UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 1: Introduction to Mininet

Document Version: 05-27-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab provides an introduction to Mininet, a virtual testbed used for testing network tools and protocols. It demonstrates how to invoke Mininet from the command-line interface (CLI) utility and how to build and emulate topologies using a graphical user interface (GUI) application. In this lab we will use Containernet, a Mininet network emulator fork that allows to use Docker containers as hosts in emulated network topologies. However, all the concepts covered are bounded to Mininet.

Objectives

By the end of this lab, students should be able to:

1. Understand what Mininet is and why it is useful for testing network topologies.
2. Invoke Mininet from the CLI.
3. Construct network topologies using the GUI.
4. Save/load Mininet topologies using the GUI.
5. Configure the interfaces of a router using the CLI.

Lab settings

The information in Table 1 provides the credentials of the machine containing Mininet.

Table 1**.** Credentials to access Client1 machine.

Device	Account	Password
Client1	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction to Mininet.
2. Section 2: Invoke Mininet using the CLI.
3. Section 3: Build and emulate a network in Mininet using the GUI.
4. Section 4: Configure router r1.

1 Introduction to Mininet

Mininet is a virtual testbed enabling the development and testing of network tools and protocols. With a single command, Mininet can create a realistic virtual network on any type of machine (Virtual Machine (VM), cloud-hosted, or native). Therefore, it provides

an inexpensive solution and streamlined development running in line with production networks[1]. Mininet offers the following features:

- Fast prototyping for new networking protocols.
- Simplified testing for complex topologies without the need of buying expensive hardware.
- Realistic execution as it runs real code on the Unix and Linux kernels.
- Open source environment backed by a large community contributing extensive documentation.

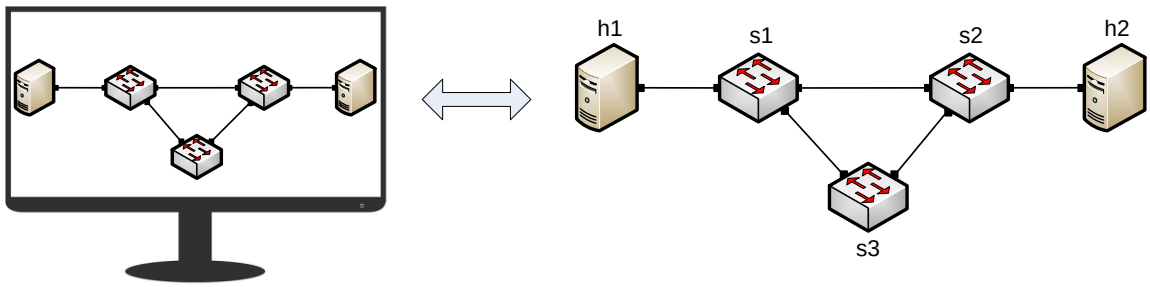

Figure 1. Hardware network vs. Mininet emulated network.

Mininet is useful for development, teaching, and research as it is easy to customize and interact with it through the CLI or the GUI. Mininet was originally designed to experiment with *OpenFlow*[2] and *Software-Defined Networking (SDN)*[3]. This lab, however, only focuses on emulating a simple network environment without SDN-based devices.

Mininet's logical nodes can be connected into networks. These nodes are sometimes called containers, or more accurately, *network namespaces*. Containers consume sufficiently fewer resources that networks of over a thousand nodes have created, running on a single laptop. A Mininet container is a process (or group of processes) that no longer has access to all the host system's native network interfaces. Containers are then assigned virtual Ethernet interfaces, which are connected to other containers through a virtual switch[4]. Mininet connects a host and a switch using a virtual Ethernet (veth) link. The veth link is analogous to a wire connecting two virtual interfaces, as illustrated below.

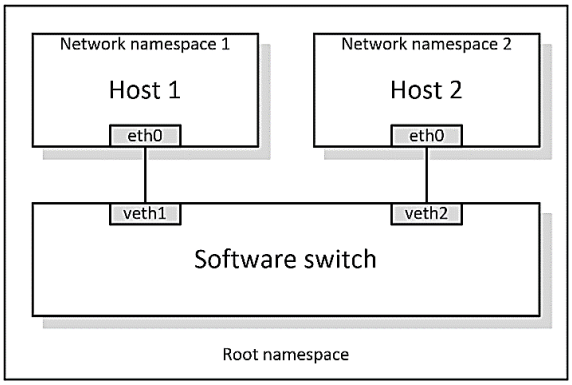

Figure 2. Network namespaces and virtual Ethernet links.

Each container is an independent network namespace, a lightweight virtualization feature that provides individual processes with separate network interfaces, routing tables, and Address Resolution Protocol (ARP) tables.
Mininet provides network emulation opposed to simulation, allowing all network software at any layer to be simply run *as is*; i.e. nodes run the native network software of the physical machine. On the other hand, in a simulated environment applications and protocol implementations need to be ported to run within the simulator before they can be used.

2 Invoke Mininet using the CLI

The first step to start Mininet using the CLI is to start a Linux terminal.

2.1 Invoke Mininet using the default topology

Step 1. Launch a Linux terminal by holding the `Ctrl+Alt+T` keys or by clicking on the Linux terminal icon.

Figure 3. Shortcut to open a Linux terminal.

The Linux terminal is a program that opens a window and permits you to interact with a command-line interface (CLI). A CLI is a program that takes commands from the keyboard and sends them to the operating system for execution.

Step 2. To start a minimal topology, enter the command shown below. When prompted for a password, type `password` and hit enter. Note that the password will not be visible as you type it.

```
sudo mn
```

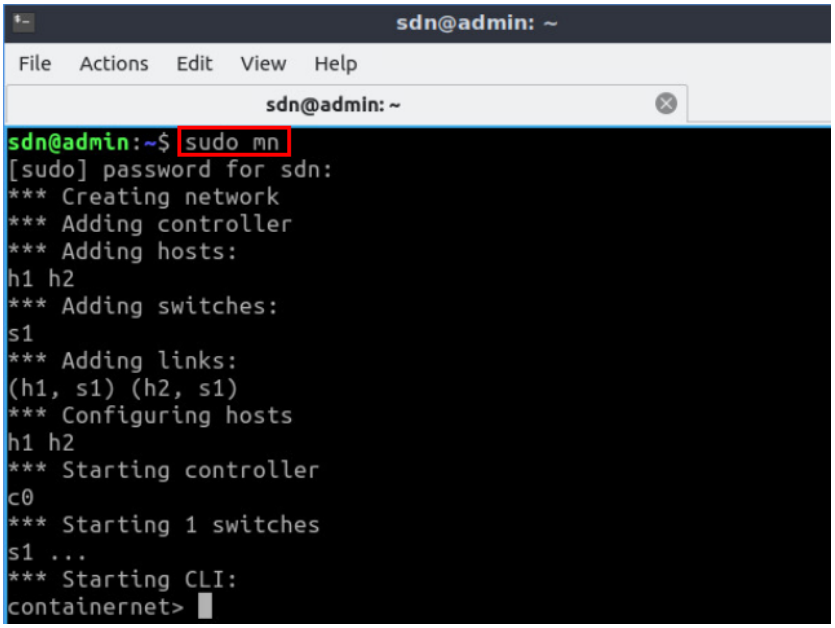

Figure 4. Starting Mininet using the CLI.

The above command starts Mininet with a minimal topology, which consists of a switch connected to two hosts as shown below.

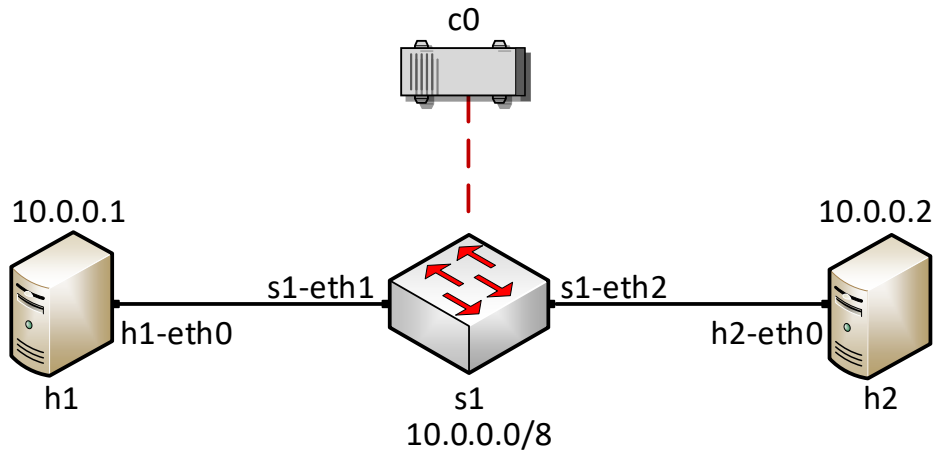

Figure 5. Mininet's default minimal topology.

When issuing the `sudo mn` command, Mininet initializes the topology and launches its command line interface which looks like this:

```
mininet>
```

Step 3. To display the list of Mininet CLI commands and examples on their usage, type the following command:

```
help
```

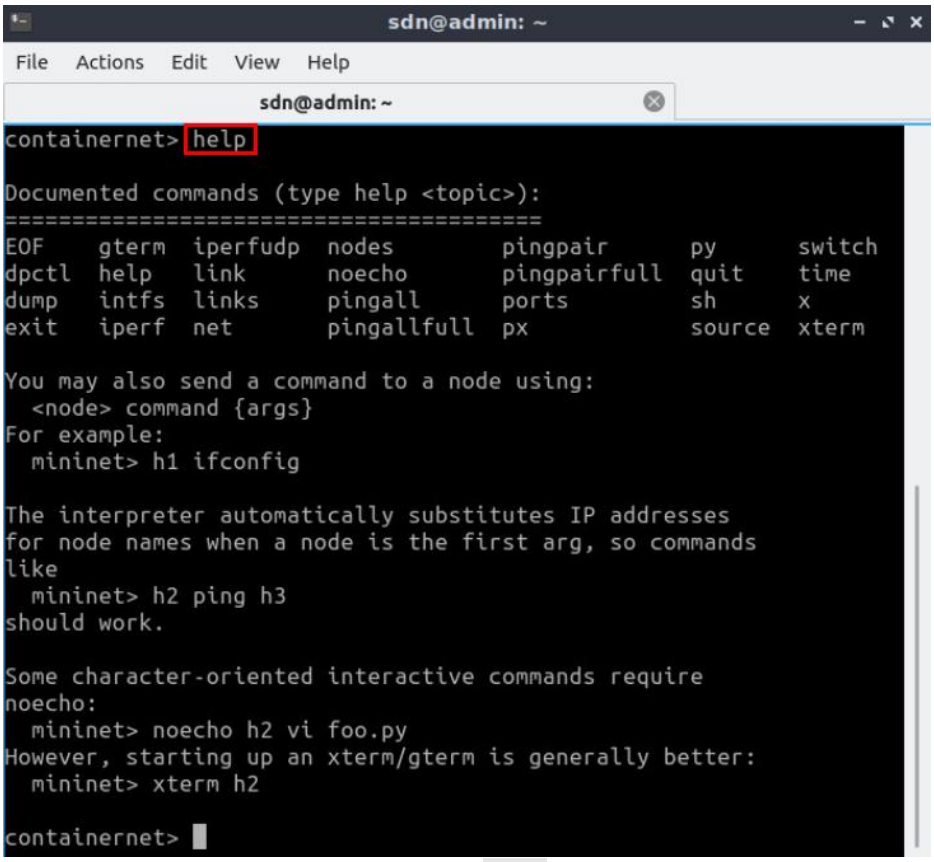

Figure 6. Mininet's `help` command.

Step 4. To display the available nodes, type the following command:

```
nodes
```

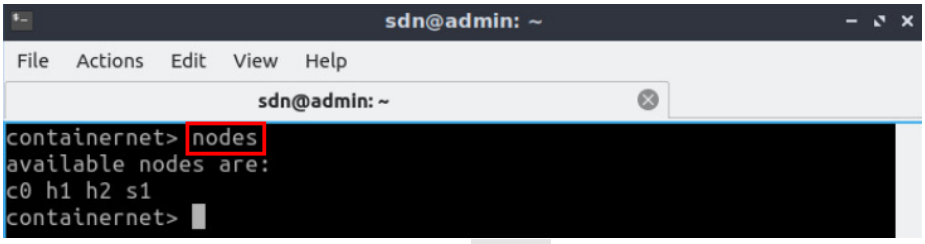

Figure 7. Mininet's `nodes` command.

The output of this command shows that there is a controller, two hosts (host h1 and host h2), and a switch (s1).

Step 5. It is useful sometimes to display the links between the devices in Mininet to understand the topology. Issue the command shown below to see the available links.

```
net
```

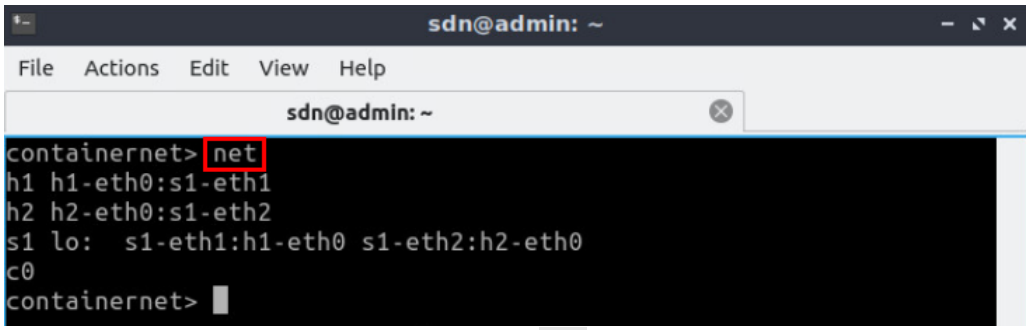

Figure 8. Mininet's `net` command.

The output of this command shows that:

1. Host h1 is connected using its network interface *h1-eth0* to the switch on interface *s1-eth1*.
2. Host h2 is connected using its network interface *h2-eth0* to the switch on interface *s1-eth2*.
3. Switch s1:
    a. has a loopback interface *lo*.
    b. connects to *h1-eth0* through interface *s1-eth1*.
    c. connects to *h2-eth0* through interface *s1-eth2*.
4. Controller c0 is the brain of the network, where it has a global knowledge about the network. A controller instructs the switches on how to forward/drop packets in the network.

Mininet allows you to execute commands on a specific device. To issue a command for a specific node, you must specify the device first, followed by the command.

Step 6. To proceed, issue the command:

```
h1 ifconfig
```

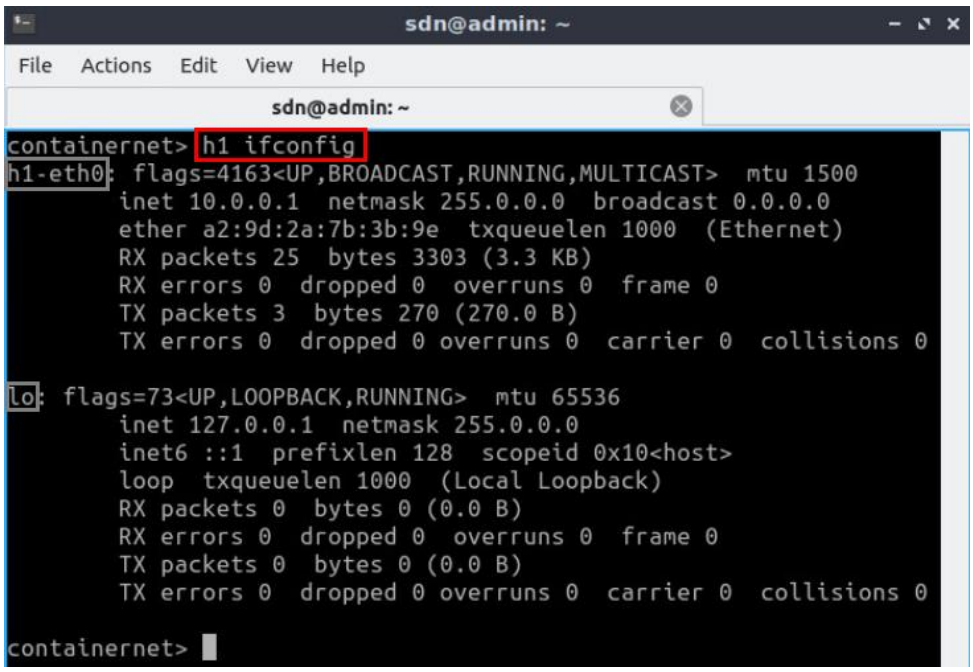

Figure 9. Output of `h1 ifconfig` command.

This command executes the `ifconfig` Linux command on host h1. The command shows host h1's interfaces. The display indicates that host h1 has an interface *h1-eth0* configured with IP address 10.0.0.1, and another interface lo configured with IP address 127.0.0.1 (loopback interface).

2.2 Test connectivity

Mininet's default topology assigns the IP addresses 10.0.0.1/8 and 10.0.0.2/8 to host h1 and host h2 respectively. To test connectivity between them, you can use the command `ping`. The `ping` command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the remote computer and waiting for a response. Information available includes how many responses are returned and how long it takes for them to return.

Step 1. On the CLI, type the command shown below. This command tests the connectivity between host h1 and host h2. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test. Host h1 (10.0.0.1) sent four packets to host h2 (10.0.0.2) and successfully received the expected responses.

```
h1 ping 10.0.0.2
```

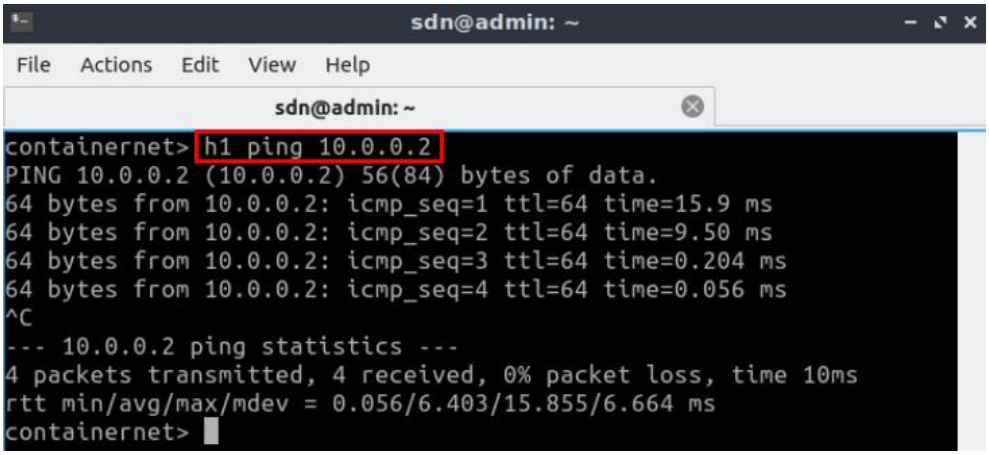

Figure 10. Connectivity test between host h1 and host h2.

Step 2. Stop the emulation by typing the following command:

```
exit
```

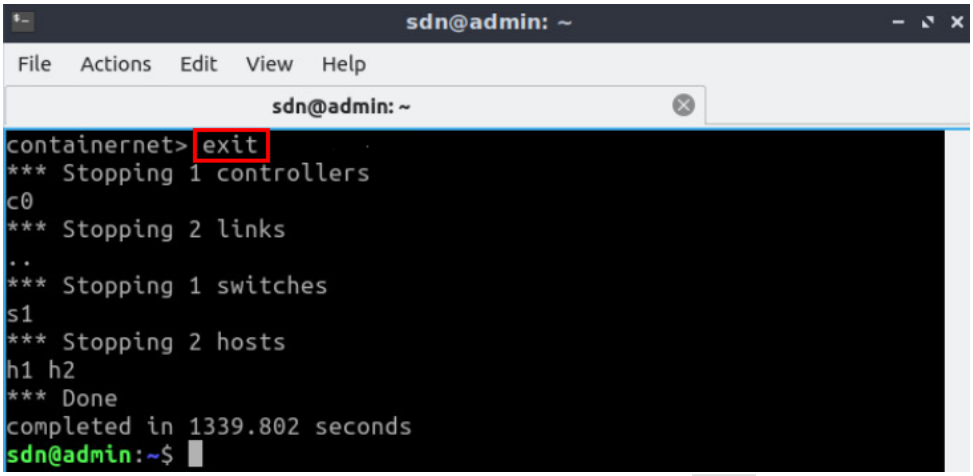

Figure 11. Stopping the emulation using `exit`.

The command `sudo mn -c` is often used on the Linux terminal (not on the Mininet CLI) to clean a previous instance of Mininet (e.g., after a crash).

3 Build and emulate a network in Mininet using the GUI

In this section, you will use the application MiniEdit[5] to deploy the topology illustrated below. MiniEdit is a simple GUI network editor for Mininet.

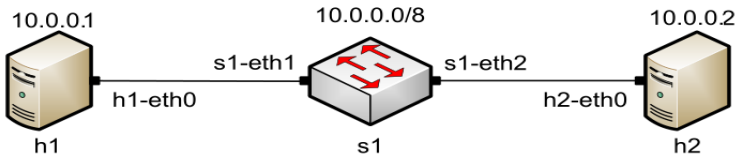

Figure 12. Lab topology.

3.1 Build the network topology

Step 1. A shortcut to MiniEdit is located on the machine's Desktop. Start MiniEdit by clicking on MiniEdit's shortcut. When prompted for a password, type `password`.

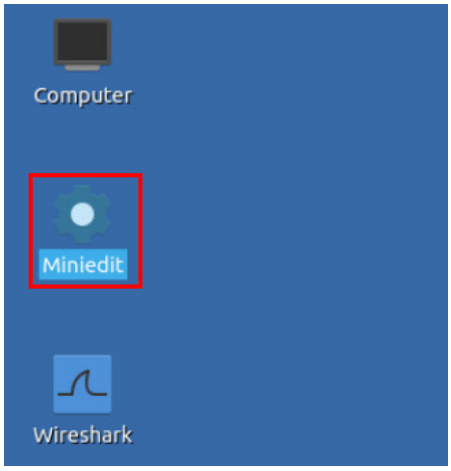

Figure 13. MiniEdit Desktop shortcut.

MiniEdit will start, as illustrated below.

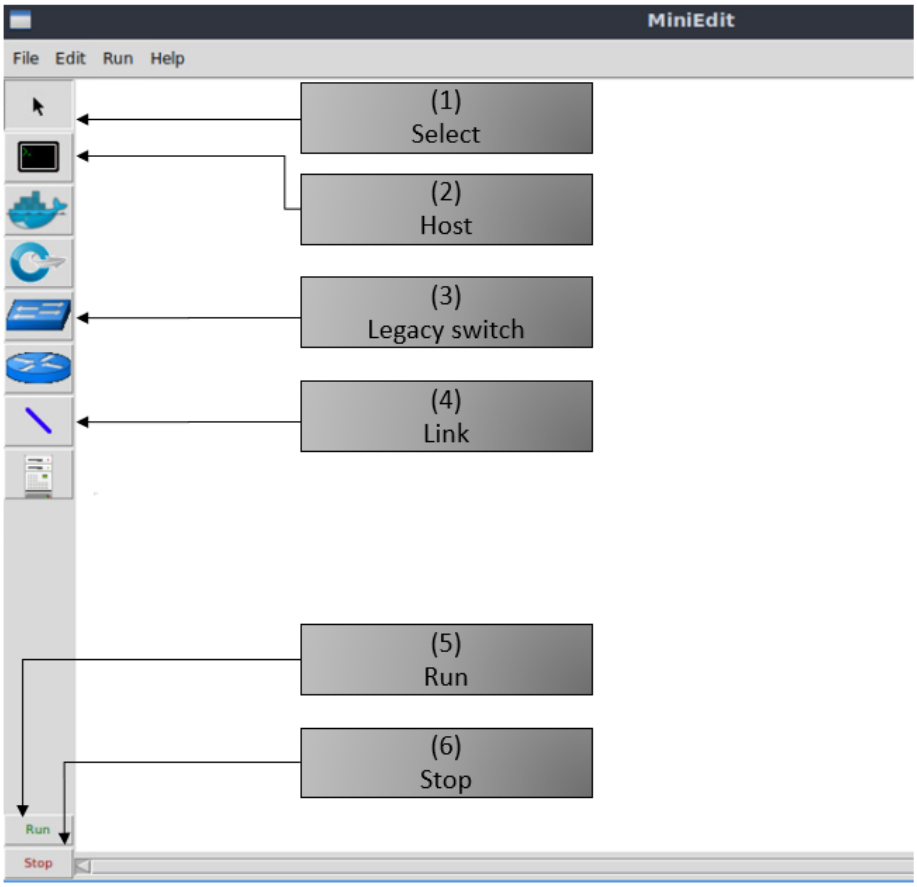

Figure 14. MiniEdit Graphical User Interface (GUI).

The main buttons in this lab are:

1. *Select*: allows selection/movement of the devices. Pressing *Del* on the keyboard after selecting the device removes it from the topology.
2. *Host*: allows addition of a new host to the topology. After clicking this button, click anywhere in the blank canvas to insert a new host.
3. *Legacy switch*: allows addition of a new legacy switch to the topology. After clicking this button, click anywhere in the blank canvas to insert the switch.
4. *Link*: connects devices in the topology (mainly switches and hosts). After clicking this button, click on a device and drag to the second device to which the link is to be established.
5. *Run*: starts the emulation. After designing and configuring the topology, click the run button.
6. *Stop*: stops the emulation.

Step 2. To build the topology illustrated in Figure 12, two hosts and one switch must be deployed. Deploy these devices in MiniEdit, as shown below.

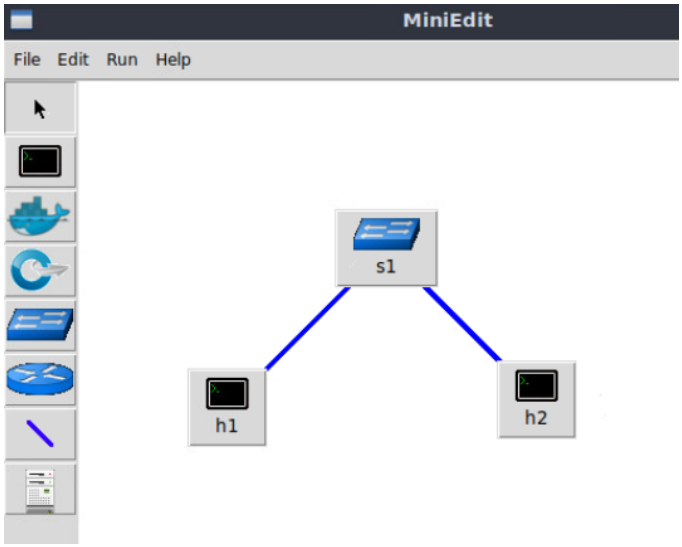

Figure 15. MiniEdit's topology.

Use the buttons described in the previous step to add and connect devices. The configuration of IP addresses is described in Step 3.

Step 3. Configure the IP addresses of host h1 and host h2. Host h1's IP address is 10.0.0.1/8 and host h2's IP address is 10.0.0.2/8. A host can be configured by holding the right click and selecting properties on the device. For example, host h2 is assigned the IP address 10.0.0.2/8 in the figure below.

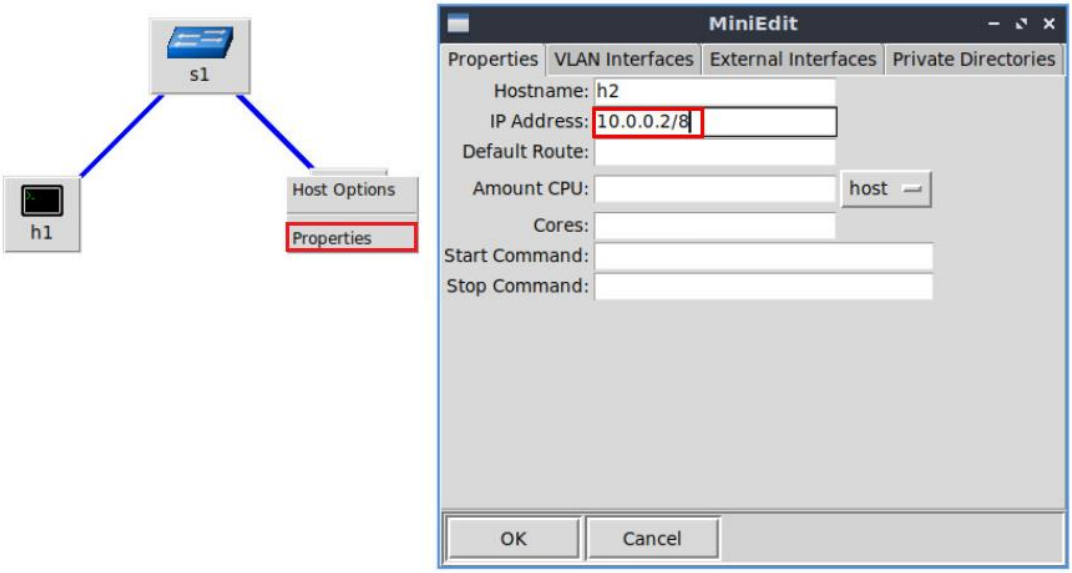

Figure 16. Configuration of a host's properties.

3.2 Test connectivity

Before testing the connection between host h1 and host h2, the emulation must be started.

Step 1. Click on the *Run* button to start the emulation. The emulation will start and the buttons of the MiniEdit panel will gray out, indicating that they are currently disabled.

Figure 17. Starting the emulation.

Step 2. Open a terminal on host h1 by holding the right click on host h1 and selecting *Terminal*. This opens a terminal on host h1 and allows the execution of commands on the host h1. Repeat the procedure on host h2.

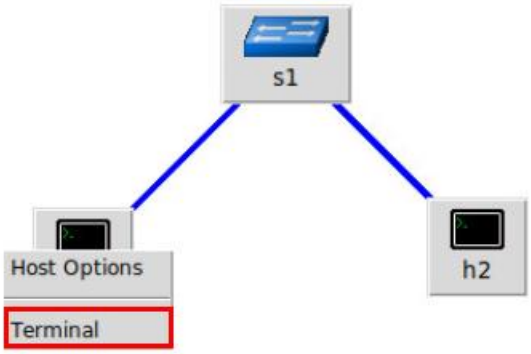

Figure 18. Opening a terminal on host h1.

The network and terminals at host h1 and host h2 will be available for testing.

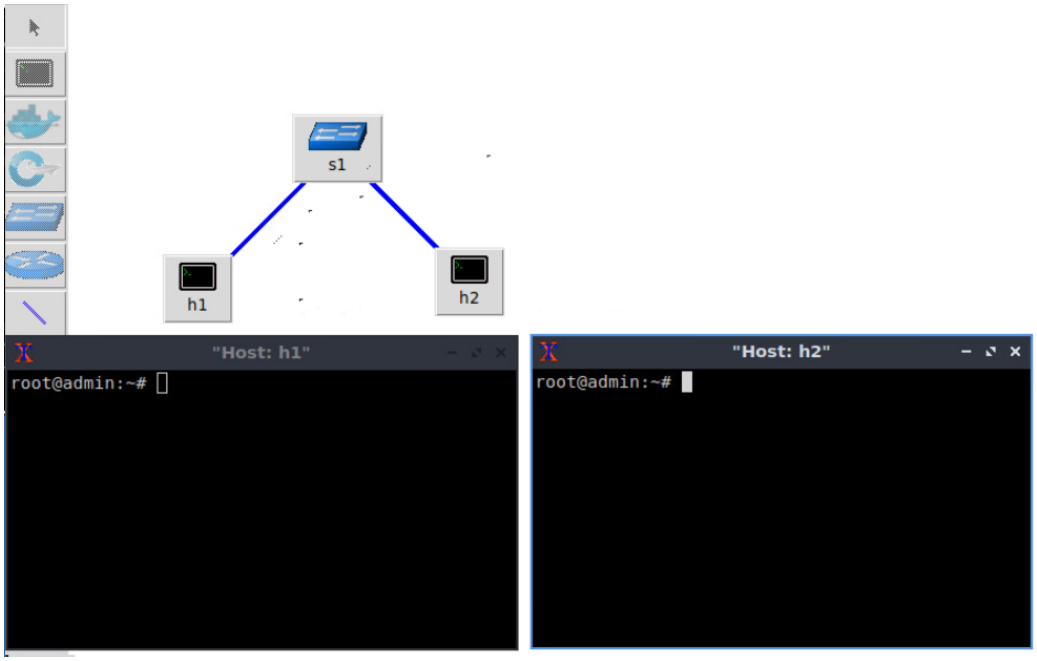

Figure 19. Terminals at host h1 and host h2.

Step 3. On host h1's terminal, type the command shown below to display its assigned IP addresses. The interface *h1-eth0* at host h1 should be configured with the IP address 10.0.0.1 and subnet mask 255.0.0.0.

```
ifconfig
```

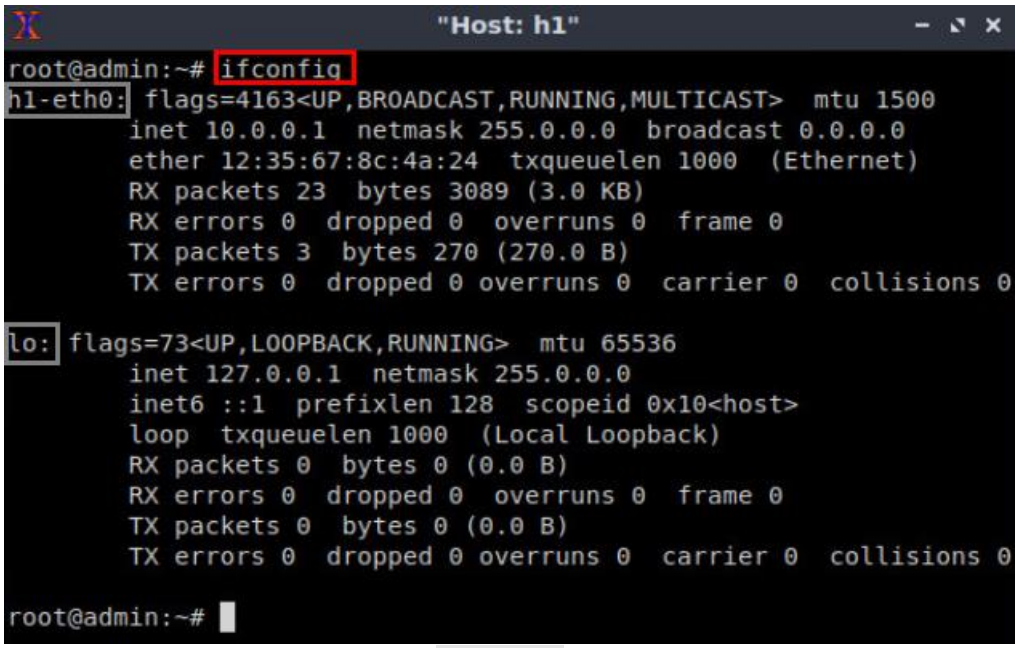

Figure 20. Output of `ifconfig` command on host h1.

Repeat Step 3 on host h2. Its interface *h2-eth0* should be configured with IP address 10.0.0.2 and subnet mask 255.0.0.0.

Step 4. On host h1's terminal, type the command shown below. This command tests the connectivity between host h1 and host h2. To stop the test, press `Ctrl+c`. The figure

below shows a successful connectivity test. Host h1 (10.0.0.1) sent six packets to host h2 (10.0.0.2) and successfully received the expected responses.

```
ping 10.0.0.2
```

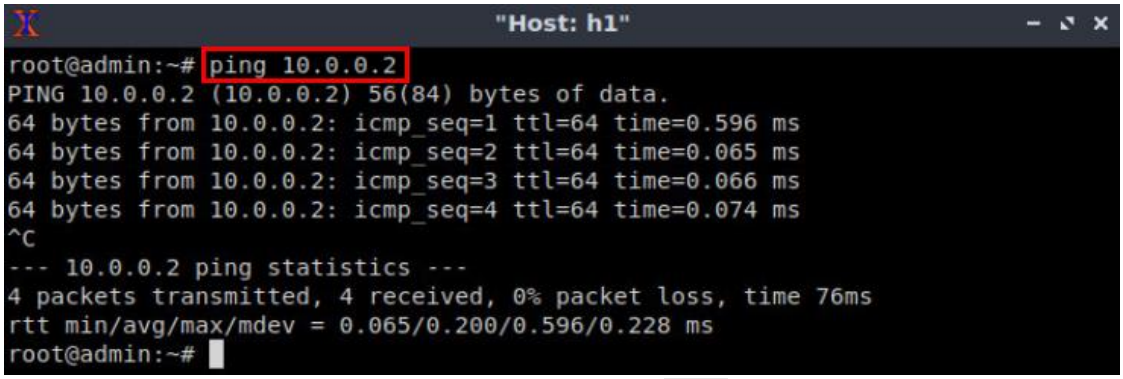

Figure 21. Connectivity test using `ping` command.

Step 5. Stop the emulation by clicking on the *Stop* button.

Figure 22. Stopping the emulation.

3.3 Automatic assignment of IP addresses

In the previous section, you manually assigned IP addresses to host h1 and host h2. An alternative is to rely on Mininet for an automatic assignment of IP addresses (by default, Mininet uses automatic assignment), which is described in this section.

Step 1. Remove the manually assigned IP address from host h1. Hold right-click on host h1, *Properties*. Delete the IP address, leaving it unassigned, and press the *OK* button as shown below. Repeat the procedure on host h2.

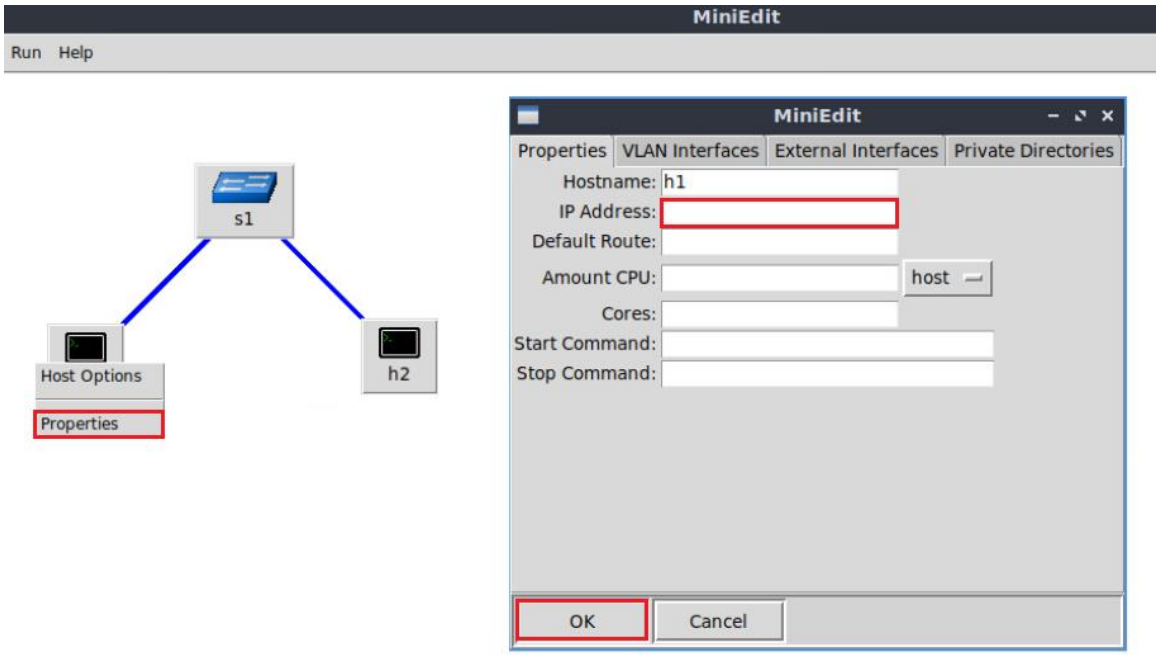

Figure 23. Host h1 properties.

Step 2. Click on *Edit*, *Preferences* button. The default IP base is 10.0.0.0/8. Modify this value to 15.0.0.0/8, and then press the *OK* button.

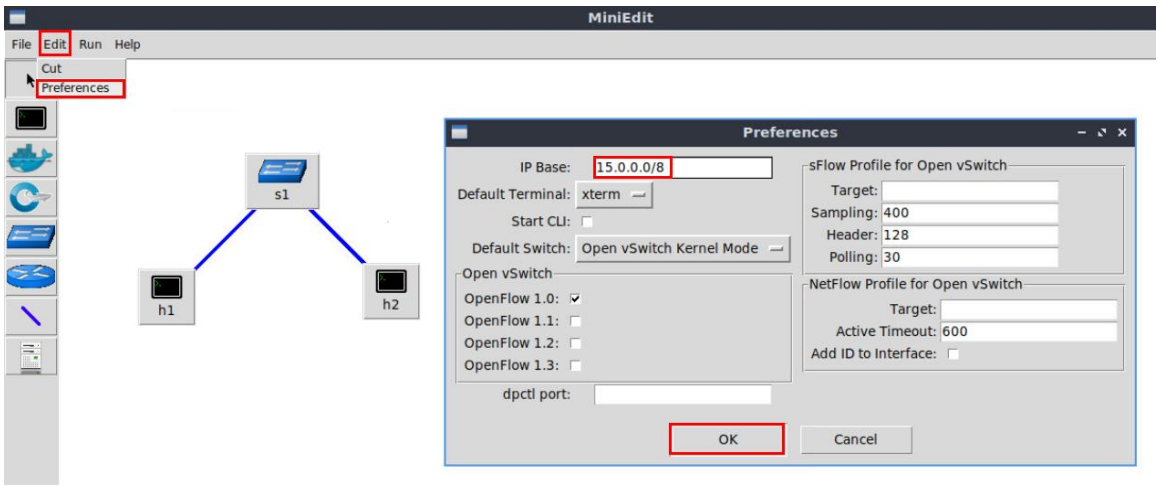

Figure 24. Modification of the IP Base (network address and prefix length).

Step 3. Run the emulation again by clicking on the *Run* button. The emulation will start and the buttons of the MiniEdit panel will be disabled.

Step 4. Open a terminal on host h1 by holding the right click on host h1 and selecting Terminal.

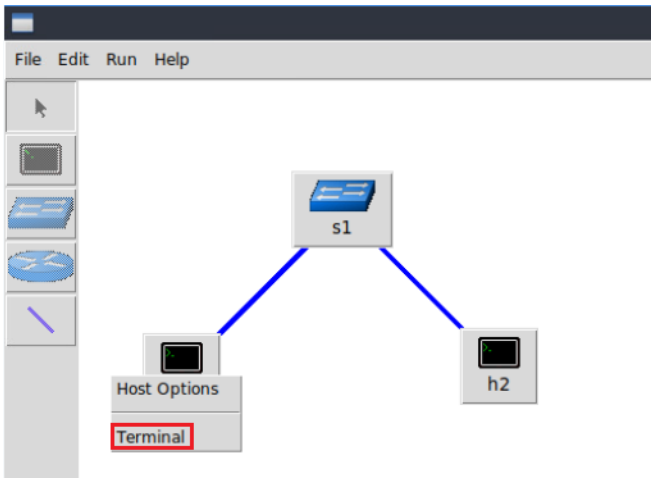

Figure 25. Opening a terminal on host h1.

