bb51203

Figure 12 – Wireshark capture

CHAPTER 19

Dynamic Host Configuration Protocol - Windows

RAECHEL FERGUSON

Dynamic Host Configuration Protocol (DHCP) is an interacting client-server protocol that automatically provides a host (PC, Laptop, Phone, etc) with an Internet Protocol (IP) address upon request. The purpose of this activity is for learners to see DHCP in action instead of just reading about the DHCP handshake theory. A secondary purpose is for learners to experience frustration when hosts do not get a DHCP IP address. Learners can see how the packets move and where they may get hung up while working on making DHCP function correctly on their network. Finally, learners will be able to see Address Resolution Protocol (ARP) in action. While DHCP occurs when a host requests an IP address for an existing MAC address, ARP is when a host has an IP address, but is unsure what MAC address it belongs to.

Estimated time for completion: 20 minutes

LEARNING OBJECTIVES

- Successfully deploy a DHCP solution using Windows on an enterprise network
- Capture and Observe DHCP packets using Wireshark
- Capture and Observe ARP packets using Wireshark
- Successfully add hosts to an enterprise network and receive IP addresses automatically

PREREQUISITES

- Chapter 6 – Adding a Virtual Machine to GNS3
- Chapter 8 – Create a Windows Server
- Chapter 17 – IPv4 Addressing

DELIVERABLES

- 4 Screenshots:
    - Wireshark – DHCP Packets for PC2
    - Wireshark – DHCP Packets for PC3
    - GNS3 Workspace

- Configuration of Windows Server

RESOURCES

- **NOTE: Each source will referenced with its corresponding number in superscript (EX: [1]) at the end of a step**
- 1. MSFT WebCast. "Basic Configuration Tasks in Windows Server 2019." YouTube, January 25, 2019. https://www.youtube.com/watch?v=1nxYJSV7-u8&list=PLUZTRmXEpBy32NP6z_qvVBOTWUzdTZVHt&index=5.
- 2. MSFT WebCast. "Install and Configure DHCP Server in Windows Server 2019 Step by Step Guide." YouTube, February 3, 2019. https://www.youtube.com/watch?v=fUK6d3s1Im4&t=414s.

CONTRIBUTORS AND TESTERS

- Mathew J. Heath Van Horn, PhD, ERAU-Prescott
- Julian Romano, Cybersecurity Student, ERAU-Prescott
- Dante Rocca, Cybersecurity Student, ERAU-Prescott
- Jacob M. Christensen, Cybersecurity Student, ERAU-Prescott

Phase I -Build the Network Topology

The following steps are to create the baseline for completing the lab. It makes assumptions about learner knowledge from completing previous labs. By the end of this lab, your network should look like the following:

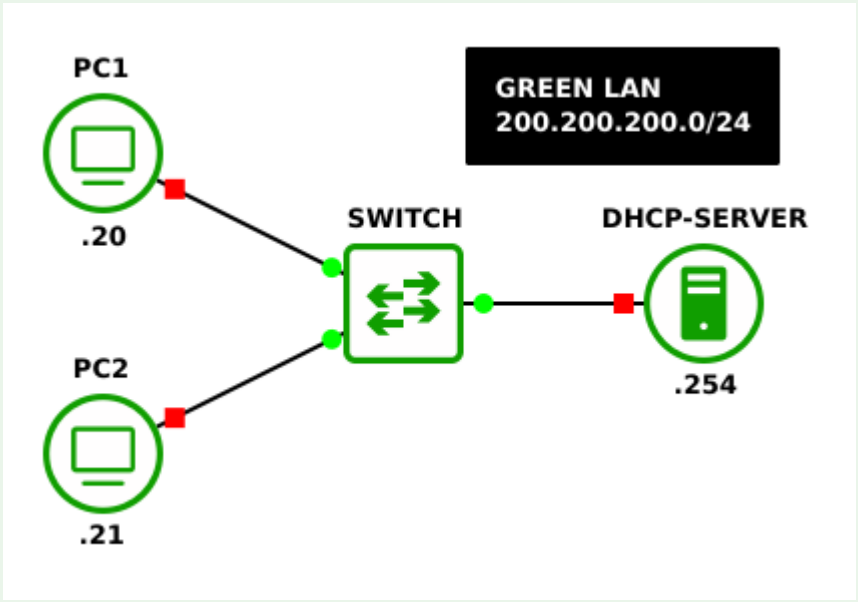


Figure 1 – Final GNS3 network environment

1. Open GNS3 and start a new workspace

1.1. Create a new project: **LAB_05**

2. Create the baseline network with an address space of **200.200.200.0/24**

> **NOTE:** This example uses the same network topology that was created in the previous chapter: DHCP using Linux.

2.1. Use two ***VPCS*** devices for PC1 and PC2

2.2. Add an ***Ethernet switch***

2.3. Add a ***Windows Server*** VM and rename it to "DHCP-server"

> **NOTE:** Ensure that the ***Allow GNS3 to use any configured VirtualBox adapter*** check box is selected for all VMs added to GNS3. Refer to step 6.2 in Chapter 11 for more information.

2.4. Connect the server and PCs to the switch

2.5. Label and organize your network as necessary

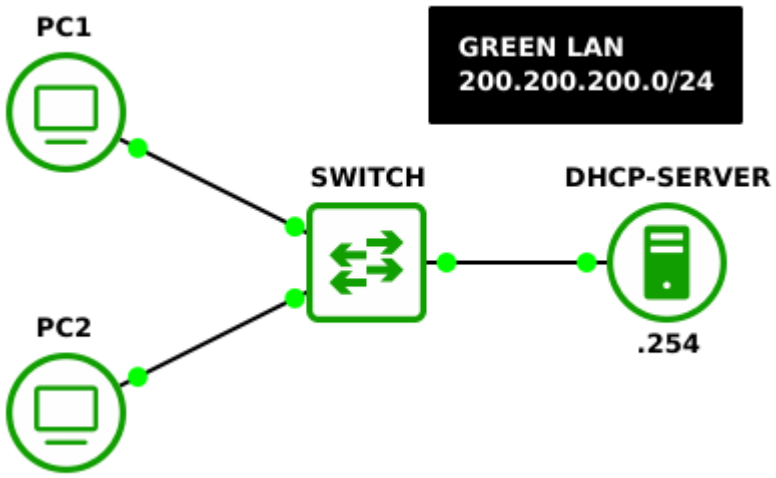


Figure 2 – GNS3 network environment

Phase II – Configure the Network Interface of the Server

In order for the server to act as a DHCP server we will need the server to be properly attached to the network.

Without giving it the proper network information our server will be unable to talk with any other computers on the network.

1. Start the server and login

2. Focus on the ***Server Manager*** dashboard window [1]

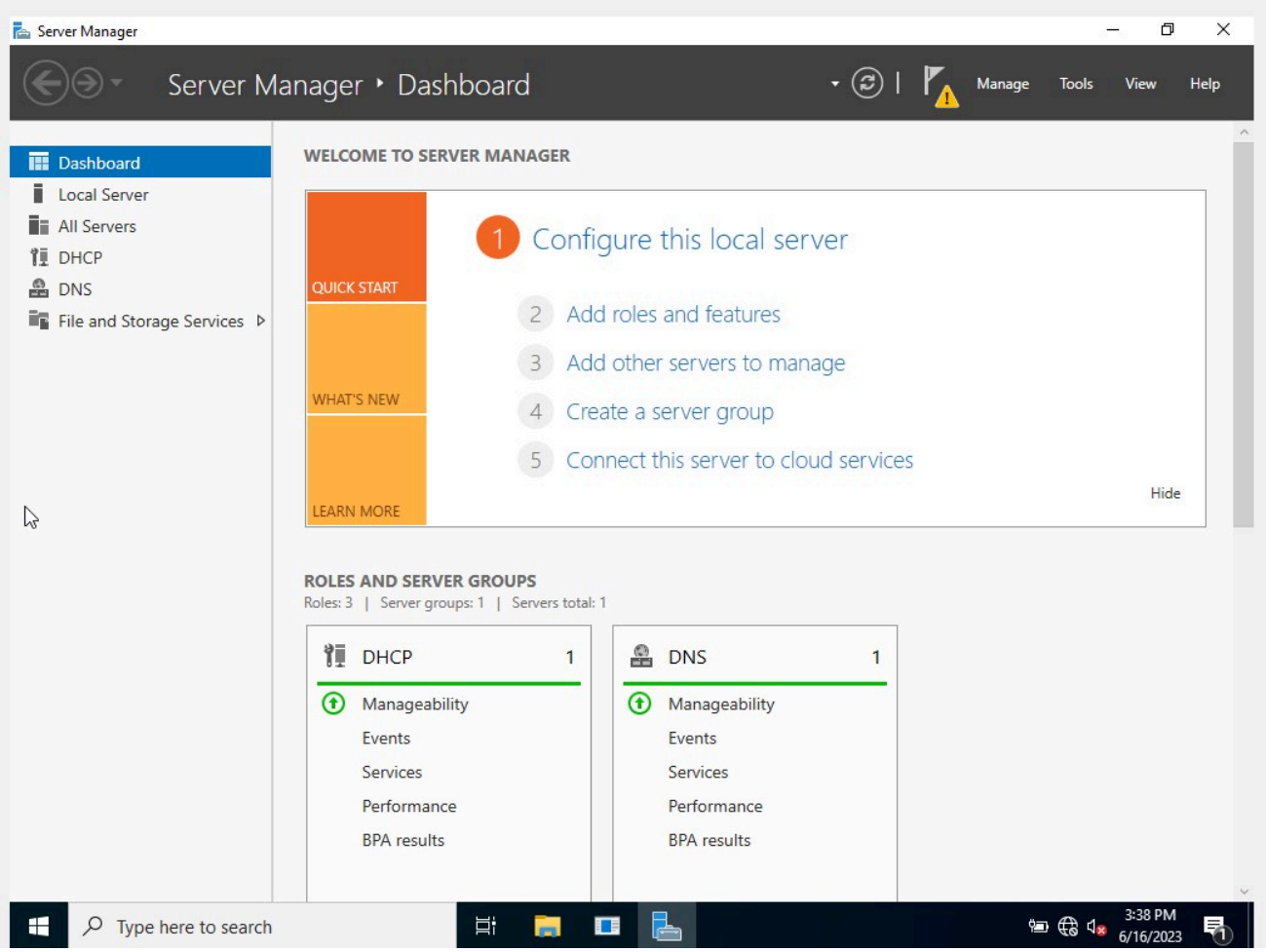


Figure 3 – Windows Server Manager Dashboard

NOTE: If the Server Manager does not start on boot, can open it via the Windows Start menu.

3. On the left side of the page, select the ***Local Server*** option

4. Assign the IP address **200.200.200.254** to the local ethernet interface

    4.1. Under the ***PROPERTIES*** table,***left-click*** on the Ethernet option (Figure 4) [1]

> **NOTE:** This is located beneath *NIC Teaming* and above *Operating system version*.

4.2. ***Right-click*** on the network interface you want to use (e.g. "Ethernet") and select ***Properties*** in the context menu (Figure 5) [1]

4.3. A new sub-window labeled ***Ethernet Properties*** should appear

4.3.1. Uncheck ***Internet Protocol Version 6 (TCP/IPv6)*** [1]

4.3.2. Check and highlight the ***Internet Protocol Version 4 (TCP/IPv4)*** option [1]

4.3.3. On the bottom right corner, click on ***Properties*** (Figure 6) [1]

4.4. A new sub-window labeled ***Internet Protocol Version 4 (TCP/IPv4) Properties*** should appear (Figure 7) [1]

4.4.1. Select ***Use the following IP address***

4.4.1.1. Enter **200.200.200.254** as the **IP address** [1]

4.4.1.2. Enter **255.255.255.0** as the **Subnet mask** [1]

4.4.1.3. Enter **200.200.200.1** as the **Default gateway** [1]

4.4.2. Select ***Use the following DNS server addresses***

4.4.2.1. Enter **200.200.200.254** as the **Preferred DNS server** [1]

4.4.2.2. Leave the **Alternate DNS server** blank [1]

4.4.3. Click ***OK*** to apply these settings [1]

4.4.4. Click ***Close*** to return to the Network Connections window

4.5. Close the Network Connections window

5. Restart Windows Server

Phase III – Setup the DHCP Server

Windows Server is capable of many functions and is very customizable. Therefore, to minimize installation times, Windows Server on initial installation does not come with many features activated. We will need to activate and configure Windows Active Directory and DHCP services.

1. Focus on the ***Server Manager*** dashboard window [2]
2. In the top right-hand corner of the screen, select ***Manage*** [2]
3. Select ***Add Roles and Features*** from the context menu [2]
4. A popup window labeled ***Add Roles and Features Wizard*** will open (Figure 8)
    - 4.1. **Before you Begin** – Click ***Next*** [2]
    - 4.2. **Installation Type** – Select the ***Role-Based*** option – then click ***Next*** (Figure 9) [2]
    - 4.3. **Server Selection** – Select your local server (this should be the only option) then click ***Next*** (Figure 10) [2]
    - 4.4. **Server Roles** – select ***DHCP Server*** [2]
        - 4.4.1. POP-UP – **Add Features** – Select ***Add Features*** towards the bottom of the new screen (Figure 11) [2]
        - 4.4.2. Return back to **Server Roles** – Click ***Next*** [2]
    - 4.5. **Features** – Click ***Next*** [2]
    - 4.6. **DHCP Server** – Click ***Next*** [2]
    - 4.7. **Confirmation** – Click ***Install*** [2]
    - 4.8. Once installation is complete click on the blue text that states ***Complete DHCP Configuration*** (Figure 12) [2]
5. POP-UP – ***DHCP Post-Install Configuration Wizard***
    - 5.1. **Description** – Click ***Next*** [2]
    - 5.2. **Authorization** – Click ***Commit*** [2]

NOTE: The default authorization should be set to use Administrator credentials. If it doesn't, you've done something wrong and need to restart.

5.3. **Summary** – Click *Close* [2]

6. Return to *Rolls and Features Wizard*

6.1. Results – Click *Close* [2]

7. Restart Windows Server

Phase IV – Configure DHCP

DHCP is widely flexible. In this lab, we are going to assign a range of IP addresses that can be used by end devices connecting to our network without a manually configured IP address.

1. Focus on the *Server Manager* dashboard window
2. In the top right-hand corner of the screen, select *Tools* [2]
3. From the drop-down menu, select *DHCP* [2]
4. A new sub-window labeled *DHCP* should appear

4.1. Expand the local computer forest (Figure 13) [2]

4.2. Right-click on the *IPv4* option and select *New Scope* from the context menu (Figure 14) [2]

5. The *New Scope Wizard* window will open

5.1. **Scope Wizard Welcome** – Click *Next* (Figure 15) [2]

5.2. **Scope Name** (Figure 16)

5.2.1. Add a name such as Scope1 [2]

5.2.2. The description field can be left blank

5.2.3. Click *Next* [2]

5.3. **IP Address Range** (Figure 17) [2]

5.3.1. Start IP address – **200.200.200.20**

5.3.2. End IP address – **200.200.200.250**

5.3.3. Length – **24** [2]

5.3.4. Subnet mask – **255.255.255.0** [2]

5.3.5. Click ***Next***

5.4. **Add Exclusion and Delay** – Click ***Next*** [2]

5.5. **Lease Duration** (Figure 18) [2]

5.5.1. Limited to 8 hours [2]

5.5.2. Click ***Next***

5.6. **Configure DHCP** [2]

5.6.1. Select ***Yes***

5.6.2. Click ***Next***

5.7. **Router (Default Gateway)** (Figure 19) [2]

5.7.1. IP address – **200.200.200.1**

5.7.2. Click ***Add***

5.7.3. Click ***Next***

5.8. **Domain Name and DNS Servers** – Click ***Next*** [2]

5.9. **WINS Servers** – Click ***Next*** [2]

5.10. **Activate Scope**

5.10.1. Click ***Yes*** [2]

5.10.2. Click ***Next*** [2]

5.11. Click ***Finish***

6. Notice that ***Scope*** now appears under the DHCP>machine_name>IPv4 tree (Figure 20)

7. Close the DHCP window

Phase V – Watch DHCP in Action

This lab is an opportunity for you to see these activities in action without the chaff that exists on an existing enterprise network.

- DHCP automatically assigns an IP address to interfaces requesting one.
- ARP is for interfaces that already have an IP address, but the interface needs to tell all of the other interfaces on the network.

1. Navigate back to GNS3

2. Start a Wireshark capture on the Server-Switch link

3. Start PC1 and open its console

    3.1. Request a new IP address

```
> ip dhcp
```

    3.2. Notice the four main DHCP packets being exchanged in Wireshark that were discussed in the previous chapter: Discover, Offer, Request, Accept

4. Start PC2 and open its console

    4.1. Request a new IP address

```
> ip dhcp
```

5. Ensure that PC1 is able to ping PC2

6. Congratulations! You now know how to configure a basic DHCP server on both Linux and Windows machines

End of Lab

Deliverables

4 screenshots are needed to receive credit for this exercise:

- Wireshark – DHCP Packets for PC2
- Wireshark – DHCP Packets for PC3
- GNS3 Workspace with 2 PCs, switch, and DHCP server – all devices labeled with their IP addresses
- Configuration settings of Windows Server DHCP

Homeworks

Assignment 1 – Combined network traffic watching

- Turn off all devices
- Replace the switch with a hub and reconnect all devices
- Monitor any of the PCs with Wireshark and capture ARP, DHCP, and ICMP packets for all three PC's as you turn devices back on
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 environment with everything labeled
    - Screenshot of DHCP for one PC
    - Screenshot of ICMP for one PC

Assignment 2 – Reconfigure the DHCP server

- Figure out the number of devices that can be attached to the switch
- Generate a random IP address and choose a subnet that will allow the use of all the switch connections with as few wasted IP addresses as possible
- Reconfigure the network to use these new network addresses
- Reconfigure the DHCP settings to issue IPv4 address in this new space
- RECOMMENDED GRADING CRITERIA
    - Screenshot of the DHCP configuration
    - Screenshot of the GNS3 workspace
    - Screenshot of DHCP of one PC
    - Screenshot of ICMP of one PC