Step 5. Type the command shown below to display the IP addresses assigned to host h1. The interface *h1-eth0* at host h1 now has the IP address 15.0.0.1 and subnet mask 255.0.0.0.

```
ifconfig
```

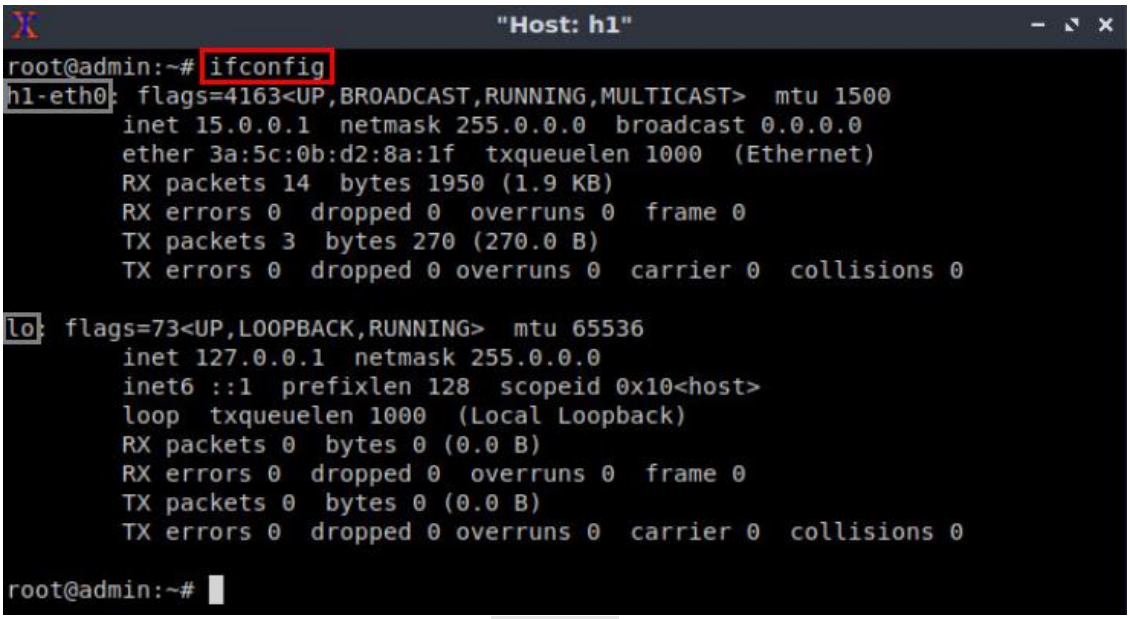

Figure 26. Output of `ifconfig` command on host h1.

You can also verify the IP address assigned to host h2 by repeating Steps 4 and 5 on host h2's terminal. The corresponding interface *h2-eth0* at host h2 has now the IP address 15.0.0.2 and subnet mask 255.0.0.0.

Step 6. Stop the emulation by clicking on *Stop* button.

Figure 27. Stopping the emulation.

3.4 Save and load a Mininet topology

In this section you will save and load a Mininet topology. It is often useful to save the network topology, particularly when its complexity increases. MiniEdit enables you to save the topology to a file.

Step 1. Save the current topology by clicking on *File* then *Save*. Provide a name for the topology and save it in the local folder. In this case, we used *myTopology* as the topology name.

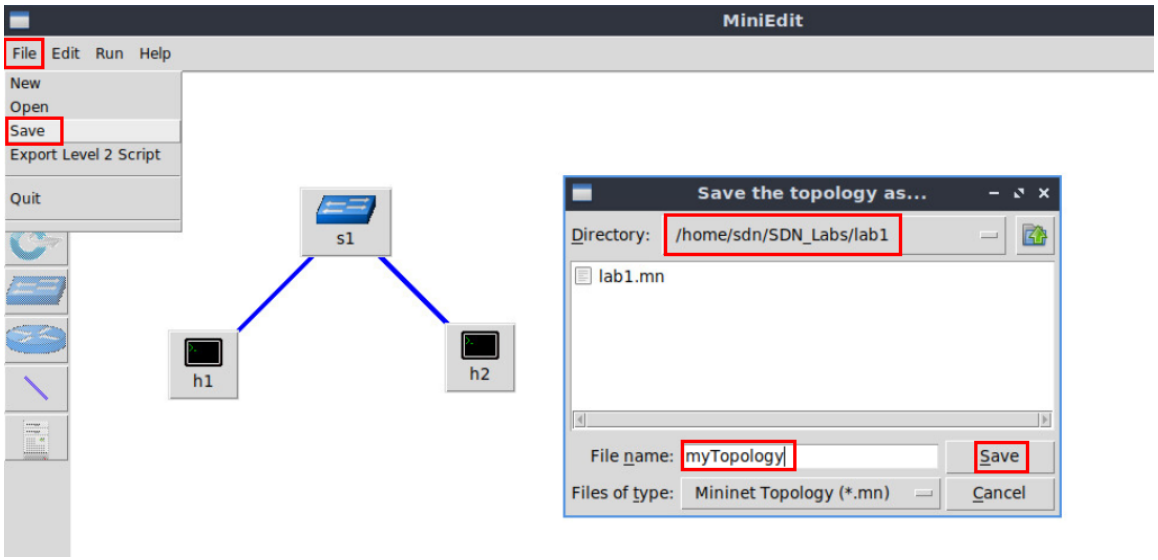

Figure 28. Saving the topology.

Step 2. Load the topology by clicking on *File* then *Open*. Search for the topology file called *lab1.mn* and click on *Open*. A new topology will be loaded to MiniEdit.

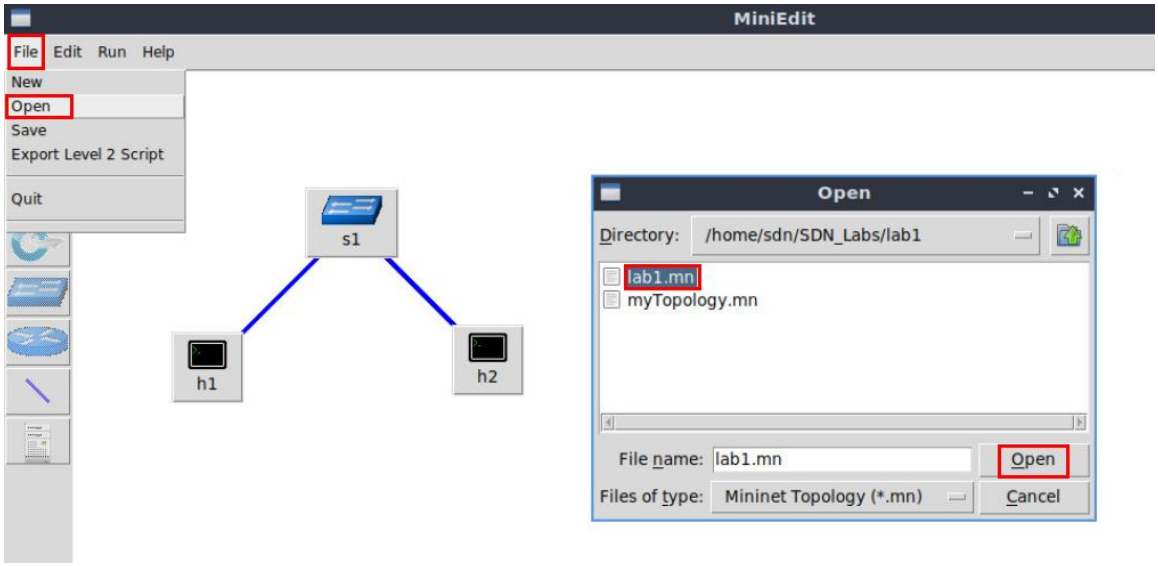

Figure 29. Opening a topology.

4 Configure router r1

In the previous section, you loaded a topology that consists in two networks directly connected to router r1. Consider Figure 30. In this topology two LANs, defined by switch s1 and switch s2 are connected to router r1. Initially, host h1 and host h2 do not have connectivity thus, you will configure router r1's interfaces in order to establish connectivity between the two networks.

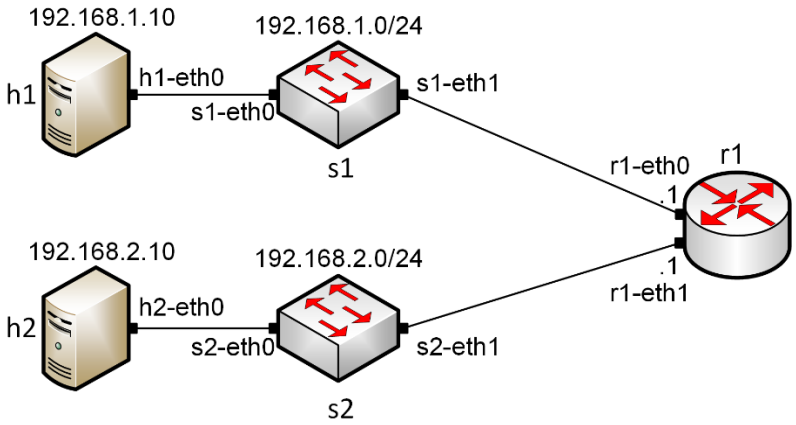

Figure 30. Topology.

Table 2 summarized the IP addresses used to configure router r1 and the end-hosts.

Table 2. Topology information.

Device	Interface	IP Address	Subnet	Default gateway
r1	r1-eth0	192.168.1.1	/24	N/A
	r1-eth1	192.168.2.1	/24	N/A
h1	h1-eth0	192.168.1.10	/24	192.168.1.1

h2	h2-eth0	192.168.2.10	/24	192.168.2.1

Step 1. Click on the *Run* button to start the emulation. The emulation will start and the buttons of the MiniEdit panel will gray out, indicating that they are currently disabled.

Figure 31. Starting the emulation.

4.1 Verify end-hosts configuration

In this section, you will verify that the IP addresses are assigned according to Table 2. Additionally, you will check routing information.

Step 1. Hold right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on that host.

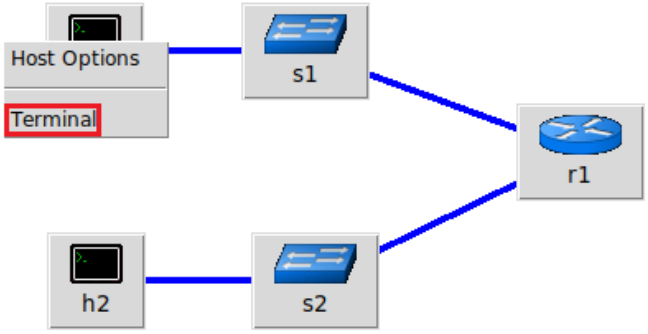

Figure 32. Opening a terminal on host h1.

Step 2. In host h1 terminal, type the command shown below to verify that the IP address was assigned successfully. You will verify that host h1 has two interfaces, *h1-eth0* configured with the IP address 192.168.1.10 and the subnet mask 255.255.255.0 and, the loopback interface *lo* configured with the IP address 127.0.0.1.

```
ifconfig
```

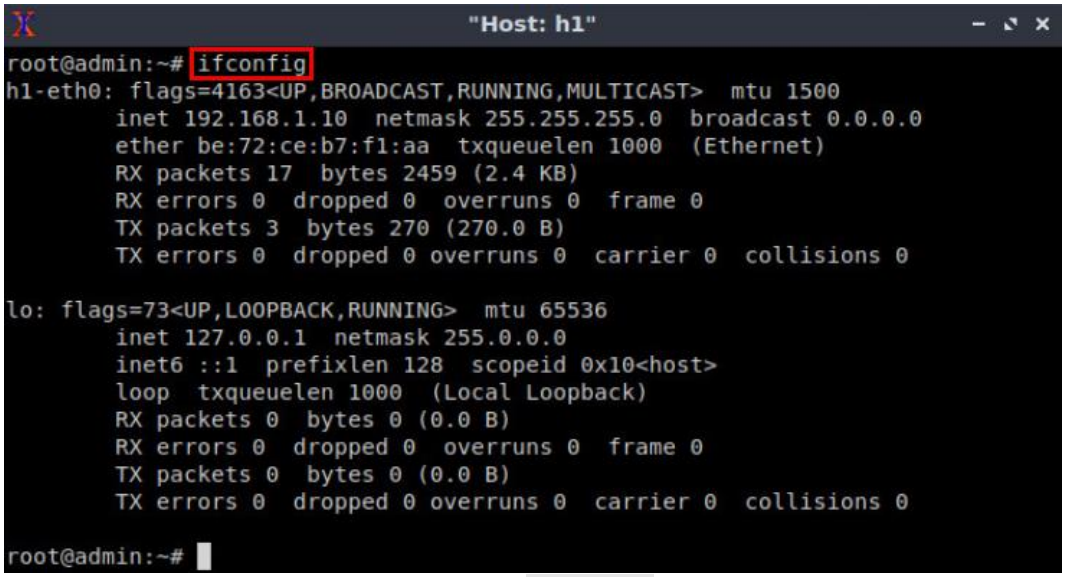

Figure 33. Output of `ifconfig` command.

Step 3. In host h1 terminal, type the command shown below to verify that the default gateway IP address is 192.168.1.1.

```
route
```

Figure 34. Output of `route` command.

Step 4. In order to verify host 2 default route, proceed similarly by repeating from step 1 to step 3 in host h2 terminal. Similar results should be observed.

4.2 Configure router's interface

Step 1. In order to configure router r1, hold right-click on router r1 and select *Terminal*.

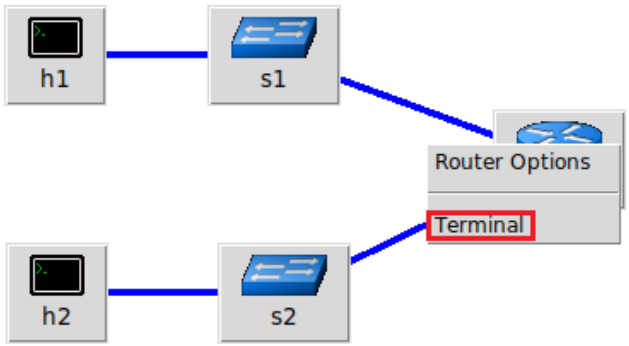

Figure 35. Opening a terminal on router r1.

Step 2. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you do not enable zebra daemon initially. In order to start the zebra, type the following command:

```
zebra
```

Figure 36. Starting zebra daemon.

Step 3. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command:

```
vtysh
```

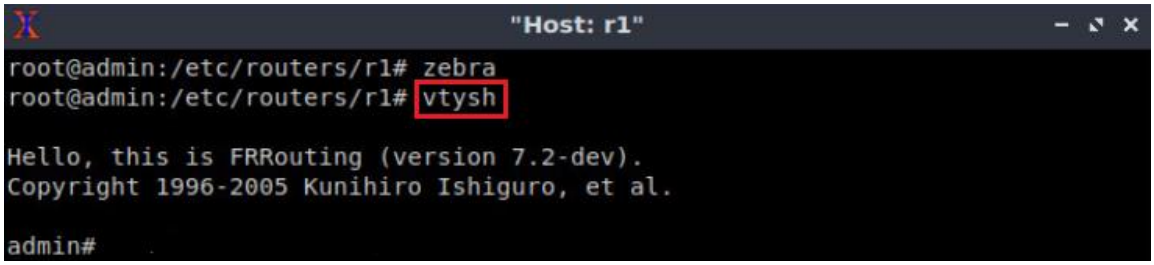

Figure 37. Starting vtysh on router r1.

Step 4. Type the following command in the router r1 terminal to enter in configuration mode.

```
configure terminal
```

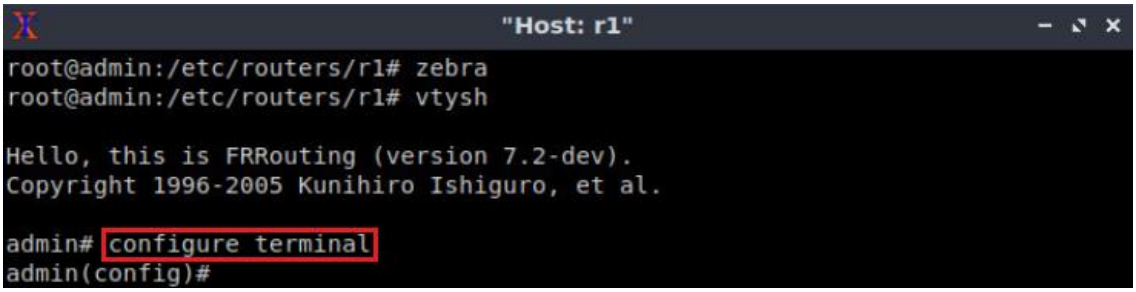

Figure 38. Entering in configuration mode.

Step 5. Type the following command in the router r1 terminal to configure interface *r1-eth0*.

```
interface r1-eth0
```

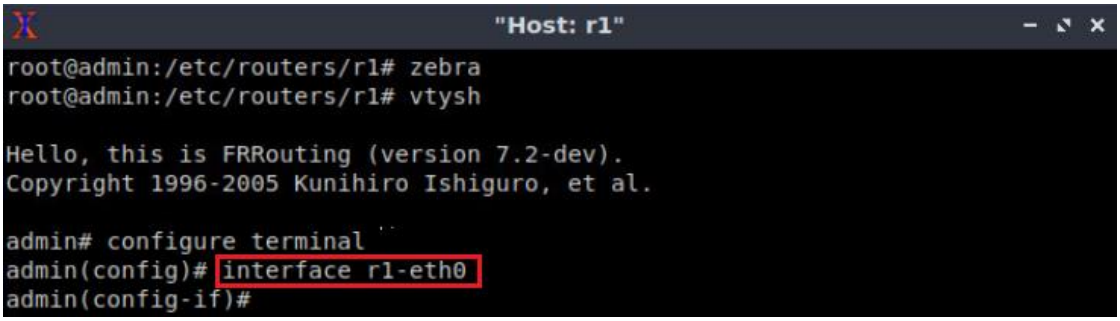

Figure 39. Configuring interface *r1-eth0*.

Step 6. Type the following command on router r1 terminal to configure the IP address of the interface *r1-eth0*.

```
ip address 192.168.1.1/24
```

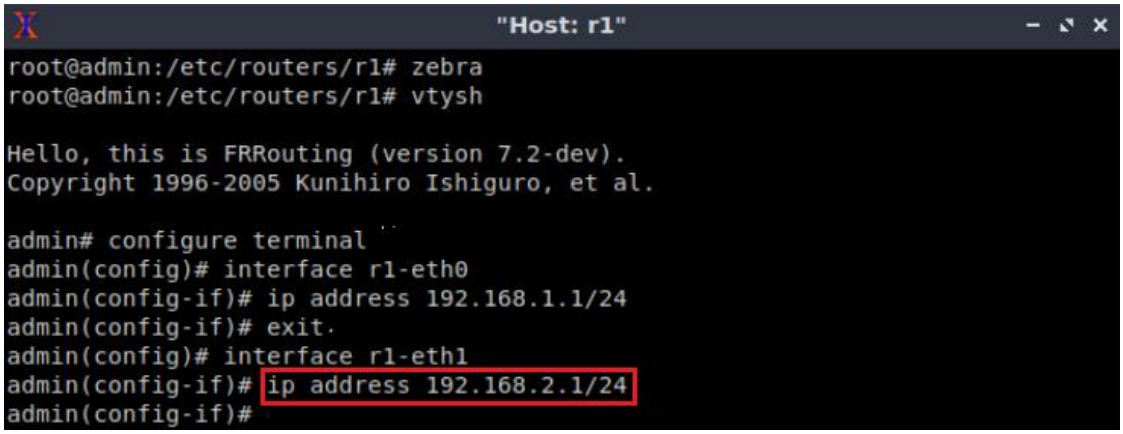

Figure 40. Configuring an IP address to interface *r1-eth0*.

Step 7. Type the following command exit from interface *r1-eth0* configuration.

```
exit
```

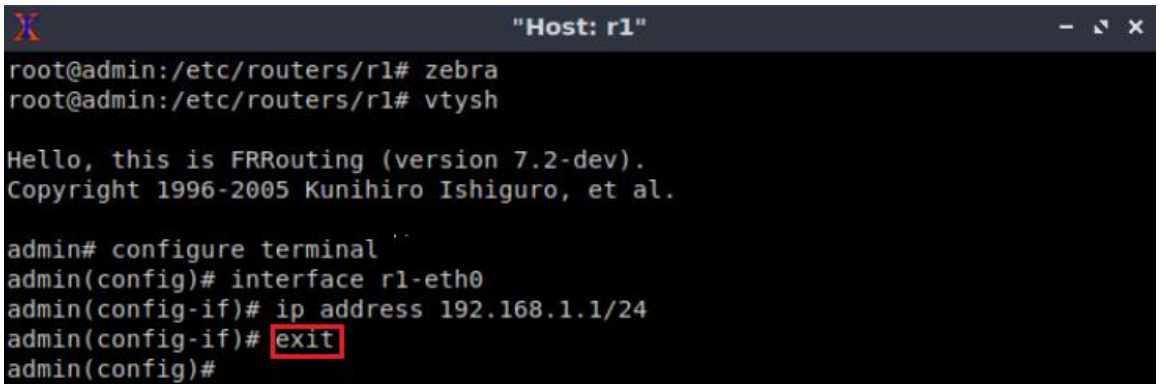

Figure 41. Exiting from configuring interface *r1-eth0*.

Step 8. Type the following command on router r1 terminal to configure the interface *r1-eth1*.

```
interface r1-eth1
```

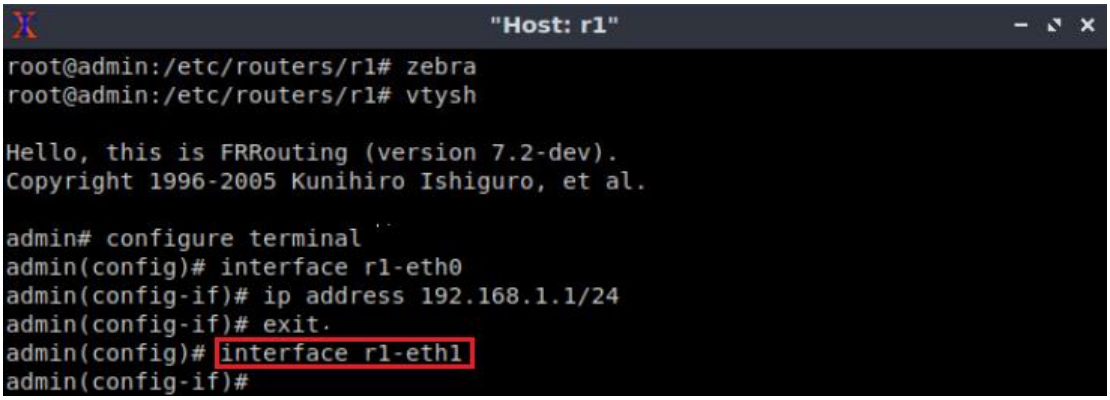

Figure 42. Configuring interface *r1-eth1*.

Step 9. Type the following command on router r1 terminal to configure the IP address of the interface *r1-eth1*.

```
ip address 192.168.2.1/24
```

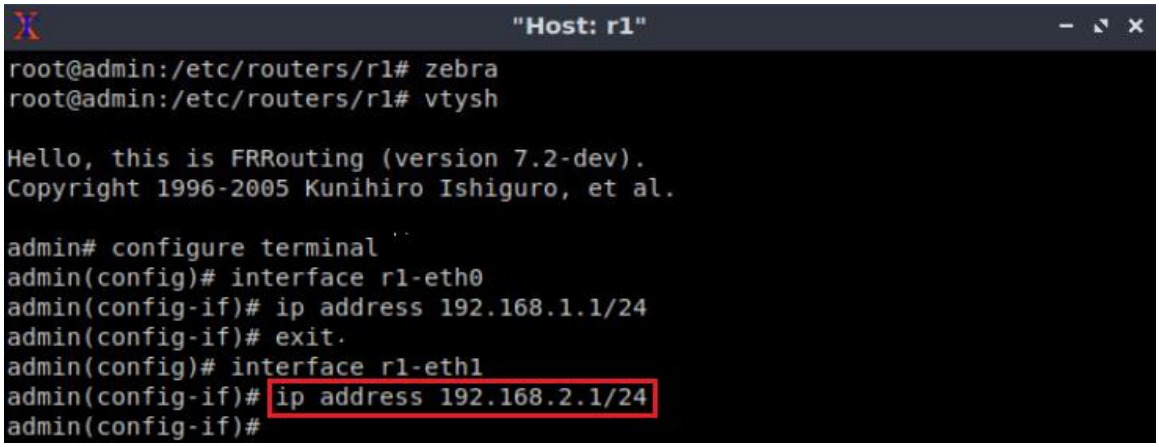

Figure 43. Configuring an IP address to interface *r1-eth1*.

Step 10. Type the following command to exit from *r1-eth1* interface configuration.

```
exit
```

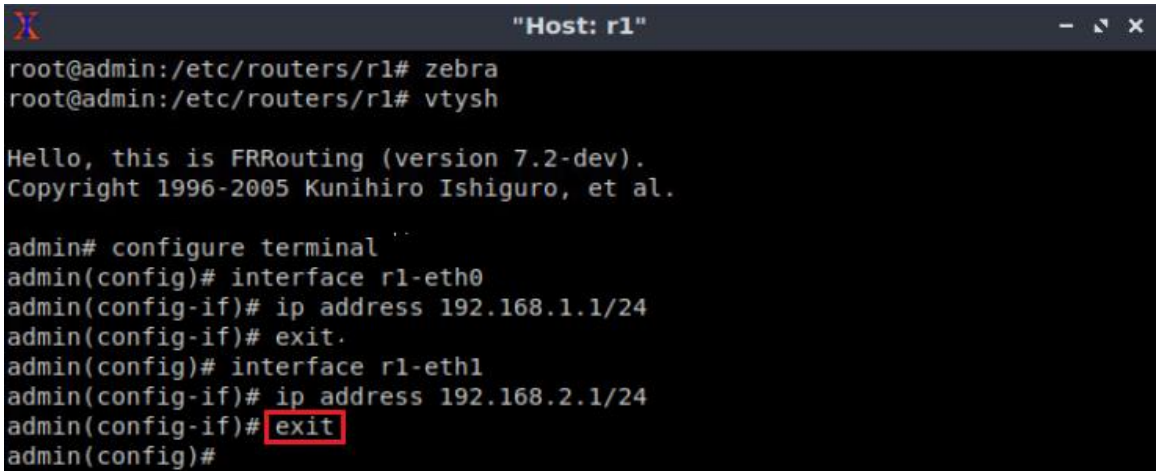

Figure 44. Exiting from configuring interface *r1-eth1*.

4.3 Verify router r1 configuration

Step 1. Exit from router r1 configuration mode issuing the following command:

```
exit
```

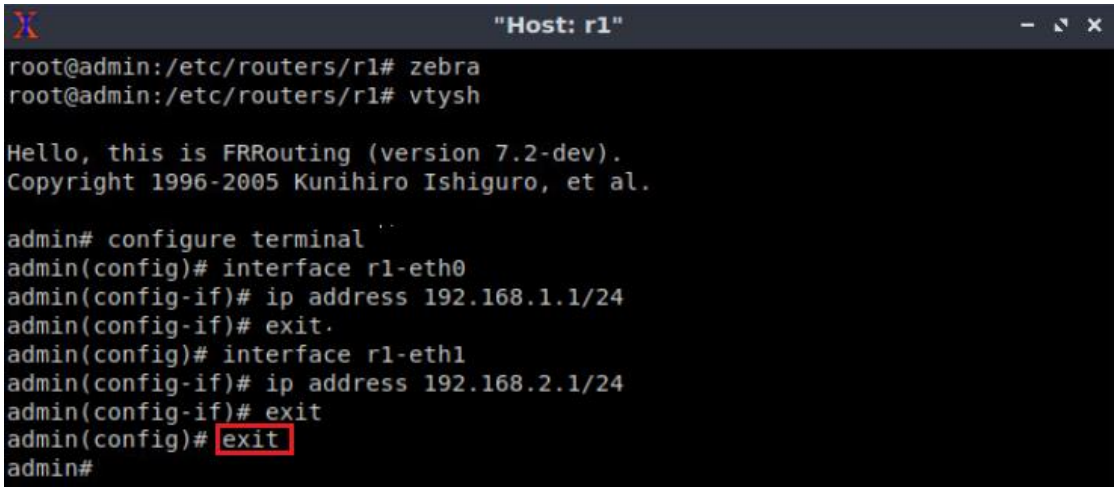

Figure 45. Exiting from configuration mode.

Step 2. Type the following command on router r1 terminal to verify the routing information of router r1. It will be showing all the directly connected networks.

```
show ip route
```

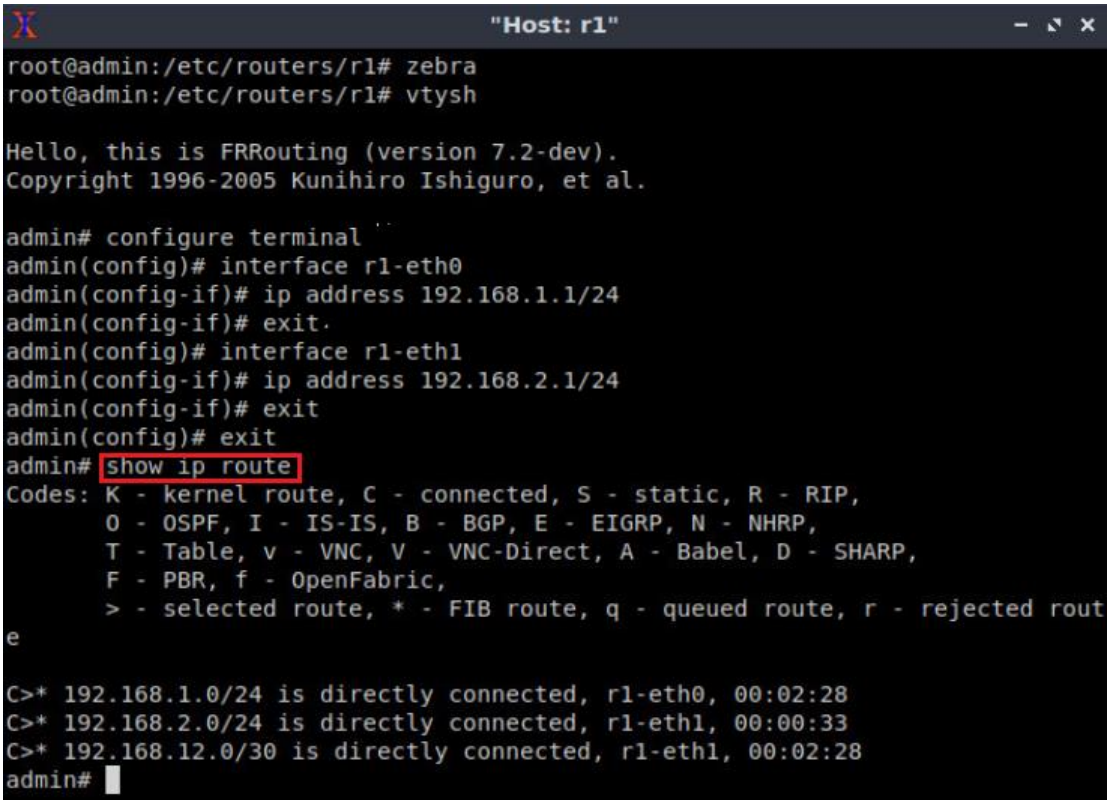

Figure 46. Displaying routing information of router r1.

4.4 Test connectivity between end-hosts

In this section you will run a connectivity test between host h1 and host h2.

Step 1. In host h1 terminal type the command shown below. Notice that according to Table 2, the IP address 192.168.2.10 is assigned to host h2. To stop the test press `ctrl+c`

```
ping 192.168.2.10
```

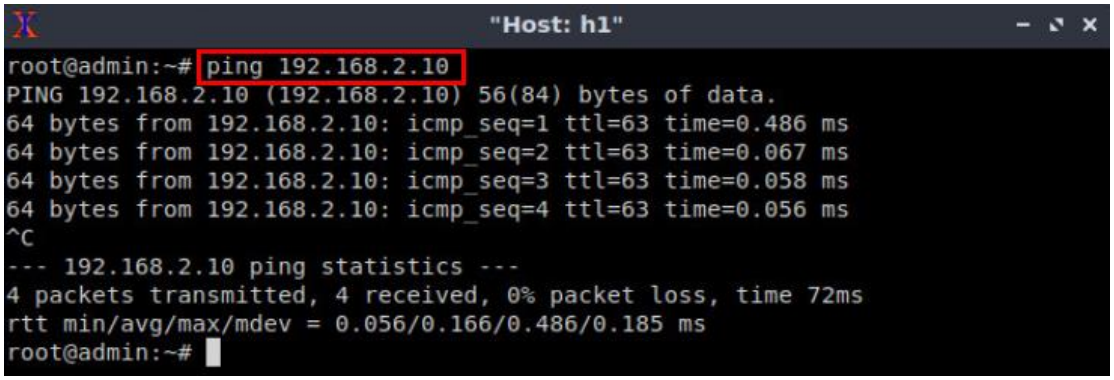

Figure 47. Connectivity test between host h1 and host h2.

This concludes Lab 1. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. Mininet walkthrough. [Online]. Available: http://Mininet.org.
2. N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "*OpenFlow,*" ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, p. 69, 2008.
3. J. Esch, *"Prolog to, software-defined networking: a comprehensive survey,"* Proceedings of the IEEE, vol. 103, no. 1, pp. 10–13, 2015.
4. P. Dordal, "*An Introduction to computer networks,*". [Online]. Available: https://intronetworks.cs.luc.edu/.
5. B. Lantz, G. Gee, "*MiniEdit: a simple network editor for Mininet,*" 2013. [Online]. Available: https://github.com/Mininet/Mininet/blob/master/examples.

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 2: Legacy Networks: BGP Example as a Distributed System and Autonomous Forwarding Decisions

Document Version: 05-27-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab is an introduction to legacy networks using Free Range Routing (FRR), which is a routing software suite that provides Transmission Control Protocol (TCP)/Internet Protocol (IP) based routing services with routing protocols support. In this lab, you will understand the main difference between legacy and Software Defined Networking (SDN) networks. Furthermore, you will explore FRR architecture, and load its basic configuration. Furthermore, this lab emulates a simple legacy network that runs Border Gateway Protocol (BGP) between two Autonomous Systems (ASes).

Objectives

By the end of this lab, the user will:

1. Understand the difference between legacy and SDN networks.
2. Understand the architecture of FRR.
3. Navigate through FRR terminal.
4. Explain the concept of BGP.
5. Configure and verify BGP between two ASes.
6. Perform a connectivity test between end hosts.

Lab settings

The information in Table 1 provides the credentials of the machine containing Mininet emulator.

Table 1. Credentials to access Client1 machine.

Device	Account	Password
Client1	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Configure BGP routing protocol.
4. Section 4: Verify connections.

1 Introduction

1.1 Traditional switch architecture

In a traditional switch architecture, the various switching functionalities are segregated into three separate categories, also called layers/planes. These layers can communicate horizontally, i.e., communicate with the same layer in a different switch. Additionally, the layers can communicate vertically, i.e., from one layer to another within the same switch[11].

Consider Figure 1. The vast majority of packets handled by the switch are only managed by the data plane. The latter is composed of ports used to receive and transmit the packets, as well as a forwarding table which instructs the switch on how to deal with incoming packets. The data plane is responsible for packet buffering, packet scheduling, header modification and forwarding[11].

Some packets can't be processed by the data plane directly, for example, their information is not yet inserted in the forwarding table. Such packets are forwarded to the control plane which lies on top of the data plane. The control plane involves in many activities, mainly, to maintain the forwarding table of the data plane. Essentially, the control plane is responsible for processing different control protocols that may affect the forwarding table[11].

The management plane lies on top of the control plane and it is used by network administrators to configure and monitor the switch. Thus, allowing them to extract information or modify data in the underlying planes (control and data planes) as appropriate[11].

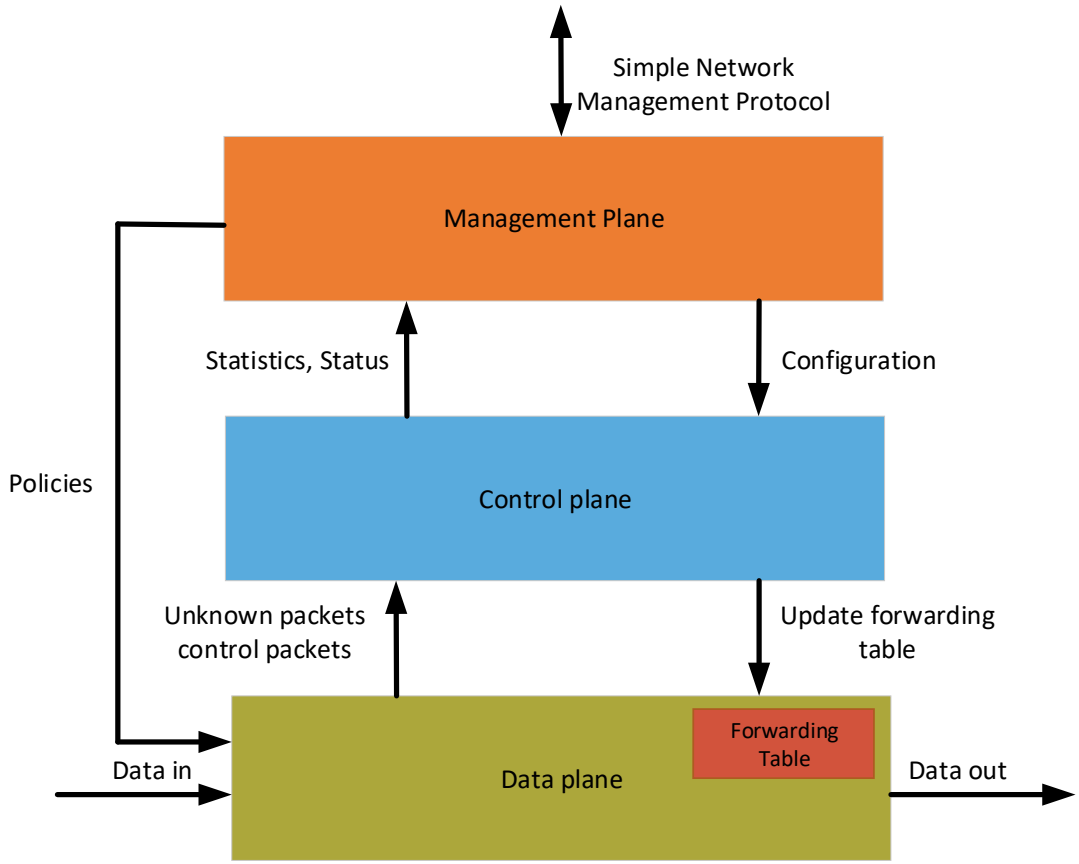

Figure 1. Roles of the control, data and management planes[3].

1.2 Legacy and SDN networks

In a legacy network, the data, control, and the management layers are aggregated into the same device, usually referred to as a router. When a packet arrives to a router, the latter checks if the packet should be forwarded out one of its interfaces or if the packet needs further processing. The decision is made based on the routing table of the router, which consists of multiple entries, each maps a network/IP prefix to a next hop. The routing table is built primarily through the use routing protocols. The latter specifies how routers communicate with each other to distribute information that enables them to select routes between any two nodes on a computer network. Routing protocols include Routing Information Protocol[2] (RIP), Open Shortest Path First[3] (OSPF), BGP[4] and Intermediate System to Intermediate System (IS-IS)[5].

Consider Figure 2. A legacy network consists of several connected devices. Each device runs a local algorithm in the control plane and inserts forwarding rules in the routing table of the data plane.

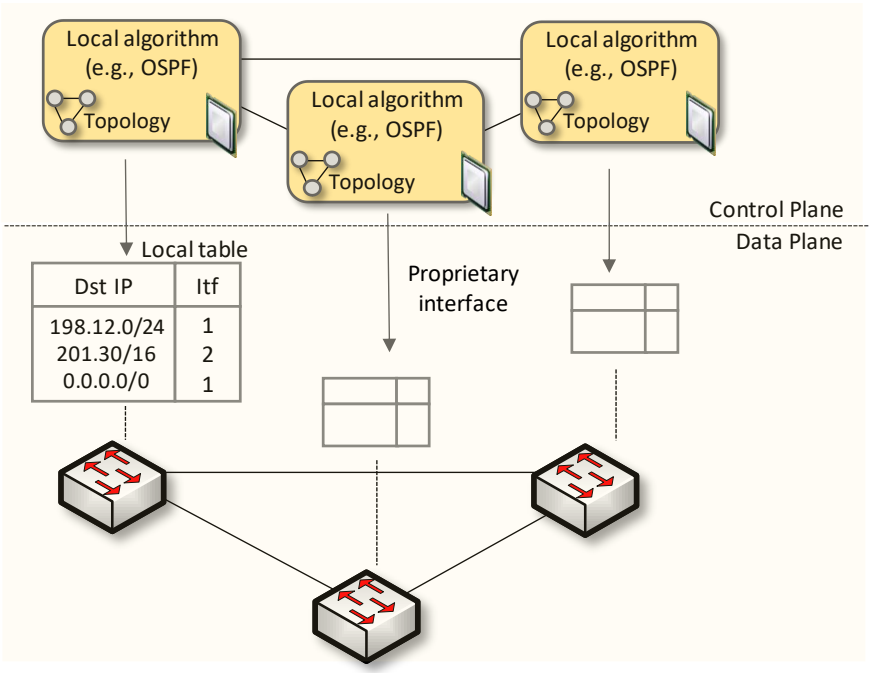

Figure 2. The control plane and the data plane are coupled in the same legacy device.

The control protocols, such as OSPF, are distributed protocols. When something in the network is changed (for example, a link is down) these distributed protocols take time to converge and insert new consistent forwarding rules. Although the mechanism of the routing protocols was essential to respond to rapidly changing network conditions, these conditions no longer exist in modern data centers. Thus, the behavior of the routing protocols wreaks temporary havoc inside the data center and hinders the ability to process large data traffic[11].

SDN is a new paradigm that solves the aforementioned problem by creating a centralized approach, rather than a distributed one. The main concept of SDN is to separate the control plane from the data plane in order to maximize the efficiency of the data plane devices. Moving the control software off the device into a centralized server makes it capable of seeing the entire network and making decisions that are optimal given a complete understanding of the situation[11].

Consider Figure 3. The control plane is decoupled from the data plane. The former is moved into a centrally located computer resource and it controls data plane devices by pushing rules into their tables[11].

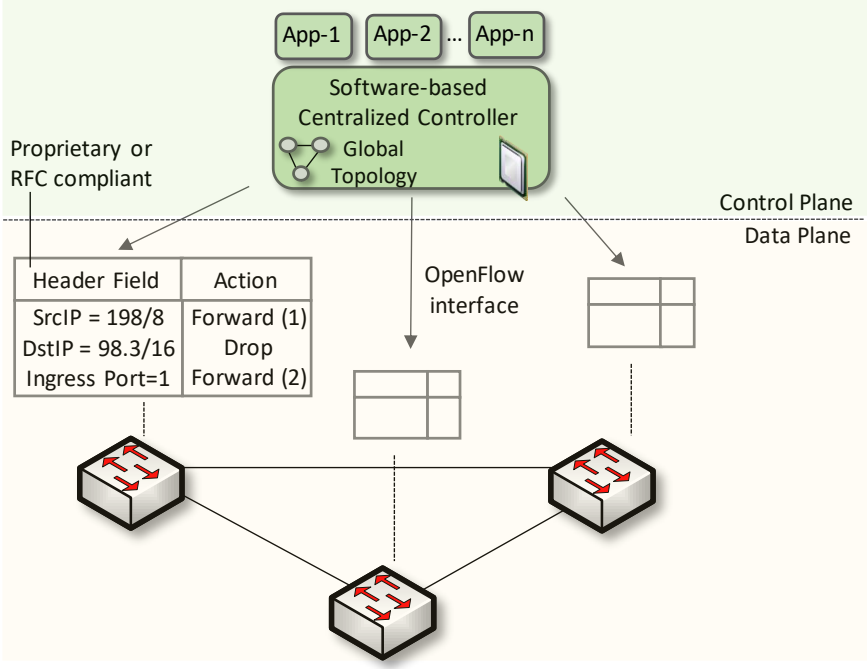

Figure 3. The control plane is embedded in a centralized server and it is decoupled from data plane devices.

This lab solely focuses on understanding how legacy networks work. You will configure BGP on legacy routers and inspect the inserted rules on each router's forwarding table. In order to do the configuration in an emulated environment, FRR will be used, which is an open source software that allows to configure the routers with a list of supported routing protocols.

1.3 Introduction to FRR

Implementing IP routing usually involves buying expensive and vertically integrated equipment from specific companies. This approach has limitation such as the cost of the hardware, closed source software and the training required to operate and configure the devices. Networking professionals, operators and researchers sometimes are limited by the capabilities of such routing products. Moreover, combining routing functionalities with existing open source software packages is usually constrained by the number of separate devices that can be deployed.

For example, operators could be interested in collecting some information about the behavior of routing devices, process them, and make them available. Therefore, in order to achieve such capabilities, additional storage and scripting capacities are required. Such resources are not available in existing routing products. On the other hand, researchers may be interested on developing routing protocols by extending an existing one without writing a complete implementation from scratch.

FRR suite[1] is a package of Unix/Linux software that implements common network routing protocols, such as RIP[2], OSPF[3], BGP[4] and IS-IS[5]. The package also includes a routing information management process, to act as intermediary between the various routing

protocols and the active routes installed with the kernel. A library provides support for configuration and an interactive command-line interface. The routing protocols supported by FRR, can be extended to enable experimentation, logging, or custom processing. In addition, libraries and kernel daemon provide a framework to facilitate the development of new routing protocol daemons. A wide range of functionalities can be attained by combining other software packages to allow the integration into a single device as well as enabling innovative solutions to networking problems.

1.4 FRR architecture

FRR takes a different approach compared to traditional routing software which, consists of a single process program that provides all the routing protocol functionalities. FRR is composed by a suite of daemons that work together to build a routing table. Each routing protocol is implemented in its own daemon. These daemons exchange information through another daemon called zebra, which is responsible for encompassing routing decisions and managing the data plane.

Since all the protocols are running independently, this architecture provides high resiliency, that means that an error, crash or exploit in one protocol daemon will generally not affect the other protocols. It is also flexible and extensible since the modularity makes it easy to implement new protocols and append them to the suite[1]. Additionally, each daemon implements a plugin system allowing new functionality to be loaded at runtime.

Figure 4 illustrates FRR architecture. It consists of a set of processes communicating via Inter-process Communication (IPC) protocol. This protocol refers to the mechanism provided by an operating system (OS) to allow the management of shared data between different processes. Network routing protocols such as BGP, OSPF and IS-IS are implemented in processes such as *bgpd*, ripd, *ospfd*, *ldpd, etc*. These processes are daemons that implement routing protocols e.g., the BGP daemon is implemented by the *bgpd* process*,* the RIP daemon is implemented by the *ripd* process and so on. Another daemon, called *zebra*, acts as an intermediary between the kernel's forwarding plane and the routing protocol processes. Additionally, an interactive command-line tool called *vtysh* allows these processes to be monitored and configured. The *vtysh* command-line tool communicates with other processes via a simple string passing protocol, where the strings are essentially identical to the commands entered.

The *zebra* process is a fundamental part of FRR architecture. Its purpose is to maintain a backup of packet forwarding state, such as the network interfaces and the table of currently active routes. The currently active routes are also referred to as the Forwarding Information Base (FIB) [6]. Usually, the kernel manages packet forwarding therefore, kernel maintains these. The *zebra* process also collects routing information from the routing protocol processes and stores these, together with its shadow copy of the FIB, in its own Routing Information Base (RIB)[6] whereas, static routes are also configured. The *zebra* process then is responsible for selecting the best route from all those available for a destination and updating the FIB[7]. Additionally, information about the current best routes may be distributed to the protocol daemons. The *zebra* process maintains the routing daemons updated if any change occurs in the network interface state.

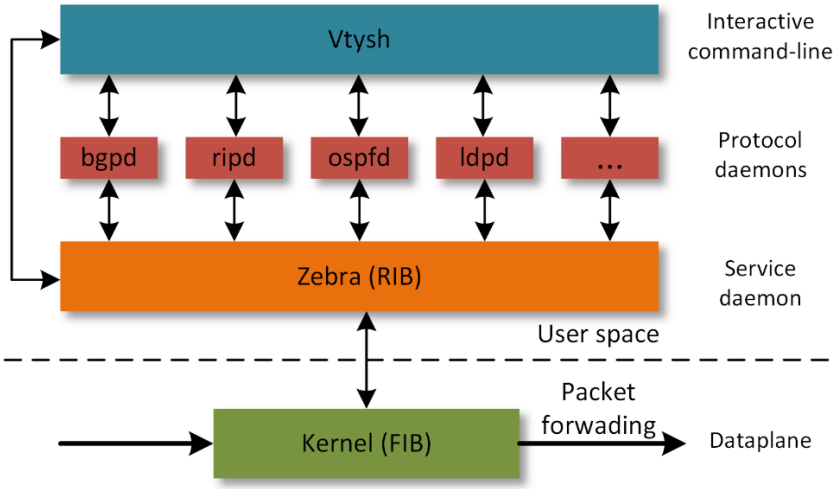

Figure 4. FRR architecture.

1.5 FRR and Mininet integration

Mininet is a network emulator which runs collection of end-hosts, switches, routers and links on a single Linux kernel[9]. Mininet provides network emulation, allowing all network software at any layer to be simply run *as is*, i.e. nodes run the native network software of the physical machine. Hence, the set of commands provided by FRR are inherited and can be run using Mininet's command-line interface. This feature allows the user to run and configure FRR in the emulated routers. FRR is production-ready but we are using it in an emulated environment.

1.6 Introduction to BGP

The Internet can be viewed as a collection of networks or ASes that are interconnected. An AS refers to a group of connected networks under the control of a single administrative entity or domain[8].

BGP is an exterior gateway protocol designed to exchange routing and reachability information among ASes on the Internet. BGP is relevant to network administrators of large organizations which connect to one or more Internet Service Providers (ISPs), as well as to ISPs who connect to other network providers. In terms of BGP, an AS is referred to as a routing domain, where all networked systems operate common routing protocols and are under the control of a single administration[8].

Two routers that establish a BGP connection are referred to as BGP peers or neighbors. BGP sessions run over TCP. If a BGP session is established between two neighbors in different ASes, the session is referred to as an External BGP (EBGP) session. If the session is established between two neighbors in the same AS, the session is referred to as Internal (IBGP)[1]. Figure 5 shows a network running BGP protocol. Routers that exchange information within the same AS use Internal BGP (IBGP), while routers that exchange information between different ASes use EBGP.

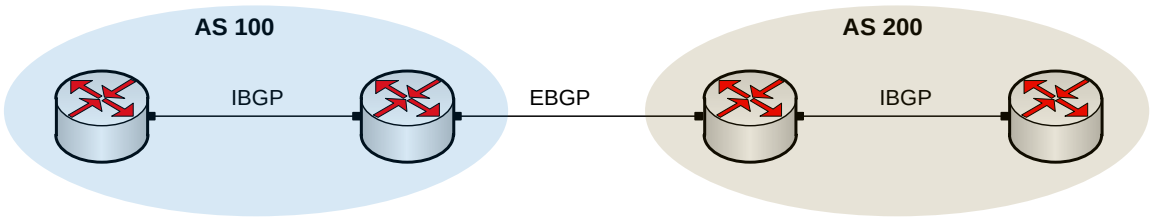

Figure 5. Routers that exchange information within the same AS use IBGP, while routers that exchange information between different ASes use EBGP.

2 Lab topology

Consider Figure 6. The topology consists of two networks, Network 1 and Network 2, each in an AS. Both networks have the following elements: a router to specify the network, a switch that defines a Local Area Network (LAN) and lastly, a host aimed to test end-to-end connectivity. The Autonomous System Numbers (ASNs) assigned to routers r1 and r2 are 100 and 200 respectively. Routers r1 and r2 exchange routing information via EBGP.

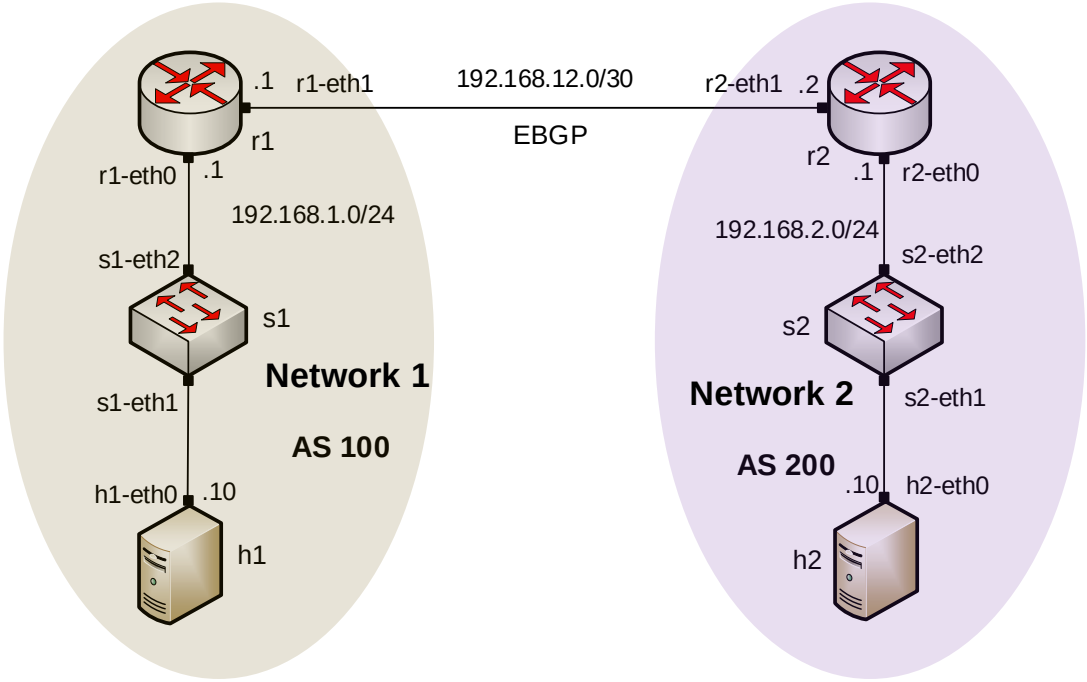

Figure 6. Lab topology.

2.1 Lab settings

Routers and hosts are already configured according to the IP addresses shown in Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet	Default gateway
	r1-eth0	192.168.1.1	/24	N/A

Router r1	r1-eth1	192.168.12.1	/30	N/A
Router r2	r2-eth0	192.168.2.1	/24	N/A
	r2-eth1	192.168.12.2	/30	N/A
Host h1	h1-eth0	192.168.1.10	/24	192.168.1.1
Host h2	h2-eth0	192.168.2.10	/24	192.168.2.1

2.2 Open the topology

In this section, you will open MiniEdit[10] and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), save the topology and export a layer 2 model.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

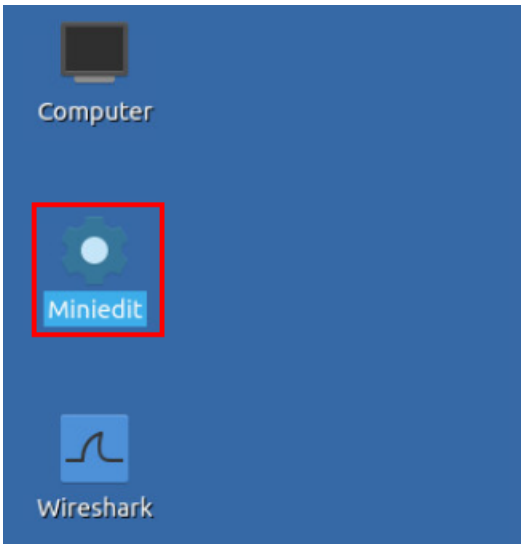

Figure 7. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab2.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab2* and click on *Open*.

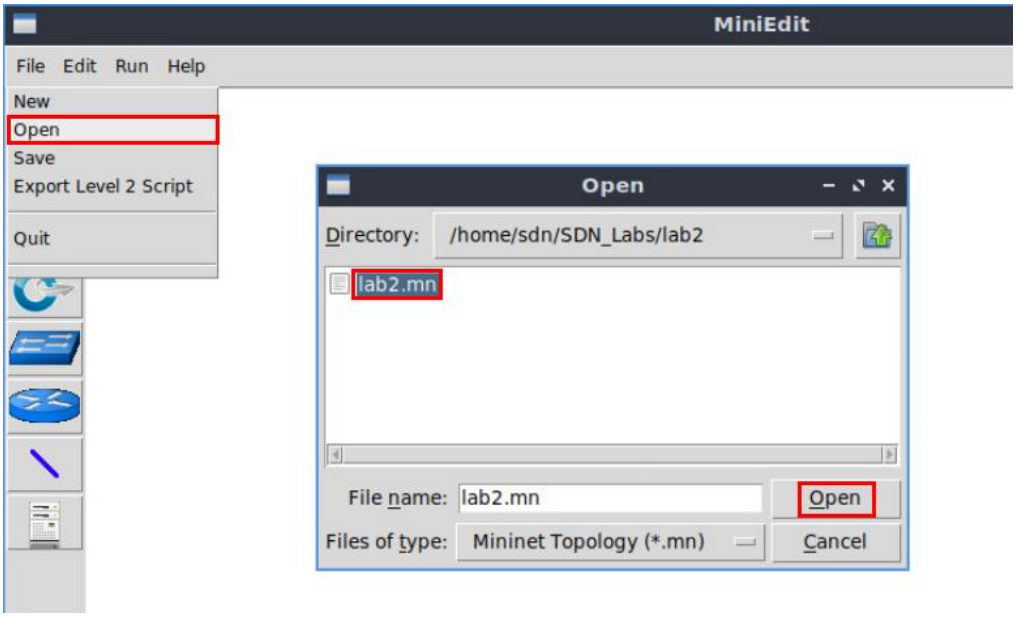

Figure 8. MiniEdit's open dialog.

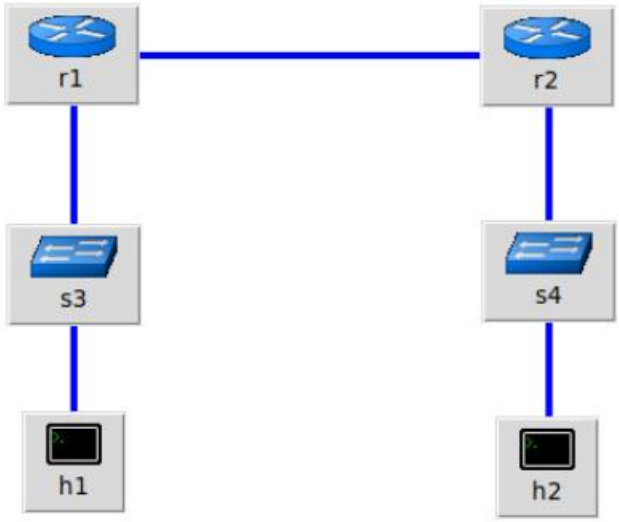

Figure 9. Mininet's topology.

2.3 Load the configuration file

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open Linux terminal.

Figure 10. Opening Linux terminal.

Step 2. Navigate into *SDN_Labs/lab2* directory by issuing the following command. This folder contains a configuration file and the script responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab2
```

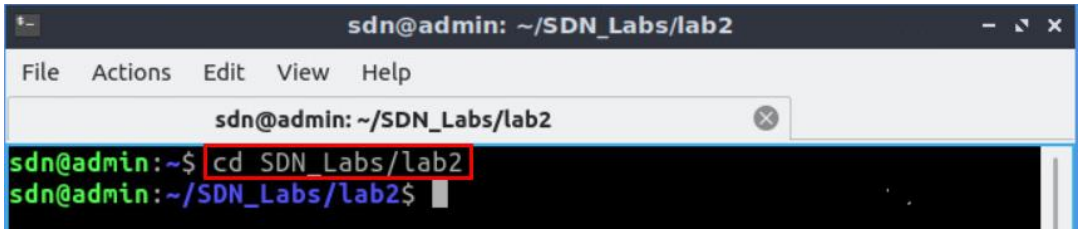

Figure 11. Entering the *SDN_Labs/lab2* directory.

Step 3. To execute the shell script, type the following command. The argument of the program corresponds to the configuration zip file that will be loaded in all the routers in the topology.

```
./config_loader.sh lab2_conf.zip
```

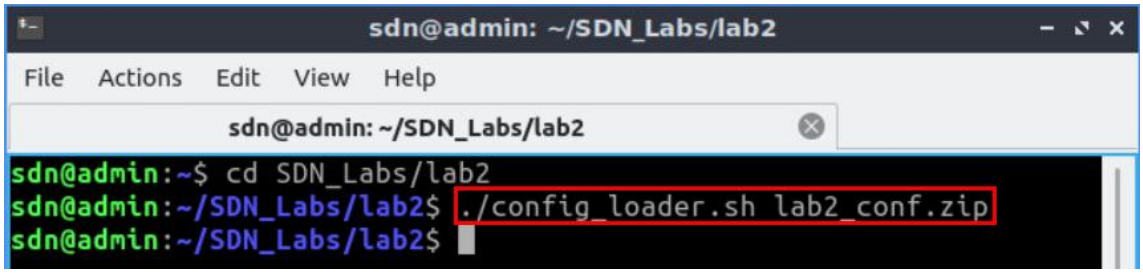

Figure 12. Executing the shell script to load the configuration.

Step 4. Type the following command to exit the Linux terminal.

```
exit
```

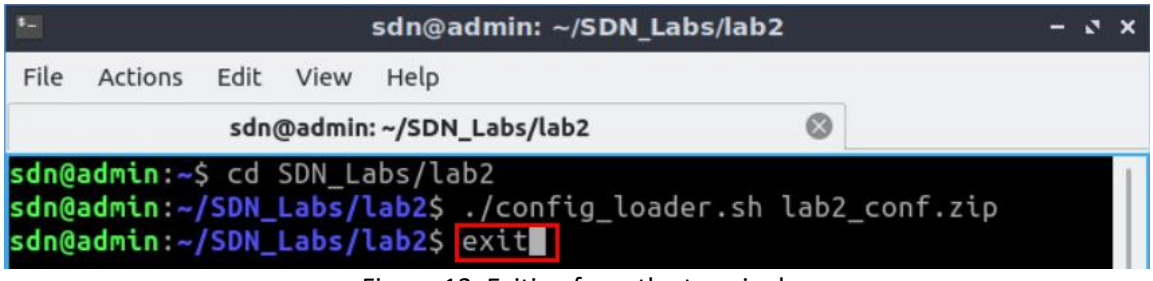

Figure 13. Exiting from the terminal.

2.4 Run the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. At this point host h1 and host h2 interfaces are configured. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

Figure 14. Starting the emulation.

Step 2. Issue the following command to display the interface names and connections.

```
links
```

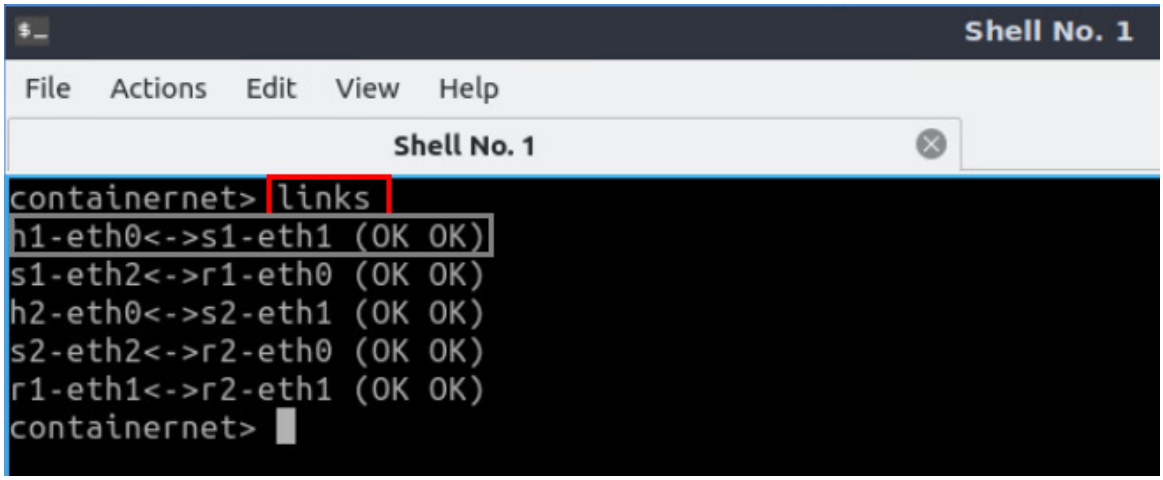

Figure 15. Displaying network interfaces.

In Figure 15, the link displayed within the gray box indicates that interface *eth2* of switch s1 connects to interface *eth0* of host h1 (i.e., *s1-eth2<->h1-eth0*).

2.5 Verify the configuration

You will verify the IP addresses listed in Table 2 and inspect the routing table of routers r1 and r2.

Step 1. Hold right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on that host.

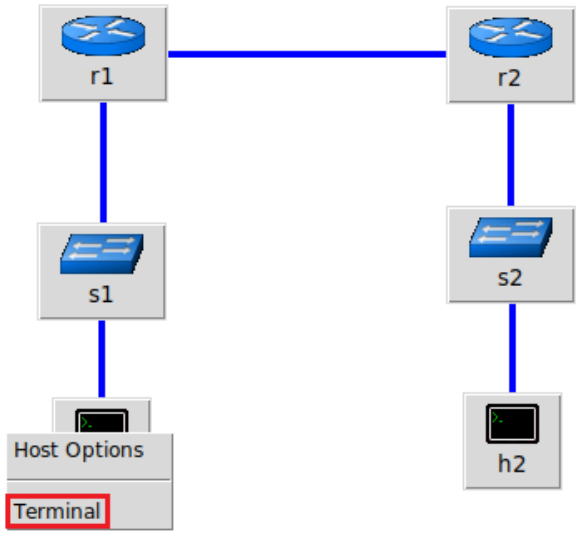

Figure 16. Opening a terminal on host h1.

Step 2. On host h1 terminal, type the command shown below to verify that the IP address was assigned successfully. You will corroborate that host h1 has two interfaces, *h1-eth0* configured with the IP address 192.168.1.10 and the subnet mask 255.255.255.0.

```
ifconfig
```

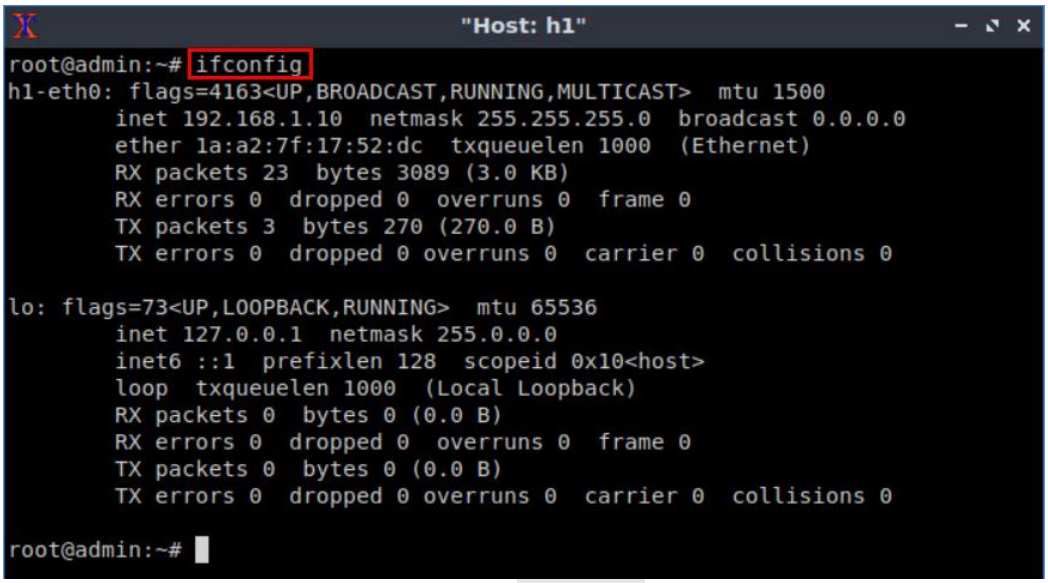

Figure 17. Output of `ifconfig` command.

Step 3. On host h1 terminal, type the command shown below to verify that the default gateway IP address is 192.168.1.1.

```
route
```

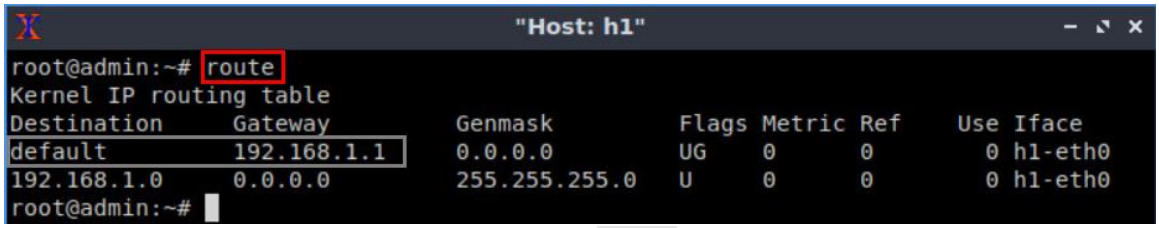

Figure 18. Output of `route` command.

Step 4. In order to verify host h2 default route, proceed similarly by repeating from step 1 to step 3 on host h2 terminal. Similar results should be observed.

Step 5. In order to verify router r1, hold right-click on router r1 and select *Terminal*.

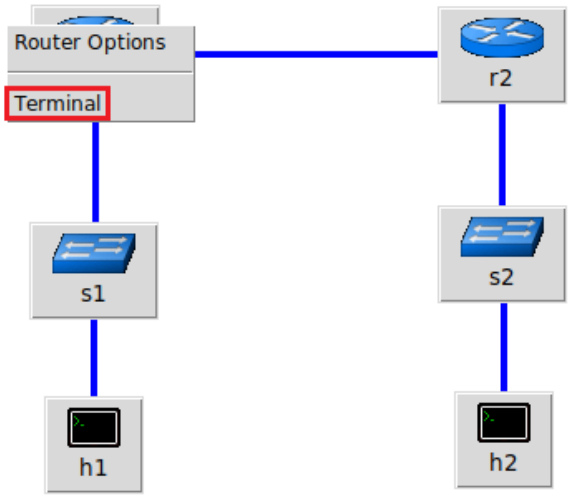

Figure 19. Opening a terminal on router r1.

Step 6. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you do not enable zebra daemon initially. In order to start the zebra, type the following command:

```
zebra
```

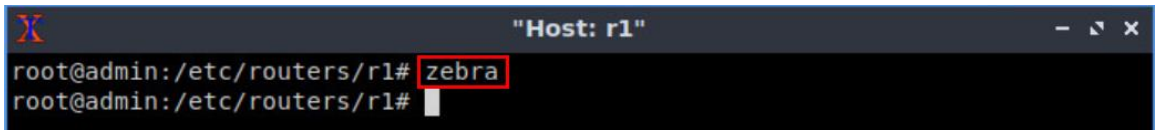

Figure 20. Starting zebra daemon.

Step 7. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command:

```
vtysh
```

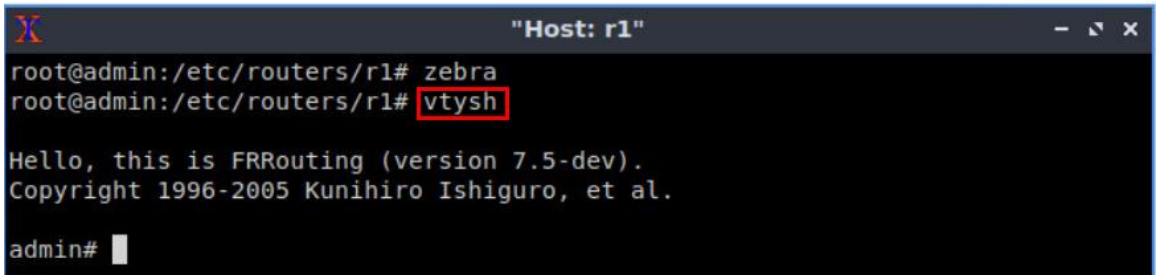

Figure 21. Starting vtysh on router r1.

Step 8. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 does not contain any route to the network of router r2 (192.168.2.0/24) as there is no routing protocol configured yet.

```
show ip route
```

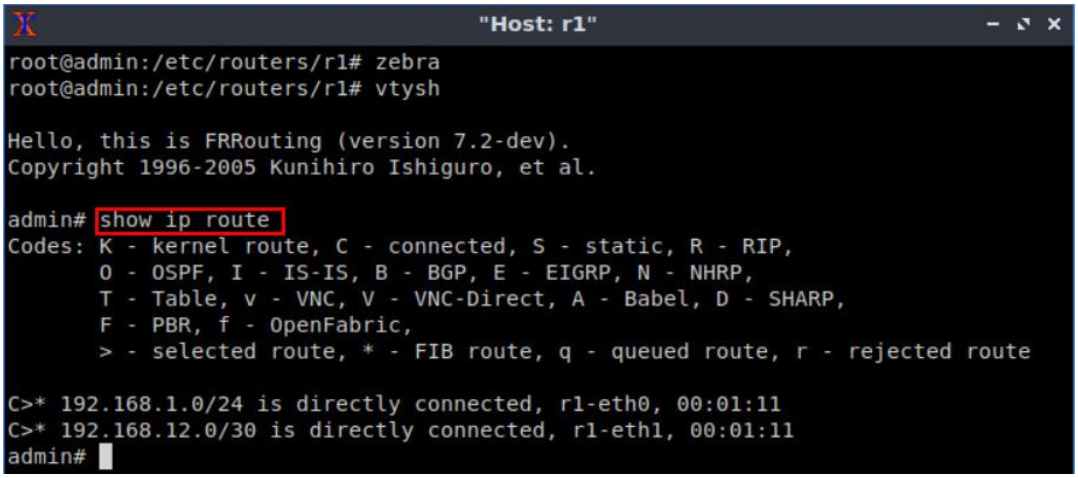

Figure 22. Displaying routing table of router r1.

The output in the figure above shows that the network 192.168.1.0/24 is directly connected through the interface *r1-eth0*. The network 192.168.12.0/30 is connected via the interface *r1-eth1*.

Step 9. Router r2 is configured similarly to router r1 but, with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r2 terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r2.

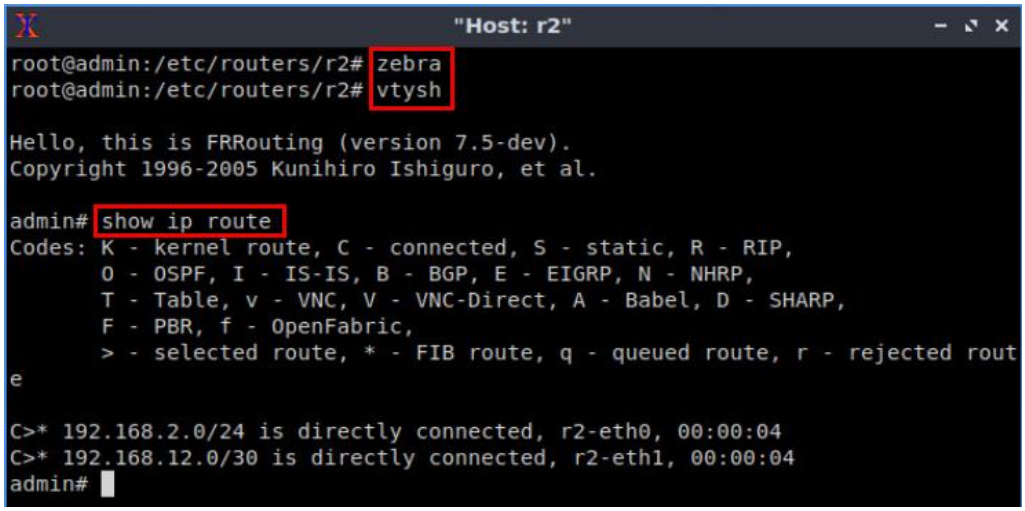

Figure 23. Displaying routing table of router r2.

2.6 Test connectivity between end-hosts

In this section you will run a connectivity test between host 1 and host 2. You will notice that there is no connectivity because there is no routing protocol configured in the routers.

Step 1. In host h1 terminal, type the command shown below. Notice that according to Table 1, the IP address 192.168.2.10 is assigned to host h2. To stop the test press `ctrl+c`.

```
ping 192.168.2.10
```

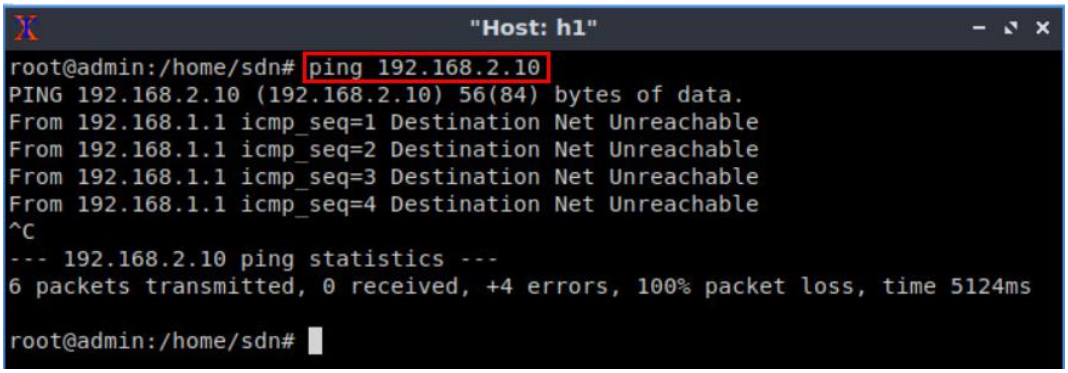

Figure 24. Connectivity test between host h1 and host h2.

3 Configure BGP routing protocol

In the previous section you used a script to assign the IP addresses to all devices' interfaces then, you performed an unsuccessful connectivity test. In this section you will configure a routing protocol in order to establish a connection between the two networks. You will configure BGP in order to establish a connection between AS 100 and AS 200. First, you will initialize the daemon that enables BGP configuration then. Then, you need to assign BGP neighbors to allow BGP peering to the remote neighbor. Additionally, you will advertise the local networks of each router.

3.1 BGP neighbors on the routers

In this section, you will add the neighbor IP address to allow BGP peering to the remote neighbor.

Step 1. To configure BGP routing protocol, you need to enable the BGP daemon first. In router r1, type the following command to exit the vtysh session:

```
exit
```

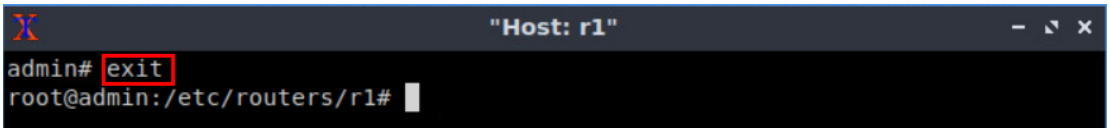

Figure 25. Exiting the vtysh session.

Step 2. Type the following command on router r1 terminal to start BGP routing protocol.

```
bgpd
```

Figure 26. Starting BGP daemon.

Step 3. In order to enter to router r1 terminal, type the following command:

```
vtysh
```

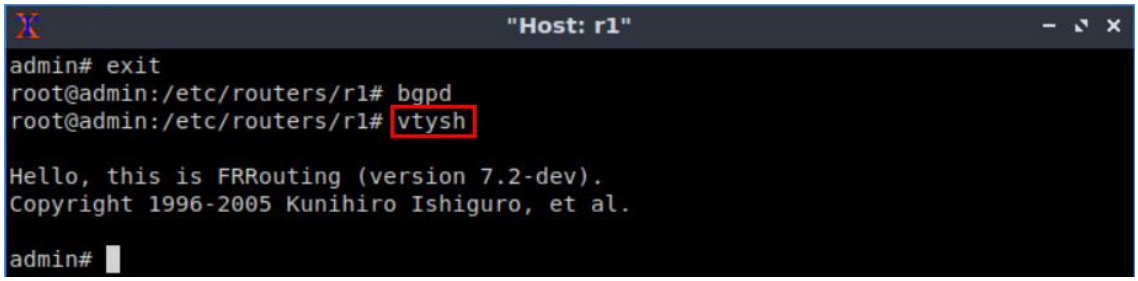

Figure 27. Starting vtysh on router r1.

Step 4. To enable router r1 configuration mode, issue the following command:

```
configure terminal
```

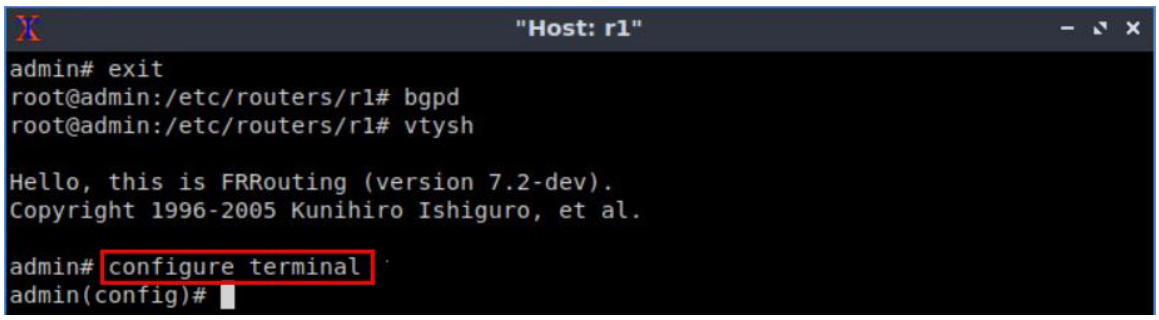

Figure 28. Enabling configuration mode on router r1.

Step 5. The ASN assigned for router r1 is 100. In order to configure BGP, type the following command:

```
router bgp 100
```

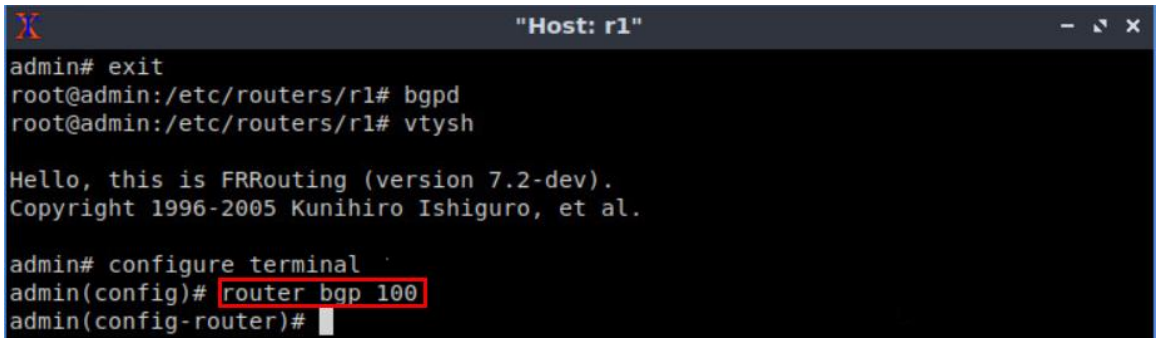

Figure 29. Configuring BGP on router r1.

Step 6. To configure a BGP neighbor to router r1 (AS 100), type the command shown below. This command specifies the neighbor IP address (192.168.12.2) and ASN of the remote BGP peer (AS 200).

```
neighbor 192.168.12.2 remote-as 200
```

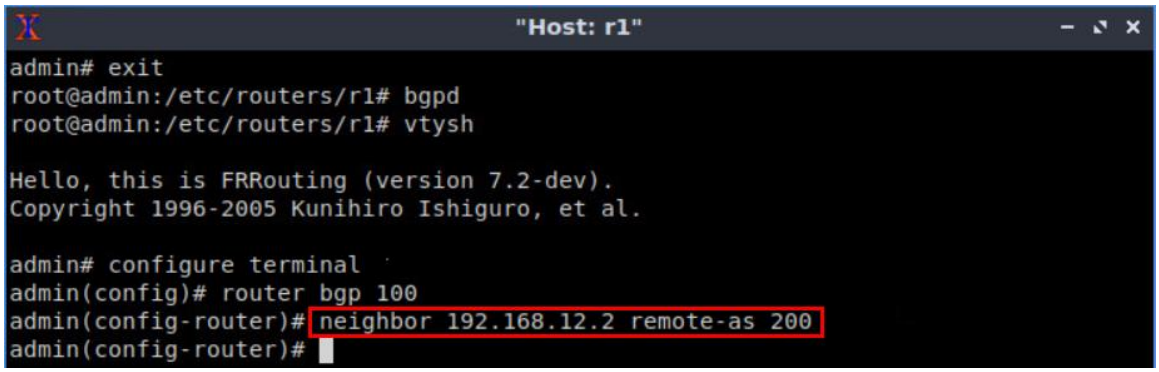

Figure 30. Assigning BGP neighbor to router r1.

Step 7. Type the following command to exit from the configuration mode.

```
end
```

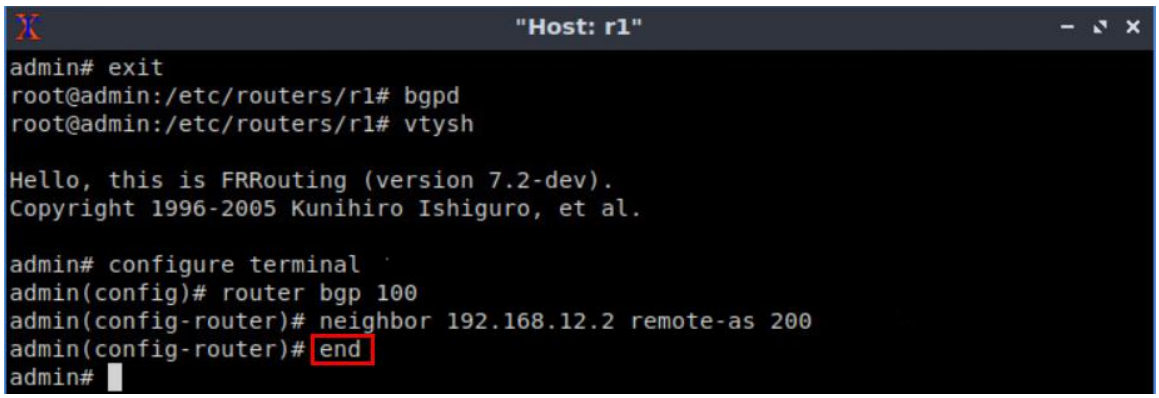

Figure 31. Exiting from configuration mode.

Step 8. Type the following command to verify BGP neighbors. You will verify that the neighbor IP address is 192.168.12.2. The corresponding ASN is 200.