Figures for Printed Version

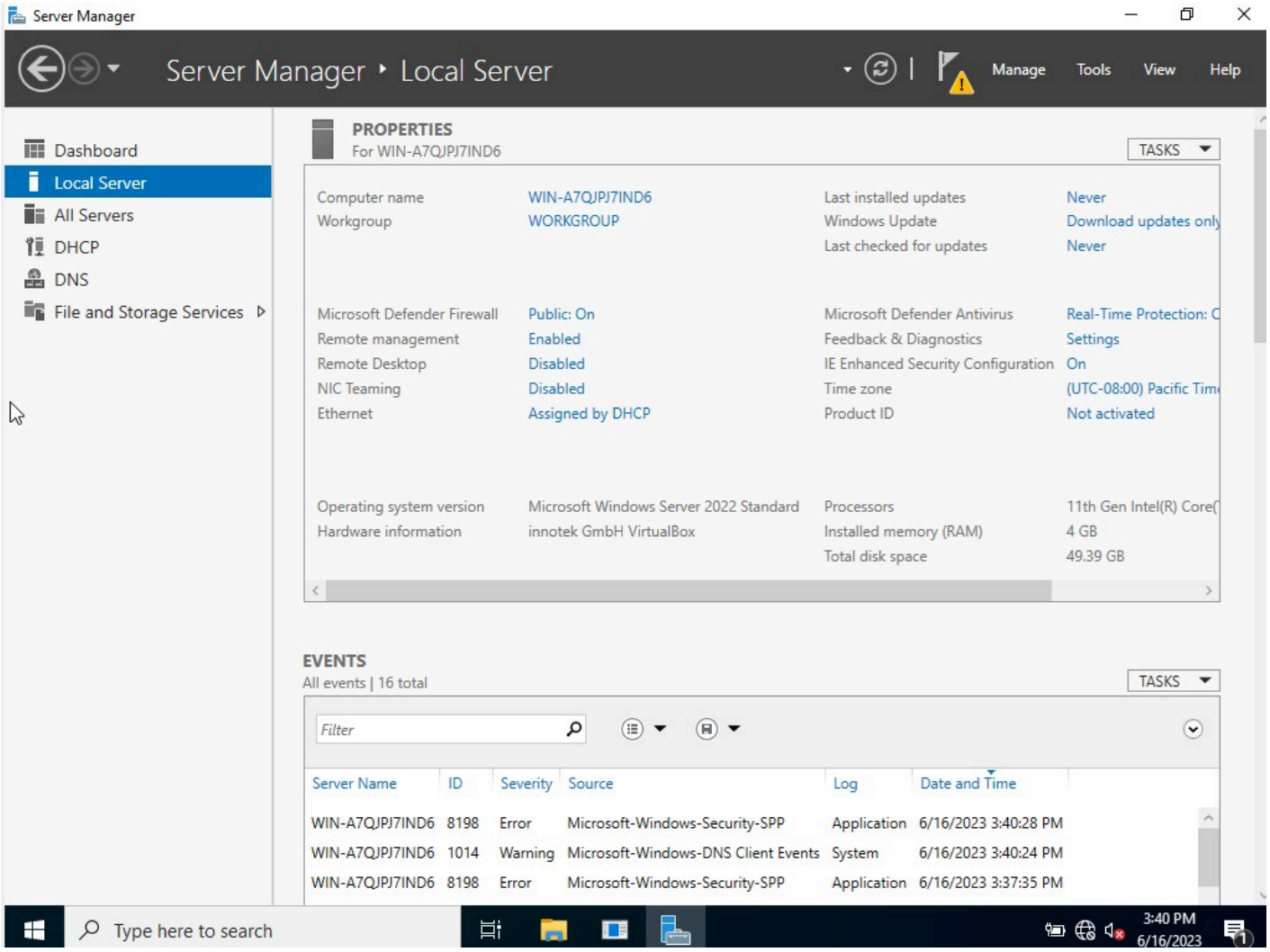


Figure 4 – Select the ethernet option

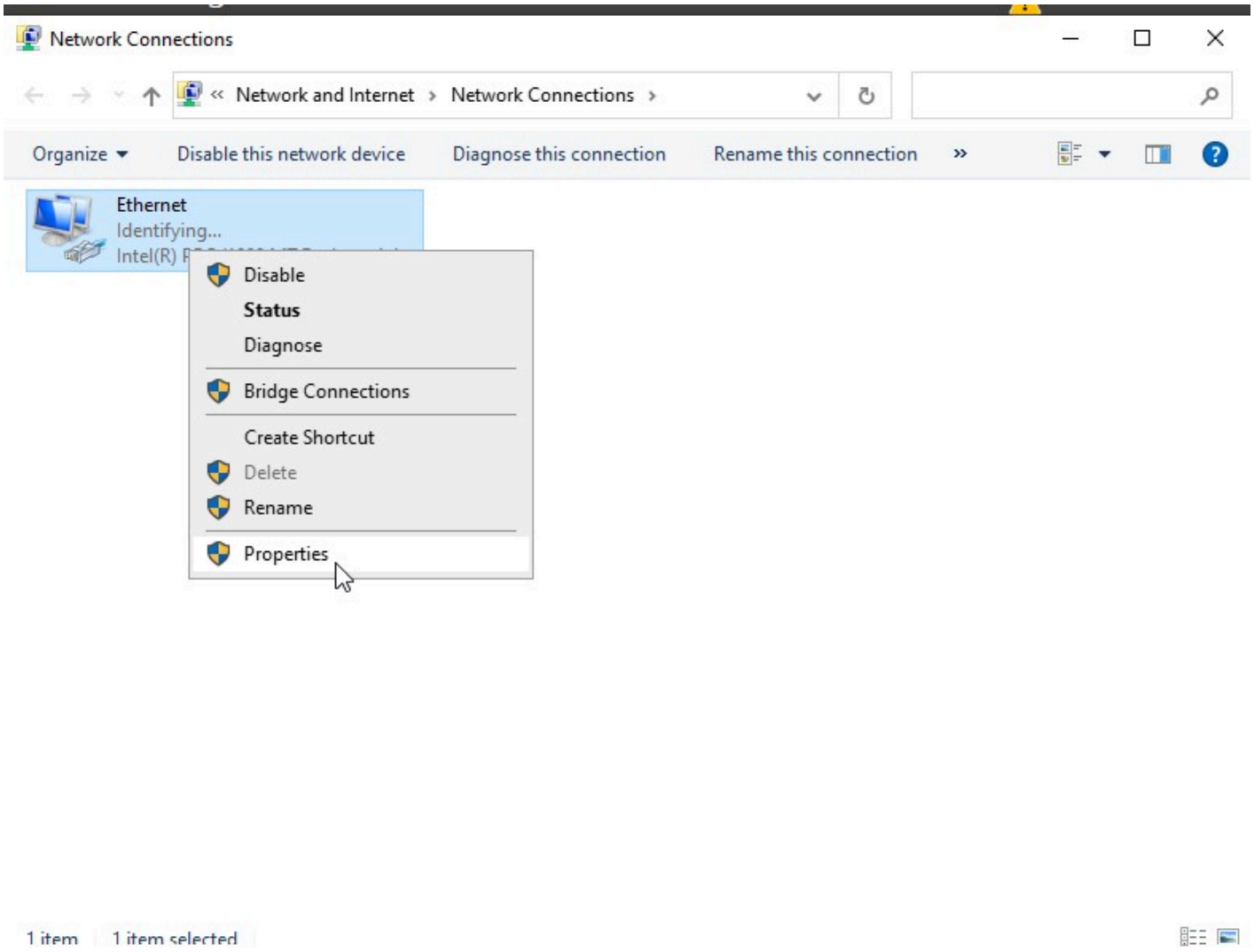


Figure 5 – Network Connections window

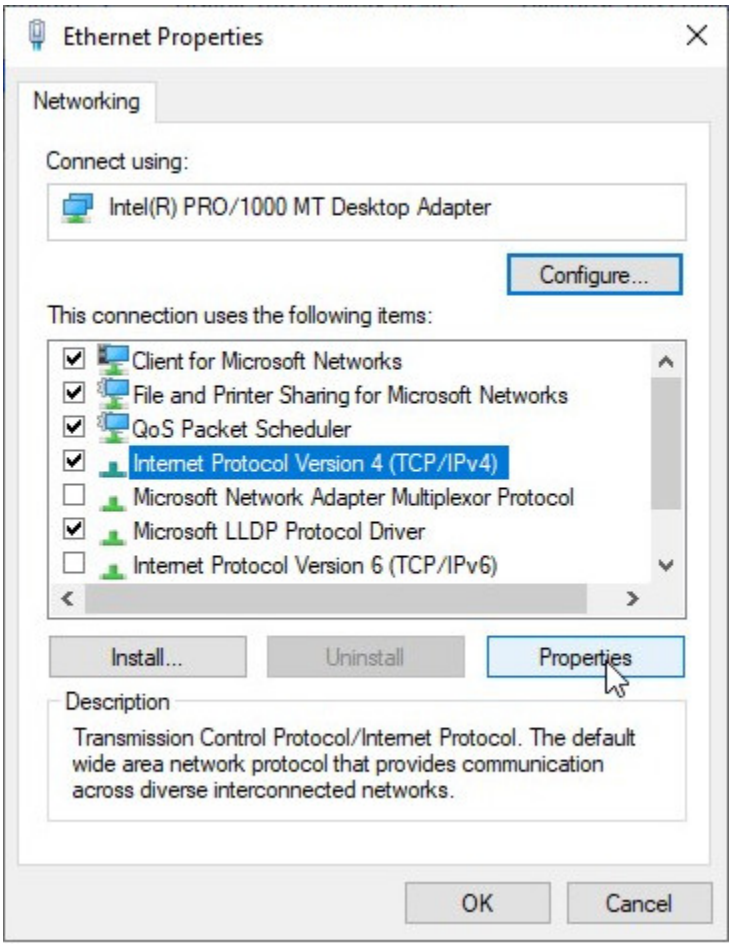


Figure 6 – Ethernet Properties window

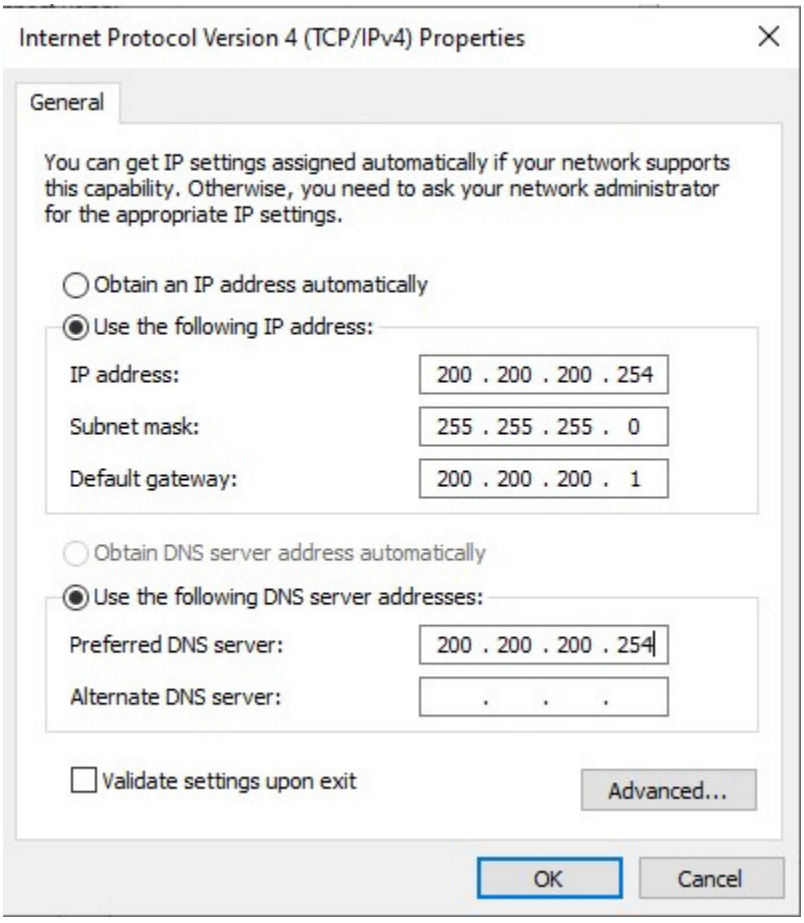


Figure 7 – IPv4 Properties window

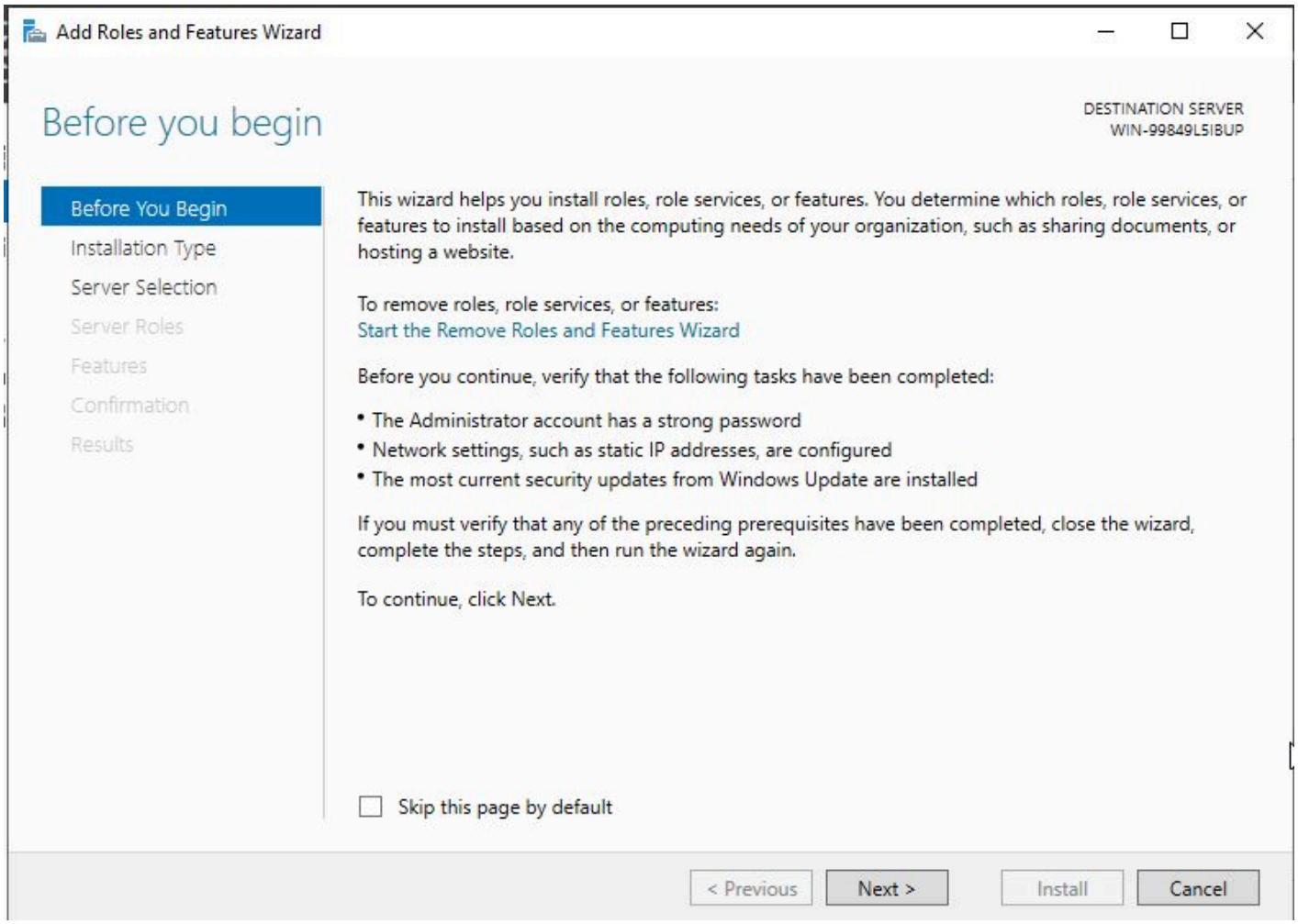


Figure 8 – Before you begin screen

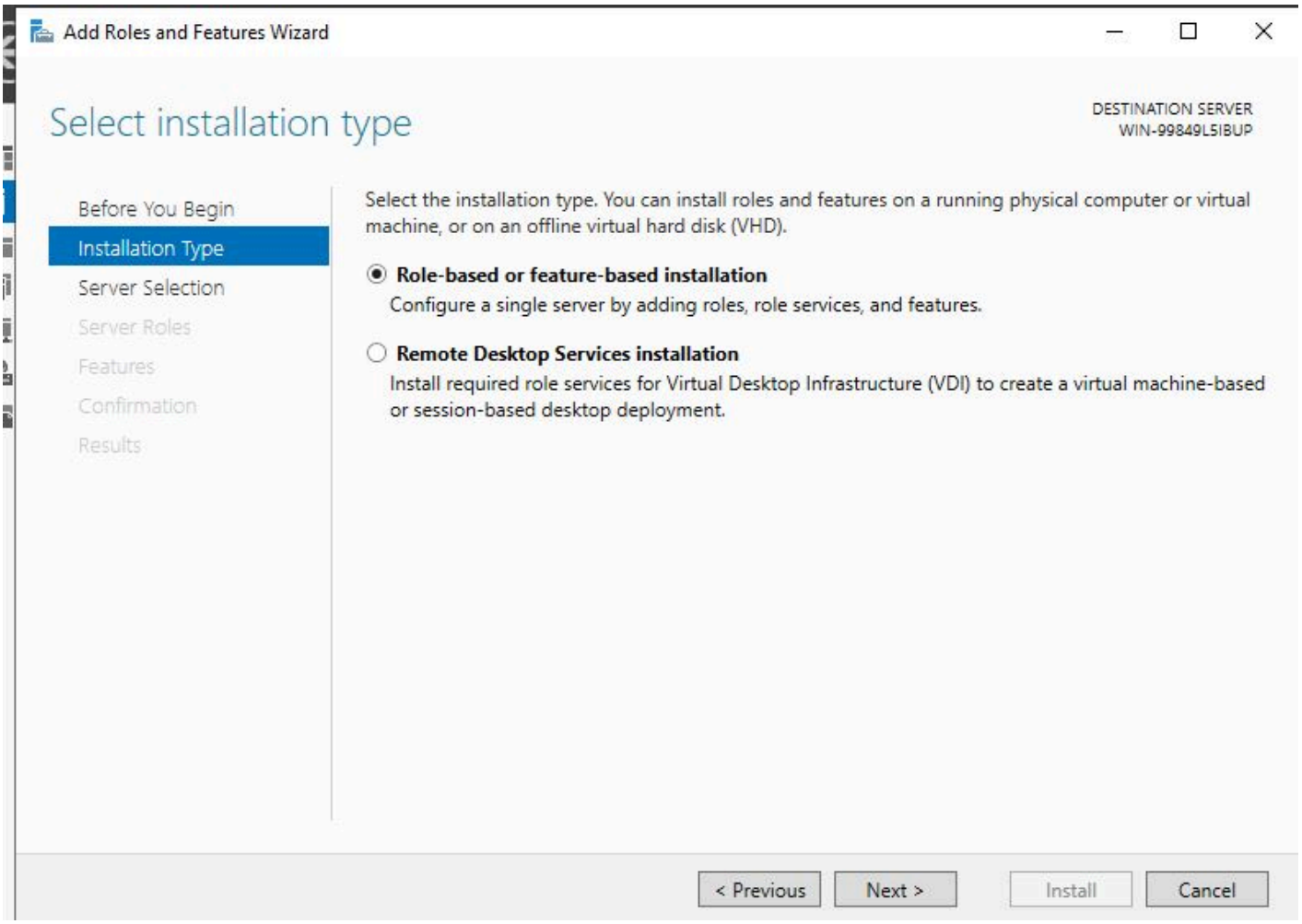


Figure 9 – Installation type screen

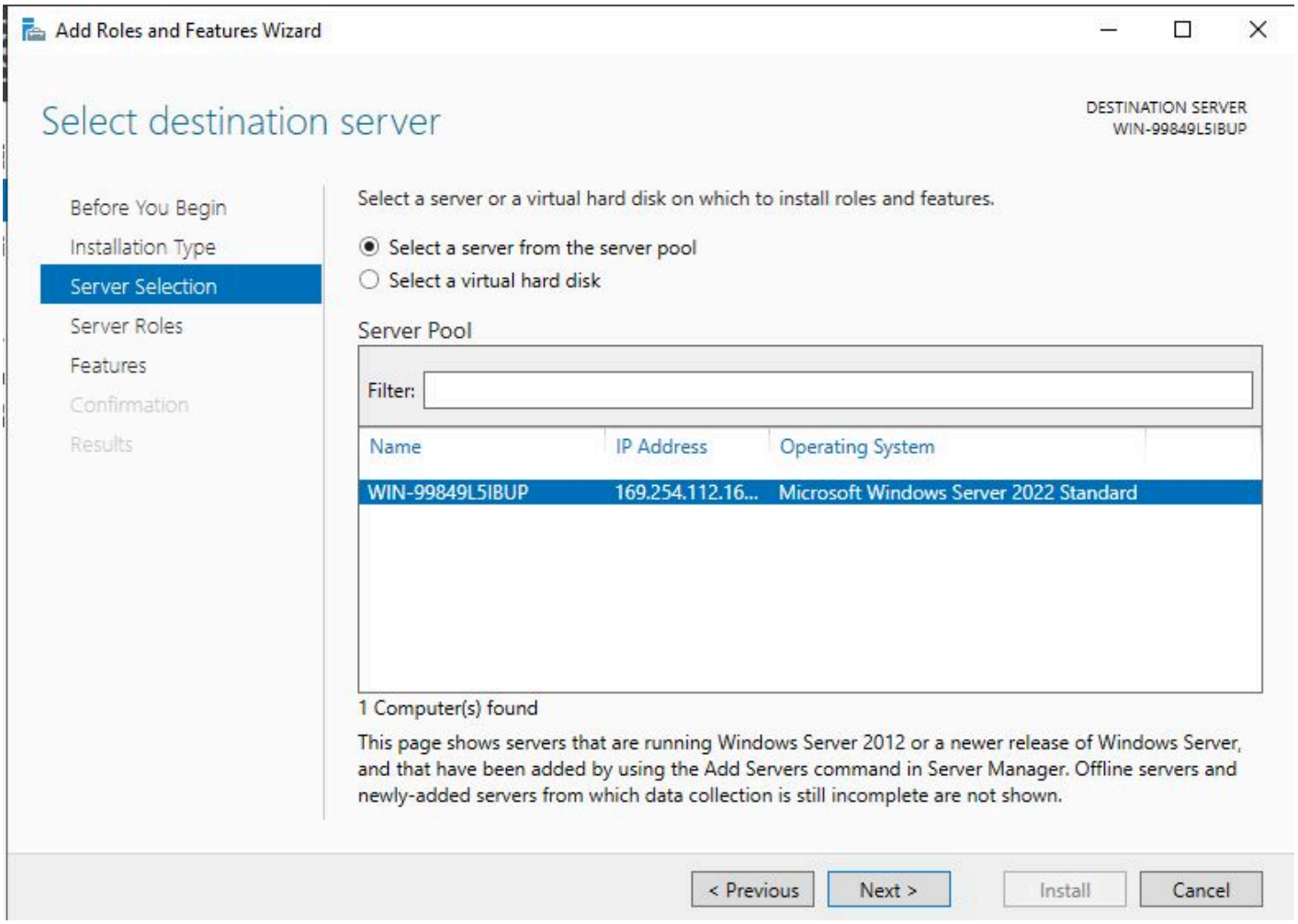


Figure 10 – Server Selection screen

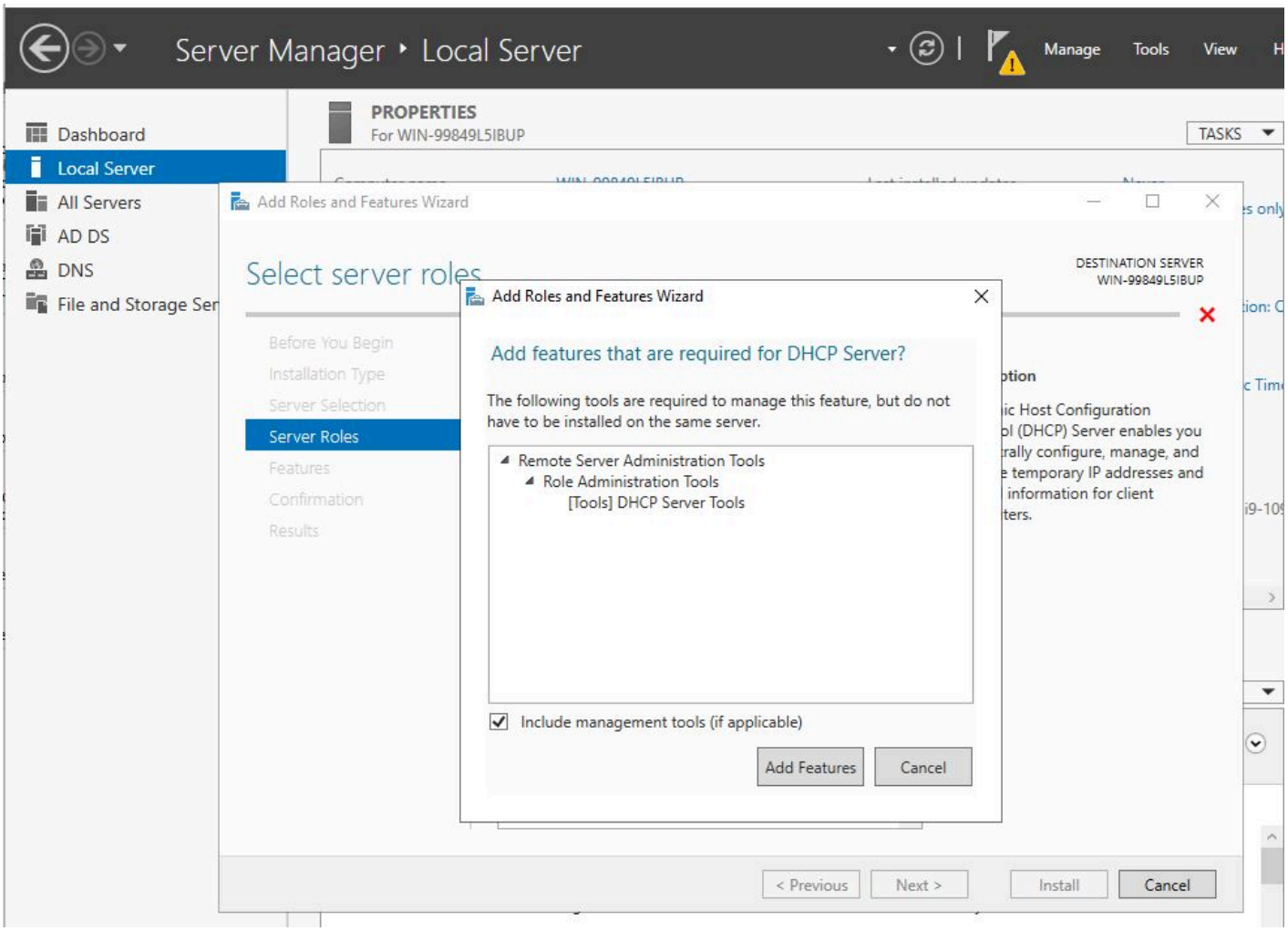


Figure 11 – Select add features

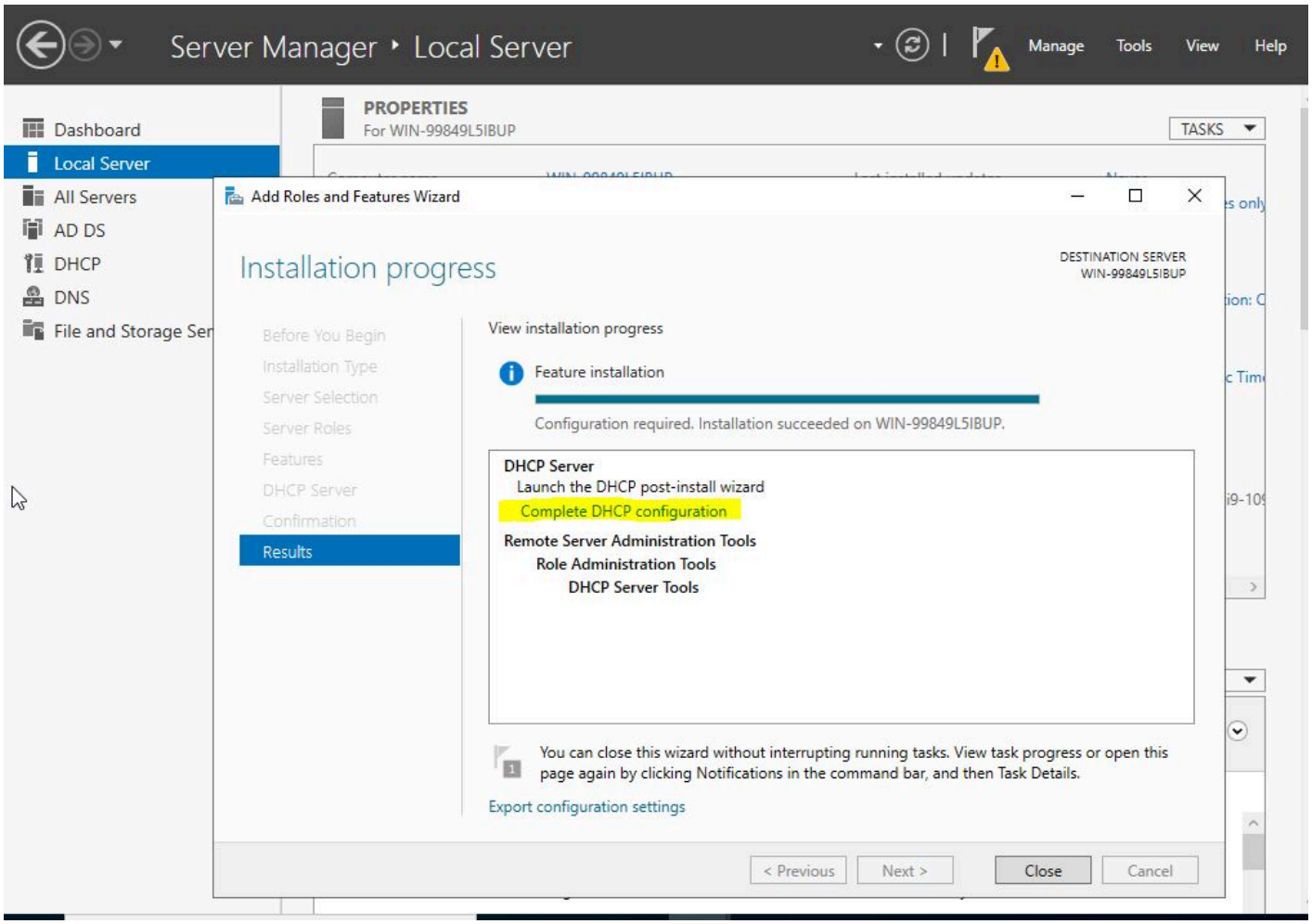


Figure 12 – Click Complete DHCP configuration

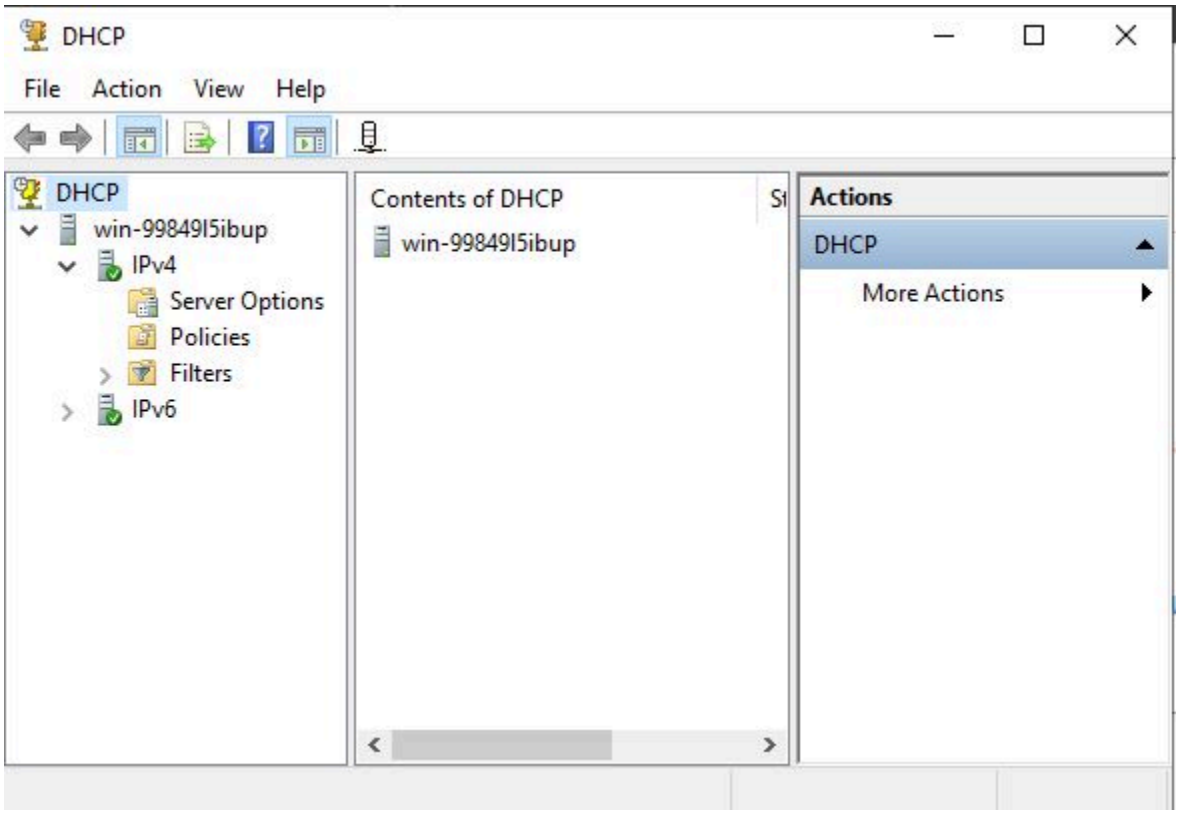


Figure 13 – Expand the local computer

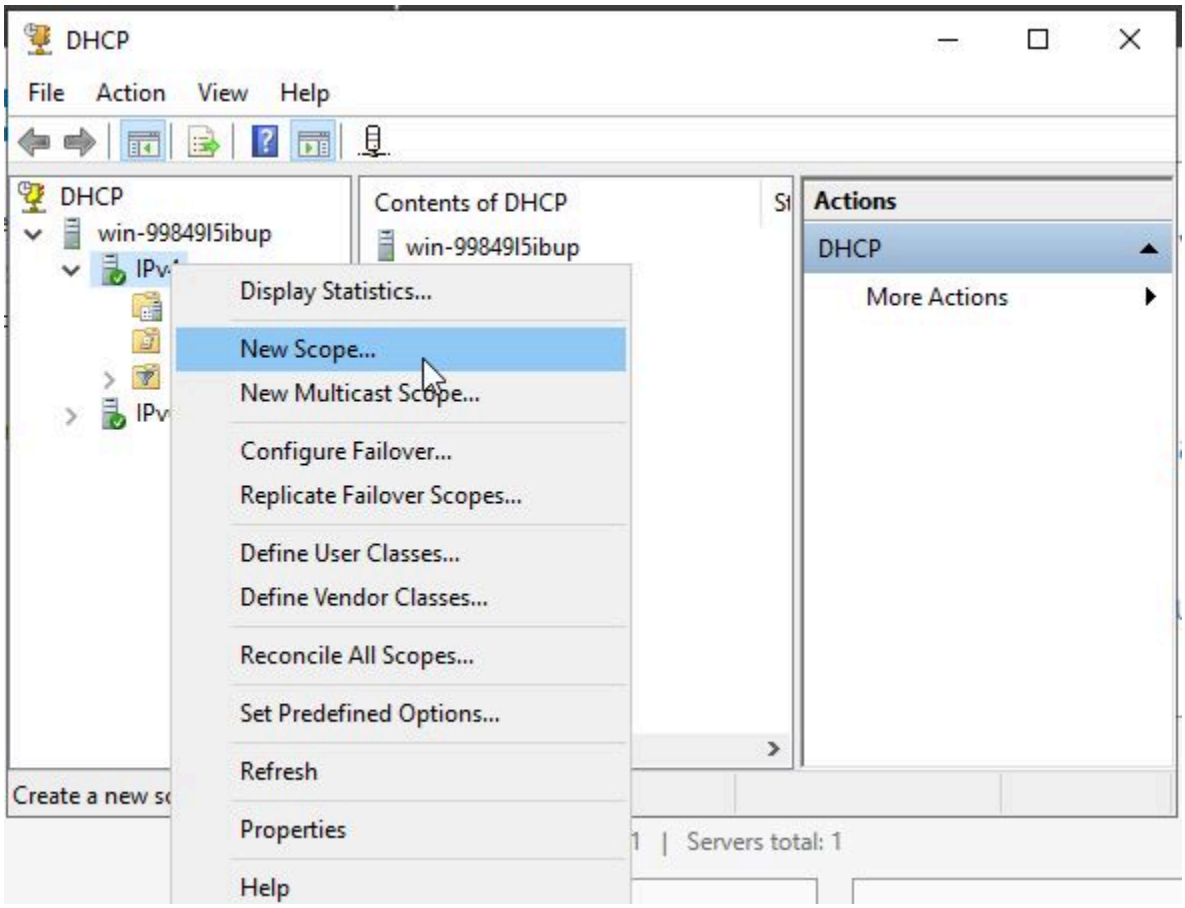


Figure 14 – Create a new scope

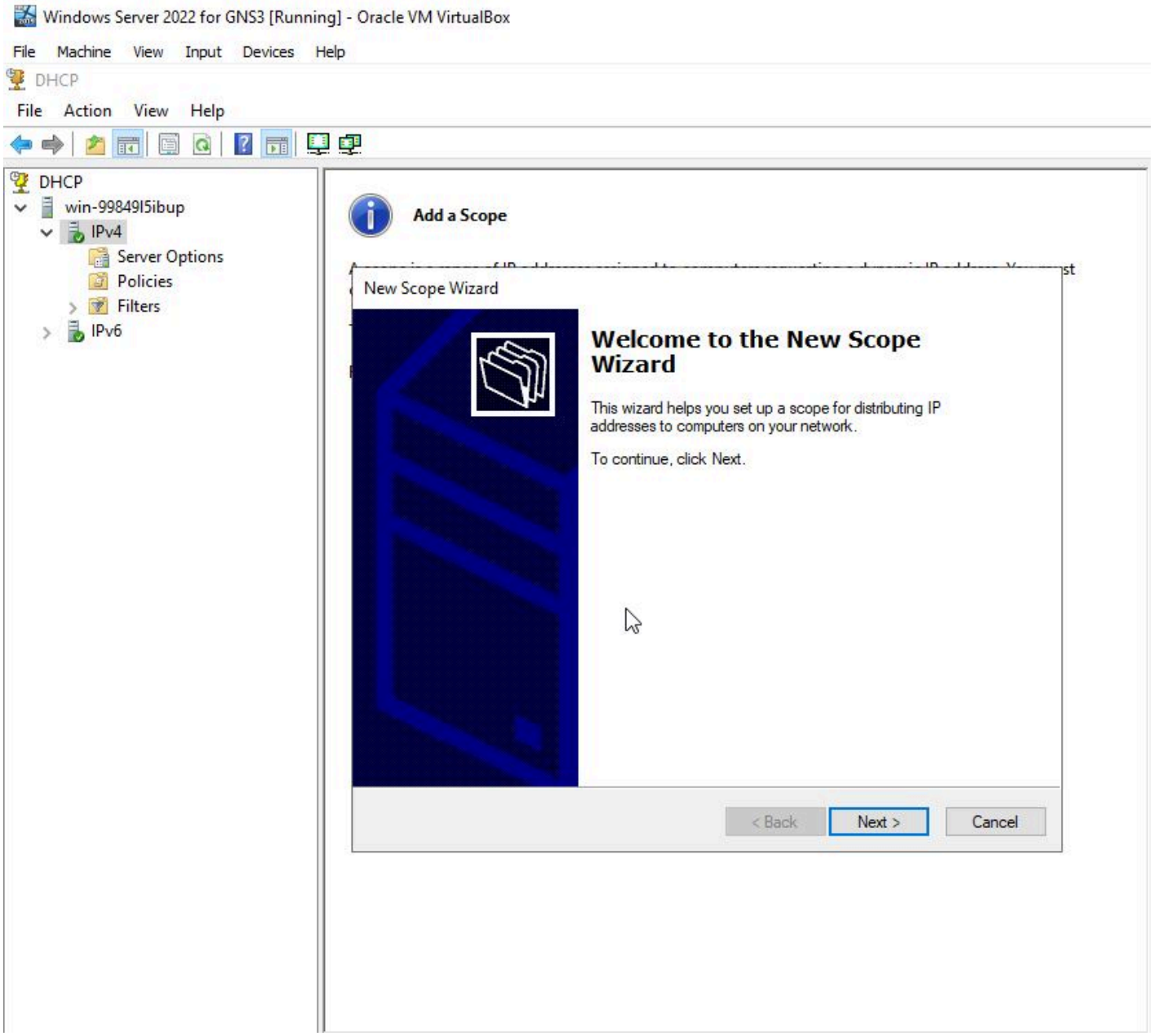


Figure 15 – Welcome to New Scope Wizard Screen

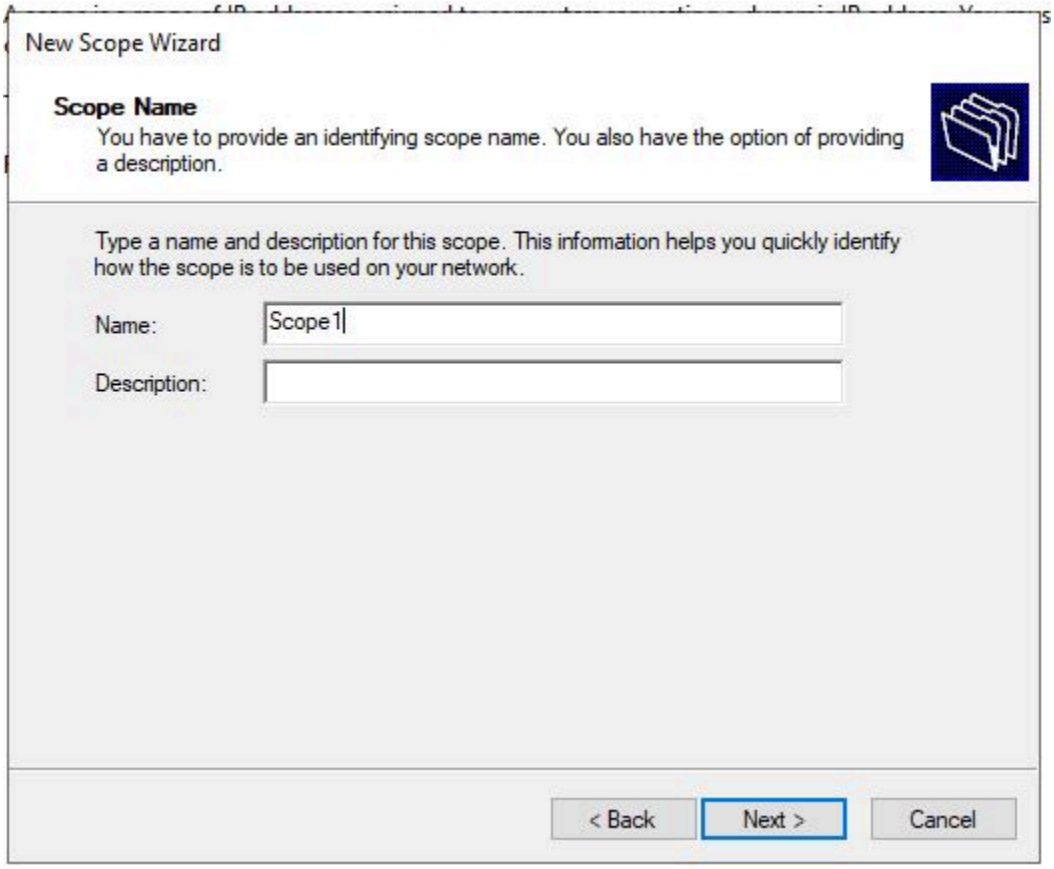


Figure 16 – Scope Name Screen

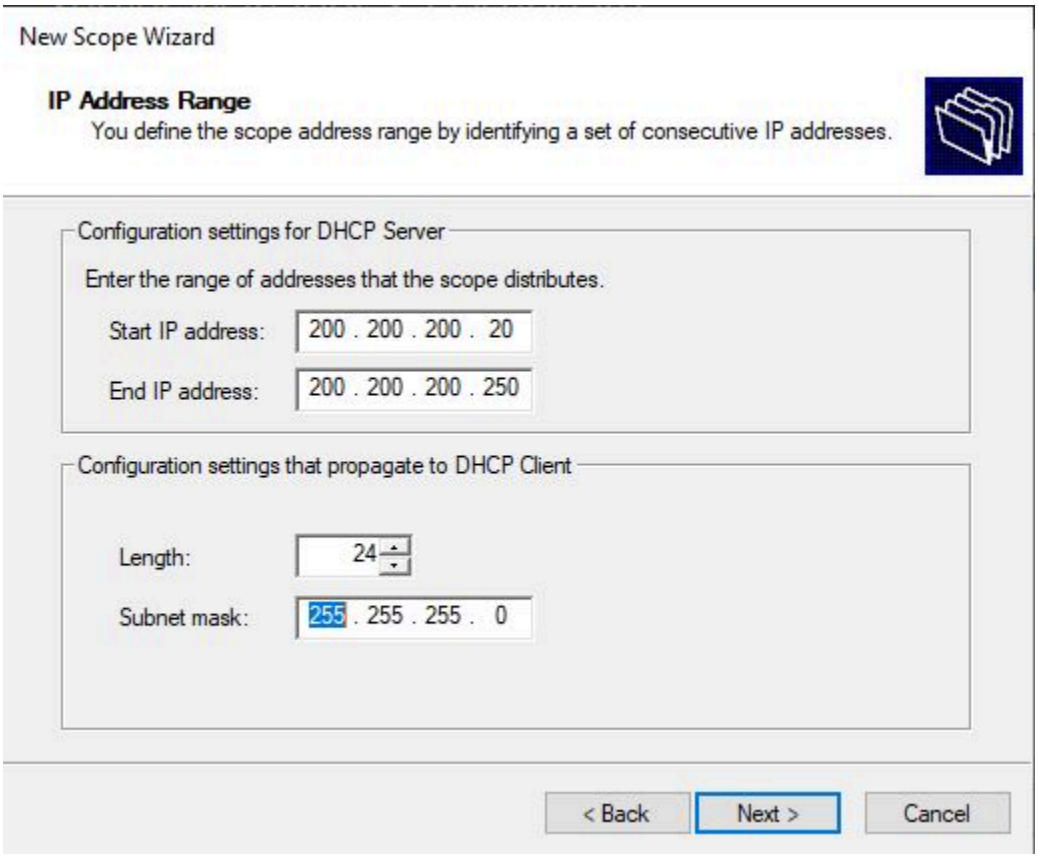


Figure 17 – IP Address Range Screen

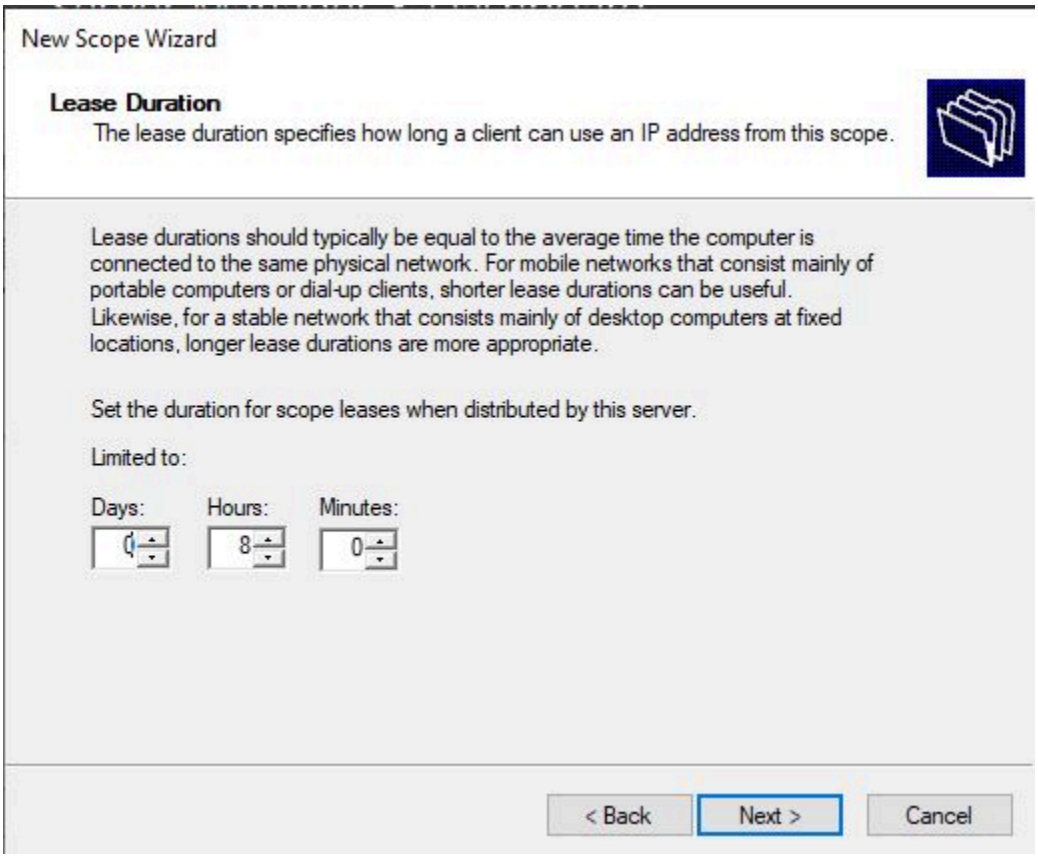


Figure 18 – Lease Duration screen

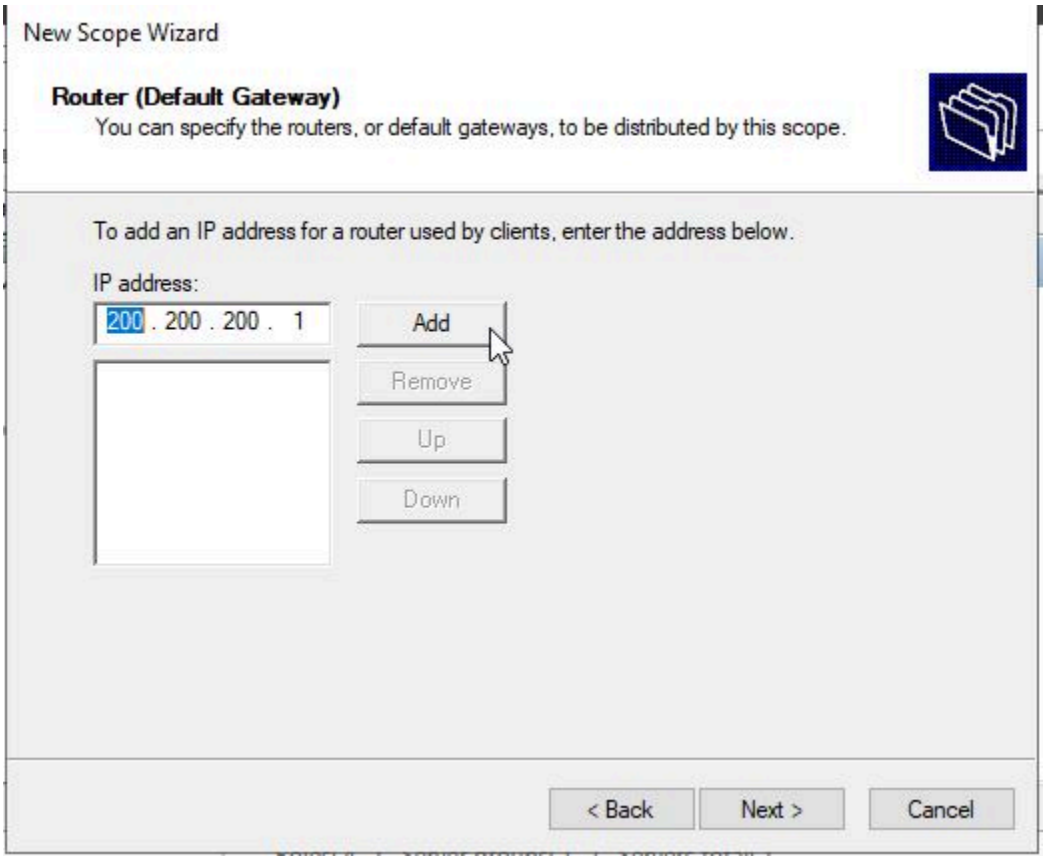


Figure 19 – Default Gateway screen

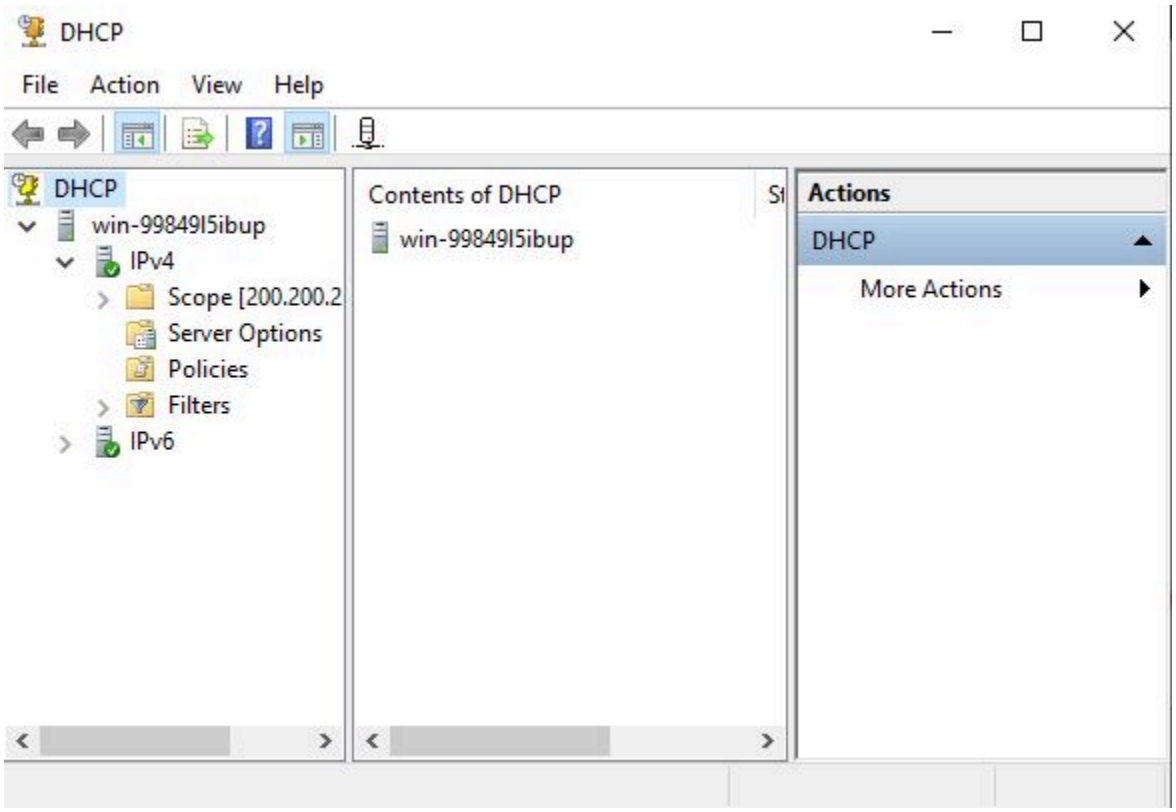


Figure 20 – Completed scope

CHAPTER 20

Dynamic Host Configuration Protocol - MikroTik CHR

JACOB CHRISTENSEN

Thus far, we've explored two approaches to integrating DHCP servers into a network using both Linux and Windows as dedicated servers. This chapter introduces yet another method for deploying DHCP, in the form of a router. This proves particularly handy in scenarios where a quick and easy solution is required. Configuring DHCP in this manner offers rapid deployment and simplicity, making it ideal fit smaller network environments.

Estimated time for completion: 10 minutes

LEARNING OBJECTIVES

- Successfully deploy a DHCP solution using a MikroTik router on an enterprise network
- Capture and Observe DHCP packets using Wireshark
- Capture and Observe ARP packets using Wireshark
- Successfully add hosts to an enterprise network and receive IP addresses automatically

PREREQUISITES

- Chapter 16 – Introduction to Routers

DELIVERABLES

Five screenshots are required:

- Neatly labeled and organized GNS3 workspace
- MikroTik router configuration
- Screenshot of Wireshark
    - DHCP packets for PC1
    - DHCP packets for PC2
    - PC1 pinging PC2

RESOURCES

- MikroTik RouterOS Documentation – "DHCP Server" – https://help.mikrotik.com/docs/display/ROS/DHCP#DHCP-Summary.2

CONTRIBUTORS AND TESTERS

- Dante Rocca, Cybersecurity Student, ERAU-Prescott

Phase I -Build the Network Topology

The following steps are to create a baseline environment for completing the lab. It makes assumptions about learner knowledge from completing previous labs.

Your final network will look like the following:

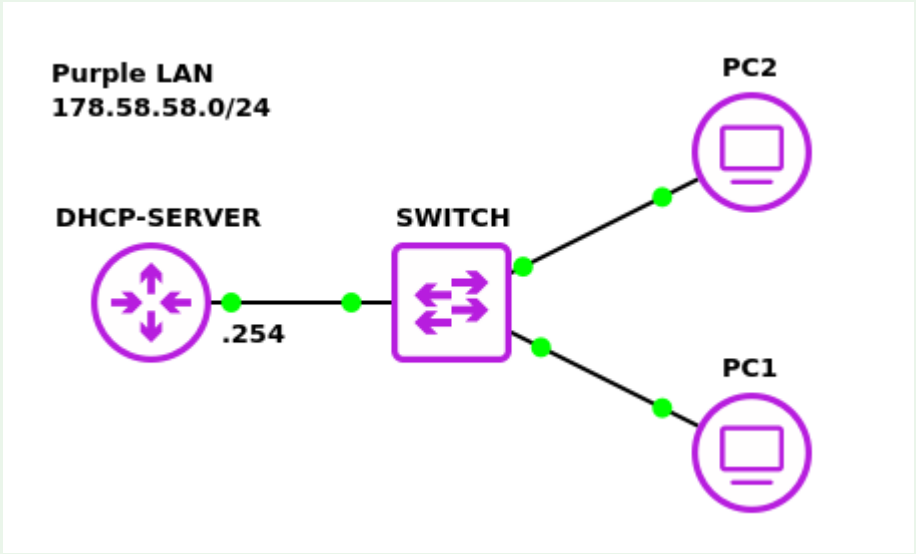


Figure 1 – Final GNS3 network environment

1. Start GNS3
    1.1. Create a new project: **LAB_06**
2. Build a Class C subnet with the network address **178.58.58.0/24**
    2.1. Two client devices – ***VPCS***
    2.2. One switch – ***Ethernet switch***
    2.3. One DHCP server – ***MikroTik router***

> **NOTE:** The MikroTik CHR version used when making this lab was **7.11.3**.

2.4. Connect the PCs to the switch

2.5. Connect port ether1 on the router to the switch

3. Label and organize your network as necessary

Phase II – Configuring the MikroTik Router

Once the network is built we need to configure the router to act as our DHCP server.

1. Start the MikroTik router and open its console

1.1. Change the hostname to reflect the router's primary purpose

```
> system identity set name=DHCP-SERVER
```

1.2. Remove the default DHCP listener on ether1

```
> ip dhcp-client remove 0
```

```
[admin@DHCP-SERVER] > ip dhcp-client print
Columns: INTERFACE, USE-PEER-DNS, ADD-DEFAULT-ROUTE, STATUS
# INTERFACE  USE-PEER-DNS  ADD-DEFAULT-ROUTE  STATUS
0 ether1     yes           yes                searching...
[admin@DHCP-SERVER] > ip dhcp-client remove 0
[admin@DHCP-SERVER] > ip dhcp-client print

[admin@DHCP-SERVER] >
```

Figure 2 – Removing the DHCP client

1.3. Assign a static IP address to its running interface

```
> ip address add address=178.58.58.254/24 interface=ether1
```

> **NOTE:** In this example, I have ***ether1*** connected to the switch. Remember to adjust this to be applicable for your environment.

```
[admin@DHCP-SERVER] > ip address add address=178.58.58.254/24 interface=ether1
[admin@DHCP-SERVER] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS           NETWORK      INTERFACE
0 178.58.58.254/24  178.58.58.0  ether1
[admin@DHCP-SERVER] >
```

Figure 3 – Assigned IPv4 addresses

1.4. Use the built-in setup wizard to configure the DHCP server (Figure 4)

```
> ip dhcp-server setup
```

1.4.1. dhcp server interface: ***ether1***

1.4.2. dhcp address space: ***178.58.58.0/24***

1.4.3. gateway for dhcp network: ***178.58.58.254***

1.4.4. addresses to give out: ***178.58.58.1-178.58.58.253***

1.4.5. dns servers: (none just hit <***Enter***>)

1.4.6. lease time: ***1800***

2. Test the DHCP service on the network

2.1. From PC1, request a new host address

```
> ip dhcp
```

2.2. From PC2, request a new host address

```
> ip dhcp
```

3. From PC1, ping PC2 to test connectivity

End of Lab

Deliverables

Five screenshots are required to receive credit for this exercise:

- GNS3 workspace with all devices labeled
- MikroTik router configuration
- Wireshark capture of PC1 devices getting and receiving DHCP IPv4 addresses
- Wireshark capture of PC2 devices getting and receiving DHCP IPv4 addresses
- Wireshark caputre of PC1 pinging PC2

Homework

Assignment 1 – Create a LAN for 43 hosts with a Mikrotik DHCP server while minimizing unused IP addresses

- Used a randomized network address
- There's no need to put in all 43 host just show the setup process for the DHCP server and that it is working with at least two hosts
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Environment
    - Screenshot of end devices receiving IP addresses
    - Screenshot of DHCP setup process

Assignment 2 – Use the Mikrotik router as both a DHCP server and a router

- Add another LAN attached to the same Mikrotik router
- Ensure devices on both LANs use the Mikrotik router as a DHCP server
- Ensure devices on both LANs can contact each other
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Environment
    - Screenshot of an end device on the first LAN receiving an IP address
    - Screenshot of an end device on the second LAN receiving an IP address
    - Screenshot of a device on one LAN pinging a device on the other LAN

Figures for the Printed Version

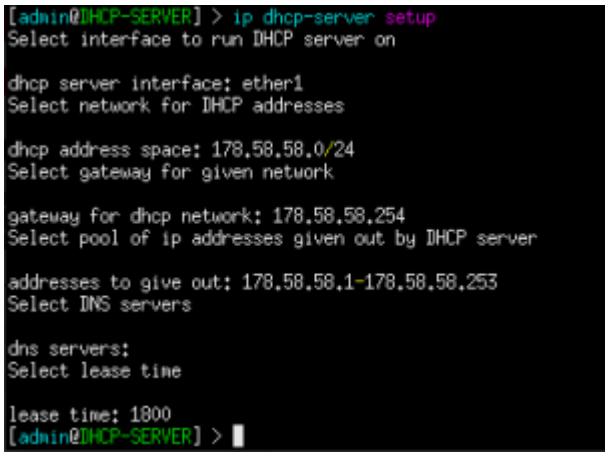


Figure 4 – DHCP Server setup wizard

CHAPTER 21

Static Networking Part 1

MATHEW J. HEATH VAN HORN, PHD AND JACOB CHRISTENSEN

Up to this point, we have only used one router in our working environments. However, you will rarely work on a network with only a single because the whole point of an enterprise network is to connect multiple LANs into a unified cohesive network.

In this lab, we will create and connect three LANs via routers. We introduce you to static routing solutions so you can become familiar with routing procedures. Static routing is impractical mainly because it is very manpower intensive to maintain and prone to human error.

Estimated time for completion: 70 minutes

LEARNING OBJECTIVES

- Successfully create two functional LANs:
    - Red (DHCP + 2 PCs)
    - Blue (DHCP + 2 PCs)
- Configure two routers to use static routing so all devices can communicate

PREREQUISITES

- Chapter 16 – Introduction to Routers
- Chapter 17 – IPv4 Addressing
- Chapter 18 – DHCP using Linux

DELIVERABLES

- Screenshot of GNS3 workspace with labels
- Screenshot of Wireshark IMCP packets showing a Red device successfully pinging a Blue device

RESOURCES

- MikroTik RouterOS Documentation – "IP Routing" – https://help.mikrotik.com/docs/display/ROS/IP+Routing

CONTRIBUTORS AND TESTERS

- Dante Rocca, Cybersecurity Student, ERAU-Prescott
- Sawyer Hansen, Cybersecurity Student, ERAU-Prescott

Overview

You are now going to combine your subnet, DHCP, and router configuration skills to create two networks, each with their own gateway routers. Your final product should look similar to the following.

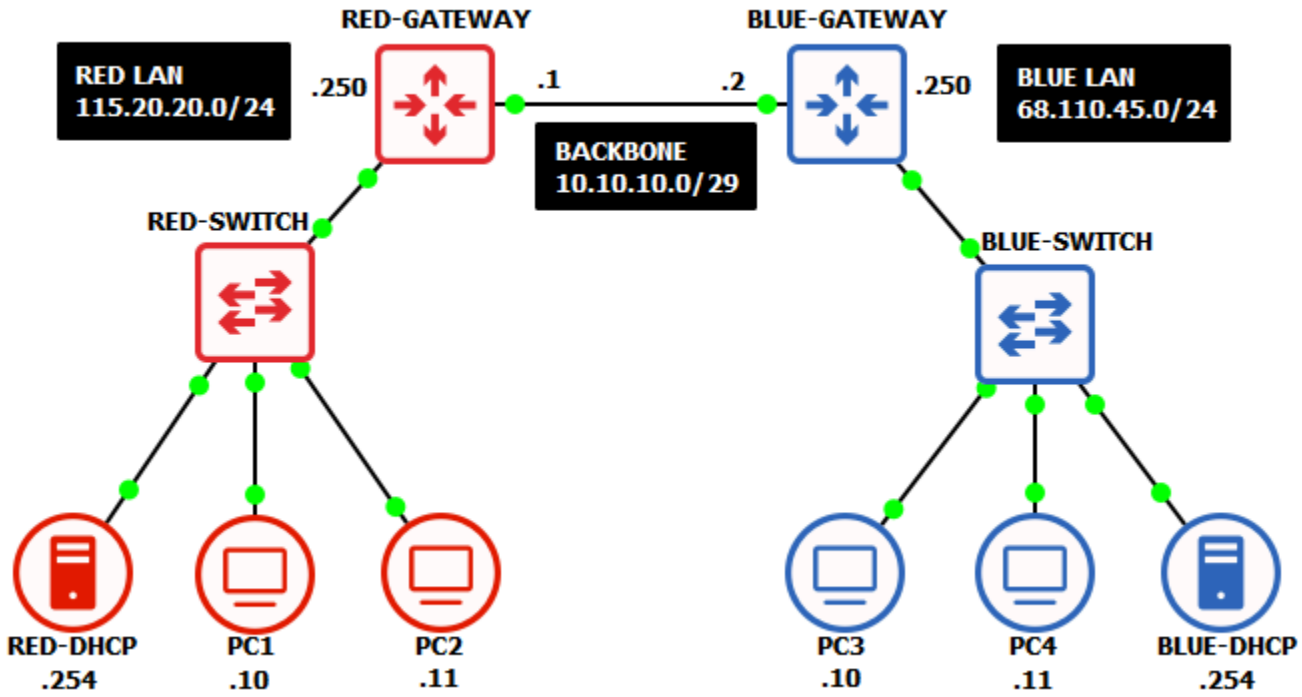


Figure 1 – Final GNS3 network environment

Phase I -Build the Network Topology

The following steps are to create the baseline for completing the lab. It makes assumptions about learner knowledge from completing previous labs. Going forward, we will be using Ubuntu Servers for all network servicing needs. For those who have limited computational resources, consider using TinyCore as an alternative.

1. Start GNS3

    1.1. Create a new project: **LAB_07**

2. Build the **Red** subnet with the following specifications:

    2.1. IP address space – **115.20.20.0/24**

    2.2. Two client machines – ***VPCS***

    2.3. One switch – ***Ethernet switch***

    2.4. One DHCP server – ***Ubuntu Server VM (isc-dhcp-server)***

    2.5. One router – ***MikroTik CHR***

    2.6. Connect the server and PCs to their associated switch

    2.7. Connect the switch to the router's ***ether1*** interface

3. Configure the MikroTik router to act as Red's gateway

> **NOTE:** Refer to Chapter 16, Phase II, Step 4 for more information.

    3.1. Set a new hostname to reflect its new purpose

```
> system identity set name=RED-ROUTER
```

    3.2. Set ***ether1*** with the IP address **115.20.20.250** for the Red network

```
> ip address add address=115.20.20.250/24 interface=ether1
```

    3.3. Verify that it was taken

```
> ip address print
```

MikroTik Configuration Preface

By default, MikroTik routers will have a DHCP client enabled on interface ***ether1***, which will automatically request an IP address once the network's DHCP server is online. To avoid unnecessary packet traffic (and troubleshooting headaches), disable this client as part of the router's setup routine. You can identify interfaces with DHCP clients enabled via the "dynamic" (*D*) flag that appears next to its identification number.

```
[admin@MikroTik] > ip addres print
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#   ADDRESS            NETWORK        INTERFACE
0   192.168.1.1/24     192.168.1.0    ether1
1   215.100.12.1/24    215.100.12.0   ether2
2   10.11.101.1/24     10.11.101.0    ether3
3 D 200.200.200.20/24  200.200.200.0  ether1
[admin@MikroTik] >
```

Figure 2 – Dynamically assigned IP address

List all DHCP clients currently enabled on the device.

```
> ip dhcp-client print
```

```
[admin@MikroTik] > ip dhcp-client print
Columns: INTERFACE, USE-PEER-DNS, ADD-DEFAULT-ROUTE, STATUS, ADDRESS
# INTERFACE  USE-PEER-DNS  ADD-DEFAULT-ROUTE  STATUS  ADDRESS
0 ether1     yes           yes                bound   200.200.200.20/24
[admin@MikroTik] >
```

Figure 3 – Default MikroTik DHCP listener

Remove the DHCP client. When editing or deleting entries in RouterOS, you “select” a target row based on its number (#) in the first column. In this instance, we are removing entry zero.

```
> ip dhcp-client remove 0
```

```
[admin@MikroTik] > ip dhcp-client remove 0
[admin@MikroTik] > ip dhcp-client print

[admin@MikroTik] >
```

Figure 4 – Remove default DHCP listener

Verify that all IP addresses are now static. Note that this method can also be used to remove or edit accidental interface IP assignements. Instead of ***ip dhcp-client remove #***, the syntax would be ***ip address remove #***, where **#** represents the row you wish to select.

```
[admin@MikroTik] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS           NETWORK        INTERFACE
0 192.168.1.1/24    192.168.1.0    ether1
1 215.100.12.1/24   215.100.12.0   ether2
2 10.11.101.1/24    10.11.101.0    ether3
[admin@MikroTik] >
```

Figure 5 – Print currently assigned IP addresses

4. Configure Ubuntu Server to act as Red's DHCP server

> **NOTE:** Refer to Chapter 18, Phase II, Steps 2-7 for more information.

4.1. Assign its Ethernet card with the static IP address of **115.20.20.254/24** and a default gateway of **115.20.20.250** (ether1 on the Red router)

```
> sudo vi /etc/netplan/00-installer-config.yaml
```

```
# This is the network config written by 'Jake M. Christensen'
# 2023.02.07
network:
  ethernets:
    enp0s3:
      optional: true
      dhcp4: false
      addresses:
        - 115.20.20.254/24
      routes:
        - to: default
          via: 115.20.20.250
  version: 2
```

Figure 6 – Red DHCP server interface configuration

4.2. Apply the configuration

```
> sudo netplan apply
```

4.3. Configure the DHCP daemon with a host range of **.10** to .**150** in addition to a gateway address to ***ether1*** on the Red MikroTik router

```
> sudo vi /etc/dhcp/dhcpd.conf
```

```
# Configuration written by 'Jake M. Christensen'
# 2024.02.08

# This is the main DHCP server on this subnet
authoritative;

# Global parameters
default-lease-time 600;
max-lease-time 7200;

# Red subnet directive
subnet 115.20.20.0 netmask 255.255.255.0 {
  range 115.20.20.10 115.20.20.150;
  option routers 115.20.20.250;
  option broadcast-address 115.20.20.255;
}
```

Figure 7 – Red DHCP configuration

> **NOTE:** Notice the addition of ***option routers*** in this configuration file. This will automatically assign a gateway address to DHCP clients.

5. Have each VPCS request a new IP address

```
> ip dhcp
```

6. Ensure that all devices within the LAN can ping each other

7. Repeat steps 2 through 6 to build the **Blue** subnet

    7.1. Blue will have the IP address space of **68.110.45.0/24**

    7.2. The Blue router's IP on ***ether1*** is **68.110.45.250**

    7.3. The Blue DHCP server's static IP is **68.110.45.254**

    7.4. The Blue DHCP daemon will have the same specifications as Red

```
#_Configuration written by 'Jake M. Christensen'
# 2024.02.16
#
# This is the main DHCP server on this subnet
authoritative;

# Global parameters
default-lease-time 600;
max-lease-time 7200;

# Blue subnet directive
subnet 68.110.45.0 netmask 255.255.255.0 {
  range 68.110.45.10 68.110.45.150;
  option routers 68.110.45.250;
  option broadcast-address 68.110.45.255;
}
```

Figure 8 – Blue DHCP configuration

8. Label and organize your network as necessary

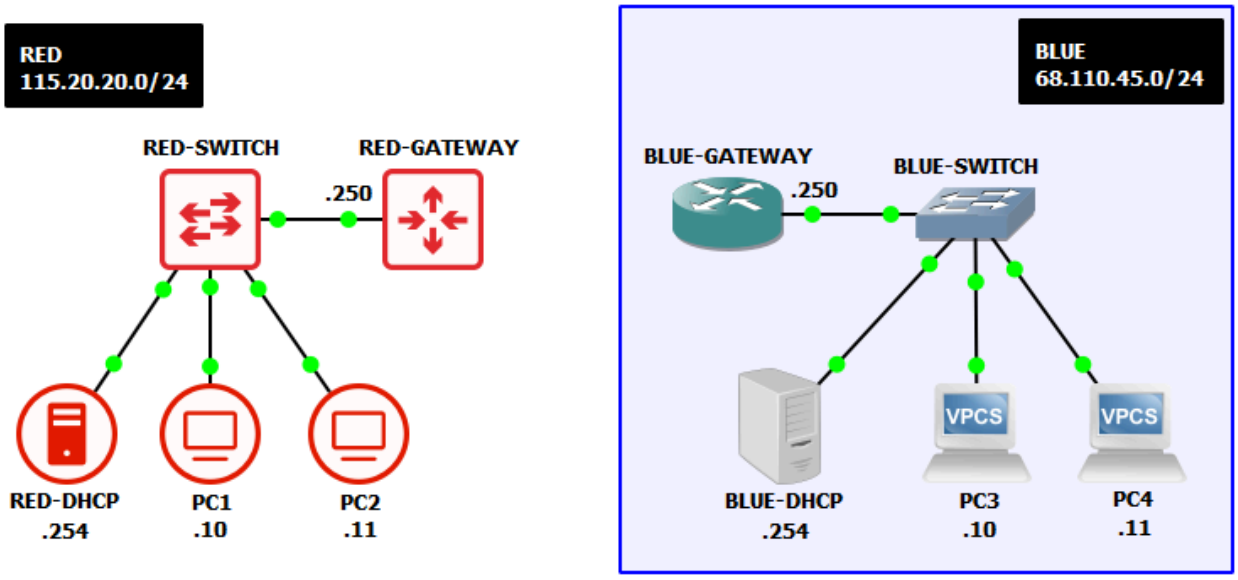


Figure 9 – GNS3 working environment

Phase II – Join Networks to their Host LANs

Routers are very similar to post offices. If a letter comes into the post office and the destination address is within the neighborhood, the post office will hand the letter to another carrier. This is of limited value, but the router earns its paycheck when the letter needs to go to another neighborhood. The postmaster will find the most efficient route to get the letter to the right neighborhood.

To get used to this idea, we are going to configure our routers just to speak to their local "homes" and the post office in the next neighborhood. We are not worried about efficiencies at this point, let's just get the postmasters talking. This kind of configuration is called static routing because nothing changes.

1. On the Red router, set *ether2* with the static IP address **10.10.10.1/29** for the **Backbone** network

```
> ip address add address=10.10.10.1/29 interface=ether2
```

NOTE: A ***backbone*** is network IP space that is only used by routers to speak to each other. In this example, we are using 10.10.10.0/29 which has a maximum of***six*** host addresses. This while only two devices (router interfaces) are currently connected on this network, this leaves room for four more potential devices.

2. On the Blue router, set *ether2* with the static backbone IP address **10.10.10.2/29**

```
> ip address add address=10.10.10.2/29 interface=ether2
```

3. Connect Red router's *ether2* interface with Blue router's *ether2* interface

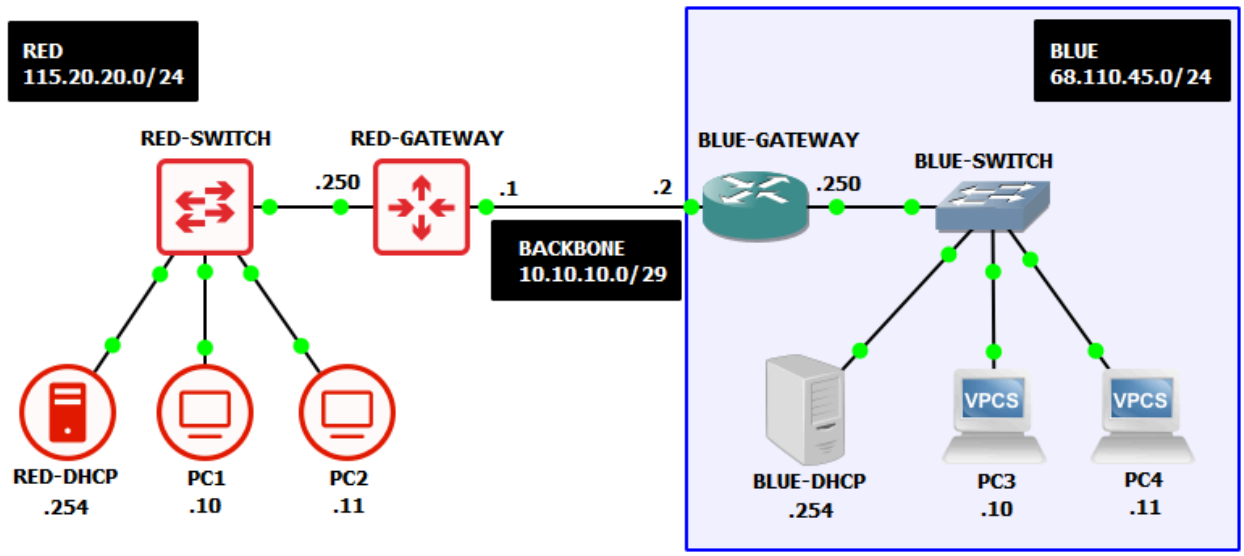


Figure 10 – Connected Gateways

Phase III – View Capabilities of our Current State

Congratulations! You have built two LANs and connected them together through routers. Or have you?

Remember, if a networked device has no idea where to send a data packet, it will discard it. This part of the lab lets you see this concept in action.

1. Start two Wireshark data captures

    1.1. Red-Switch to Red-Router

    1.2. Red-Router to Blue-Router

2. Open PC1 and ping Red-Router

```
> ping 115.20.20.250
```

    2.1. Observe the ICMP packets on the Red-switch to Red-Router Wireshark window

icmp

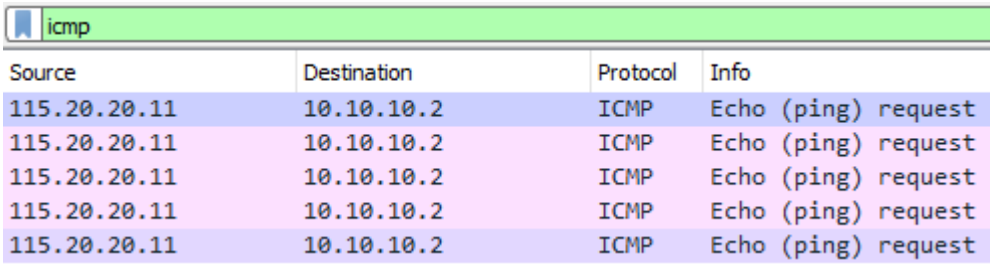
Source	Destination	Protocol	Info
115.20.20.11	115.20.20.250	ICMP	Echo (ping) request
115.20.20.250	115.20.20.11	ICMP	Echo (ping) reply
115.20.20.11	115.20.20.250	ICMP	Echo (ping) request
115.20.20.250	115.20.20.11	ICMP	Echo (ping) reply
115.20.20.11	115.20.20.250	ICMP	Echo (ping) request
115.20.20.250	115.20.20.11	ICMP	Echo (ping) reply
115.20.20.11	115.20.20.250	ICMP	Echo (ping) request
115.20.20.250	115.20.20.11	ICMP	Echo (ping) reply
115.20.20.11	115.20.20.250	ICMP	Echo (ping) request
115.20.20.250	115.20.20.11	ICMP	Echo (ping) reply

Figure 11 – ICMP ping packets

3. Now from PC1, ping Blue-Router

```
> ping 10.10.10.2
```

    3.1. Observe the ICMP packets on the Red-Router to Blue-Router Wireshark Window



icmp

Source	Destination	Protocol	Info
115.20.20.11	10.10.10.2	ICMP	Echo (ping) request
115.20.20.11	10.10.10.2	ICMP	Echo (ping) request
115.20.20.11	10.10.10.2	ICMP	Echo (ping) request
115.20.20.11	10.10.10.2	ICMP	Echo (ping) request
115.20.20.11	10.10.10.2	ICMP	Echo (ping) request

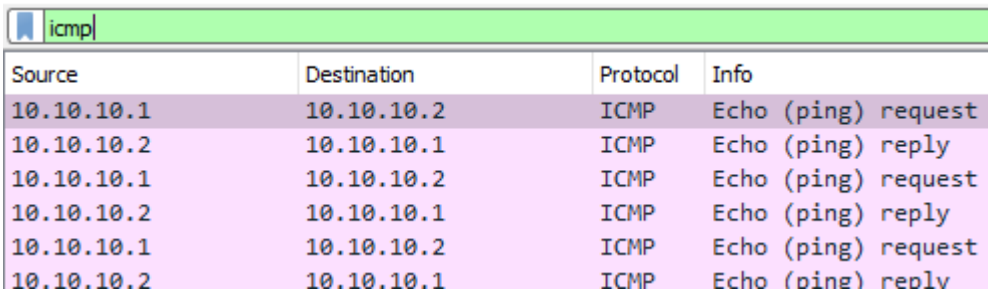
Figure 12 – Failed ICMP ping packets

> **What happened? What is the problem? Maybe we configured the routers wrong.**

4. Open the Red-Router console and ping the Blue-Router by typing

```
> ping 10.10.10.2
```

4.1. Observe the ICMP packets on the Red-Router to Blue-Router Wireshark Window



icmp

Source	Destination	Protocol	Info
10.10.10.1	10.10.10.2	ICMP	Echo (ping) request
10.10.10.2	10.10.10.1	ICMP	Echo (ping) reply
10.10.10.1	10.10.10.2	ICMP	Echo (ping) request
10.10.10.2	10.10.10.1	ICMP	Echo (ping) reply
10.10.10.1	10.10.10.2	ICMP	Echo (ping) request
10.10.10.2	10.10.10.1	ICMP	Echo (ping) reply

Figure 13 – ICMP ping packets between routers

> **That worked, so what is the problem?**

5. Open the PC1 console and ping the Blue Network DHCP server by typing ping 68.110.45.250

```
> ping 68.110.45.254
```

5.1. Observe the ICMP packets on both Wireshark data packet captures

icmp

Source	Destination	Protocol	Info
115.20.20.11	68.110.45.254	ICMP	Echo (ping) request id=
115.20.20.250	115.20.20.11	ICMP	Destination unreachable
115.20.20.11	68.110.45.254	ICMP	Echo (ping) request id=
115.20.20.250	115.20.20.11	ICMP	Destination unreachable
115.20.20.11	68.110.45.254	ICMP	Echo (ping) request id=
115.20.20.250	115.20.20.11	ICMP	Destination unreachable
115.20.20.11	68.110.45.254	ICMP	Echo (ping) request id=
115.20.20.250	115.20.20.11	ICMP	Destination unreachable
115.20.20.11	68.110.45.254	ICMP	Echo (ping) request id=
115.20.20.250	115.20.20.11	ICMP	Destination unreachable

Figure 14 – ICMP destination unreachable error

What do you see? What is happening?

Phase IV – Configure the Routers

Even though the routers know what networks are connected to them, they have no knowledge of the networks that are not connected to them. Let's look at our current network.

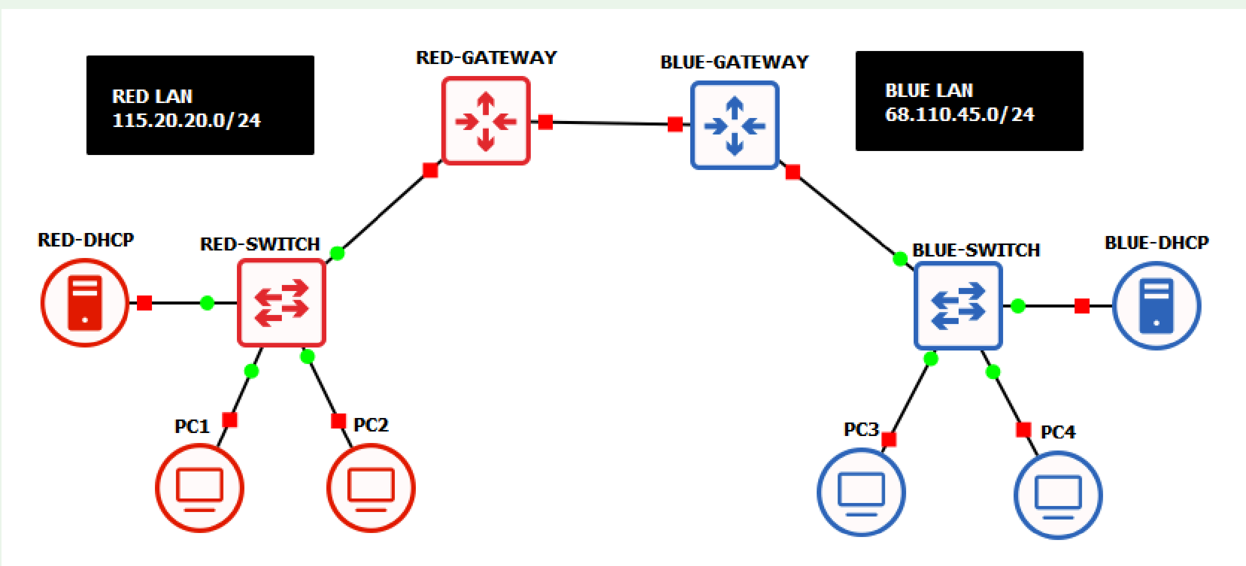


Figure 15 – The current network

Ok, we are only concerned with one path, so let's simplify our diagram to the essentials.

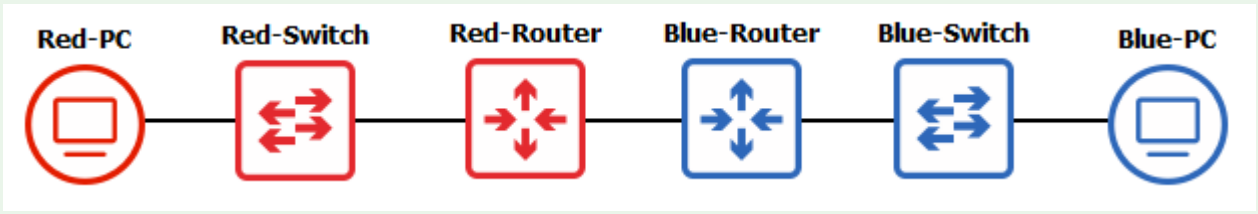


Figure 16 – The current network simplified

The switches are unmanaged, so we don't even need them for this explanation. So we'll take those out, change our routers to use color symbols, add the IP addresses, and label the simplified links, which takes our diagram down to the essentials.

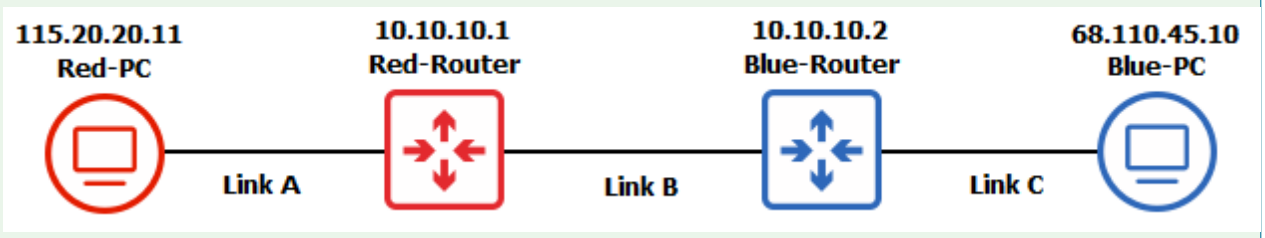


Figure 17 – The current network even more simplified

Remember what a ping packet (ICMP) does: It sends a request to a target interface and asks that interface to send it back to the originator.

When the Red PC pings the Red Router we can see the packets on Link A for both request and response.

Try it – Open a Wireshark capture for Link A. Then from the PC 1 console type *ping 115.20.20.250*

The Red Router knows that network 115.20.20.0 is connected to ether1 and sends the response.

Now when Red Router pings Blue Router we can see the packets on Link B for both request and response.

Try it – Open a Wireshark capture for Link B. Then from the Red Router console type *ping 10.10.10.2*

The Blue Router knows that network 10.10.10.0 is on ether2 and sends the response.

However, when Red PC pings the Blue Router, the Red Router forwards the packets to Blue Router, but when Blue Router tries to send the response packets back to Red PC, it has no idea what interface to use to send packets to network 115.20.20.0, so it never sends the responses.

Finally, when Red PC pings the Blue PC, it has no idea where to send the request packets, so it just throws up its arms and tells us, "Nope. I'm out."

We are going to fix this problem.

1. Stop the Wireshark captures (this saves our host machine's resources)
2. Configure a static routing table on the Red subnet's router
    2.1. Open the Red-Router console
    2.2. Add a new static route to our Red-Router

```
> ip route add dst-address=68.110.45.0/24 gateway=10.10.10.2
```

Command	Purpose
ip route add	Add a new IPv4 route
dst-address=68.110.45.0/24	Any packet trying to go to the 68.110.45.0 network
gateway=10.10.10.2	Forward the packet to this destination interface

3. Configure a static routing table on the Blue subnet's routers

   3.1. Add a new static route to our Blue Router by navigating to the Blue Router Console and typing

```
> ip route add dst-address=115.20.20.0/24 gateway=10.10.10.1
```

4. Now navigate to PC1 and try to ping the Blue router or any Blue end device

icmp

Source	Destination	Protocol	Info
115.20.20.11	68.110.45.11	ICMP	Echo (ping) request
68.110.45.11	115.20.20.11	ICMP	Echo (ping) reply
115.20.20.11	68.110.45.11	ICMP	Echo (ping) request
68.110.45.11	115.20.20.11	ICMP	Echo (ping) reply
115.20.20.11	68.110.45.11	ICMP	Echo (ping) request
68.110.45.11	115.20.20.11	ICMP	Echo (ping) reply

Figure 18 – ICMP ping working

Congratulations! You successfully implemented a basic network using static routing!

Phase V – Add a Green Subnet

You now have all the tools to build a LAN, add it to a network, and configure routers so that the LANs can send packets back and forth. Time to try it on your own.

1. Create a Green LAN similar to our Red and Blue LANs

   1.1. Use a ***randomly generated*** IPv4 network address space

   1.2. Use a network address that ***minimizes*** wasted IPs for 35 hosts

   1.3. Ensure that it has ***two VPCS's***, ***one DHCP server***, and ***one Ethernet switch***

2. Connect the Green subnet to the Red router
3. Configure the Red Router so the Green and Red LANs can ping each other
4. Configure the Blue Router so the Green and Blue LANs can ping each other
5. Label and organize your network as necessary

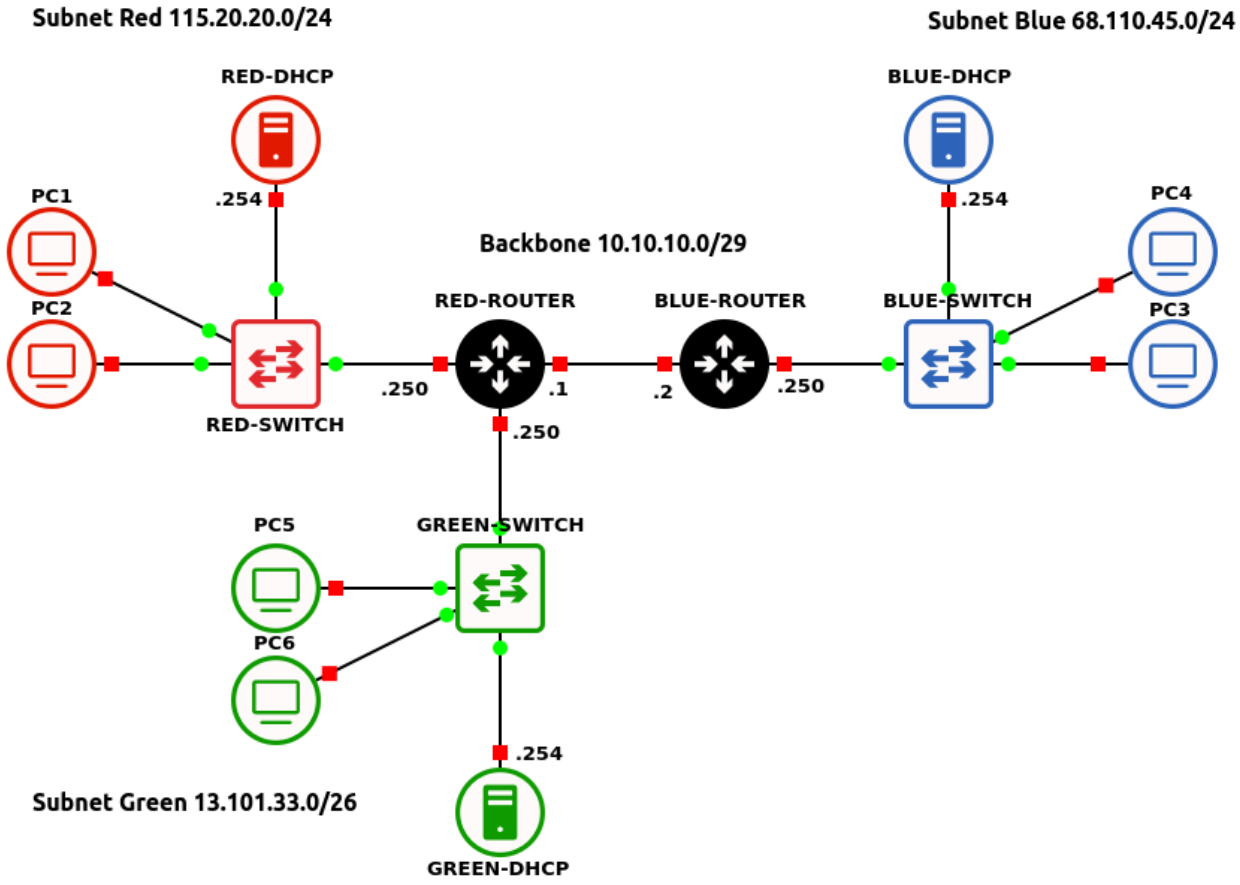


Figure 19 – Green LAN added to the network example

End of Lab

Deliverables

4 screenshots are required to receive credit for this lab

- Screenshot of GNS3 workspace with everything labeled
- Screenshot of Wireshark IMCP packets

- A Green device successfully pinging a Blue device
- A Green device successfully pinging a Red device
- A Red device successfully pinging a Blue device

Homeworks

Assignment 1 – Add two more LANs to the Blue Router

- Add a gray network to the Blue Router
- Add a Purple network to the Blue Router
- RECOMMENDED GRADING CRITERIA
  - Screenshot of GNS3 Workspace with all devices labeled
    - Green network is using randomly generated IP address
    - Grey network is using randomly generated IP address
  - Wireshark Packet Captures
    - Grey end device successfully pinging a Red end device
    - Grey end device successfully pinging a Green end device
    - Green end device successfully pinging a Red end device
    - Green end device successfully pinging a Blue end device

Assignment 2 – Add a new router and LAN

- Add a third router and LAN to the GNS3 Workspace called Network Gray
- The Grey LAN should use a randomly generated IP space
- You will need to configure router interfaces for all routers using the backbone IP space
- You will need to configure static routes for each router
- RECOMMENDED GRADING CRITERIA
  - Screenshot of GNS3 Workspace
    - Gray LAN is using randomly generated IP address space
  - Wireshark PacketCaptures
    - Gray LAN device successfully pinging Red LAN device
    - Gray LAN device successfully pinging Blue LAN device

CHAPTER 22

Dynamic Host Configuration Protocol - MikroTik DHCP Relay

MATHEW J. HEATH VAN HORN, PHD AND JACOB CHRISTENSEN

Typically, larger networks are segmented into smaller LANs. However, one concern that can arise from this is the issue of distributing IP addresses across individual subnets. This is because DHCP discover packets are designed to be broadcasted within local networks. While a network administrator could configure a dedicated DHCP server for each LAN, we demonstrated in Static Networking Part 1 that this can quickly become tedious to configure and maintain. Luckily, DHCP relays can be configured to re-transmit IP requests to remote servers. This way, one server can lease addresses to multiple networks at once. In this chapter, we will configure a DHCP relay with a MikroTik router to service two networks.

Estimated time for completion: 60 minutes

LEARNING OBJECTIVES

- Create three functional LANs
    - RED – PCs
    - BLUE – PCs
    - GRAY – DHCP Server
- Configure a router to serve as a functional DHCP relay
- Configure a DHCP server successfully

PREREQUISITES

- Chapter 16 – Introduction to Routers
- Chapter 17 – IPv4 Addressing
- Chapter 18 – DHCP using Linux

DELIVERABLES

- Screenshot of GNS Workspace with labels

- Screenshot of Wireshark DHCP
    - LAN1 end devices requesting and receiving IP addresses
    - LAN2 end devices requesting and receiving IP addresses

RESOURCES

- MikroTik RouterOS Documentation – "DHCP Relay" – https://wiki.mikrotik.com/wiki/Manual:IP/DHCP_Relay

CONTRIBUTORS AND TESTERS

- Dante A. Rocca, Cybersecurity Student, ERAU-Prescott

Phase I -Build the Network Topology

The following steps are to create a baseline environment for completing the lab. It makes assumptions about learner knowledge from completing previous labs.

By the end of this lab your network should look like the following:

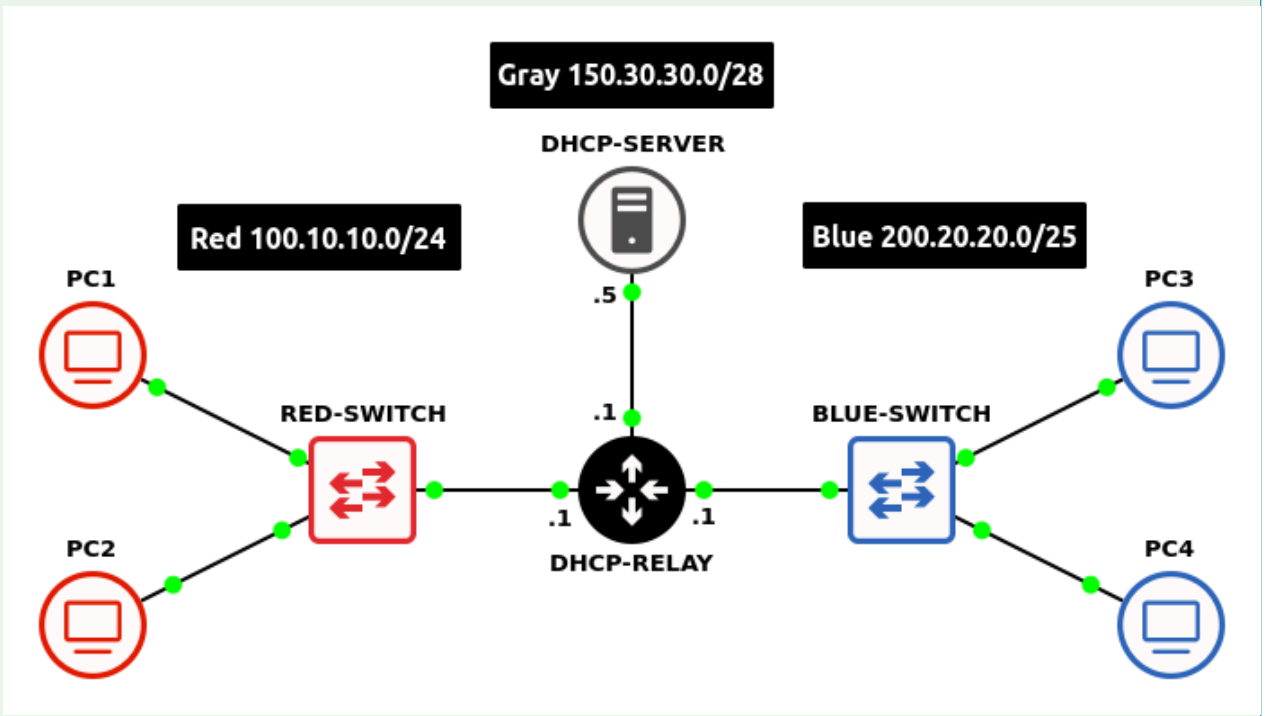


Figure 1 – Final network environment

1. Start GNS3

    1.1. Create a new project: **LAB_08**

2. Build a small network will the following specifications:

    2.1. Subnet – **Red**

        2.1.1. One switch – ***Ethernet switch***

        2.1.2. Two client machines – ***VPCS***

        2.1.3. Minimize wasted address space for ***250 hosts***

Network Information	
Network	**100.10.10.0**
Netmask	**255.255.255.0 (/24)**
Broadcast	**100.10.10.255**
Gateway	**100.10.10.1**
DHCP Lower Bound	**100.10.10.2**
DHCP Upper Bound	**100.10.10.254**

    2.2. Subnet – **Blue**

        2.2.1. One switch – ***Ethernet switch***

        2.2.2. Two client machines – ***VPCS***

        2.2.3. Minimize wasted address space for ***100 hosts***

Network Information	
Network	**200.20.20.0**
Netmask	**255.255.255.128 (/25)**
Broadcast	**200.20.20.127**
Gateway	**200.20.20.1**
DHCP Lower Bound	**200.20.20.2**
DHCP Upper Bound	**200.20.20.126**

    2.3. Subnet – **Gray**

        2.3.1. One DHCP server – ***Ubuntu Server / Windows Server / Tiny Core / MikroTik CHR***

> **NOTE:** This example will use 150.30.30.5 as the server's static IP address.

        2.3.2. Minimize wasted address space for ***10 hosts***

Network Information	
Network	**150.30.30.0**
Netmask	**255.255.255.240 (/28)**
Broadcast	**150.30.30.15**
Gateway	**150.30.30.1**

3. Add a MikroTik router to the workspace

    3.1. Connect all three networks to the router

    3.2. Configure the router with static IP addresses on all active interfaces (Chapter 16, Phase II, Step 4)

> **NOTE:** This example uses the following router ports:
> – ether1 -> Red LAN
> – ether2 -> Blue LAN
> – ether3 -> Gray LAN

4. Label and organize your network as necessary

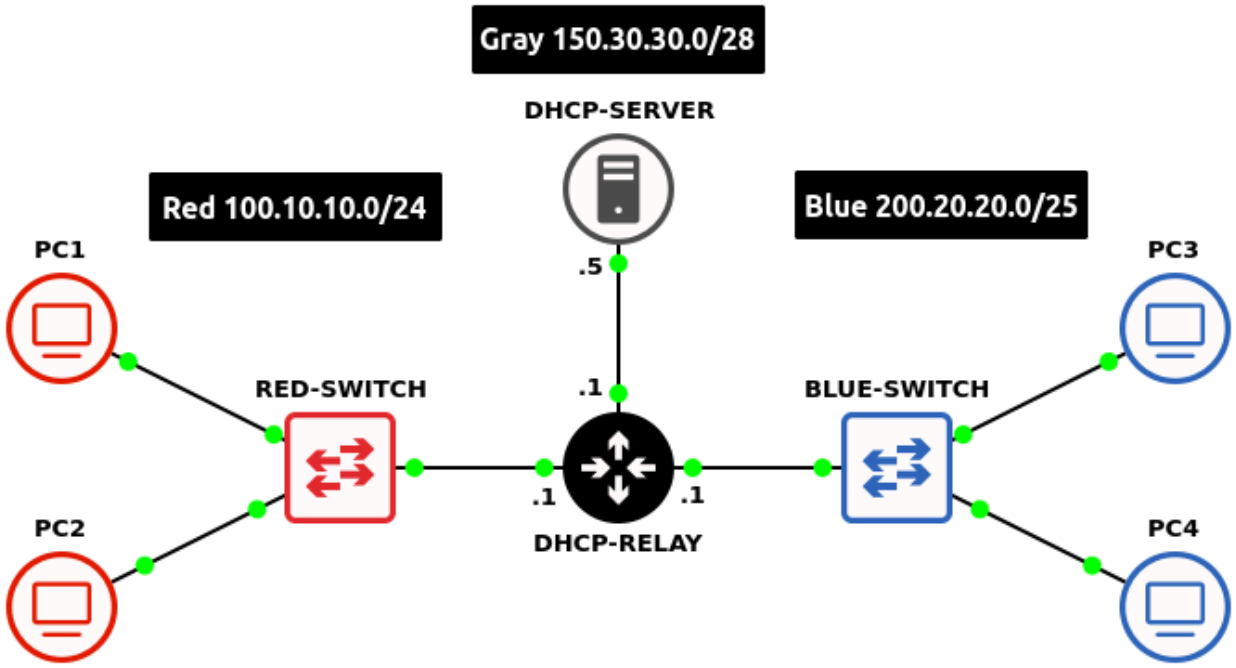


Figure 2 – Final network topology

Phase II – Configure the Router as a DHCP Relay

Since the server is not in the same network as our clients, the router needs to serve as a relay point for the "DHCP Discover" packets.

1. Setup the relay

    1.1. Open the MikroTik console

    1.2. Configure the Red subnet's relay path

```
> ip dhcp-relay add name=Red-Relay interface=ether1 dhcp-server=150.30.30.5 local-address=100.10.10.1 disabled=no
```

Command	Description
ip dhcp-relay	Access the DHCP relay menu.
add name=Red-Relay	Name this group of configuration settings "Red-Relay".
interface=ether1	Assign the ether1 interface to listen for DHCP requests.
dhcp-server=150.30.30.5	DHCP request packets will forwarded to 150.30.30.5.
local-address=100.10.10.1	DHCP response packets will arrive with the address 100.10.10.1.
disabled=no	The default setting is to disable configurations unless otherwise specified.

    1.3. Configure the Blue subnet's relay path

```
> ip dhcp-relay add name=Blue-Relay interface=ether2 dhcp-server=150.30.30.5 local-address=200.20.20.1 disabled=no
```

NOTE: Don't forget to disable the DHCP client that is listening by default on ether1!

```
> ip dhcp-client remove 0
```

2. Verify that the router's settings are configured properly (Figure 3)

```
> ip address print
```

```
> ip dhcp-relay print
```

Phase III – Configure the Linux DHCP Server

We are going to configure our DHCP server in a similar fashion as we've done in previous labs.

Backing up your files...

ALWAYS MAKE A BACKUP! In the following section and the chapters going forward, rewriting configuration files will be commonplace. Whether or not you are someone who makes mistakes, it is always good practice to make backup copies of everything you change. When something goes wrong (and it will), you will be thankful for having these.

Create a backup folder in your home directory:

```
> mkdir ~/backups
```

Verify the directory was created:

```
> ls ~ | grep backups
```

Copy an existing file to the directory:

```
> cp /path/to/file/example.txt ~/backups
```

Verify the backup was made:

```
> ls ~/backups
```

If you ever need to restore your backup:

```
> cp ~/backups/example.txt /path/to/file
```

1. Start the Ubuntu Server VM
2. Configure the server to have a static host address (Figure 4)

```
> sudo vi /etc/netplan/00-installer-config.yaml
```

```
> sudo netplan apply
```

NOTE: Ensure that the server's IP is ***NOT*** the same address as its gateway and is ***WITHIN*** the range of available hosts for your subnet! Also, notice that the configuration provided includes the addition of a **default gateway**. This is important for the server to know how to respond to DHCP request packets. Check if you have a default route set in your server:

```
> ip route
```

```
iako@dhcp-server:~$ ip route
default via 150.30.30.1 dev enp0s3 proto static
150.30.30.0/28 dev enp0s3 proto kernel scope link src 150.30.30.5
iako@dhcp-server:~$ _
```

Figure 5 – Server default gateway

The above image illustrates that the enp0s3 interface is assigned the address 150.30.30.5 on the 150.30.30.0/24 network with a gateway address pointing towards 150.30.30.1.

3. Modify the DHCP daemon configuration file to support all three subnets (Figure 6)

```
> sudo vi /etc/dhcp/dhcpd.conf
```

NOTE: Declare the Gray subnet. Although there are no clients to service here, it is good practice to help the server understand the layout of our network.

4. Ensure that the daemon is enabled and active (Figure 7)

Phase IV – Connect Devices

With the router configured and the DHCP server running, all that is left is to connect devices to see if everything works.

1. Start capturing packets on the PC1-Relay connection

    1.1. In PC1's terminal, request a new IP address

    NOTE: If using a Tiny Core VM, it should automatically request an IP address at startup.

    1.2. Look for the DHCP broadcast packets on Wireshark (you should see the same packets from when you completed the DHCP labs)

1.3. You can check the VPCS's assigned IP address in its console

```
> show ip
```

1.4. If you want to see more traffic, just add more end devices or renew current leases

2. Repeat for the Blue clients

3. Ensure that all three subnets can ping each other and have full connectivity

PHASE V – TROUBLESHOOTING TIPS

There are many moving pieces in this lab and future labs. You might have to do some troubleshooting. Here are some tips offered by our testers.

RTFQ (Read the "Full" Question): This trips up a lot of people. There are many devices, many IP addresses, many different commands. Whether it be configuring Windows, Linux, or GNS3, it is easy to slip up. Read slowly and the lab will work.

SERVER TROUBLESHOOTING: Testers have found that simply restarting the service resolved many of the common errors you many encounter.

```
> systemctl restart example.service
```

However, if a problem persists, you can check the logs for a more detailed explanation:

```
> journalctl -xeu example.service
```

or...

```
> journalctl _PID=1234
```

Obviously replace "example.service" and "1234" with the appropriate service name/process identifier of the daemon you are trying to troubleshoot.

TYPOS: From personal experience, most errors have typically stemmed from hard-to-catch typos. For example, **subnet** looks a lot like **subent** at a glance. Step away and come back later with a fresh mind.

MISCONFIGURED IP ADDRESSES: Does ping not work? Try not to get frustrated. Check, double check, and triple check that your IP addresses and subnet masks all make sense!!! Remember, no client can have an IP address ending in ".0" and no two machines can have the same addresses. Also, ensure that the addresses you are assigning to your hosts are ***within the range*** of your subnet! A /29 network cannot accommodate for the same number of machines as a /24 network.

REBOOT: If all else fails, try rebooting the system. GNS3, VirtualBox, and different OSs can really tax your bare-metal host machine. Sometimes a reboot can help clear up an odd issue.

MOST IMPORTANTLY... do not skip over errors. Read them carefully. If things are not working as they are supposed to, go back to a previous lab where you were successful and think about how you were able to make it work previously.

End of Lab

Deliverables

Three screenshots are required to receive credit for this exercise

- Screenshot of GNS Workspace with all equipment labeled
- Screenshot of Wireshark DHCP
    - LAN1 end devices requesting and receiving IP addresses
    - LAN2 end devices requesting and receiving IP addresses

Homeworks

Assignment 1 – Add another network

- Use a ***randomly generated*** IP address space
- ***Minimize*** wasted address space for 500 hosts
- Connect it to the Relay
- Make sure to change the router settings and DHCP server appropriately
- Label and organize your network as necessary
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 environment with every device labeled
    - Screenshot of Wireshark showing DHCP handshake from a device on the new network
    - Screenshot of the updated DHCP daemon configuration file

Assignment 2 – Add two networks

- Same as Assignment 1 but with two networks

List of Figures for Print Copy

```
[admin@DHCP-RELAY] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS          NETWORK       INTERFACE
0 100.10.10.1/24   100.10.10.0   ether1
1 200.20.20.1/25   200.20.20.0   ether2
2 150.30.30.1/28   150.30.30.0   ether3
[admin@DHCP-RELAY] > ip dhcp-relay print
Columns: NAME, INTERFACE, DHCP-SERVER, LOCAL-ADDRESS
# NAME        INTERFACE  DHCP-SERVER  LOCAL-ADDRESS
0 Red-Relay   ether1     150.30.30.5  100.10.10.1
1 Blue-Relay  ether2     150.30.30.5  200.20.20.1
[admin@DHCP-RELAY] >
```

Figure 3 – Ensuring router is configured properly

```
# This is the network config written by 'Jake M. Christensen'
# 2024.05.12
network:
  ethernets:
    enp0s3:
      optional: true
      dhcp4: false
      addresses:
        - 150.30.30.5/28
      routes:
        - to: default
          via: 150.30.30.1
  version: 2
```

Figure 4 – /etc/netplan/00-installer-config.yaml file

```
# Configuration written by 'Jake M. Christensen'
# 2024.05.13

# -------------------
# DHCPD CONFIGURATION
# -------------------

# Global Parameters
# -----------------
authoritative;
default-lease-time 600;
max-lease-time 7200;


# Gray Subnet Directive
# ---------------------
subnet 150.30.30.0 netmask 255.255.255.240 {
}

# Red Subnet Directive
# --------------------
subnet 100.10.10.0 netmask 255.255.255.0 {
  option routers            100.10.10.1;
  option subnet-mask        255.255.255.0;
  option broadcast-address 100.10.10.255;
  range                     100.10.10.2 100.10.10.254;
}

# Blue Subnet Directive
# ---------------------
subnet 200.20.20.0 netmask 255.255.255.128 {
  option routers            200.20.20.1;
  option subnet-mask        255.255.255.128;
  option broadcast-address 200.20.20.127;
  range                     200.20.20.2 200.20.20.126;
}
```

Figure 6 – /etc/dhcp/dhcpd.conf file

```
iako@server:~$ systemctl status isc-dhcp-server
● isc-dhcp-server.service - ISC DHCP IPv4 server
     Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-02-23 01:37:18 UTC; 9min ago
       Docs: man:dhcpd(8)
   Main PID: 1561 (dhcpd)
      Tasks: 4 (limit: 1013)
     Memory: 4.6M
        CPU: 6ms
     CGroup: /system.slice/isc-dhcp-server.service
             └─1561 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhcpd.pid -cf /etc/dh

Feb 23 01:37:18 server sh[1561]: PID file: /run/dhcp-server/dhcpd.pid
Feb 23 01:37:18 server dhcpd[1561]: Wrote 0 leases to leases file.
Feb 23 01:37:18 server sh[1561]: Wrote 0 leases to leases file.
Feb 23 01:37:18 server dhcpd[1561]: Listening on LPF/enp0s3/08:00:27:e0:58:cb/150.30.30.0/24
Feb 23 01:37:18 server sh[1561]: Listening on LPF/enp0s3/08:00:27:e0:58:cb/150.30.30.0/24
Feb 23 01:37:18 server dhcpd[1561]: Sending on   LPF/enp0s3/08:00:27:e0:58:cb/150.30.30.0/24
Feb 23 01:37:18 server sh[1561]: Sending on   LPF/enp0s3/08:00:27:e0:58:cb/150.30.30.0/24
Feb 23 01:37:18 server dhcpd[1561]: Sending on   Socket/fallback/fallback-net
Feb 23 01:37:18 server sh[1561]: Sending on   Socket/fallback/fallback-net
Feb 23 01:37:18 server dhcpd[1561]: Server starting service.
lines 1-21/21 (END)
```

Figure 7 – DHCP server is up

CHAPTER 23

Domain Name System Part 1- Authoritative DNS

JACOB CHRISTENSEN

In previous labs, communication between devices was verified via pinging IP addresses. With the addition of a Domain Name System (DNS) server to our network, these computers can be referenced by a more human-friendly name rather than a hard-to-remember string of numbers.

This lab will demonstrate the configuration and implementation of a basic primary authoritative (local) DNS server for multiple LANs.

Estimated time for completion: 60 minutes

NOTE: There is a large amount of explaining in this lab. Sometimes the dot "." has different meanings depending on where it is located:

- An octet separator as in 8.8.8.8
- A domain name separator as in red.net
- A Fully Qualified Domain Name (FQDN) indicator as in red.net.
- As a separator between ideas. Idea one. Idea two. Idea three

Experienced cyber operators understand the context of the dot placement and apply its meaning appropriately. Beginners may get confused. Therefore when testers reported confusion we added brackets around the dots to enhance clarity. e.g. in the following examples the domain name separator dots are placed in brackets

e.g. 80[.]30[.]89[.]in-addr[.]arpa. – *Ends in a dot denoting a FQDN*

e.g. 80[.]30[.]89[.]in-addr[.]arpa – *Does not end in a dot denoting it is not a FQDN*

LEARNING OBJECTIVES

- Learn how DNS works on an enterprise network
- Demonstrate how to configure zone files

PREREQUISITES

- One (1) Ubuntu Server VM with Bind9 installed (created in Chapter 7)

- Three (3) Tiny Core Linux VMs (created in Chapter 5)

DELIVERABLES

- Screenshot of the GNS3 working environment
- Screenshot of PC1 successfully forward resolving the details of PC2
- Screenshot of PC1 successfully reverse resolving the details of PC3
- Screenshot of PC3 successfully pinging PC2 by name instead of IP

RESOURCES

- BIND9 Administrator Reference Manual – https://bind9.readthedocs.io/en/v9.18.16/
- MikroTik RouterOS Documentation – https://help.mikrotik.com/docs/display/ROS/RouterOS

CONTRIBUTORS AND TESTERS

- Mathew J. Heath Van Horn, PhD, ERAU-Prescott
- Kyle Wheaton, Cybersecurity Student, ERAU-Prescott

Phase I – Building the Network Topology

The following steps are to create a baseline network for completing this chapter. It makes assumptions about learner knowledge from completing previous labs.

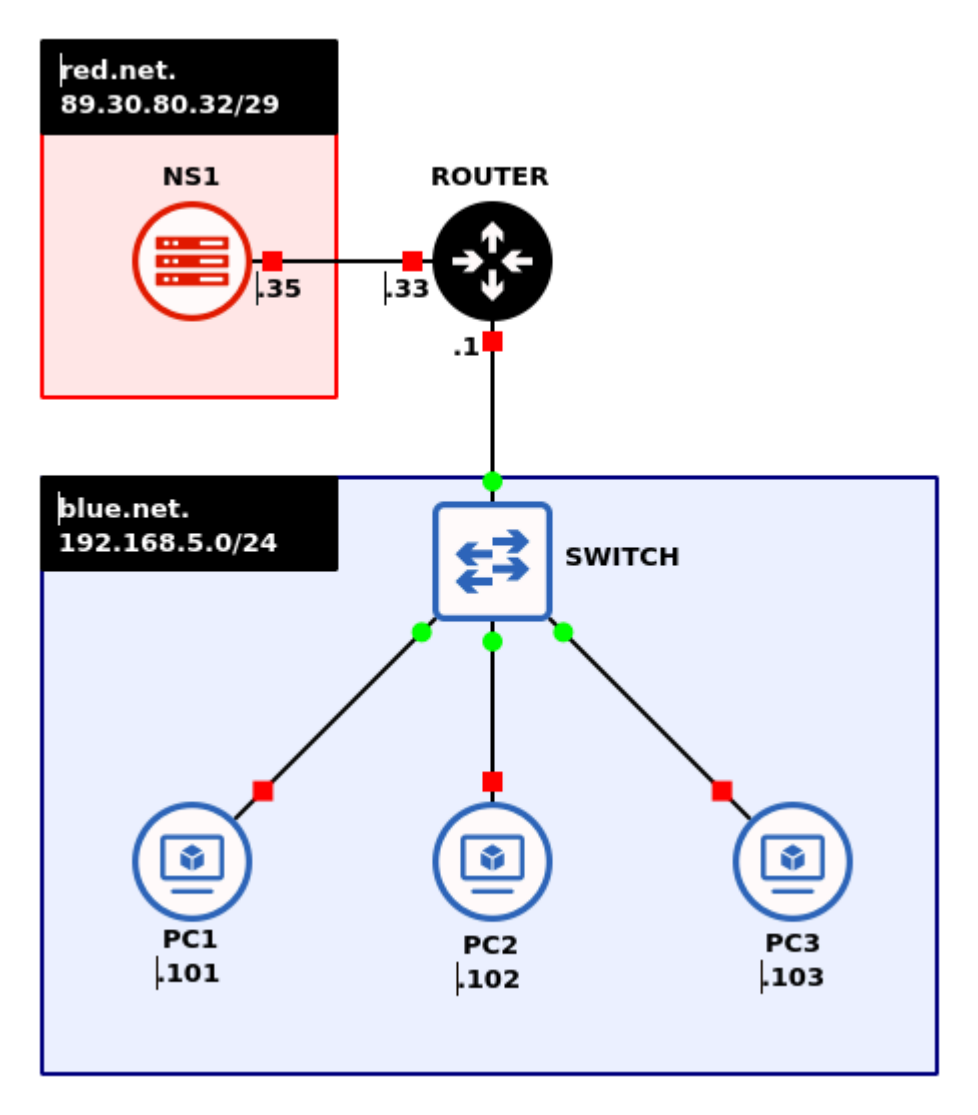


Figure 1 – The environment we are building

1. Preliminary step in VirtualBox

    1.1. Create/clone 3 TinyCore virtual machines

    > **NOTE:** For easy reference you can name them PC1, BLUE1, etc.

    1.2. Create/clone a fresh copy of the Linux Server VM with BIND9 installed

2. Start GNS3

2.1. Create a new project: **LAB_09**

2.2. Add the VMs created in Step 1 (Chapter 6)

3. Build a subnet – ***blue.net*** – with the following specifications:

3.1. Use a ***randomly*** generated IPv4 network address space

3.2. Use a network address that ***minimizes*** wasted IPs for 250 hosts

3.3. Three client machines – ***Tiny Core Linux***

> **NOTE:** You may use GNS3's VPCS instead of Tiny Core, but this is not recommended since they do not have the tools dig or nslookup installed. These are tools that are necessary to help troubleshoot DNS. Furthermore, you may need to flush DNS caches to reliably see the Wireshark traffic. It is just easier to use Tiny Core end-user devices from the beginning.

3.4. One Switch – ***Ethernet switch***

3.5. Label each Tiny Core client with the static IP address

> **NOTE:** This example uses the following IPv4 addresses:
> – Network: **192.168.5.0/24**
> – Gateway: **192.168.5.1**
> – PC1: **192.168.5.101**
> – PC2: **192.168.5.102**
> – PC3: **192.168.5.103**

4. Build a second subnet – ***red.net*** – with the following specifications:

4.1. Use a ***randomly*** generated IPv4 network address space

4.2. Use a network address that ***minimizes*** wasted IPs for 5 hosts

4.3. One DNS server – ***Ubuntu Server (NS1)*** with the static IP set

> **NOTE:** the netplan configuration file will always be a .YAML file, however, the name may change between releases or user modifications. This screenshot is using a Ubuntu 22.04.X LTS .YAML file. Adjust as necessary.
> **NOTE**: This example uses the following IP addresses:
> – Network: **89.30.80.32/29**

– Gateway: **89.30.80.33**
– NS1: **89.30.80.35**

```
> vi /etc/netplan/00-installer-config.yaml
```

```
# This is the network config written by 'Dr. HVH for DNS1'
network:
  ethernets:
    enp0s3:
      dhcp4: false
      addresses:
        - 89.30.80.35/29
      routes:
        - to: default
          via: 89.30.80.33
  version: 2
```

Figure 2 – NS1 static IP configuration file

5. Add a MikroTik router to the workspace

    5.1. Connect both networks to the router

    5.2. Configure the router with static IPs on all active interfaces

NOTE: This example uses the following IP addresses:
– ether1 – **89.30.80.33/29**
– ether2 – **192.168.5.1/24**

6. Label and organize your network as necessary. When complete it should look like Figure 1

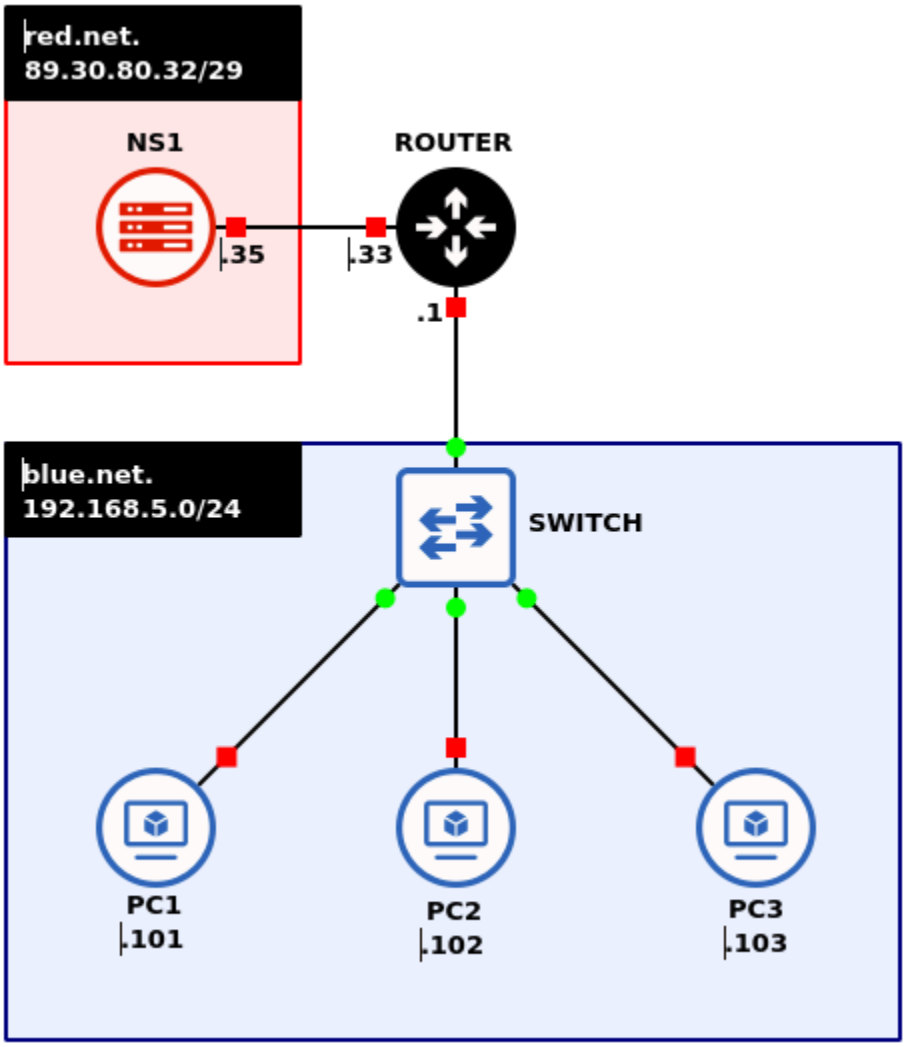


Figure 1 – The environment we are building

NOTE: Notice the addition of **[.]net** in our labels. These will be our local domain names for each subnet.

Phase II – Configuring the Tiny Core Linux Clients

Before we can map static DNS entries in our server, we must give our clients static IP addresses and hostnames. If you are not using Tiny Core (or already know how to do this with a different method) skip this section.

1. Configure the first client *PC1*

    1.1. Navigate to PC1's console

1.2. Modify the system startup script to execute the ***sethostname*** binary on boot

> **NOTE:** ***vi*** is the only text editor installed by default in Tiny Core.

```
> vi /opt/bootsync.sh
```

1.2.1. Modify the **/usr/bin/sethostname** line and replace **box** with **PC1**

```
/usr/bin/sethostname PC1
```

```
#!/bin/sh
# put other system startup commands here, the boot process will wait until they
# Use bootlocal.sh for system startup commands that can run in the background
# and therefore not slow down the boot process.
/usr/bin/sethostname PC1
/opt/bootlocal.sh &
```

Figure 3 – Setting static IP for PC1

1.2.2. Save (write) and exit the editor

1.3. Configure the first user with a static IP address (**192.168.5.101**)

1.3.1. Modify the local startup script to include our custom interface configuration by typing

```
> vi /opt/bootlocal.sh
```

1.3.2. Add the following items to the startup script

1.3.2.1. Kill the DHCP client program

```
pkill udhcpc
```

1.3.2.2. Set the static IP address, netmask, and broadcast address for the ***eth0*** interface

```
ifconfig eth0 192.168.5.101 netmask 255.255.255.0
broadcast 192.168.5.255 up
```

1.3.2.3. Add a default gateway pointing to the router's inward-facing interface

```
route add default gw 192.168.5.1
```

1.3.2.4. Add the local domain name ***blue.net*** to the local DNS configuration file

```
echo "domain blue.net" > /etc/resolv.conf
```

1.3.2.5. Append the address for the primary nameserver (NS1) to the local DNS configuration file

```
echo "nameserver 89.30.80.35" >> /etc/resolv.conf
```

1.3.2.6. Use Figure 4 for configuration reference

```
#!/bin/sh
# put other system startup commands here
pkill udhcpc
ifconfig eth0 192.168.5.101 netmask 255.255.255.0 broadcast 192.168.5.255 up
route add default gw 192.168.5.1
echo "domain blue.net" > /etc/resolv.conf
echo "nameserver 89.30.80.35" >> /etc/resolv.conf
```

Figure 4 – Startup script modifications

1.3.2.7. Save and exit the editor

1.3.3. Reboot the machine to apply the changes

```
> sudo reboot
```

1.3.4. Verify that the changes went into effect

```
> ifconfig
```

```
tc@PC1:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:11:93:42
          inet addr:192.168.5.101  Bcast:192.168.5.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Figure 5 – Checking to see if the settings took effect

2. Repeat Step 1 above for Tiny Core clients (PC2 and PC3)

3. Ensure full network connectivity by pinging all the devices from PC1 before continuing to the next section

Phase III – Configuring the Authoritative DNS Server

When you built the Linux server VM for the first time, the software BIND9 should have been installed along with additional utilities and services. Berkley Internet Name Domain, or BIND, is the most popular software suite for interacting with the DNS protocol on Linux systems. If it is not installed, you'll have to do that now.

1. Start **ns1.red.net** and login

2. Modify the machine's hostname to **ns1**

2.1. Open the terminal and assign a new permanent device name using Systemd's hostnamectl binary

```
> sudo hostnamectl hostname ns1
```

NOTE: You can check the hostname in Linux by typing

```
> hostname
```

2.2. Add the new name in the ***hosts*** file in the ***/etc*** directory to prevent name resolution conflicts

```
> vi /etc/hosts
```

```
127.0.0.1 localhost
127.0.0.1 ns1 # <-- new hostname is bound with localhost
              #     address

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Figure 6 – Modifying the /etc/hosts file to prevent resolution conflict in the future

> **NOTE:** If you notice that commands executed with ***sudo*** have slowed to a crawl, you may have forgotten to do this step.

2.3. Reboot the machine to apply changes

```
> reboot
```

3. Assign a static IP address, default gateway, primary DNS server, and local domain name

3.1. Configure the server's netplan YAML file

```
> sudo vi /etc/netplan/00-installer-config.yaml
```

```
# This is the network config written by 'Jake M. Christensen'
# 2023.03.30
network:
  ethernets:
    enp0s3:
      optional: true
      dhcp4: false
      addresses:
        - 89.30.80.35/29
      routes:
        - to: default
          via: 89.30.80.33
      nameservers:
        addresses:
          - 127.0.0.1
          - 89.30.80.35
        search:
          - red.net
  version: 2
```

Figure 7 – Modifying the YAML again

3.2. Apply the new configuration

```
> netplan apply
```

3.3. Verify that the IP address was established

```
> ip route
```

```
iako@ns1:~$ ip route
default via 89.30.80.33 dev enp0s3 proto static
89.30.80.32/29 dev enp0s3 proto kernel scope link src 89.30.80.35
iako@ns1:~$ _
```

Figure 8 – Verifying the IP route was successfully established

3.4. Verify that the nameserver information is correctly pointing towards itself (127.0.0.1) and that the current domain is **red[.]net**

```
> resolvectl status
```

```
iako@ns1:~$ resolvectl status
Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: stub

Link 2 (enp0s3)
    Current Scopes: DNS
         Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 127.0.0.1
       DNS Servers: 127.0.0.1 89.30.80.35
        DNS Domain: red.net
iako@ns1:~$ _
```

Figure 9 – Verifying that the nameserver information is correct

4. Stop and disable Systemd's time synchronization daemon to avoid unnecessary DNS traffic

```
> systemctl stop systemd-timesyncd.service
```

```
> systemctl disable systemd-timesyncd.service
```

5. Open the DNS daemon configuration file using any text editor you prefer

```
> vi /etc/bind/named.conf.options
```

5.1. At the start of the file, before the options directive, add the following access control list (ACL)

```
// Written by: Jake Christensen
// 2024.03.30

// Access Control List
// Only trusted networks may access this service

acl labnets {
  127.0.0.1;       // localhost
  89.30.80.32/29; // localnet (red.net.)
  192.168.5.0/24; // blue.net.
};
```

Figure 10 – The addition of ACLs to the configuration file

NOTE: ACLs are important for preventing unauthorized access to machines or services. In this case, all subnets can use this server for DNS queries, however, clear specification is always good practice.

5.2. In the same file, add the following commands in the options directive

```
// Server configuration parameters

options {
        directory "/var/cache/bind";

        version "not currently available";
        recursion no;
        empty-zones-enable no;

        allow-query { labnets; };
        allow-query-cache { none; };
        allow-recursion { none; };
        allow-transfer { none; };

        dnssec-validation no;

        auth-nxdomain no;
        listen-on port 53 { 127.0.0.1; 89.30.80.35; };
        listen-on-v6 { none; };
};
```

Figure 11 – Adding options to the same configuration file

NOTE: There may already be configuration options here by default. Feel free to delete or modify them as you see fit.

Command	Description
version	Specifies the version number of the BIND9 server to return in response to a version query. Official BIND9 documentation suggests that this be set to "not currently available" to avoid detailed service enumeration from threat actors.
recursion	Defines whether recursion (and caching of queries as a result of recursion) are allowed.
empty-zones-enable	Enables or disables all empty zones. Setting to "no", allows the mapping of private IP addresses to hostnames (such as 192.168.x.x) if they are used on the network. By default, this is set to "yes".
allow-query	Defines which networks, if any, are allowed to query its service.
allow-query-cache	Defines which networks, if any, are allowed to query cached domains.
allow-recursion	Defines which networks, if any, are allowed to access recursion services on the server.
dnssec-validation	Defines whether DNS validation is enabled.
listen-on	Defines which ports and addresses to listen on and respond to DNS queries.

5.3. Save (write) and exit the editor

5.4. Verify that the file was configured correctly

```
> named-checkconf /etc/bind/named.conf.options
```

NOTE: No response indicates that no errors were detected. Otherwise, read the error carefully and fix syntax as necessary.

6. Modify the service zone configuration file to specify the domains our server will have authority over

```
> vi /etc/bind/named.conf.local
```

NOTE: Two types of zones must be created for each domain: forward lookup zones and reverse lookup zones. The former translates domain names to IP addresses while the latter does... the reverse. AKA, IP addresses to domain names.

6.1. Add the following forward lookup zones to the configuration file

```
// FORWARD ZONE FOR RED.NET.
zone "red.net." {
  type master;
  file "/var/lib/bind/db.red.net";
};

// FORWARD ZONE FOR BLUE.NET.
zone "blue.net." {
  type master;
  file "/var/lib/bind/db.blue.net";
};
```

Figure 12 – Add forward lookup zones for red and blue networks

NOTE: Notice how the zone declarations end with a dot [.]. This is a **fully qualified domain name (FQDN)**. Make sure to pay special attention when FQDNs are being used going forward. Forgetting to include the period is an easy typo, but it can cause a major headache when debugging later.

Command	Description
zone	FQDN of the domain the server will have authority over.
type	Declares whether this machine is the primary (master) or secondary (slave) server for this zone.
file	Declares the location of the database file for this zone.

6.2. In the same file, add the following reverse lookup zones

```
// REVERSE ZONE FOR RED.NET.
zone "80.30.89.in-addr.arpa." {
  type master;
  file "/var/lib/bind/db.red.net.rev";
};

// REVERSE ZONE FOR BLUE.NET.
zone "5.168.192.in-addr.arpa." {
  type master;
  file "/var/lib/bind/db.blue.net.rev";
};
```

Figure 13 – Add reverse lookup zones for the red and blue networks by IP addresses

NOTE: Notice how the first tree octets in the zone declaration are in***LITTLE ENDIAN*** order. This is known as a reverse ARPA address. Make sure to double-check everything for typos. In reverse ARPA addresses, you only need to declare the octets that will be static. For instance, the correct syntax for the network **176[.]55[.]0[.]0/20** would be **55[.]176[.]in-addr[.]arpa[.]** since host addresses can range from **176[.]55[.]0[.]1** to **172[.]55[.]15[.]254**. Because this CIDR mask allows for such a large range of host addresses, the third octet can vary in value, so its reverse ARPA address must be adjusted in response.

Little Endian vs Big Endian

Endian can be confusing to people who have never programmed computer memory space. It means which end of a number goes first. Remember back to second grade, numbers have a one's place, ten's place, hundred's place, etc...

Little Endian – Write the number starting at the little end (the one's place).

Big Endian – Write the number starting at the big number (the highest number place).

Computer memory and programs use First In First Out (FIFO) and First In Last Out (FILO) mechanisms for various reasons. Reasons that are beyond the scope of these labs. It is enough to know that FIFO and FILO exist, making Endian notation necessary.

SIMILE TIME: Endian is like a crowded elevator with two doors; front and back. The following people enter the elevator from the front doors.

– Miracle (Enters first, stands at the back of the elevator)
– Maribel
– Scott
– Rory
– Halle
– Lucian (Enters last, stands at the front of the elevator)

If they travel to the 5th floor, the front doors open and Lucian will exit the elevator first and Miracle will exit the elevator last (FILO).

However, if they travel to the 6th floor, the rear doors open and Miracle will exit the elevator first and Lucian will exit the elevator last (FIFO).

Now, let's say the people must exit the elevator in a particular order, so the usher needs to know which order to have the folks enter the elevator. The usher knows that the 5th and 6th floors open differently, so when the ordered list is prepared, the usher will use Little Endian notation for the 5th floor or Big Endian notation for the 6th floor.

Back to computers: Using an IP in Endian notation. We can say our PC has an IP address of 192.168.1.5 so that translates to:
– Big Endian Notation 192.168.1.5
– Little Endian Notation 5.1.168.192

6.3. Save and exit the configuration file editor

6.4. Verify the file was configured correctly using the following command

```
> named-checkconf /etc/bind/named.conf.local
```

NOTE: Again, no response indicates that no errors were detected. Otherwise, fix file syntax as necessary. Pay attention to FQDNs.

Phase IV – Configure Zone Resource Record (RR) Files

Each zone specified in the previous section needs a dedicated database to store domain information. Several have already been provided in the */etc/bind* directory. The database files can be identified by the ones starting with "db" (database) and are loaded in the "*named.conf.default-domains*" configuration file. We can use these as a starting point for making our own RR files.

1. Create a new forward data file for the Red subnet

1.1. View the contents of the bind directory to see what database files are available

```
> ls -l /etc/bind
```

1.2. Copy any db file of your choice

```
> cp /etc/bind/db.empty /var/lib/bind/db.red.net
```

NOTE: Ensure the renamed file **matches** the one specified in Phase III when we declared our zones. We used:
db.red.net
db.blue.net

1.3. Configure the newly created file to act as the Red subnet's forward lookup data file

```
> vi /var/lib/bind/db.red.net
```

1.3.1. Rename the comment at the start of the file to reflect its new purpose

NOTE: In these files, the semicolon denotes a note instead of a command.

```
; BIND9 forward data file for red.net.
```

1.3.2. Below $TTL, add the following directive

```
$ORIGIN red.net.
```

NOTE: This will append the base domain name to any domain that is not terminated with a dot [.]. For example, the string **ns1** will be interpreted as ns1[.]red[.]net[.]. Conversely, the string **ns1[.]** will be read as only **ns1[.]**. This is why paying attention to FQDNs is important, for strings such as **ns1[.]red[.]net** (without the terminating dot) will be read as **ns1[.]red[.]net[.]red[.]net[.]** by the server.

1.3.3. In the Start of Authority (SOA) declaration, overwrite the local host information with the master DNS server domain

```
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
```

1.3.4. Increment the Serial value by one

```
2               ; Serial
```

> **NOTE:** Whenever this file is edited in the future, the serial number should be incremented again.

1.3.5. Leave the Refresh/Retry/Expire/Negative Cache TTL values as their defaults for now

1.3.6. Replace the current name server (NS) entry with all available domains (in this case, there is only one server)

```
@       IN      NS      ns1.red.net.
```

1.3.7. Since we only have one device in the Red zone, we need to map it with its static layer 3 internet (IN) address (A)

```
ns1     IN      A       89.30.80.35
```

1.3.8. Save the file and exit the editor

1.3.9. Use this image for configuration reference

```
;
; BIND9 Forward Data File for red.net.
; Written by: Jacob Christensen
; 2024.03.30
;
$TTL    86400
$ORIGIN red.net.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              4         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.
ns1     IN      A       89.30.80.35
```

Figure 14 – Configuring the forward data file for red.net.

1.4. Verify that the file was configured correctly and correct any errors as necessary

```
> named-checkzone red.net. /var/lib/bind/db.red.net
```

2. Create a new reverse data file for the Red subnet

2.1. Copy any db file of your choice

```
> cd /var/lib/bind
```

```
> cp db.red.net db.red.net.rev
```

NOTE: We recommend the new file name ends with **[.]rev** to quickly identify it as a reverse lookup database file.

2.2. Open the new reverse data file with any text editor

```
> vi /var/lib/bind/db.red.net.rev
```

2.2.1. Rename the comment at the start of the file to reflect its new purpose

```
; BIND9 reverse data file for red.net.
```

2.2.2. Below $TTL add the following directive

```
$ORIGIN 80.30.89.in-addr.arpa.
```

NOTE: Once again, the octets are in ***LITTLE ENDIAN*** order. This dictates the base internet address (***in-addr***) that will be appended to any incomplete IP addresses later. For example, the IP address **35** will be interpreted as **35[.]80[.]30[.]89[.]in-addr[.]arpa[.]** otherwise known as **89[.]30[.]80[.]35**.

2.2.3. In the Start of Authority (SOA) declaration, specify the master DNS server domain

```
@      IN      SOA     ns1.red.net. hostmaster.red.net. (
```

2.2.4. Increment the Serial value by one

```
2       ; Serial
```

2.2.5. Leave the Refresh/Retry/Expire/Negative Cache TTL values as their default for now

2.2.6. Replace the current NS entry with the master DNS server domain

```
@      IN      NS      ns1.red.net.
```

2.2.7. Map the server's currently assigned IP addresses (last octet) with its associated FQDN

```
35    IN      PTR     ns1.red.net.
```

NOTE: If you are mapping IP addresses with a shorted reverse ARPA address, then do not forget to include both host octets. For example, the address **176[.]55[.]2[.]15/20** would be represented as **55[.]176[.]in-addr[.]arpa[.]** in the $ORIGIN directive and mapped as **15[.]2 IN PTR pc.example[.]com[.]**.

2.2.8. Save the file and exit the editor

2.2.9. Use this image for configuration reference

```
;
; BIND9 Reverse Data File for red.net.
; Written by: Jacob Christensen
; 2024.03.30
;
$TTL    86400
$ORIGIN 80.30.89.in-addr.arpa.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.
35      IN      PTR     ns1.red.net.
```

Figure 15 – Reverse data file for red.net.

2.3. Verify that the file was configured correctly

```
> named-checkzone 80.30.89.in-addr.arpa. /var/lib/bind/db.red.net.rev
```

3. Repeat the above process and make another set of forward and reverse lookup files for **blue.net**

> **NOTE:** Don't get confused. The name server is in the red network. That is why the SOA and NS will remain unchanged.

3.1. /var/lib/bind/db.blue.net

```
; _
; BIND9 Forward Data File for blue.net.
; Written by: Jacob Christensen
; 2024.03.30
;
$TTL    86400
$ORIGIN blue.net.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.
pc1     IN      A       192.168.5.101
pc2     IN      A       192.168.5.102
pc3     IN      A       192.168.5.103
```

Figure 16 – Forward data file for blue.net.

```
> named-checkzone blue.net. /var/lib/bind/db.blue.net
```

3.2. /var/lib/bind/db.blue.net.rev

```
;
; BIND9 Reverse Data File for blue.net.
; Written by: Jacob Christensen
; 2024.03.30
;
$TTL    86400
$ORIGIN 5.168.192.in-addr.arpa.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.
101     IN      PTR     pc1.blue.net.
102     IN      PTR     pc2.blue.net.
103     IN      PTR     pc3.blue.net.
```

Figure 17 – Reverse data file for blue.net.

```
> named-checkzone 5.168.192.in-addr.arpa. /var/lib/bind/db.blue.net.rev
```

4. Modify the permissions for the zone files

    4.1. Change the owner and group from ***root*** to the ***bind*** user

```
> chown bind:bind /var/lib/bind/db.*
```

    4.2. Change the file permissions to have read and write access for both the owner and group

```
> chmod 664 /var/lib/bind/db.*
```

5. Start the DNS daemon and ensure that all zones loaded properly

```
> systemctl daemon-reload
```

```
> systemctl enable named.service
```

```
> systemctl restart named.service
```

```
> systemctl status named.service
```

```
iako@ns1:/var/lib/bind$ sudo systemctl status named.service
● named.service - BIND Domain Name Server
     Loaded: loaded (/lib/systemd/system/named.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-03-31 00:56:04 UTC; 6s ago
       Docs: man:named(8)
    Process: 1353 ExecStart=/usr/sbin/named $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 1354 (named)
      Tasks: 4 (limit: 1013)
     Memory: 4.6M
        CPU: 18ms
     CGroup: /system.slice/named.service
             └─1354 /usr/sbin/named -u bind

Mar 31 00:56:04 ns1 named[1354]: zone localhost/IN: loaded serial 2
Mar 31 00:56:04 ns1 named[1354]: zone 127.in-addr.arpa/IN: loaded serial 1
Mar 31 00:56:04 ns1 named[1354]: zone 5.168.192.in-addr.arpa/IN: loaded serial 2
Mar 31 00:56:04 ns1 named[1354]: zone 255.in-addr.arpa/IN: loaded serial 1
Mar 31 00:56:04 ns1 named[1354]: zone 80.30.89.in-addr.arpa/IN: loaded serial 3
Mar 31 00:56:04 ns1 named[1354]: zone blue.net/IN: loaded serial 2
Mar 31 00:56:04 ns1 named[1354]: zone red.net/IN: loaded serial 4
Mar 31 00:56:04 ns1 named[1354]: all zones loaded
Mar 31 00:56:04 ns1 systemd[1]: Started BIND Domain Name Server.
Mar 31 00:56:04 ns1 named[1354]: running
iako@ns1:/var/lib/bind$ _
```

Figure 18 – BIND is running without errors

Phase V – Wireshark Captures

Now that our network is set up, let's watch live packet captures to see what DNS queries look like in action.

1. Start a Wireshark capture between NS1 and the router
2. Open the terminal of PC1
    2.1. Test the service by performing a DNS query on PC2

```
> nslookup pc2
```

```
tc@PC1:~$ nslookup pc2
Server:     89.30.80.35
Address 1: 89.30.80.35 ns1.red.net

Name:       pc2
Address 1: 192.168.5.102 pc2.blue.net
tc@PC1:~$ _
```

Figure 19 – Successful nslookup query

> **NOTE:** Since they are both in the same network, the***blue[.]net*** domain does not need to be specified. To communicate with a device outside the LAN, the full domain name of the recipient is needed.

2.2. In Wireshark, you should see a ***Standard query*** IPv4 (A) packet sent to NS1 and a ***Standard query response*** packet containing the resolution

```
Standard query 0x94b8 A pc2.blue.net
Standard query 0x06a4 AAAA pc2.blue.net
Standard query response 0x94b8 A pc2.blue.net A 192.168.5.102
```

Figure 20 – successful nslookup query as viewed on Wireshark

> **NOTE:** You may see additional IPv6 queries (***AAAA***) in addition to IPv4. You can safely ignore these since we have not configured IPv6 on our DNS server.

2.3. You can also perform a reverse DNS lookup with the ***nslookup*** binary to discover the hostname of a machine given its IP address

```
> nslookup 192.168.5.102
```

```
tc@PC1:~$ nslookup 192.168.5.102
Server:    89.30.80.35
Address 1: 89.30.80.35 ns1.red.net

Name:      192.168.5.102
Address 1: 192.168.5.102 pc2.blue.net
tc@PC1:~$ _
```

Figure 21 – Successful reverse nslookup

2.4. Again, you should see a ***Standard query*** hostname (PTR) packet sent to the NS1 and a ***Standard query response*** packet containing the mapped IPv4 address

```
Standard query 0x8202 PTR 102.5.168.192.in-addr.arpa
Standard query response 0x8202 PTR 102.5.168.192.in-addr.arpa PTR pc2.blue.net
```

Figure 22 – reverse nslookup on Wireshark

3. From PC1, ping the DNS server

```
> ping ns1.red.net
```

3.1. If the full domain name is not given, the local name will be appended instead and the name will not be resolved

```
> ping ns1
```

```
Standard query 0x0089 A ns1.blue.net
Standard query 0x898d AAAA ns1.blue.net
Standard query response 0x0089 No such name A ns1.blue.net SOA ns1.red.net
```

Figure 23 – nslookup fails

> **NOTE:** Notice how the domain ***ns1[.]blue[.]net*** was queried instead of ***ns1[.]red[.]net***.

4. Congratulations! You have successfully configured an authoritative DNS server with static IP addresses in your network

End of Lab

Deliverables

- Screenshot of the GNS3 Working environment
- Screenshot of PC1 successfully forward resolving the details of PC2
- Screenshot of PC1 successfully reverse resolving the details of PC3
- Screenshot of PC3 successfully pinging PC2 by name instead of IP

Homeworks

Assignment 1 – Add a Green network to the router:

- Use a random IP address space
- Use two end devices
- Plan for the addition of 313 machines
- Modify the DNS server to include the new subnet

Assignment 2 – Add a Purple network to the router:

- Use a random IP address space
- Use two end devices

- Minimize wasted address space for 245 machines
- Modify the DNS server to include the new subnet zones

RECOMMENDED GRADING CRITERIA

- Wireshark screenshots of successful purple.net forward and reverse domain resolution
- Wireshark screenshots of successful green.net forward and reverse domain resolution
- Screenshot of GNS3 workspace that is neatly organized and labeled

Figures for Printed Document

There are no clickable figures in the digital edition which need to be placed here for the print edition.

CHAPTER 24

Domain Name System Part 2 - Forwarding DNS

JACOB CHRISTENSEN

In the last lab, we configured our network so our devices could communicate via human-readable names rather than IP addresses. However, we did not build the ability to interact with Internet domains. In this chapter, we will reconfigure our DNS server to forward non-authoritative domain queries (such as www.google.com) to a public resolver and further extend our network's capabilities.

Estimated time for completion: 25 minutes

LEARNING OBJECTIVES

- Demonstrate how to configure and implement a forwarding DNS server for Internet communications
- Learn how to implement a network address translation (NAT) rule between a LAN and WAN

PREREQUISITES

- Chapter 23 – Domain Name System Part 1

DELIVERABLES

- Screenshot of RED1
  - Successful nslookup of Google
  - Successful ping of Google
- Screenshot of BLUE1
  - Successful nslookup of Google
  - Successful ping of Google
- Screenshot of the dig trace of Google
- Screenshot of GNS3 environment

RESOURCES

- BIND9 Administrator Reference Manual – https://bind9.readthedocs.io/en/v9.18.16/

- MikroTik RouterOS Documentation – https://help.mikrotik.com/docs/display/ROS/RouterOS

CONTRIBUTORS AND TESTERS

- Mathew J. Heath Van Horn, PhD, ERAU-Prescott
- Kyle Wheaton, Cybersecurity Student, ERAU-Prescott

DNS Hierarchy Explained

When you look up a website for the first time, you may notice it will load considerably slower when compared to subsequent revisits. This is because your computer does not yet know the web server's IP address, and must ask around first before it can start making HTTP requests. However, once the domain-to-IP translation is known, various entities such as your browser, computer, router, and ISP servers all have local storage reserved for caching this information, which speeds up query resolutions. This is a feature for end-user convenience and saving bandwidth, but how does your DNS server know where to look when it receives an unknown domain?

Reference the diagram below while you follow these steps to understand the process.

1. The client device makes a DNS query to the LAN's server for the domain <erau.edu.>
2. The domain is neither cached nor listed under any of its zones, so it forwards the request to a DNS resolver
3. The Resolver has a built-in list of all of the root DNS servers
    - There are 13 root <.> servers in the world. They are named sequentially from <a.root-servers.net> to <m.root-servers.net>
    - Each one knows the location of the Top Level Domain (TLD) servers
4. One of the root servers will provide the address of the <edu> TLD server, which is sent back to the Resolver
5. The Resolver will query the <edu> TLD server for the <erau> subdomain
6. The subdomain DNS identifies the webserver for <erau.edu.> and sends this information back to the end user

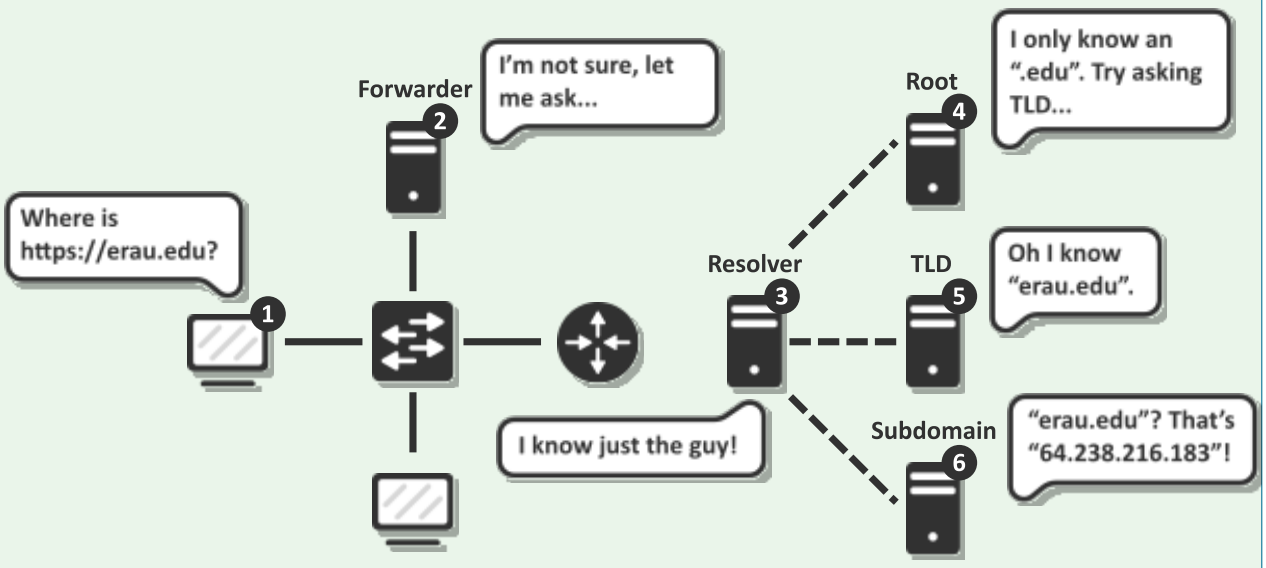


Diagram used with permission of Cody Shinkyu Park

Now that we understand how a Forwarding DNS server works, let's build one for our network.