```
show ip bgp neighbors
```

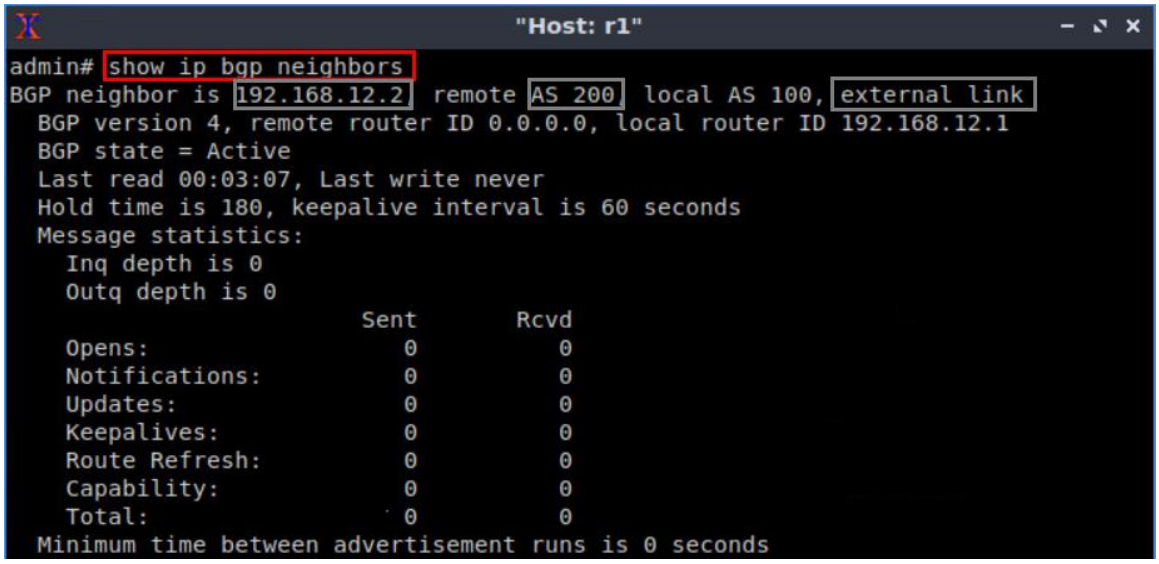

Figure 32. Verifying BGP neighbors on router r1.

Step 9. Router r2 is configured similarly to router r1 but, with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r2 terminal, issue the commands depicted below. At the end, you will verify all the directly connected networks of router r2.

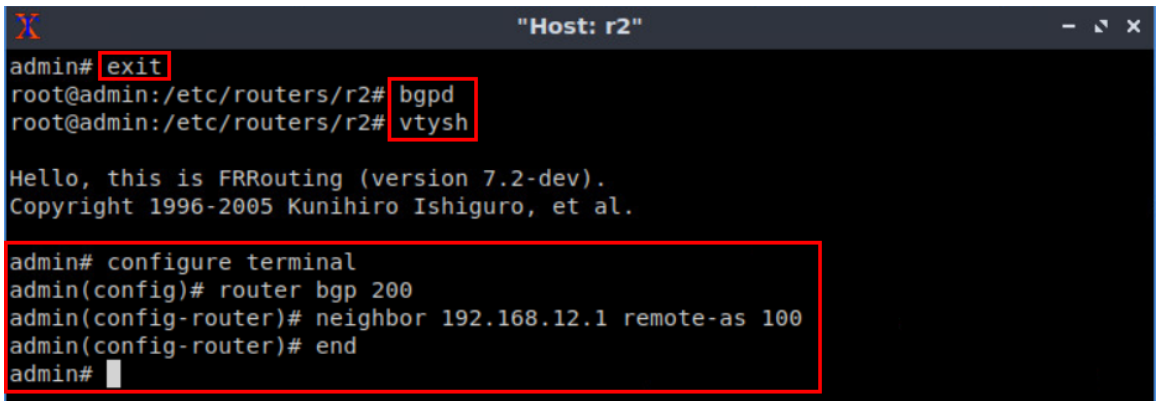

Figure 33. Assigning BGP neighbor to router r2.

Step 10. Type the following command to verify BGP neighbors. You will verify that the neighbor IP address is 192.168.12.1. The corresponding ASN is 100.

```
show ip bgp neighbors
```

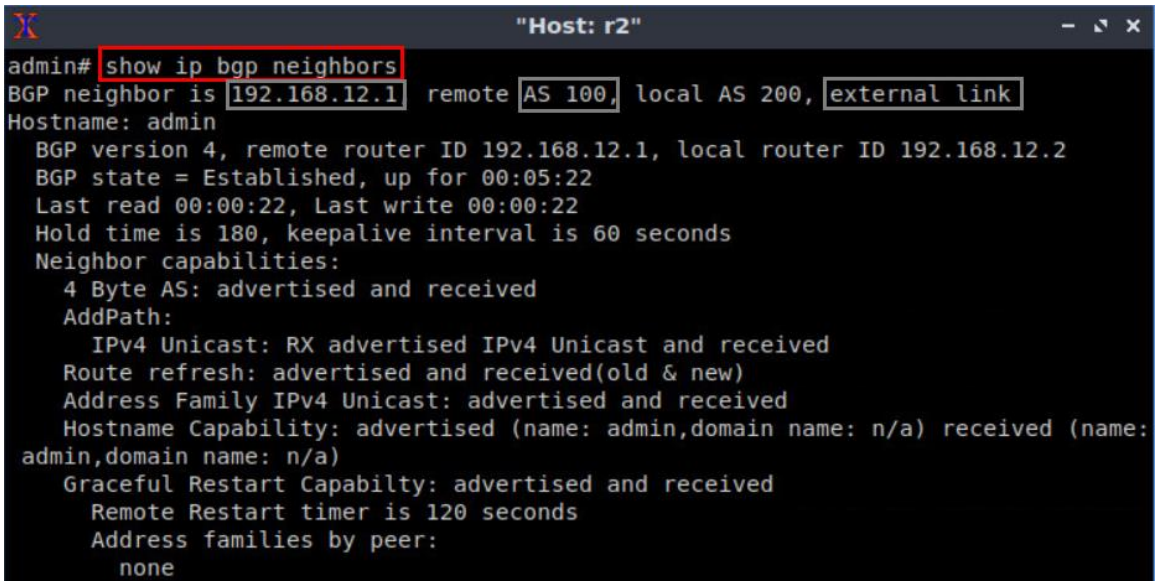

Figure 34. Verifying BGP neighbors on router r2.

Step 11. In router r2 terminal, perform a connectivity test by running the command shown below. To stop the test, press `Ctrl+c`. The result will show a successful connectivity test between router r1 and router r2.

```
ping 192.168.12.1
```

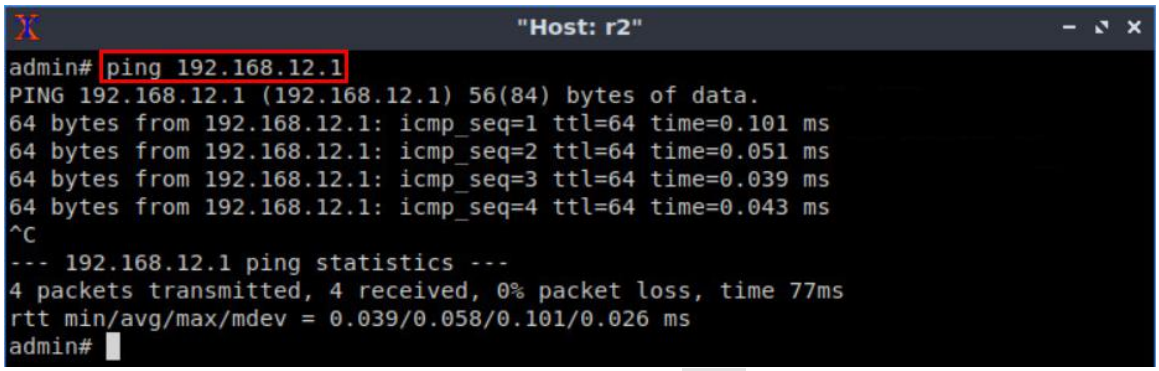

Figure 35. Connectivity test using `ping` command.

Step 12. In router r2 terminal, perform a connectivity between router r2 and host h1 by issuing the command shown below. To stop the test, press `Ctrl+c`. Router r2 cannot reach to host h1 at this point as the routing table of router r2 does not contain the network address of host h1.

```
ping 192.168.1.10
```

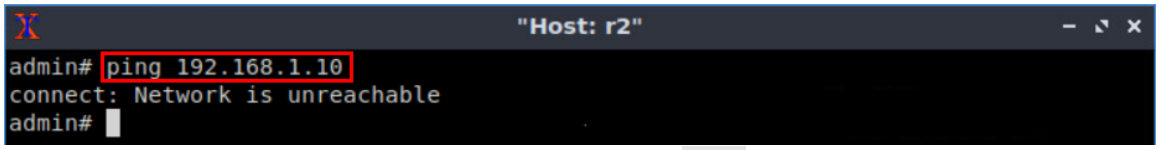

Figure 36. Connectivity test using `ping` command.

3.2 Advertise local networks on the routers

In this section, you will advertise a LAN so that the neighbor can receive the network address through EBGP.

Step 1. In router r1 terminal, issue the following command.

```
configure terminal
```

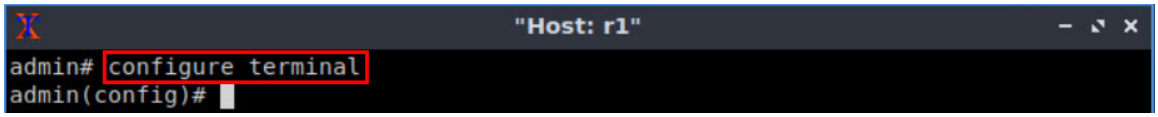

Figure 37. Enabling configuration mode on router r1.

Step 2. You will advertise the LAN network the router r1 is connected to, via BGP. Type the following command to enable BGP configuration mode.

```
router bgp 100
```

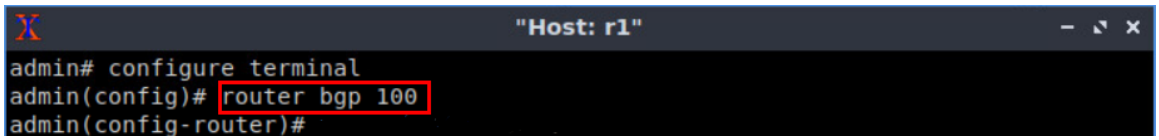

Figure 38. Entering to BGP configuration mode.

Step 3. Issue the following command so that router r1 advertises the network 192.168.1.0/24:

```
network 192.168.1.0/24
```

Figure 39. Advertising the network connected to router r1.

Step 4. Type the following command to exit from the configuration mode.

```
end
```

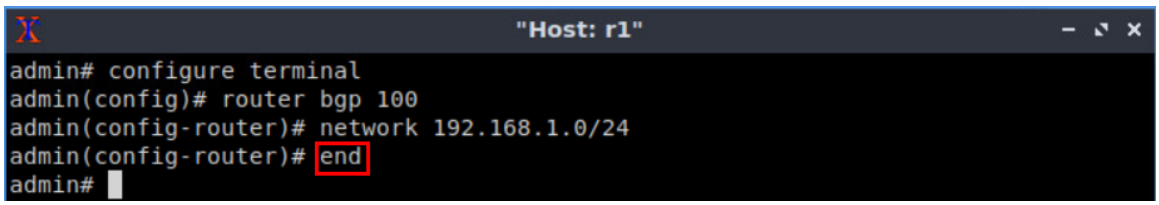

Figure 40. Exiting from configuration mode.

Step 5. Type the following command to verify BGP networks.

```
show ip bgp
```

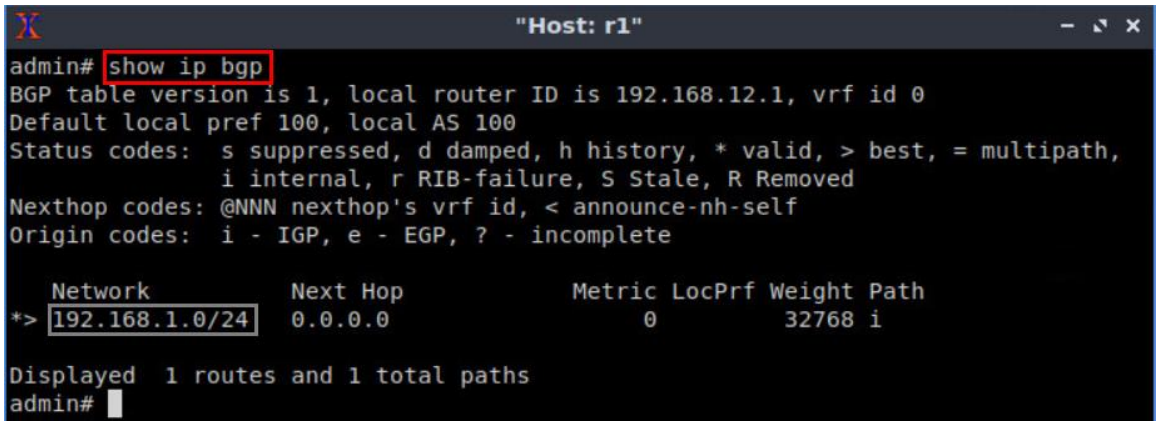

Figure 41. Verifying BGP networks on router r1.

Step 6. Type the following command to verify the routing table of router r2. You will observe the route to network 192.168.1.0/24, which is advertised by router r1. It also shows that router r2 will use the neighbor IP 192.168.12.1 to reach the network 192.168.1.0/24.

```
show ip route
```

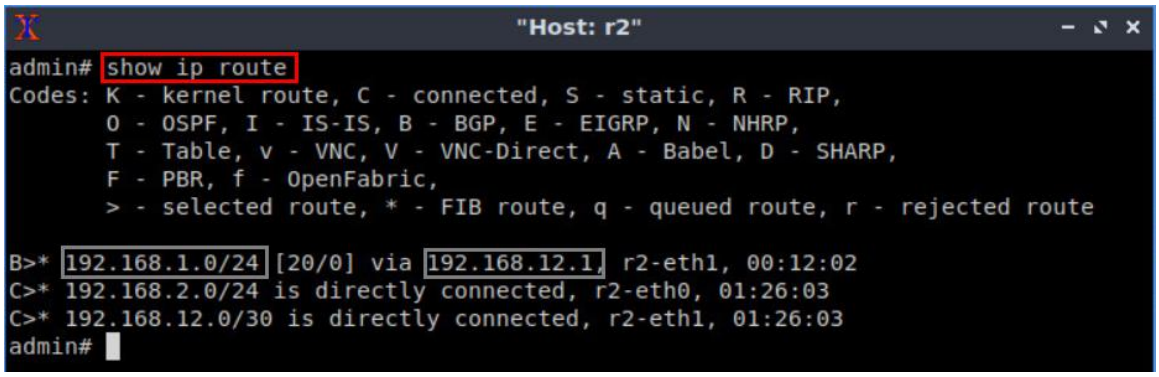

Figure 42. Verifying routing table of router r2.

Step 7. In order to verify the BGP table of router r2, issue the command shown below. The output indicates that the network connected to router r1 is listed in the BGP table of router r2. Additionally, it displays the next hop IP address (192.168.12.1) which corresponds to router r2's neighbor IP address (router r1).

```
show ip bgp
```

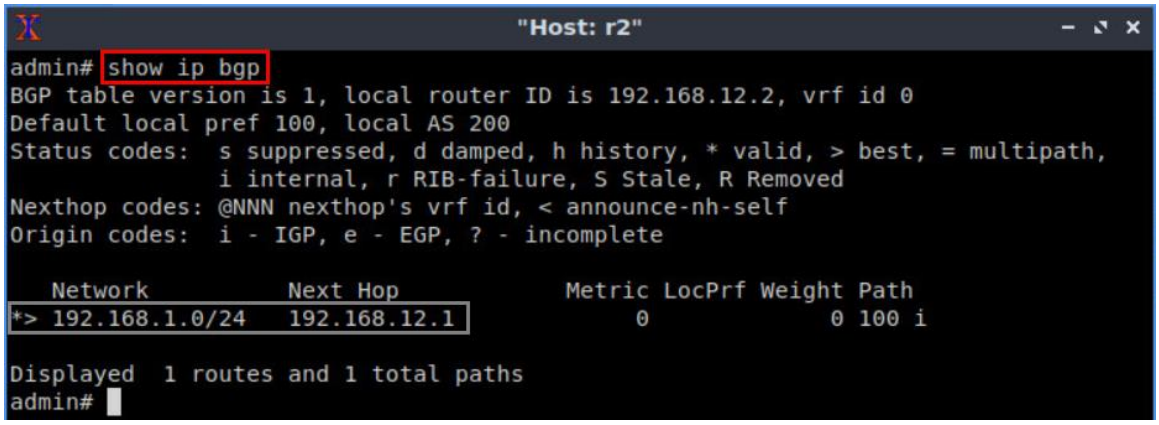

Figure 43. Verifying BGP table of router r2.

Step 8. Follow from step 1 to step 4 but with different metrics in order to advertise the LAN connected to router r2. All these steps are summarized in the following figure.

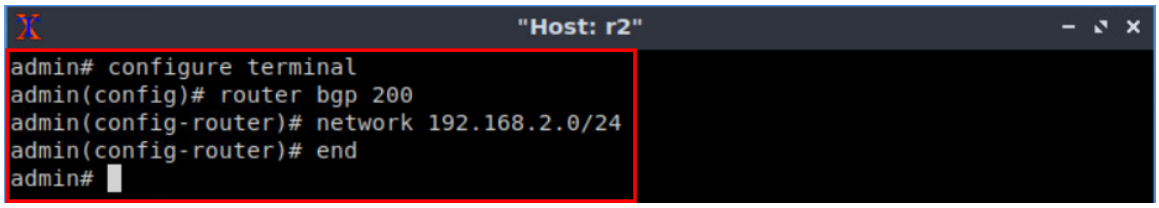

Figure 44. Advertising the network connected to router r2.

Step 9. In router r2 terminal, issue the following command to verify the BGP table of router r2. The output will list all the available BGP networks. In particular, the routing table contains its own network (192.168.2.0/24) and the remote network (192.168.1.0/24) which was advertised via EBGP.

```
show ip bgp
```

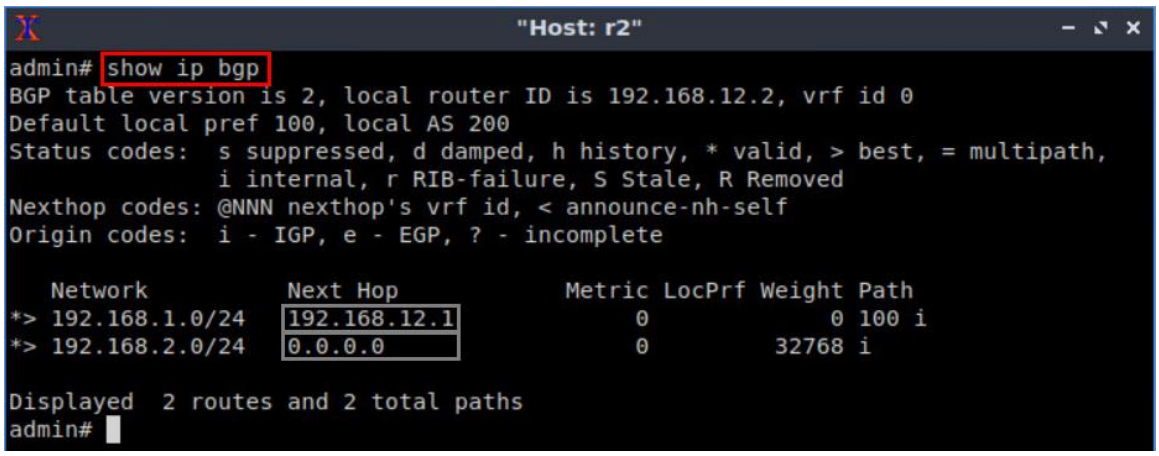

Figure 45. Verifying BGP table of router r2.

Step 10. In router r1 terminal, verify the routing table by typing the following command. The output lists that router r1 contains a route to the network 192.168.2.0/24. Notice that, this route was advertised by router r2.

```
show ip route
```

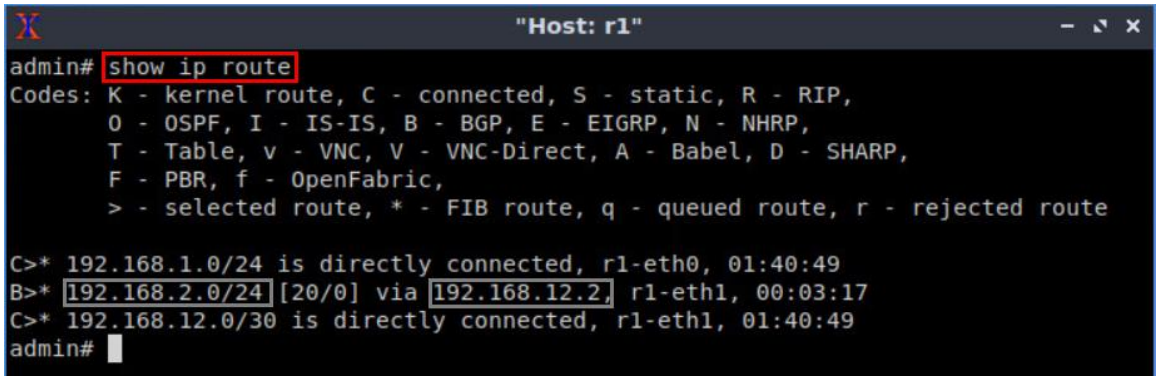

Figure 46. Verifying routing table of router r1.

4 Verify connections

In this section, you will verify that the applied configuration is working correctly by running a connectivity between host h1 and host h2.

Step 1. On host h1 terminal, perform a connectivity between host h1 and host h2 by issuing the command shown below. To stop the test, press `Ctrl+c`. The result will show a successful connectivity test.

```
ping 192.168.2.10
```

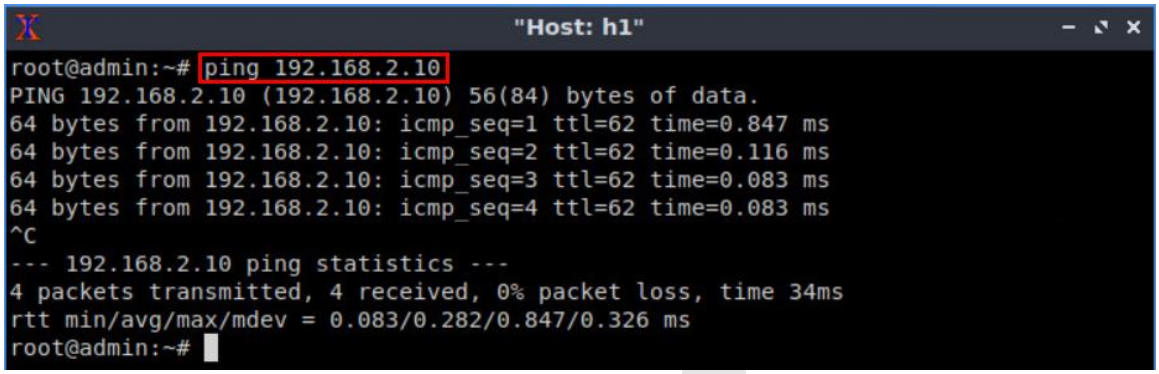

Figure 47. Connectivity test using `ping` command.

Step 2. Hold right-click on host h2 and select *Terminal*. This opens the terminal of host h2.

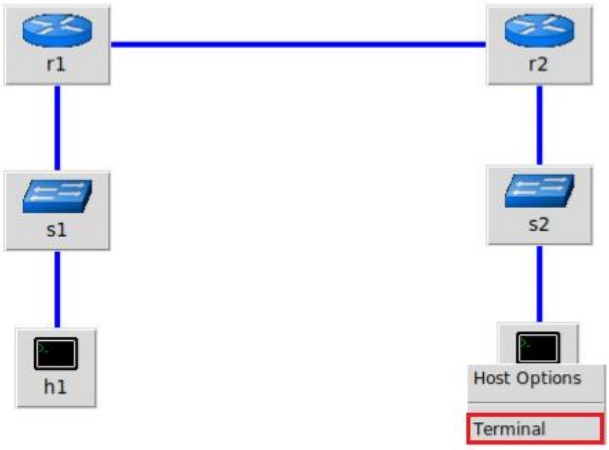

Figure 48. Opening host h2 terminal.

Step 3. Similarly, on host h2 terminal, perform a connectivity between host h2 and host h1 by issuing the command shown below. To stop the test, press `Ctrl+c`. The result will show a successful connectivity test.

```
ping 192.168.1.10
```

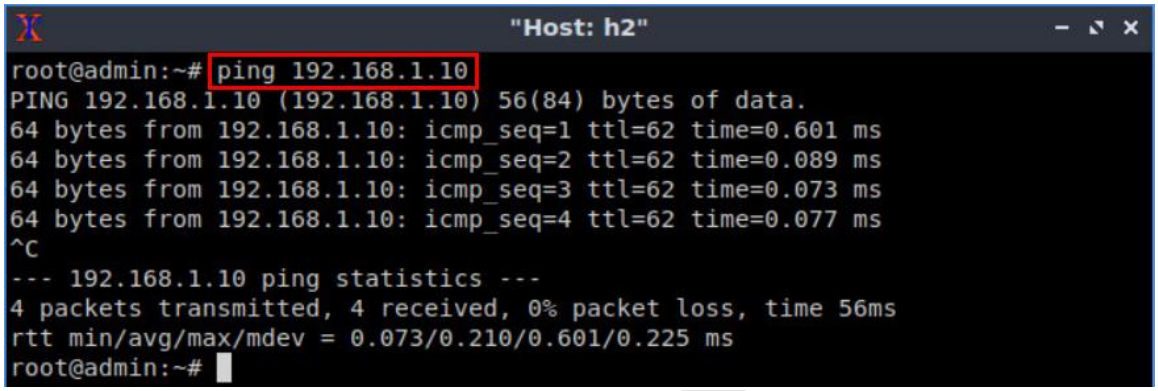

Figure 49. Connectivity test using `ping` command.

This concludes Lab 2. Stop the emulation and then exit out of MiniEdit.

References

1. Linux foundation collaborative projects, "*FRR routing documentation*", 2017. [Online]. Available: http://docs.frrouting.org/en/latest/
2. G. Malkin, "*RIP Version 2*," RFC 2453 updated by RFC 4822, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2453.txt.
3. J. Moy, "*OSPF version 2*", 1998. [Online]. Available: https://www.hjp.at/doc/rfc/rfc2178.html
4. Y. Rekhter, T. Li, S. Hares, "*A border gateway protocol 4 (BGP-4),*" RFC 4271 updated by RFCs 6286, 6608, 6793, 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt.

5. D. Oran, "*OSI IS-IS intra-domain routing protocol,*" RFC 1142, 1990. [Online]. Available: http://www.ietf.org/rfc/rfc1142.txt.
6. P. Jakma, D. Lamparter. "*Introduction to the quagga routing suite*," 2014, *IEEE Network* 28.
7. K. Ishiguro, "*Gnu zebra,*". [Online]. Available: *http://www. zebra. org* (2002).
8. A. Tanenbaum, D. Wetherall, "*Computer networks*", 5th Edition, Pearson, 2012.
9. Mininet walkthrough. [Online]. Available: http://Mininet.org.
10. B. Lantz, G. Gee, "*MiniEdit: a simple network editor for Mininet,*" 2013. [Online]. Available: https://github.com/Mininet/Mininet/blob/master/examples.
11. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach"*. Morgan Kaufmann, 2016

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 3: Early efforts of SDN: MPLS example of a control plane that establishes semi-static forwarding paths

Document Version: 06-01-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

The lab discusses the concept of Multiprotocol Label Switching (MPLS). Furthermore, static MPLS will be configured and verified between two hosts that are required to exchange routes.

Objectives

By the end of this lab, the user should be able to:

1. Explain the concept of MPLS.
2. Understand the concept of Software Defined Networking (SDN).
3. Configure and verify static MPLS.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Configuring static MPLS from router r1 to router r2.
4. Section 4: Configuring static MPLS from router r2 to router r1.
5. Section 5: Verifying configuration.

1 Introduction

MPLS is a deviation from the autonomous and distributed forwarding decisions characteristic of traditional Internet routers, in that sense it was a small step toward a more SDN-like Internet switching paradigm[1].

In a traditional IP network, each network router performs an IP lookup on the routed data or packet, the router determines a next-hop-based on its routing table and forwards the

packet to the next-hop. Every router does the same on the same data or packet each making its own independent routing decision until the final destination is reached.

MPLS was originally envisioned as a technology that could improve the scaling of Layer 3 IP routing by avoiding the large number of "fully connected" routers that were required for the Internet core. In an MPLS enabled network, MPLS does “label switching”; which means the first router or network device does a routing lookup, but instead of finding a next-hop, it finds the final destination router. An MPLS configured router applies a “label” on the data, other routers use the label to route the traffic without needing to perform any additional IP lookups which makes packet forwarding faster.

Similar to MPLS, SDN is meant to address the fact that the static architecture of traditional networks is decentralized and complex while current networks require more flexibility and easy troubleshooting. SDN attempts to centralize network intelligence in one network component by disassociating the forwarding process of network packets (data plane) from the routing process (control plane). The control plane makes routing decisions. The data plane forwards data (packets) through the router. With SDN routing, decisions are made remotely instead of on each individual router.

1.1 Introduction to MPLS

MPLS is data forwarding technology that increases the speed and controls the flow of network traffic. The packet enters the edge of the MPLS backbone, is examined, classified, given an appropriate label, and forwarded to the next hop in the pre-set Label Switched Path (LSP). The predetermined paths that make MPLS work are called label-switched paths (LSPs). As the packet travels that path, each router on the path uses the label – not other information, such as the IP header – to make the forwarding decision that keeps the packet moving along the LSP. An MPLS router that performs routing based only on the label is called a label switch router (LSR). Each LSR has the ability to do three main things: push, pop, or swap labels from a packet. To push a label simply means to add a label to a packet, to pop is to remove a label from a packet and to swap is to remove and add an alternative label to the packet. It is possible for a packet to have multiple labels attached which are arranged in a stack and are considered in the order from the most recent label to the oldest label.

Within each router, the incoming label is examined, and its next hop is matched with a new label. The old label is replaced with the new label for the packet’s next destination, and then the freshly labeled packet is sent to the next router. Each router repeats the process until the packet reaches the exit router. The label information is removed at either the last hop or the exit router, so that the packet goes back to being identified by an IP header instead of an MPLS label.

There are three different types of LSR: ingress, egress and intermediate. The ingress LSR is at the edge of an MPLS network and is the first to insert an MPLS header and label on a packet. The egress LSR is at the edge of the network and is the last point before leaving

the network and thus removes all the MPLS labels and header. Both the ingress and egress LSR's are considered Provider Edge (PE) routers.

The LSRs which exist within the network are called intermediate LSR's and are responsible for pushing, popping, and swapping labels based on the routing with the MPLS network; the intermediate LSR's are considered Provider (P) routers.

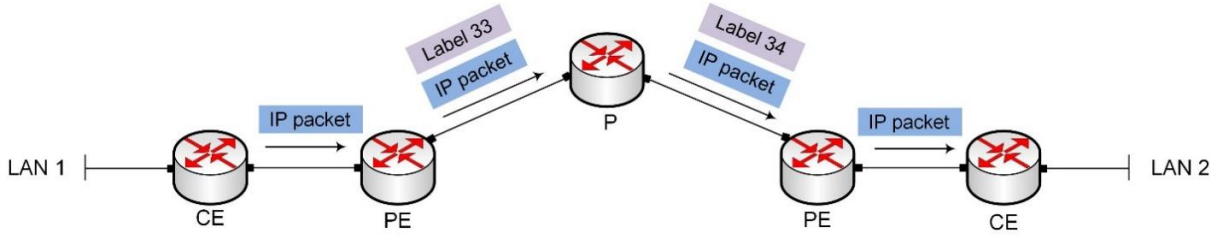

Figure 1. MPLS label forwarding.

Consider Figure 1. The Customer Edge (CE) of LAN 1 will forward the IP packet to the provider Edge (PE). The PE will push label 33 along with the IP packet to the provider (P). P will replace the label with label 34 which will be forwarded to the Provider Edge (PE). The PE will pop the label and the IP packet will be delivered to the Customer Edge (CE) of LAN 2 which is the destination router.

1.2 MPLS Header architecture

MPLS operates at a layer that is generally considered to lie between traditional definitions of OSI Layer 2 (data link layer) and Layer 3 (network layer), and thus is often referred to as a layer 2.5 protocol.

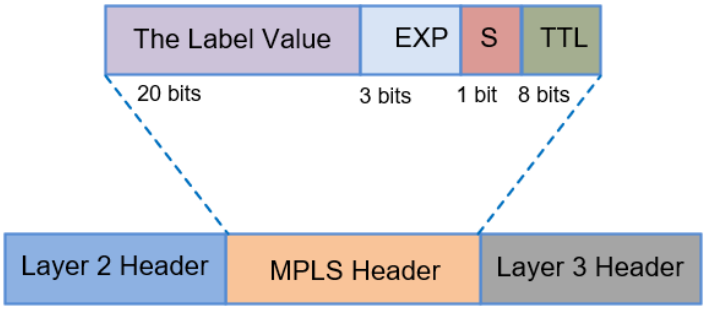

Figure 2. MPLS Header architecture.

Consider Figure 2. MPLS works by prefixing packets with an MPLS header, containing one or more labels. This is called a label stack. Each entry in the label stack contains four fields.

1. The Label Value: The first 20 bits are the label value. This value can be between 0 and 1,048,575. However, the first 16 values are exempted from normal use; that is, they have a special meaning.

2. EXP: Three bits are the experimental (EXP) bits. These bits are used solely for quality of service (QoS) which represents the set of techniques necessary to manage network bandwidth, delay, jitter, and packet loss.

3. S: 1 bit is the Bottom of Stack (BoS) bit. It is 0, unless this is the bottom label in the stack. If so, the BoS bit is set to 1. The stack is the collection of labels that are found on top of the packet. The stack can consist of just one label, or it might have more.

4. TTL: The last 8 bits are used for Time To Live (TTL). This TTL has the same function as the TTL found in the IP header. It is simply decreased by 1 at each hop, and its main function is to avoid a packet being stuck in a routing loop. If a routing loop occurs and no TTL is present, the packet loops forever. If the TTL of the label reaches 0, the packet is discarded[2].

2 Lab topology

Consider Figure 3. The topology consists of three networks, Customer 1, ISP and Customer 2. Customer 1 and Customer 2 exchange routes through static MPLS.

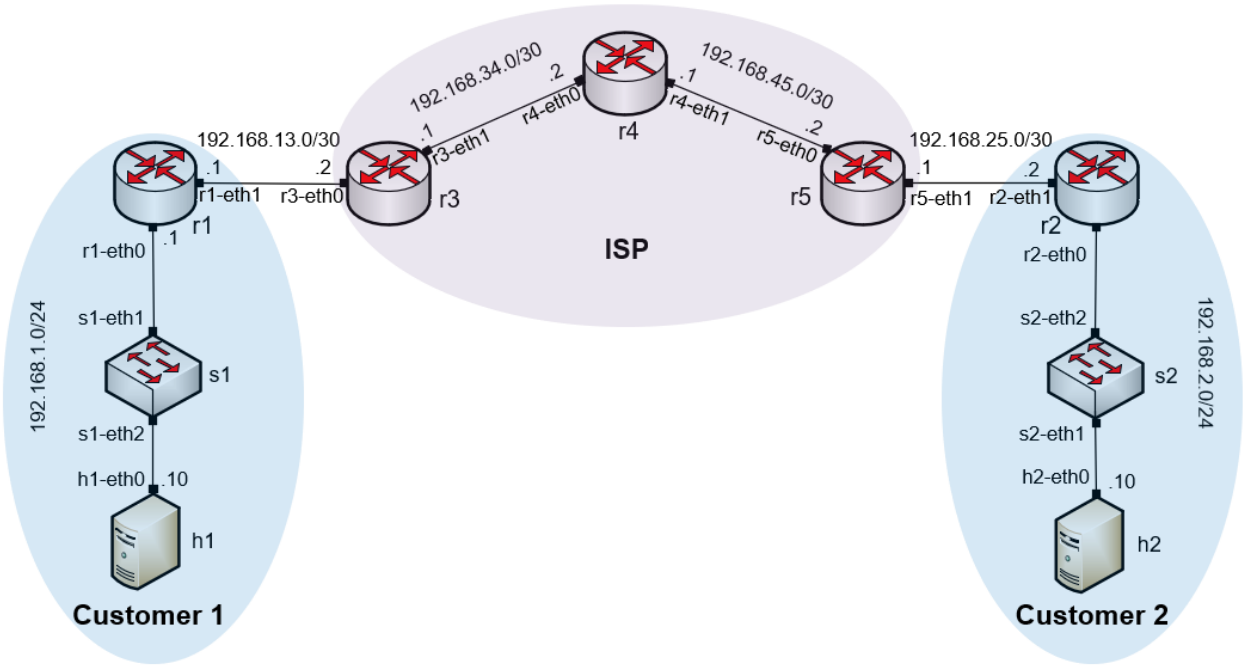

Figure 3. Lab topology.

2.1 Lab settings

Routers and hosts are already configured according to the IP addresses shown in Table 2.

Table 2. Topology information.

Device	Interface	IPV4 Address	Subnet	Default gateway
r1	r1-eth0	192.168.1.1	/24	N/A
	r1-eth1	192.168.13.1	/30	N/A
r2	r2-eth0	192.168.2.1	/24	N/A
	r2-eth1	192.168.25.1	/30	N/A
r3	r3-eth0	192.168.13.2	/30	N/A
	r3-eth1	192.168.34.1	/30	N/A
r4	r4-eth0	192.168.34.2	/30	N/A
	r4-eth1	192.168.45.1	/30	N/A
r5	r5-eth0	192.168.45.2	/30	N/A
	r5-eth1	192.168.25.2	/30	N/A
h1	h1-eth0	192.168.1.10	/24	192.168.1.1
h2	h2-eth0	192.168.2.10	/24	192.168.2.1

2.2 Opening the topology

In this section, you will open MiniEdit[7] and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), saving topologies, and exporting layer 2 models.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

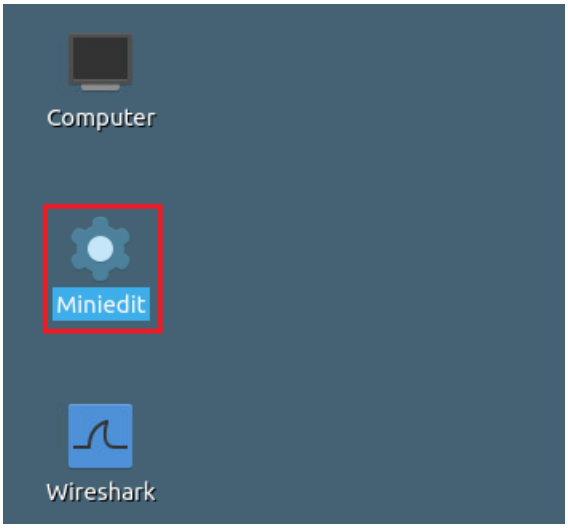

Figure 4. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab topology. Open the *Lab3.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab3* and click on *Open*.

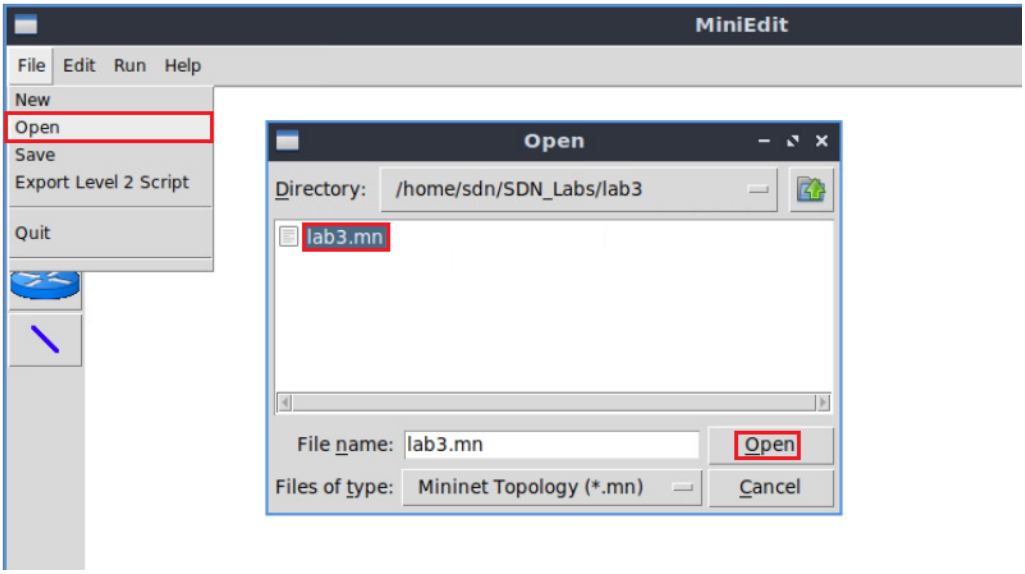

Figure 5. MiniEdit's open dialog.

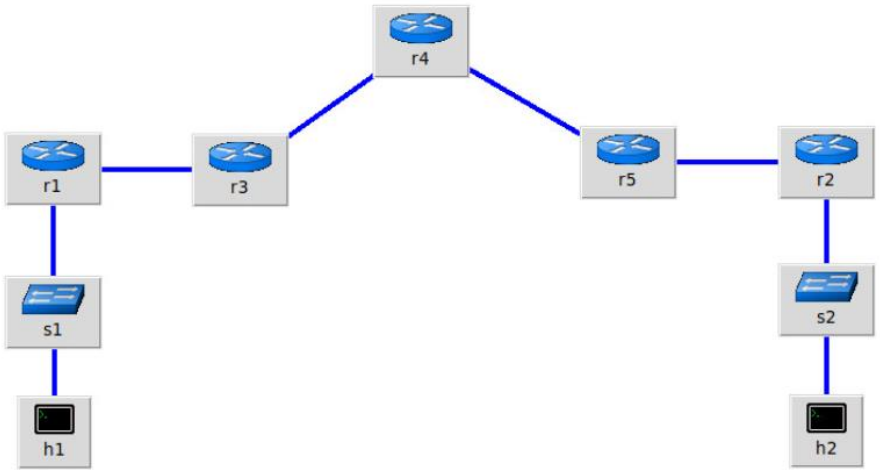

Figure 6. Mininet topology.

2.3 Loading the configuration file

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open Linux terminal.

Figure 7. Opening the Linux terminal.

Step 2. Click on the Linux terminal and navigate into the *SDN_Labs/lab3* directory by issuing the following command. This folder contains a configuration file and the script responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab3
```

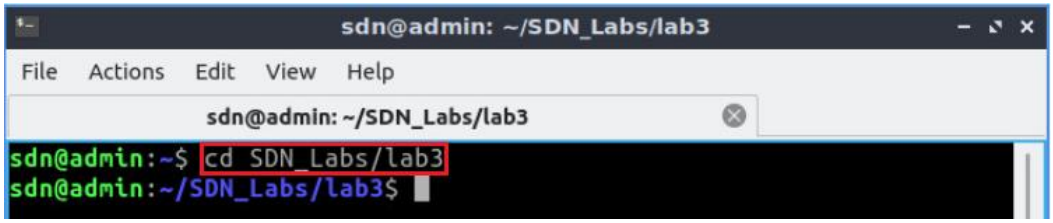

Figure 8. Entering the *SDN_Labs/lab3* directory.

Step 3. To execute the shell script, type the following command. The argument of the program corresponds to the configuration zip file that will be loaded in all the routers in the topology.

```
./config_loader.sh lab3_config.zip
```

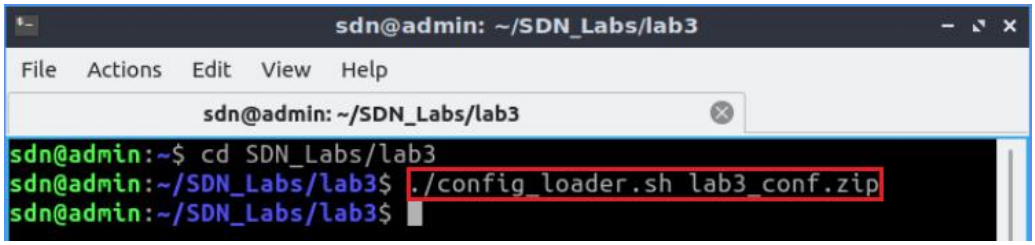

Figure 9. Executing the shell script to load the configuration.

Step 4. Type the following command to exit the Linux terminal.

```
exit
```

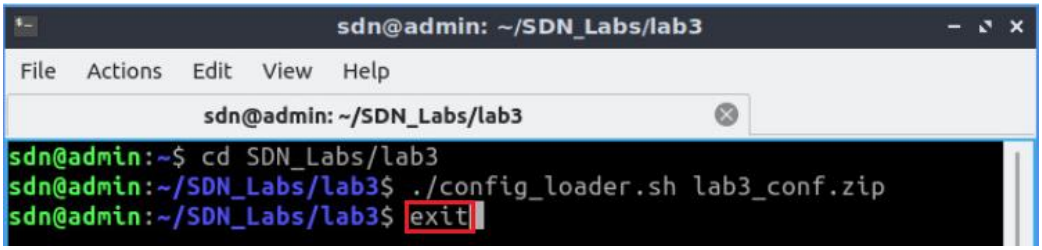

Figure 10. Exiting the terminal.

2.4 Running the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. At this point the interfaces of hosts h1 and h2 are configured. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

Figure 11. Starting the emulation.