Phase I – Building Network Topology

This lab is an extension of Chapter 23. If you have not completed it yet, it is recommended that you do so first before continuing. You will be building a topology that looks like this:

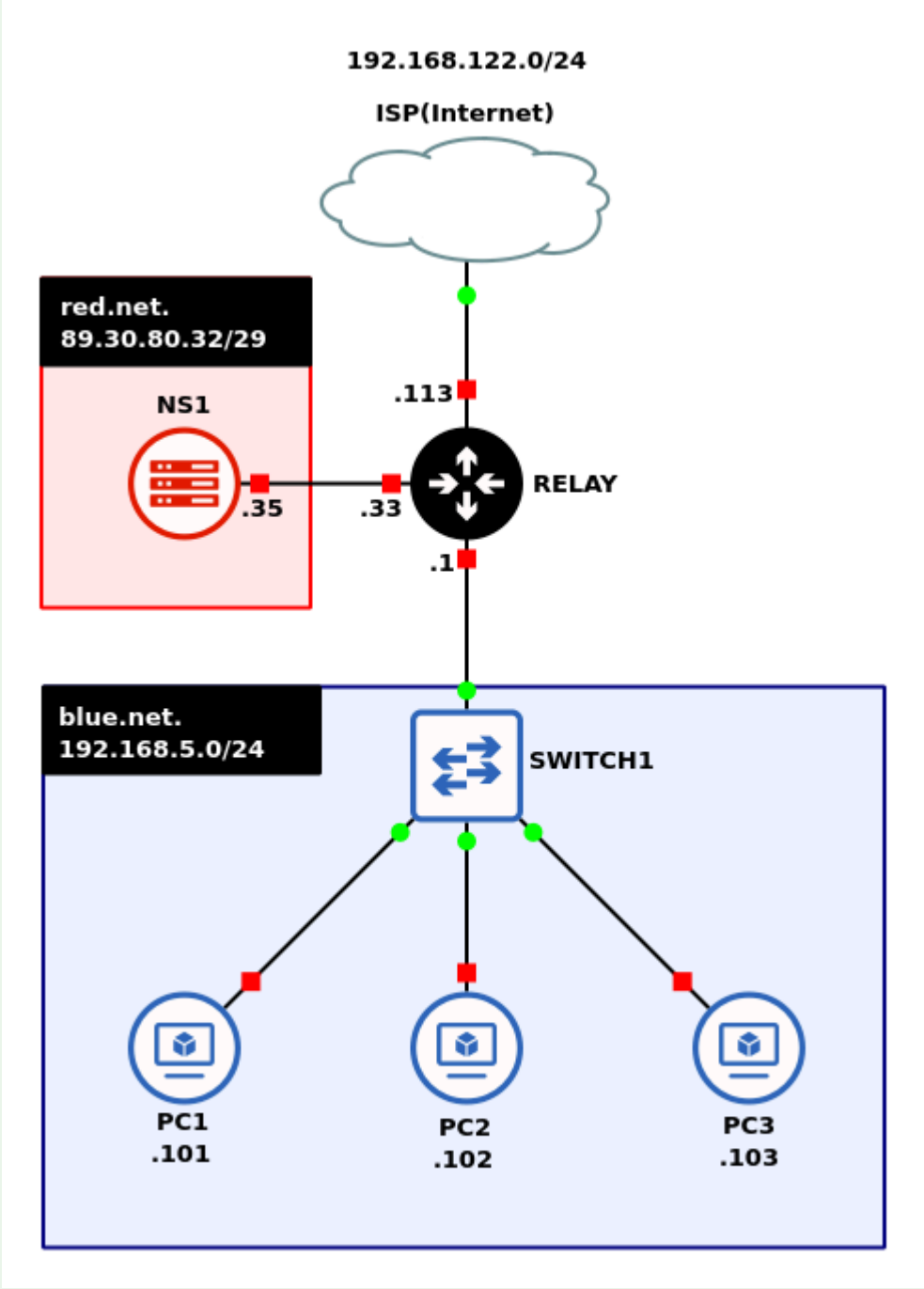


Figure 2 – The infrastructure we are building

1. Start GNS3

    1.1. Open the Chapter 23 Lab

    1.2. Save lab as a new project: **LAB_10**

2. Add a ***NAT*** node to the workspace and connect it to the router

    2.1. Select ***browse all devices***

    2.2. Drag the ***NAT cloud*** to the GNS3 Workspace

    2.3. Connect the NAT Cloud to the router (in this case we are using **ether1** on the router, but you can use any available interface)

    2.4. Start the router and open the console

    2.5. Configure the ether1 interface for DHCP by typing

```
> ip dhcp-client add interface=ether1 disabled=no
```

    2.6. Verify that an IP address has successfully been assigned to the router (Figure 1)

```
> ip address print
```

3. Start the remaining devices in GNS3

4. Ensure that the Authoritative DNS server is working properly and troubleshoot network connectivity as necessary

Phase II – Configuring the Forwarding DNS Server

Here we will make some changes to our name server's configuration. It will still have authority over the LAN's zones (RED, BLUE, and GRAY), but now it will forward all other requests to an outside resolver.

1. Navigate to NS1 and log in

2. Open the DNS daemon configuration file using any text editor you prefer

```
> vi /etc/bind/named.conf.options
```

    2.1. Modify the file to allow recursion and allow the labnets ACL access cache and recursion services

```
recursion yes;
```

```
  allow-query-cache { labnets; };
```

```
  allow-recursion { labnets; };
```

2.2. Add the ***forwarders*** directive inside ***options*** to allow our network to use Google's server as our DNS resolvers

```
  forwarders {
8.8.8.8;
8.8.4.4;
};
```

2.3. Use the following image for configuration reference (Figure 3)

2.4. Save and exit the editor

3. Check the configuration file for syntax errors

```
> named-checkconf /etc/bind/named.conf.options
```

4. Restart the BIND9 service and check its status to ensure it is active and that all zones were loaded properly

```
> systemctl restart named
```

```
> systemctl status named
```

Phase III – Configuring the MikroTik Router with NAT

If you were to test pinging www.google.com at this point, you would likely receive the following error "Temporary failure in name resolution." This is because the network does not yet know where Google's servers are, let alone how to reach them. Let's configure our router so we can start communicating with the Internet.

1. Open the router console

2. Configure the router with NAT to allow both subnets to access the Internet

2.1. Create a firewall group called ***labnets*** which includes both subnets

```
> ip firewall address-list add list=labnets address=89.30.80.32/27
```

```
> ip firewall address-list add list=labnets address=192.168.5.0/24
```

2.2. Add a NAT rule to the router's forward-facing interface (in our example it is ether1). Since it has a dynamic address, its action will be set to masquerade (masq). Otherwise, this value should be set to "src-nat" with its associated static destination address

```
> ip firewall nat add chain=srcnat action=masq src-address-list=labnets
out-interface=ether1 disabled=no
```

Command	Description
ip firewall nat	Needed to access the control menu for configuring internet protocol firewall network address translation (NAT)
add	add the following commands to the configuration
chain=srcnat	Mikrotik calls the grouping of firewall rules "chains". In this case, we are telling the router to use the srcnat chain – because the source (src) NAT for our packets that originate from outside the network are from our NAT cloud.
action=masq	For all packets, the router will make a masquerade (masq) action and replace the source port of the IP packet with one provided by the router. This is done because our NAT cloud is using non-static IPs.
src-address-list=labnets	the source (src) addresses the firewall should be looking for are the ones in the list named 'labnets'. In step 2.1 above we added two IP addresses to this list: 89.30.80.32/27 and 192.168.5.0/24
out-interface=ether1	the outbound packet interface will be ether1
disabled = no	keep this rule in

3. Test the name resolution service by looking up any public website of your choice from all devices within the LAN

```
> nslookup www.google.com
```

```
> ping www.google.com
```

View Figure 4 for an example of the results

4. Open the NS1 server for a more detailed examination of the DNS hierarchy with the following Domain Information Groper (dig) utility command

```
> dig +trace www.google.com +nodnssec
```

View Figure 5 for an example of the results

End of Lab

Deliverables

To receive credit for completing this lab, submit the following 4 screenshots

- Screenshot of RED1
  - Successful nslookup of Google
  - Successful ping of Google
- Screenshot of BLUE1
  - Successful nslookup of Google
  - Successful ping of Google
- Screenshot of the dig trace of Google
- Screenshot of GNS3 environment

Homeworks

NOTE: – some websites block DIG requests, so if you get no response, try another site

DIG is short for a tool named "Domain Information Groper". It is used to interrogate DNS name servers. We use DIG to perform DNS lookups and evaluate the returned responses. It is flexible to use and can be used as a command line or batch function. Visit the man pages in a Linux machine to view all the options available.

Assignment 1 – Explore the functions of the DIG tool

- Use the dig manual in the NS1 server (man dig) to explore various ways of using the dig tool. This is very helpful when assessing how much information your web pages reveal or performing reconnaissance on a target.

Assignment 2 – Use DIG in batch mode

- Use dig's batch mode to conduct the same 4 inquiries as Assignment 1 on three different websites. You will need to create a file to contain the website names. Submit 4 screenshots as evidence:

RECOMMENDED GRADING CRITERIA FOR BOTH ASSIGNMENTS – 4 screenshots

- Demonstrated using Dig to perform a DNS lookup

- Retrieved all DNS records along with the IP addresses using Dig
- Utilized advanced Dig commands to retrieve only the IP address associated with the target domain name
- Used Dig to look up the target domain by its IP address

Screenshots for Printed Copies

```
[admin@MikroTik] > ip add p
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS          NETWORK       INTERFACE
0 192.168.5.1/24   192.168.5.0   ether2
1 89.30.80.33/29   89.30.80.32   ether3
[admin@MikroTik] > ip dhcp-client add interface=ether1 disabled=no
[admin@MikroTik] > ip add p
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#   ADDRESS           NETWORK        INTERFACE
0   192.168.5.1/24    192.168.5.0    ether2
1   89.30.80.33/29    89.30.80.32    ether3
2 D 192.168.122.8/24  192.168.122.0  ether1
[admin@MikroTik] >
```

Figure 1 – Configuring ether1 for DHCP

```
acl labnets {
  127.0.0.1;              // localhost
  89.30.80.32/29;         // localnet (red.net.)
  192.168.5.0/24;         // blue.net.
};

options {
        directory "/var/cache/bind";

        version "not currently available";
        recursion yes;
        empty-zones-enable no;

        allow-query { labnets; };
        allow-query-cache { labnets; };
        allow-recursion { labnets; };
        allow-transfer { none; };

        dnssec-validation no;

        auth-nxdomain no;
        listen-on port 53 { 127.0.0.1; 89.30.80.35; };
        listen-on-v6 { none; };

        forwarders {
        8.8.8.8;
        8.8.4.4;
        };
};
```

Figure 3 – Modified NS1 settings

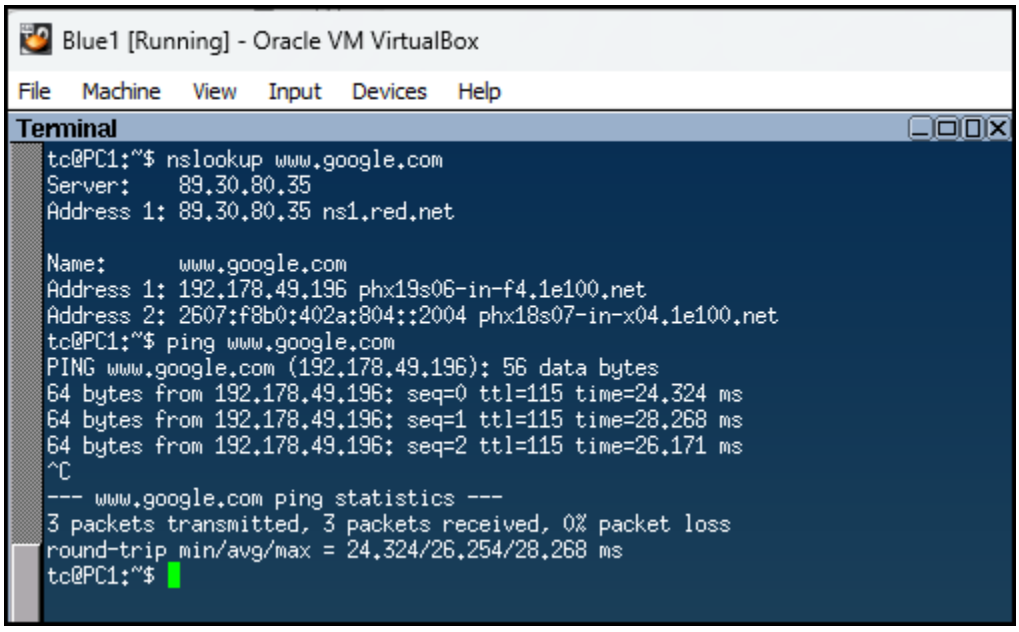


Figure 4 – nslookup for google.com

```
student@ns1:~$ dig +trace www.google.com +nodnssec

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> +trace www.google.com +nodnssec
;; global options: +cmd
.                       517812  IN      NS      e.root-servers.net.
.                       517812  IN      NS      d.root-servers.net.
.                       517812  IN      NS      f.root-servers.net.
.                       517812  IN      NS      b.root-servers.net.
.                       517812  IN      NS      m.root-servers.net.
.                       517812  IN      NS      l.root-servers.net.
.                       517812  IN      NS      i.root-servers.net.
.                       517812  IN      NS      j.root-servers.net.
.                       517812  IN      NS      g.root-servers.net.
.                       517812  IN      NS      c.root-servers.net.
.                       517812  IN      NS      a.root-servers.net.
.                       517812  IN      NS      h.root-servers.net.
.                       517812  IN      NS      k.root-servers.net.
;; Received 811 bytes from 127.0.0.53#53(127.0.0.53) in 0 ms

;; UDP setup with 2001:500:2f::f#53(2001:500:2f::f) for www.google.com failed: network unreachable.
;; UDP setup with 2001:500:2f::f#53(2001:500:2f::f) for www.google.com failed: network unreachable.
;; UDP setup with 2001:500:2f::f#53(2001:500:2f::f) for www.google.com failed: network unreachable.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
;; Received 870 bytes from 192.112.36.4#53(g.root-servers.net) in 64 ms

;; UDP setup with 2001:503:d2d::30#53(2001:503:d2d::30) for www.google.com failed: network unreachable.
;; UDP setup with 2001:503:a83e::2:30#53(2001:503:a83e::2:30) for www.google.com failed: network unreachable.
;; UDP setup with 2001:503:eea3::30#53(2001:503:eea3::30) for www.google.com failed: network unreachable.
google.com.             172800  IN      NS      ns2.google.com.
google.com.             172800  IN      NS      ns1.google.com.
google.com.             172800  IN      NS      ns3.google.com.
google.com.             172800  IN      NS      ns4.google.com.
;; Received 291 bytes from 192.55.83.30#53(m.gtld-servers.net) in 28 ms

www.google.com.         300     IN      A       192.178.48.228
;; Received 59 bytes from 216.239.38.10#53(ns4.google.com) in 44 ms

student@ns1:~$
```

Figure 5 – Results of running DIG on www.google.com

CHAPTER 25

Domain Name System Part 3 - Dynamic DNS

JACOB CHRISTENSEN

In Chapter 23, we manually configured our DNS entries with the IP addresses that were statically assigned to our client devices. However, it is uncommon to use static IP assignments, especially in networks with hundreds or even thousands of clients! If we can get our DHCP and ADNS servers to communicate with each other, then IPs can automatically be assigned and our zone files can automatically be populated. This practice is called Dynamic DNS (DDNS).

This lab will expand on the work in Chapter 23 and Chapter 24 to learn how to configure and manage DDNS in a small network environment.

Estimated time for completion: 45 minutes

LEARNING OBJECTIVES

- Learn how DDNS works on an enterprise network
- Learn how to work with multiple services (DHCP and DNS) on one network

PREREQUISITES

- Chapter 21 – DHCP Relay
- Chapter 24 – Domain Name System Part 2

DELIVERABLES

- Four (4) screenshots
- BLUE1 (PC1) successfully pinging BLUE2 (PC2) by name
- BLUE2 (PC2) successfully pinging BLUE3 (PC3) by name
- BLUE1 (PC1) successfully performing a DNS query and resolution with the NS1 server via Wireshark monitoring
- Live stream of NS1 reports of the DHCP handshake and zone logs when (BLUE3) PC3 is rebooted

RESOURCES

- BIND9 Administrator Reference Manual – https://bind9.readthedocs.io/en/v9.18.16/
- MikroTik RouterOS Documentation – https://help.mikrotik.com/docs/display/ROS/RouterOS
- Archy's Blog – Dynamic DNS with BIND and ISC-DHCP – https://archyslife.blogspot.com/2018/02/dynamic-dns-with-bind-and-isc-dhcp.html
- Configuring Dynamic DNS with BIND9 – https://doncrawley.com/soundtraining.net/files/configuringdynamicdnswithbind9.pdf

CONTRIBUTORS AND TESTERS

Mathew J. Heath Van Horn, PhD, ERAU-Prescott
Kyle Wheaton, Cybersecurity Student, ERAU-Prescott

Phase I – Building the Network Topology

This lab is an extension of the previous two chapters. If you have not completed them yet, it is recommended that you do so first before continuing. By the end your network should look like the following:

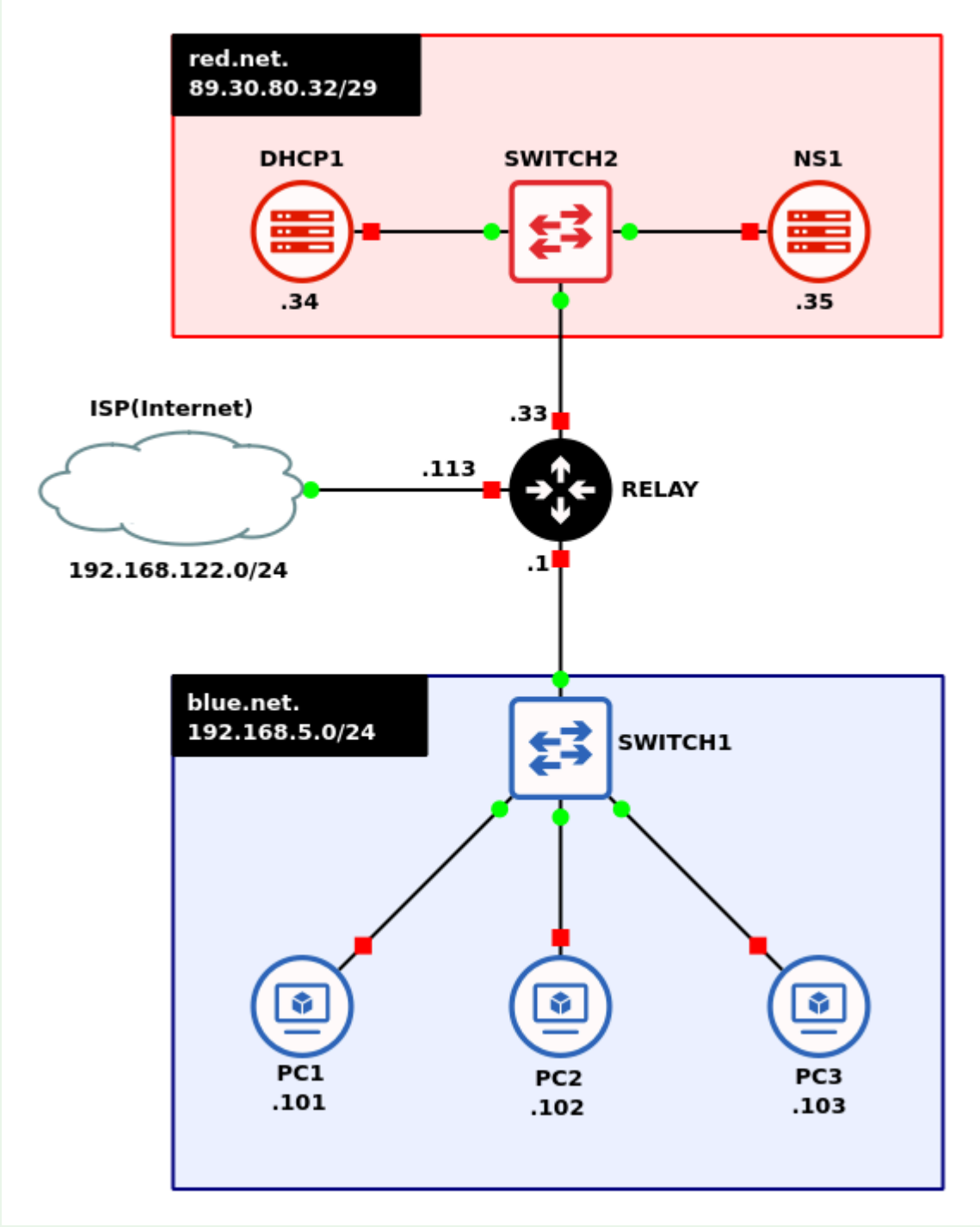


Figure 1 – Final network topology

1. Preliminary step in VirtualBox

    1.1. Create/clone a fresh copy of the Linux Server VM with ISC's DHCP server installed

2. Start GNS3

    2.1. Save the previous lab as a new project: **LAB_11**

3. Reuse the network that you built in the previous chapter

    3.1. Configure the router to act as a **DHCP relay** for the Blue subnet (Chapter 21, Phase II, Step 1)

    3.2. Add a DHCP server to the Red network – *Ubuntu server (DHCP1)*

> **NOTE:** In reality, both the DHCP and DNS services can operate from the same machine; however, we are splitting them into two separate devices for clarity.

    3.3. Add a switch to the Red network – *Ethernet switch*

    3.4. Connect NS1 and DHCP1 to the switch so that they are on the same LAN

    3.5. Remove the static IP addresses from Tiny Core Linux clients by commenting out the rules for static IP assignment

```
> vi /opt/bootlocal.sh
```

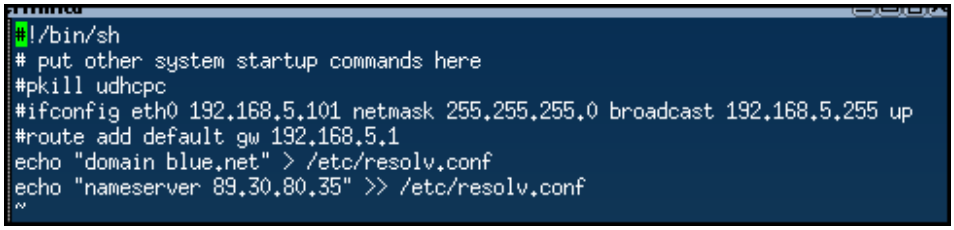


Figure 1a – removing the static IP

> **NOTE:** Make sure that each client device keeps its unique hostnames!

4. Label and organize your network as necessary

Phase II – Modify the DNS Server

This section assumes that all configurations made in the previous chapter have carried over to this one.

1. Start NS1 and login

2. Generate a new Remote Name Daemon Control (RNDC) key to use to communicate with DHCP1

> **NOTE**: Ensure that you have the **bind9-utils** package installed!

```
> rndc-confgen -a -b 512
```

Switch	Description
-a	Automatically generate the RNDC file in the /etc/bind directory.
-b	Set the size of the key as a value between 1 and 512 bits.

3. Modify the file permissions of ***rndc[.]key***

    3.1. Change the owner and group from root to the bind user

    ```
    > chown bind:bind /etc/bind/rndc.key
    ```

    3.2. Change the file permissions to give read access for the group

    ```
    > chmod 640 /etc/bind/rndc.key
    ```

4. Modify the ***named[.]conf[.]local*** file

```
> vi /etc/bind/named.conf.local
```

    4.1. At the beginning of the file, before the zone declarations, include the RNDC key

    ```
    include "/etc/bind/rndc.key";
    ```

```
//
// Do any local configuration here

include "/etc/bind/rndc.key";

// ***** FORWARD ZONES HERE *****
```

Figure 2 – BIND9 configuration

4.2. Within each zone declaration, add the following directive to allow the zones to be updated with the key

```
allow-update { key rndc-key; };
```

```
//
// Do any local configuration here

include "/etc/bind/rndc.key";

// ***** FORWARD ZONES HERE *****

//forward red.net
zone "red.net." {
  type master;
  allow-update { key rndc-key; };
  file "/var/lib/bind/db.red.net";
};

//forward blue.net
zone "blue.net." {
  type master;
  allow-update { key rndc-key; };
  file "/var/lib/bind/db.blue.net";
};
```

Figure 3 – Updating zone directives

4.3. Save and exit the editor

4.4. Verify the configuration settings

```
> named-checkconf
```

5. Add a new static entry in the ***red[.]net*** zone files for DHCP1

5.1. Modify the forward lookup zone file

```
> vi /var/lib/bind/db.red.net
```

```
; BIND9 forward data file for red.net.
;
; Written by Kyle W.
;
$TTL    86400
$ORIGIN red.net.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              5         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.
ns1     IN      A       89.30.80.35
dhcp1   IN      A       89.30.80.34
~
~
~
~
~
~
-- INSERT -- W10: Warning: Changing a readonly file          17,1          All
```

Figure 4 – Updating static forward DNS entry

5.2. Modify the reverse lookup zone file

```
> vi /var/lib/bind/db.red.net.rev
```

```
; BIND9 reverse data file for red.net.
;
; Written by Kyle W.
;
$TTL    86400
$ORIGIN 80.30.89.in-addr.arpa.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              5         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.
35      IN      PTR     ns1.red.net.
34      IN      PTR     dhcp1.red.net.
~
~
~
~
~
~
-- INSERT -- W10: Warning: Changing a readonly file          17,1          All
```

Figure 5 – Updating static reverse DNS entry

6. Remove all client static entries in the ***blue[.]net*** zone files

    6.1. Modify the forward lookup zone file

```
> vi /var/lib/bind/db.blue.net
```

```
; BIND9 forward data file for red.net.
;
; Written by Kyle W.
;
$TTL    86400
$ORIGIN blue.net.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              6         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net.

~
~
~
```

Figure 6 – Purging static forward DNS entries for an internal subnet

6.2. Modify the reverse lookup zone file

```
> vi /var/lib/bind/db.blue.net.rev
```

```
; BIND9 reverse data file for red.net.
;
; Written by Kyle W.
;
$TTL    86400
$ORIGIN 5.168.192.in-addr.arpa.
@       IN      SOA     ns1.red.net. hostmaster.red.net. (
                              6         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      ns1.red.net
.
~
~
~
```

Figure 7 – Purging static reverse DNS entries for an internal subnet

7. Restart the DNS server and verify that it is running

```
> systemctl restart named.service
```

```
> systemctl status named.service
```

NOTE: Errors at this stage are likely due to incorrect permissions on the RNDC file. Ensure that both the owner (bind) and group (bind) can read the file.

8. Start and enable Systemd's time synchronization daemon

```
> systemctl start systemd-timesyncd.service
```

```
> systemctl enable systemd-timesyncd.service
```

Phase III – Configuring the DHCP Server

The following section outlines the configuration of the DHCP daemon for the network.

1. Start DHCP1 and login – Refer to (Chapter 23, Phase III, Step 3)

    1.1. Modify the machine's hostname to **dhcp1**

```
> hostnamectl hostname dhcp1
```

    1.2. Assign a static IP address, default gateway, primary DNS server, and local domain name

NOTE: This example uses the following information:
– IP Address: **89.30.80.34/29**
– Local Domain: **red.net**
– Nameservers: **89.30.80.35**
– Gateway: **89.30.80.33**

        1.2.1. Verify that the name server information is correctly pointing towards **ns1** (89.30.80.35) and that the current domain is **red[.]net**

```
> resolvectl status
```

```
student@dhcp1:~$ resolvectl status
Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: stub

Link 2 (enp0s3)
    Current Scopes: DNS
         Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: 89.30.80.35
       DNS Servers: 89.30.80.35
        DNS Domain: red.net
student@dhcp1:~$
```

Figure 8 – DNS information verification

2. Having all elements in a network synchronized to the same time is critical. SNTP clients provide this service. Linux uses Systemd's timesync daemon to provide SNTP packets. Start it and enable (start on boot) the service by typing

```
> systemctl start systemd-timesyncd.service
```

```
> systemctl enable systemd-timesyncd.service
```

2.1. Transfer the RNDC key file from NS1 to the primary user's home directory in DHCP1

> **NOTE:** The default user on my machines is ***iako***; be sure to change the following commands as necessary.

2.1.1. Login to the NS1 terminal

```
> ssh iako@89.30.80.35
```

> **NOTE:** You should see the hostname change to ***ns1*** when you login successfully.

2.1.2. Switch to the root user

```
> sudo su
```

2.1.3. Use the Secure Copy (SCP) command to make a copy of ***rndc[.]key*** to the primary user of DHCP1

```
> scp /etc/bind/rndc.key iako@89.30.80.34:/home/iako
```

2.1.4. Exit from root

```
> exit
```

2.1.5. Exit the SSH session

```
> exit
```

2.1.6. Verify you see the RNDC key in your home directory

```
> ls -l ~
```

2.2. Modify the permissions for the RNDC key file

2.2.1. Ensure that the owner and group are both ***root***

```
> chown root:root ~/rndc.key
```

2.2.2. Change the permissions to remove read access from others

```
> chmod 640 ~/rndc.key
```

2.2.3. Move the file to the DHCP configuration directory

```
> mv ~/rndc.key /etc/dhcp/ddns-keys
```

NOTE: If the ***ddns-keys*** folder does not exist, make it with the following permissions:

```
> mkdir /etc/dhcp/ddns-keys
```

```
> chown root:dhcpd /etc/dhcp/ddns-keys
```

```
> chmod 710 /etc/dhcp/ddns-keys
```

2.3. Modify the DHCP daemon configuration file to support both subnets and their domains

```
> vi /etc/dhcp/dhcpd.conf
```

2.3.1. Include global parameters that apply to all subnets, including the new RNDC key file

```
# Written by 'Jake M. Christensen'
# 2024.03.31

###################################
######## Global Parameters ########
###################################

authoritative;
default-lease-time 600;
max-lease-time 7200;

###################################
######## DDNS Parameters ########
###################################

ddns-updates on;
ddns-update-style standard;
include "/etc/dhcp/ddns-keys/rndc.key";
```

Figure 9 – DHCP daemon configuration part 1

2.3.2. Add the forward lookup zones for the subnets included in our BIND9 server

```
##############################
####### Forward Zones #######
##############################

# red.net
zone red.net. {
  primary 89.30.80.35;
  key rndc-key;
}

# blue.net
zone blue.net. {
  primary 89.30.80.35;
  key rndc-key;
}
```

Figure 10 – DHCP daemon configuration part 2

2.3.3. Add the reverse lookup zones for the subnets included in our BIND9 server

```
##############################
####### Reverse Zones #######
##############################

# red.net
zone 80.30.89.in-addr.arpa. {
  primary 89.30.80.35;
  key rndc-key;
}

# blue.net
zone 5.168.192.in-addr.arpa. {
  primary 89.30.80.35;
  key rndc-key;
}
```

Figure 11 – DHCP daemon configuration part 3

2.3.4. Add the DHCP subnet directives for both the Red and Blue networks

```
#################################
####### Subnet Directives #######
#################################

# red.net.
subnet 89.30.80.32 netmask 255.255.255.248 {
}

# blue.net.
subnet 192.168.5.0 netmask 255.255.255.0 {
  option routers             192.168.5.1;
  option subnet-mask         255.255.255.0;
  option domain-name         "blue.net";
  option domain-name-servers 89.30.80.35;
  range                      192.168.5.100 192.168.5.200;
}
```

Figure 12 – DHCP daemon configuration part 4

2.4. Restart the DHCP service

2.5. Ensure that each client in ***blue[.]net*** is able to receive an IP address and that its domain information is correct

2.5.1. Login to the terminal of PC1

2.5.2. Verify it was assigned an IP address

```
> ifconfig
```

```
tc@PC1:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:0A:74:7D
          inet addr:192.168.5.100  Bcast:192.168.5.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:52 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2277 (2.2 KiB)  TX bytes:17222 (16.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

tc@PC1:~$
```

Figure 13 – Tiny Core static IP verification

2.5.3. Verify its DNS information is correct

```
> cat /etc/resolv.conf
```

```
tc@PC1:~$ cat /etc/resolv.conf
search blue.net
nameserver 89.30.80.35
tc@PC1:~$
```

Figure 14 – DNS information verification

2.5.4. Repeat for the other two client devices

2.6. **Troubleshoot** as necessary before proceeding to the next section

Phase IV – Wireshark Captures

Now that our network is set up, lets watch at some live packet captures to what DNS queries look like in action.

1. Start a Wireshark capture between NS1 and the switch
2. Test the dynamic DNS updates
    2.1. In Wireshark, filter for ***DNS***

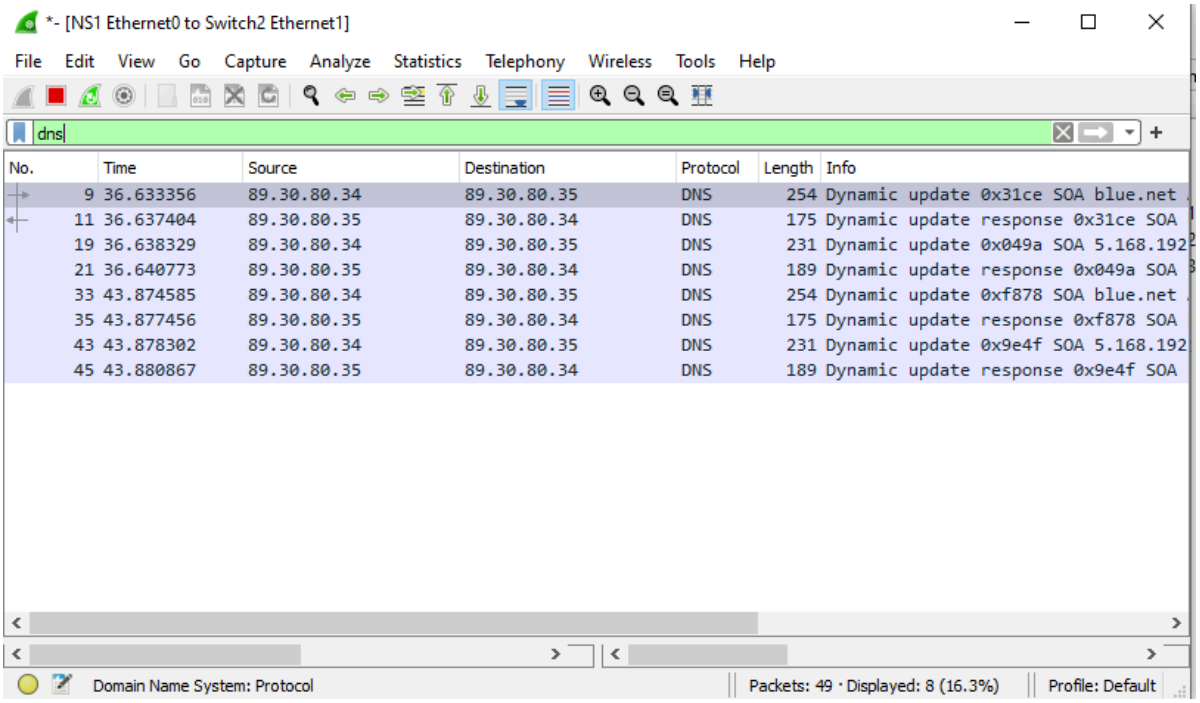


Figure 15 – DNS traffic

2.2. In the NS1 terminal, start monitoring the system log

```
> tail -f /var/log/syslog
```

```
student@ns1:~$ tail -f /var/log/syslog
Apr 13 23:02:20 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#39739/key rn
dc-key: updating zone 'blue.net/IN': adding an RR at 'PC2.blue.net' DHCID AAEBi4
5Tm5N1zOHmP3tkkGlMxw3NOFBiydgAtTVNX447kus=
Apr 13 23:02:20 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#44359/key rn
dc-key: signer "rndc-key" approved
Apr 13 23:02:20 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#44359/key rn
dc-key: updating zone '5.168.192.in-addr.arpa/IN': deleting rrset at '101.5.168.
192.in-addr.arpa' PTR
Apr 13 23:02:20 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#44359/key rn
dc-key: updating zone '5.168.192.in-addr.arpa/IN': adding an RR at '101.5.168.19
2.in-addr.arpa' PTR PC2.blue.net.
Apr 13 23:02:27 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#44027/key rn
dc-key: signer "rndc-key" approved
Apr 13 23:02:27 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#44027/key rn
dc-key: updating zone 'blue.net/IN': adding an RR at 'PC3.blue.net' A 192.168.5.
102
Apr 13 23:02:27 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#44027/key rn
dc-key: updating zone 'blue.net/IN': adding an RR at 'PC3.blue.net' DHCID AAEBqK
vHR2EWYBRNUWpZFdFd6mLRx1VUVUZdtFkjbOcft+0=
Apr 13 23:02:27 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#43343/key rn
dc-key: signer "rndc-key" approved
Apr 13 23:02:27 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#43343/key rn
```

Figure 16 – Live monitoring system logs

2.3. Reboot PC1 and watch the logs for the DHCP handshake and zone file mapping

```
Apr 13 23:04:25 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#45305/key rndc-key: signer "rndc-key" approved
Apr 13 23:04:25 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#45305/key rndc-key: updating zone 'blue.net/IN': adding an R
R at 'PC1.blue.net' A 192.168.5.100
Apr 13 23:04:25 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#45305/key rndc-key: updating zone 'blue.net/IN': adding an R
R at 'PC1.blue.net' DHCID AAEBZ9NZ4YN4XdoDFvPTKePWRvZ2kiWT6KykJVbGkpaXOFw=
Apr 13 23:04:25 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#58277/key rndc-key: signer "rndc-key" approved
Apr 13 23:04:25 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#58277/key rndc-key: updating zone '5.168.192.in-addr.arpa/IN
': deleting rrset at '100.5.168.192.in-addr.arpa' PTR
Apr 13 23:04:25 ns1 named[3275]: client @0x7fe1ac14a4d8 89.30.80.34#58277/key rndc-key: updating zone '5.168.192.in-addr.arpa/IN
': adding an RR at '100.5.168.192.in-addr.arpa' PTR PC1.blue.net.
```

Figure 17 – Dynamic DNS updates

2.4. In the Wireshark window, you should see *DNS Update* packets

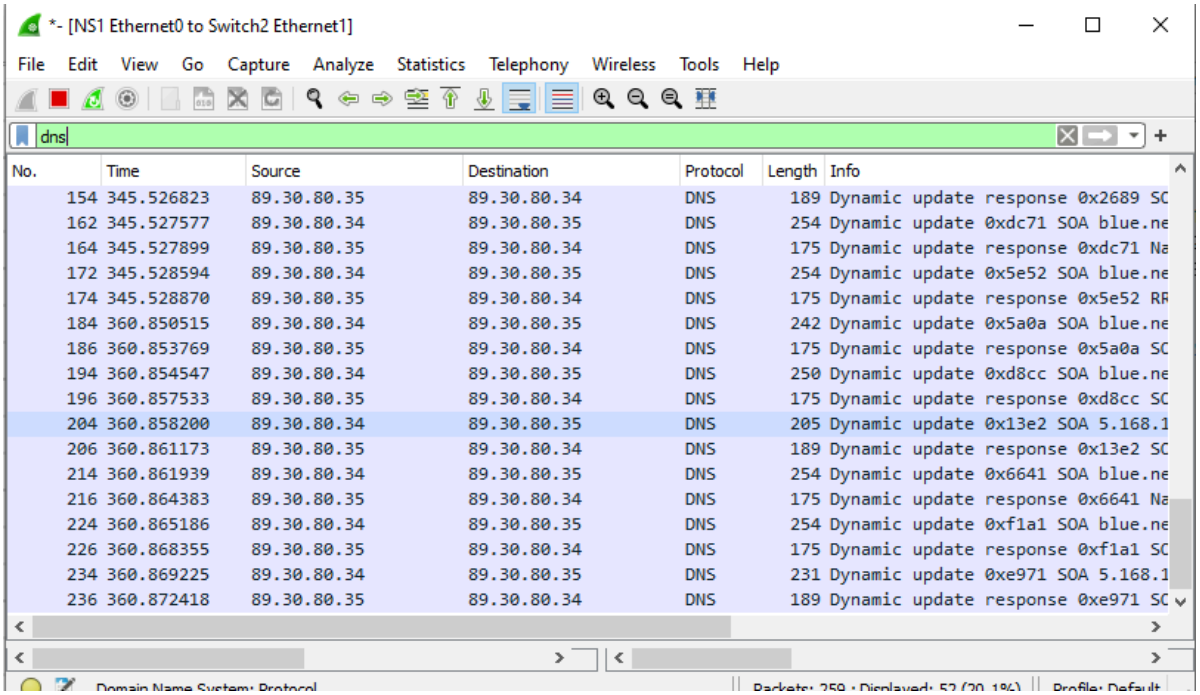


Figure 18 – Wireshark DNS updates

2.5. Repeat for the other two PC clients

> **NOTE:** You can request new IP address leases in Tiny Core with the following command:
>
> ```
> > sudo udhcpc
> ```

3. After about 15 minutes, BIND9 will populate the db files with the updated host/IP records

> **NOTE:** By default, this information is stored in a journal file (.JNL) which is located in the same directory as the zone files. The main database files are not frequently updated to increase efficiency.

End of Lab

Deliverables

4 screenshots are needed to receive credit for this exercise:

- BLUE1 (PC1) successfully pinging BLUE2 (PC2) by name as observed on BLUE1 (PC1)

- BLUE2 (PC2) successfully pinging BLUE3 (PC3) by name as observed on BLUE2 (PC2)
- BLUE1 (PC1) successfully performing a DNS query and resolution with the NS1 server via Wireshark monitoring
- Live stream of NS1 reports of the DHCP handshake and zone logs when (BLUE3) PC3 is rebooted

Homeworks

Assignment 1 – Add the GREEN network to the workspace on the existing router

- Minimize wasted address space for 313 machines
    - Add 3 Tiny Core machines to prove functionality
    - It is okay to turn off the 3 BLUE end devices to save VM resources
- Modify the router, DHCP1, and NS1 to provide the same functionality to the GREEN network that exists on the BLUE network

Assignment 2 – Complete Assignment 1 and then add a PURPLE network

- Minimize wasted address space for 512 machines
    - Add 3 Tiny Core machines to prove functionality
    - It is okay to turn off the other end devices to save VM resources
- Modify the router, DHCP1, and NS1 to provide the same functionality to the PURPLE network that exists on the BLUE and GREEN networks

RECOMMENDED GRADING CRITERIA: Same as deliverables with appropriate substitutions for added devices.

CHAPTER 26

Static Networking Part 2

JACOB CHRISTENSEN

Up to this point, we have been using one router in our working environments that use DHCP. However, you will rarely work on a network with only one router because the whole point of an enterprise network is to connect multiple LANs into a single cohesive network.

In this lab, we will create and connect three LANs via three routers. We introduce you to static routing solutions so you can become familiar with routing procedures. Static routing is impractical mainly because it is very manpower intensive to maintain and prone to human error.

Estimated time for completion: 60 minutes

LEARNING OBJECTIVES

- Successfully create three functional LANs:
    - Gray (DHCP Server)
    - Red (Switch + 2 PCs)
    - Blue (Switch + 2 PCs)
- Configure three routers to use static routing so all devices can communicate

PREREQUISITES

- Chapter 20 – Static Networking Part 1
- Chapter 21 – DHCP Relay

DELIVERABLES

4 screenshots are required to receive credit for this lab

- Screenshot of GNS3 workspace with everything labeled
- Screenshot of the DHCP configuration
- Wireshark Screenshots of a Red host successfully pinging:
    - Blue Host

- Gray Host

RESOURCES

- MikroTik RouterOS Documentation – IP Routing – https://help.mikrotik.com/docs/display/ROS/IP+Routing

CONTRIBUTORS AND TESTERS

- Dante Rocca, Cybersecurity Student, ERAU-Prescott

Phase I -Building the Topology

The following steps are to create the baseline for completing the lab. It makes assumptions about learner knowledge from completing previous labs. To reduce the amount of stress on the PC, we will be using Linux boxes for DHCP.

By the end of this chapter, your network should look like the following:

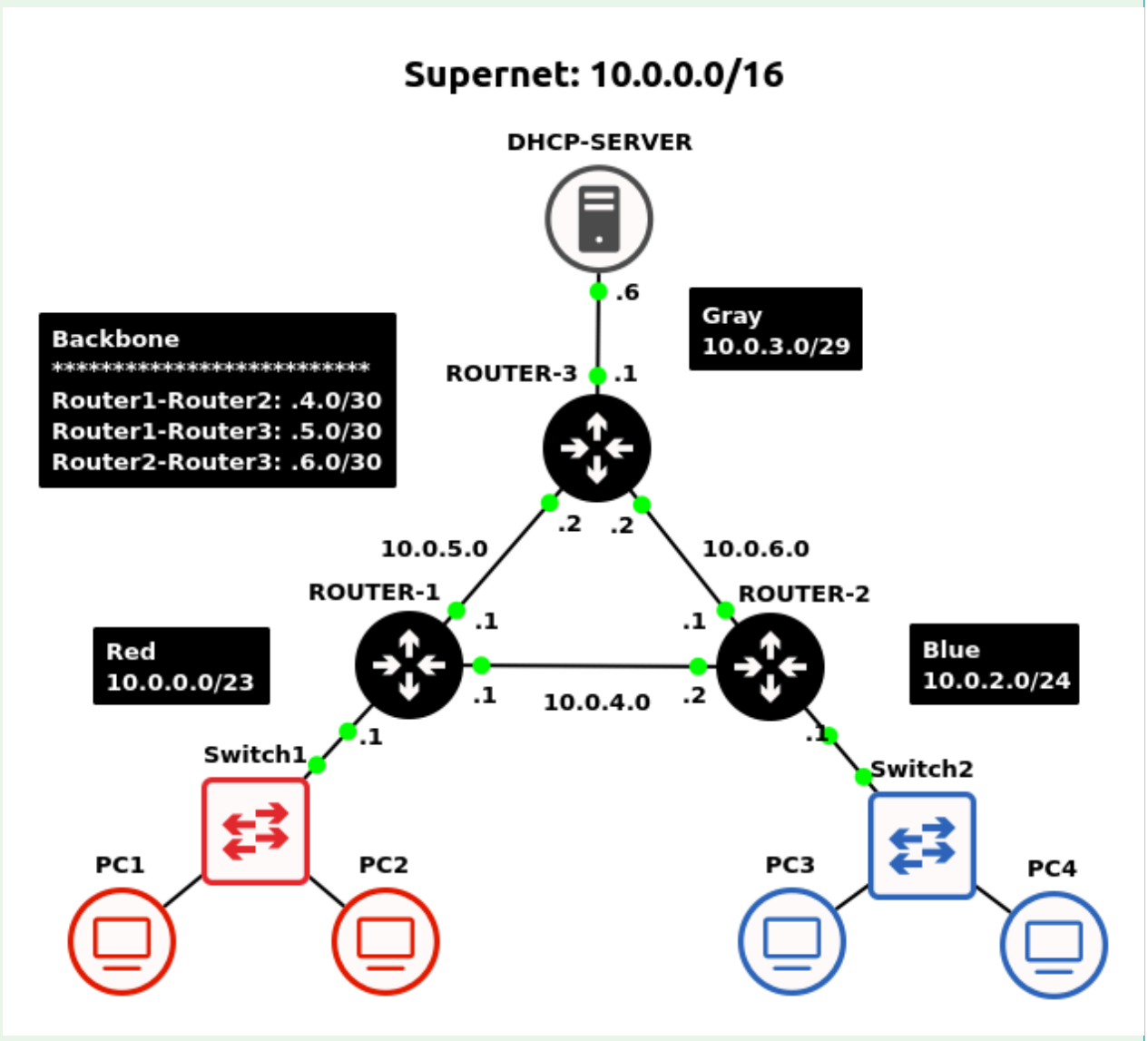


Figure 1 – Final GNS3 network

1. Open GNS3

    1.1. Create a new project: **LAB_12**

2. Build a small network with the following specifications:

    2.1. Class B Supernet – **10.0.0.0/16**

Host Range	
Host Lower Bound	**10.0.0.1**
Host Upper Bound	**10.0.255.254**

NOTE: Our ***supernet*** is the total IP address space we are allowed to use for this network. We will subnet this as necessary to fit our needs for each LAN. If you still confused how subnetting works, there are plenty of online tools that can help augment your learning!

2.2. Subnet – **Red**

2.2.1. One switch – ***Ethernet switch***

2.2.2. Two client machines – ***VPCS***

2.2.3. Minimize wasted address space for ***300hosts***

Network Information	
Network	**10.0.0.0**
Netmask	**255.255.254.0 (/23)**
Broadcast	**10.0.1.255**
Gateway	**10.0.0.1**
DHCP Lower Bound	**10.0.0.100**
DHCP Upper Bound	**10.0.1.250**

NOTE: I am choosing to reserve the first usable host for my gateway addresses. In addition, my DHCP range does not include every single host address available (mostly because I like clean numbers). These are not hard and fast rules. Feel free to adjust as necessary.

2.3. Subnet – **Blue**

2.3.1. One switch – ***Ethernet switch***

2.3.2. Two client machines – ***VPCS***

2.3.3. Minimize wasted address space for ***150 hosts***

Network Information	
Network	**10.0.2.0**
Netmask	**255.255.255.0 (/24)**
Broadcast	**10.0.2.255**
Gateway	**10.0.2.1**
DHCP Lower Bound	**10.0.2.100**
DHCP Upper Bound	**10.0.2.250**

2.4. Subnet – **Gray**

2.4.1. One DHCP server – *Ubuntu 22.04.X LTS*

> **NOTE:** In this example, the server will have a static IP address of **10.0.3.6**.

2.4.2. Minimize wasted address space for *6 hosts*

Network Information	
Network	**10.0.3.0**
Netmask	**255.255.255.248 (/29)**
Broadcast	**10.0.3.7**
Gateway	**10.0.3.1**

2.5. Subnet – **Backbone**

2.5.1. Three routers – *MikroTik CHR*

2.5.2. Full-mesh topology

> **NOTE:** The term ***full-mesh*** simply means that each node is connected to every other node.

2.5.3. Minimize wasted address space for each router-to-router connection

Connection	Network
Router1 <-> Router2	**10.0.4.0/30**
Router1 <-> Router3	**10.0.5.0/30**
Router2 <-> Router3	**10.0.6.0/30**

3. Connect each LAN to their own router

4. Label and organize your network as necessary

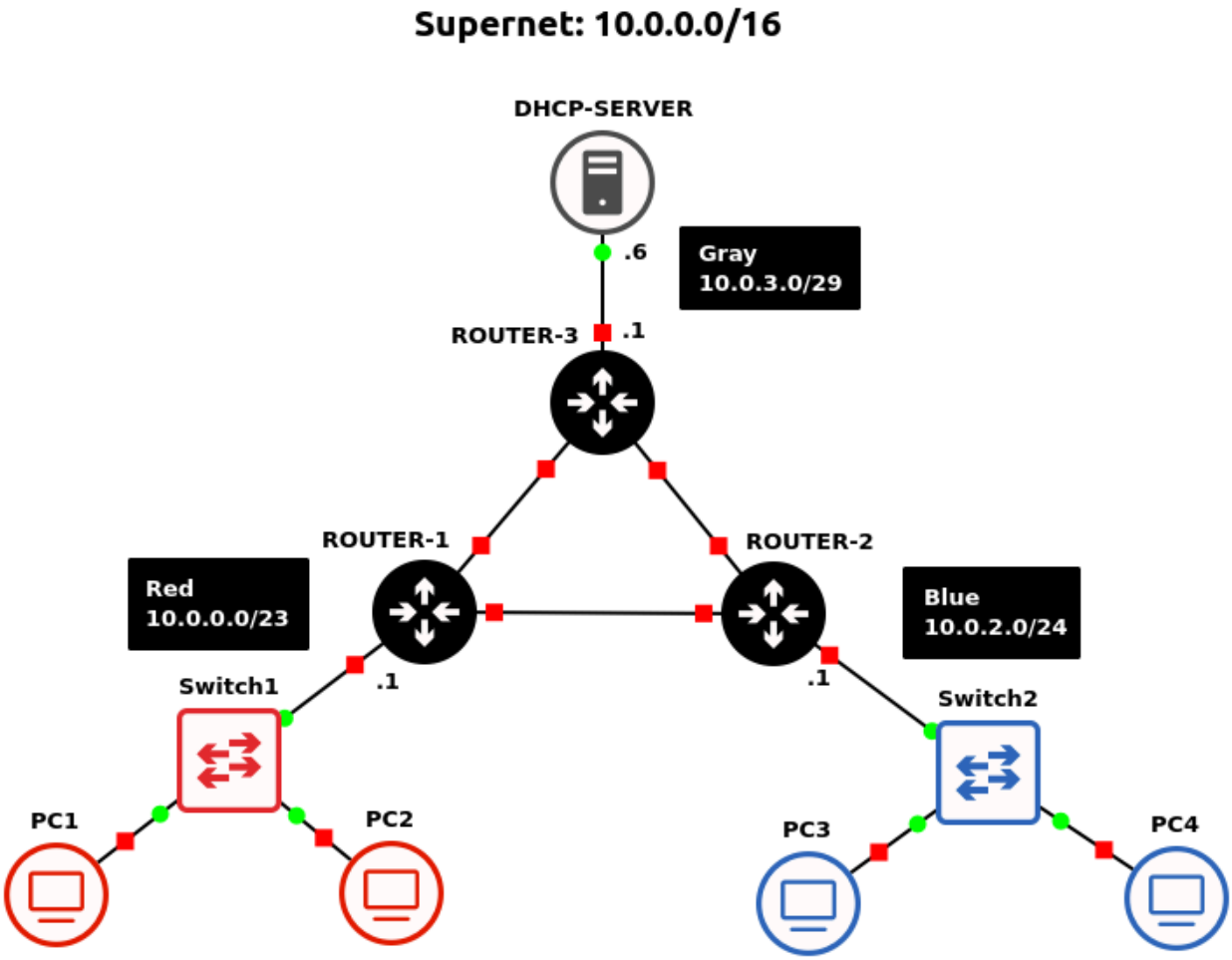


Figure 2 – GNS3 working environment

Phase II – Configuring the Backbone Network

Before any of the clients can receive IP addresses, we need to ensure that the routers can communicate with each other. This phase will focus on how to configure MikroTik routers and establishing static routes.

1. Login to **Router1** and open its console

    1.1. Set static IP addresses for all active network interfaces (Figure 3)

Interface	Network	IPv4 Address
ether1 -> Red	10.0.0.0/23	10.0.0.1
ether2 -> Router2	10.0.4.0/30	10.0.4.1
ether3 -> Router3	10.0.5.0/30	10.0.5.1

NOTE: Refer to Chapter 16, Phase II for additional information on how to configure IP address in MikroTik.

1.2. Configure Router1 to act as a relay for DHCP discover packets (Figure 4)

```
> ip dhcp-relay add name=Red-Relay interface=ether1 dhcp-server=10.0.3.6 local-address=10.0.0.1 disabled=no
```

NOTE: You only need to configure DHCP forwarders for networks directly connected to the relay. In this case, only the Red subnet is attached to this router, so only one relay needs to be made. Refer to Chapter 21, Phase II for additional information.

1.3. Statically update Router1's routing table with routes to the Blue and Gray networks (Figure 5)

NOTE: Two routes need to be created for every subnet, with each specifying the same destination via different gateways (Router2 and Router3). This is a form of redundancy that ensures network functionality even in the case of either path going offline. When building networks, it is important to mitigate as many single point of failures as possible to ensure availability and reliability.

1.3.1. Add all routes to the Blue subnet

```
> ip route add dst-address=10.0.2.0/24 gateway=10.0.4.2 distance=1
```

```
> ip route add dst-address=10.0.2.0/24 gateway=10.0.5.2 distance=2
```

NOTE: The *distance* option specifies how many additional routers are needed to

reach the destination network. The route with the shortest number of hops will take priority over the other.

1.3.2. Add all routes to the Gray subnet

```
  > ip route add dst-address=10.0.3.0/29 gateway=10.0.5.2
distance=1
```

```
  > ip route add dst-address=10.0.3.0/29 gateway=10.0.4.2
distance=2
```

2. Login to **Router2** and open its console

2.1. Set static IP addresses for all active network interface (Figure 6)

Interfaces	Network	IPv4 Address
ether1 -> Blue	10.0.2.0/24	10.0.2.1
ether2 -> Router1	10.0.4.0/30	10.0.4.2
ehter3 -> Router3	10.0.6.0/30	10.0.6.1

2.2. Configure Router2 to act as a relay for DHCP discover packets (Figure 7)

```
  > ip dhcp-relay add name=Blue-Relay interface=ether1 dhcp-
server=10.0.3.6 local-address=10.0.2.1 disabled=no
```

2.3. Statically update Router2's routing table with routes to the Red and Gray networks (Figure 8)

2.3.1. Add all routes to the Red subnet

```
  > ip route add dst-address=10.0.0.0/23 gateway=10.0.4.1
distance=1
```

```
  > ip route add dst-address=10.0.0.0/23 gateway=10.0.6.2
distance=2
```

2.3.2. Add all routes to the Gray subnet

```
> ip route add dst-address=10.0.3.0/29 gateway=10.0.6.2
distance=1
```

```
> ip route add dst-address=10.0.3.0/29 gateway=10.0.4.1
distance=2
```

3. Login to **Router3** and open its console

3.1. Set static IP addresses for all active network interfaces (Figure 9)

Interfaces	Network	IPv4 Address
ether1 -> Gray	10.0.3.0/29	10.0.3.1
ether2 -> Router1	10.0.5.0/30	10.0.5.2
ether3 -> Router2	10.0.6.0/30	10.0.6.2

> **NOTE:** We will not configure any DHCP relays on this device since there is no DHCP-dependent LAN that is directly connected to it. The Gray subnet will only consist of statically assigned host addresses.

3.2. Statically update Router3's routing table with routes to the Red and Blue networks (Figure 10)

3.2.1. Add all routes to the Red subnet

```
> ip route add dst-address=10.0.0.0/23 gateway=10.0.5.1
distance=1
```

```
> ip route add dst-address=10.0.0.0/23 gateway=10.0.6.1
distance=2
```

3.2.2. Add all routes to the Blue subnet

```
> ip route add dst-address=10.0.2.0/24 gateway=10.0.6.1
distance=1
```

```
  >  ip  route  add  dst-address=10.0.2.0/24  gateway=10.0.5.1
distance=2
```

4. Verify that all three routers can ping each other before continuing to the next section

Phase III – Configure the DHCP Server

Now that the network is setup, we can configure our server and test the reliability of the routes.

1. Start the DHCP server and login

    1.1. Configure the network interface with the static IPv4 address **10.0.3.6** (Figure 11)

    1.2. Modify the DHCP daemon configuration file to support the Red and Blue networks (Figure 12)

2. Start PC1 and open its console

    2.1. Test the DHCP service by requesting a new IP address

```
> ip dhcp
```

    2.2. Test the reliability of the network by cutting the Router1-Router3 link

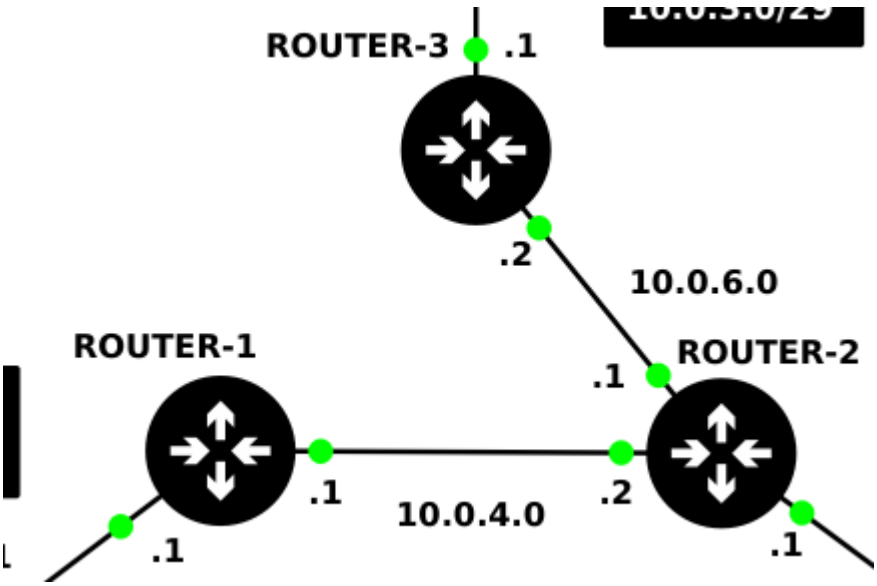


Figure 13 – Cut wire in network

2.3. Request a new IP address

```
> ip dhcp
```

3. Repeat step 2 with a client device from the Blue network

> **NOTE:** Try cutting the Router2-Router3 link instead. We are trying to see if the routers can successfully redirect packets via the longest path!

Congratulations! You were able to create small network with multiple routers by manually administering the routing tables. Hopefully by the end of this exercise you realize how tedious and error-prone this can be as network sizes increases. Luckily, the next few chapters will introduce new protocols that can automate this process for a much friendlier experience.

End of Lab

Deliverables

4 screenshots are required to receive credit for this lab

- Screenshot of GNS3 workspace with everything labeled
- Screenshot of the DHCP configuration

- Wireshark Screenshots of a Red host successfully pinging:
    - Blue Host
    - Gray Host

Homeworks

Assignment 1 – Add another LAN and router to our enterprise

- Add a Green network to the enterprise
- It is projected to use 73 hosts
- The new router needs to connect to both Router1 and Router2 for redundancy
- The Green network needs to get DHCP addresses from the DHCP server
- Hint: don't forget to update the old routers with new paths as well!
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled
    - Screenshot of the DHCP configuration
    - Wireshark Packet Captures where a Green host can ping
        - Red Host
        - Blue Host
        - Gray Host
- Sample network environment:

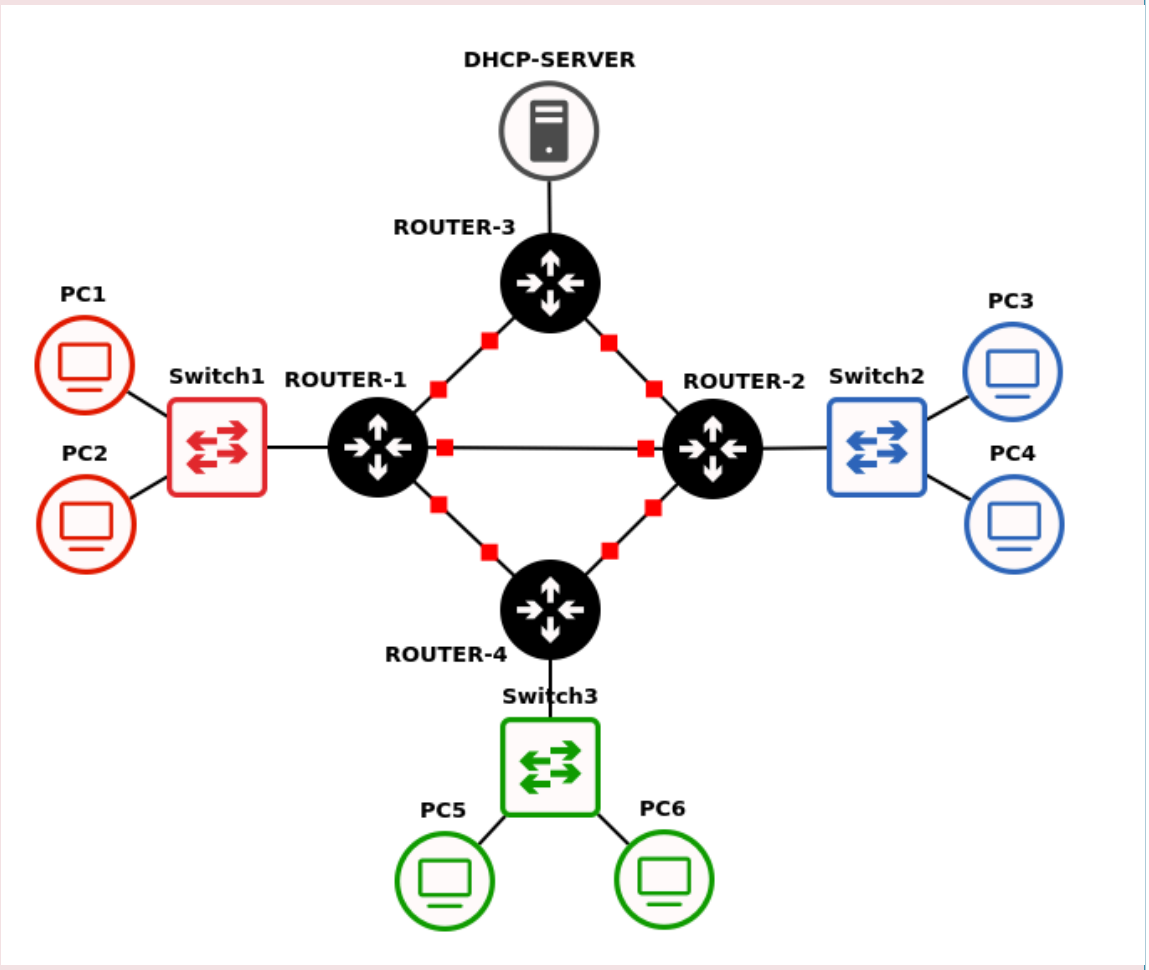


Figure 14 – Assignment 1 network

Assignment 2 – Create a full mesh network

- Building off of Assignment 1
- Add a Purple network to the enterprise
- It is projected to use 600 hosts
- Add network paths so each router has a link to every other router. (e.g. as it stands, Router3 has no direct connection to Router4)
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled
    - Wireshark capture on the following links showing that an ICMP packet from a Blue host takes different paths to reach the Purple host (You may have to disconnect some connections to force the change in path)
        - Router1 <-> Router5
        - Router2 <-> Router5

- Router3 <-> Router5
- Router4 <-> Router5

- Sample network environment:

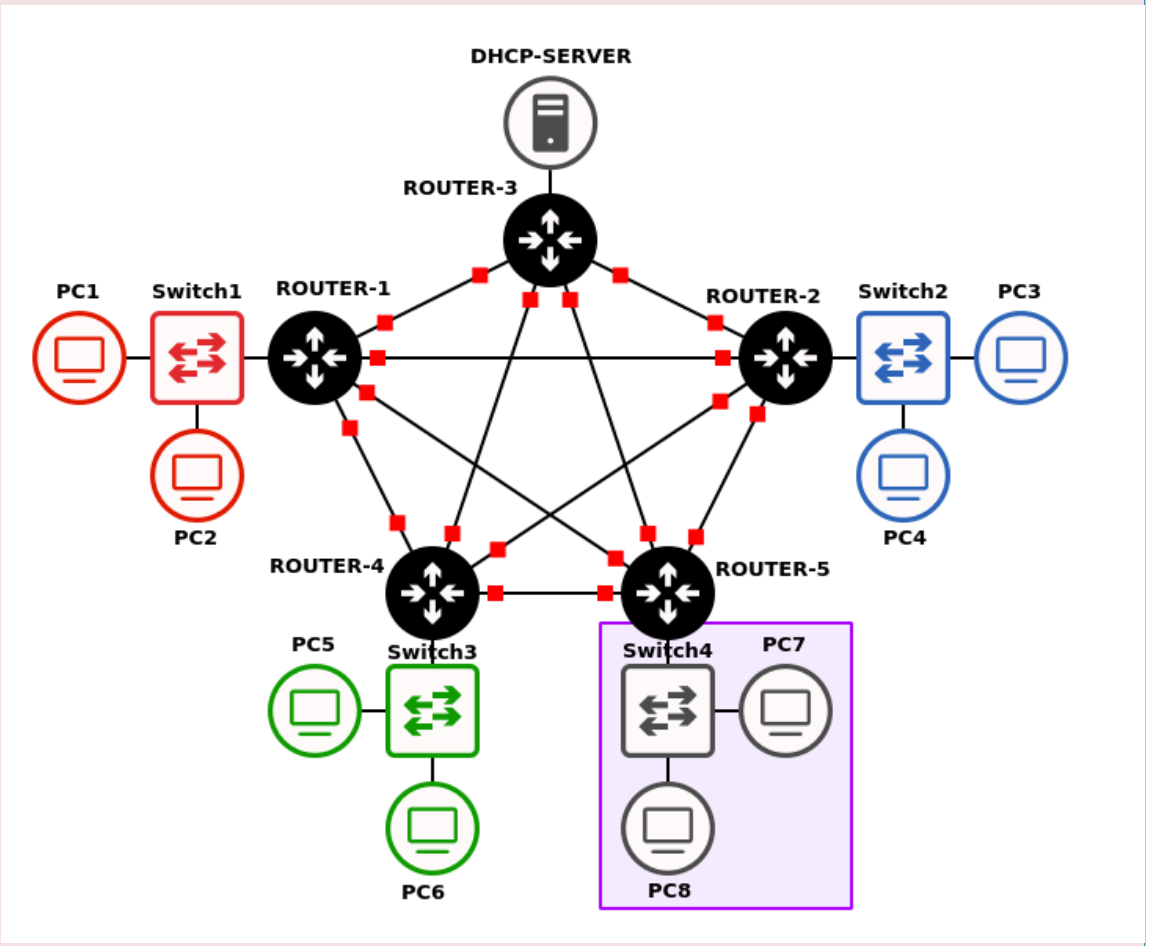


Figure 15 – Assignment 2 network

Figures for Printed Version

```
[admin@Router1] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS      NETWORK   INTERFACE
0 10.0.0.1/23  10.0.0.0  ether1
1 10.0.4.1/30  10.0.4.0  ether2
2 10.0.5.1/30  10.0.5.0  ether3
[admin@Router1] >
```

Figure 3 – Set static addresses for each interface

```
[admin@Router1] > ip dhcp-relay print
Columns: NAME, INTERFACE, DHCP-SERVER, LOCAL-ADDRESS
# NAME       INTERFACE  DHCP-SERVER  LOCAL-ADDRESS
0 Red-Relay  ether1     10.0.3.6     10.0.0.1
[admin@Router1] >
```

Figure 4 – Set the router for DHCP relay traffic

```
[admin@Router1] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, s - STATIC
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#     DST-ADDRESS  GATEWAY   DISTANCE
  DAc 10.0.0.0/23  ether1           0
0   s 10.0.2.0/24  10.0.5.2         2
1  As 10.0.2.0/24  10.0.4.2         1
2   s 10.0.3.0/29  10.0.4.2         2
3  As 10.0.3.0/29  10.0.5.2         1
  DAc 10.0.4.0/30  ether2           0
  DAc 10.0.5.0/30  ether3           0
[admin@Router1] >
```

Figure 5 – Add routes to Blue and Grey networks

```
[admin@Router2] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS      NETWORK   INTERFACE
0 10.0.2.1/24  10.0.2.0  ether1
1 10.0.4.2/30  10.0.4.0  ether2
2 10.0.6.1/30  10.0.6.0  ether3
[admin@Router2] >
```

Figure 6 – Set static IPs for router 2's interfaces

```
[admin@Router2] > ip dhcp-relay print
Columns: NAME, INTERFACE, DHCP-SERVER, LOCAL-ADDRESS
# NAME        INTERFACE  DHCP-SERVER  LOCAL-ADDRESS
0 Blue-Relay  ether1     10.0.3.6     10.0.2.1
[admin@Router2] >
```

Figure 7 – Set router 2 to act as a DHCP relay

```
[admin@Router2] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, s - STATIC
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#     DST-ADDRESS  GATEWAY   DISTANCE
0   s 10.0.0.0/23  10.0.6.2         2
1  As 10.0.0.0/23  10.0.4.1         1
  DAc 10.0.2.0/24  ether1           0
2   s 10.0.3.0/29  10.0.4.1         2
3  As 10.0.3.0/29  10.0.6.2         1
  DAc 10.0.4.0/30  ether2           0
  DAc 10.0.6.0/30  ether3           0
[admin@Router2] >
```

Figure 8 – Add routes to Red and Grey networks on router 2

```
[admin@Router3] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
# ADDRESS      NETWORK   INTERFACE
0 10.0.3.1/29  10.0.3.0  ether1
1 10.0.5.2/30  10.0.5.0  ether2
2 10.0.6.2/30  10.0.6.0  ether3
[admin@Router3] >
```

Figure 9 – Set static IPs on router 3

```
[admin@Router3] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, s - STATIC
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#     DST-ADDRESS  GATEWAY   DISTANCE
0   s 10.0.0.0/23  10.0.6.1         2
1  As 10.0.0.0/23  10.0.5.1         1
2   s 10.0.2.0/24  10.0.5.1         2
3  As 10.0.2.0/24  10.0.6.1         1
  DAc 10.0.3.0/29  ether1           0
  DAc 10.0.5.0/30  ether2           0
  DAc 10.0.6.0/30  ether3           0
[admin@Router3] >
```

Figure 10 – Add routes to Red and Blue network on router 3

```
# This is the network config written by 'Jake M. Christensen'
# 2024.05.12
network:
  ethernets:
    enp0s3:
      optional: true
      dhcp4: false
      addresses:
        - 10.0.3.6/29
      routes:
        - to: default
          via: 10.0.3.1
  version: 2
```

Figure 11 – Static IP on DHCP relay server

```
# Configuration written by 'Jake M. Christensen'
# 2024.05.12

# --------------------
# DHCPD CONFIGURATION
# --------------------

# Global Parameters
# -----------------
authoritative;
default-lease-time 600;
max-lease-time 7200;

# Gray Subnet Directive
# ---------------------
subnet 10.0.3.0 netmask 255.255.255.248 {
}

# Red Subnet Directive
# --------------------
subnet 10.0.0.0 netmask 255.255.254.0 {
  option routers            10.0.0.1;
  option subnet-mask        255.255.254.0;
  option broadcast-address 10.0.1.255;
  range                     10.0.0.100 10.0.1.250;
}

# Blue Subnet Directive
# ---------------------
subnet 10.0.2.0 netmask 255.255.255.0 {
  option routers            10.0.2.1;
  option subnet-mask        255.255.255.0;
  option broadcast-address 10.0.2.255;
  range                     10.0.2.100 10.0.2.250;
}
```

Figure 12 – Add DHCP support for Red and Blue networks

CHAPTER 27

Dynamic Networking - Routing Information Protocol

JACOB CHRISTENSEN AND MATHEW J. HEATH VAN HORN, PHD

As demonstrated in the previous lab, routers need to be told about distant networks in order to communicate with devices 1+ hops away. Doing this task manually is tedious and highly prone to human error, especially as networks start increasing in size. As a result, the Routing Information Protocol (RIP) was developed to allow routing devices to advertise their routing tables with their surrounding neighbors autonomously. Not only did this save configuration time, but it allowed routers to essentially re-calibrate themselves even as devices were added or removed over time.

Estimated time for completion: 15 minutes

LEARNING OBJECTIVES

- Implement the RIPv2 network routing protocol
- Practice using DHCP from a remote server
- Determine a network topology from a captured network packet

PREREQUISITES

- Chapter 26 – Static Networking Part 2

DELIVERABLES

3 Screenshots are required to consider this lab complete:

- Screenshot of GNS3 workspace (LANS labeled with correct IPs and Subnets)
- Screenshot of DHCP configuration settings
- Screenshot of Wireshark packet showing RIPv2 network advertisement for all networks

RESOURCES

- MikroTik RouterOS Documentation – RIP – https://help.mikrotik.com/docs/display/ROS/RIP
- MikroTik Router OS Documentation – RouterOSv7 changes to RIP – https://help.mikrotik.com/docs/

display/ROS/Moving+from+ROSv6+to+v7+with+examples

CONTRIBUTORS AND TESTERS

- Dante Rocca, Cybersecurity Student, ERAU-Prescott

Phase I – Building the Network Topology

The following steps are to create a baseline for completing this lab. It makes assumptions about learner knowledge from completing previous labs.

By the end of this lab your network will look like the following:

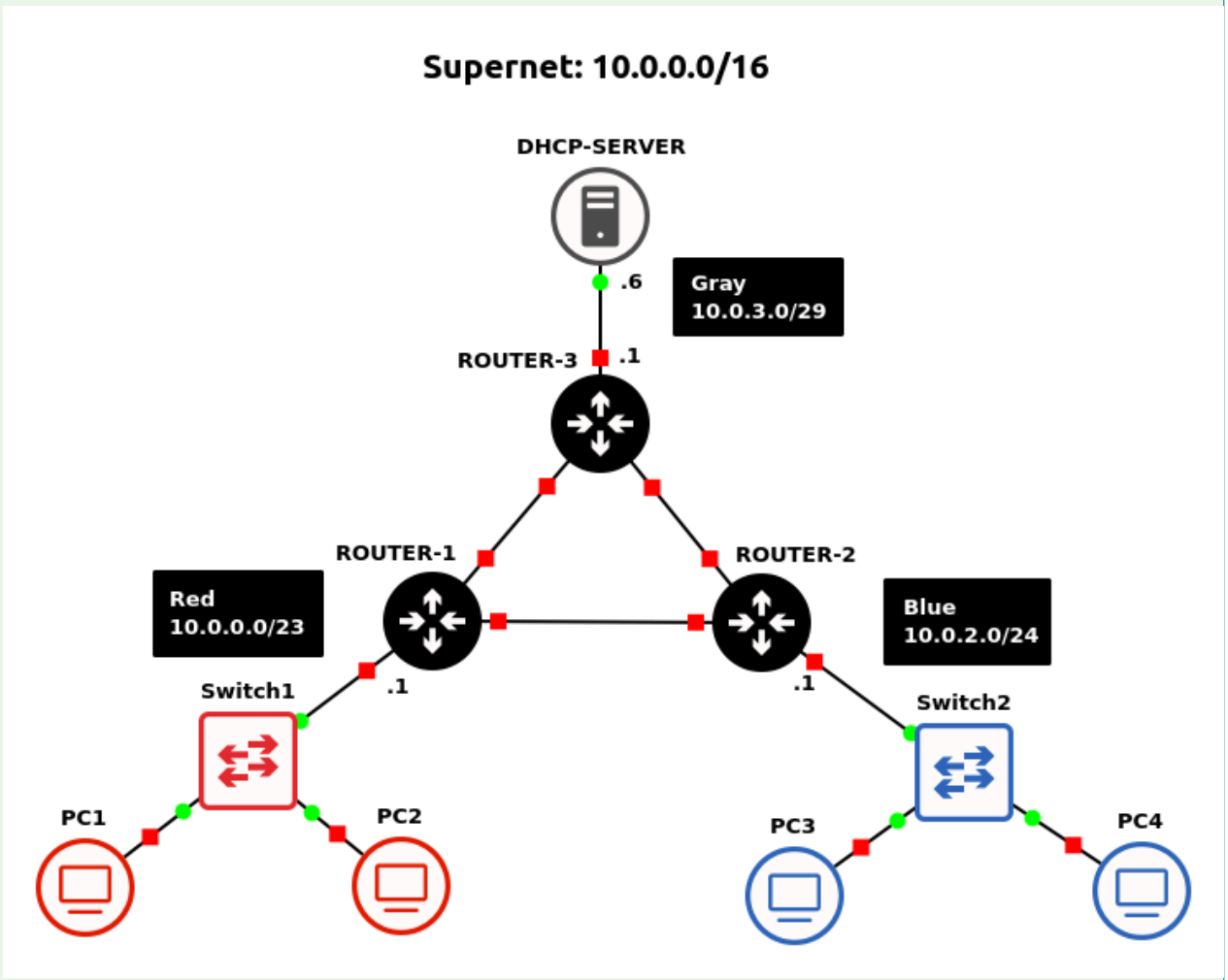


Figure 1 – Final GNS3 environment

1. Open GNS3

    1.1. Open the previous Chapter 26 lab

1.2. Save it as a new project: **LAB_13**

2. Modify the network environment:

2.1. Remove the manually assigned static routes from Router1

```
> ip route remove 0,1,2,3
```

```
[admin@ROUTER-01] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, s - STATIC
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#     DST-ADDRESS  GATEWAY   DISTANCE
  DAc 10.0.0.0/23  ether1           0
0   s 10.0.2.0/24  10.0.5.2         2
1  As 10.0.2.0/24  10.0.4.2         1
2   s 10.0.3.0/29  10.0.4.2         2
3  As 10.0.3.0/29  10.0.5.2         1
  DAc 10.0.4.0/30  ether2           0
  DAc 10.0.5.0/30  ether3           0
[admin@ROUTER-01] > ip route remove 0,1,2,3
[admin@ROUTER-01] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT
Columns: DST-ADDRESS, GATEWAY, DISTANCE
    DST-ADDRESS  GATEWAY  DISTANCE
DAc 10.0.0.0/23  ether1          0
DAc 10.0.4.0/30  ether2          0
DAc 10.0.5.0/30  ether3          0
[admin@ROUTER-01] >
```

Figure 2 – Removing static routes

2.2. Repeat for the other two routers

3. Label and organize your network as necessary

Phase II – Configuring RIPv2 on MikroTik RouterOS

RIP (Routing Information Protocol) is one of the original protocols used by the Internet. Version 2 is the current protocol standard. RIPv2 is a distance-vector protocol in that the routers must communicate with each other about the routes they know about. The term “hop” is used to describe the distance from A to B. In our example, PC1 would take 3 hops to reach PC3: (start) 10.0.0.0 –> 10.0.4.0 –> 10.0.2.0 (end). RIP advertisement packets contain the distance vector hop information. We are going to configure our RED and BLUE networks to use RIPv2 and look at the vector tables. Fortunately for us, MikroTik has simplified RIPv2 configuration immensely!

1. Initialize a Wireshark capture between Router1 and Router2

2. Create a new RIPv2 instance on Router1

```
> routing rip instance add name=RIP-ROUTER-01 redistribute=connected,rip
```

```
> routing rip interface-template add interfaces=all instance=RIP-ROUTER-01
```

3. Create a new RIPv2 instance on Router2

```
> routing rip instance add name=RIP-ROUTER-02 redistribute=connected,rip
```

```
> routing rip interface-template add interfaces=all instance=RIP-ROUTER-02
```

4. Focus on the Wireshark capture window

4.1. You should start to see RIPv2 Request and Response messages being exchanged to the IP 224.0.0.9 over port 520

Source	Port	Destination	Protocol	Info
10.0.4.1	520	224.0.0.9	RIPv2	Response
10.0.4.1	520	224.0.0.9	RIPv2	Response
10.0.4.1	520	224.0.0.9	RIPv2	Response
10.0.4.2	520	224.0.0.9	RIPv2	Request
10.0.4.1	520	10.0.4.2	RIPv2	Response
10.0.4.1	520	224.0.0.9	RIPv2	Response
10.0.4.2	520	224.0.0.9	RIPv2	Response

Figure 3 – Wireshark packet capture filtered for RIP

4.2. Opening any one of these packets will reveal the routing table being distributed

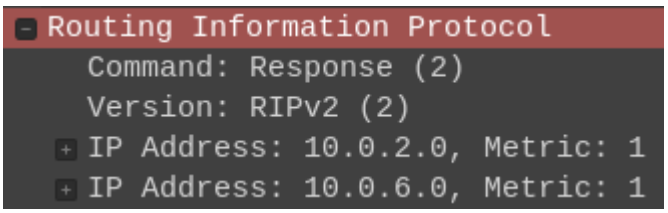


Figure 4 – RIP packet analysis

4.3. The recipient routers will use this information to update their own routing tables

```
> ip route print
```

```
[admin@ROUTER-01] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, r - RIP
Columns: DST-ADDRESS, GATEWAY, DISTANCE
    DST-ADDRESS  GATEWAY          DISTANCE
DAc 10.0.0.0/23  ether1                  0
DAr 10.0.2.0/24  10.0.4.2%ether2       120
DAc 10.0.4.0/30  ether2                  0
DAc 10.0.5.0/30  ether3                  0
DAr 10.0.6.0/30  10.0.4.2%ether2       120
[admin@ROUTER-01] >
```

Figure 5 – Updated routing table

5. Configure RIPv2 on Router3

6. Test the network's new ability to dynamically update its routes

   6.1. Try requesting a new IP address on PC1 and PC3

> **NOTE:** Did you remember to configure Router1 for DHCP-Relay?

   6.2. View the route taken from PC1 to PC3

```
> trace 10.0.2.X -P 1
```

```
PC1> trace 10.0.2.100 -P 1
trace to 10.0.2.100, 8 hops max (ICMP), press Ctrl+C to stop
 1   10.0.0.1   1.265 ms  0.835 ms  0.403 ms
 2   10.0.4.2   2.023 ms  2.213 ms  0.599 ms
 3   10.0.2.100   2.965 ms  0.674 ms  0.735 ms

PC1>
```

Figure 6 – Tracing path to PC3

   6.3. Cut the path that the ICMP packet took to test if RIP can dynamically update network paths

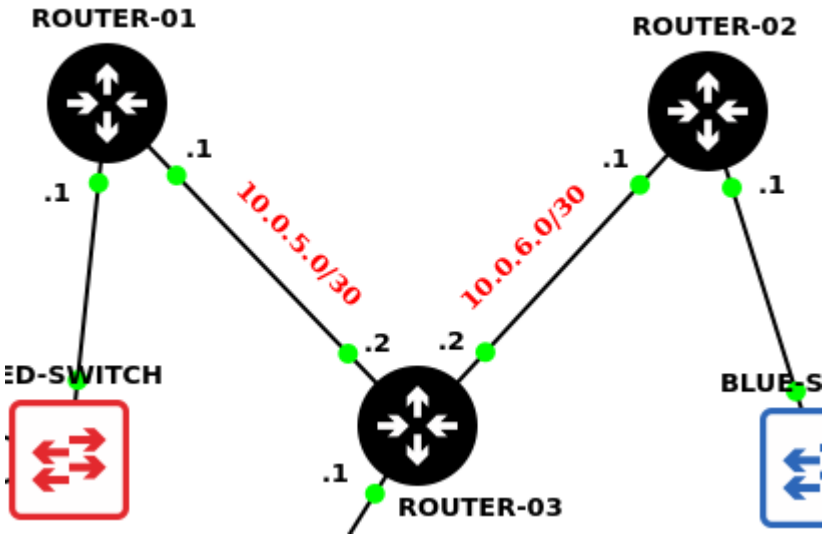


Figure 7 – Cutting path in GNS3

```
> trace 10.0.2.X -P 1
```

```
PC1> trace 10.0.2.100 -P 1
trace to 10.0.2.100, 8 hops max (ICMP), press Ctrl+C to stop
 1   10.0.0.1   1.259 ms  0.602 ms  1.424 ms
 2   10.0.5.2   4.122 ms  2.539 ms  0.931 ms
 3   10.0.6.1   2.373 ms  0.889 ms  0.825 ms
 4   10.0.2.100   1.571 ms  1.469 ms  0.774 ms

PC1>
```

Figure 8 – Tracing route to PC3

Hopefully this exercise proved how significantly easier routing protocols are compared to manually assigning routes in networks.

End of Lab

Deliverables

3 Screenshots are required to consider this lab complete:

- Screenshot of GNS3 workspace (LANS labeled with correct IPs and Subnets)
- Screenshot of DHCP configuration settings
- Screenshot of Wireshark packet showing RIPv2 network advertisement for all networks

Homeworks

Assignment 1 – Update the network build in Assignment 1 from the previous chapter

- Configure DHCP to support the network
- Replace static routes with RIPv2
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled
    - Screenshot of the DHCP configuration
    - Screenshot of RIPv2 packets
    - Wireshark Packet Captures where a Green host can ping
        - Red Host
        - Blue Host
        - Gray Host
- Sample network environment:

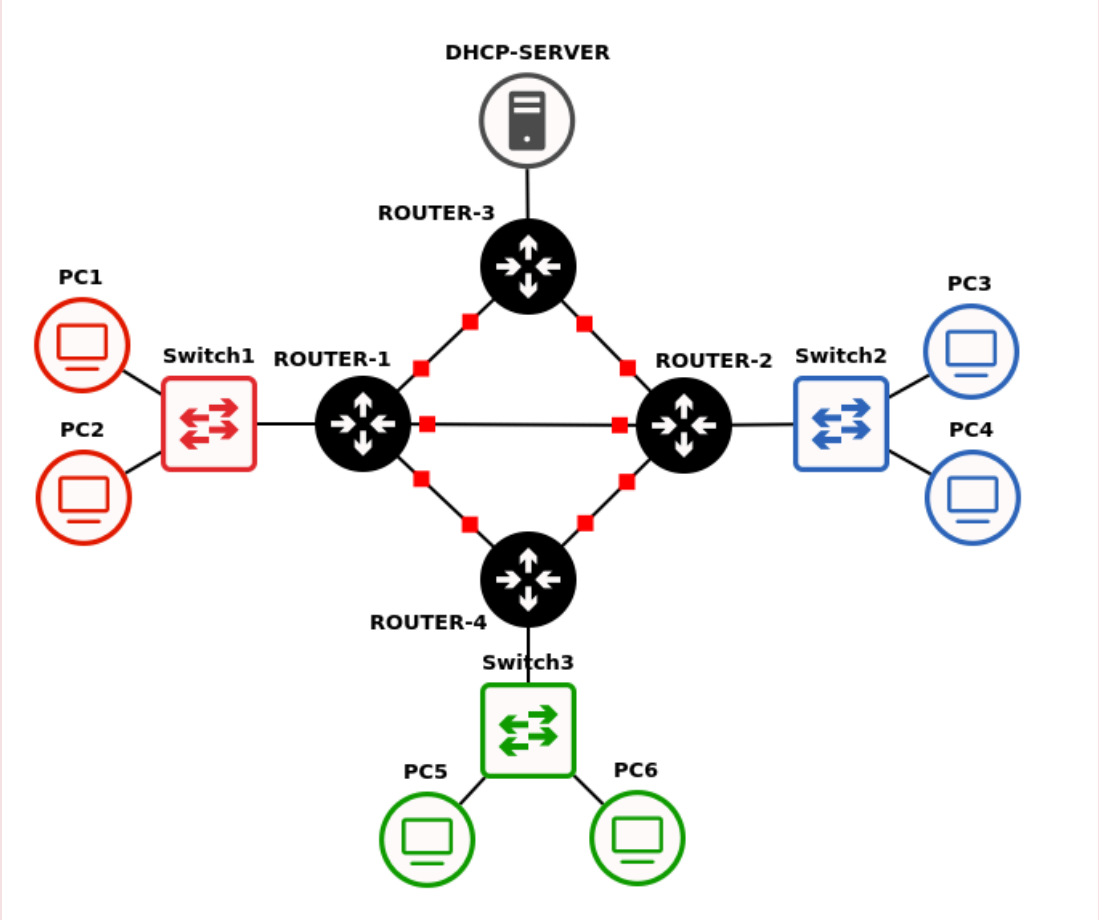


Figure 9 – Assignment 1 network

Assignment 2 – Update the network build in Assignment 2 from the previous chapter

- Configure DHCP to support the network
- Replace static routes with RIPv2
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled
    - Trace route command showing that an ICMP packet from a Blue host takes different paths to reach the Purple host (You may have to disconnect some connections to force the change in path)
        - Router2 –> Router5
        - Router2 –> Router1 -> Router5
        - Router2 –> Router3 -> Router1 -> Router5
        - Router2 –> Router3 -> Router1 -> Router4 -> Router5

- Sample network environment:

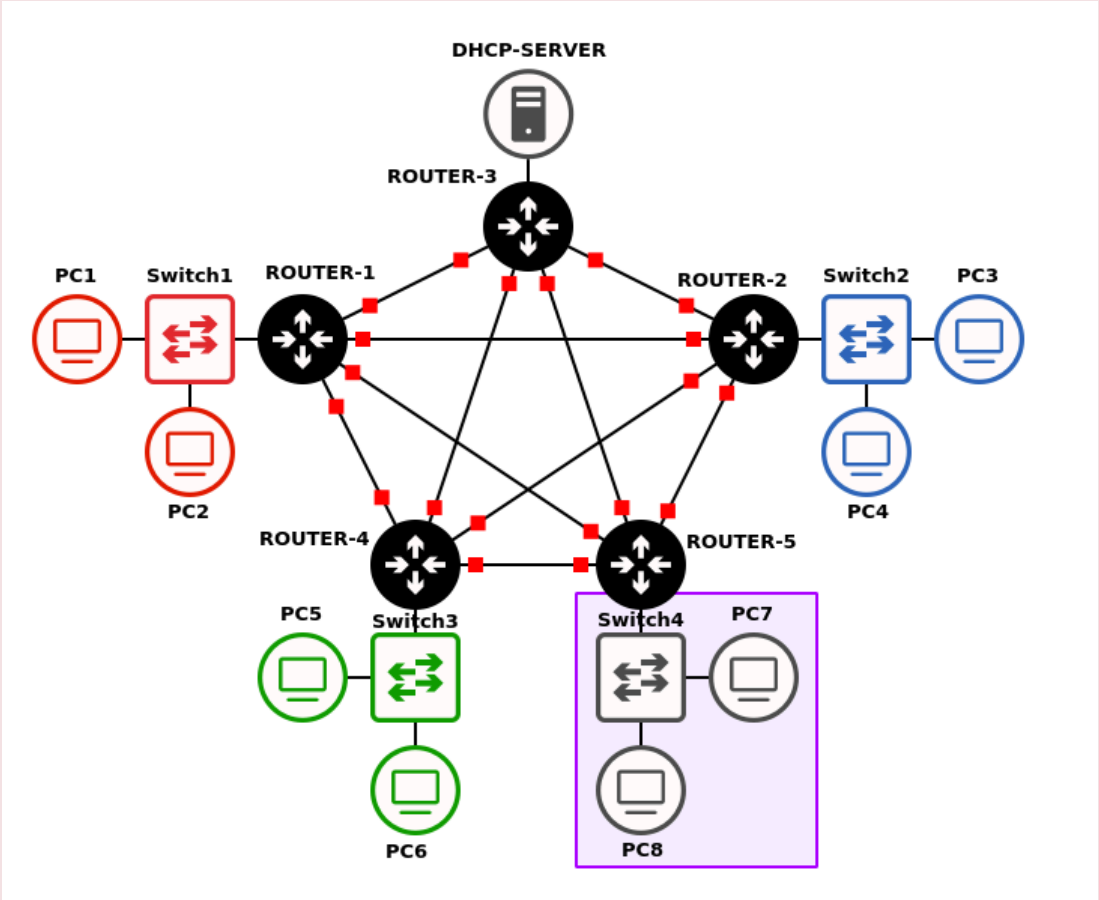


Figure 10 – Assignment 2 network

CHAPTER 28

Dynamic Networking - Open Shortest Path First

MATHEW J. HEATH VAN HORN, PHD AND JACOB CHRISTENSEN

Open Shortest Path First (OSPF) is quite complicated to implement, but it makes things very simple for users. Its essence is that routers share information with each other so that when a data packet needs to go from Point A to Point B, all the routers know the fastest path through the network. The "fastest path" can be decided by the physical distance between routers (speed of light energy loss), electrical distance (router hops), and availability and reliability. This means you can't just look at a network diagram and make assumptions about speed. In this lab, we will build an OSPF network with several paths. We will look at how the OSPF builds a network topology and shares the information amongst the routers. We will also watch how the familiar ICMP packets traverse the network.

Estimated time for completion: 30 minutes

LEARNING OBJECTIVES

- Successfully configure an enterprise network to use OSPF routing
- Use Wireshark to identify packets specifically associate with OSPF
- Implement a DHCP Relay solution to an enterprise network
- Use CIDR subnetting techniques to minimize the IP network space waste

PREREQUISITES

- Chapter 26 – Static Networking Part 2
- Chapter 27 – RIPv2 Networking

DELIVERABLES

- 5 screenshots are required to receive full credit for this assignment
    - GNS3 working environment will have all devices on and labeled correctly
    - A router display of all the IP routes – all four router IDs should be visible
    - Wireshark showing OSPF Hello Packets
    - Wireshark shows ICMP packets between PC1 and PC3 using one path

    - Wireshark shows ICMP packets between PC1 and PC3 using a different path

RESOURCES

- MikroTik RouterOS Documentation – OSPF – https://help.mikrotik.com/docs/display/ROS/OSPF
- IBM – Packet Types for OSPF – https://www.ibm.com/docs/en/i/7.1?topic=concepts-packet-types-ospf

CONTRIBUTORS AND TESTERS

- Dante Rocca, Cybersecurity Student, ERAU-Prescott

Phase I – Terminology

OSPF relies on many terms to describe the relationship between the routers and the routing processes. We will use many of these in this lab, so we will list them here. The list is from the MikroTik RouterOS manual but will be summarized. so you don't have to flip back and forth between websites.

- **Adjacency** – A logical connection between a router and a designated router and a backup designated router. No routing information is exchanged unless adjacencies are formed.
- **Area Border Router (ABR)** – A router that is connected to multiple areas and is responsible for summarizing and update suppression between network areas.
- **Autonomous System (AS)** – Routers that use a common routing protocol to exchange information.
- **Autonomous System Boundary Router (ASBR)** – A term used to describe a router that is connected to an external network and imports the external routes into the OSPF topology.
- **Back-up Designated Router (BDR)** – A hot standby for the designated router and receives all routing updates from adjacent routers, but does not flood with updates.
- **Broadcast** – Network protocols that allow broadcasting (e.g. Ethernet)
- **Cost** – Each link in the network is assigned a cost, a value that is dependent upon the speed of the media. Also known as the *interface output cost* since the time inside a router is not counted.
- **Designated Router (DR)** – A router unique to broadcast network protocols that are used to minimize the number of adjacencies formed.
- **Interface** – The router's physical interface (e.g. ether1). Also known as a **link** in OSPF parlance.
- **Link State** – The status of a link between two routers. It defines the relations between a router's interface and its neighboring routers.
- **Link State Advertisement (LSA)** – A specialized data packet that contains link-state and routing information and is shared between routers.
- **Neighbor** – A connected OSPF router with adjacent routers in the same area.
- **Non-broadcast multi-access (NBMA)** – Routers that allow access, but do not broadcast their information.
- **Point-to-Point** – A network solution that eliminates the need for DRs and BDRs.

- **Router ID** – IP address used to identify the OSPF router. Can be manually or automatically assigned.

Phase II – Setup

The purpose of this lab is to set up and configure OSPF. However, to get to that point, some initial configuration is required. If you saved your configuration from Static Routing or RIPv2, you can reuse it. However, you gain more experience and suffer fewer "I forgot to reset XXXX" problems if you start from scratch.

By the end of this lab, your network will look like the following

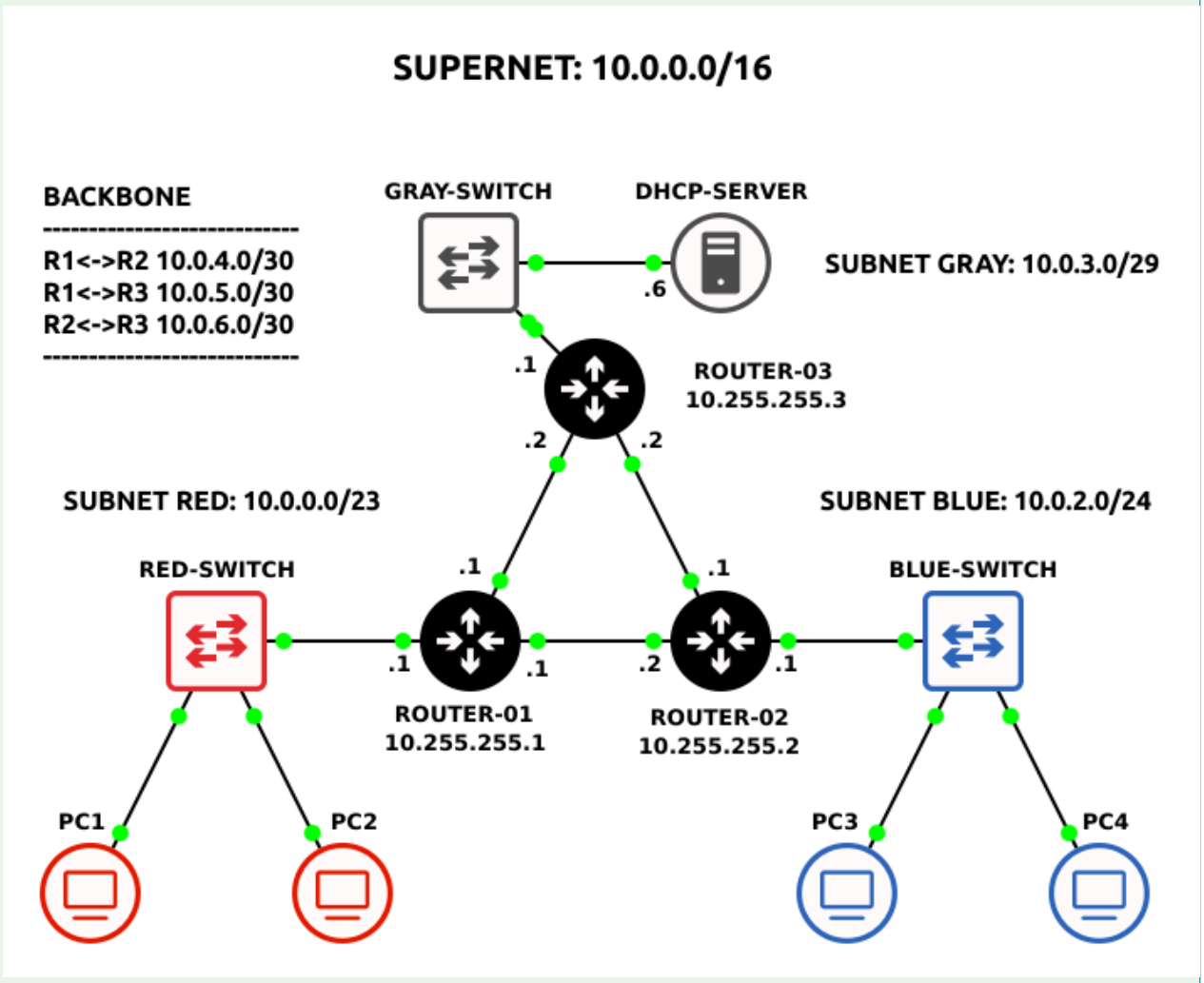


Figure 1 – Final GNS3 network environment

1. Open GNS3

    1.1. Open the previous Chapter 27 lab

    1.2. Save it as a new project: **LAB_14**

2. Remove RIP from the network environment

    2.1. In Router1, remove RIPv2 advertisements from all interfaces

```
> routing rip interface-template remove 0
```

    2.2. Terminate the RIPv2 instance

```
> routing rip instance remove 0
```

3. Assign Router1 a new loopback address to be used as device identifiers for OSPF

    3.1. Create a new loopback interface

```
> interface bridge add name=loopback
```

Command	Meaning
interface	Access the interface menu directly
bridge	every ethernet frame received on this point gets transmitted to all other points
add	create a new interface
name=	what follows will be a name for this new interface
loopback	This self-documenting name allows humans to know the purpose of this bridge interface

NOTE: Loopback interfaces are useful in that they are always online. They cannot go up and down like a physical interface.

    3.2. Assign it a unique IPv4 address – *10.255.255.1*

```
> ip address add address=10.255.255.1/32 interface=loopback
```

Command	Meaning
ip address	access the ip address menu directly
add	create a new IP address
address=	what follows will be the new IP address
10	10 is the start of the backbone IP space
255.255	255 can have several meanings such as “all”, wildcard, no change, etc. In this case, it is a notation saying that this is not a network address for use by the network.
1	this indicates it is router 1
/32	CIDR notation means that only 1 IP address is allowed. It serves as another notation about non-network address
interface=	what follows is the interface that will use this IP address
loopback	the interface name. In this case, it is referring to the bridge interface we created earlier

NOTE: Loopback addresses can be anything (with some exceptions) as long as they are unique to the device. Since we are using them as device identifiers, it is important to be able to quickly differentiate between a routing IP address and a loopback address. For the purpose of documentation and organization, we are using the format 10.255.255.X for the loopback addresses on this network, where X represents the router’s number (ex 1, 2, and 3). This is not a hard-and-fast rule… feel free to adjust as necessary.

4. Repeat steps 2 and 3 above on both Router2 and Router3

5. Update your network diagram with new router ID labels

Phase III – OSPF

OSPF is a link-state routing protocol that finds the shortest path between two network points and then uses this path to send packets. In our current configuration, there is no “shortest path” per se, we are just looking to get OSPF working so you can see it in action.

OSPF configuration in MikroTik Routers follows a basic format:

- Create a loopback interface
- Enable the OSPF routing protocol
- Configure the OSPF area
- Configure the OSPF network

1. Start a Wireshark packet capture on the Router1-Router2 link

2. Configure OSPF on Router1

2.1. Create an instance of OSPF and assign the router's loopback address to serve as the router's ID

```
> routing ospf instance add name=Bob version=2 router-id=10.255.255.1
```

Command	Meaning
routing ospf instance	access the routing ospf instance menu directly
add	create a new OSPF instance
name=	what follows is the name we are giving this instance
Bob	The name, it could be anything that helps humans understand why it is here: marketing, default, building-17, etc. We are using Bob because why not Bob? We are only going to have 1 instance, so it doesn't matter.
version=2	version 2 indicates that we are using OSPF IPv4 networks, version=3 would indicate IPv6
router-id=	what follows is the ID number of the router
10.255.255.1	Not a real IP address meaning router 1 on network space 10.0.0.0

2.2. Create the OSPF area by typing

```
> routing ospf area add name=backbone area-id=0.0.0.0 instance=Bob
```

Command	Meaning
routing ospf area	access the routing ospf area menu directly
add	create a new area
name=	what follows is the name we are giving this area
backbone	this is our main area
area-id=	what follows is the ID number for the area name – NOTE: 0.0.0.0 is always the backbone
instance=Bob	what follows is the instance we are going to use, in this case, we will use the instance 'Bob' that we created earlier

2.3. Add each interface connected to Router1 to the OSPF backbone area

```
> routing ospf interface-template add area=backbone interfaces=all
```

NOTE: There are several additional ways to create the template by....
1. Assigning interface names individually

```
> routing ospf interface-template add area=backbone
interfaces=loopback
```

```
> routing ospf interface-template add area=backbone
interfaces=ether1
```

2. Assigning specific networks instead of interfaces

```
> routing ospf interface-template add area=backbone
network=x.x.x.x/x
```

3. In Wireshark, you should now see ***OSPF Hello*** packets broadcasted at a regular interval of every 10 seconds (further dissected in Phase III)

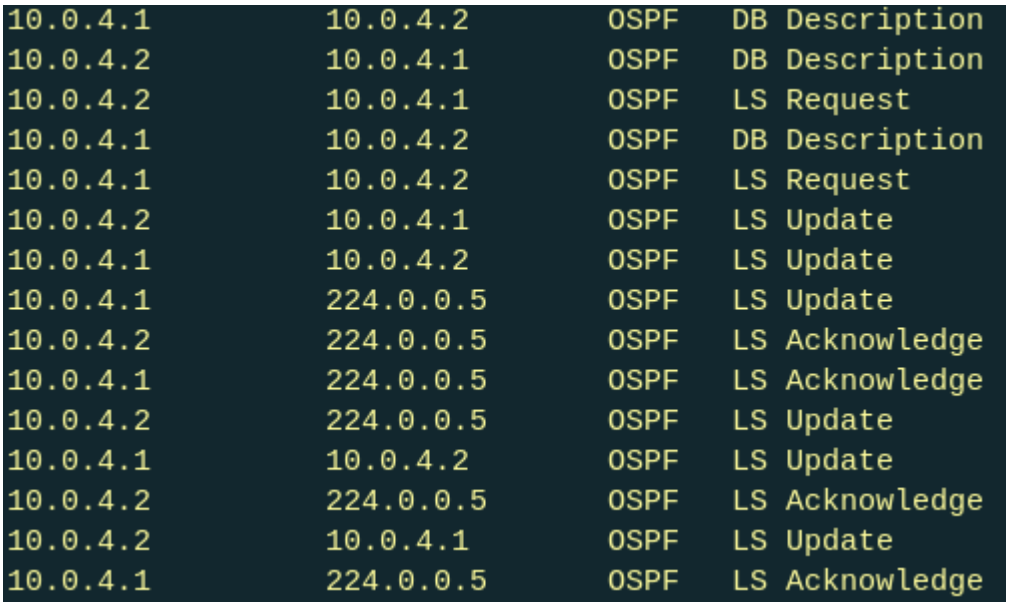
```
10.0.4.1        224.0.0.5       OSPF   Hello Packet
10.0.4.1        224.0.0.5       OSPF   Hello Packet
10.0.4.1        224.0.0.5       OSPF   Hello Packet
10.0.4.1        224.0.0.5       OSPF   Hello Packet
10.0.4.1        224.0.0.5       OSPF   Hello Packet
10.0.4.1        224.0.0.5       OSPF   Hello Packet
10.0.4.1        224.0.0.5       OSPF   Hello Packet
```

Figure 2 – Wireshark packet capture

4. Repeat the above steps to configure OSPF for Router2 and Router3

NOTE: Remember, in this example, Router 2's ID is **10.255.255.2** and Router3's ID is **10.255.255.3**.

5. You will know when you are successful if you see the following OSPF packets in Wireshark



10.0.4.1	10.0.4.2	OSPF	DB Description
10.0.4.2	10.0.4.1	OSPF	DB Description
10.0.4.2	10.0.4.1	OSPF	LS Request
10.0.4.1	10.0.4.2	OSPF	DB Description
10.0.4.1	10.0.4.2	OSPF	LS Request
10.0.4.2	10.0.4.1	OSPF	LS Update
10.0.4.1	10.0.4.2	OSPF	LS Update
10.0.4.1	224.0.0.5	OSPF	LS Update
10.0.4.2	224.0.0.5	OSPF	LS Acknowledge
10.0.4.1	224.0.0.5	OSPF	LS Acknowledge
10.0.4.2	224.0.0.5	OSPF	LS Update
10.0.4.1	10.0.4.2	OSPF	LS Update
10.0.4.2	224.0.0.5	OSPF	LS Acknowledge
10.0.4.2	10.0.4.1	OSPF	LS Update
10.0.4.1	224.0.0.5	OSPF	LS Acknowledge

Figure 3 – OSPF routing data exchange

6. Stop the Router1-Router2 Wireshark packet capture

> **NOTE:** Do not close the window yet! We will come back to this in the next section.

7. Wait a minute for OSPF to fully exchange routing tables for the entire network…

Figure Zzzzzz

8. Test the environment

    8.1. Request an IP address to devices in both the red and blue subnets to verify DHCP is operational

    8.2. From the PC1 console, trace the route taken to PC3

```
> trace 10.0.2.100 -P 1
```

```
PC1> trace  10.0.2.100 -P 1
trace to 10.0.2.100, 8 hops max (ICMP), press Ctrl+C to stop
 1   10.0.0.1   1.677 ms  0.785 ms  0.572 ms
 2   10.0.4.2   2.724 ms  2.459 ms  1.053 ms
 3   10.0.2.100   1.644 ms  0.904 ms  0.593 ms

PC1>
```

Figure 4 – Tracing connection between PC1 and PC3

8.3. Cut the connection between Router1 and Router2

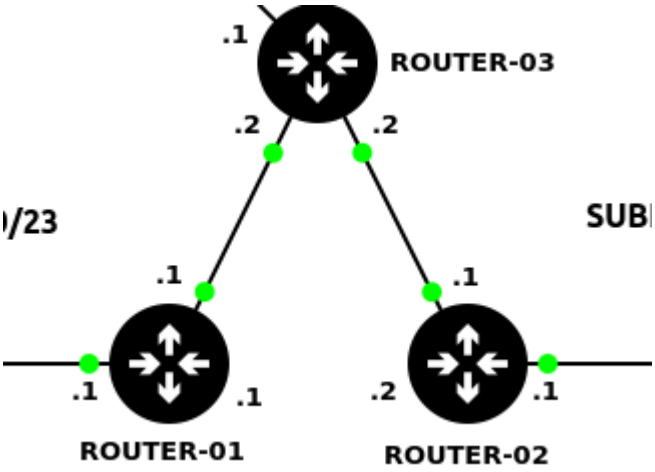


Figure 5 – Cutting Router1-Router2 link

8.4. Retrace the PC1-PC3 route to verify it can dynamically update the optimal network path

```
> trace 10.0.2.100 -P 1
```

```
PC1> trace  10.0.2.100 -P 1
trace to 10.0.2.100, 8 hops max (ICMP), press Ctrl+C to stop
 1   10.0.0.1   2.143 ms  1.164 ms  0.955 ms
 2   10.0.5.2   3.582 ms  1.458 ms  0.518 ms
 3   10.0.6.1   1.009 ms  0.817 ms  0.805 ms
 4   10.0.2.100   1.236 ms  1.047 ms  0.981 ms

PC1>
```

Figure 6 – Tracing connection between PC1 and PC3

OSPF Troubleshooting

The following router commands are useful in troubleshooting errors you might encounter. Below is the expected output for **Router1**.

1. All created OSPF instances. In this example, there should only be one per router.

```
> routing ospf instance print
```

```
[admin@ROUTER-01] > routing ospf instance print
Flags: X - disabled, I - inactive
 0   name="OSPF-ROUTER-01" version=2 vrf=main router-id=10.255.255.1
[admin@ROUTER-01] >
```

Figure 7 – Router1 OSPF instance

2. All created OSPF areas. In this example, there should only be one ***backbone*** area per router.

```
> routing ospf area print
```

```
[admin@ROUTER-01] > routing ospf area print
Flags: X - disabled, I - inactive, D - dynamic; T - transit-capable
 0    name="backbone" instance=OSPF-DATA area-id=0.0.0.0 type=default
[admin@ROUTER-01] >
```

Figure 8 – Router1 OSPF area

2. Instances set to be configured with OSPF. Your output may vary depending whether you specified individual interfaces or networks.

```
> routing ospf interface-template print
```

```
[admin@ROUTER-01] > routing ospf interface-template print
Flags: X - disabled, I - inactive
 0   area=backbone interfaces=all instance-id=0 type=broadcast
     retransmit-interval=5s transmit-delay=1s hello-interval=10s
     dead-interval=40s priority=128 cost=1
[admin@ROUTER-01] >
```

Figure 9 – Router1 OSPF interfaces

3. All current OSPF neighbors currently sharing routing information. There should be two neighbors listed: Router2 and Router3. Pay attention to the ***router-id*** values to identify which is which.

```
> routing ospf neighbor print
```

```
[admin@ROUTER-01] > routing ospf neighbor print
Flags: V - virtual; D - dynamic
 0  D instance=OSPF-ROUTER-01 area=backbone address=10.0.4.2 priority=128
      router-id=10.255.255.2 dr=10.0.4.1 bdr=10.0.4.2 state="Full"
      state-changes=6 adjacency=15m59s timeout=32s

 1  D instance=OSPF-ROUTER-01 area=backbone address=10.0.5.2 priority=128
      router-id=10.255.255.3 dr=10.0.5.1 bdr=10.0.5.2 state="Full"
      state-changes=6 adjacency=3m58s timeout=32s
[admin@ROUTER-01] >
```

Figure 10 – Router1 OSPF neighbors

NOTE: If the Router1-Router2 link is still cut, there will only be one neighbor shown until this connection is restored.

5. All routes currently known by the host router. This should contain every subnet ID on this network.

```
> ip route print
```

```
[admin@ROUTER-01] > ip route print
Flags: D - DYNAMIC; A - ACTIVE; c - CONNECT, o - OSPF; + - ECMP
Columns: DST-ADDRESS, GATEWAY, DISTANCE
     DST-ADDRESS      GATEWAY          DISTANCE
DAc  10.0.0.0/23      ether1                0
DAo  10.0.2.0/24      10.0.4.2%ether2     110
DAo  10.0.3.0/29      10.0.5.2%ether3     110
DAc  10.0.4.0/30      ether2                0
DAc  10.0.5.0/30      ether3                0
DAo+ 10.0.6.0/30      10.0.5.2%ether3     110
DAo+ 10.0.6.0/30      10.0.4.2%ether2     110
DAc  10.255.255.1/32  loopback              0
DAo  10.255.255.2/32  10.0.4.2%ether2     110
DAo  10.255.255.3/32  10.0.5.2%ether3     110
[admin@ROUTER-01] >
```

Figure 11 – Router1 routing table

Phase IV – Dissecting OSPF Traffic

OSPF is a noisy protocol when no constraints are made. You should see many different kinds of packets appearing on the Wireshark capture.

1. Focus on the previous Router1-Router2 Wireshark capture

NOTE: You can always generate more OSPF traffic by deleting then restoring any router-adjacent connection.

1.1. Filter only for *OSPF* packets

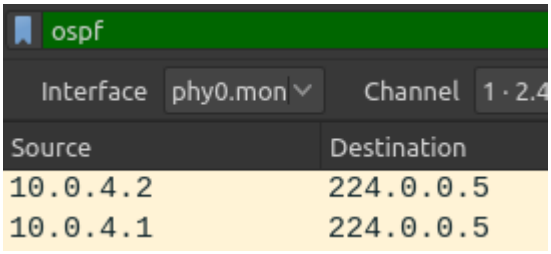


Figure 12 – Filtered Wireshark capture

1.2. **[Hello Packet]**

These packets are sent every 10 seconds (default) out of configured interfaces. The Hello Packets are used to discover OSPF neighbors and help build adjacency. Notice that the destination for the Hello Packets is 224.0.0.5. This is the broadcast address for the OSPF protocol.

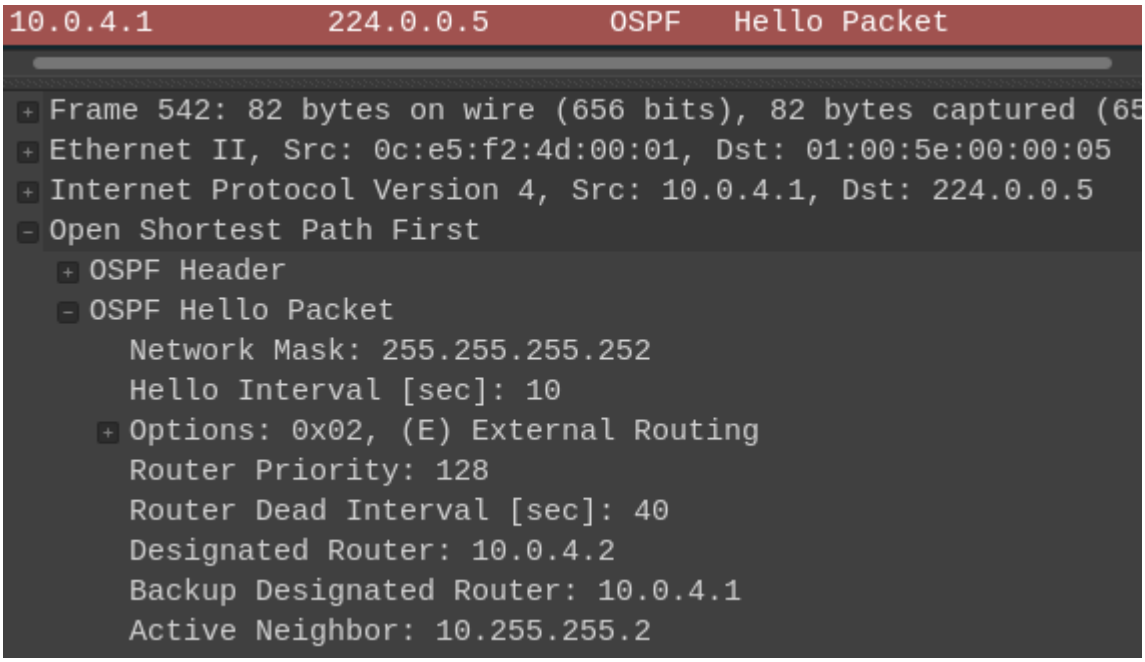


Figure 13 – OSPF Hello

1.3. **[DB Description]**

Database description packets are distributed after the OSPF handshake between two routers has been established. Here, they will advertise the current state of their internal OSPF database. In the example below, Router2 is telling Router1 that it currently has five links to offer: three directly connected and two remote.