Step 2. Issue the following command to display the interface names and connections.

```
links
```

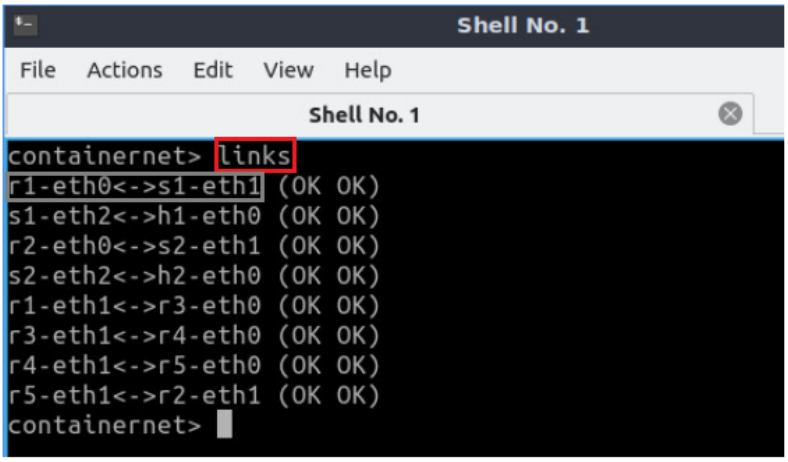

Figure 12. Displaying network interfaces.

In Figure 12, the link displayed within the gray box indicates that interface *eth1* of router r1 connects to interface *eth1* of switch s1 (i.e., *r1-eth0<->s1-eth1*).

2.5 Enabling MPLS forwarding

In this section, you will enable MPLS forwarding in the kernel. All the router interfaces assign labels which are used to forward packets. You will assign value 1 to all the router interfaces so that they participate in the labeling process. *Platform_labels* is the table which recognizes all the assigned labels and participates in the label forwarding. You will

assign value 100000 (maximum value for label forwarding) to *Platform_labels* in order to enable label forwarding.

Step 1. In order to open router r1's terminal, hold right-click on router r1 and select Terminal.

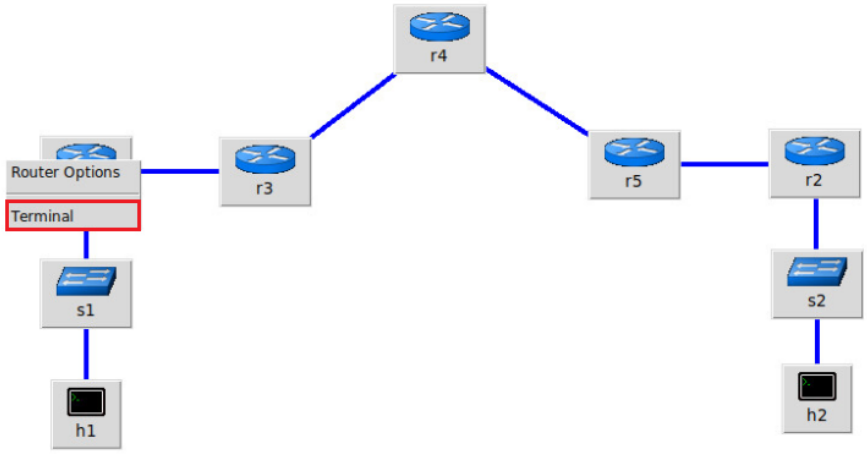

Figure 13. Opening a terminal on router r1.

Step 2. Type the following command to assign value 1 to interface *r1-eth0* in order to enable the interface to participate in the label processing.

```
sysctl -w net.mpls.conf.r1-eth0.input=1
```

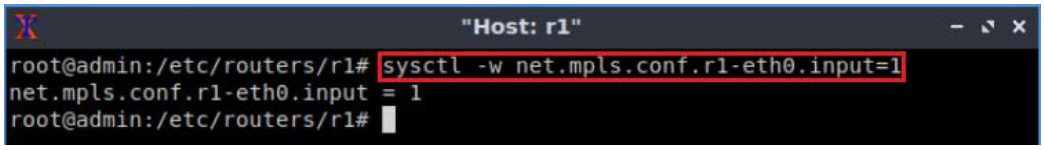

Figure 14. Enabling interface *r1-eth0* to participate in the label processing.

Step 3. Type the following command to assign value 1 to the interface *r1-eth1* in order to enable the interface to participate in the label processing.

```
sysctl -w net.mpls.conf.r1-eth1.input=1
```

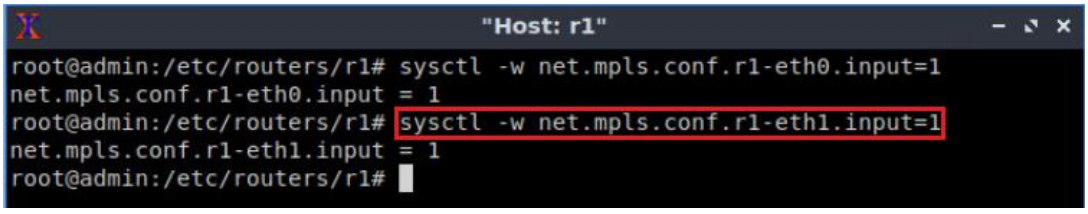

Figure 15. Enabling interface *r1-eth1* to participate in the label processing.

Step 4. In order to enable label forwarding, type the following command to assign the value 100000 to the *Platform_labels*.

```
sysctl -w net.mpls.platform_labels=100000
```

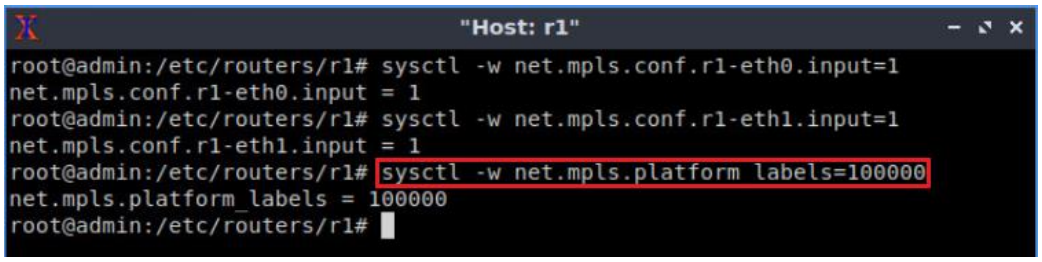

Figure 16. Enabling label forwarding in router r1.

Step 5. Follow steps 2 to 4 in order to enable MPLS forwarding in router r2. All the steps are summarized in the following figure.

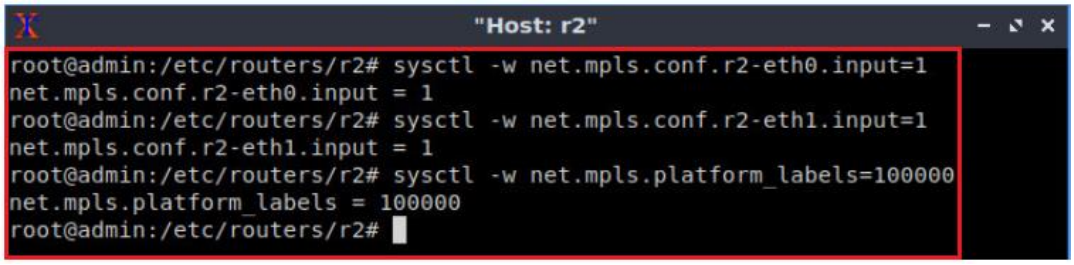

Figure 17. Enabling MPLS forwarding in router r2.

Step 6. Follow steps 2 to 4 in order to enable MPLS forwarding in router r3. All the steps are summarized in the following figure.

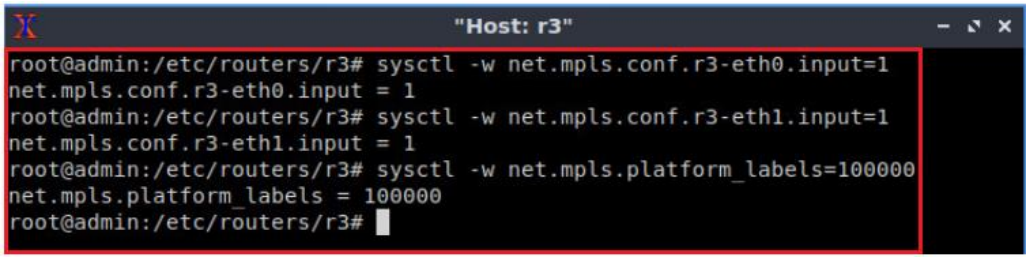

Figure 18. Enable MPLS forwarding in router r3.

Step 7. Follow steps 2 to 4 in order to enable MPLS forwarding in router r4. All the steps are summarized in the following figure.

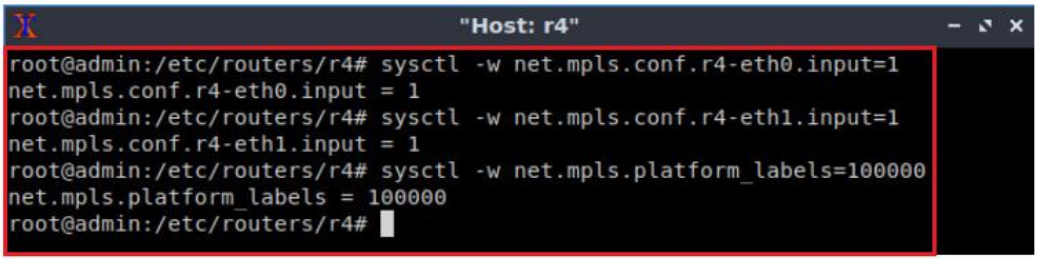

Figure 19. Enable MPLS forwarding in router r4.

Step 8. Follow steps 2 to 4 in order to enable MPLS forwarding in router r5. All the steps are summarized in the following figure.

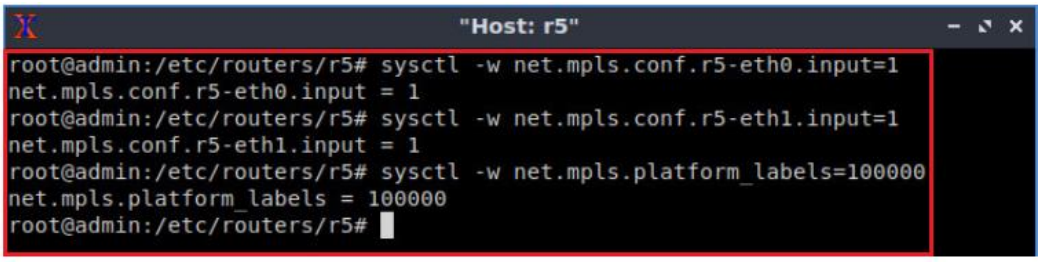

Figure 20. Enable MPLS forwarding in router r5.

2.6 Verifying the configuration

You will verify the IP addresses listed in Table 2 and inspect the routing table of routers r1, r2, r3, r4 and r5.

Step 1. Hold right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands in that host.

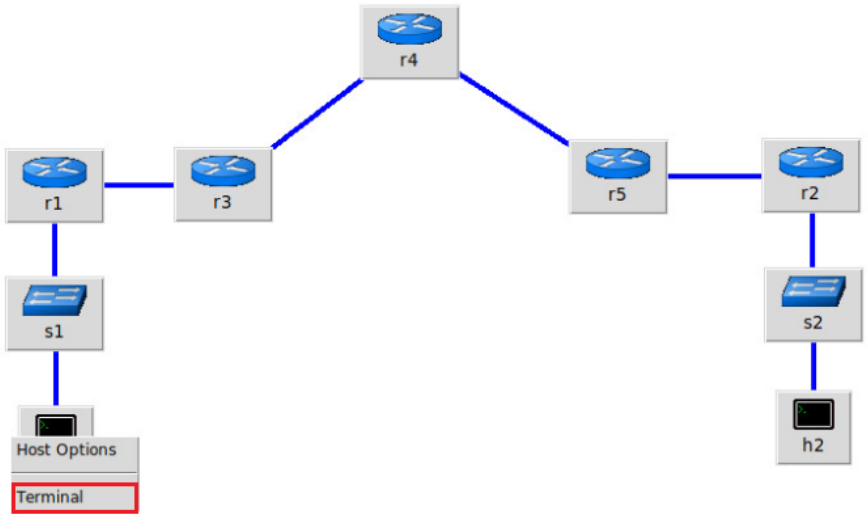

Figure 21. Opening a terminal in host h1.

Step 2. On host h1 terminal, type the command shown below to verify that the IP address was assigned successfully. You will corroborate that host h1 has interface, *h1-eth0* configured with the IP address 192.168.1.10 and the subnet mask 255.255.255.0.

```
ifconfig
```

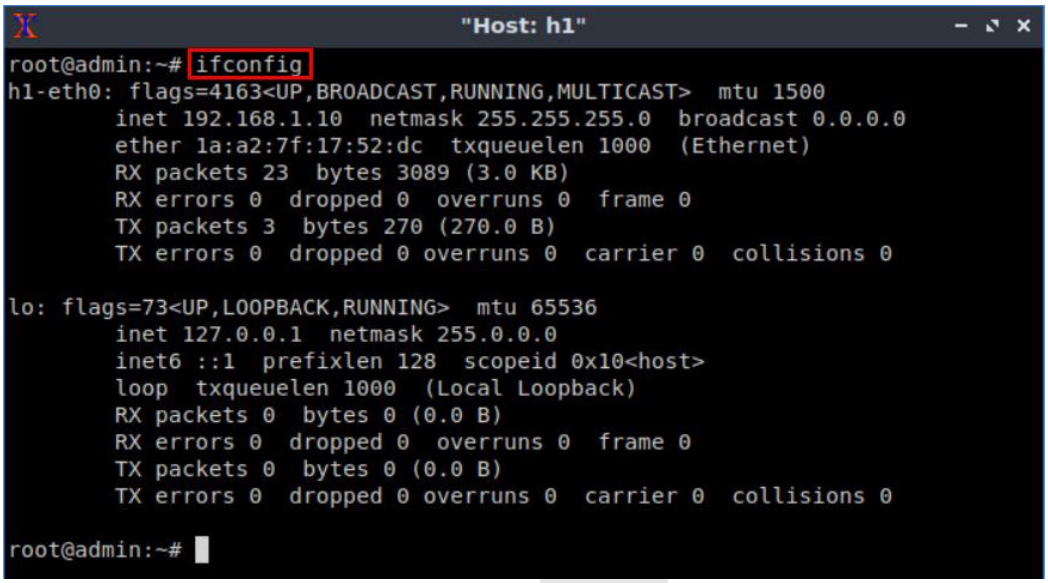

Figure 22. Output of the `ifconfig` command.

Step 3. In host h1's terminal, type the command shown below to verify that the default gateway IP address is 192.168.1.1.

```
route
```

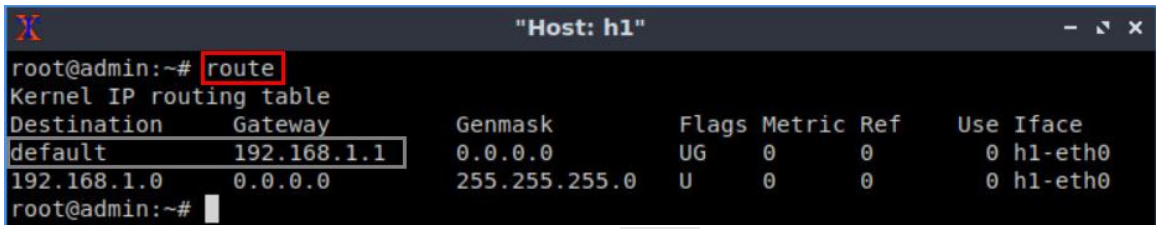

Figure 23. Output of the `route` command.

Step 4. In order to verify host h2's default route, proceed similarly by repeating from step 1 to step 3 in host h2's terminal. Similar results should be observed.

Step 5. In order to verify router r1, hold right-click on router r1 and select *Terminal*.

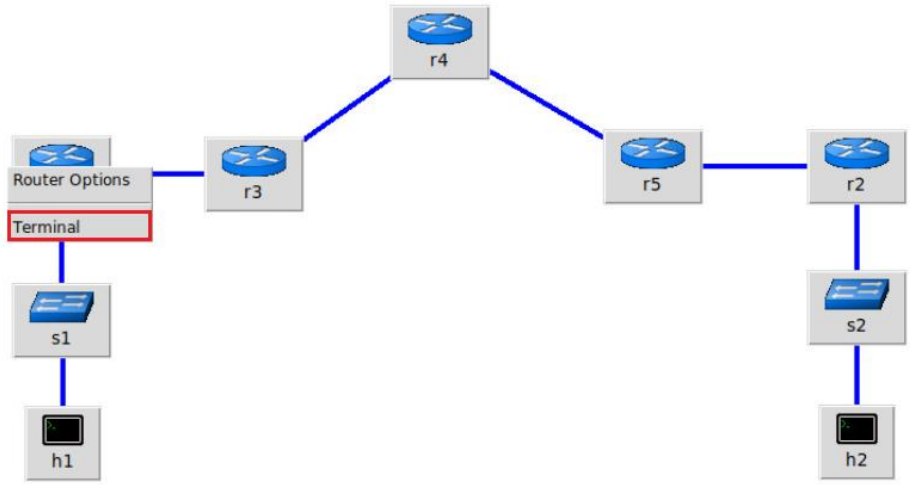

Figure 24. Opening a terminal in router r1.

Step 6. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you do not enable the zebra daemon initially. In order to start zebra, type the following command:

```
zebra
```

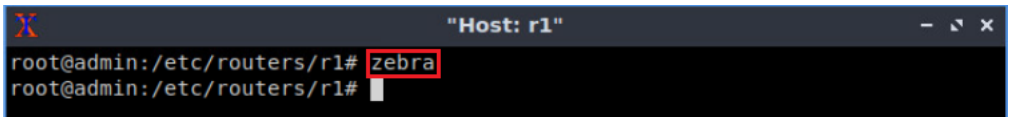

Figure 25. Starting the zebra daemon.

Step 7. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command:

```
vtysh
```

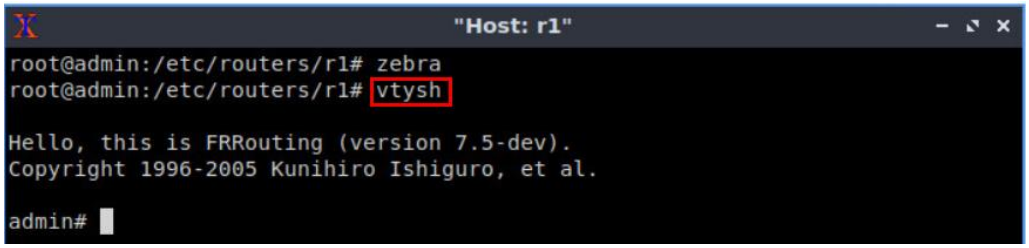

Figure 26. Starting vtysh on router r1.

Step 8. Type the following command in router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 does not contain any route to the network of router r2 (192.168.2.0/24) as there is no routing protocol configured yet.

```
show ip route
```

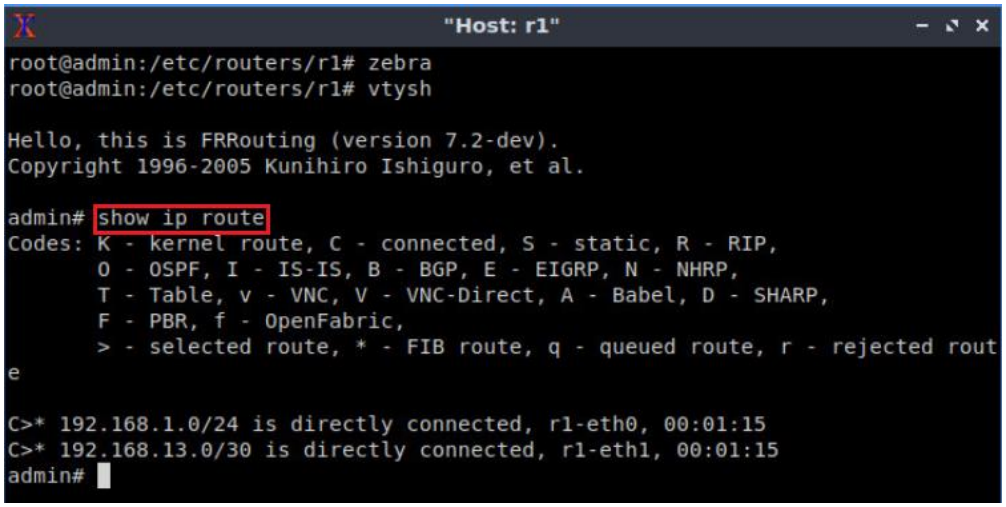

Figure 27. Displaying routing table of router r1.

The output in the figure above shows that the network 192.168.1.0/24 is directly connected through the interface *r1-eth0*. The network 192.168.13.0/30 is connected via the interface *r1-eth1*.

Step 9. Router r2 is configured similarly to router r1, but with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r2's terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r2.

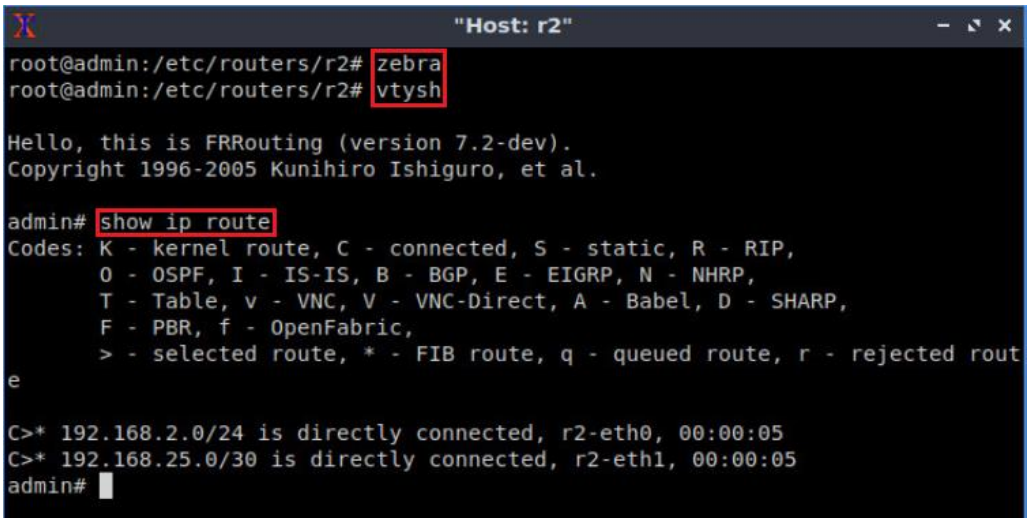

Figure 28. Displaying the routing table of router r2.

Step 10. Router r3 is configured similarly to router r1, but with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r3's terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r3.

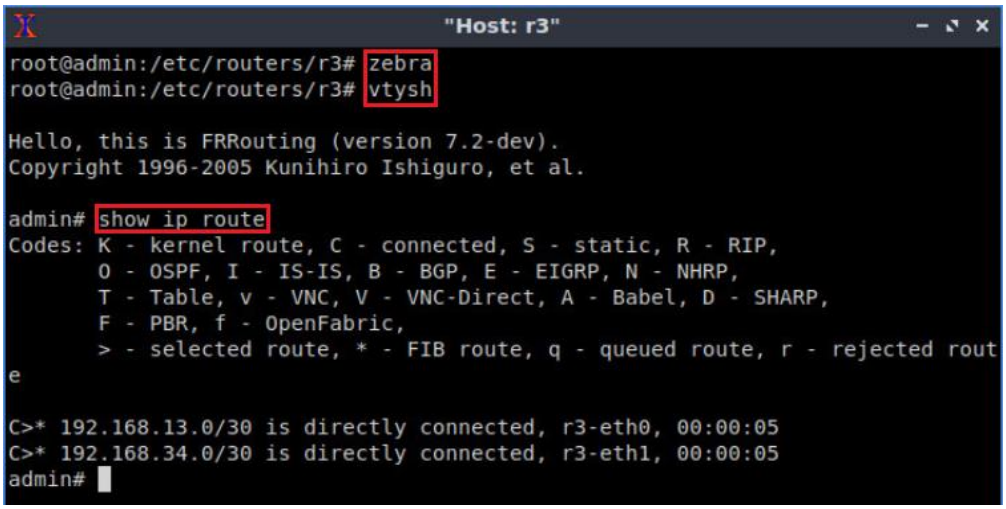

Figure 29. Displaying the routing table of router r3.

Step 11. Router r4 is configured similarly to router r1, but with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r4 terminal issue the commands depicted below. Towards the end, you will verify all the directly connected networks of router r4.

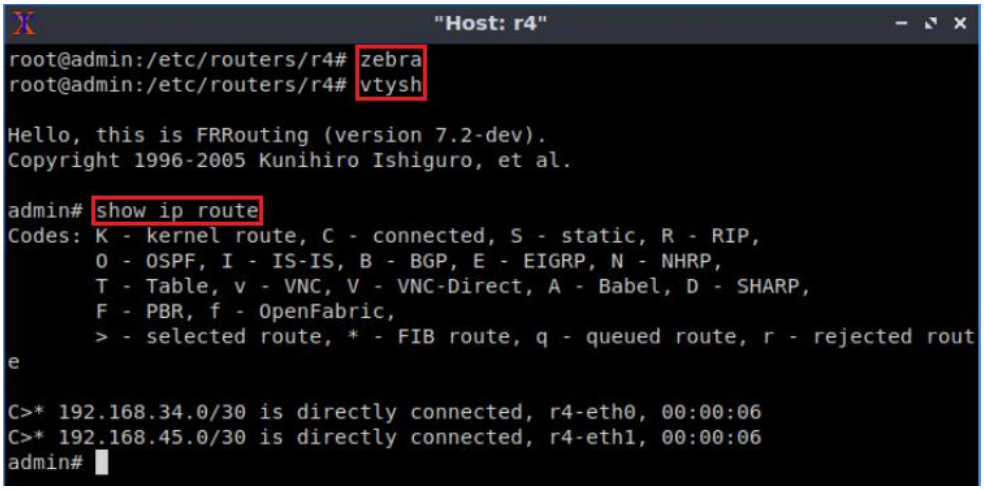

Figure 30. Displaying routing table of router r4.

Step 12. Router r5 is configured similarly to router r1, but with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r5's terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r5.

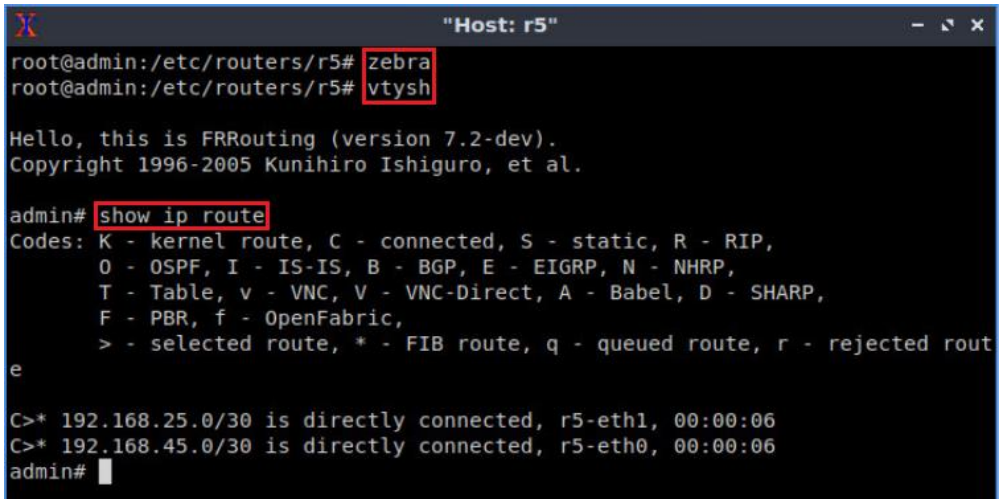

Figure 31. Displaying routing table of router r5.

3 Configuring static MPLS from router r1 to router r2

In this section, you will configure static MPLS on routers so that host h1 can reach host h2. Router r1 will push a label to router r3. Routers r3, r4 and r5 will swap the labels and the data packet will reach router r2. Router r2 will pop the label and the data packet will be delivered to the destination host h2.

3.1 Pushing labels

Step 1. At this point, router r1 can reach the directly connected network 192.168.13.0/30. To communicate with other networks, you will configure static routing on router r1. To configure static routing, you need to enable the static daemon first. In router r1, type the following command to exit the vtysh session:

```
exit
```

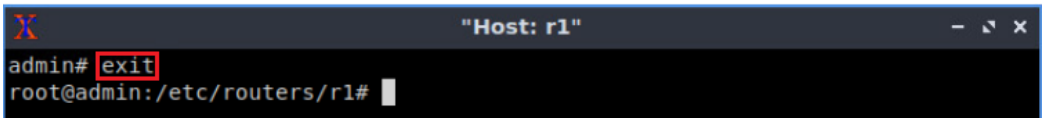

Figure 32. Exiting the vtysh session.

Step 2. Type the following command to enable the static routing daemon on router r1.

```
staticd
```

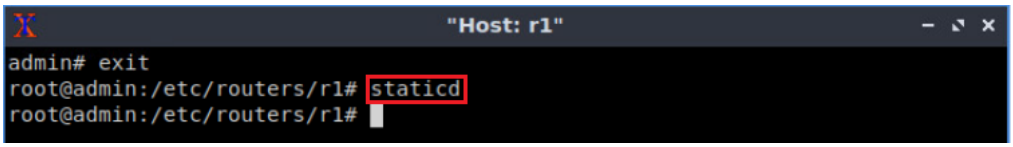

Figure 33. Starting the staticd daemon.

Step 3. In order to enter router r1 terminal, issue the following command:

```
vtysh
```

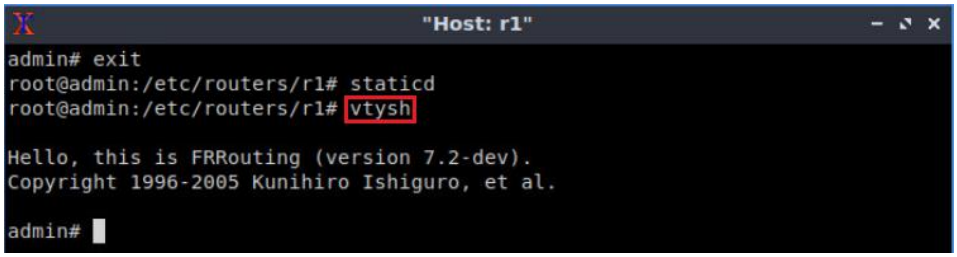

Figure 34. Starting vtysh on router r1.

Step 4. To enable router r1 configuration mode, issue the following command:

```
configure terminal
```

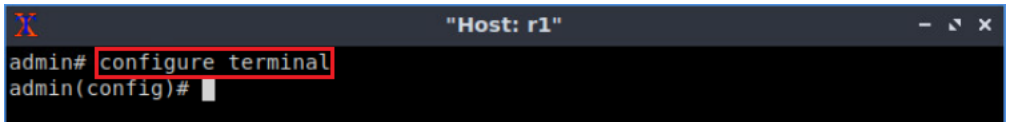

Figure 35. Enabling configuration mode on router r1.

Step 5. Type the following command to add a static route to the network 192.168.34.0/30. Router r1 will communicate with router r3 via 192.168.13.2 and the label 100 will be assigned.

```
ip route 192.168.34.0/30 192.168.13.2 label 100
```

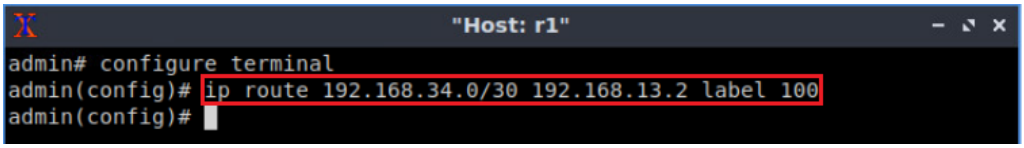

Figure 36. Pushing label for the network 192.168.34.0/30.

Step 6. Type the following command to add a static route to the network 192.168.45.0/30. Router r1 will communicate with router r3 via 192.168.13.2 and the label 100 will be assigned.

```
ip route 192.168.45.0/30 192.168.13.2 label 100
```

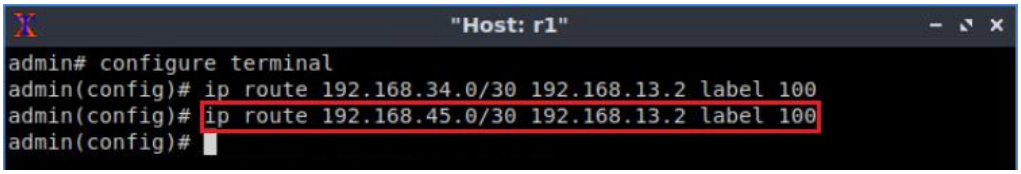

Figure 37. Pushing label for the network 192.168.45.0/30.

Step 7. Type the following command to configure static routing to the network 192.168.25.0/30. Router r1 will communicate with router r3 via 192.168.13.2 and the label 100 will be assigned.

```
ip route 192.168.25.0/30 192.168.13.2 label 100
```

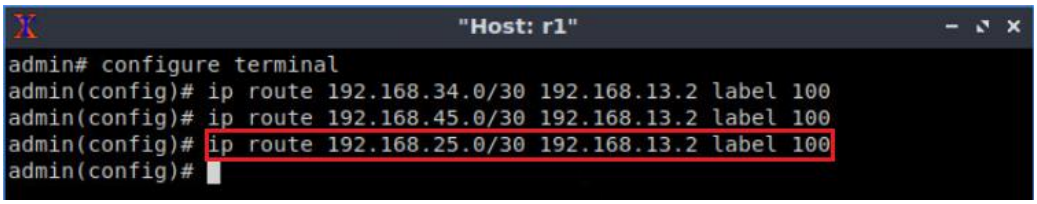

Figure 38. Pushing label for the network 192.168.25.0/30.

Step 8. Type the following command to configure static routing to the network 192.168.2.0/24. Router r1 will communicate with router r3 via 192.168.13.2 and the label 100 will be assigned.

```
ip route 192.168.2.0/24 192.168.13.2 label 100
```

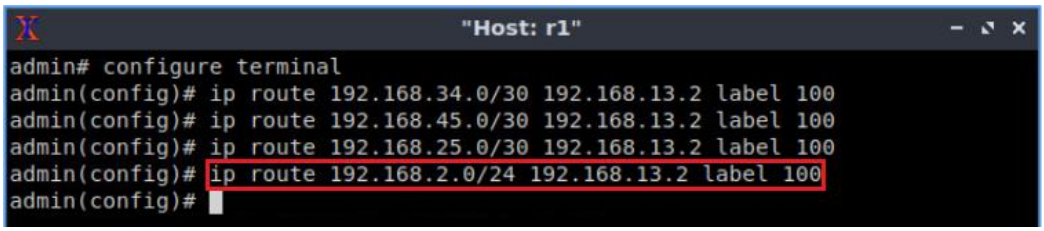

Figure 39. Pushing label for the network 192.168.2.0/24.

Step 9. Type the following command to exit configuration mode.

```
end
```

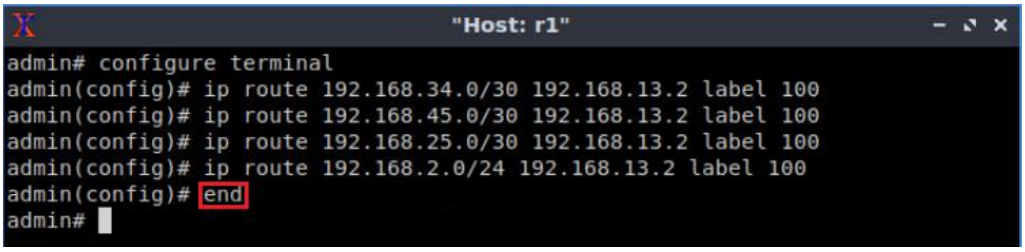

Figure 40. Exiting configuration mode.

Step 10. In the following step, you will verify if router r1 can communicate with router r3. In router r3, type the following command to exit the vtysh session:

```
exit
```

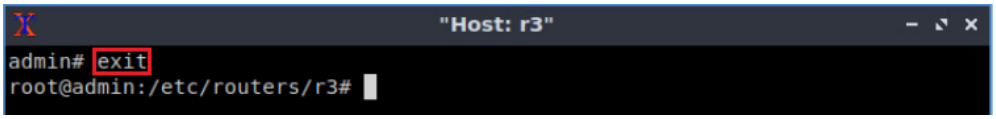

Figure 41. Exiting the vtysh session.

Step 11. In router r3, type the following command in order to capture and analyze network traffic received by the interface *r3-eth0*.

```
tcpdump -i r3-eth0
```

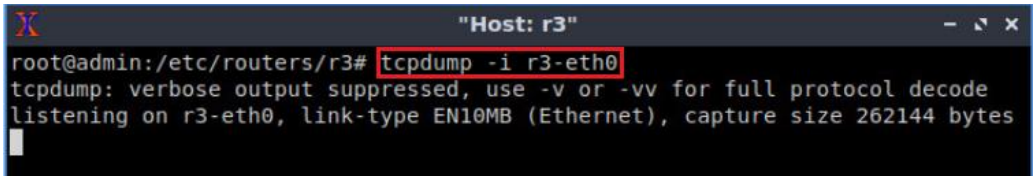

Figure 42. Enabling tcpdump to capture packets in router r3.

Step 12. Test the connectivity between router r1 and router r3 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test.

```
ping 192.168.13.2
```

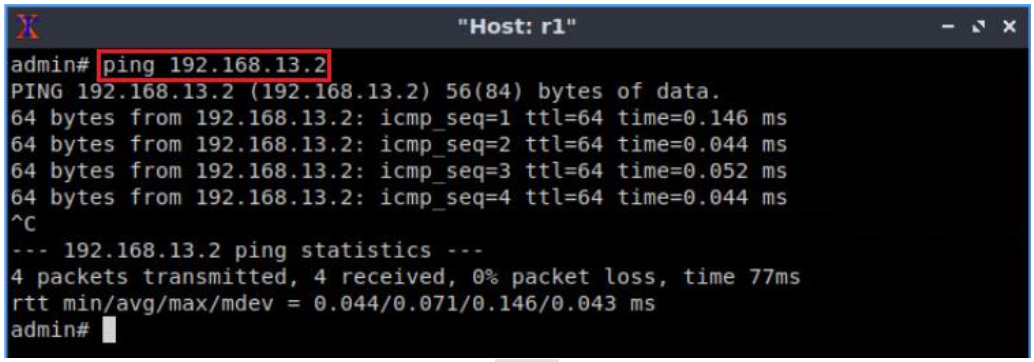

Figure 43. Output of the `ping` command in router r1.

Step 13. In router r3, you will notice that router r1 (192.168.13.1) can communicate with router r3 (192.168.13.2) through IP routing. Router r1 will not use any label to reach router r3 as the network is directly connected. To stop the test, press `Ctrl+c`.

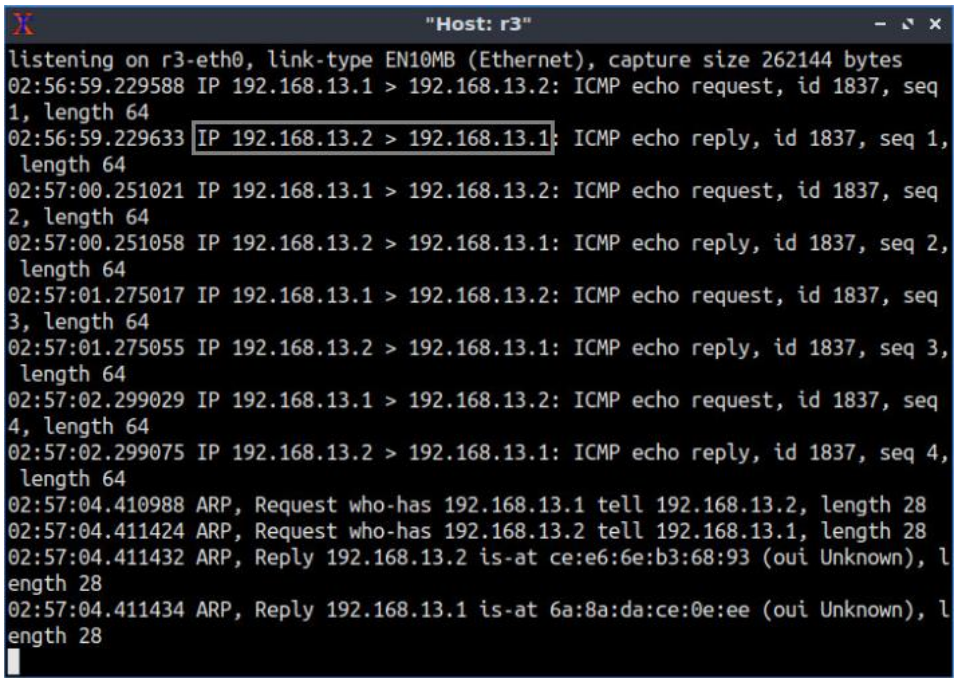

Figure 44. Verifying connectivity in router r3.

3.2 Swapping labels

Step 1. Type the following command in order to capture and analyze network traffic received by the interface, *r3-eth1*.

```
tcpdump -i r3-eth1
```

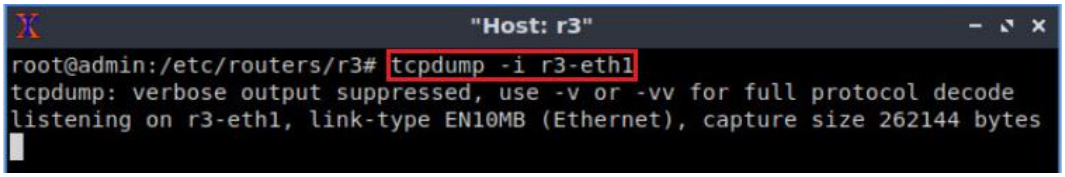

Figure 45.Enabling tcpdump to capture packets in router r3.

Step 2. Test the connectivity between router r1 and router r3 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`.

```
ping 192.168.34.1
```

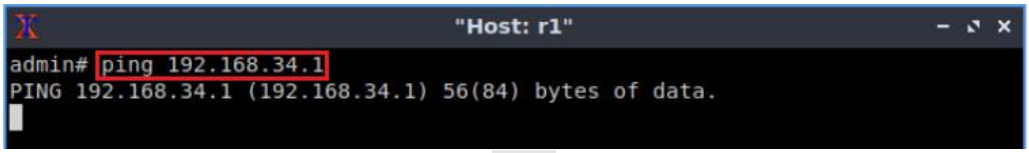

Figure 46. Output of the `ping` command in router r1.

At this point, router r1 will use label 100 to reach router r3 but router r3 cannot reach the network 192.168.34.0/30 as there is no label assigned to router r3. To stop the test, press `Ctrl+c` in router r3.

Step 3. In order to enter router r3 terminal, issue the following command:

```
vtysh
```

Figure 47. Starting vtysh on router r3.

Step 4. To enable router r3 configuration mode, issue the following command:

```
configure terminal
```

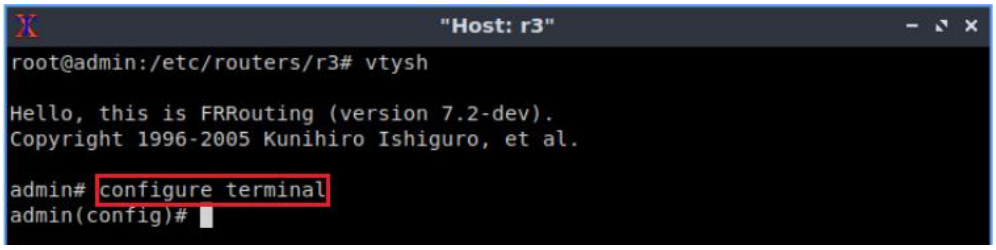

Figure 48. Enabling configuration mode in router r3.