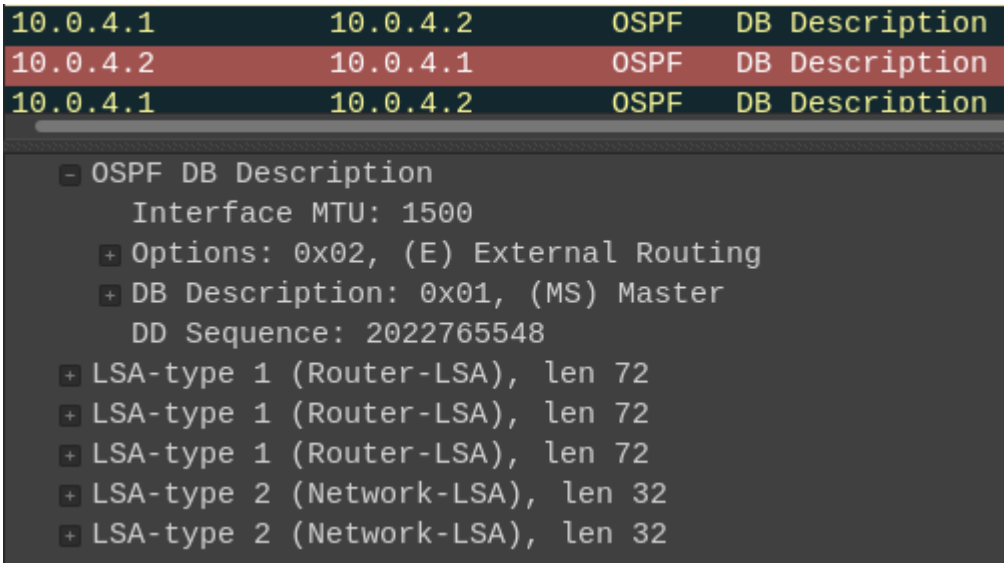


Figure 14 – OSPF database descriptor

1.4. **[LS Update]**

Link state update packets are used exchange network information between OSPF neighbors. This will occur any time routing information is altered, such as a cable being cut, an interface going offline, or the addition of new links. The example below shows Router2 advertising the networks 10.0.2.0/24 and 10.255.255.2/32 to Router1.

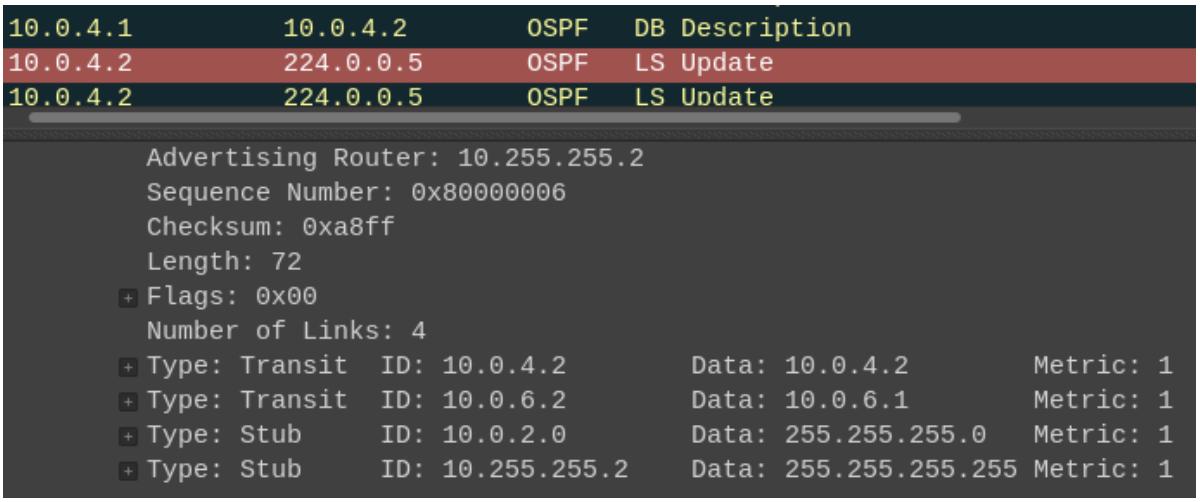


Figure 15 – OSPF link-state update

1.5. **[LS Request]**

After an LS Update is received, the router will transmit a link state update packet for further information about the network. This will then be followed by additional LS Update packets contain the requested data.

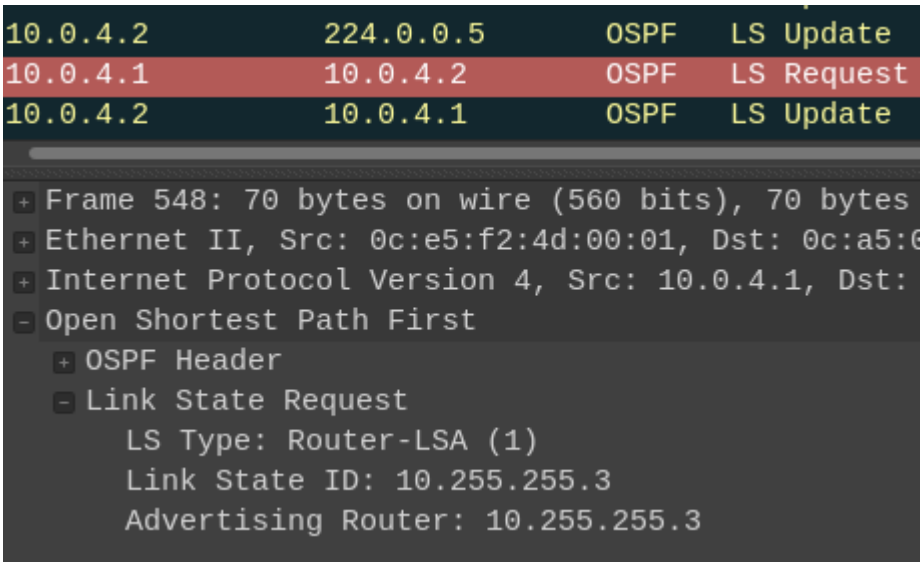


Figure 16 – OSPF link-state request

1.6. **[LS Acknowledge]**

An acknowledgment of given after every LS Update is received.

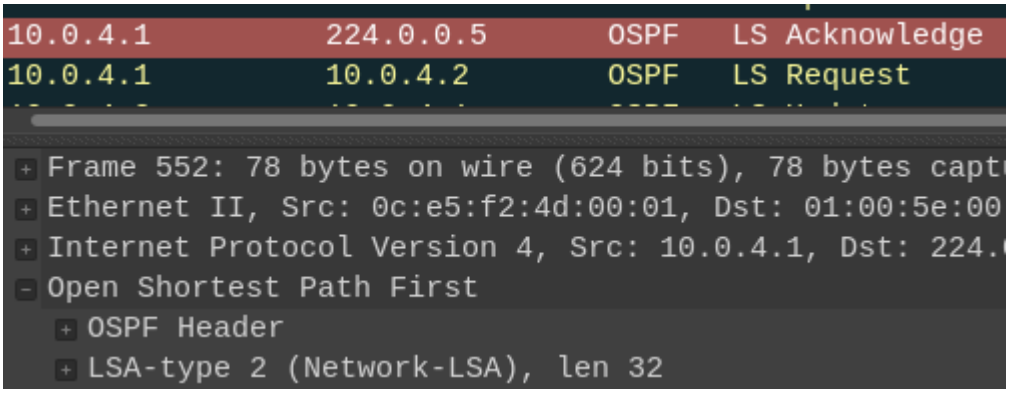


Figure 17 – OSPF link-state acknowledgement

End of Lab

Deliverables

5 screenshots are required to receive full credit for this assignment

- GNS3 working environment will have all devices on and labeled correctly
- A router display of all the IP routes – all four router IDs should be visible
- Wireshark showing OSPF Hello Packets
- Wireshark shows ICMP packets between PC1 and PC3 using one path
- Wireshark shows ICMP packets between PC1 and PC3 using a different path

Homeworks

Assignment 1 – Update the network build in Assignment 1 from the previous chapter

- Configure DHCP to support the network
- Replace RIPv2 routing with OSPF
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled including Router IDs
    - Screenshot of the DHCP configuration
    - Screenshot of OSPF packets
    - Wireshark Packet Captures where a Green host can ping
        - Red Host
        - Blue Host
        - Gray Host
- Sample network environment:

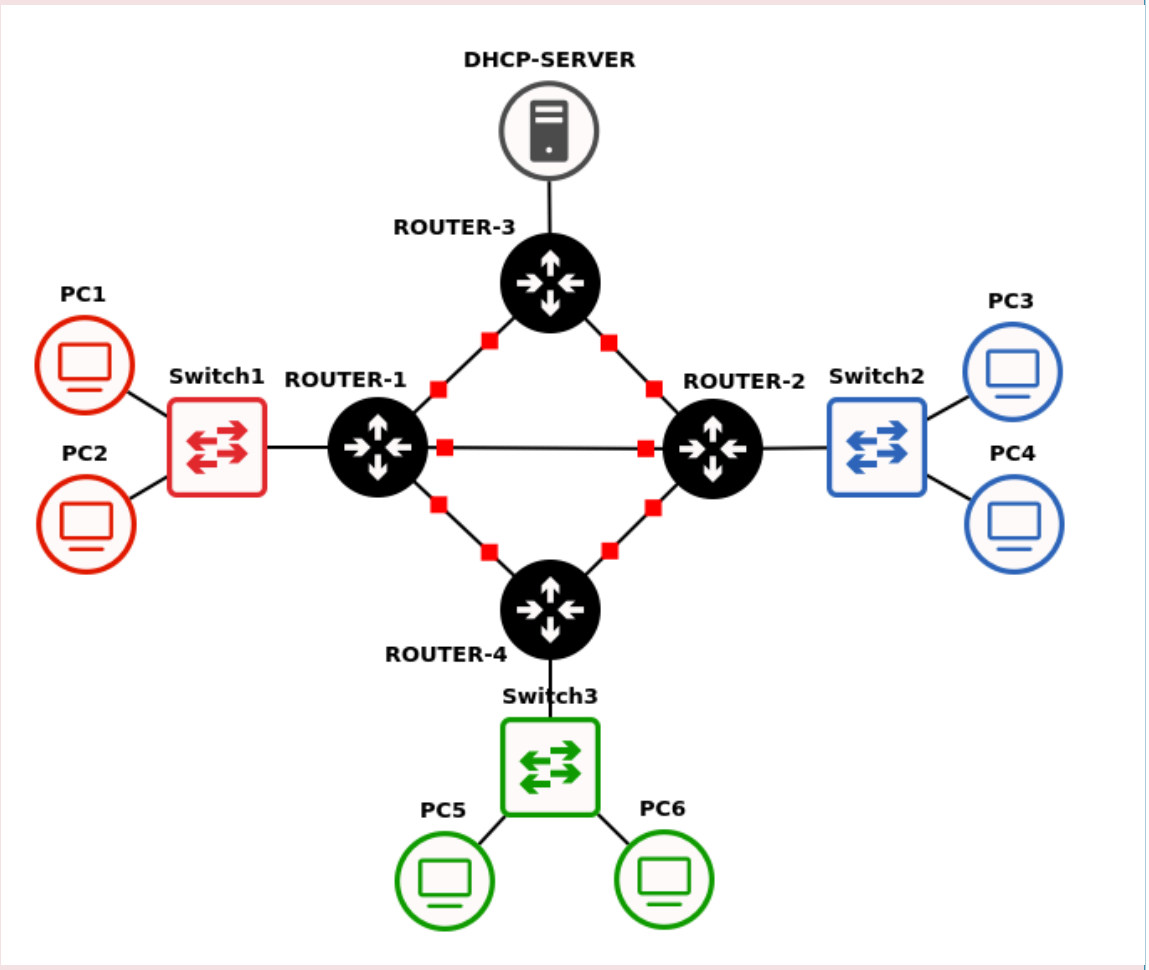


Figure 18 – Assignment 1 network

Assignment 2 – Update the network build in Assignment 2 from the previous chapter

- Configure DHCP to support the network
- Replace RIPv2 routing with OSPF
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled including Router IDs
    - Trace route command showing that an ICMP packet from a Blue host takes different paths to reach the Purple host (You may have to disconnect some connections to force the change in path)
        - Router2 –> Router5
        - Router2 –> Router1 -> Router5
        - Router2 –> Router3 -> Router1 -> Router5
        - Router2 –> Router3 -> Router1 -> Router4 -> Router5

- Sample network environment:

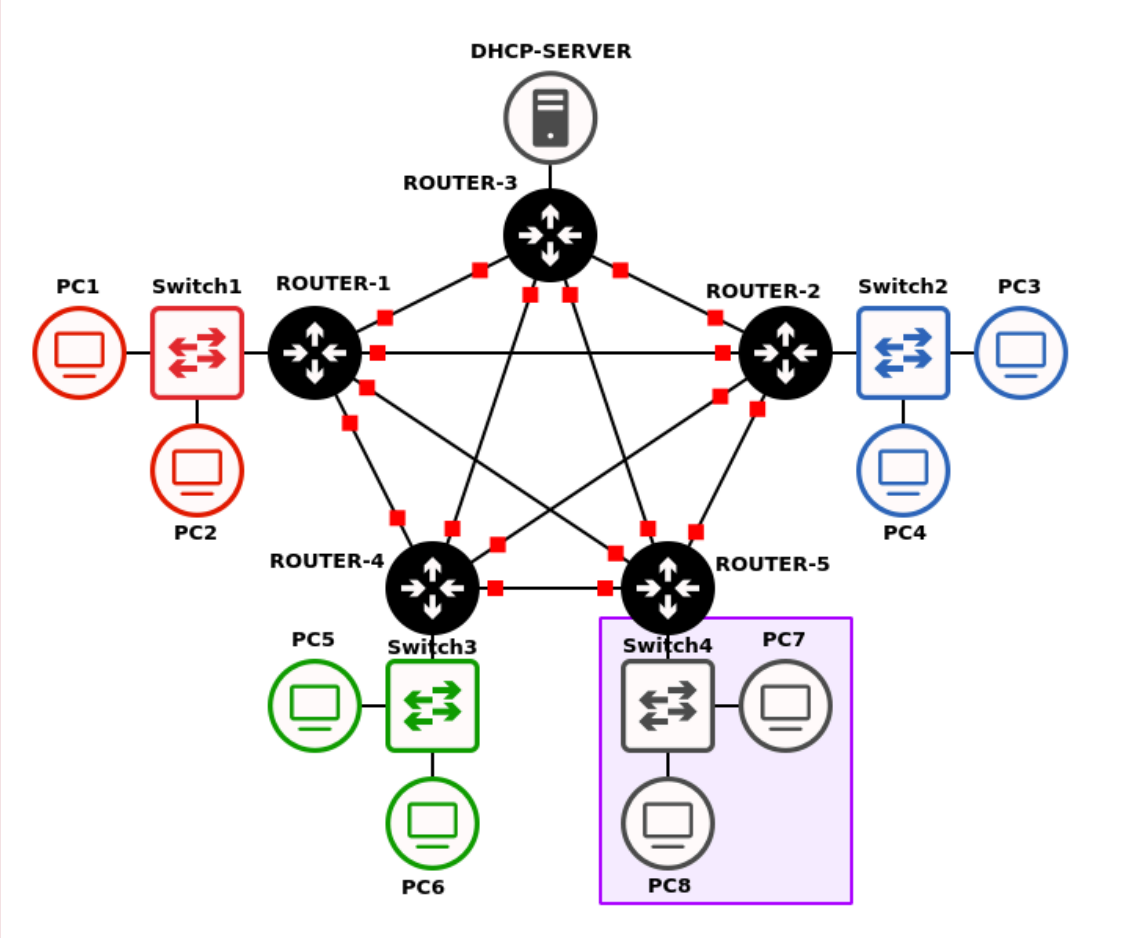


Figure 19 – Assignment 2 network

Assignment 3 – Preparation for BGP lab

- Create the following OSPF full-mesh network
- This will be used in the setup for the next chapter – Border Gateway Protocol Networking
- Network environment
  < insert image >

CHAPTER 29

Dynamic Networking - Border Gateway Protocol

JACOB CHRISTENSEN AND MATHEW J. HEATH VAN HORN, PHD

Although OSPF has fast convergence rates, it can put a lot of strain on computing resources as networks become larger, making it better suited within a LAN or autonomous system (AS). The Border Gateway Protocol (BGP) on the other hand is the only networking protocol currently in use that can handle the Internet's ever-increasing size and complexity while minimizing overhead.

Estimated time for completion: 65 minutes

LEARNING OBJECTIVES

- Learn how to implement BGP in an enterprise network
- Be able to identify and understand BGP packets in Wireshark

PREREQUISITES

- Chapter 28 – OSPF Networking

DELIVERABLES

- Screenshot of GNS3 Workspace with all devices labeled
- Wireshark capture of the TCP and BGP packets being exchanged
- Wireshark view of the keep alive messages
- Wireshark view of the update packets with the NLRI view

RESOURCES

- MikroTik RouterOS Documentation – BGP – https://help.mikrotik.com/docs/pages/viewpage.action?pageId=328220

CONTRIBUTORS AND TESTERS

- Dante Rocca, Cybersecurity student, ERAU-Prescott

A note from the authors:

At the beginning of this book, we focused on how end devices communicate with each other. We used overly-simplistic definitions, but they helped organize our focus on the topics covered. If we abstract ourselves into a video-game perspective looking down on our sims, we can zoom in and out and break our network functions into some abstract views. This is still an oversimplification of how networks work, but it helps us conceptualize what we have learned and where we are going in our learning.

- 100' level
    - Immediate area or workspace, such as an office work center or a living room
    - LAN – End devices connected to each other by a switch or a hub
    - We see many end devices
- 500' level
    - Multiple individual workspaces connected such as an office department or a home
    - Network – LANs connected to each other by a router
    - We see fewer end devices
- 5,000' level
    - We use routers to extend our connectivity such as neighborhoods or a corporate enterprise
    - Enterprise Network – Routers connected to each other within one enterprise
    - We see one end device

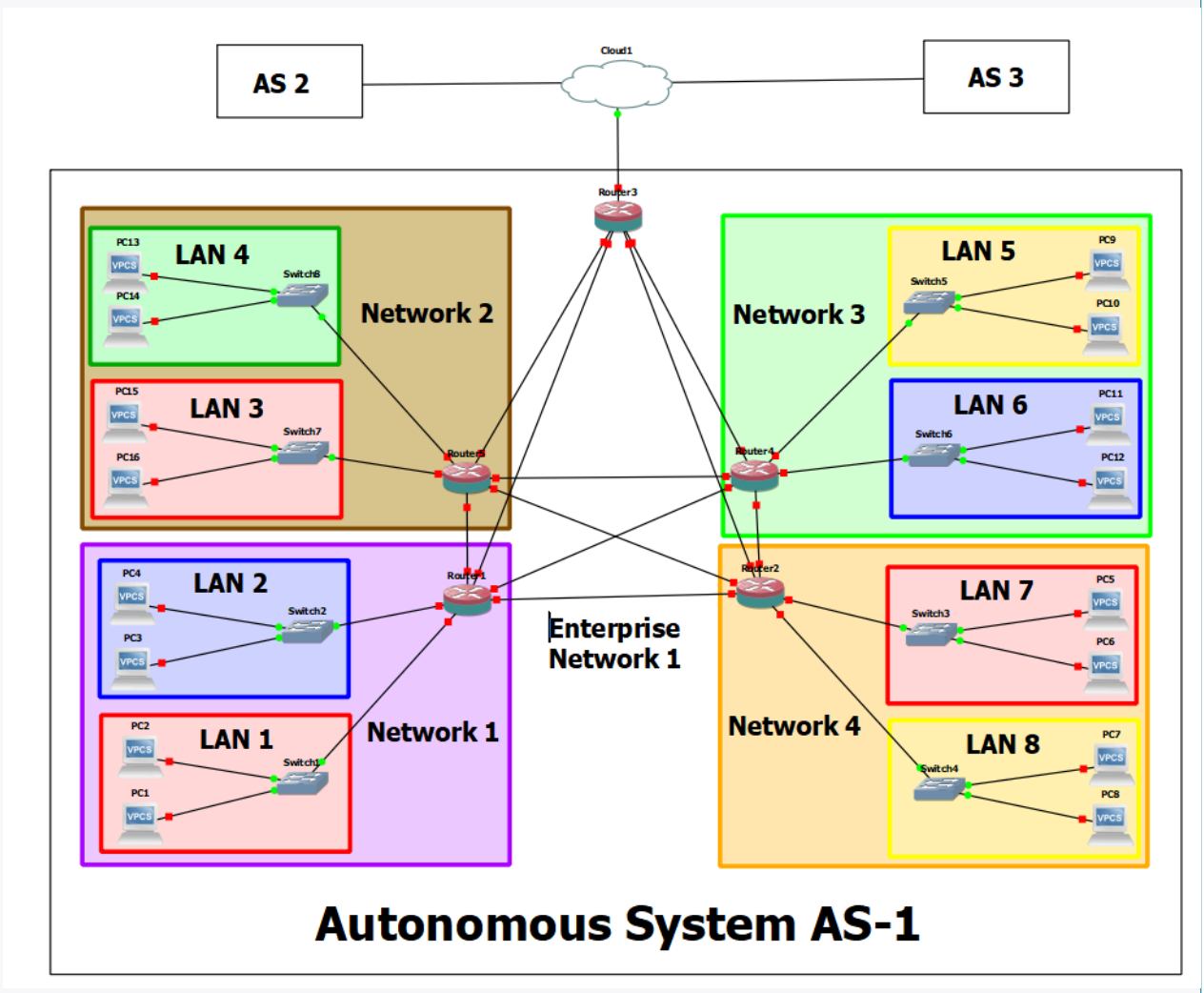


Figure 1 – Complete Autonomous System

In this chapter, we are going to zoom out once again to about 20,000' and look at how our enterprise connects to other enterprises. We will abstract our Enterprise Network and now call it an Autonomous System. At this point, our video-game view would not see any end devices or LAN devices such as switches and routers.

However, we still think learners need to see how end devices share information with each other through the network. Therefore, in this chapter, we are going to abstract some of the intermediary functions we have already learned. Here are some things to keep in mind as you complete this lab:

- A host machine is used to represent an entire LAN
- A router is used to represent an entire Network
- A border router is used to represent an Enterprise Network (Autonomous System)

Phase I – Building the Network Topology

The following steps are to create a baseline for completing the lab. It makes assumptions about learner knowledge from completing previous labs.

Terminology used in this lab:

- **Autonomous System (AS) –** A network or cluster of networks following the same routing policies. Typically, each AS is controlled by a single entity.
- **Autonomous System Number (ASN)** – A 16-bit integer assigned to an AS for identification purposes.
- **Autonomous System Border Router (ASBR)** – Routers at the edge of an AS that connects to external networks.

By the end of this chapter, your network topology should look like the following:

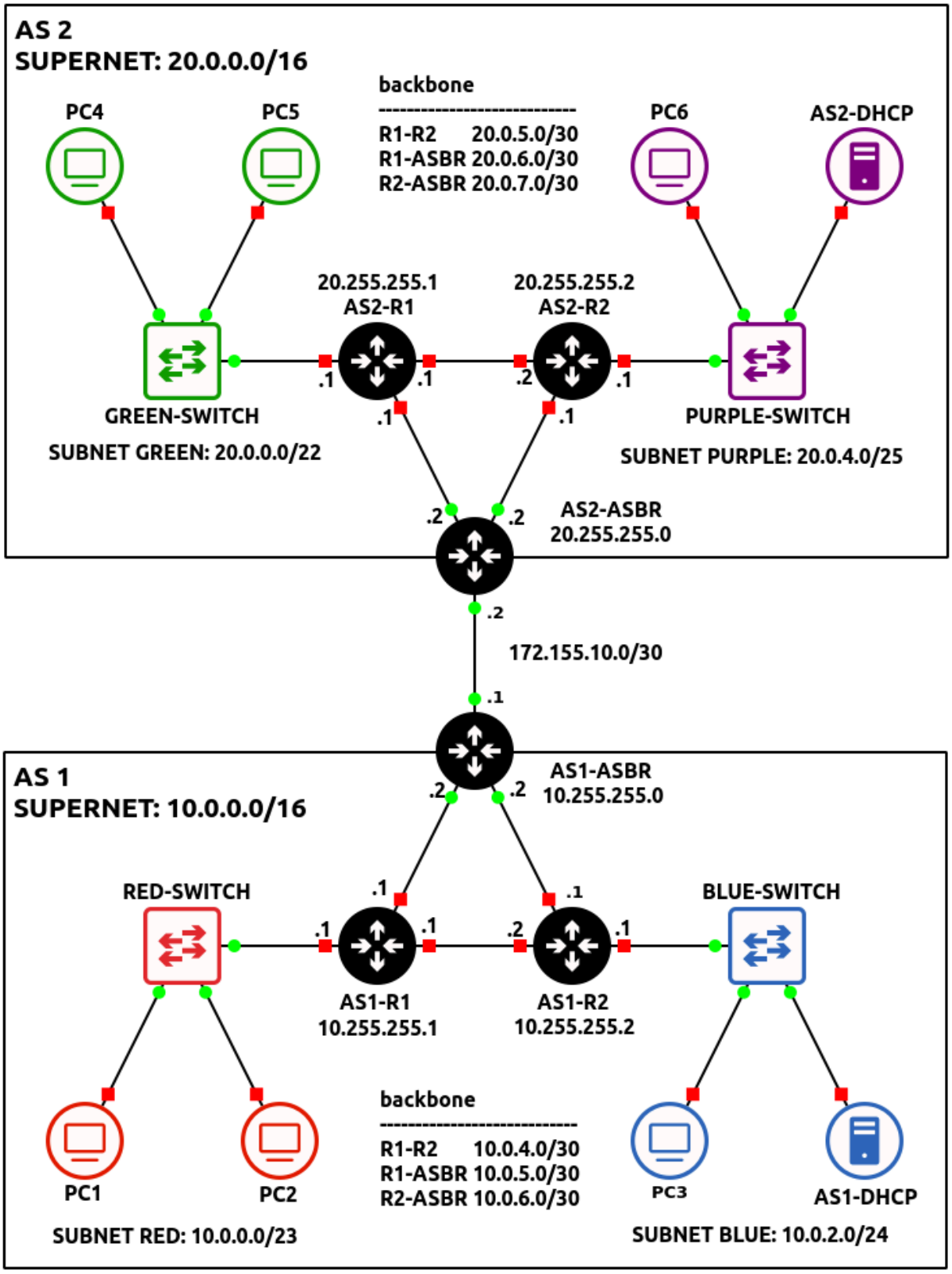

AS 2
SUPERNET: 20.0.0.0/16
backbone
R1-R2 20.0.5.0/30
R1-ASBR 20.0.6.0/30
R2-ASBR 20.0.7.0/30
PC4
PC5
PC6
AS2-DHCP
20.255.255.1
AS2-R1
20.255.255.2
AS2-R2
GREEN-SWITCH
PURPLE-SWITCH
SUBNET GREEN: 20.0.0.0/22
SUBNET PURPLE: 20.0.4.0/25
AS2-ASBR
20.255.255.0
172.155.10.0/30
AS1-ASBR
10.255.255.0
AS 1
SUPERNET: 10.0.0.0/16
RED-SWITCH
BLUE-SWITCH
AS1-R1
10.255.255.1
AS1-R2
10.255.255.2
backbone
R1-R2 10.0.4.0/30
R1-ASBR 10.0.5.0/30
R2-ASBR 10.0.6.0/30
PC1
PC2
PC3
AS1-DHCP
SUBNET RED: 10.0.0.0/23
SUBNET BLUE: 10.0.2.0/24

Figure 2 – Final GNS3 environment

1. Start GNS3

    1.1. Open the previous Chapter 28 lab

    1.2. Save it as a new project: **LAB_15**

2. Modify the network environment to function as **AS-1**

    2.1. Remove the Gray subnet

    2.2. Replace PC4 with a DHCP server

    2.3. Router3 will now act as the networks border router (AS1-ASBR)

> **NOTE:** In this example, AS1-ASBR's Router ID will be **10.255.255.0**. If you are reusing a previously configured router, and want to change the router-id value, ensure to adjust your previous loopback/OSPF configurations as necessary.

    2.4. Adjust OSPF to only operate only on ***internal facing*** ethernet ports

> **NOTE:** Remember that the purpose of a border router is to manage packets that are leaving (egress) or entering (ingress) through a network. With this in mind, some of its ports will be categorized as ***internal*** and ***external***. Below is a simplified example of the network we are trying to build to illustrate this idea clearly.

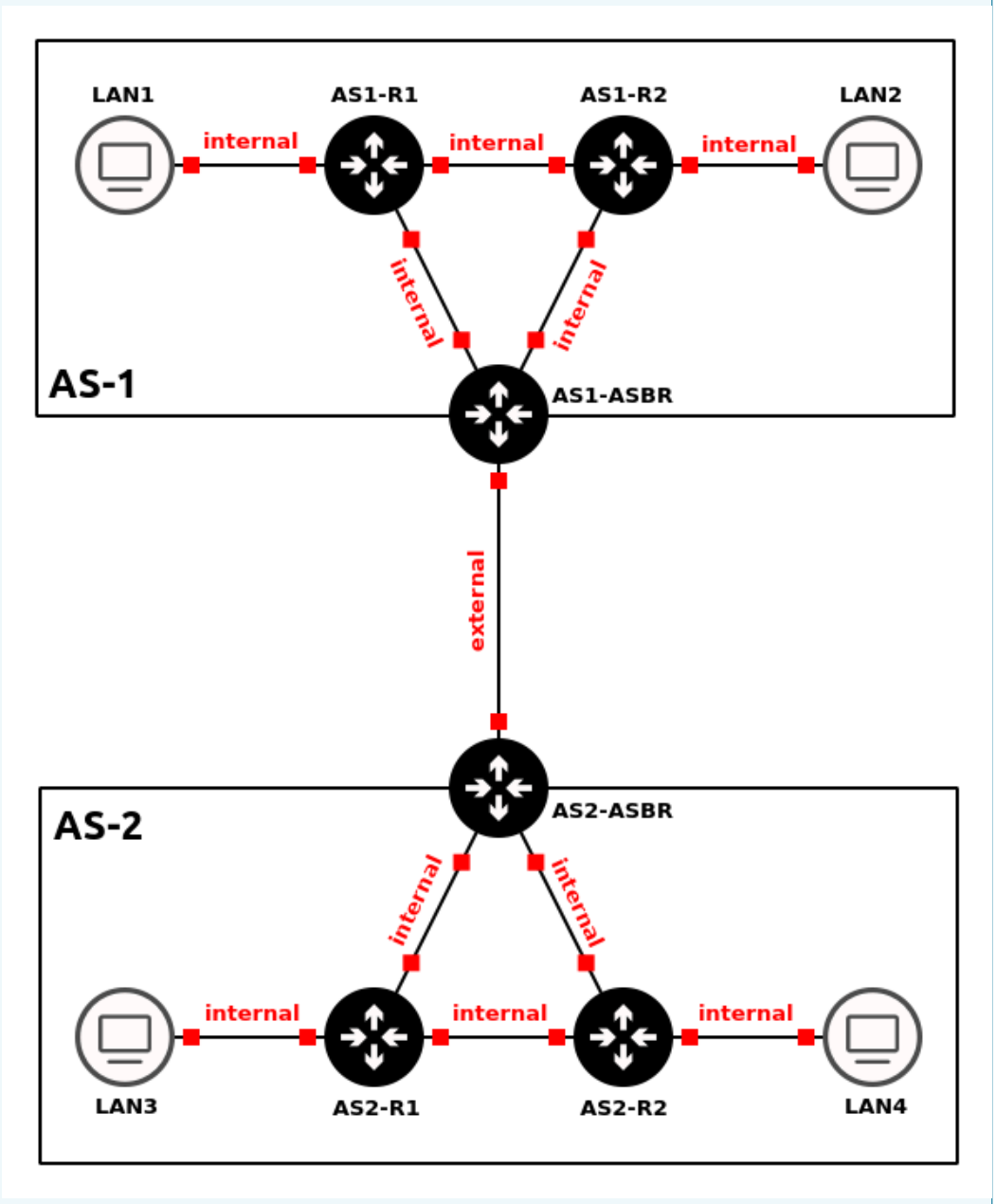


Figure 3 – Internal vs external links

In the previous chapter, we configured OSPF to operate on ALL active interfaces.

```
[admin@AS1-ASBR] > routing ospf interface-template print
Flags: X - disabled, I - inactive
 0   area=backbone interfaces=all instance-id=0 type=broadcast
     retransmit-interval=5s transmit-delay=1s hello-interval=10s
     dead-interval=40s priority=128 cost=1
[admin@AS1-ASBR] >
```

Figure 4 – OSPF configured on all interfaces

However, here we only want to share routes that are within the AS-1 network. Therefore, OSPF should only be configured to operate on internal interfaces. In this example, ***ether2*** is connected to AS1-R1 and ***ether3*** is connected to AS1-R2. These are inward-facing (internal) ports. In contrast, ***ether1*** will be used as the network's outward-facing (external) interface. Below is my OSPF interface configuration for AS1-ASBR.

```
[admin@ROUTER-03] > routing ospf interface-template print
Flags: X - disabled, I - inactive
 0   area=backbone interfaces=ether2,ether3 instance-id=0 type=broadcast
     retransmit-interval=5s transmit-delay=1s hello-interval=10s
     dead-interval=40s priority=128 cost=1
[admin@ROUTER-03] >
```

Figure 5 – OSPF configured on select interfaces

2.5. Configure AS1-DHCP to service both the Red and Blue subnets

NOTE: Remember to adjust and/or remove your DHCP-relays as you modify the network.

2.5.1. Ensure that all devices can receive IPv4 addresses

2.5.2. Ensure that all devices can ping one another

2.6. Assign default routes for each ***internal router*** that points to AS1-ASBR's inward-facing addresses

NOTE: In most cases, it is impossible for a router to know the location and hop distance of every single network at all times. Sometimes networks are just too big to store that much information in a single routing table. Because of this, ***default routes*** are used to forward packets automatically with unknown/foreign destination addresses (represented as 0.0.0.0/0) to another router who might know the answer. In the context of this network, our ***internal routers*** (AS1-R1 and AS1-R2) should only contain records of subnets within the 10.0.0.0/16 supernet, thanks to our OSPF configuration. Therefore, packets trying to reach outside this network should be sent to the border gateway router (AS1-ASBR), which might contain this information.

2.6.1. Type the following commands in AS1-R1

> **NOTE:** We are assigning two default routes because AS1-ASBR has two inward-facing interfaces. If one does down, the other will serve as a backup for redundancy. Remember, route priority is based on distance value.

```
> ip route add dst-address=0.0.0.0/0 gateway=10.0.5.2 distance=1
```

```
> ip route add dst-address=0.0.0.0/0 gateway=10.0.4.2 distance=2
```

2.6.2. Type the following commands in AS1-R2

```
> ip route add dst-address=0.0.0.0/0 gateway=10.0.6.2 distance=1
```

```
> ip route add dst-address=0.0.0.0/0 gateway=10.0.4.1 distance=2
```

2.7. Label and organize your network as necessary

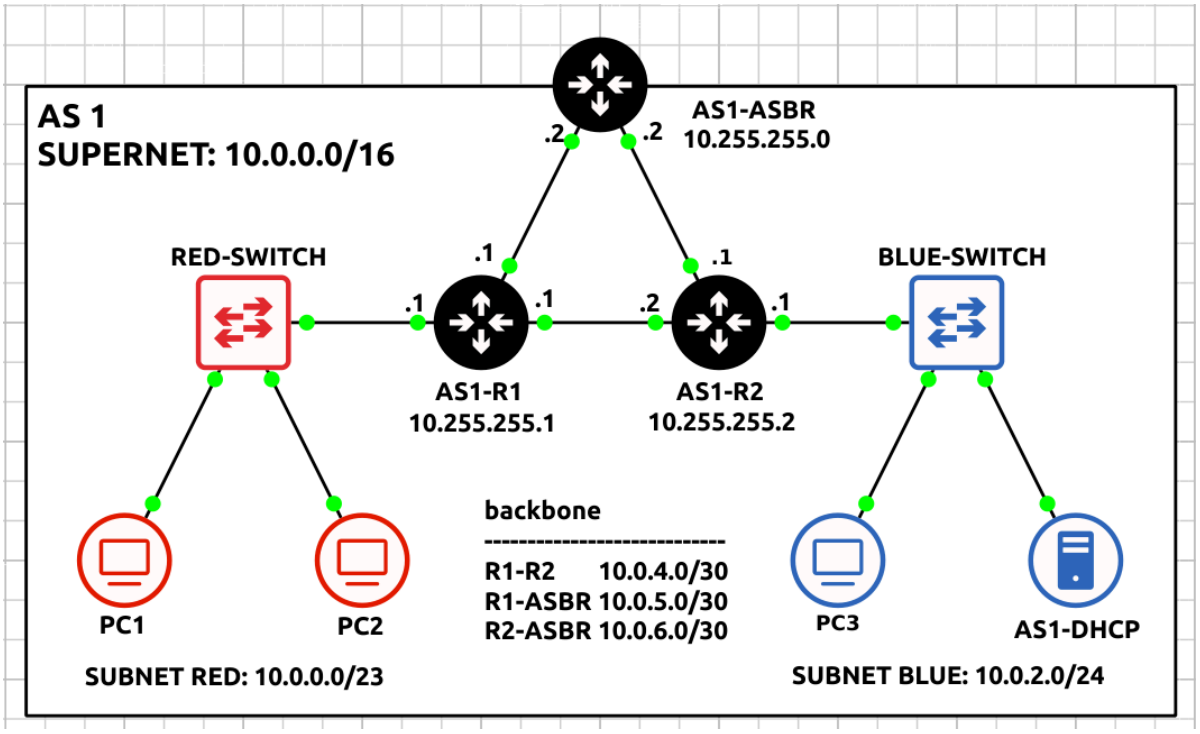


Figure 6 – AS1 network complete

3. Build a small network analogous with the following specifications to function as **AS-2**:

    3.1. Class B Supernet – **20.0.0.0/16**

Host Range	
Host Lower Bound	**20.0.0.1**
Host Upper Bound	**20.0.255.254**

    3.2. Subnet – **Green**

        3.2.1. One switch – ***Ethernet switch***

        3.2.2. Two client machines – ***VPCS***

        3.2.3. Minimize wasted address space for ***1000 hosts***

Network Information	
Network	**20.0.0.0**
Netmask	**255.255.252.0 (/22)**
Broadcast	**20.0.3.255**
Gateway	**20.0.0.1**
DHCP Lower Bound	**20.0.0.2**
DHCP Upper Bound	**20.0.3.254**

    3.3. Subnet – **Purple**

        3.3.1. One switch – ***Ethernet switch***

        3.3.2. One client machine – ***VPCS***

        3.3.3. One DHCP server – ***Ubuntu / Windows / MikroTik CHR***

        3.3.4. Minimize wasted address space for ***100 hosts***

Network Information	
Network	**20.0.4.0**
Netmask	**255.255.255.128 (/25)**
Broadcast	**20.0.4.127**
Gateway	**20.0.4.1**
DHCP Lower Bound	**20.0.4.3**
DHCP Upper Bound	**20.0.4.126**

3.4. Subnet – **Backbone**

3.4.1. Three routers –***MikroTik CHR***

3.4.2. Full-mesh topology

3.4.3. Minimize wasted address space for each router-to-router connection

Connection	Network
AS2-R1 <-> AS2-R2	**20.0.5.0/30**
AS2-R1 <-> AS2-ASBR	**20.0.6.0/30**
AS2-R2 <-> AS2-ASBR	**20.0.7.0/30**

3.5. Configure OSPF to only operate only on ***internal facing*** ethernet ports

3.6. Configure AS2-DHCP to service both Green and Purple subnets

3.6.1. Ensure that all devices can receive IPv4 addresses

3.6.2. Ensure that all devices can ping one another

3.7. Assign default routes for each ***internal router*** that points to AS2-ASBR

3.7.1. Type the following command in AS2-R1

```
> ip route add dst-address=0.0.0.0/0 gateway=20.0.6.2 distance=1
```

```
> ip route add dst-address=0.0.0.0/0 gateway=20.0.5.2 distance=2
```

3.7.2. Type the following command in AS2-R2

```
> ip route add dst-address=0.0.0.0/0 gateway=20.0.7.2 distance=1
```

```
> ip route add dst-address=0.0.0.0/0 gateway=20.0.5.1 distance=2
```

3.8. Label and organize your network as necessary

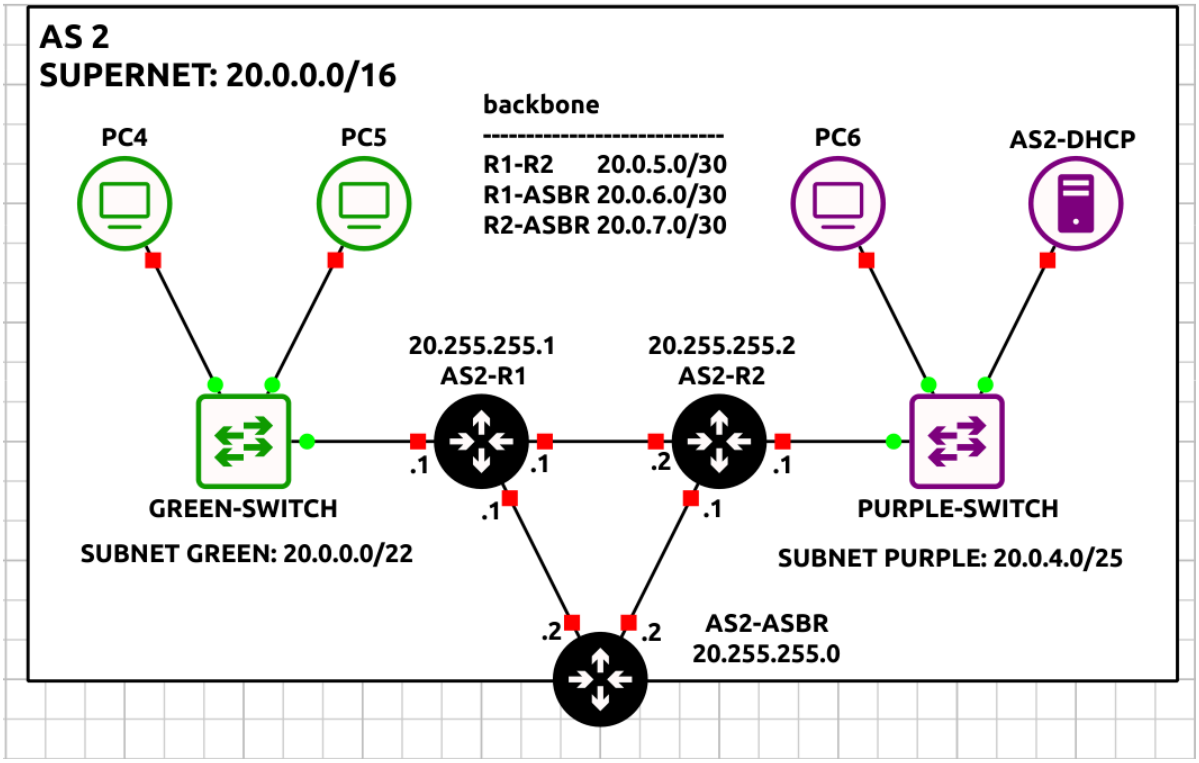
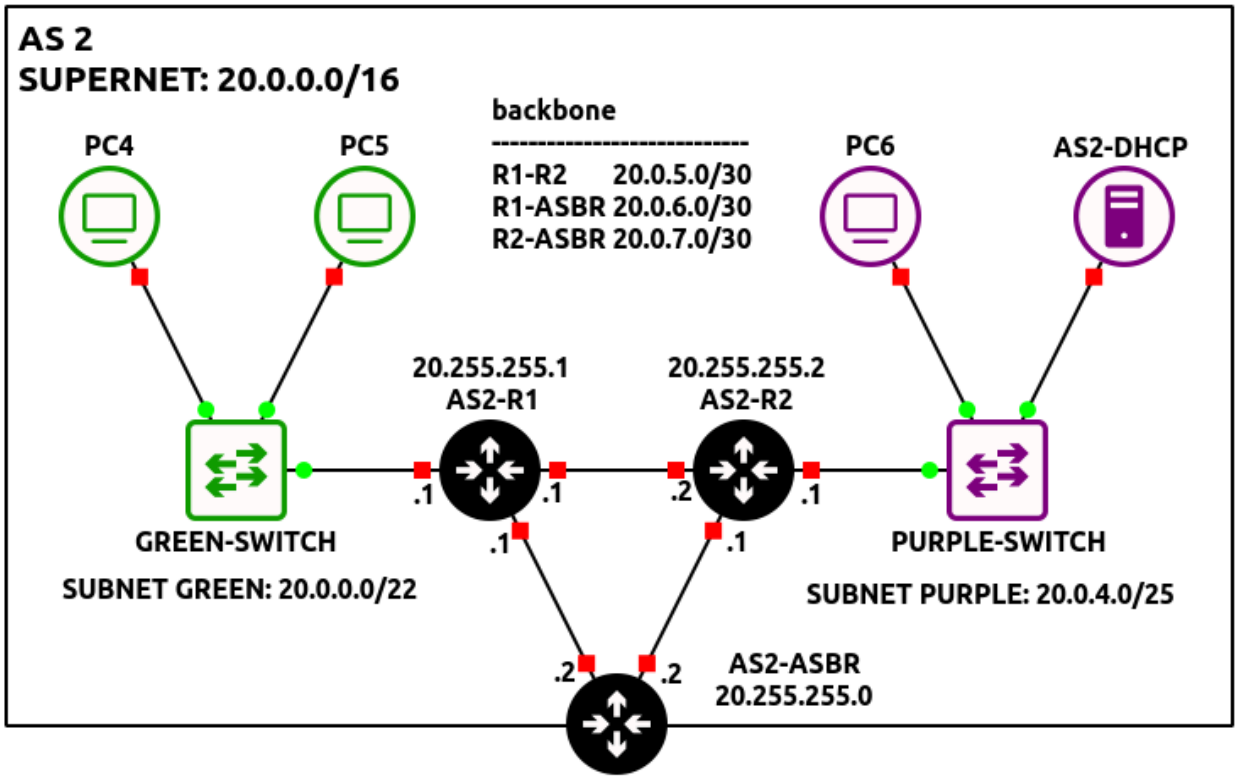


Figure 7 – AS2 network complete

Phase II – Configuring BGPv4 on MikroTik RouterOS

BGPv4, the routing protocol for connecting independent autonomous systems (AS), is used between border routers to relay any information necessary by each AS. It's vital to understand that BGP is utilized by many ASNs because they may not all be part of the same entity, as well as to give administrators more control over how data is transferred between destinations.

1. At this point, you should two small networks that resemble the following figure:


AS 2
SUPERNET: 20.0.0.0/16
backbone
R1-R2 20.0.5.0/30
R1-ASBR 20.0.6.0/30
R2-ASBR 20.0.7.0/30
PC4
PC5
PC6
AS2-DHCP
20.255.255.1
AS2-R1
20.255.255.2
AS2-R2
GREEN-SWITCH
PURPLE-SWITCH
SUBNET GREEN: 20.0.0.0/22
SUBNET PURPLE: 20.0.4.0/25
AS2-ASBR
20.255.255.0

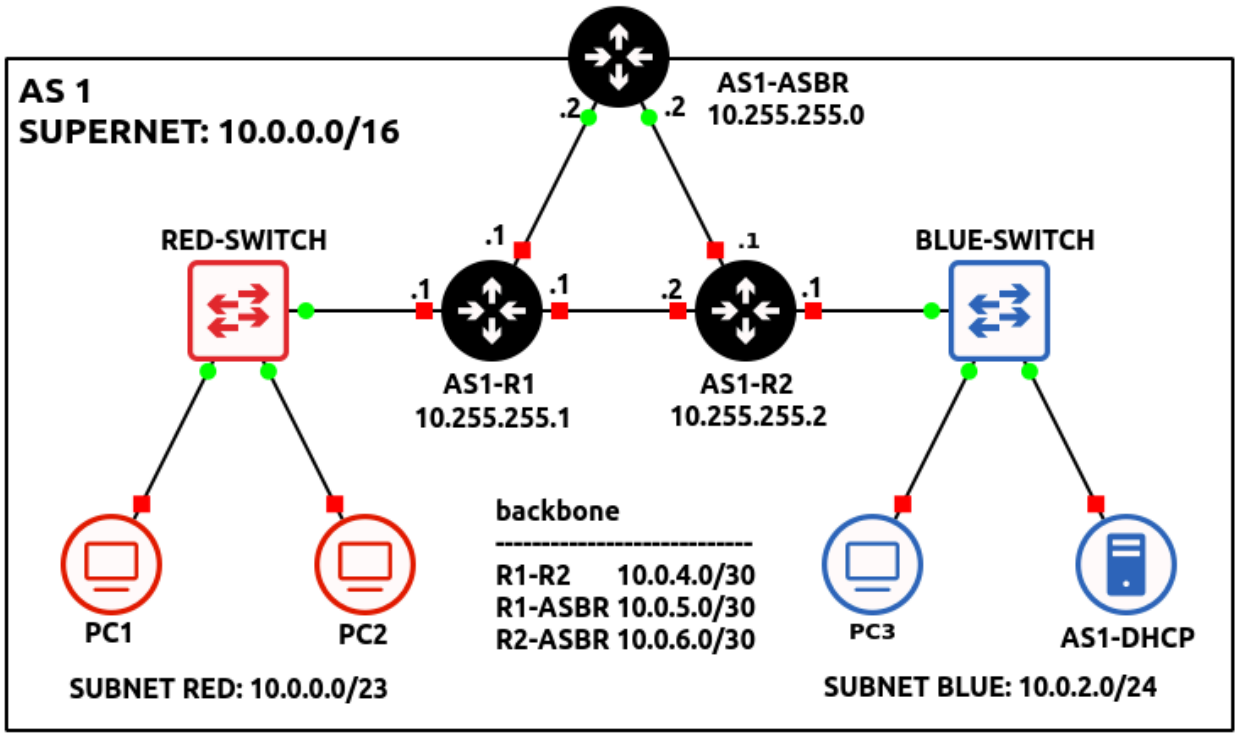

AS 1
SUPERNET: 10.0.0.0/16
AS1-ASBR
10.255.255.0
RED-SWITCH
BLUE-SWITCH
AS1-R1
10.255.255.1
AS1-R2
10.255.255.2
backbone
R1-R2 10.0.4.0/30
R1-ASBR 10.0.5.0/30
R2-ASBR 10.0.6.0/30
PC1
PC2
PC3
AS1-DHCP
SUBNET RED: 10.0.0.0/23
SUBNET BLUE: 10.0.2.0/24

Figure 8 – Both networks in GNS3

2. Connect two ASBR's together using a random network of your choice

> **NOTE:** In this example, ***ether1*** of both border routers are connected over the **172.155.10.0/30** network. Don't forget to update the interface IP addresses!

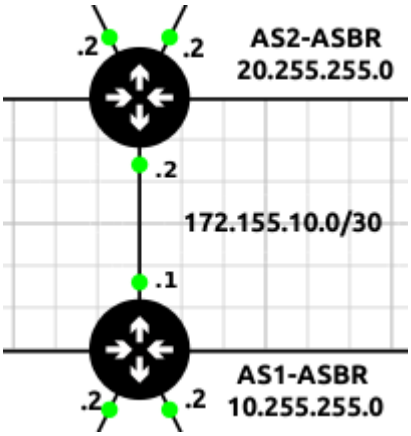


Figure 9 – Connecting AS1 and AS2

3. Initialize a Wireshark packet capture between the two AS networks

4. Create a new BGP instance on **AS1-ASBR**

```
  > routing bgp connection add name=HOST-TO-AS2 as=1 local.role=ebgp router-
id=10.255.255.0 remote.address=172.155.10.2 output.redistribute=connected,ospf
```

Command	Definition
name	Name of the new connection instance. This can be anything, but it should represent its function for best practice.
as	ASN integer of the local autonomous system.
local.role	Specifies whether BGP is being used internally (ibgp) or externally (ebgp). Since we are connecting two different AS-es, eBGP is preferred.
router-id	The identification of the local router for the receiving router to recognize.
remote.address	Specifies the remote interface address of the receiving router.
output.redistribute	Specified which routes to be shared with any connected BGP neighbors

5. In Wireshark, you should now see TCP SYN packets attempting to establish a BGP session with AS2-ASBR

> **NOTE:** You should ***NOT*** see any OSPF Hello packets on this link!!

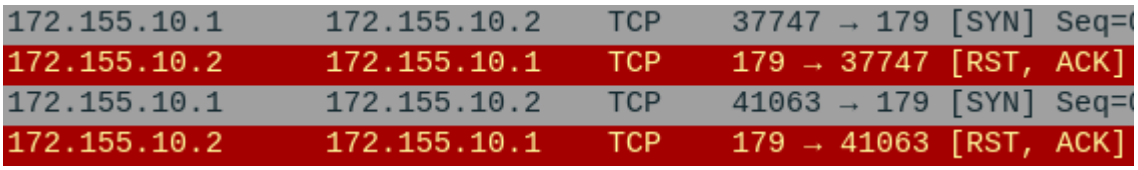


Figure 10 – Wireshark packet capture

6. Create a new BGP instance on **AS2-ASBR**

```
  > routing bgp connection add name=HOST-TO-AS1 as=2 local.role=ebgp router-
id=20.255.255.0 remote.address=172.155.10.1 output.redistribute=connected,ospf
```

7. You will know when you are successful if you see the following packets in Wireshark

> **NOTE:** Looking at Wireshark, you should notice a TCP handshake occur between the two routers followed by two OPEN messages and a steady stream of KEEPALIVE messages. This means that the connection was successful and both routers are able to communicate with each other. After every BGP packet, the recipient will respond with an obligatory TCP ACK segment, acknowledging a successful transmission of data.

172.155.10.2	172.155.10.1	TCP	35717 → 179 [SYN]
172.155.10.1	172.155.10.2	TCP	179 → 35717 [SYN,
172.155.10.2	172.155.10.1	TCP	35717 → 179 [ACK]
172.155.10.2	172.155.10.1	BGP	OPEN Message
172.155.10.1	172.155.10.2	TCP	179 → 35717 [ACK]
172.155.10.1	172.155.10.2	BGP	OPEN Message
172.155.10.2	172.155.10.1	TCP	35717 → 179 [ACK]
172.155.10.2	172.155.10.1	BGP	KEEPALIVE Message
172.155.10.1	172.155.10.2	TCP	179 → 35717 [ACK]
172.155.10.1	172.155.10.2	BGP	KEEPALIVE Message
172.155.10.2	172.155.10.1	TCP	35717 → 179 [ACK]
172.155.10.2	172.155.10.1	BGP	KEEPALIVE Message
172.155.10.1	172.155.10.2	TCP	179 → 35717 [ACK]
172.155.10.1	172.155.10.2	BGP	KEEPALIVE Message
172.155.10.2	172.155.10.1	TCP	35717 → 179 [ACK]
172.155.10.2	172.155.10.1	BGP	UPDATE Message
172.155.10.1	172.155.10.2	BGP	UPDATE Message
172.155.10.1	172.155.10.2	TCP	179 → 35717 [ACK]
172.155.10.2	172.155.10.1	TCP	35717 → 179 [ACK]

Figure 11 – Wireshark packet capture

8. You should also start to see UPDATE messages containing Network Layer Reachability Information (NLRI)

> **NOTE:** Looking at the details of this packet will reveal the network addresses being distributed by BGP. In the example below, AS2-ASBR is telling AS2 about the nine subnets on its network.

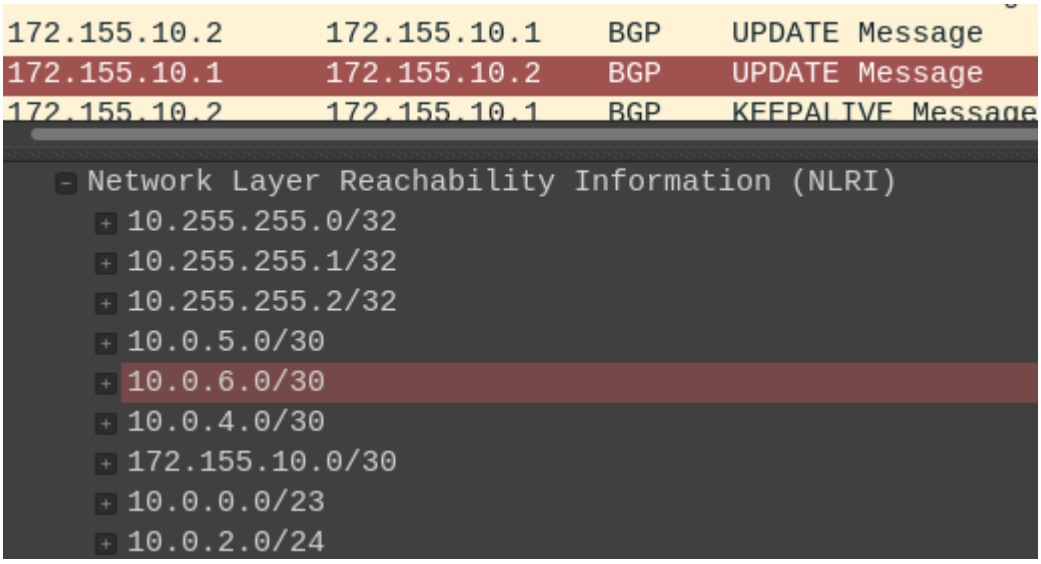


Figure 12 – BGP Update packet analysis

Phase III – Testing the BGP Connection

Now that we have a BGP session started, hopefully we have cross-network communication working.

1. From PC1, trace the path to any device on the Green subnet

```
PC1> trace 20.0.0.3 -P 1
trace to 20.0.0.3, 8 hops max (ICMP), press Ctrl+C to stop
 1   10.0.0.1   0.635 ms  0.442 ms  0.387 ms
 2   10.0.5.2   1.527 ms  1.077 ms  0.768 ms
 3   172.155.10.2   2.428 ms  0.888 ms  0.778 ms
 4   20.0.6.1   2.606 ms  1.440 ms  2.038 ms
 5   20.0.0.3   2.746 ms  1.404 ms  1.751 ms

PC1>
```

Figure 13 – PC1 tracing path to PC4

2. Cut some links PC1 used to test the integrity of both networks

```
PC1> trace 20.0.0.3 -P 1
trace to 20.0.0.3, 8 hops max (ICMP), press Ctrl+C to stop
 1   10.0.0.1   0.519 ms  1.128 ms  0.305 ms
 2   10.0.4.2   0.998 ms  0.872 ms  0.615 ms
 3   10.0.6.2   2.081 ms  1.977 ms  0.927 ms
 4   172.155.10.2   3.194 ms  1.616 ms  1.471 ms
 5   20.0.7.1   4.157 ms  2.608 ms  3.211 ms
 6   20.0.5.1   5.536 ms  2.607 ms  2.840 ms
 7   20.0.0.3   4.890 ms  2.711 ms  2.346 ms

PC1>
```

Figure 14 – PC1 re-tracing path to PC4

Congratulations! You made a small enterprise network that can dynamically update using various routing protocols. You may be asking yourself, whats the point of BGP? Couldn't we have simply used OSPF to create the same results? Look at the routing table on AS1-R1:

```
[admin@AS1-R1] > ip route print
Flags: D - DYNAMIC; I - INACTIVE, A - ACTIVE; c - CO
 H - HW-OFFLOADED
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#      DST-ADDRESS       GATEWAY          DISTANCE
0  As  0.0.0.0/0         10.0.4.2                2
1  IsH 0.0.0.0/0         10.0.5.2                1
  DAc  10.0.0.0/23       ether1                  0
  DAo  10.0.2.0/24       10.0.4.2%ether2       110
  DAc  10.0.4.0/30       ether2                  0
  DIcH 10.0.5.0/30       ether3                  0
  DAo  10.0.6.0/30       10.0.4.2%ether2       110
  DAo  10.255.255.0/32   10.0.4.2%ether2       110
  DAc  10.255.255.1/32   loopback                0
  DAo  10.255.255.2/32   10.0.4.2%ether2       110
[admin@AS1-R1] >
```

Figure 15 – AS1-R1 routing table

Now compare this with the routing table on AS1-ASBR:

```
[admin@AS1-ASBR] > ip route print
Flags: D - DYNAMIC; I - INACTIVE, A - ACTIVE; c - C
- HW-OFFLOADED
Columns: DST-ADDRESS, GATEWAY, DISTANCE
     DST-ADDRESS       GATEWAY          DISTANCE
DAo  10.0.0.0/23       10.0.6.1%ether3       110
DAo  10.0.2.0/24       10.0.6.1%ether3       110
DAo  10.0.4.0/30       10.0.6.1%ether3       110
DIcH 10.0.5.0/30       ether2                  0
DAc  10.0.6.0/30       ether3                  0
DAc  10.255.255.0/32   loopback                0
DAo  10.255.255.1/32   10.0.6.1%ether3       110
DAo  10.255.255.2/32   10.0.6.1%ether3       110
DAb  20.0.0.0/22       172.155.10.2           20
DAb  20.0.4.0/25       172.155.10.2           20
DAb  20.0.5.0/30       172.155.10.2           20
DAb  20.0.7.0/30       172.155.10.2           20
DAb  20.255.255.0/32   172.155.10.2           20
DAb  20.255.255.1/32   172.155.10.2           20
DAb  20.255.255.2/32   172.155.10.2           20
D b  172.155.10.0/30   172.155.10.2           20
DAc  172.155.10.0/30   ether1                  0
[admin@AS1-ASBR] >
```

Figure 16 – AS1-ASBR routing table

Notice how AS1-R1 only cares about the routes in its local AS, while the border router takes on the burden of inter-network communication. This is the power of BGP: to offload some networking tasks to dedicated machines, so that other routers can perform more efficiently in their given area.

BGP Troubleshooting

The following router commands are useful in troubleshooting errors you might encounter. Below is the expected output for AS1-ASBR.

1. View established sessions. In this example, there should only be one on both border routers. If there is no output, that means that the neighboring router failed to connect either due to faulty configuration or no configuration at all.

```
> routing bpg session print
```

```
[admin@AS1-ASBR] > routing bgp session print
Flags: E - established
 0 E name="HOST-TO-AS2-1"
     remote.address=172.155.10.2 .as=2 .id=20.255.255.0
     .capabilities=mp,rr,gr,as4 .messages=132 .bytes=2654 .eor=""
     local.address=172.155.10.1 .as=1 .id=10.255.255.0
     .capabilities=mp,rr,gr,as4 .messages=138 .bytes=2902 .eor=""
     output.procid=20
     input.procid=20 ebgp
     hold-time=3m keepalive-time=1m uptime=2h7m450ms
     last-started=2024-05-21 01:00:29 prefix-count=8
[admin@AS1-ASBR] >
```

Figure 17 – BGP session status

2. View routes currently being advertised to peers. Remember, we only want to advertise connected and OSPF routes. If a specific network is missing, verify that the problem router is configured properly. Check interface IP addresses and OSPF configuration.

```
> routing bgp advertisements print
```

```
[admin@AS1-ASBR] > routing bgp advertisements print
 0 peer=HOST-TO-AS2-1 dst=10.255.255.0 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=10.255.255.1 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=10.255.255.2 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=10.0.6.0/30 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=172.155.10.0/30 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=10.0.4.0/30 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=10.0.0.0/23 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

 0 peer=HOST-TO-AS2-1 dst=10.0.2.0/24 afi=ip nexthop=172.155.10.1 origin=0
   as-path=sequence 1

[admin@AS1-ASBR] >
```

Figure 18 – BGP advertised links

3. View all routes currently known by the router.

```
> ip route print
```

```
[admin@AS1-ASBR] > ip route print
Flags: D - DYNAMIC; I - INACTIVE, A - ACTIVE; c -
- HW-OFFLOADED
Columns: DST-ADDRESS, GATEWAY, DISTANCE
     DST-ADDRESS      GATEWAY          DISTANCE
DAo  10.0.0.0/23      10.0.6.1%ether3       110
DAo  10.0.2.0/24      10.0.6.1%ether3       110
DAo  10.0.4.0/30      10.0.6.1%ether3       110
DIcH 10.0.5.0/30      ether2                  0
DAc  10.0.6.0/30      ether3                  0
DAc  10.255.255.0/32  loopback                0
DAo  10.255.255.1/32  10.0.6.1%ether3       110
DAo  10.255.255.2/32  10.0.6.1%ether3       110
DAb  20.0.0.0/22      172.155.10.2            20
DAb  20.0.4.0/25      172.155.10.2            20
DAb  20.0.5.0/30      172.155.10.2            20
DAb  20.0.7.0/30      172.155.10.2            20
DAb  20.255.255.0/32  172.155.10.2            20
DAb  20.255.255.1/32  172.155.10.2            20
DAb  20.255.255.2/32  172.155.10.2            20
D b  172.155.10.0/30  172.155.10.2            20
DAc  172.155.10.0/30  ether1                   0
[admin@AS1-ASBR] >
```

Figure 19 – AS1-ASBR routing table

NOTE: Also be sure to verify that all internal routers are assigned default routes that point to their associated ASBR.

End of Lab

Deliverables

4 Screenshot are needed to earn credit for this exercise:

- Screenshot of GNS3 Workspace with all devices labeled
- Wireshark capture of the TCP and BGP packets being exchanged
- Wireshark view of the 'keep alive' messages
- Wireshark view of the "update" packets with the NLRI view

Homeworks

Assignment 1 – Extend your OSPF network into a BGP Network

- Turn your previous OSPF Lab (assignment 4) into an Autonomous System (AS-1)
- Create two simulated LANs using VPCs
- Create AS-2 and add the two simulated LANs
- Connect AS-1 and AS-2 using two border routers
- RECOMMENDED GRADING CRITERIA
    - Screenshot of GNS3 Workspace with all devices labeled
    - Wireshark capture of the TCP and BGP packets being exchanged
    - Wireshark view of the 'keep alive' messages
    - Wireshark view of the "update" packets with the NLRI view

Screenshots for Printed Copies

CHAPTER 30

IPv6 Addressing - Introduction

SAWYER HANSEN; DANTE ROCCA; AND MATHEW J. HEATH VAN HORN, PHD

A standard IPv4 address is comprised of 32 bits of information, resulting in 4,294,967,296 possible permutations. That's a lot of unique identifiers!... until you realize that a significant number of these IP's are reserved for special purposes and the estimated number of connected internet devices today ranges in the magnitude of tens of billions. If it were not for technologies such as NAT, our available address pool would have been depleted many years ago. However, the increasing number of internet devices has yet to show signs of slowing down any time soon, and we may reach a point where IP supply cannot keep up with demand. Fortunately, researchers from the Internet Engineering Task Force developed IPv6, the sixth version of the Internet Protocol.

Estimated time for completion: 25 minutes

LEARNING OBJECTIVES

- Understand the properties of an IPv6 address
- How to create an IPv6 Host ID from a MAC address
- How to create IPv6 address from IPv4 addresses

PREREQUISITES

- IPv4 Addressing – a Vary Brief Review
- A Ubuntu Desktop VM
- Introduction to Routers

DELIVERABLES

- Worksheet

RESOURCES

- IPv6 Compression Tool – https://findipv6.com/ipv6-compress
- IPv6 Calculator – https://www.calculator.net/ip-subnet-calculator.html
- IPv6 Address Generator – https://www.ipvoid.com/random-ipv6/

CONTRIBUTORS AND TESTERS

- Berkley Rocca, 11th-Grader, Grand Rapids Christian High School
- Jacob M. Christensen, Cybersecurity Student, ERAU-Prescott

Phase I – A very brief review

This instructional material is not designed to replace people's favorite learning materials. We want to simply augment what already exists. However, it was pointed out by some of our testers that a very abbreviated review would be a helpful inclusion within the textbook.

Generally, when you ask someone what a device's IP address is, they provide you with an IPv4 address. That's fine and dandy, but there's more to that picture. Devices also have an IPv6 address. Like IPv4 addresses, these addresses represent the routing prefix and the host identifier. However, IPv6 addresses are structured differently than IPv4 addresses.

Learning IPv4 required knowledge of binary and decimal. IPv6 requires knowledge of hexadecimal. IPv6 addresses use hexadecimal values, or base-16 values, meaning there are 16 possible values in each digit, 0-9, a-f.

Here's a translation table:

Decimal	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hex	0	1	2	3	4	5	6	7
Decimal	8	9	10	11	12	13	14	15
Binary	1000	1001	1010	1011	1100	1101	1110	1111
Hex	8	9	a	b	c	d	e	f

As you can see, using the hexadecimal term 'f' is a much more abbreviated representative symbol of the decimal number '15'. At least much easier on us humans than the binary term '1111'. This is helpful as the binary digits grow in size. Look at the difference between an IPv4 address in decimal vs binary.

IPv4 address represented by decimal	192.168.1.1
IPv4 address represented by binary	11000000.10101000.00000001.00000001

You can count the 1s and 0s if you want, but trust us when we say there are 32 bits there. To prevent us from running out of IP space again, IPv6 uses 128 bits! To put it in perspective, there are 100 times more usable IPv6 addresses than the number of atoms on the surface of the Earth. If we look at the different representations of an IPv6 address, we get the following:

IPv6 Decimal	43962 : 40734 : 51742 : 51400 : 33428 : 48204 : 11497 : 4970
IPv6 Binary	1010101110111010 : 1001111100011110 : 1100101000011110 : 1100100011001000 : 1000001010010100 : 1011110001001100 : 0010110011101001 : 0001001101101010
IPv6 Hexidecimal	abba:9f1e:ca1e:c8c8:8294:bc4c:2ce9:136a

As you can see, hexadecimal notation is MUCH easier for us humans to handle than binary. However, IPv6

addresses can still be pretty long. The average person can only recall lists of 7 items. Thankfully, there are conventions to shorten them such as omitting any redundant zeros.

Let's look at a sample IPv6 address such as 2001:ef48:64a3:0000:0000:0000:32ad:0792

1. Let's look at the leading zeros. Look at the last octet (last 4 characters) in our example address. Like how we don't write the number 8 as 008, we don't want to write 792 as 0792, so we remove the leading zero.

Original	2001:ef48:64a3:0000:0000:0000:32ad:0792
Removed the leading zeros	2001:ef48:64a3:0000:0000:0000:32ad:792

2. Next, we remove octets with 0000 as a value. We will refer to them as "gaps". The convention for removing strings of gaps is to replace them with two colons (::).

Removed the leading Zeros	2001:ef48:64a3:0000:0000:0000:32ad:792
Removed 'gaps"	2001:ef48:64a3::32ad:792

3. Much better, right? Keep in mind that you can only remove one string of gaps. In other words, you cannot have two instances of double colons in your address. If there are multiple gaps not in a string, the left-most gap is reduced to double colons, and the remainder is reduced to a single zero. Except for in the case of double colons, there must always be at least one character per octet. For example:

Original, with two sets of 'gaps'	2001:ef48:64a3:0000:0000:32ad:0000:0792
Removing the 'gaps'	**2001:ef48:64a3::32ad:0:0792**

4. You can still remove the leading zeros if there are any. We only have 1 octet with a leading 0 now. The octet 0792. So seeing the process in its entirety would look like this:

Original, with two sets of 'gaps'	2001:ef48:64a3:**0000:0000**:32ad:**0000**:0792
Remove the 'gaps' – left most string of zeros gets the :: and any other sting of zeros gets a single 0	2001:ef48:64a3**::**32ad:**0**:0792
Remove any remaining leading 0s	2001:ef48:64a3::32ad:0:792

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-11

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-10

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-9

IPv6 is very similar to IPv4 in its use. Recall that IPv4 addresses are broken up into a street and a house number.

The street (routing prefix) is dictated by the subnet mask. If you have a mask of 255.255.255.0, or x.x.x.x/24 in CIDR, the first 24 bits of your address determine your routing prefix. In IPv6, it works similarly. Like IPv4, the routing prefix is determined by the mask. In IPv6, your subnet mask can be upwards of 64 bits. For instance, if our IPv6 address is 2001:ef48:64a3:0000:0000:0000:32ad:0792, then it would be broken into two parts:

Routing Prefix	Host Identifier
2001:ef48:64a3:0000:	0000:0000:32ad:0792

Returning to the street address analogy, think of the routing prefix as a street name, and the host identifiers are the house numbers on that street. Just like how you would visit house 1234 on Elm Street, you would visit host 0000:0000:32ad:0792 at prefix 2001:ef48:64a3:0000

Now, with IPv6 we can be even more specific. Say you have a netmask of /48. Now only the first 48 bits represent the routing prefix. However, the 16 bits that were previously in the prefix are not allocated to the host identifier. These bits now identify the subnet your device is on.

Using the street address analogy, imagine the same road again. However, now imagine Elm Street has several alleys where people built houses after the neighborhood was constructed. If you want to find a house, in one of the alleys, you must first go down Elm Street and then the correct alley to get to the house. In order to get to house 1234, I must take Elm Street to Roadrunner Alley, then continue until I find house 1234. Returning to IPv6 land, if I want to find host 0000:0000:32ad:0792, I must first look on subnet 0000 on prefix 2001:ef48:64a3.

We want to go to the house 2001:ef48:64a3:0000:0000:0000:32ad:0792/48							
2001	ef48	64a3	0000	0000	0000	32ad	0792
Street			Alley	House number			
Routing Prefix			Subnet	Host ID			

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-12

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-13

Now that we have covered routing prefixes and subnets, let's look at the Host ID portion of the IPv6 address called the Extended Unique Identifier (EUI). The EUI consists of the MAC address of the interface. Recall from IPv4 that a MAC address is 48-bits long and consists of two parts, the manufacturer's ID and the serial number. However, the IPv6 Host ID (EIU) is 64-bits long, so some conversion is necessary.

Uppercase or Lowercase when writing Hex digits

3c:27:be:56:1d:d0 vs. 3C:27:BE:56:1D:D0

- Officially
  - In mathematics, any hexadecimal representation is done in lowercase.
  - RFC 5952 says lowercase for cyber functions.

- Real-world
  - It really depends on the interface being used. Sometimes it is easier to read the address if it uses all uppercase, other times, lowercase is easier to read.
  - It doesn't matter. We have more important things to care about.

1. We split the MAC address into its constituent parts.

Given a MAC Address: 3c:27:be:56:1d:d0	
3c:27:be	56:1d:d0
Manufacture's ID (OUI – Organizationally Unique ID)	Serial Number

2. Then we insert the MAC into the mold of an IPv6 host ID and add the reserved bits of FFFE to indicate an EUI-64 generated IPv6 address.

Given a MAC Address: 3c:27:be:56:1d:d0							
3c	27	be	add FF FE		56	1d	d0
3c	27	be	ff	fe	56	1d	d0

3. Now we have to look at the first octet and change the universal/local bit. The bit is 7th from the left. The bit should be a 1, which indicates local, not a 0 which means universal.

Given a MAC Address: 3c:27:be:56:1d:d0							
OUI			IPv6 Reserve		Serial Number		
3c	27	be	ff	fe	56	1d	d0
0011 1100	Convert to binary						
0011 11**0**0	Locate the universal/local (U/L) bit and check its setting. This is set to universal (0)						
0011 11**1**0	We flip this bit to local (1)						
3e	Convert the binary back to Hexidecmal which has now changed from 3c to 3e						
3e	27	be	ff	fe	56	1d	d0
Resulting IPv6 Host ID	3e27:beff:fe56:1dd0						

4. Practice your IPv6 knowledge with these questions:

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-14

An interactive H5P element has been excluded from this version of the text. You can view it online here:
https://eaglepubs.erau.edu/mastering-enterprise-networks-labs/?p=868#h5p-15

Now we look at how an IPv4 address is converted to an IPv6 address. This occurs when a network is using IPv4, but the packets need to tunnel through an IPv6 network to another IPv4 network. This is called 6to4 notation. The 6to4 tunnel concatenates the IPv4 address to the IPv6 address 2002::/16

Let's use the following IPv4 address as our example: 192.168.50.14

Start with IPv4 address	192	168	50	14
Convert to binary	1100 0000	1010 1000	0011 0010	0000 1110
Convert to Hex	c0	a8	32	0e
CAT 2002::/16	**2002**	c0 a8	32 0e	**::1/64**
Result an IPv6 address	2002:c0a8:320e::1/64			

Phase II – IPv6 MiniLab

Here we'll showcase IPv6 in action. It's important to remember that IPv6 functions very similarly to IPv4 in practice.

1. Open GNS3 and create a new project. Title the project appropriately

2. Add a switch and two VPCs to the network

3. Connect the VPCs to the switch

4. Select a routing prefix through random generation. In our example we'll be using 2001:db8::/32

5. Select a host ID for each VPC. In our example the host IDs are ::2 and ::3

6. Open the console for the first VPC and assign it the IPv6 address. Note that you can use the abbreviated address in the VPC

```
ip 2001:db8::2/32
```

7. Now open a Wireshark capture on either of the connections. Then, use the ping command to ping the other host on the network

```
ping 2001:db8::3/32
```

8. On Wireshark you should see that this operates exactly like an IPv4 address in practice. The only difference is the elimination of Network Address Translation and DHCP since we have enough IP

addresses for every device to have one

Phase III – Prefix Designation

Similar to DHCP, IPv6 has a method for assigning IP addresses to hosts. The method for determining the host ID was explained above leaving the network prefix to be determined. A device determines this prefix by soliciting a router to assign it a prefix.

1. On the same GNS3 project add a MikroTik router to the workspace and connect it to the switch
2. Start the MikroTik router and open the console
3. Use the following command to assign the router an IPv6 address

```
ipv6 address add address=2001:db8::1/32 interface=ether1
```

4. Attach a Ubuntu Desktop machine to the switch but do not start it yet
5. Open a Wireshark capture on the link between the router and the switch
6. Start the Ubuntu Desktop machine and open a terminal. Utilize the command

```
ip add
```

7. In Wireshark you should see a router solicitation and a router reply. In the terminal the machine should now have an IPv6 address assigned with the prefix 2001:db8

End of Lab

Deliverables

Complete this worksheet and turn it in to receive credit for this exercise: Worksheet

Homework

Assignment 1 – Create your own GNS3 IPv6 network

- Use a different IPv6 routing prefix

- Connect it to the router from the minilab and use it for prefix designation
- Make sure the new network can ping the old one

Suggested Grading Criteria:

- Screenshot of GNS3 network
- Screenshot of pinging the old network form the new network

PART III

DEFENDING AN ENTERPRISE NETWORK

The chapters in this part are focused on hardening an Enterprise Network in various ways. It is assumed that the learner has already worked through the chapters in Building an Enterprise Network. Learners will need that baseline of knowledge in order to complete these labs.

CHAPTER 31

Network Hardening - pfSense Intranet

MATHEW J. HEATH VAN HORN, PHD AND JACOB CHRISTENSEN

This chapter walks the learner through the steps needed to add a pfSense server to an enterprise network. Specifically, we are going to set up the lab, configure the pfSense server to act as our DHCP server, open all the firewall ports so you can see what is going on in the network, and then watch how the addition of each firewall rule affects our enterprise network.

LEARNING OBJECTIVES

- Configure pfSense for first use in GNS3
- Create various firewall rules to regulate IPv4 traffic
- Use pfSense as the DHCP server
- Use Wireshark to observe the effects of firewall rules

PREREQUISITES

- Chapter 6 – Adding a Virtual Machine to GNS3
- Chapter 10 – Create a pfSense Server
- Chapter 11 – Create a Ubuntu Desktop

DELIVERABLES

- Screenshot of GNS3 Working environment once everything works
- Screenshot of the pfSense Dashboard
- Screenshot of all inside devices being able to ping
- Screenshot of the 3 rules for the DMZ

RESOURCES

- We consolidated information from a wide variety of resources. However, three sources stand out as being particularly helpful to this lab and we want to recognize them here:

- Saifudeen Sidheeq – "How to Configure PfSense DMZ Setup? | Step by Step" – https://getlabsdone.com/how-to-configure-pfsense-dmz-setup/
- Frank at WunderTech – "How to Set Up a DMZ in pfSense" – https://www.wundertech.net/how-to-set-up-a-dmz-in-pfsense/
- Nikhath K – "pFSense DMZ Setup Guide" – https://bobcares.com/blog/pfsense-dmz-setup/

CONTRIBUTORS AND TESTERS

- Julian H. Romano, Cybersecurity Student, ERAU-Prescott
- Dante Rocca, Cybersecurity Student, ERAU-Prescott
- Jungsoo Noh, Cybersecurity Student, ERAU-Prescott

Phase I – Setting up the Lab

The following steps are to create a baseline environment for completing the lab. It makes assumptions about learner knowledge from completing previous labs.

The completed network topology will look like this:

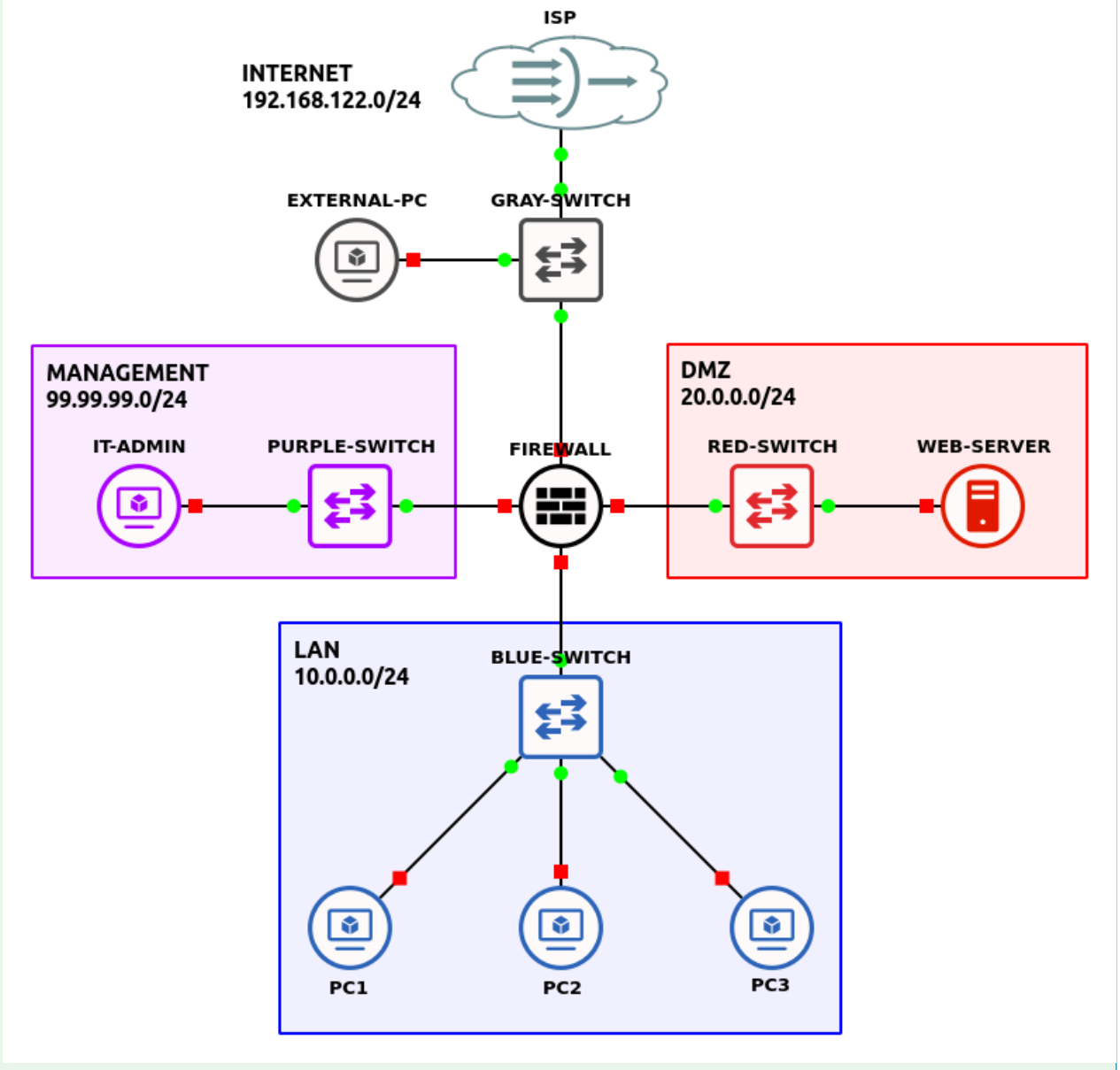


Figure 1 – Final network

1. In VirtualBox, create clones of your pfSense firewall and your Ubuntu Linux Desktop

> **NOTE:** When importing new VMs into GNS3, ensure that ***Allow GNS3 to use any configured VirtualBox adapter*** is selected in their network settings!

2. Start GNS3

2.1. Import the devices

2.2. On the pfSense Server, change the network settings to accommodate 4 adapters (Figure 2)

2.3. Create a new project: **LAB_16**

3. Build the following network:

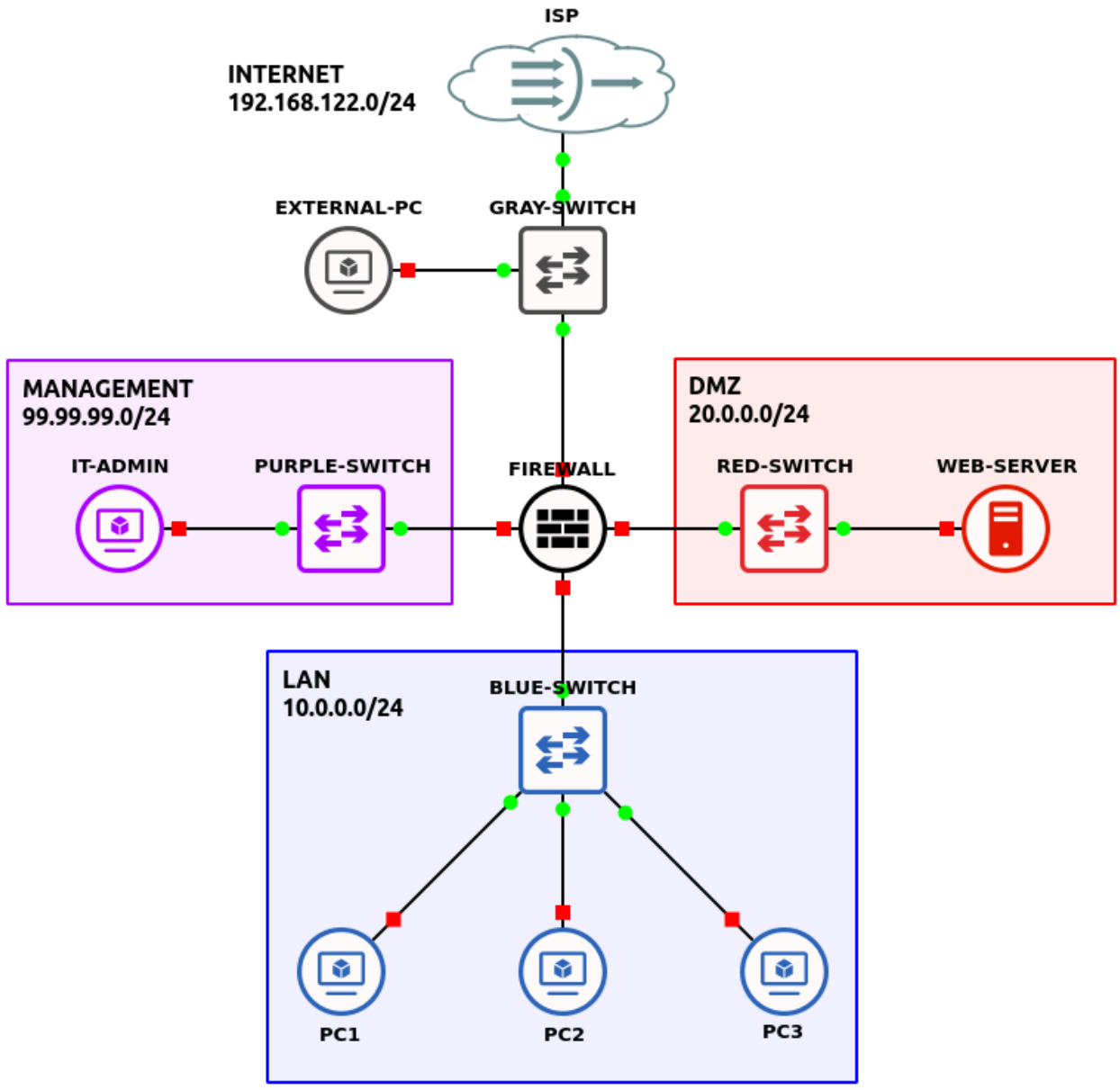


Figure 3 – GNS3 workspace

3.1. "Internet": (NO BOX) – **192.168.122.0/24**

3.1.1. One switch – *Ethernet switch*

3.1.2. Internet connectivity with host machine – *NAT Cloud (ISP)*

3.1.3. One client machine – *Guest/VM with browser (external pc)*

> **NOTE:** While this example uses GNS3's Chromium appliance, any device that has a browser installed will suffice.

3.2. Management: (PURPLE BOX) – **99.99.99.0/24**

3.2.1. One switch – *Ethernet switch*

3.2.2. One client machine – *Guest/VM with browser (it admin)*

3.3. DMZ: (RED BOX) – **20.0.0.0/24**

3.3.1. One switch – *Ethernet switch*

3.3.2. One client machine – *Ubuntu Server (webserver)*

> **NOTE:** Ensure that the Apache2 package is installed.
>
> ```
> > apt update
> ```
>
> ```
> > apt install apache2
> ```
>
> Since this lab isn't focused on configuring a real webserver, we will simply use the default webpage that Apache provides out of the box. For now, simply verify that the daemon is running:

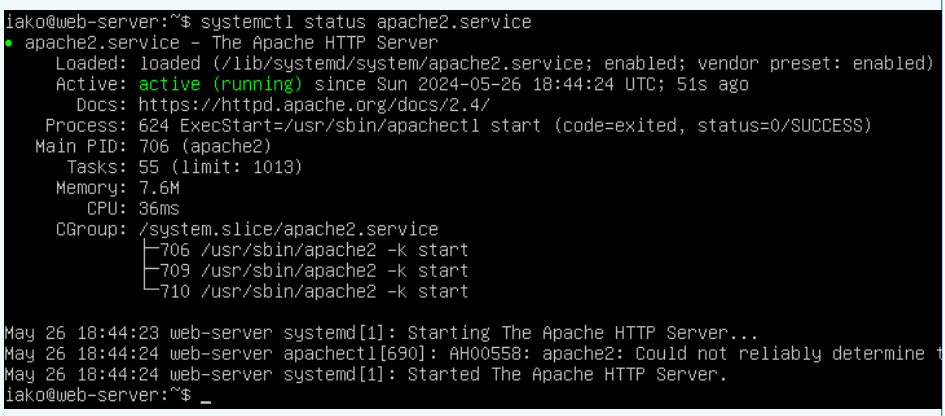


Figure 4 – Apache2 daemon status

If you are having issues with getting Ubuntu or Apache to work, you can "simulate" a webserver with a VPCS client. The idea is to have an end device in the DMZ that we can ping from other areas on the network.

3.4. LAN: (BLUE BOX) – **10.0.0.0/24**

3.4.1. One switch – ***Ethernet switch***

3.4.2. Three client machines – ***VPCS***

3.5. pfSense Firewall connections:

NOTE: This example uses the version 2.7.0 of pfSense Community Edition. You can either host this device as a VirtualBox VM or as a GNS3 appliance.

3.5.1. Connect ethernet0 to the ISP switch (Internet)

3.5.2. Connect ethernet1 to the Management switch

3.5.3. Connect ethernet2 to the DMZ switch

3.5.4. Connect ethernet3 to the LAN switch

Phase II – Configuring pfSense via CLI Console

Using VirtualBox instead of a physical box has its unique challenges. Mostly VirtualBox tries to help us do what

we are trying to do and that can cause us some conflicts. We can work around these issues, but it may stress your cyber knowledge!

1. In GNS3, start the pfSense server

NOTE: There are like 3 seconds where you can change your boot options, but just let the timer click down and let it boot. The first time it starts can take a few minutes.

1.1. Once the VM finishes booting, you should see the CLI menu below

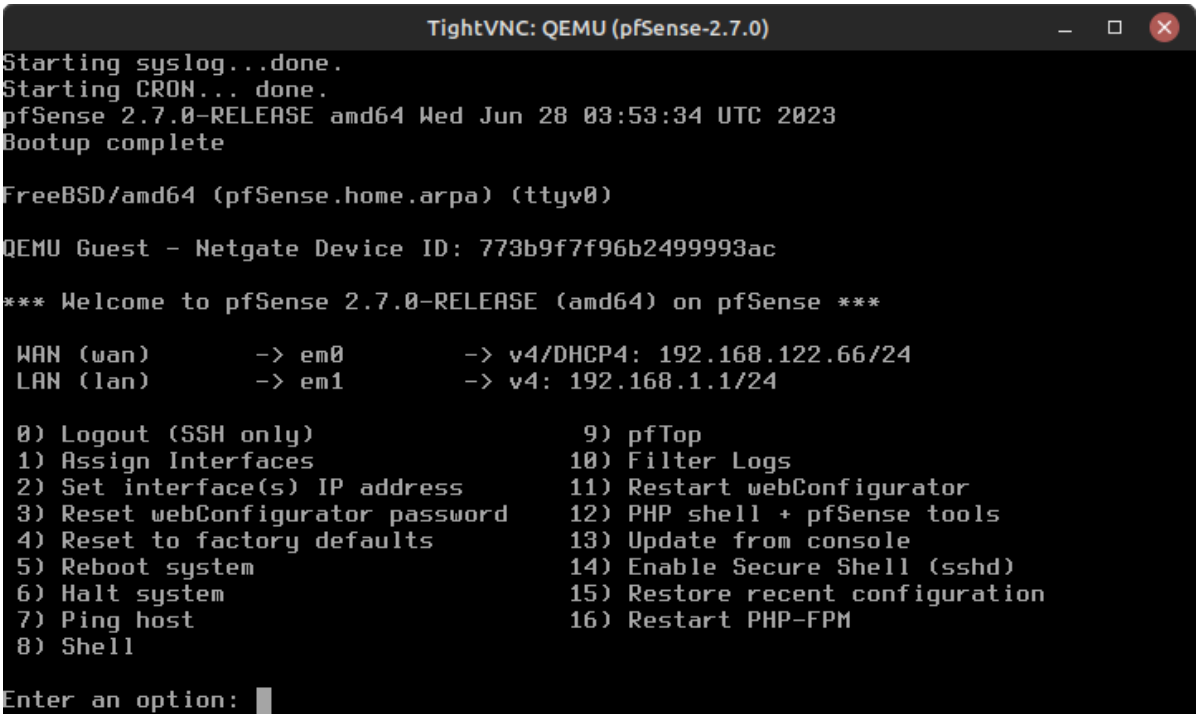


Figure 5 – pfSense command line console

2. As you can see, pfSense only recognizes two interfaces – **em0 (WAN)** and **em1 (LAN)** – as currently active

2.1. Assign each of the pfSense interfaces on this device with a network

> **NOTE:** Use the table below as a configuration guide for this step.

GNS3 Interface	pfSense Interface	pfSense Interface Name
Ethernet0	em0	WAN
Ethernet1	em1	LAN
Ethernet2	em2	OPT1
Ethernet4	em3	OPT2

2.2. Select option 1 to **Assign Interfaces** (Figure 6)

2.2.1. When prompted – **Should VLANs be set up now [y|n]?** – type

```
n
```

2.2.2. **Enter the WAN interface name or 'a' for auto-detection**

```
em0
```

2.2.3. **Enter the LAN interface name…**

```
em1
```

2.2.4. Similarly, use ***em2*** for the Optional 1 (OPT1) and ***em3*** for the Optional 2 (OPT2)

2.2.5. Verify the settings and type ***y*** to proceed (Figure 7)

2.3. You can see that all interfaces are now correctly assigned and active (Figure 8)

3. Configure IP and DHCP settings

3.1. Select option 2 to **Set interface(s) IP address**

3.2. You will see a menu of the 4 interfaces with their current network settings (Figure 8)

> **NOTE:** We will walk you through the first two interfaces, and leave the rest for you to complete on your own. Use the table below to assist with configuration settings.

pfSense Interface	pfSense Interface	IPv4 Address
em0	WAN	Dynamic – DHCP
em1	LAN	Static – 20.2.2.1/24
em2	OPT1	Static – 10.1.1.1/24
em3	OPT2	Static – 212.10.10.1/24

3.3. Select interface 1 – WAN

3.3.1. **Configure IPv4 address WAN interface via DHCP? (y/n)**

```
y
```

3.3.2. **Configure IPv6 address WAN interface via DHCP6? (y/n)**

```
n
```

3.3.3. **Enter the new WAN IPv6 address**
Press *Enter* for none (we are not using IPv6)

3.3.4. **Do you want to revert to HTTP as the webConfigurator protocol? (y/n)**

```
n
```

3.3.5. Press *Enter* to finish em0 configuration and proceed

3.4. Select interface 2 – LAN

3.4.1. **Configure IPv4 address LAN interface via DHCP?**

```
n
```

3.4.2. **Enter the new LAN IPv4 address**

```
99.99.99.1/24
```

3.4.3. **Enter the new LAN IPv4 upstream gateway address**
Press *Enter* for none

3.4.4. **Configure IPv6 address LAN interface via DHCP6? (y/n)**

```
n
```

3.4.5. **Enter the new WAN IPv6 address**
Press *Enter* for none (we are not using IPv6)

3.4.6. **Do you want to enable the DHCP server on LAN? (y/n)**

```
y
```

3.4.7. **Enter the start address of the IPv4 client address range:**

```
99.99.99.5
```

3.4.8. **Enter the end address of the IPv4 client address range:**

```
99.99.99.100
```

3.4.9. **Do you want to revert to HTTP as the webConfigurator protocol?**

```
n
```

3.4.10. You should get a message on how to access the Web Configurator and be instructed to press ***Enter*** to continue (Figure 10)

3.5. Setup em2 (OPT1), and em3 (OPT2) in a similar fashion as em1 using the IP address spaces we picked earlier (Figure 11)

4. In GNS3, start the desktop in the Management LAN and check the current network settings

4.1. Verify that DHCP works and that our management PC has an IP address in the range of **99.99.99.5 – 99.99.99.100**

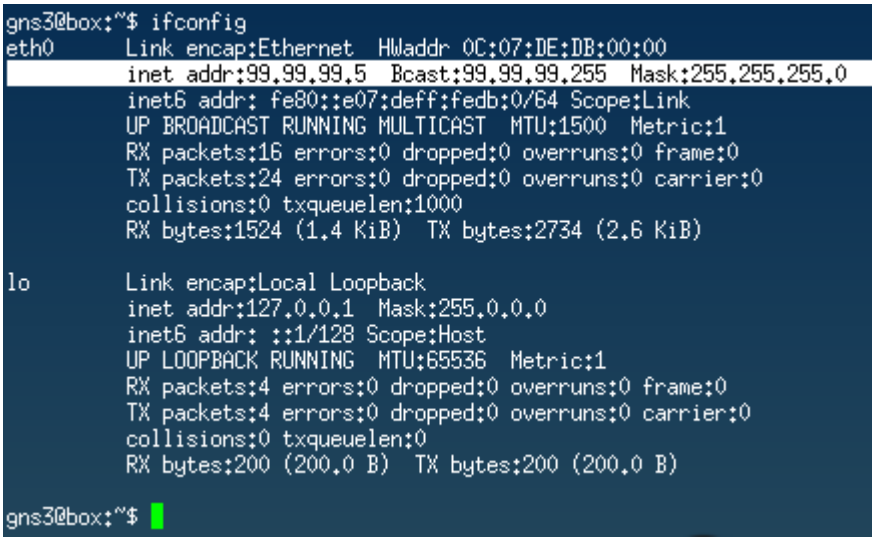


Figure 12 – Client network settings verified

NOTE: You can request a new IP address at any time with the following commands (some Linux distros may vary...)

Release the currently assigned address on the interface named enp0s3.

```
> dhclient -r -i enp0s3 -v
```

Broadcast DHCP Discover packets to assign a new address.

```
> dhclient -i enp0s3 -v
```

4.2. Open Firefox and type the IP address you were given for webConfigurator (it should be *https://99.99.99.1/*)

We haven't set up any certificates yet, so you will get a big warning. Just click ***Advanced...*** and go to the site anyway by accepting the risk and continuing. This will take you to the pfSense GUI interface to Sign In. Previously, when you were asked to revert to HTTP, if you said ***y***, you will not get any warnings. Great choice to avoid a security message, but bad practice because everything you do can be read by others monitoring your network traffic.

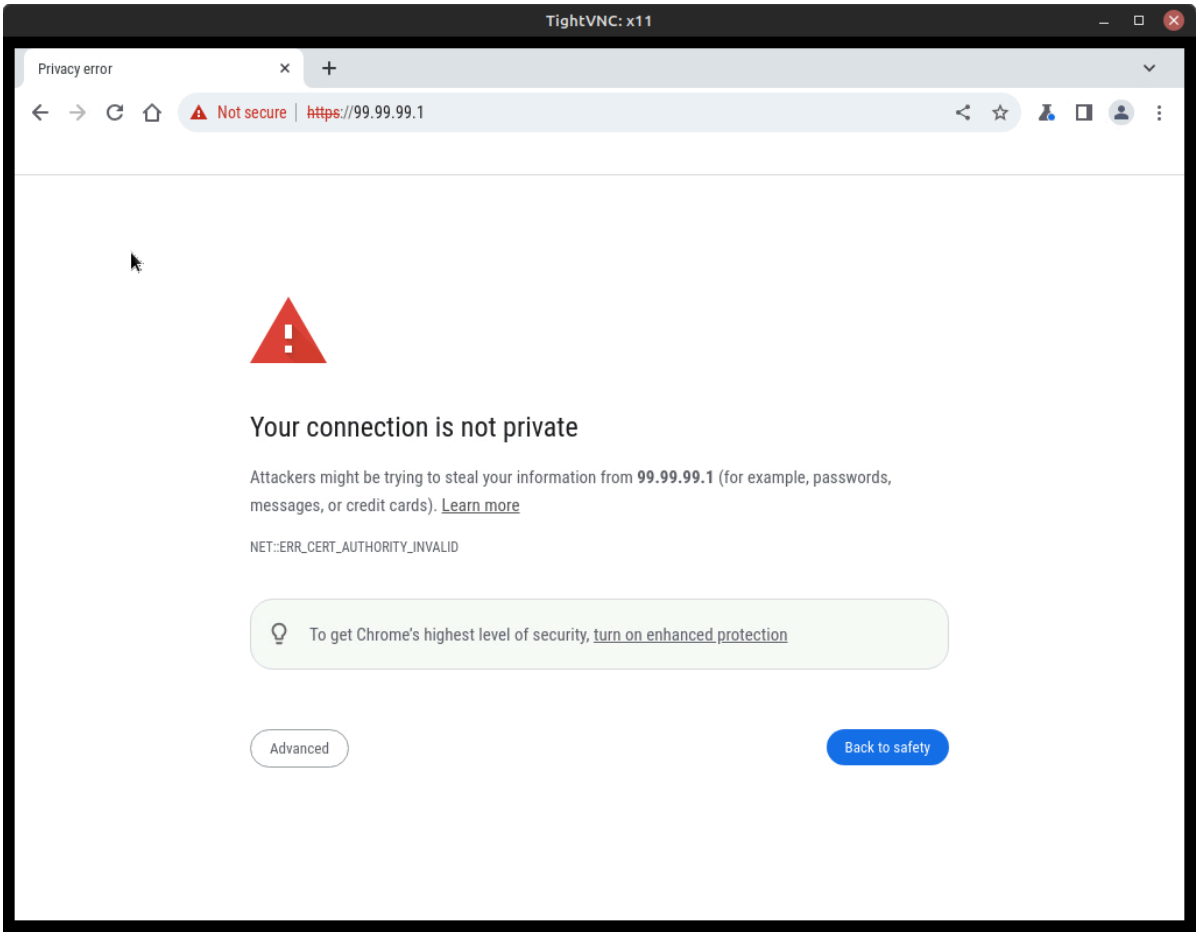


Figure 13 – Caution page connecting to pfSense

5. Return to the GNS3 workspace, start the other devices in the DMZ and LAN spaces, and verify they are receiving IP addresses

> **NOTE:** If a PC doesn't get a DHCP IP, don't worry about it, we'll address it later using the GUI.

Phase III – Configuring pfSense via GUI Console

pfSense is not generally configured using the CLI menu. The GUI interface provides much more options and is easier to work with. At this point, all of your devices should be getting IP addresses from the pfSense DHCP server. If they aren't getting a DHCP IP address, don't worry, we'll check them in the GUI.

1. On the Management PC, return to the login page at ***https://99.99.99.1*** (Figure 14)

    1.1. At the Sign In screen use the default creds to log in:

Username: ***admin***
Password: ***pfsense***

2. Once logged in, on the top ribbon menu select ***Status–>Dashboard*** (Figure 15)

> **NOTE:** At the top, you will see a large warning about using the default username and password. Normally I would say change this, but I've had too many students ask me, "Dr. HVH? What's my password?" so please leave this alone for this exercise. On the right, you can see the interface settings we made earlier. I recommend that you click on these to get an idea of how the GUI and CLI commands line up but do not make any changes.

3. At the top menu bar, select ***Interfaces–>Assignments*** (Figure 16)

3.1. Click on***WAN*** to bring up the configuration settings for just that interface (Figure 17)

3.1.1. Review the various options to get familiar with the available options

3.1.2. Make the following changes as necessary

Option	Value
Description	ISP
IPv4 Configuration Type	DHCP
IPv6 Configuration Type	None

3.1.3. Scroll to the bottom and press ***Save***

3.1.4. Now you will see a double-check prompt, so select ***Apply Changes*** (Figure 18)

3.2. Return to the Interface Assignments page and make the following adjustments to the remain adapters (Figure 19):

3.2.1. LAN – change to "Management" and verify static IP assignment of 99.99.99.1/24

3.2.2. OPT1 – change to "DMZ" and verify static IP assignment of 20.0.0.1/24

3.2.3. OPT2 – change to "LAN" and verify static IP assignment of 10.0.0.1/24

> **NOTE:** Yes, we could have assigned LAN to em3 from the get-go. However, by doing it this way, you gain experience in using the GUI. You will encounter a few of these moments in these labs. The goal is to help you learn, not just read and click.

4. Select***Services –>DHCP Server*** to open the DHCP settings (Figure 20)

    4.1. View the details for the DHCP services on the Management interface

    4.2. Let's pretend that we have reserved the IP addresses 99.99.99.101-99.99.99.110 so we will add a DHCP pool to use the remaining IP address

        4.2.1. Click on **+** ***Add Pool***

        4.2.2. Pool Description –> "Management"

        4.2.3. Range –> 99.99.99.111 to 99.99.99.254

        4.2.4. Click on ***save***

        4.2.5. Your settings should look like (Figure 21)

    4.3. Verify DHCP services for the ISP, DMZ, and LAN networks and change as necessary

> **NOTE:** Remember that all DHCP servers need to provide a gateway address to their clients too.

5. Return to the Dashboard by selecting ***Status –>Dashboard***

Phase IV – Open Everything Up

In a default two-interface LAN and WAN configuration, pfSense software utilizes default deny on the WAN and default allow on the LAN. Everything inbound from the Internet is denied, and everything out to the Internet from the LAN is permitted. All home-grade routers use this methodology, as do all similar open-source projects and most similar commercial offerings. It's what most people expect out of the box, therefore it is the default configuration. That said, while it is a convenient way to start, it is not the recommended means of long-term operation. – (Last accessed 25 October 2023 https://docs.netgate.com/pfsense/en/latest/firewall/best-practices.html)

We are going to violate these settings for demonstration purposes. Don't worry, we'll put them back.

1. Navigate to the pfSense top ribbon and select ***Firewall–>Rules***

2. Click on the ***Management*** tab, observe the firewall rules and read the descriptions (Figure 22)

    2.1. **Anti-Lockout Rule** – keeps users from accidentally locking themselves out of the GUI interface

    2.2. **Default allow LAN to any rule** – Does not restrict any access for IPv4 hosts

2.3. **Default allow LAN IPv6 to any rule** – Does not restrict any access for IPv6 hosts

> **NOTE:** We aren't using IPv6, so you can delete the **Default allow LAN IPV6 to any rule** by pressing the***trashcan button***.

3. Click on the ***DMZ*** tab to open the rules for the DMZ

4. Click on ***Add*** (there are no rules so it doesn't matter which one) to open the Edit Firewall Rule page (Figure 23)

4.1. Make the following changes using the drop-down menus and textboxes:

Option	Value
Action	Pass
Interface	DMZ
Address Family	IPv4
Protocol	Any
Source	DMZ net
Destination	Any
Description	Allow DMZ to any rule

4.2. *Save* and *apply changes*

5. Repeat the above steps for the LAN interface

> **NOTE:** We are going to leave the ISP interface completely blocked for now.

6. Test the firewall settings

6.1. From the Management PC, ping the simulated webserver and any of the LAN PCs

> **NOTE:** If you still can't ping, you have done something wrong and will need to troubleshoot the problem. Did you remember to apply DHCP to all the end devices?

6.2. From the webserver, ping a LAN PC and the Management PC

6.3. From the Management PC, view the webpage hosted on the webserver (**http://20.0.0.5:80**)

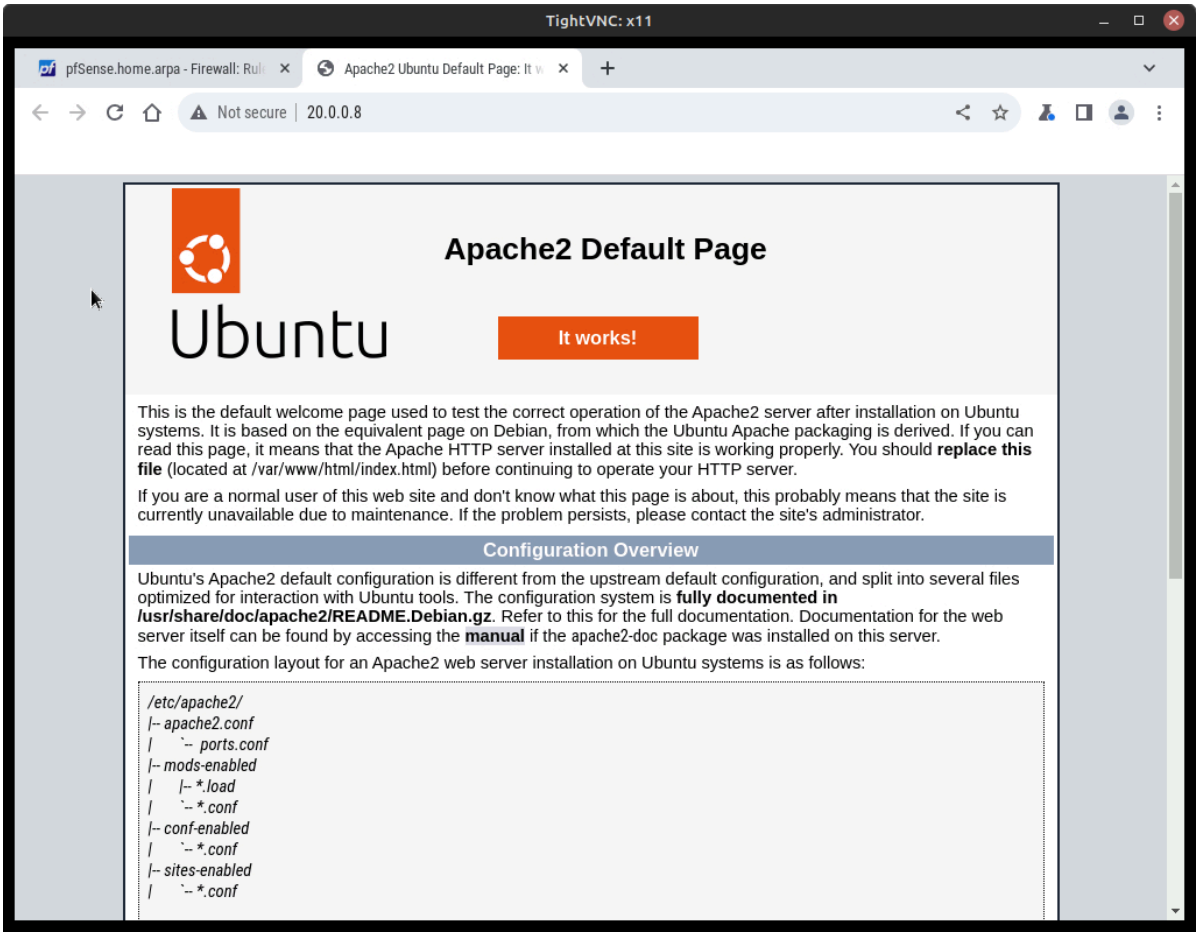


Figure 24 – Default Apache2 website

Phase V – Separate the DMZ from the LAN

The whole point of a DMZ is to have two separate infrastructures that can't interact with each other directly, they have to negotiate data transfer through a 3rd party, in this case, pfSense. So we are going to set up traffic blocking between the DMZ and the LAN.

1. On the pfSense top menu bar, select *Firewall–>Rules–>DMZ*

    1.1. Edit the existing rule with the following changes:

Option	Value
Action	Block
Interface	DMZ
Address Family	IPv4
Protocol	Any
Source	DMZ net
Destination	LAN net
Description	Separating the LAN from the DMZ

1.2. Click on ***Save*** and of course ***Apply Changes***

2. Now navigate to the web server console and try pinging the LAN PC again. You should get a timeout error

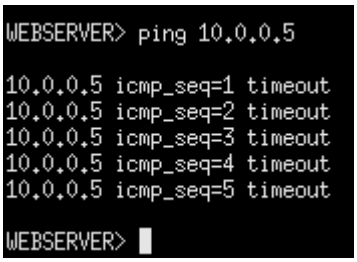


Figure 25 – Webserver pinging PC1

3. ***Right-click*** on the connection on the webserver-pfSense link and start a Wireshark capture

3.1. Try to ping the LAN PC again. Notice that there is no response from the pfSense server now

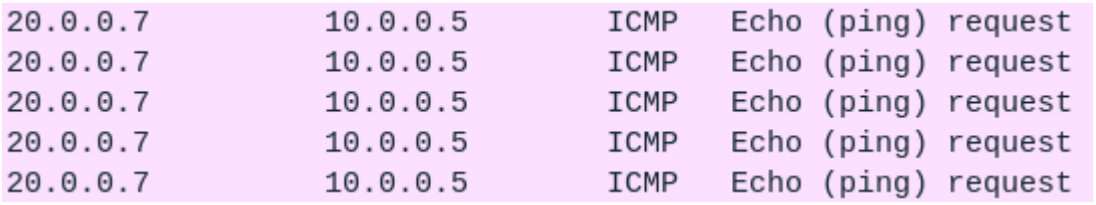


Figure 26 – Failed communication from DMZ to LAN

3.2. Navigate to one of the LAN PCs and try to ping the webserver. It should be successful

```
10.0.0.5        20.0.0.7      ICMP   Echo (ping) request
20.0.0.7        10.0.0.5      ICMP   Echo (ping) reply
10.0.0.5        20.0.0.7      ICMP   Echo (ping) request
20.0.0.7        10.0.0.5      ICMP   Echo (ping) reply
10.0.0.5        20.0.0.7      ICMP   Echo (ping) request
20.0.0.7        10.0.0.5      ICMP   Echo (ping) reply
10.0.0.5        20.0.0.7      ICMP   Echo (ping) request
20.0.0.7        10.0.0.5      ICMP   Echo (ping) reply
10.0.0.5        20.0.0.7      ICMP   Echo (ping) request
20.0.0.7        10.0.0.5      ICMP   Echo (ping) reply
```

Figure 27 – Successful communication from LAN to DMZ

Phase VI – Access the Internet

Our web server needs access to the Internet. So we are going to add a rule to allow this to occur. Remember, we are going to use the ISP_Test_PC as our simulated Internet. We are going to allow only the Web_Server to access the Internet through the Firewall. We are going to introduce using Aliases in pfSense for this phase.