Step 5. In this step, you will configure label swapping in router r3. Router r3 will receive label 100 from router r1 and swap the label with another one (200) to reach the interface *r4-eth0* (192.168.34.2). Type the following command to enable label swapping in router r3.

```
mpls lsp 100 192.168.34.2 200
```

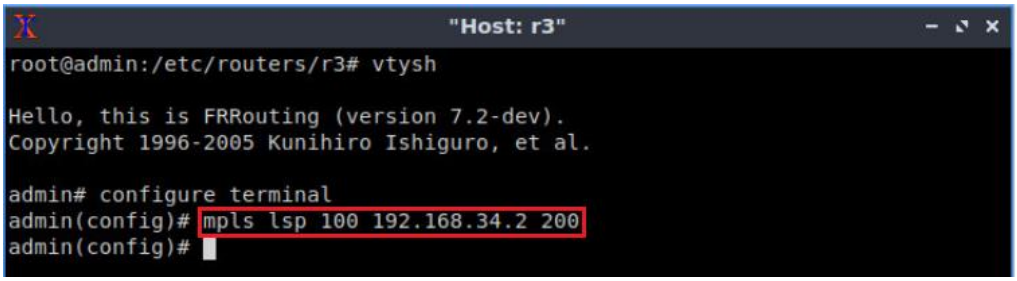

Figure 49. Enabling label swapping on router r3.

Step 6. Type the following command to exit configuration mode.

```
end
```

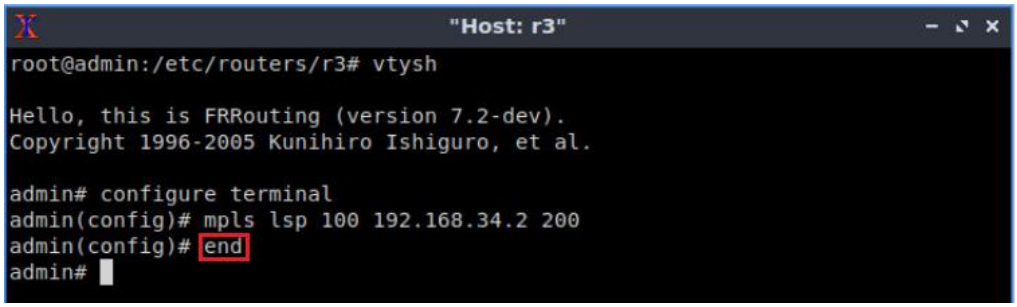

Figure 50. Exiting from configuration mode.

Step 7. In the following step, you will verify that router r1 can communicate with router r3 (interface *r3-eth1*). In router r3, type the following command to exit the vtysh session:

```
exit
```

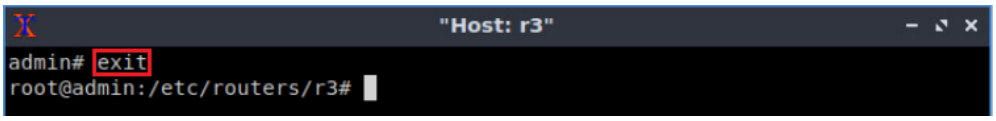

Figure 51. Exiting the vtysh session.

Step 8. Type the following command in order to capture and analyze network traffic received by the interface *r3-eth1.*

```
tcpdump -i r3-eth1
```

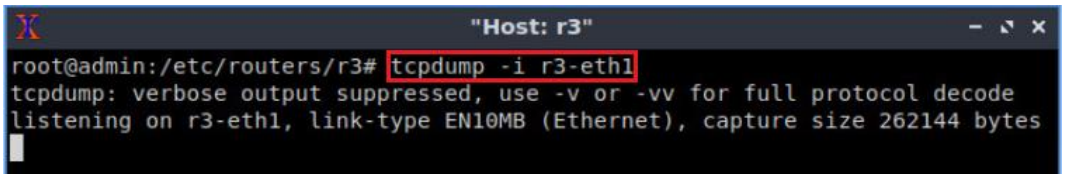

Figure 52. Enabling tcpdump to capture packets in router r3.

Step 9. Test the connectivity between router r1 and router r3 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r3's terminal. You will notice that router r3 is receiving packets from router r1.

```
ping 192.168.34.1
```

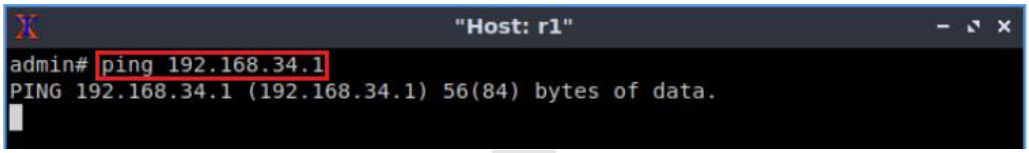

Figure 53. Output of the `ping` command in router r1.

Step 10. If you observe router r3, you will notice that router r1 (192.168.13.1) will communicate with router r3 (192.168.34.1) through MPLS routing. Router r1 uses label 100 to reach router r3 and router r3 swaps the label to 200 to reach network 192.168.34.0/30. To stop the test, press `Ctrl+c`.

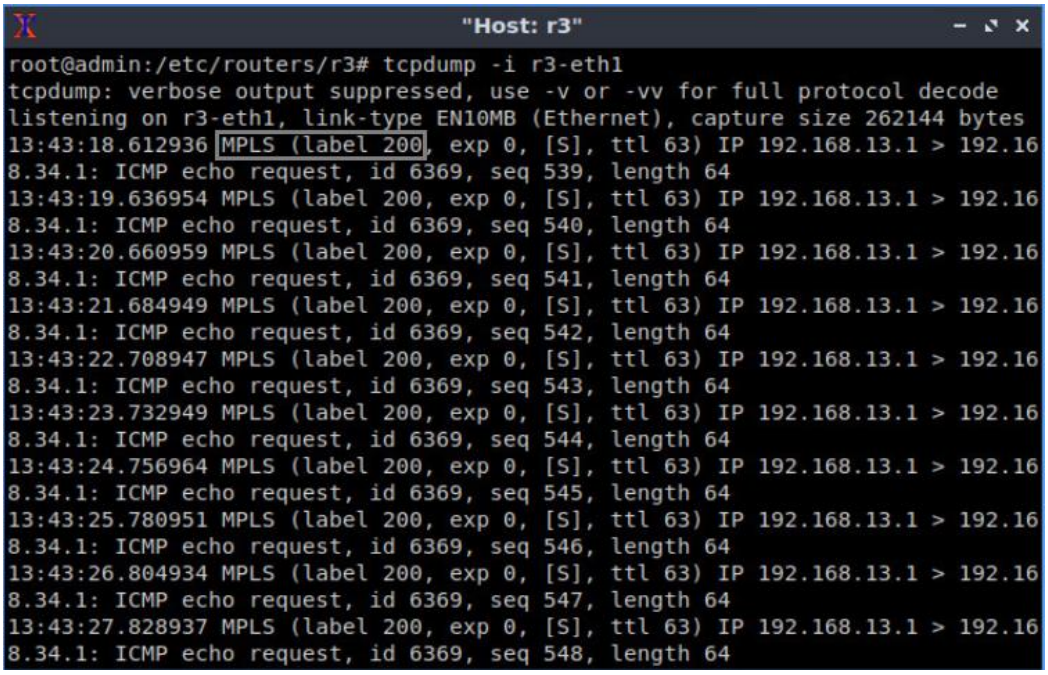

Figure 54. Verifying connectivity in router r3.

At this point, an MPLS packet can reach the network 192.168.34.0/30. Router r4 will not be able to connect to the network 192.168.45.0/30 as there is no label assigned for the network.

Step 11. To enable router r4 configuration mode, issue the following command:

```
configure terminal
```

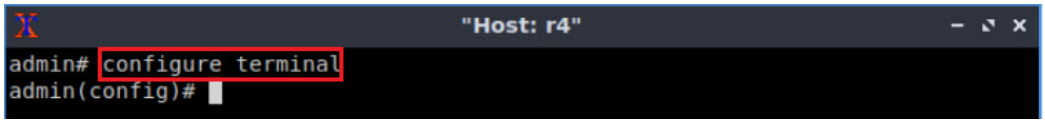

Figure 55. Enabling configuration mode in router r4.

Step 12. In this step, you will configure label swapping in router r4. Router r4 will receive label 200 from router r3 and swap the label with another one (300) to reach the interface *r5-eth0* (192.168.45.2). Type the following command to enable label swapping on router r4.

```
mpls lsp 200 192.168.45.2 300
```

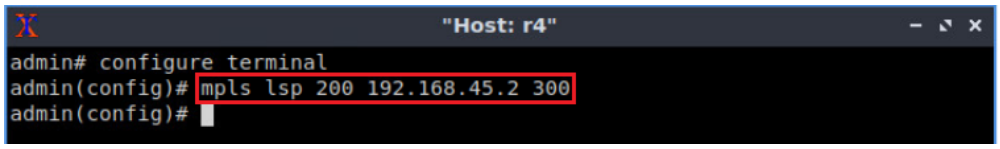

Figure 56. Enabling label swapping in router r4.

Step 13. Type the following command to exit configuration mode.

```
end
```

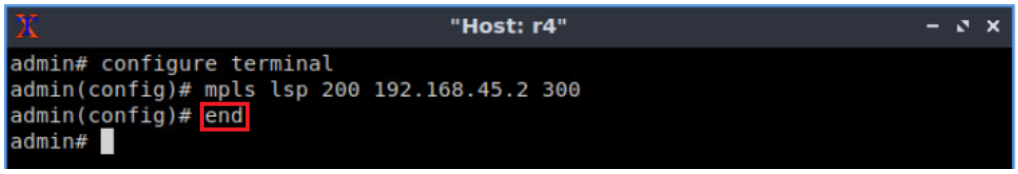

Figure 57. Exiting from configuration mode.

Step 14. In the following step, you will verify that router r1 can communicate with router r4 (interface *r4-eth1*). In router r4, type the following command to exit the vtysh session:

```
exit
```

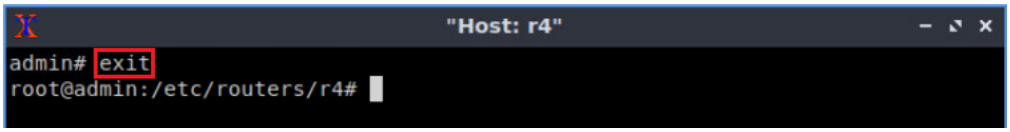

Figure 58. Exiting the vtysh session.

Step 15. Type the following command in order to capture and analyze network traffic received by the interface *r4-eth1*.

```
tcpdump -i r4-eth1
```

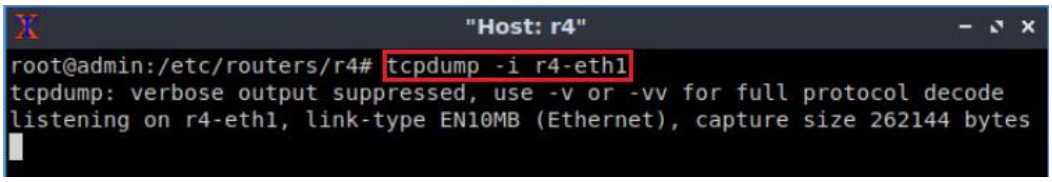

Figure 59. Enabling tcpdump to capture packets in router r4.

Step 16. Test the connectivity between router r1 and router r4 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r4's terminal. You will notice that router r4 is receiving packets from router r1.

```
ping 192.168.45.1
```

Figure 60. Output of the `ping` command on router r1.

Step 17. If you observe router r4, you will notice that router r1 (192.168.13.1) will communicate with router r4 (192.168.45.1) through MPLS routing. Router r1 uses label 100 to reach router r3 and router r3 swaps the label to 200 to reach network 192.168.34.0/30. Router r4 receives label 200 and swaps to label 300 to reach the network 192.168.45.0/30. To stop the test, press `Ctrl+c`.

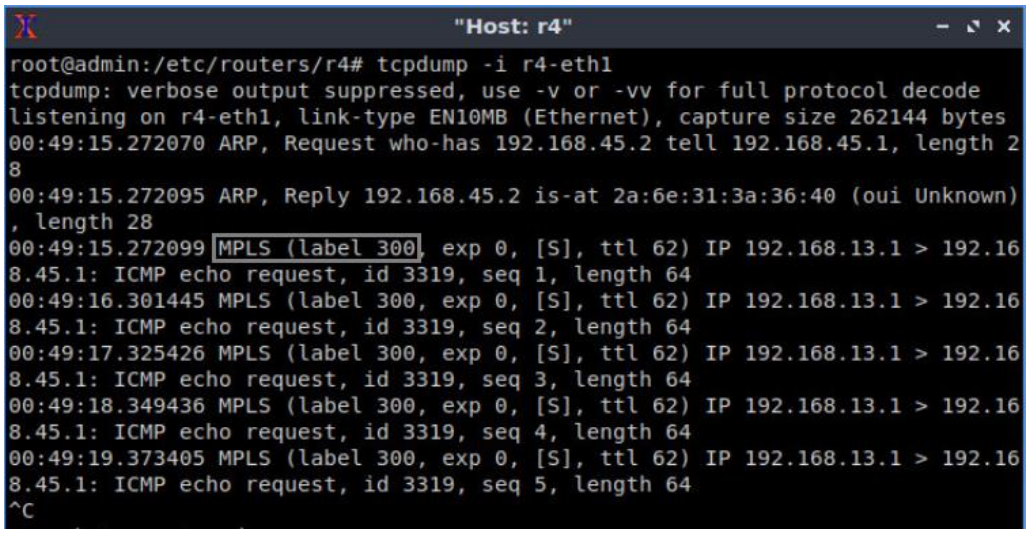

Figure 61. Verifying connectivity in router r4.

At this point, an MPLS packet can reach the network 192.168.45.0/30. Router r5 needs to assign label for the network 192.168.25.0/30.

Step 18. To enable router r5 configuration mode, issue the following command:

```
configure terminal
```

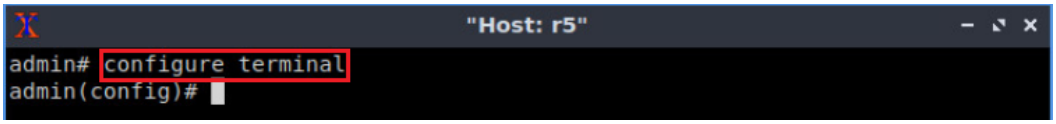

Figure 62. Enabling configuration mode in router r5.

Step 19. In this step, you will configure label swapping in router r5. Router r5 will receive label 300 from router r4 and swap the label with another one (220) to reach the interface *r2-eth1* (192.168.25.1). Type the following command to enable label swapping in router r4.

```
mpls lsp 300 192.168.25.1 220
```

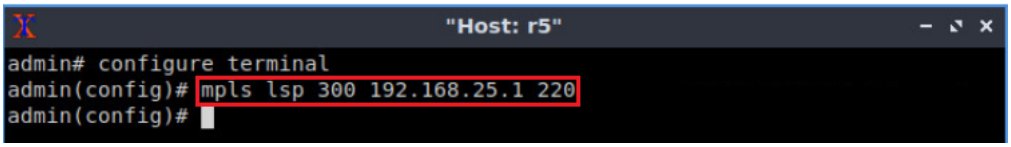

Figure 63. Enabling label swapping in router r5.

Step 20. Type the following command to exit configuration mode.

```
end
```

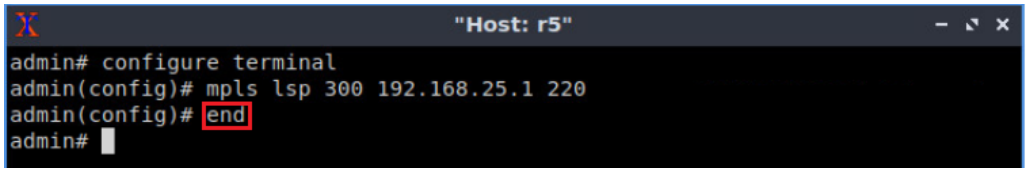

Figure 64. Exiting configuration mode.

Step 21. In the following step, you will verify that router r1 can communicate with router r5 (interface *r5-eth1*). In router r5, type the following command to exit the vtysh session:

```
exit
```

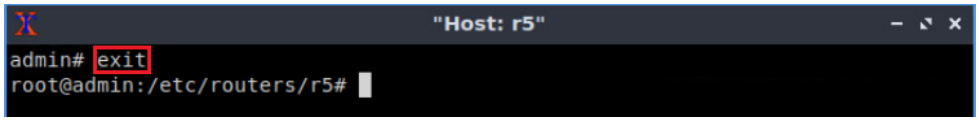

Figure 65. Exiting the vtysh session.

Step 22. Type the following command in order to capture and analyze network traffic received by the interface *r5-eth1*.

```
tcpdump -i r5-eth1
```

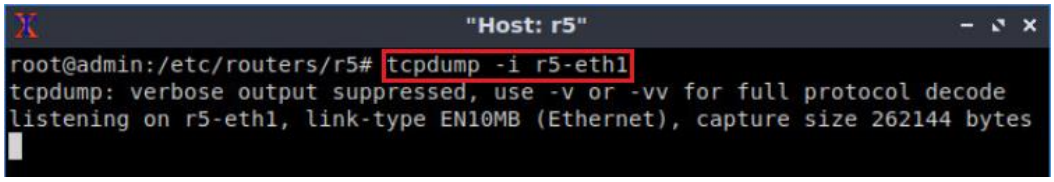

Figure 66. Enabling tcpdump to capture packets on router r5.

Step 23. Test the connectivity between router r1 and router r5 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r5's terminal. You will notice that router r5 is receiving packets from router r1.

```
ping 192.168.25.2
```

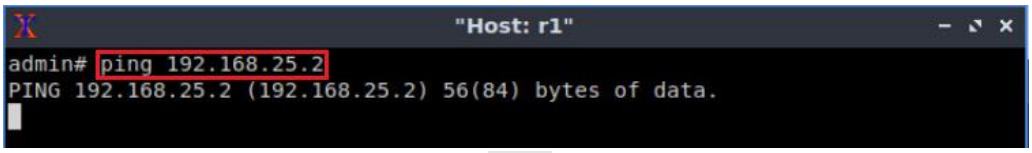

Figure 67. Output of the `ping` command on router r1.

Step 24. If you observe router r5, you will notice that router r1 (192.168.13.1) can communicate with router r5 (192.168.25.2) through MPLS routing. Router r1 uses label 100 to reach router r3 and router r3 swaps the label to 200 to reach network 192.168.34.0/30. Router r4 receives label 200 and swaps to label 300 to reach the network 192.168.45.0/30. Finally, router r5 swaps the label to 220 and delivers the packet to router r2. To stop the test, press `Ctrl+c`.

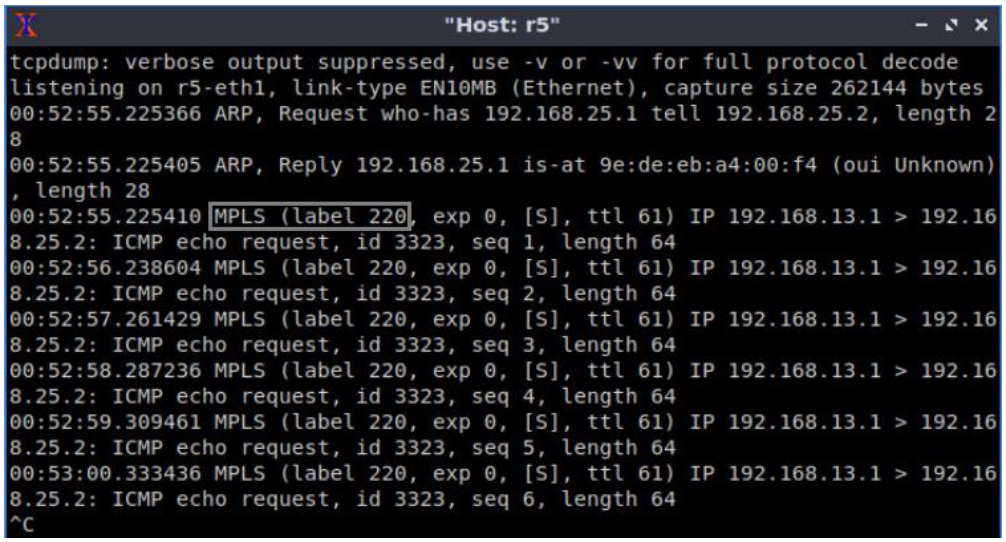

Figure 68. Verifying connectivity in router r5.

3.3 Popping labels

Step 1. To enable router r2 configuration mode, issue the following command:

```
configure terminal
```

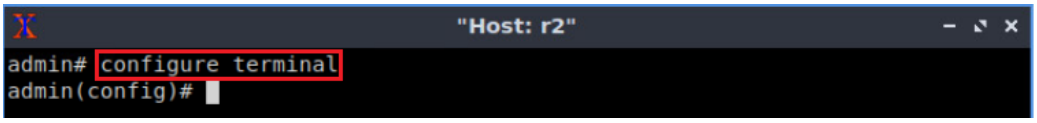

Figure 69. Enabling configuration mode in router r2.

Step 2. Type the following command to assign label 220 in router r2. The label is the same as the one router r2 received from router r5. Router r2 will pop the label (220) and the IP packet will be delivered to the destination host, h2 (192.168.2.10).

```
mpls lsp 220 192.168.2.10 implicit-null
```

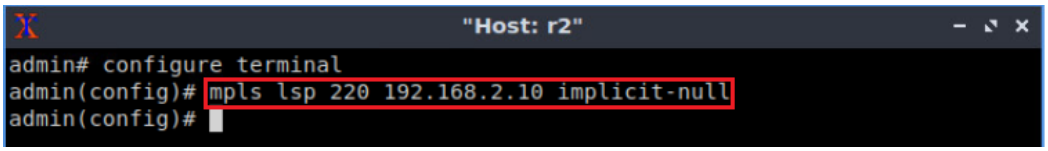

Figure 70. Popping the label in router r2.

The keyword *implicit-null* indicates that the router will not perform a swap; instead it performs a pop and delivers the IP packet to the destination.

Step 3. Type the following command to exit configuration mode.

```
end
```

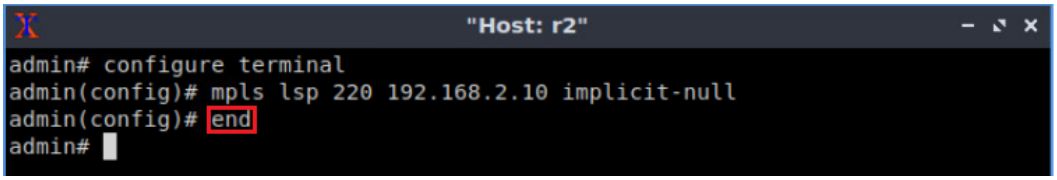

Figure 71. Exiting configuration mode.

Step 4. Type the following command to exit the vtysh session:

```
exit
```

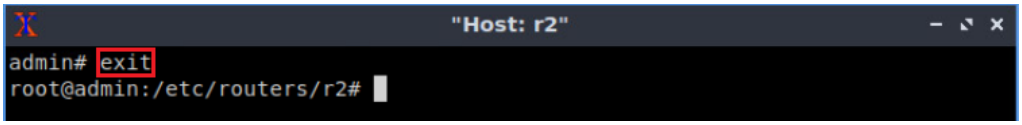

Figure 72. Exiting the vtysh session.

Step 5. Type the following command in order to capture and analyze network traffic received by the interface *r2-eth1*.

```
tcpdump -i r2-eth1
```

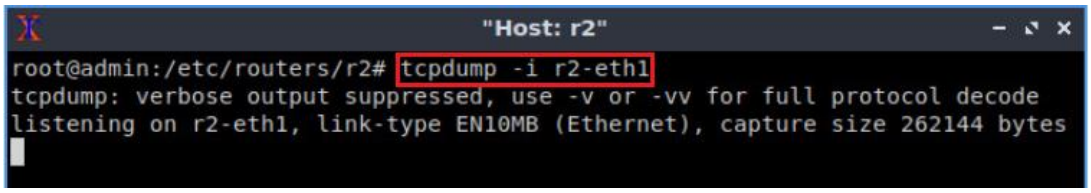

Figure 73. Enabling tcpdump to capture packets on router r2.

Step 6. Test the connectivity between router r1 and router r2 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r2 terminal. You will notice that router r2 is receiving packets from router r1.

```
ping 192.168.25.1
```

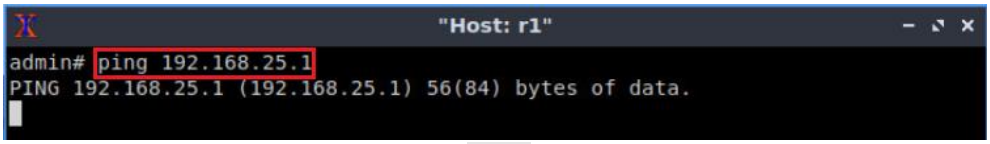

Figure 74. Output of the `ping` command on router r1.

Step 7. If you observe router r2, you will notice that label 220 is used to reach router r2. To stop the test, press `Ctrl+c`.

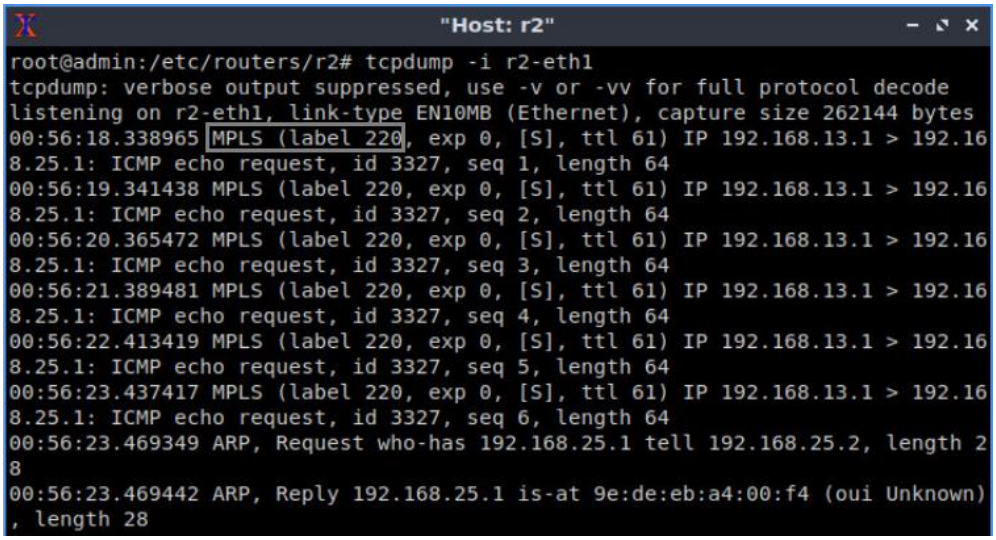

Figure 75. Verifying connectivity in router r2.

Step 8. Type the following command in order to capture and analyze network traffic received by the interface *r2-eth0*.

```
tcpdump -i r2-eth0
```

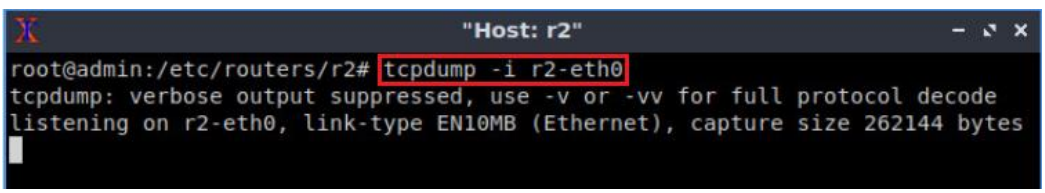

Figure 76. Enabling tcpdump to capture packets on router r2.

Step 9. Test the connectivity between router r1 and router r2 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r2 terminal. You will notice that router r2 is receiving packets from router r1.

```
ping 192.168.2.1
```

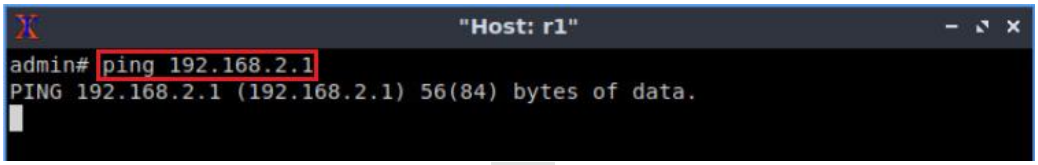

Figure 77. Output of the `ping` command on router r1.

Step 10. If you observe router r2, you will notice that router r1 (192.168.13.1) can communicate with router r2 (192.168.2.1). You will notice there is no MPLS label attached to it as router r2 popped the label and delivered the IP packet to the destination.

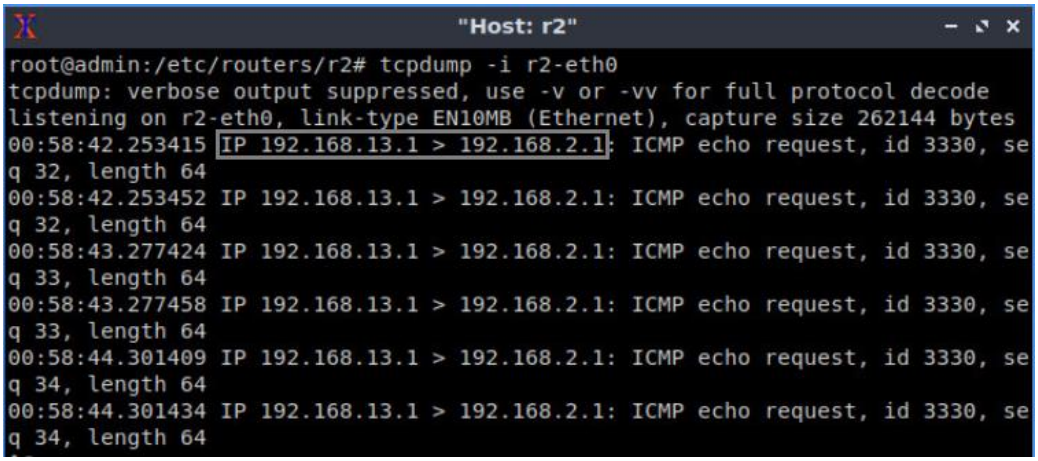

Figure 78. Verifying connectivity in router r2.

Step 11. Test the connectivity between router r1 and host h2 using the `ping` command. In router r1, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r2 terminal. You will notice that router r2 is receiving packets from router r1.

```
ping 192.168.2.10
```

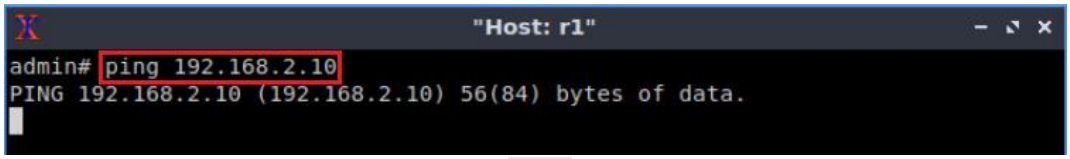

Figure 79. Output of the `ping` command on router r1.

Step 12. If you observe router r2, you will notice that router r1 (192.168.13.1) can finally reach the destination host, h2 (192.168.2.10).

```
tcpdump -i r2-eth0
```

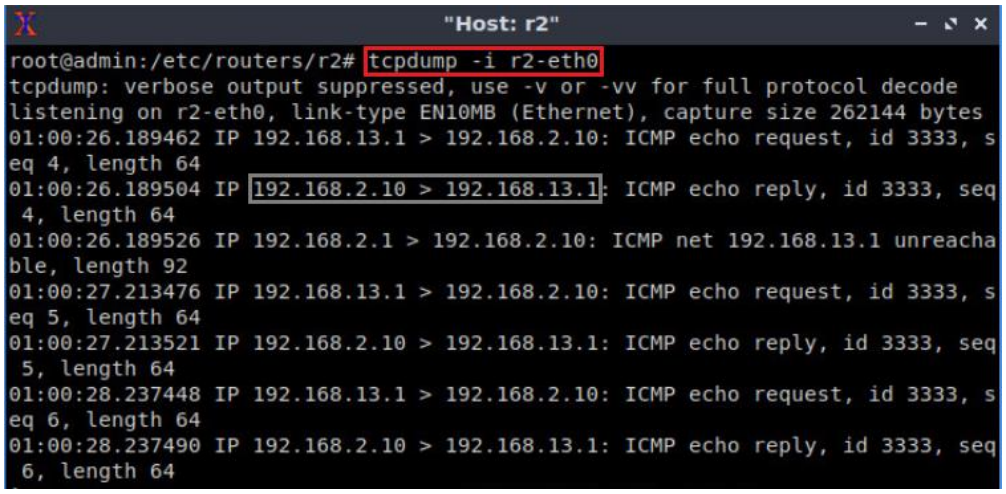

Figure 80. Verifying connectivity in router r2.

Step 13. Test the connectivity between host h1 and host h2 using the `ping` command. In host h1, type the command specified below. To stop the test, press `Ctrl+c`.

```
ping 192.168.2.10
```

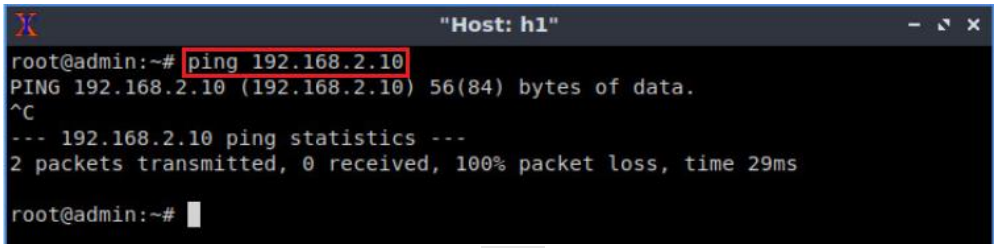

Figure 81. Output of the `ping` command on host h2.

Consider Figure 81. Host h1 cannot ping host h2 as the IP packet reaches host h2 but cannot reply back to host h1 as router r2 does not have the route back to router r1.

4 Configuring static MPLS from router r2 to router r1

In this section, you will configure static MPLS so that router r2 can communicate with all the routers and both the hosts, h1 and h2 can ping each other.

Step 1. Router r2 has a route to the network 192.168.25.0/30. Configure static routing with label 400 in router r1 in order to get the routes to other networks. All the steps are summarized in the following figure.

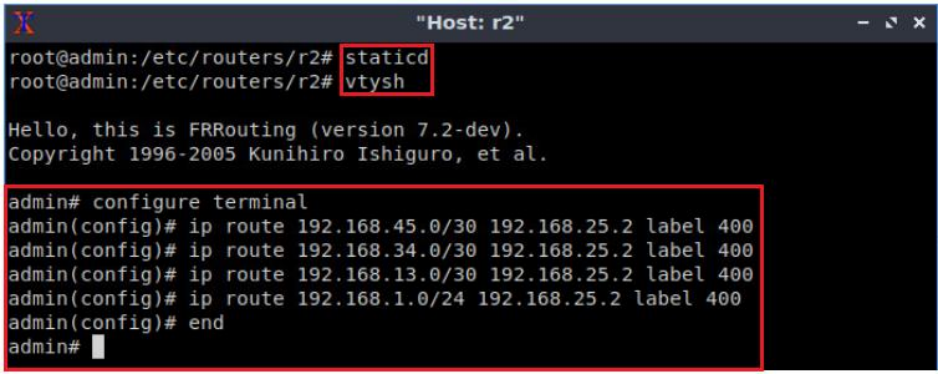

Figure 82. Pushing labels in router r2.

Step 2. Type the following commands in router r5. Router r5 will receive label 400, swap the label to 300 which will be delivered to router r4 (192.168.45.1). All the commands are summarized in the following figure.

```
mpls lsp 400 192.168.45.1 300
```

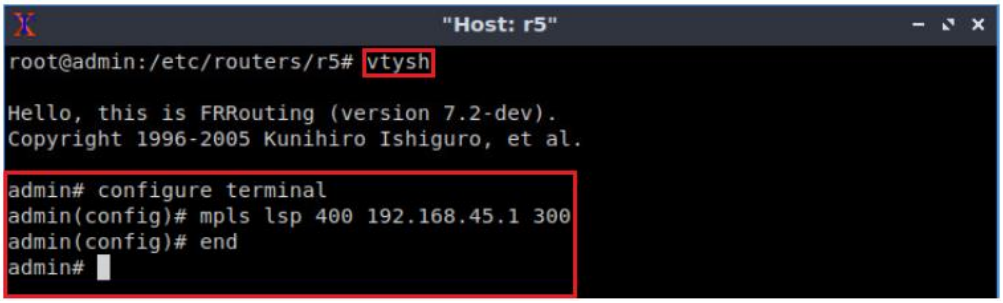

Figure 83. Enabling label swapping in router r5.

Step 3. Type the following commands in router r4 terminal. Router r4 will receive label 300, swap the label to 200 which will be delivered to router r3 (192.168.34.1). All the commands are summarized in the following figure.

```
mpls lsp 300 192.168.34.1 200
```

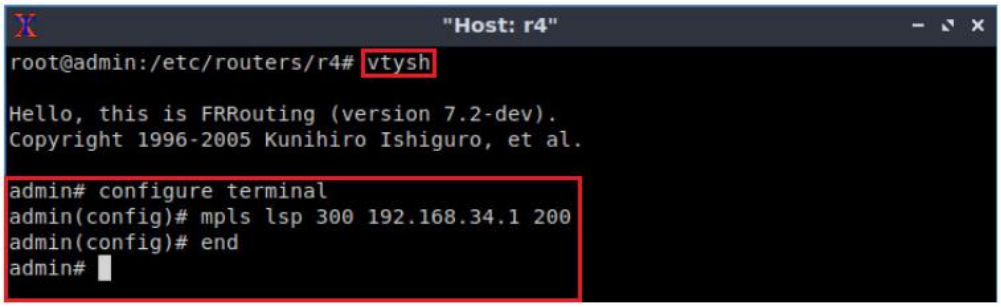

Figure 84. Enabling label swapping in router r4.

Step 4. Type the following commands in router r3 terminal. Router r3 will receive label 200, swap the label to 110 which will be delivered to router r1 (192.168.13.1). All the commands are summarized in the following figure.

```
mpls lsp 200 192.168.13.1 110
```

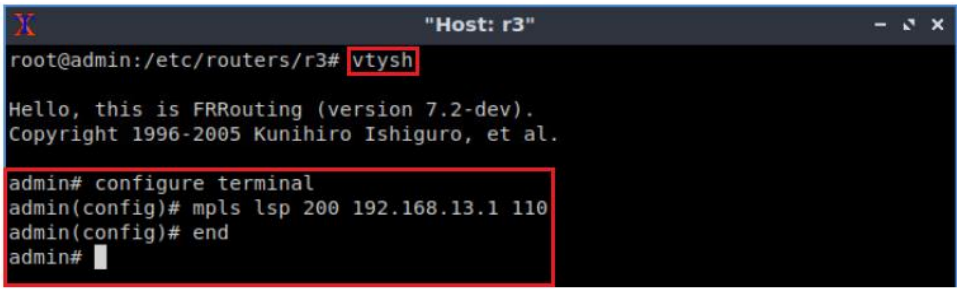

Figure 85. Enabling label swapping in router r3.

Step 5. Type the following commands to assign label 110 to router r1. The label is the same as the one router r1 received from router r3. Router r1 will pop the label (110) and the IP packet will be delivered to the destination host, h1 (192.168.1.10).

```
mpls lsp 110 192.168.1.10 implicit-null
```

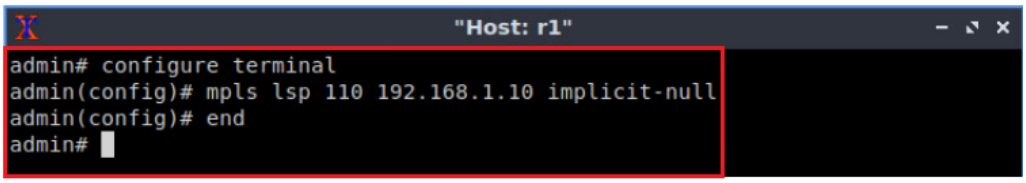

Figure 86. Popping the label in router r1.

5 Verifying configuration

In this section, you will verify the MPLS configuration and the connectivity between two hosts.

Step 1. You will verify MPLS configuration in router r3. Router r3 uses the MPLS label 200 to reach interface *r3-eth1* (192.168.34.1). Type the following command to exit the vtysh session in router r3:

```
exit
```

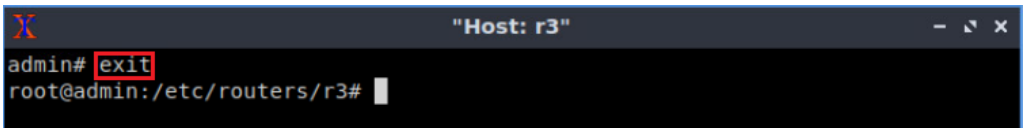

Figure 87. Exiting configuration mode.

Step 2. Type the following command in order to capture and analyze network traffic received by the interface *r3-eth1*.

```
tcpdump -i r3-eth1
```

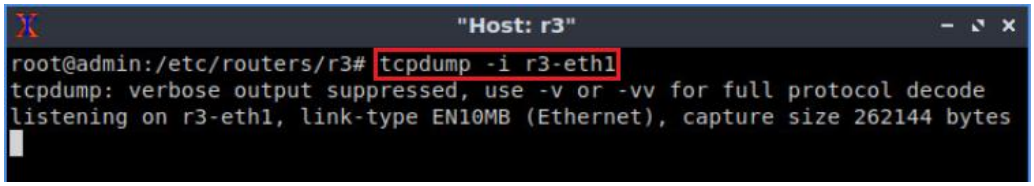

Figure 88. Enabling tcpdump to capture packets in router r3.

Step 3. Test the connectivity between router r2 and router r3 using the `ping` command. In router r2, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity in router r3's terminal. You will notice that router r3 is receiving packets from router r2.

```
ping 192.168.34.1
```

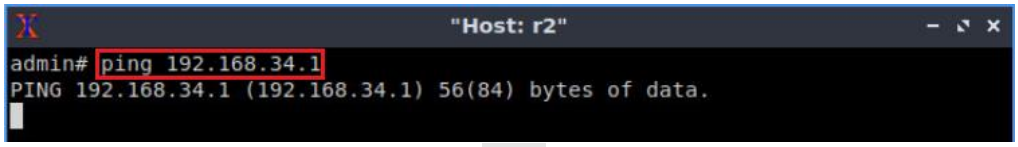

Figure 89. Output of the `ping` command on router r2.

Step 4. If you observe router r3, you will notice that router r2 (192.168.25.1) can communicate with router r3 (192.168.34.1) through MPLS routing. Router r2 uses label 400 to reach router r5 and router r5 swaps the label to 300 to reach router r4. Router r4 uses label 200 to reach network 192.168.34.0/30. To stop the test, press `Ctrl+c`.

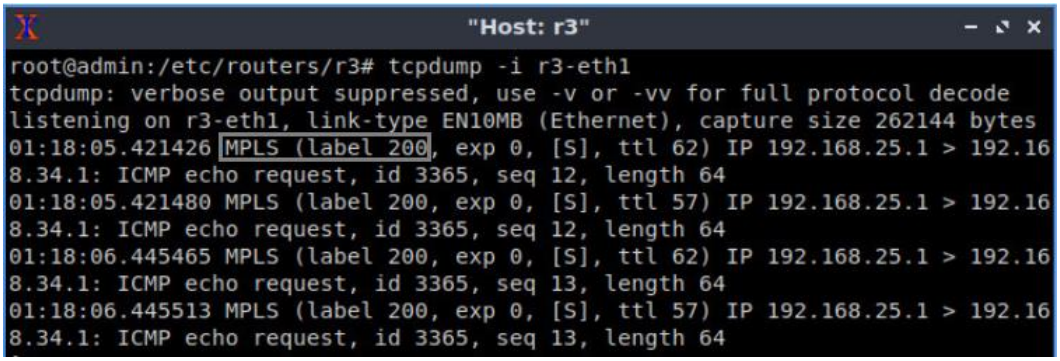

Figure 90. Verifying connectivity in router r3.

Step 5. You can also verify the configuration through wireshark. Type the following command in router r3 to open wireshark.

```
wireshark
```

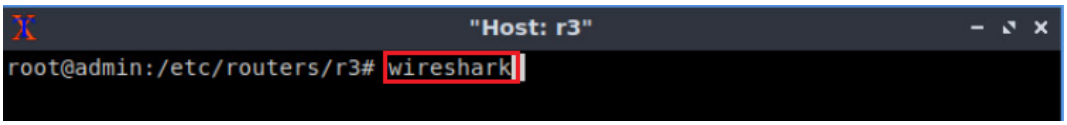

Figure 91. Opening wireshark in router r3.

Step 6. Click on *r3-eth1* in order to see all the traffic received by the interface *r3-eth1*.

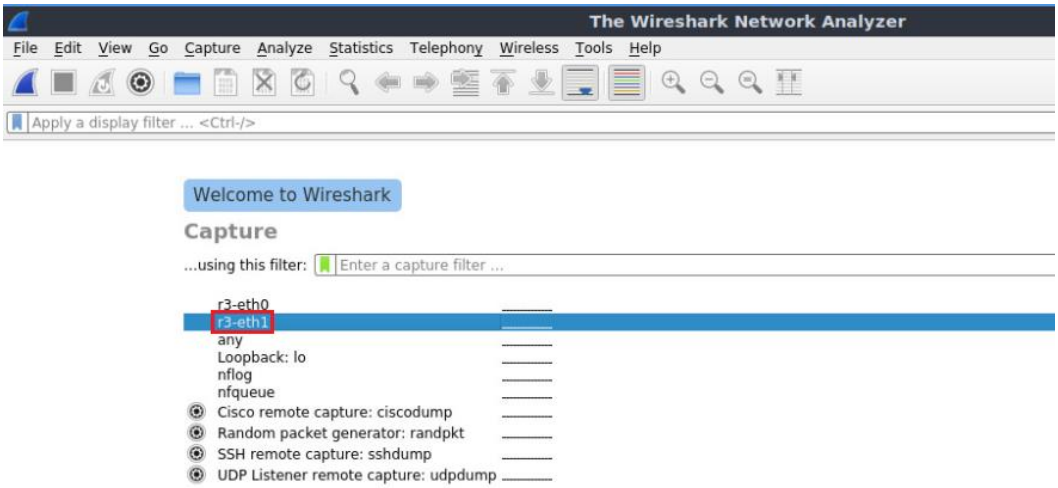

Figure 92. Selecting interface *r3-eth1* in wireshark.

Step 7. Test the connectivity between router r2 and router r3 using the `ping` command. In router r2, type the command specified below. To stop the test, press `Ctrl+c`. Verify the connectivity on wireshark.

```
ping 192.168.34.1
```

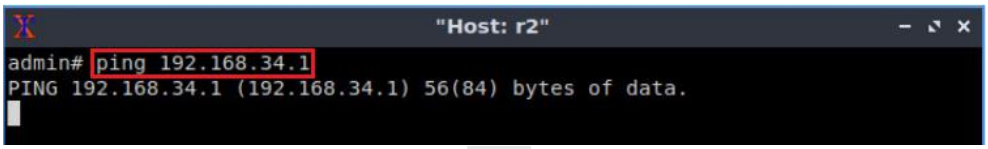

Figure 93. Output of the `ping` command on router r2.

Step 8. If you verify packets on wireshark, you will notice the source address, 192.168.25.1 can ping to the destination address, 192.168.34.1. You can also notice the MPLS header used to reach the destination, 192.168.34.1 (label: 200, Exp: 0, S: 1, TTL: 62).

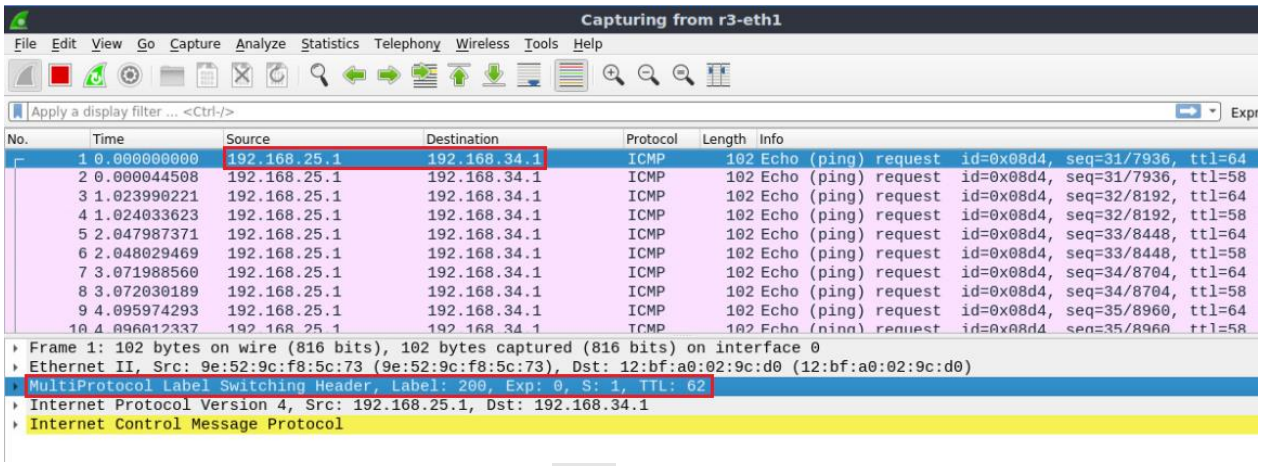

Figure 94. Output of the `ping` command in wireshark.

Step 9. Type the following command to display the MPLS table of router r1. You will notice the label assigned to router r1.

```
show mpls table
```

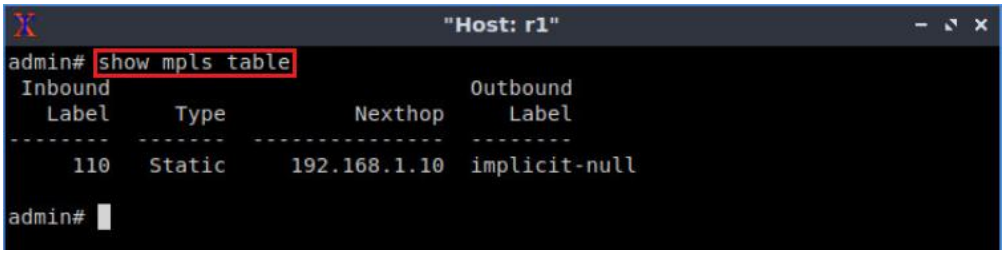

Figure 95. Displaying the MPLS table of router r1.

Step 10. Type the following command to display the routing table (control plane) of router r1. The control plane makes use of a label exchange protocol to create and maintain labels internally, and to exchange these labels with other devices.

```
show ip route
```

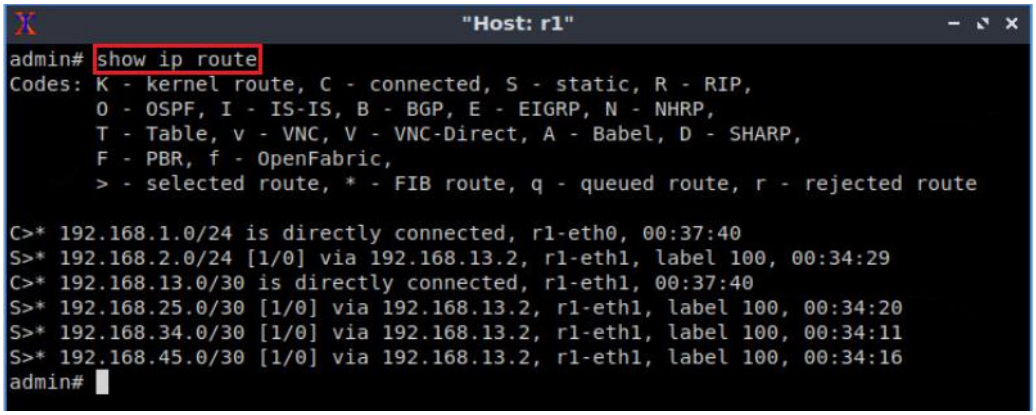

Figure 96. displaying the routing table of router r1.

Step 11. Type the following command to exit the vtysh session:

```
exit
```

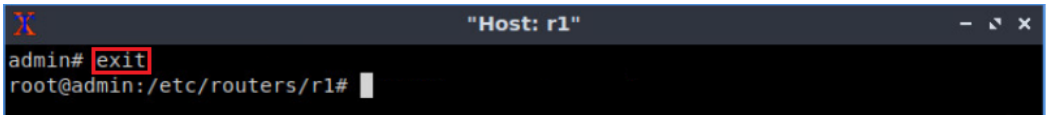

Figure 97. Exiting the vtysh session.

Step 12. Type the following command to display the kernel routes (data plane) in router r1. The data plane indicates how the data will flow from router r1 (192.168.1.0/24) to router r2 (192.168.2.0/24).

```
route
```

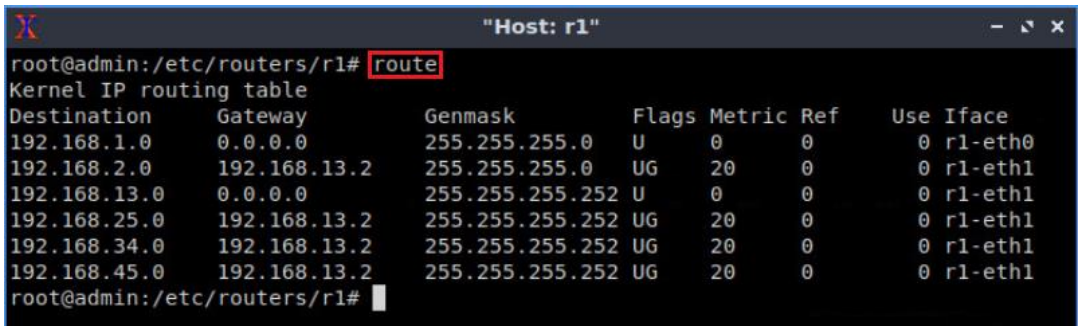

Figure 98. Displaying kernel routes in router r1.

Consider Figure 98. An IP lookup for the destination, 192.168.2.0/24 will be examined at this point. Other routers (r2, r3, r4 and r5) will forward the packet according to label information. The label information is removed at the exit router (r2), and the IP packet will be delivered to the destination, 192.168.2.10.

Step 13. Test the connectivity between host h1 and host h2 using the `ping` command. In host h1, type the command specified below. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test.

```
ping 192.168.2.10
```

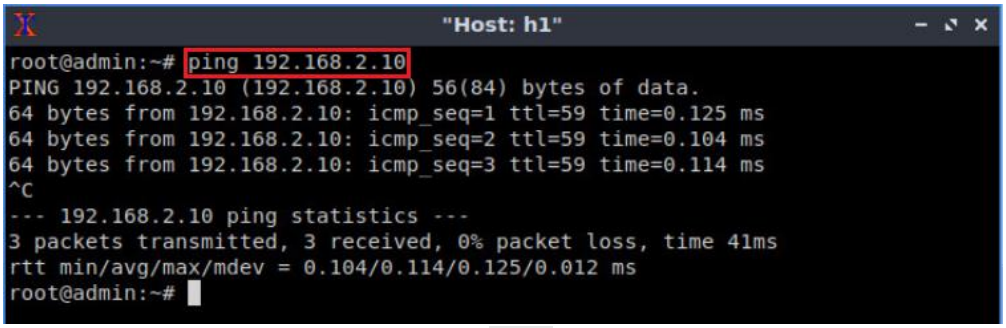

Figure 99. Output of the `ping` command in host h2.

This concludes Lab 3. Stop the emulation and then exit out of MiniEdit.

References

1. Paul Goransson, Chuck Black, "*Software Defined Networks: A Comprehensive Approach*", 2014.
2. Luc De Ghein, "*MPLS Fundamentals*", 1st Edition, Pearson, 2006.
3. Benzekki, Kamal; El Fergougui, Abdeslam; Elbelrhiti Elalaoui, Abdelbaki, "*Software-defined networking (SDN): A survey*", 2016.
4. Juniper, "*MPLS Label Distribution Protocols Overview*", 2018. [Online]. Available: https://www.juniper.net/documentation/en_US/junose15.1/topics/concept/mpls-label-distribution-protocols.html

5. Linux foundation collaborative projects, "*FRR Routing documentation*", 2017. [Online]. Available: http://docs.frrouting.org/en/latest/zebra.html#mpls-commands
6. Juniper, "*Understanding MPLS Label Operations*", 2020. [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/concept/mpls-label-operations-qfx-series.html
7. B. Lantz, G. Gee, "*MiniEdit: a simple network editor for Mininet*," 2013. [Online]. Available: https://github.com/Mininet/Mininet/blob/master/examples.

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 4: Introduction to SDN

Document Version: 05-28-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab is an introduction to Software Defined Networking (SDN), a new networking paradigm that overcomes several limitations of the current network infrastructure. In this lab, we will introduce the components of SDN and describe how they operate. The focus in this lab is to gain in depth knowledge about the role of the control plane and the data plane.

Objectives

By the end of this lab, the user should be able to:

1. Understand the concept of SDN.
2. Enable ONOS controller.
3. Navigate through ONOS environment.
4. Activate basic ONOS applications and understand their effects.
5. Understand flow tables in SDN devices.
6. Perform a connectivity test.
7. Visualize topology information in the GUI dashboard.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Starting ONOS
4. Section 4: Navigating ONOS GUI.

1 Introduction

In just a few years, SDN has created enormous interest in academia and industry. An open, vendor neutral, control-data plane interface such as OpenFlow[1] allows network hardware

and software to evolve independently. Furthermore, it facilitates the replacement of expensive, proprietary hardware and firmware with commodity hardware and a free, open source Network Operating System (NOS). By managing network resources and providing high-level abstractions and APIs for interacting with, managing, monitoring, and programming network switches, the NOS provides an open platform that simplifies the creation of innovative and beneficial network applications and services that work across a wide range of hardware[2].

1.1 Introduction to SDN

Traditional IP networks depend on distributed routing protocols running inside the routers to exchange routing information. Despite their widespread adoption, traditional IP networks are complicated by their nature, and they are not trivial to be managed. For example, network administrators need to configure each network separately using low-level, vendor-specific commands. A traditional networking device bundles the control plane (that decides how to handle network traffic) and the data plane (that forward network traffic). Thus, reducing the flexibility and hindering innovation and evolution of the networking infrastructure[3].

SDN is an emerging networking paradigm that gives hope to change the limitations of current network infrastructures[3]. It breaks the control plane from the data plane and implements each separately. The disaggregation of the control plane and the data plane allows network switches to become simple forwarding devices and the control logic to be implemented in a logically centralized controller. Thus, amplifying the flexibility in the network, breaking the network control problem into tractable pieces, introducing new abstractions, and facilitating network evolution and innovation[3].

Consider Figure 1. the vast majority of packets handled by the switch are only managed by the data plane. The latter is composed of ports used to receive and transmit the packets, as well as a forwarding table that instructs the switch on how to deal with incoming packets. The data plane is responsible for packet buffering, packet scheduling, header modification, and forwarding[4].

Some packets cannot be processed by the data plane directly, for example, their information is not yet inserted in the forwarding table. Such packets are forwarded to the control plane, which lies on top of the data plane. The control plane involves many activities, mainly to maintain the forwarding table of the data plane. Essentially, the control plane is responsible for processing different control protocols that may affect the forwarding table.

SDN applications run on top of the controller. They are ultimately responsible for managing the flow table on the network devices (data plane). For example, route packets through the best path between two endpoints, or balance traffic loads across multiple paths[4].

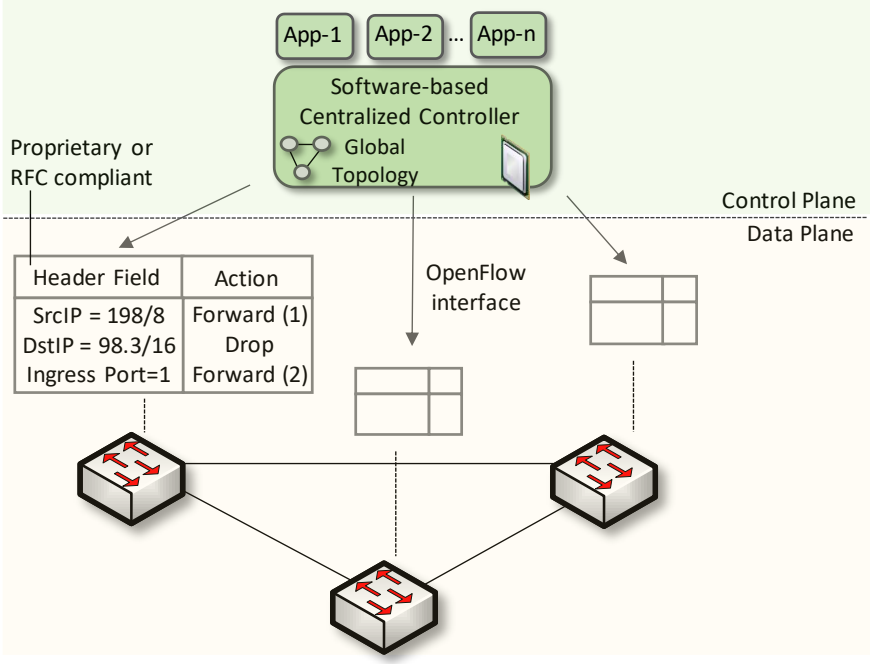

Figure 1. The control plane is embedded in a centralized server and it is decoupled from data plane devices in SDN networks.

1.2 SDN data plane architecture

Consider Figure 2. An SDN device is composed of an Application Programming Interface (API) for communication with the controller, an abstraction layer, and a packet-processing function.

The standard API used to communicate with the controller is OpenFlow[1], or it could be some proprietary alternative in certain SDN solutions[4].

The abstraction layer embodies one or more flow tables, which are the fundamental data structures in an SDN device. These flow tables allow the device to evaluate incoming packets and take the appropriate action. Flow tables consist of a number of flow entries, each of which typically consists of two components: match fields and actions. Match fields are used to compare against incoming packets, for example, matching against the Media Access Control (MAC) address, User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) port. Actions are the instructions that the forwarding device should perform if an incoming packet matches a flow entry. These actions may include forwarding the packet to a specific port, dropping the packet, or flooding the packet on all ports, among others[4].

The packet-processing logic consists of the mechanisms to take actions based on the results of evaluating incoming packets. In a hardware switch, these mechanisms are implemented by specialized hardware, such as Ternary Content Addressable Memory (TCAM) and Content Addressable Memory (CAM)[4].

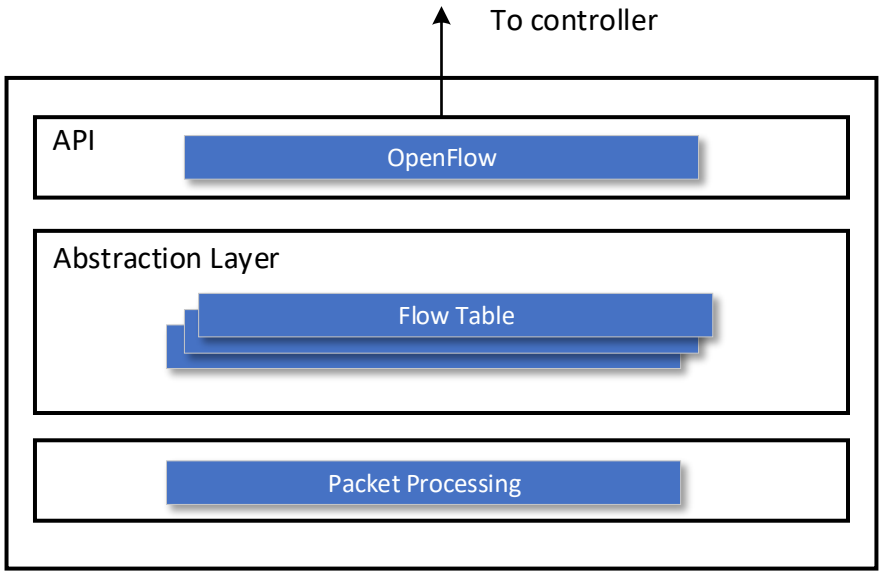

Figure 2. SDN switch architecture.

1.3 SDN controller architecture

A controller keeps a view of the entire network, implements policy decisions, controls all the SDN devices that comprise the network infrastructure, and provides a northbound API for the application. Also, controllers implement policy decisions about routing, forwarding, redirecting, and load balancing. Controllers often come with their own set of common application modules, such as a learning switch, a router, a basic firewall, and a simple load balancer. Such features are really SDN applications. However, they are often bundled with the controller[4].

Figure 3 depicts the modules that provide the controller's core functionality, both a northbound and southbound API, and a few sample applications. The southbound API is used to interface with the SDN device. Commonly, this API is OpenFlow. The controller abstracts the details of the SDN controller-to-device protocol so that the applications such as the GUI, learning switch, router and others can transparently communicate with the SDN devices. Every controller provides core functionality between these raw interfaces. Core features in the controller include[4]:

- End-user device discovery: discovery of end-user devices, such as laptops, desktops, printers, mobile devices, etc.
- Network device discovery: discovery of network devices which comprise the infrastructure of the network, such as switches, routers, and wireless access points.
- Network device topology management: maintain information about the interconnection details of the network devise to each other, and to the end-user devices to which they are directly attached.
- Flow management: maintain a database of the flows being managed by the controller and perform all necessary coordination with the devices to ensure synchronization of the device flow entries with that database.

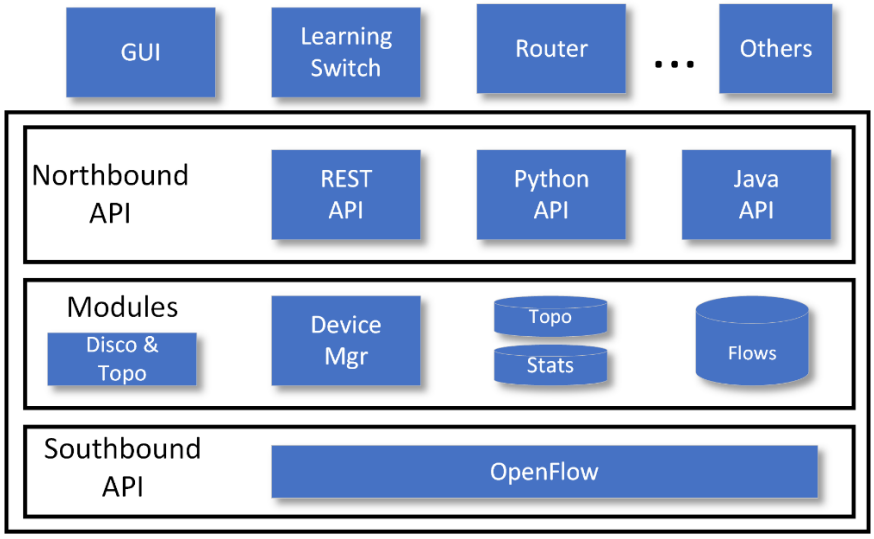

Figure 3. SDN controller architecture.

2 Lab topology

Consider Figure 4. The topology consists of four end-hosts, three switches and a controller. The blue devices represent OpenFlow switches. All switches are connected to the controller c0.

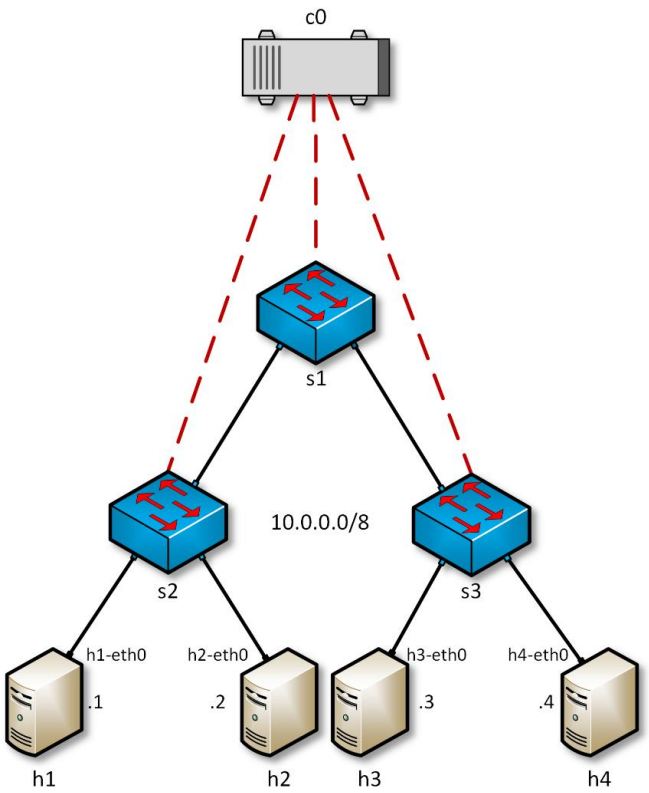

Figure 4. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet
h1	h1-eth0	10.0.0.1	/8
h2	h2-eth0	10.0.0.2	/8
h3	h3-eth0	10.0.0.3	/8
h2	h4-eth0	10.0.0.4	/8
c0	n/a	127.0.0.1	/32

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet[5]. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), save the topology and export a layer 2 model.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

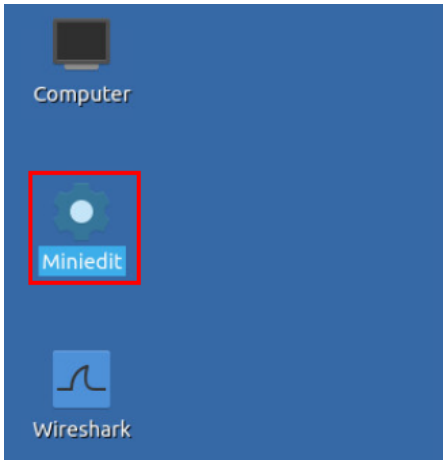

Figure 5. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab4.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab4* and click on *Open*.

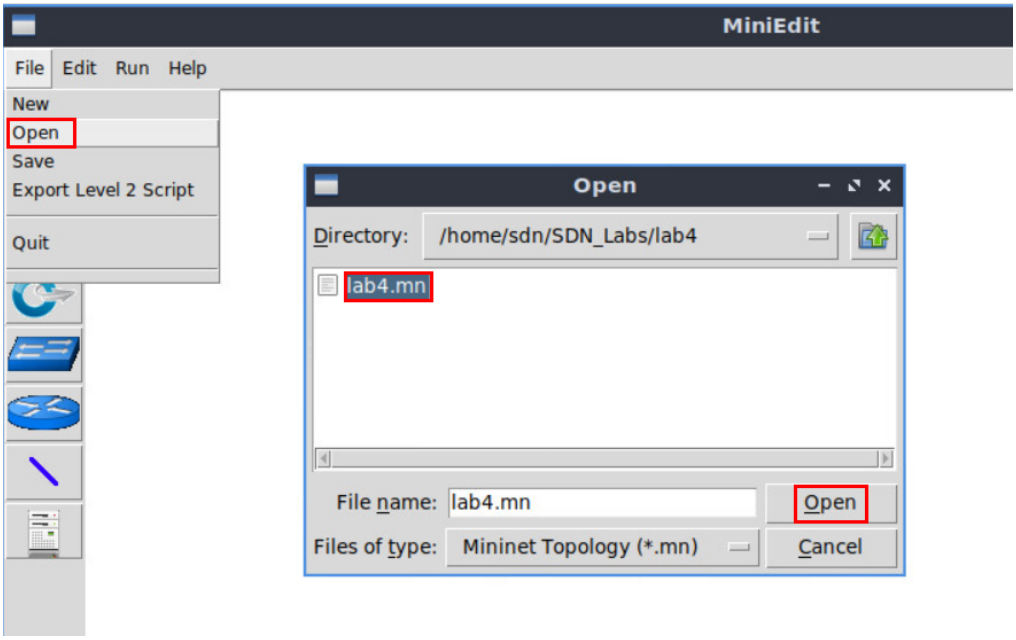

Figure 6. Opening topology.

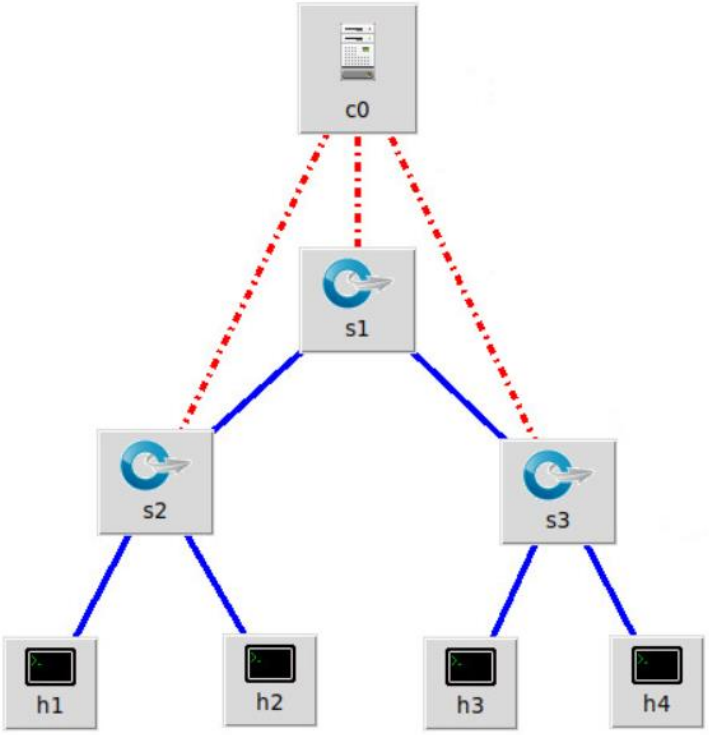

Figure 7. MiniEdit's topology.

Step 3. Click on the *Run* button to start the emulation. The emulation will start and the buttons of the MiniEdit panel will gray out, indicating that they are currently disabled.

Figure 8. Starting the emulation.

3 Starting ONOS

Step 1. Open Linux terminal by clicking on the shortcut depicted below.

Figure 9. Opening Linux terminal.

Step 2. Navigate into *SDN_Labs/lab4* directory by issuing the following command. This folder contains the script responsible for starting ONOS. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab4
```

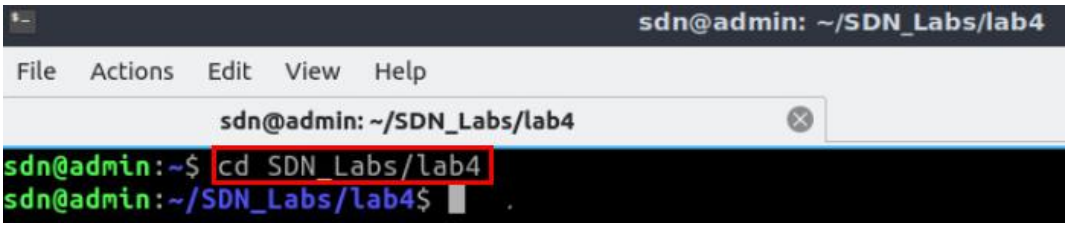

Figure 10. Entering the *SDN_Labs/lab4* directory.

Step 3. A script was written to run ONOS and enter its Command Line Interface (CLI). In order to run the script, issue the following command.

```
./run_onos.sh
```

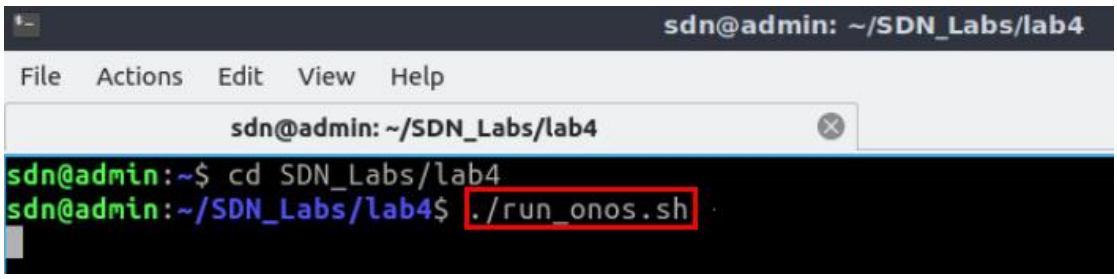

Figure 11. Starting ONOS.

Once the script finishes executing and ONOS is ready, you will be able to execute commands on ONOS CLI as shown in the figure below. Note that this script may take a couple of minutes.

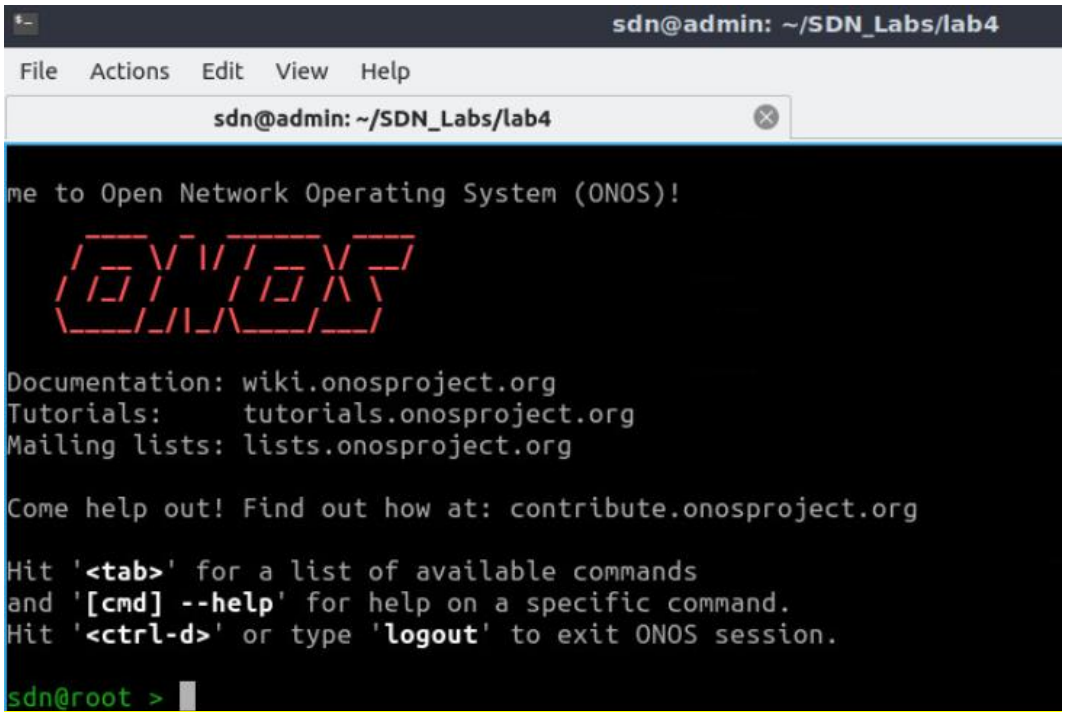

Figure 12. ONOS CLI.

Step 4. ONOS supplies a set of its own commands, in order to list all available commands, click the TAB key.

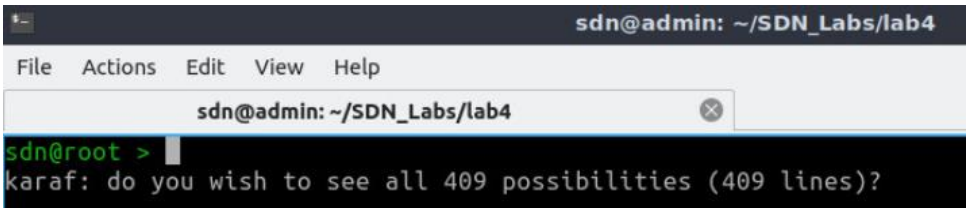

Figure 13. Displaying a list of ONOS commands.

Step 5. To confirm displaying all 401 possibilities of ONOS commands, click the TAB key one more time. This will display all ONOS commands on the left-hand side of the CLI, as well as their explanation on the right-hand side of the CLI.

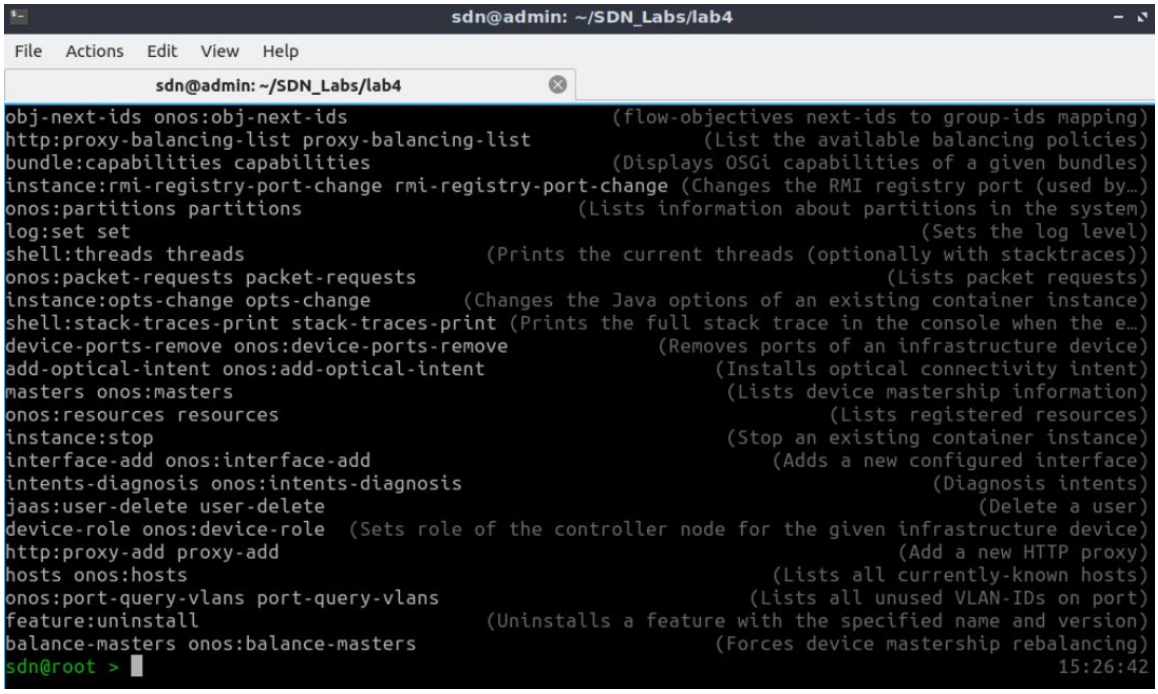

Figure 14. Displaying a list of ONOS commands.

Step 6. To display the list of all currently known flows for the ONOS controller, type the following command.

```
flows
```

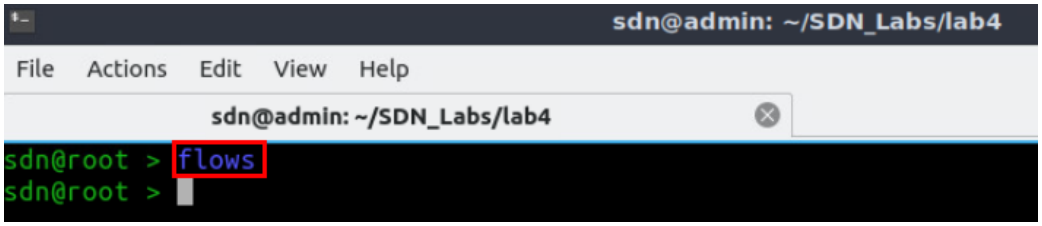

Figure 15. Displaying the current known flows.

Step 7. To display the list of all currently known devices (OVS switches), type the following command.

```
devices
```

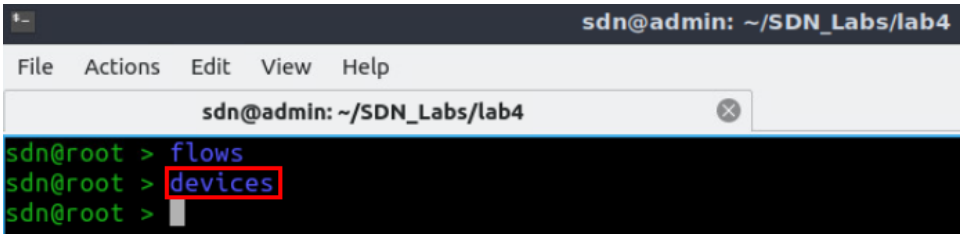

Figure 16. Displaying the current known devices (switches).

Step 8. To display the list of all currently known hosts, type the following command.

```
hosts
```

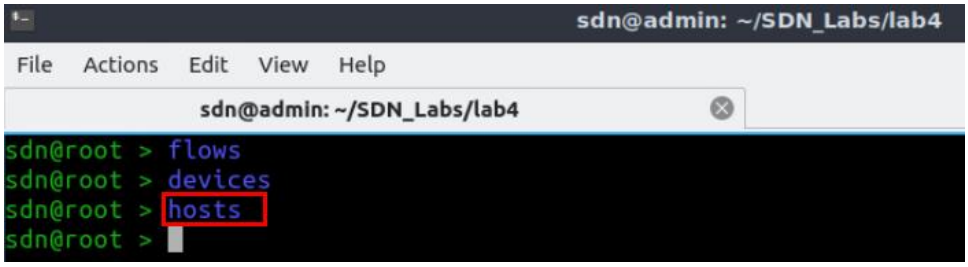

Figure 17. Displaying the current known hosts.

Step 9. To display the list of all currently known links, type the following command.

```
links
```

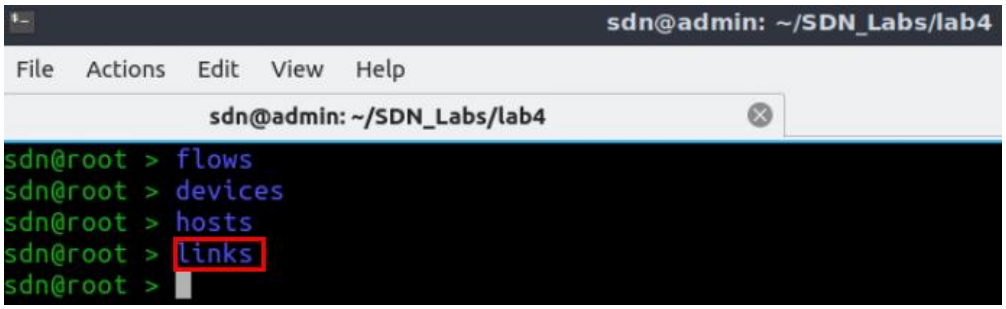

Figure 18. Displaying the current known link.

Consider Figures 15, 16, 17, and 18. No flows, devices, hosts, or link are displayed since we have not activated the necessary applications that allow the controller to discover them.

3.1 Activating ONOS OpenFlow application

In this section you will activate OpenFlow application that comes with ONOS and allows to speak OpenFlow protocol with the devices. You will notice how the Mininet topology becomes visible, i.e., the devices (switches), and hosts are now recognized by ONOS.

Step 1. In ONOS terminal, issue the following command to activate the OpenFlow application.

```
app activate org.onosproject.openflow
```

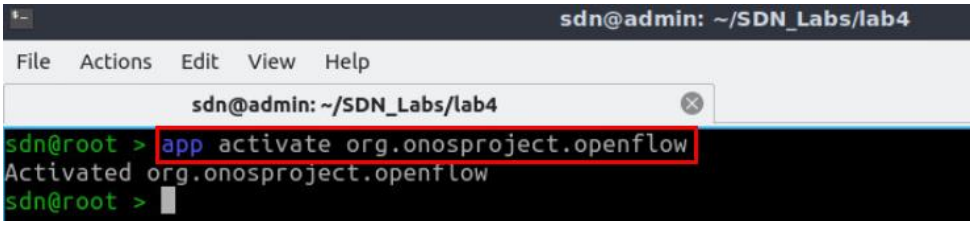

Figure 19. Activating OpenFlow application.

Step 2. Open Mininet terminal.

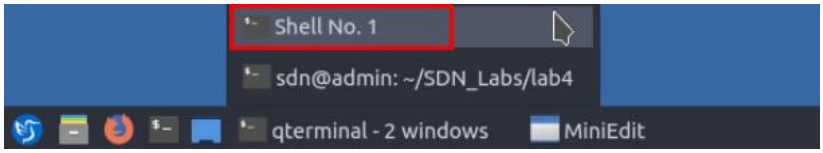

Figure 20. Opening Mininet terminal.

Step 3. Once the OpenFlow application is activated, you might not get accurate results immediately, for example, not all the hosts are discovered yet. In order to stimulate ONOS and the activated application, perform a `pingall` command. This command pings from every host in the network to all other hosts. To do so, write the following command.