1. Navigate to the Ubuntu_Desktop and use the browser to get access to the pfSense GUI
2. Using the top ribbon menu select *Firewall –>Aliases–>Ports*
   2.1. Select *Add* and adjust the properties of the Alias as follows (Figure 28)

Option	**Value**
Name	Internet_Ports
Description	Access to the Internet
Type	Port(s)
Port(s)	**Description**
443	https
80	http
53	dns

   2.2. *Save* and *Apply Changes*
3. Using the top ribbon menu select *Firewall –>Aliases–>IP*
   3.1. Select *Add* and adjust the properties of the Alias as follows (Figure 29)

Option	Value
Name	DMZ_Internet_Enabled_Hosts
Description	Allows machines in DMZ to access the Internet
Type	Host(s)
Host(s)	**Description**
20.0.0.5-20.0.0.100	DMZ network

3.2. *Save* and *Apply Changes*

4. Using the top ribbon menu select *Firewall–>Rules–>DMZ*

4.1. Select ***Add*** and adjust the properties as follows (Figure 30)

Option	Value
Action	Pass
Interface	DMZ
Address Family	IPv4
Protocol	TCP/UDP
Source	Single host or alias
Source Address	DMZ_Internet_Enabled_Hosts
Destination	Any
Description	Allow DMZ Internet access
Destination Port Range	**Value**
From	(other)
Custom	Internet_Ports
To	(other)
Custom	Internet_Ports

4.2. *Save* and *Apply Changes* (Figure 31)

Phase VII – ICMP

If you were working ahead, you might have noticed that you can't ping from the webserver to the ISP_Test_IP. Remember we opened some ports, but Ping is a function of ICMP that doesn't use ports. We need a separate Alias to allow this.

1. Navigate back to *Firewall –> Rules–>DMZ*

1.1. Add another rule, below the others, with the following settings:

Option	Value
Action	Pass
Interface	DMZ
Address Family	IPv4
Protocol	ICMP
ICMP Subtypes	Any
Source	DMZ net
Destination	Any
Description	Allow for Pings

1.2. *Save* and *Apply Changes*

2. Verify that you now have three firewalls in place for the DMZ network

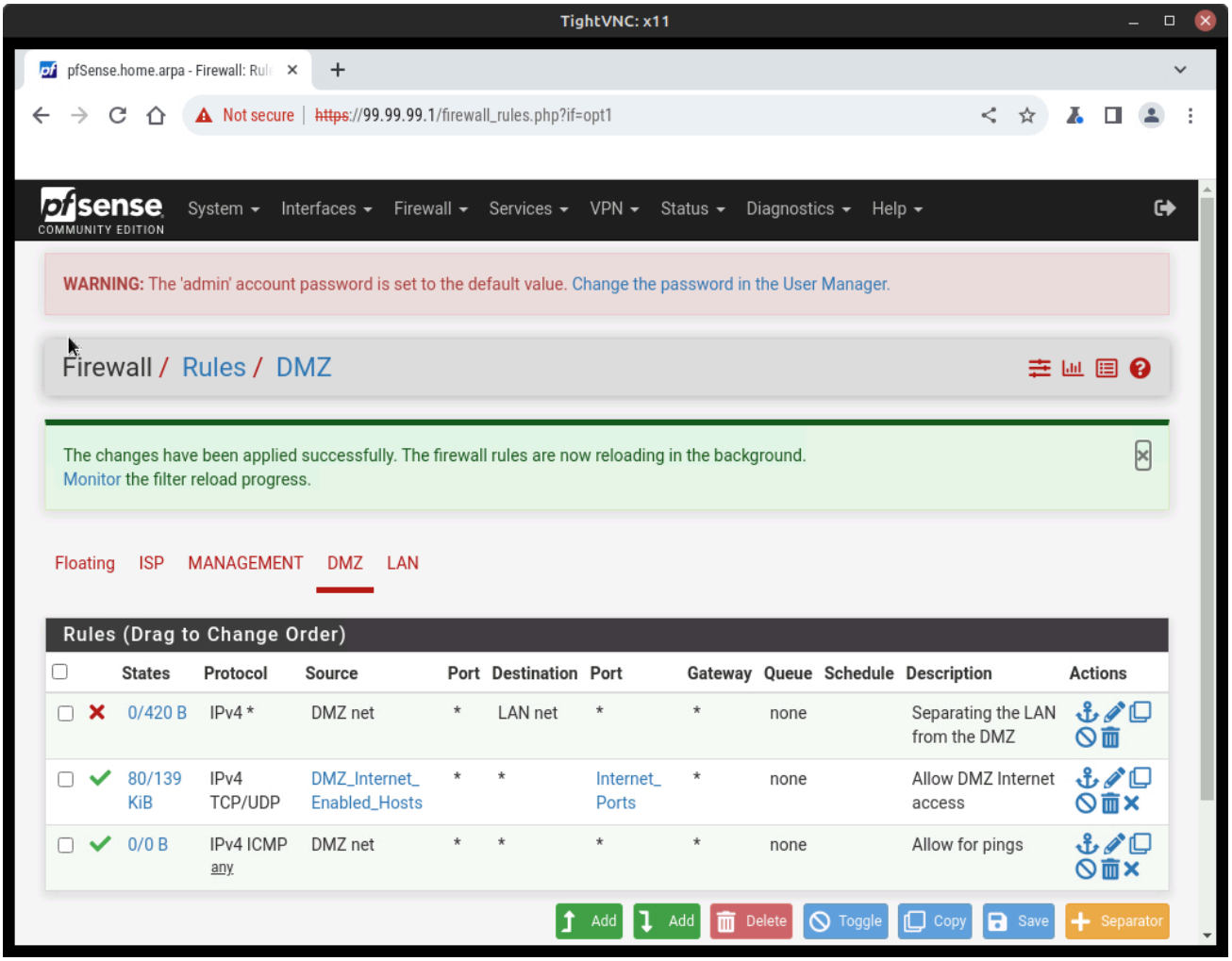


Figure 32 – DMZ firewall rules

3. Test the communication of the network as it currently stands

    3.1. From the Webserver, ping the External PC (it should be successful)

    3.2. From PC1, ping the webserver (it should be successful)

    3.3. From the External PC, ping the Webserver…

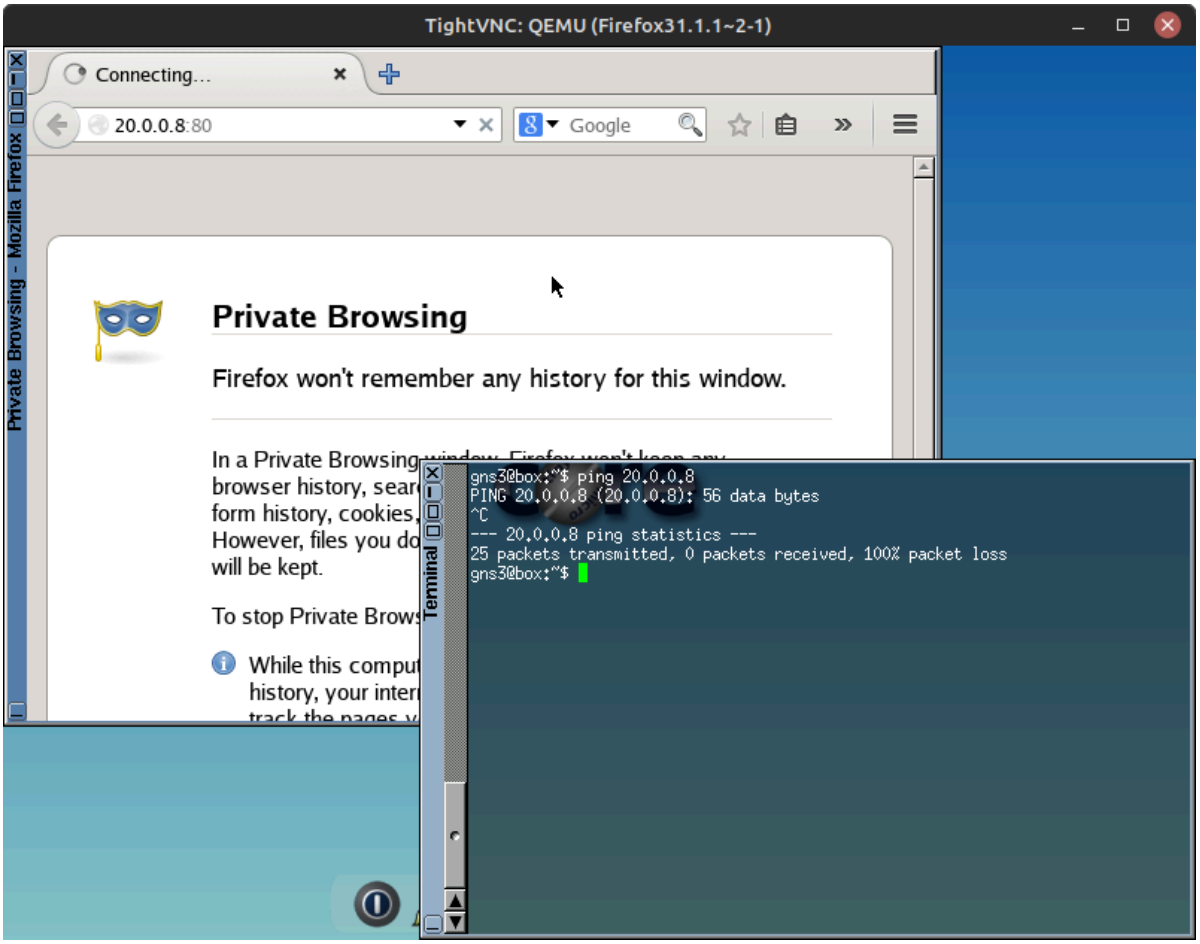


Figure 33 – External PC failed to see webserver

> Failure! Whether we try to ping or view the default Apache webpage, the external PC is unable to communicate with our DMZ. So how are people going to be able to reach our webserver?

End of Lab

Deliverables

Four screenshots are required to receive credit for completing this exercise:

- Screenshot of the GNS3 workspace with all devices placed and labeled (Phase II)
- Screenshot of the pfSense services dashboard after DHCP has been set up (Phase III)
- Screenshot of the web server successfully pinging a LAN PC and the Management PC (Phase IV)
- Screenshot of the 3 rules for the DMZ (Phase VII)

Homeworks

Assignment 1 – Scan the networks using Kali Linux and nmap

- Import a Kali Linux VM into the GNS3 environment. Use the same network settings as the other devices used in this chapter.
- Attach a cable from the Kali machine to a switch and run nmap looking for active IP addresses and open ports. (type man nmap at the command prompt to read instructions about using nmap)
    - Screenshot of ISP switch
    - Screenshot of Management Switch
    - Screenshot of DMZ switch
    - Screenshot of LAN Switch

RECOMMENDED GRADING CRITERIA:

- four screenshots
    - ISP has no open ports
    - Management has open ports
    - DMZ has open ports
    - LAN has open ports

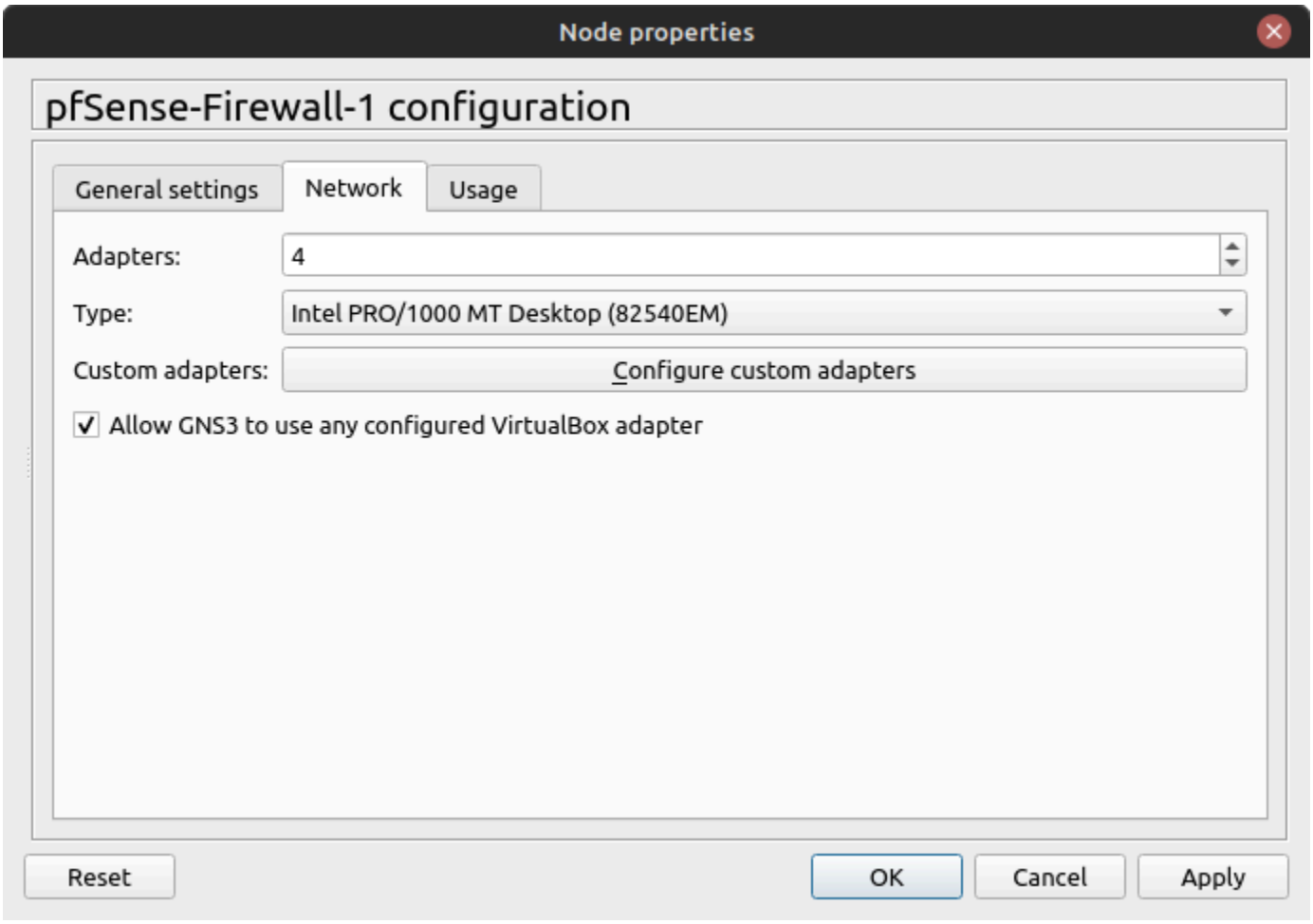


Figure 2 – GNS3 pfSense template configuration

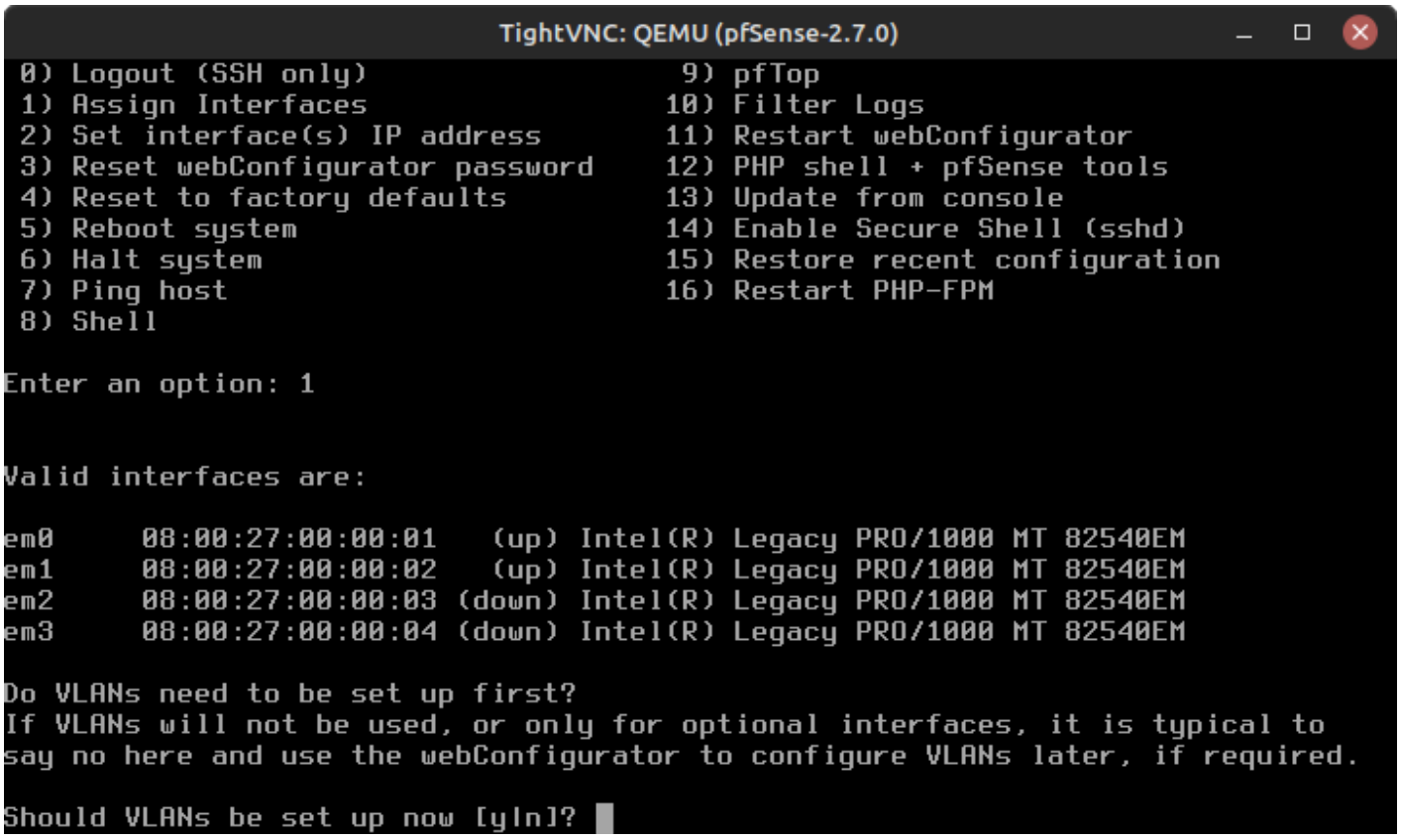


Figure 6 – List of pfSense interfaces ready to be assigned

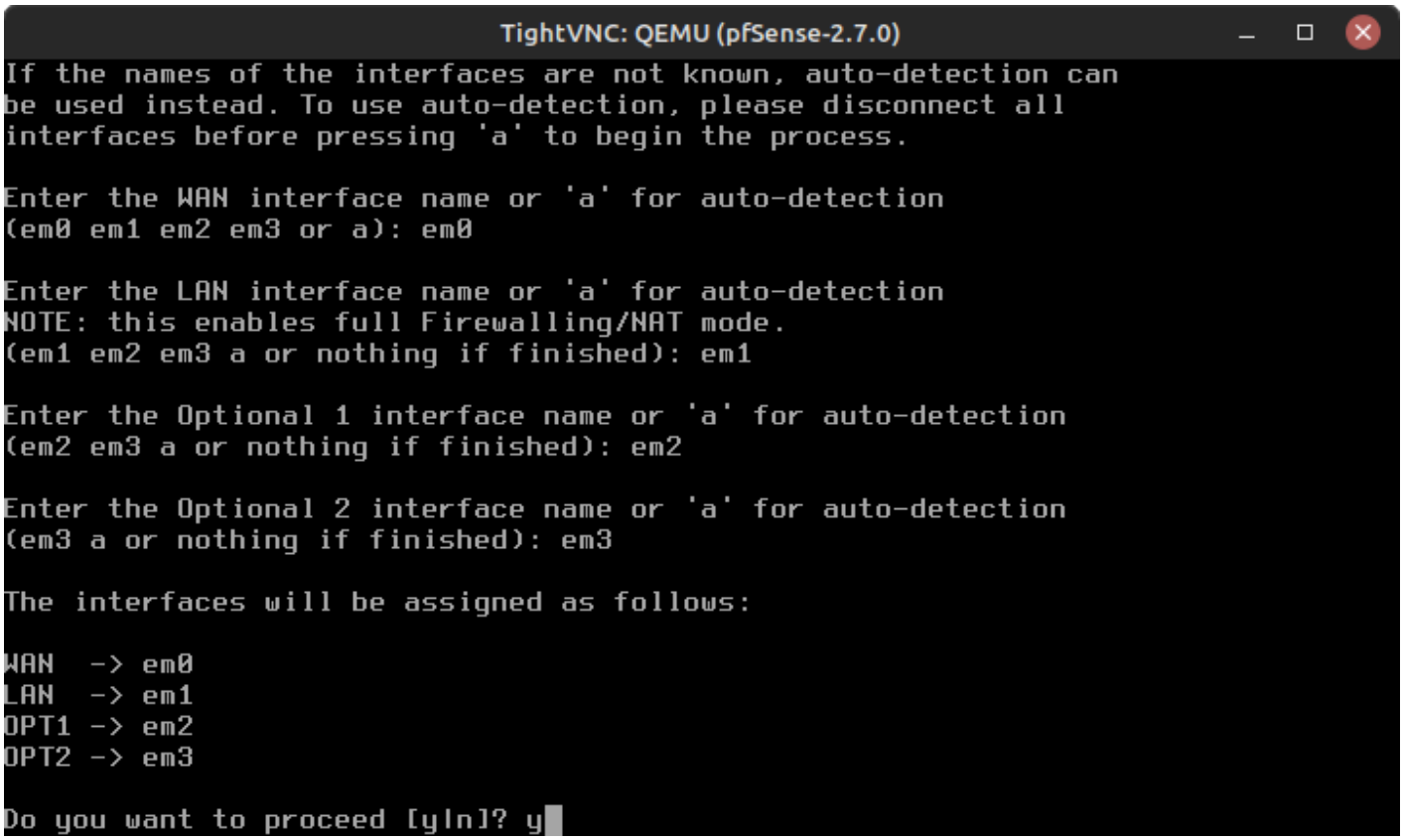


Figure 7 – pfSense interfaces correctly assigned

Figure 8 – Updated pfSense CLI console

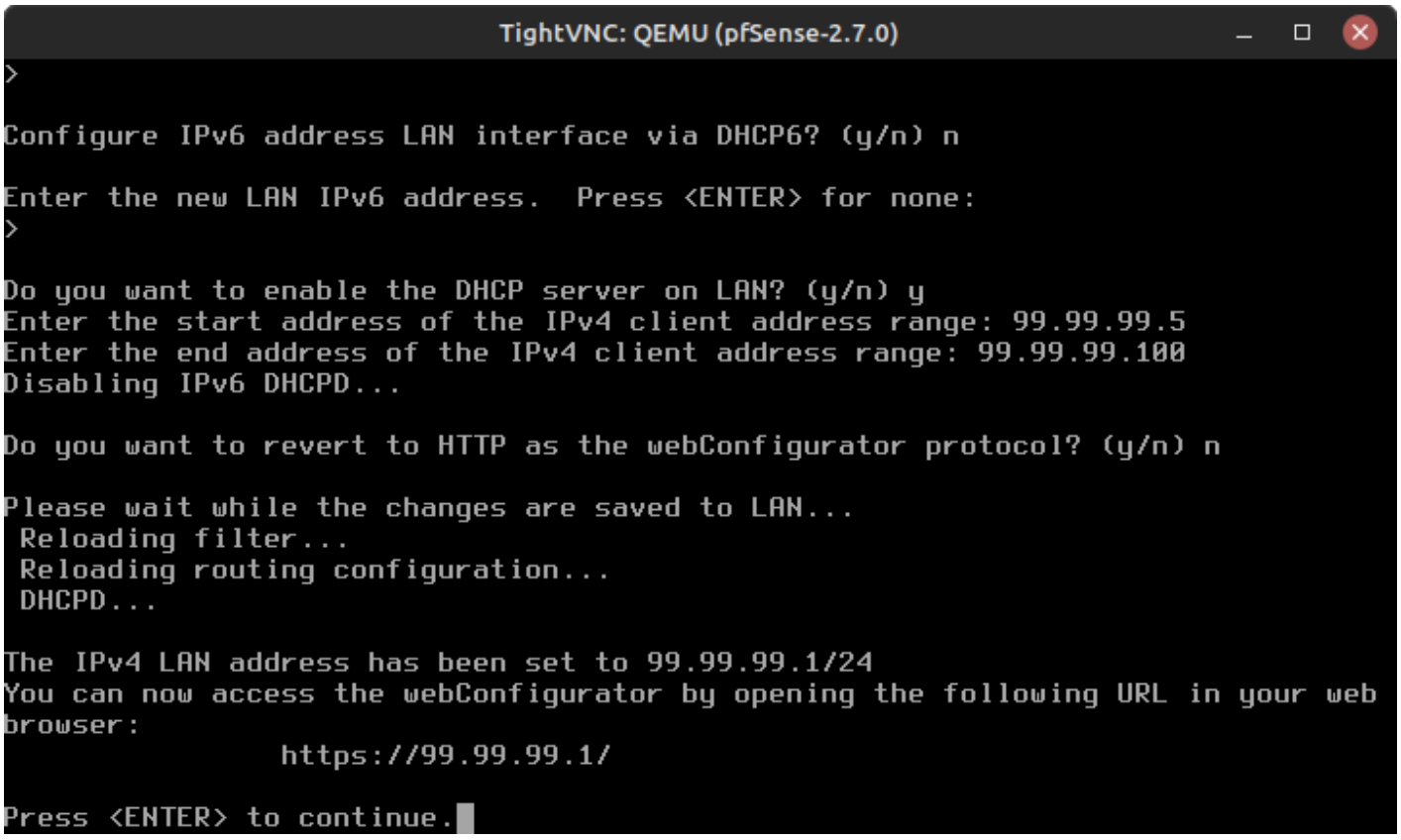


Figure 10 – em1 interface configured

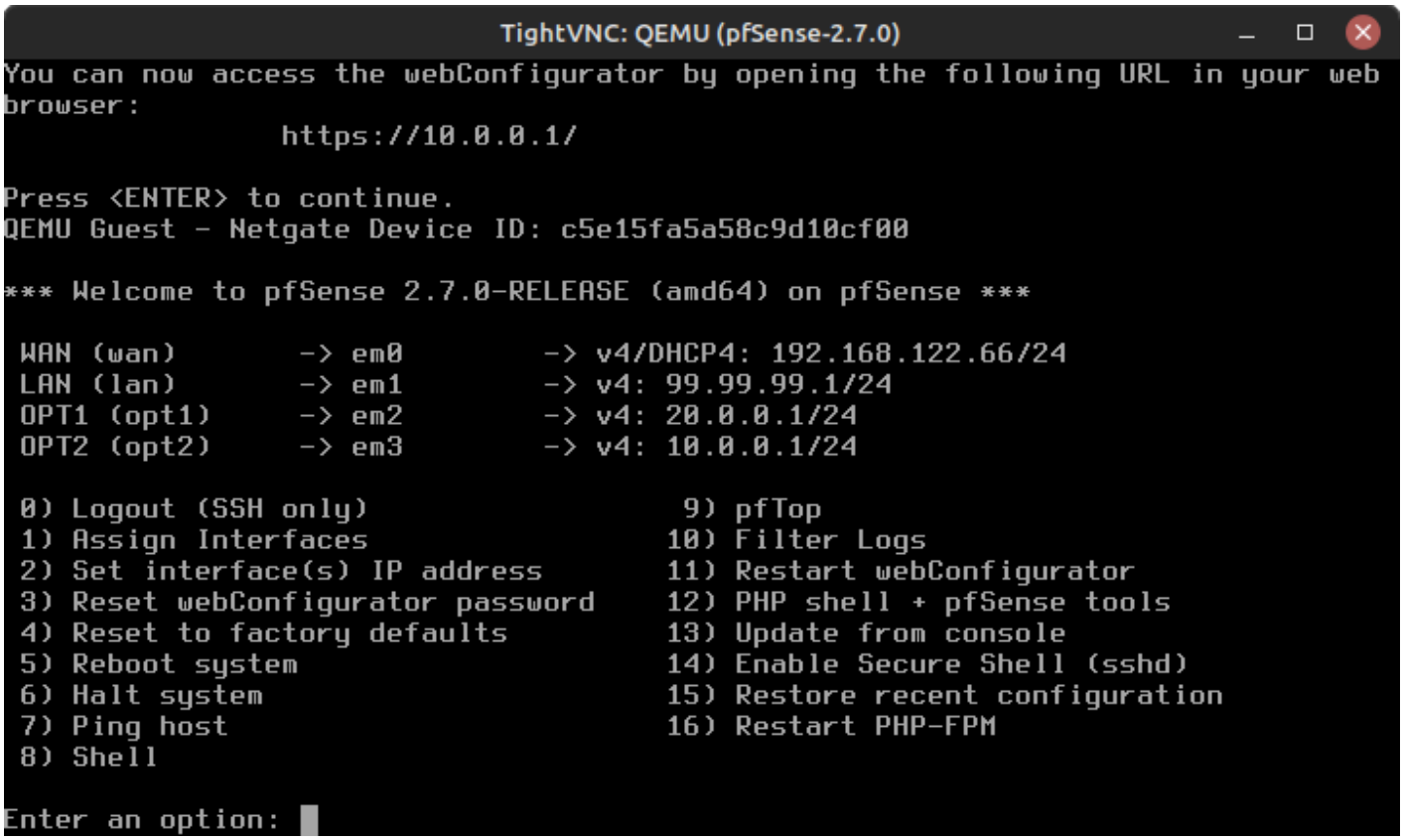


Figure 11 – All pfSense interfaces configured

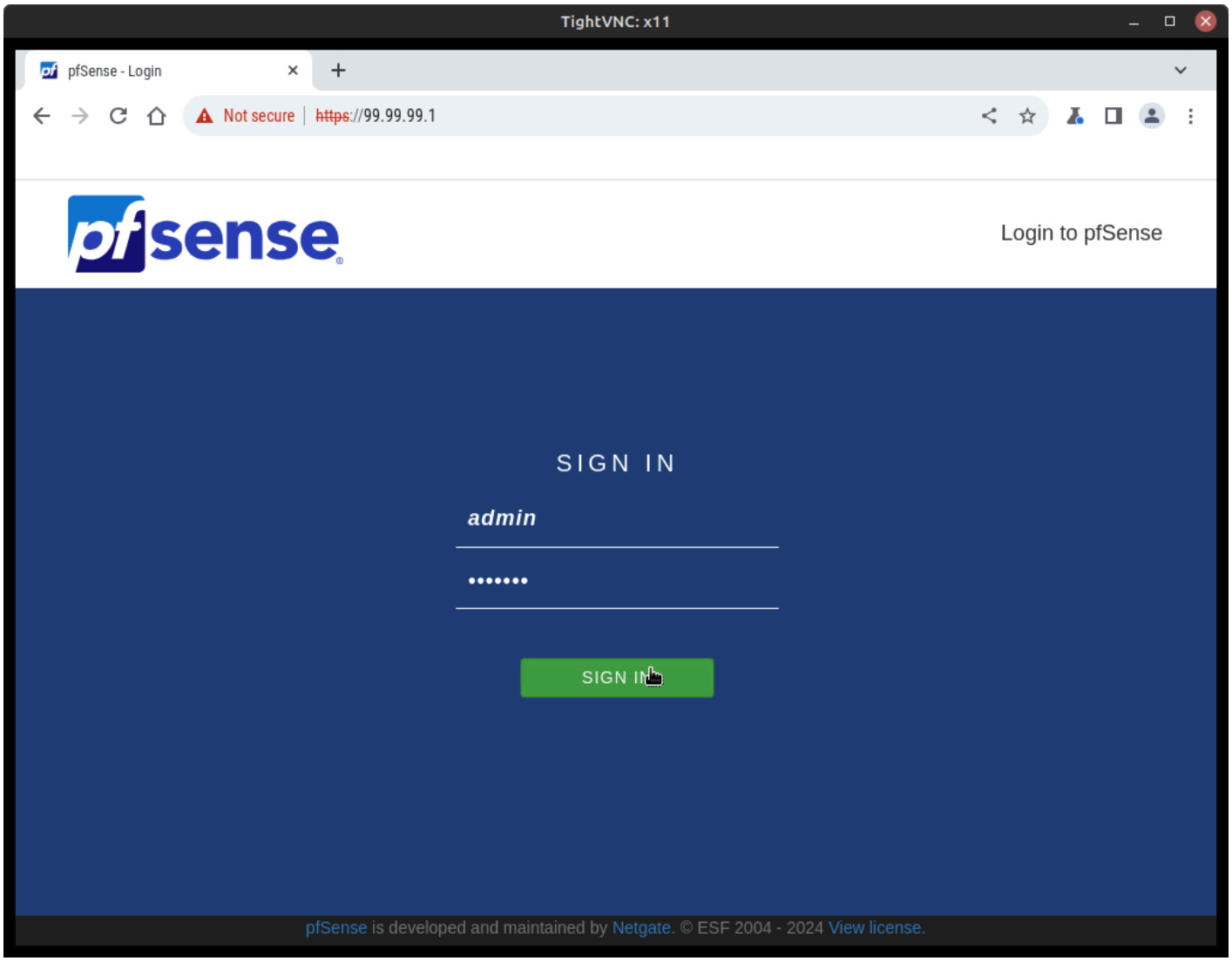


Figure 14 – pfSense webConfigurator login page

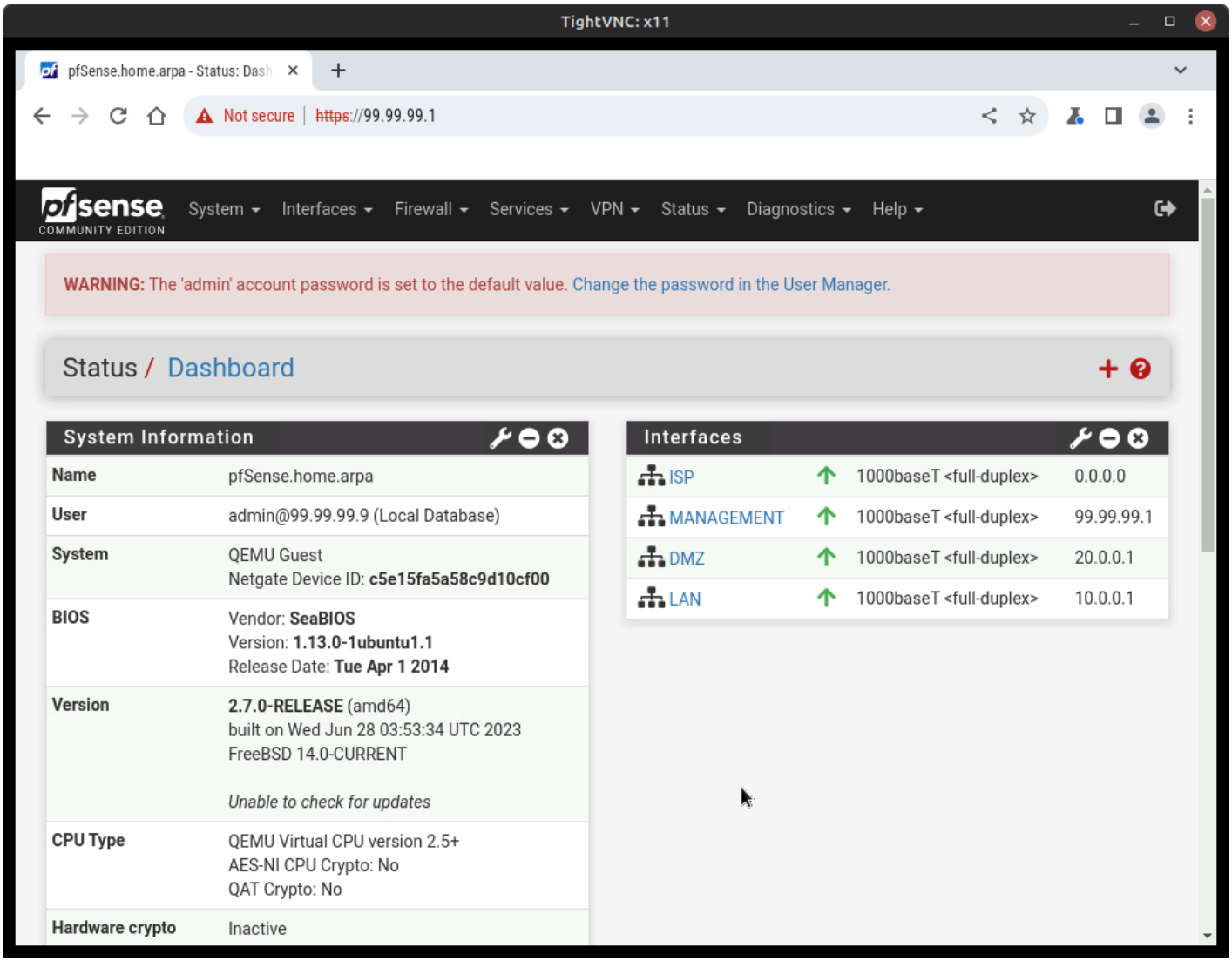


Figure 15 – pfSense main dashboard

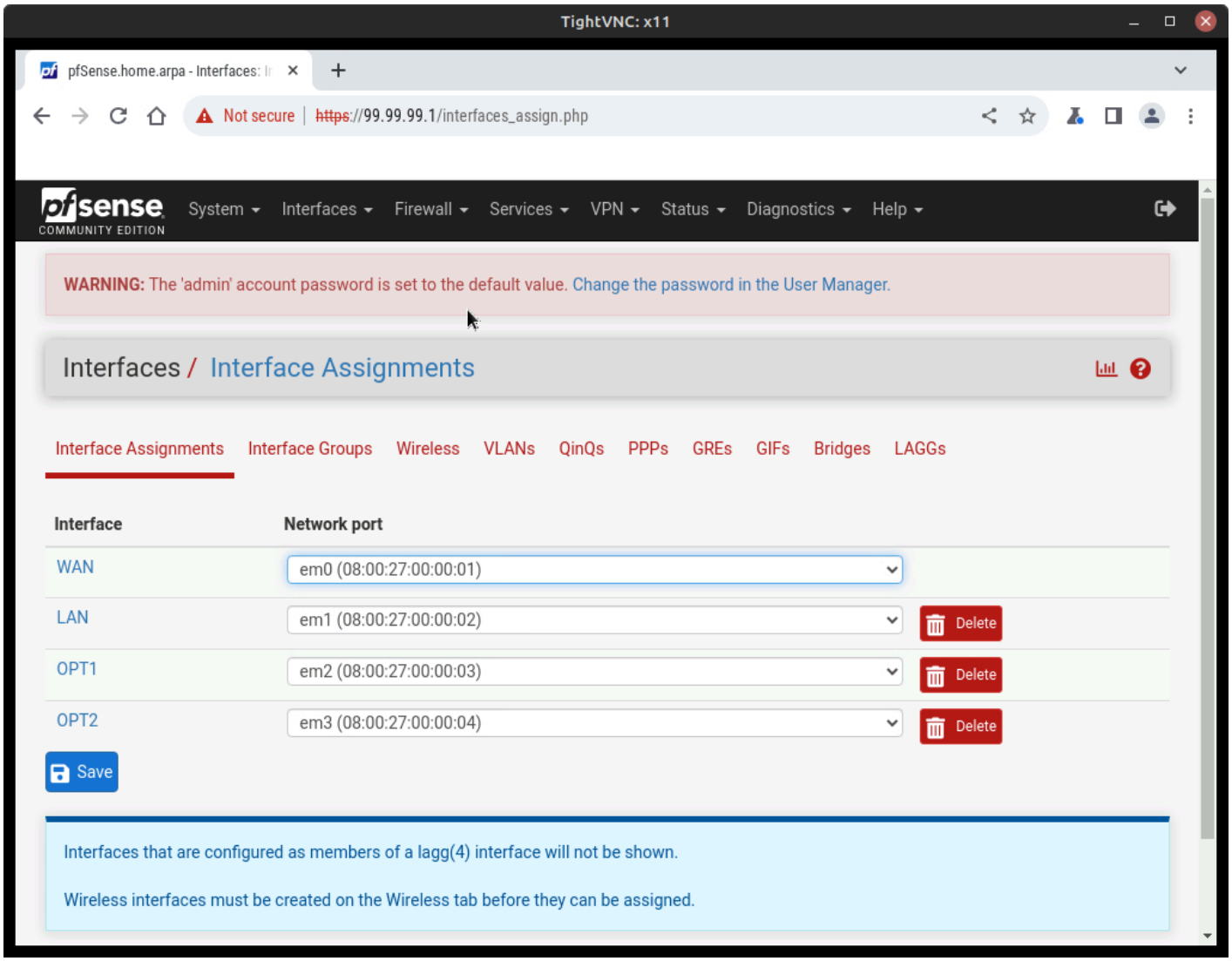


Figure 16 – pfSense interface assignments

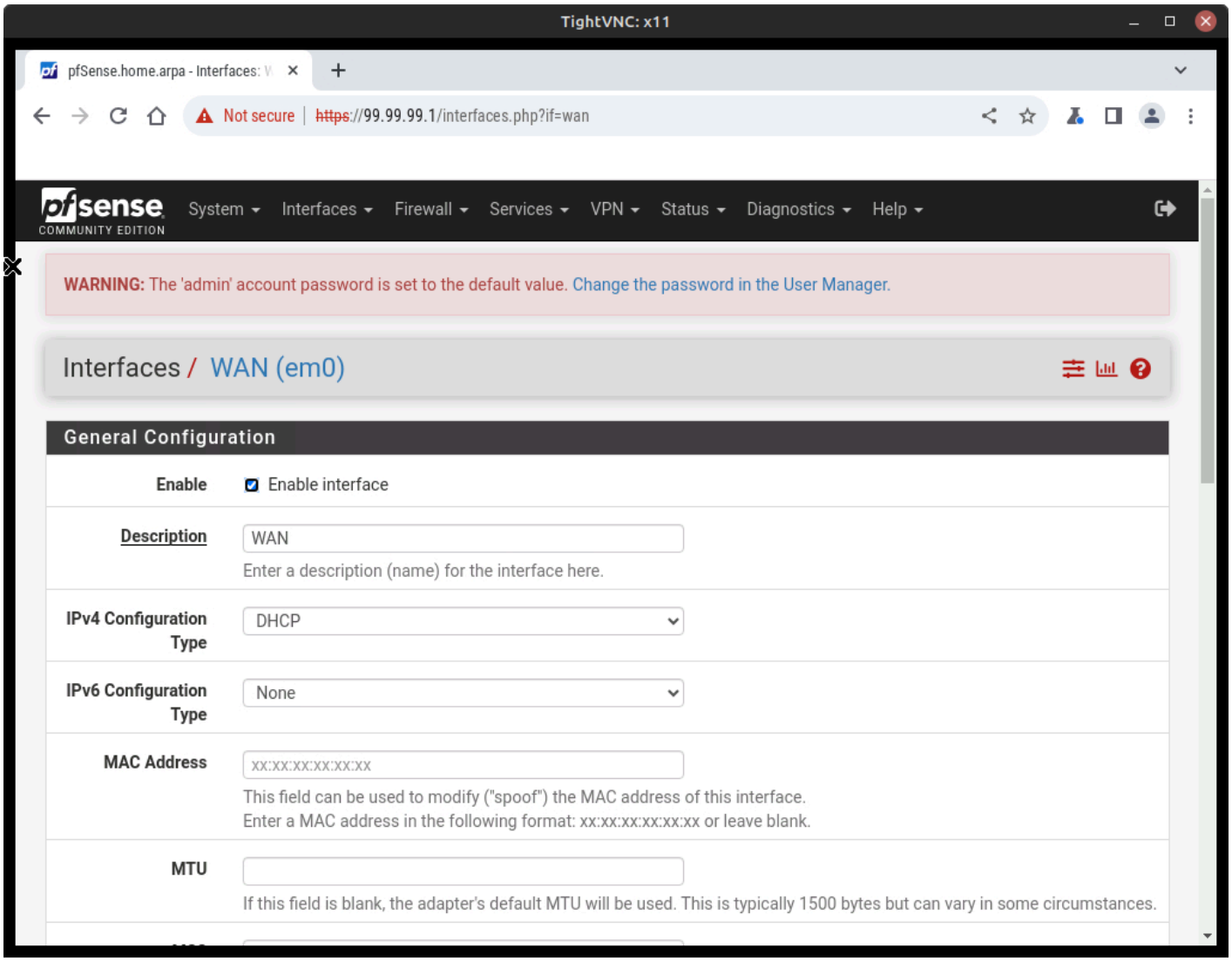


Figure 17 – em0 interface configuration

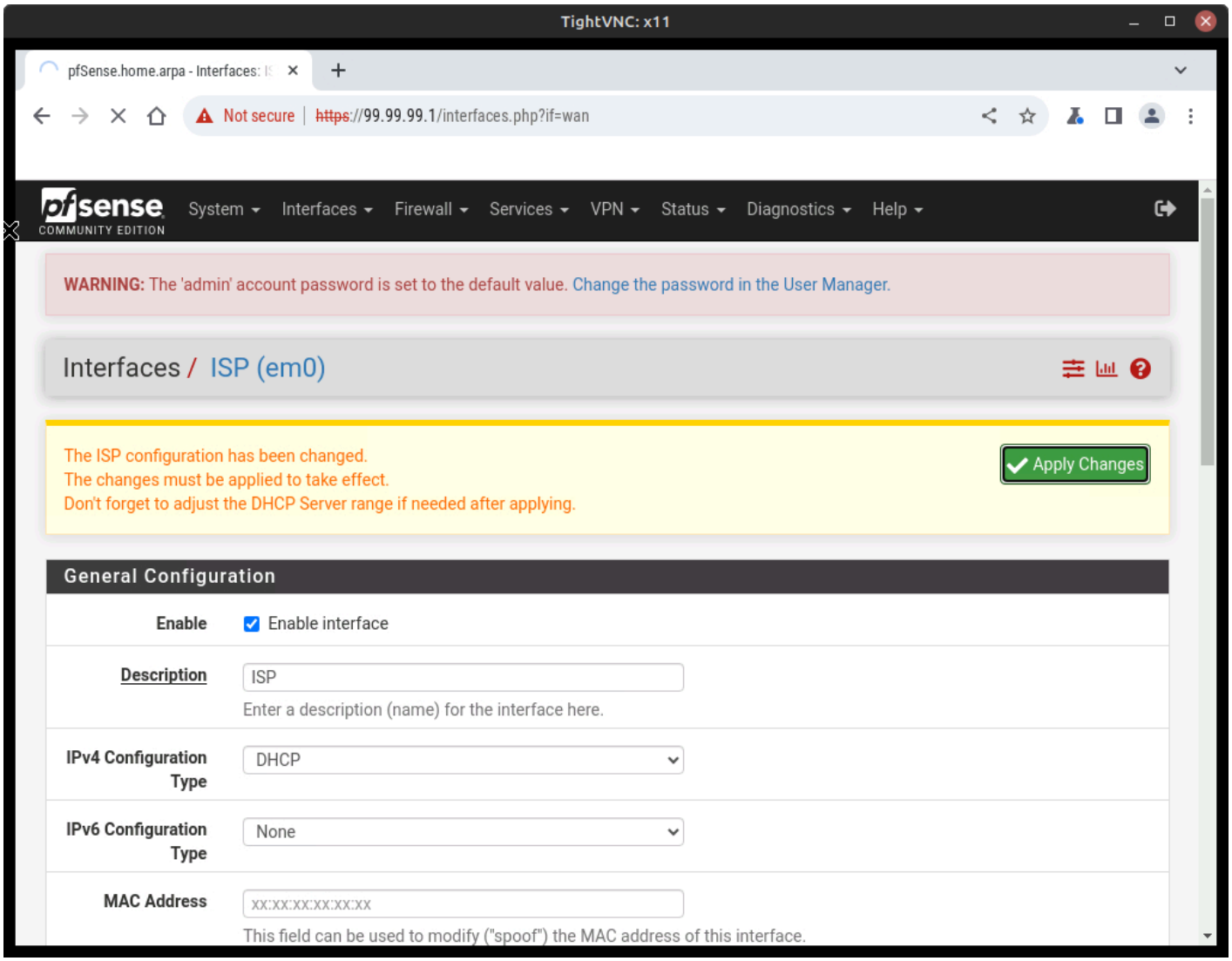


Figure 18 – Apply changes after saving

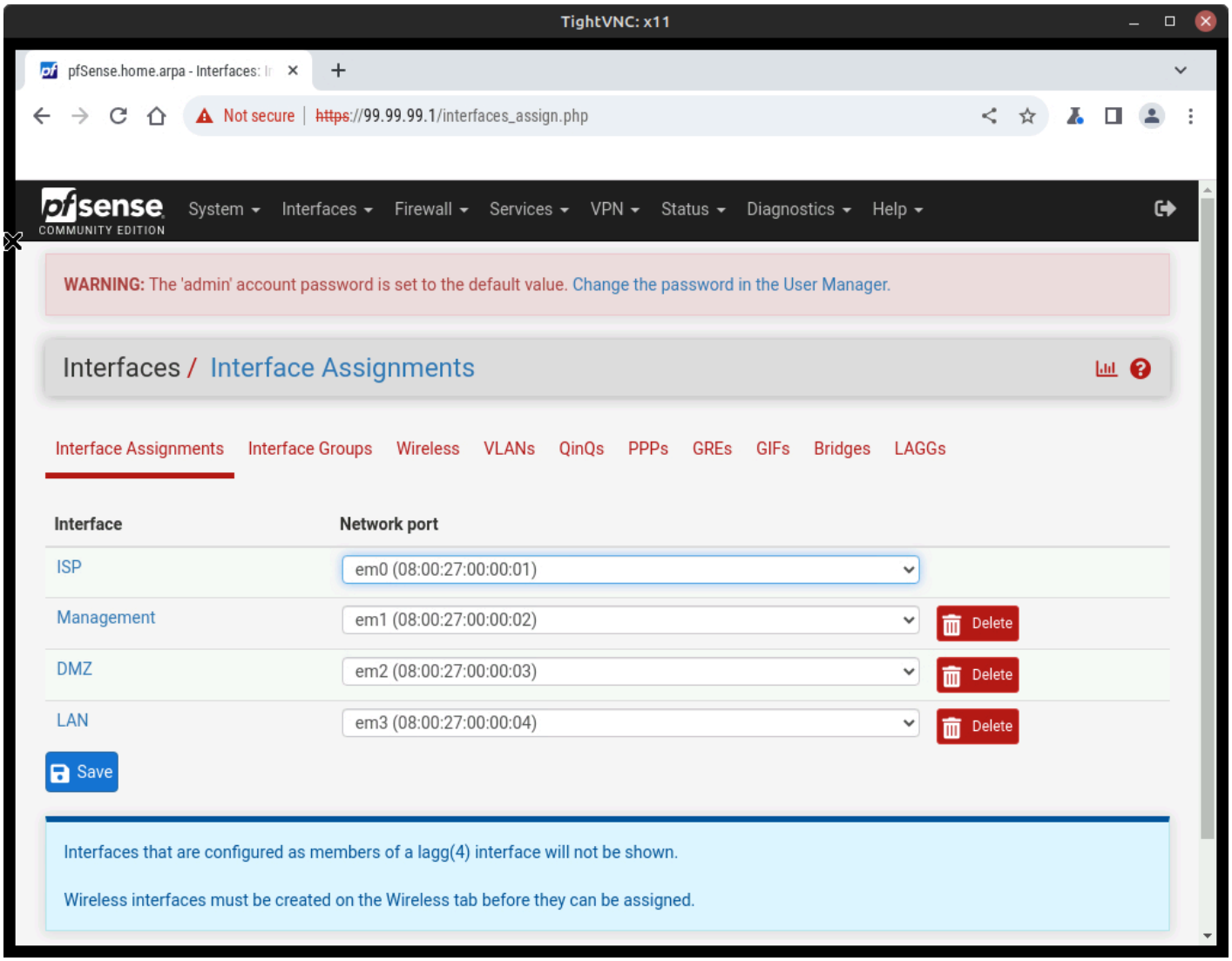


Figure 19 – Updated interface assignments

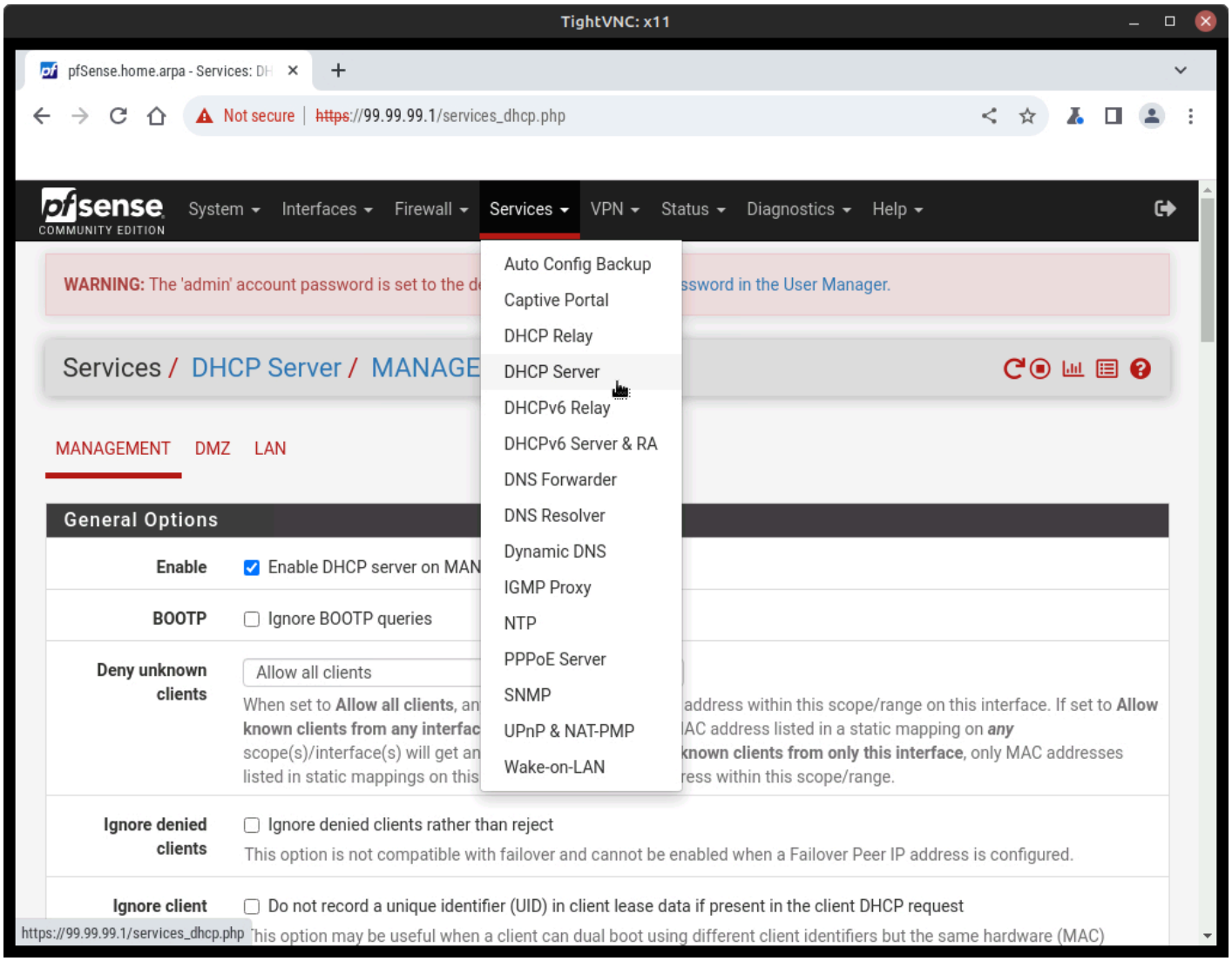


Figure 20 – DHCP server management screen

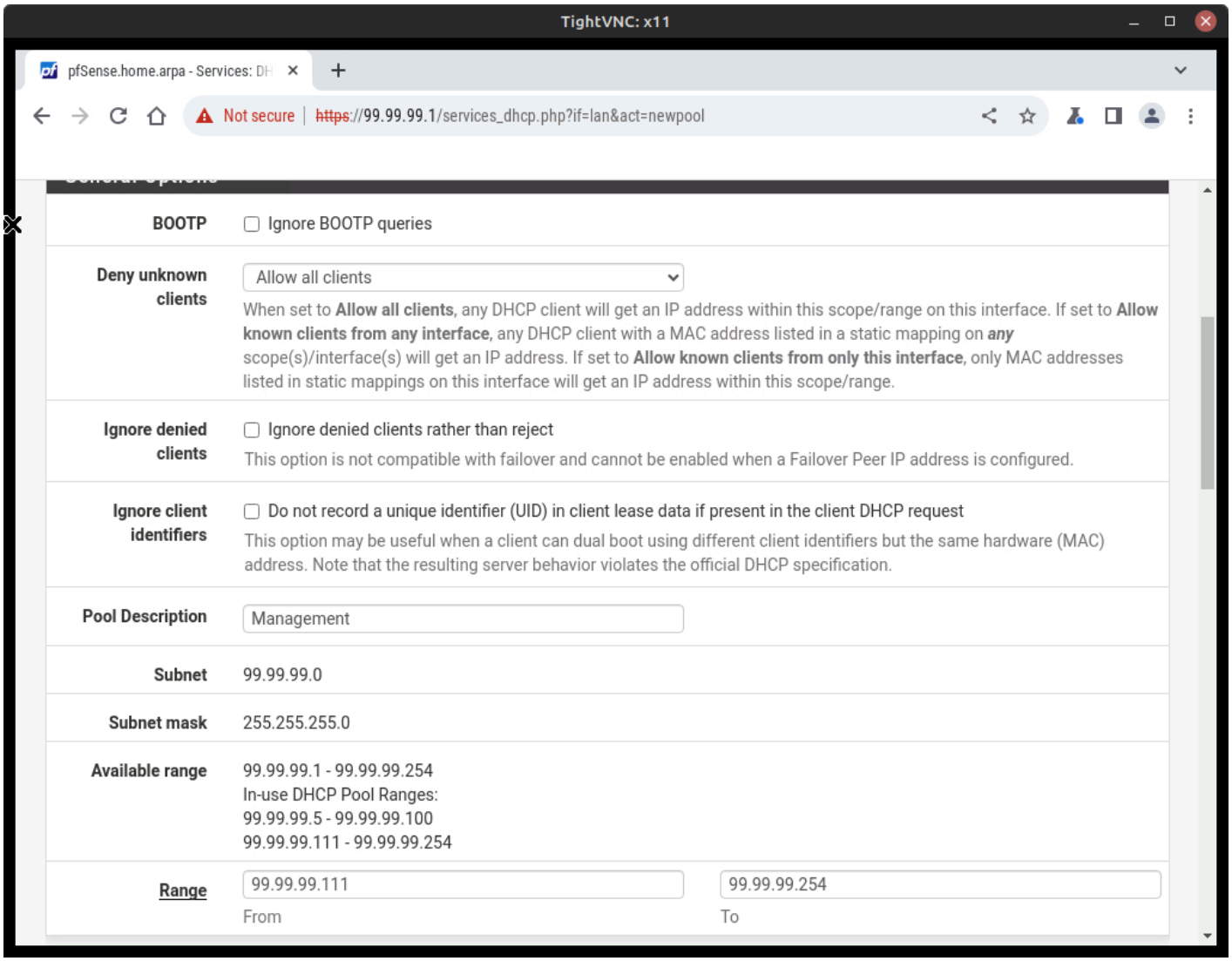


Figure 21 – Updated DHCP settings for Management LAN