```
pingall
```

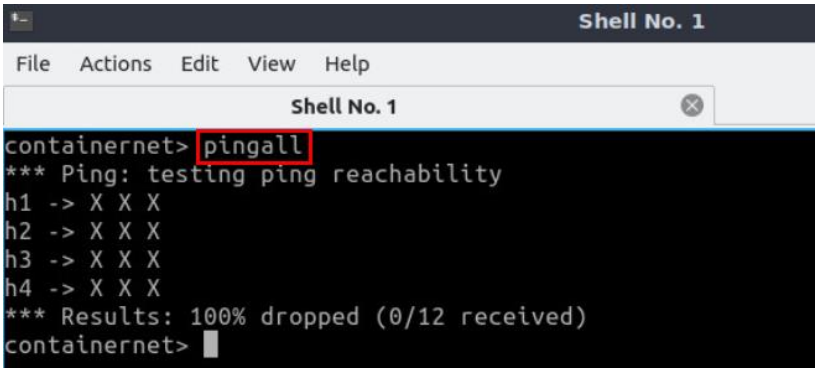

Figure 21. Pinging all hosts to stimulate host discovery in the Controller.

Consider Figure 21. The connectivity test resulted in 100% dropped. The pings sent from a host to a destination are IP packets received by a switch. For example, pinging host h2 from h1 is received by switch s1. Since there are no flows inserted that deal with IP packets, then the packets will be dropped immediately.

Step 4. Go to ONOS terminal.

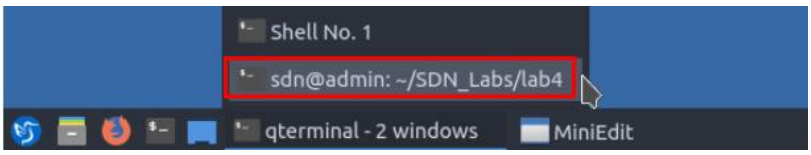

Figure 22. Opening ONOS terminal.

Step 5. To display the list of all currently known devices (OVS switches), type the following command.

```
devices
```

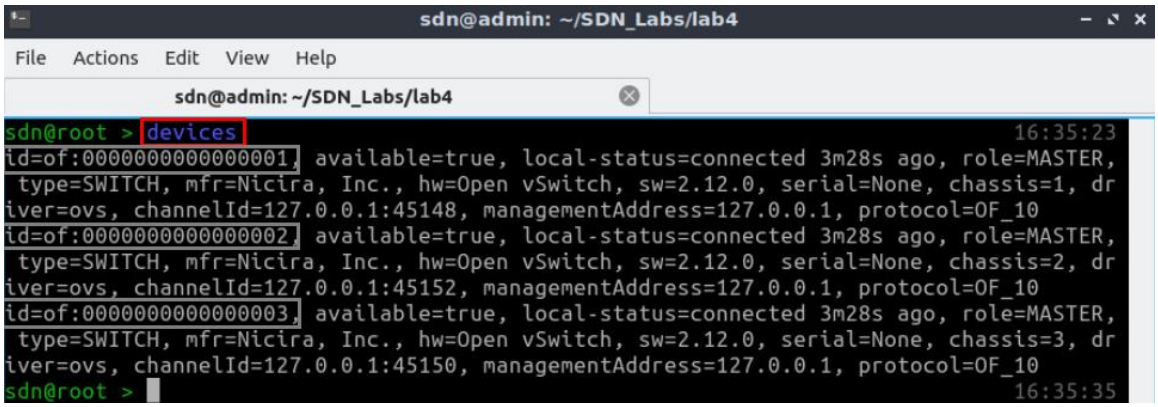

Figure 23. Displaying the current known devices (switches).

Consider Figure 23. The three switches are displayed with their corresponding ids, as well as additional attributes, such as its status (`available=true`), the type of the switch (`hw=Open vSwitch`), and which OpenFlow version the switch is running (`protocol=OF_10`).

Step 6. ONOS commands might have multiple options, to display the list of options for the flows command, write the command `flows` and click the `TAB` button.

```
flows
```

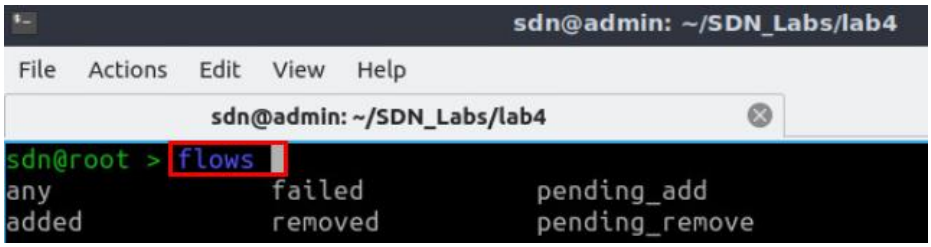

Figure 24. Displaying the current known flows.

Consider Figure 24. The `flows` command has the multiple options according to the state of the flow, which could be:

- `added`: the flow has been added to the switch.
- `failed`: the flow failed to be added.
- `pending_add`: the flow has been submitted and forwarded to the switch.
- `pending_remove`: the request to remove the flow has been submitted and forwarded to the switch.
- `removed`: the rule has been removed.
- `any`: all flows.

Step 7. To display the list of all added switches, complete the above command by writing `added`, then click the `TAB` button.

```
flows added
```

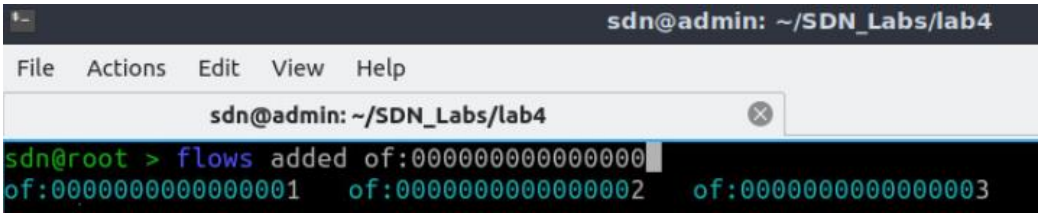

Figure 25. Displaying the current known added flows for switch 1.

Consider Figure 25. Once you click the `TAB` button, ONOS CLI automatically fills the common prefix of all the switches' ids (000000000000000). You can also alternate between the displayed switches' ids using the `TAB` button.

Step 8. To display the list of all added flows for switch 1 (id: 00000000000001), complete the id of the switch (you only have to enter the number `1` for switch s1) and hit enter.

```
flows added of:00000000000001
```

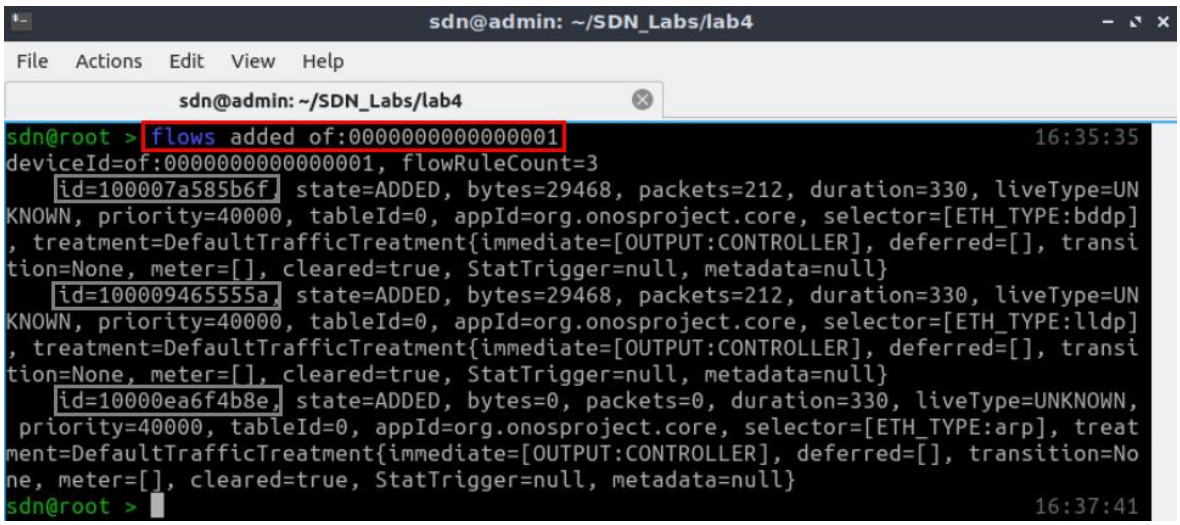

Figure 26. Displaying the current known added flows for switch 1.

Consider Figure 26. Three flows for switch s1 were added. ONOS provides many details about the flows inserted in the switches. For example, each flow entry defines a `selector` and `treatment` which are the set of traffic matched by the flow entry and how this traffic should be handled, respectively.

The controller has installed three initial flows which are:

- Flow 1 (`ETH_TYPE=lldp`): forwards Link Layer Discovery Protocol (LLDP) packets to the controller (`[OUTPUT:CONTROLLER]`).
- Flow 2 (`ETH_TYPE=bddp`): forwards Broadcast Domain Discovery Protocol (BDDP) to the controller (`[OUTPUT:CONTROLLER]`).
- Flow 3 (`ETH_TYPE=arp`): forwards Address Resolution Protocol (ARP) packets to the controller (`[OUTPUT:CONTROLLER]`).

The above flows are used for link and host discovery. Notice as well that each flow entry is tagged by an `appId` (application id), this `appId` identifies which application installed the corresponding flow entry. Other important details include:

- `packets`: number of packets forwarded or dropped for that flow.
- `bytes`: byte count of the matched packets.
- `tableId`: id of the table in which these flows are installed.
- `duration`: time elapsed in seconds since the flow was installed.

Step 9. To display the list of all currently known hosts, type the following command.

```
hosts
```

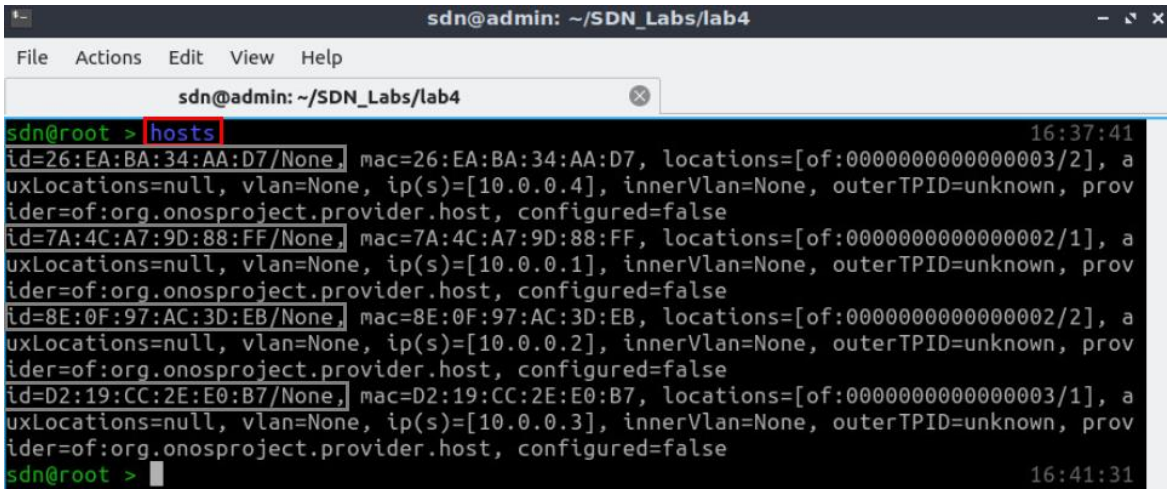

Figure 27. Displaying the current known hosts.

Consider Figure 27. Each discovered host is displayed along some details, such as the id of the host, the location where the host is connected to (with the id of the switch), and the IP address of it. Notice that the id of the hosts, as well as their mac address might be different than those depicted in Figure 27, since Mininet generates new values at each execution.

3.2 Activating ONOS forwarding application

In the previous section, when performing a connectivity test, 100% of the packets were dropped. The pings sent from a host to a destination are IP packets received by a switch, which does not have flows that match IP packets. In this section, you will activate a simple forwarding application that comes with ONOS. This application installs flows in response to every *miss* IP packet that arrives at the controller.

Step 1. To enable the forwarding application, type the command shown below.

```
app activate org.onosproject.fwd
```

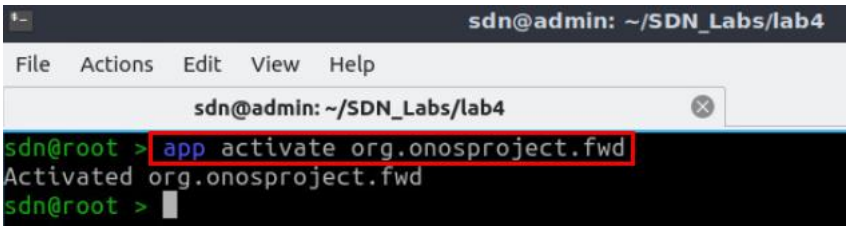

Figure 28. Activating the forwarding application.

Step 2. To display the flows of switch s1, type the following command.

```
flows added of:0000000000000001
```

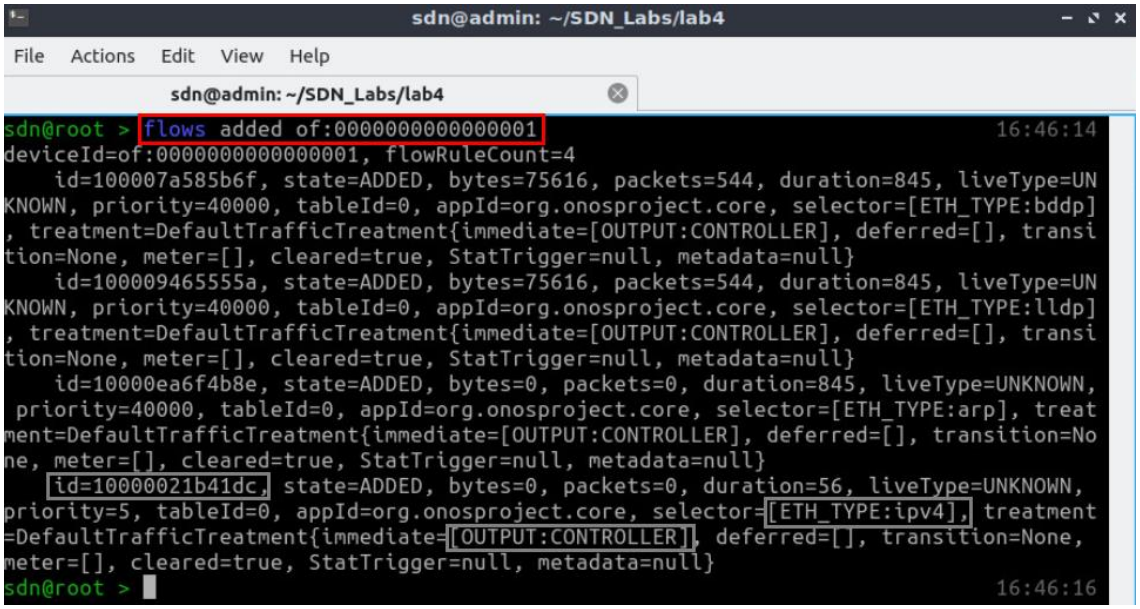

Figure 29. Displaying the current known devices (switches).

Consider Figure 29. A new flow is added with `ETH_TYPE:ipv4` that deals with IPv4 packet by forwarding them the controller (`OUTPUT:CONTROLLER`), which in turns decides the action on the corresponding packet.

Step 3. On host h1 terminal, run a connectivity test by issuing the command shown below. To stop the test, press `Ctrl+c`.

```
ping 10.0.0.2
```

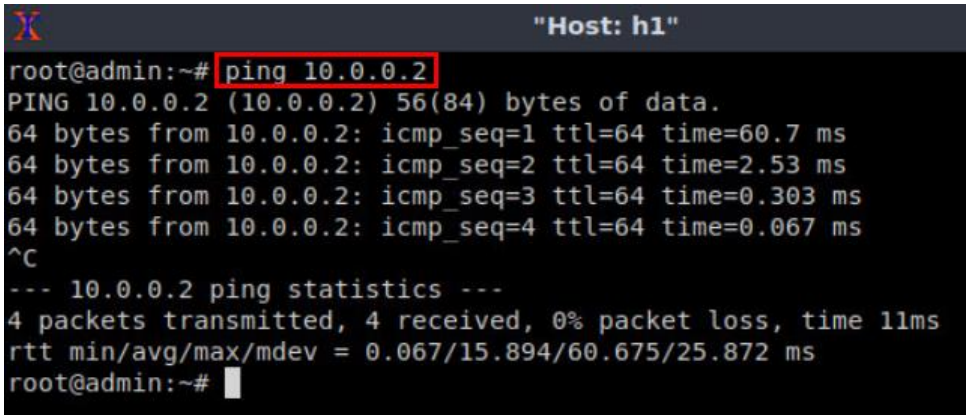

Figure 30. Pinging host h2 from host h1.

Step 4. To display the flows of switch s2, type the following command on ONOS terminal.

```
flows added of:0000000000000002
```

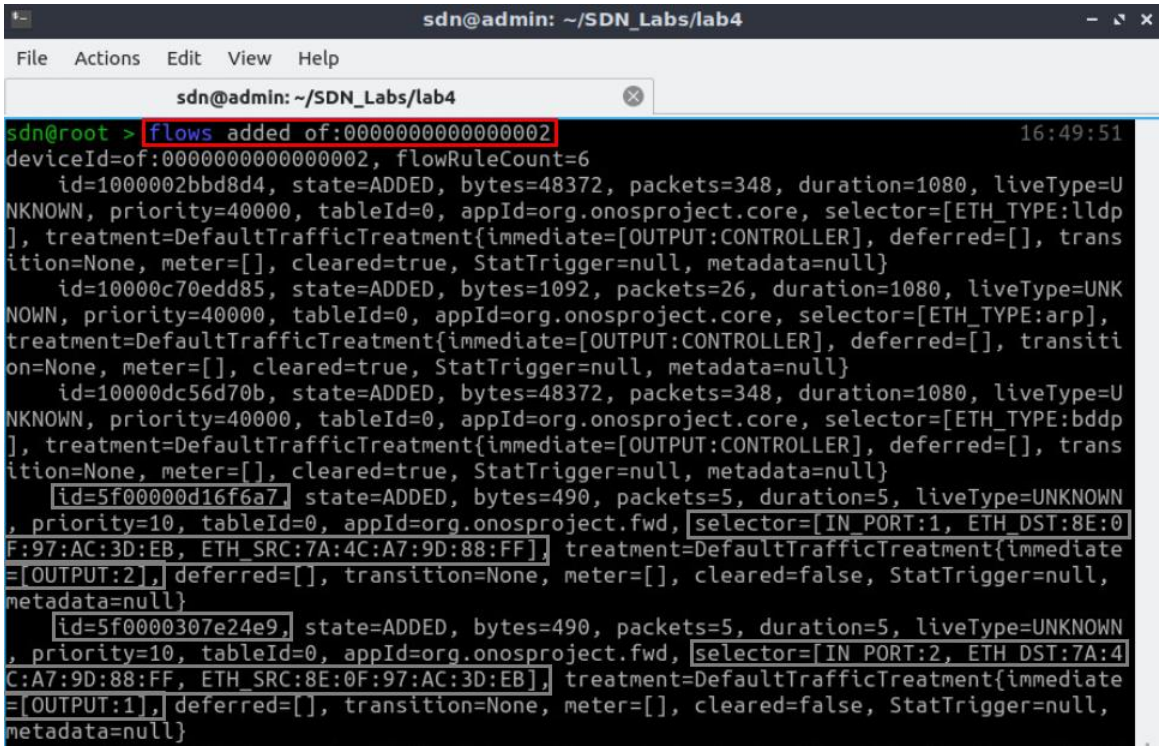

Figure 31. Displaying the added flows on switch s2.

Consider Figure 31. After pinging host h2 from h1, two flows are installed on switch s2. The first flow (`id=5f0000d16f6a7`) instructs the switch to forward incoming packets at port 1 (`IN_PORT:1`) out of port 2 (`OUTPUT:2`). Similarly, the second flow instructs the switch to forward the packets from port 2 to port 1. The inserted flows allow switch s1 to exchange packets between hosts h1 and h2 without relaying them to the controller.

Note that these two flows expire after a certain duration. This is because the forwarding application sets an expiry time for the inserted flows to avoid overflowing the flow table of the switches.

4 Navigating ONOS GUI

The ONOS GUI is a *single-page web-application*, providing a visual interface to the ONOS controller. In this section, you will navigate through ONOS GUI and discover a number of its features.

4.1 Accessing the web user interface

Step 1. Open the web browser by clicking on the shortcut located in the lower left-hand side.

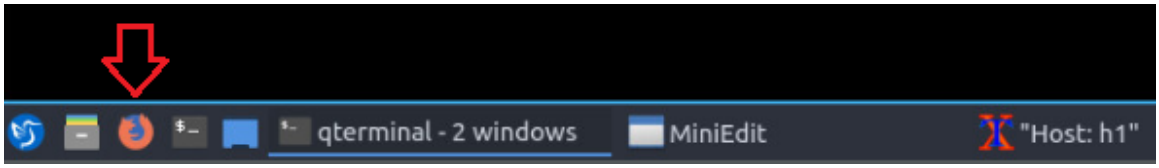

Figure 32. Opening the web browser

Step 2. Navigate to the following URL to access ONOS web user interface:

```
localhost:8181/onos/ui
```

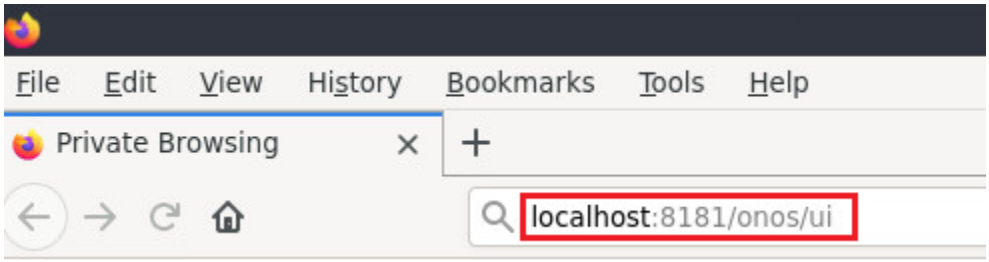

Figure 33. Opening the ONOS web user interface.

Step 3. Provide the following credentials to access the web user interface:

- User: `onos`
- Password: `rocks`

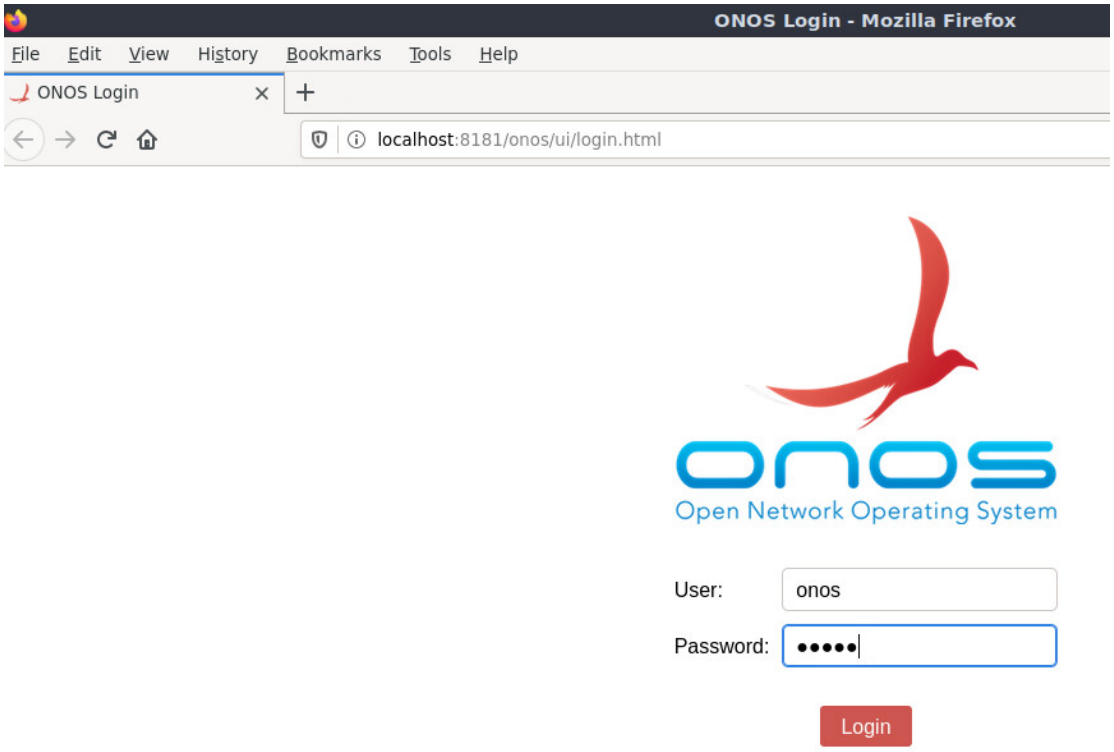

Figure 34. ONOS authentication window.

A topology consisting of three switches will be displayed. Such topology corresponds to the one created on Mininet.

Figure 35. Displaying the topology in ONOS GUI.

Notice that the network components will show up in a random arrangement. The user can click on the components and drag them to their preferred location.

4.2 Exploring network components

ONOS web user interface provides the tools to monitor the specification of each device that comprises the network.

Step 1. To open ONOS menu, click on the upper left-hand side icon. A drop-down menu will be displayed. To check the devices, click on devices. A new window will emerge.

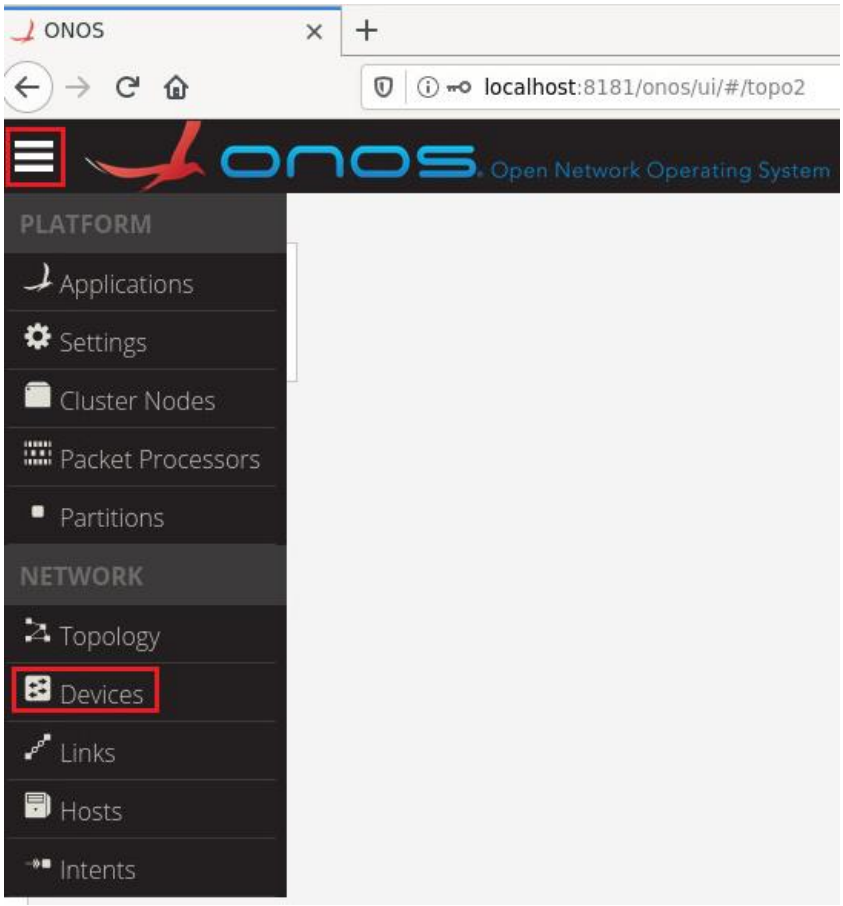

Figure 36. Opening devices.

The emerging window will display three devices. Those three devices correspond to the three switches that compounds the topology. The information provided in the GUI include:

- FRIENDLY NAME: if not specified, it is the name of the device.
- DEVICE ID: name of the switch.
- MASTER: IP address where ONOS is running. ONOS is running locally (127.0.0.1).
- PORTS: number of ports of the switch.
- PROTOCOL: OpenFlow version running on the switch.

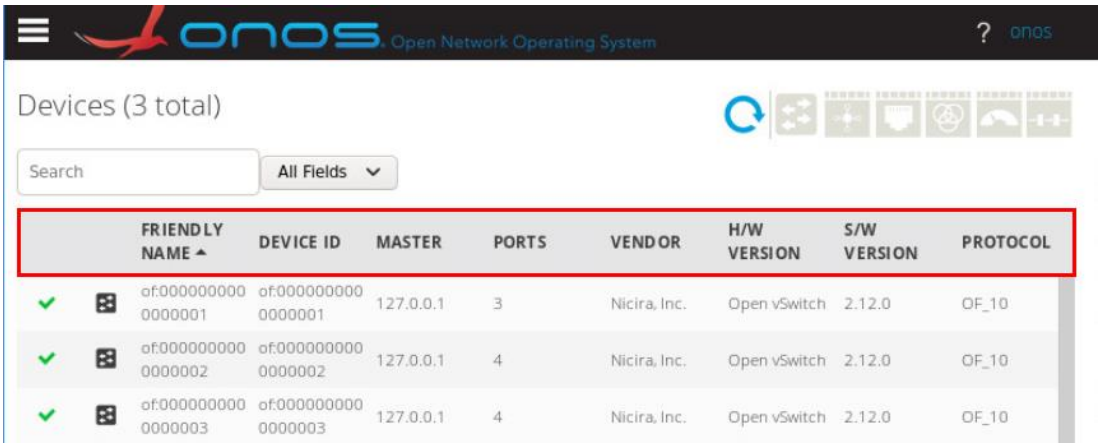

Figure 37. Devices' information.

Step 2. From the menu list, click on hosts to verify the hosts' information.

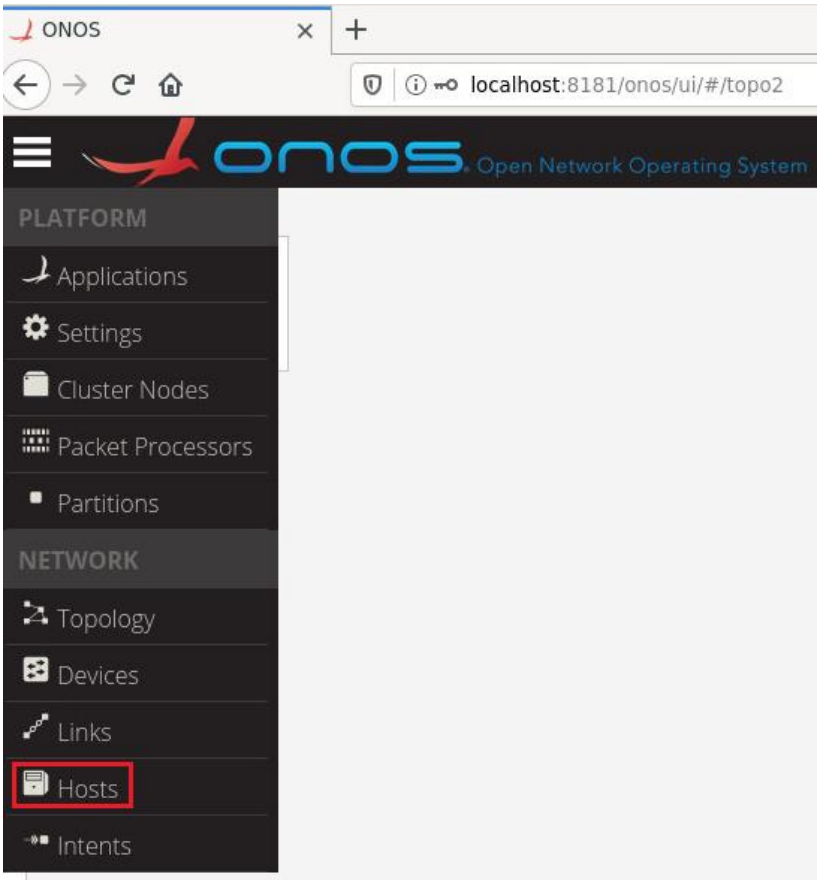

Figure 38. Opening hosts.

Similarly, the new window presents the information regarding the hosts that compound the topology. The information provided in the GUI include:

- Host ID/MAC ADDRESS: mac address of the host.
- IP ADDRESSES: IP address of the host.
- Location: the port of the switch connected to the host.

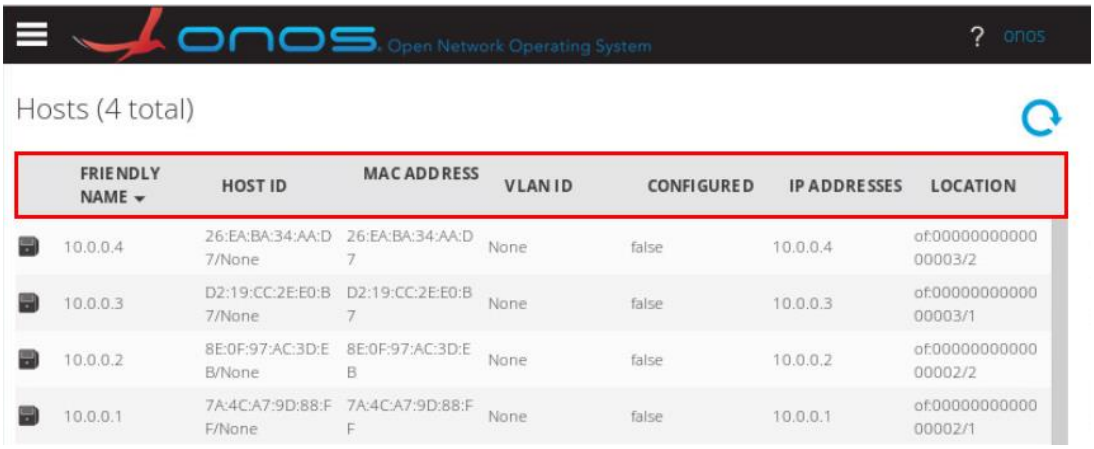

Figure 39. Hosts' information.

Step 3. Go back to the main window (topology).

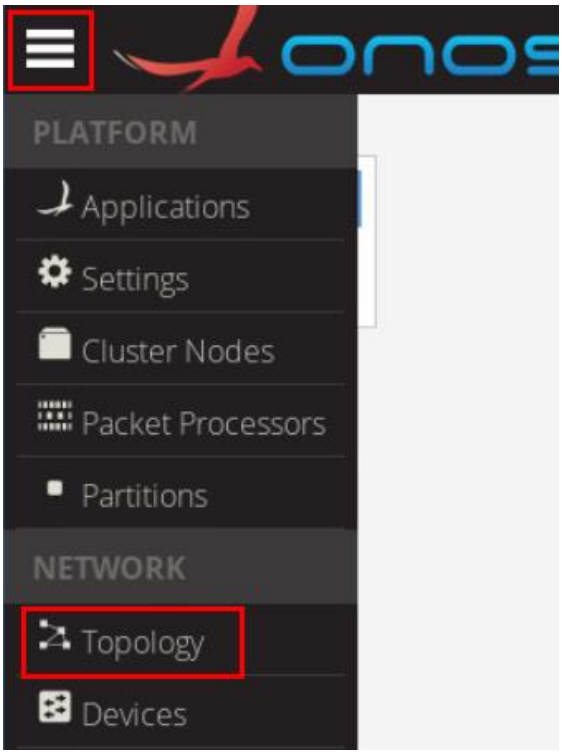

Figure 40. Opening the topology.

Step 4. Click on the bar located in the lower left-hand side. A toolbox will show up. Select the host icon to see the hosts connected to the topology.

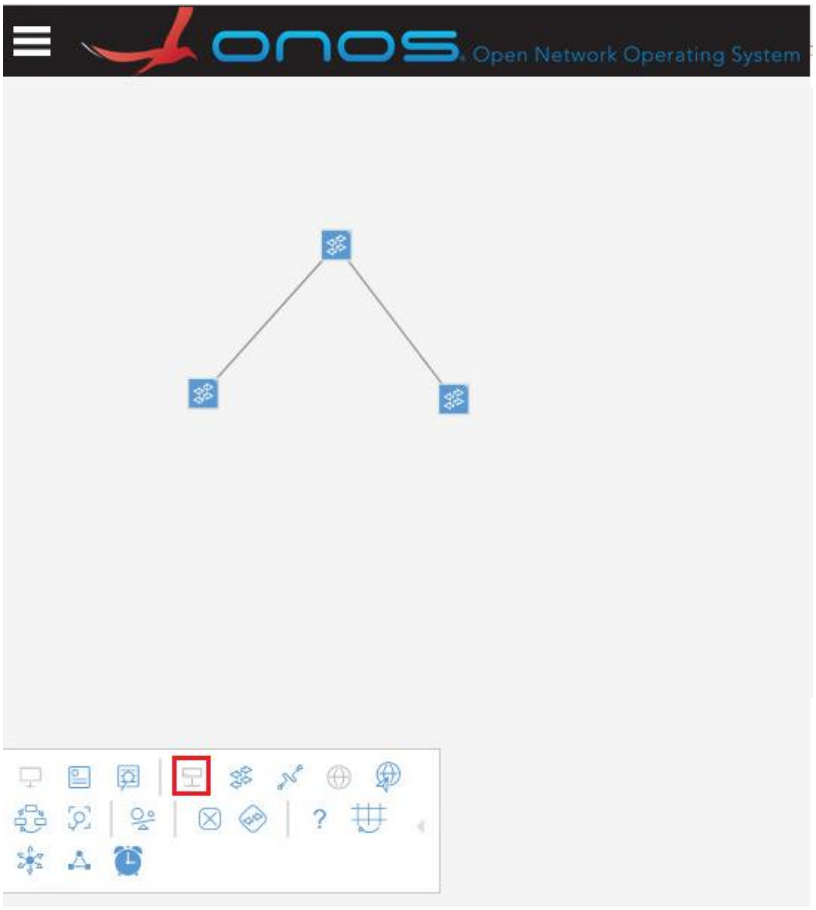

Figure 41. Enabling hosts visualization.

Step 5. Go back to the dashboard to verify the network components.

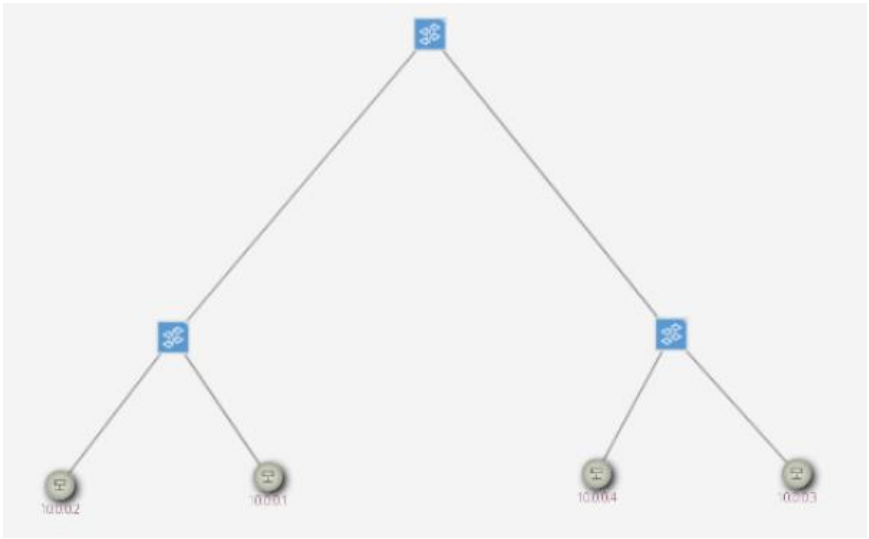

Figure 42. Network components.

Consider Figure 42. The information attached to the hosts include their IP addresses.

Step 6. To view a summary of a specific device, you can click on that device as shown in the figure below.

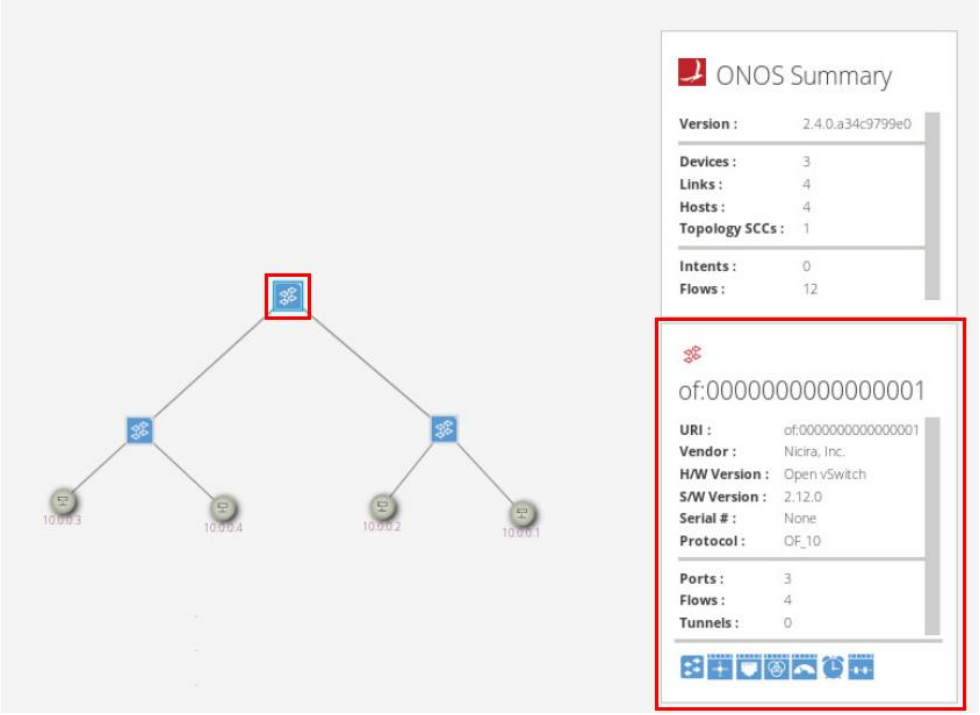

Figure 43. Network components.

Consider Figure 43. Upon clicking on switch s1, a panel opens on the right-hand side of the topology. The panel displays a summary of the device. For example, if it is a switch, it would display information such the name, number of ports, and flows in that switch.

Step 7. From the menu list, click on Applications to load the applications available in ONOS.

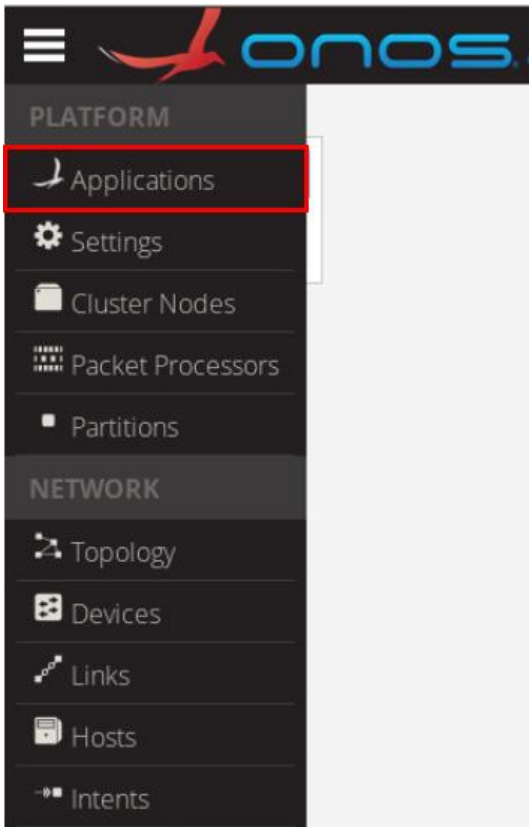

Figure 44. Opening applications.

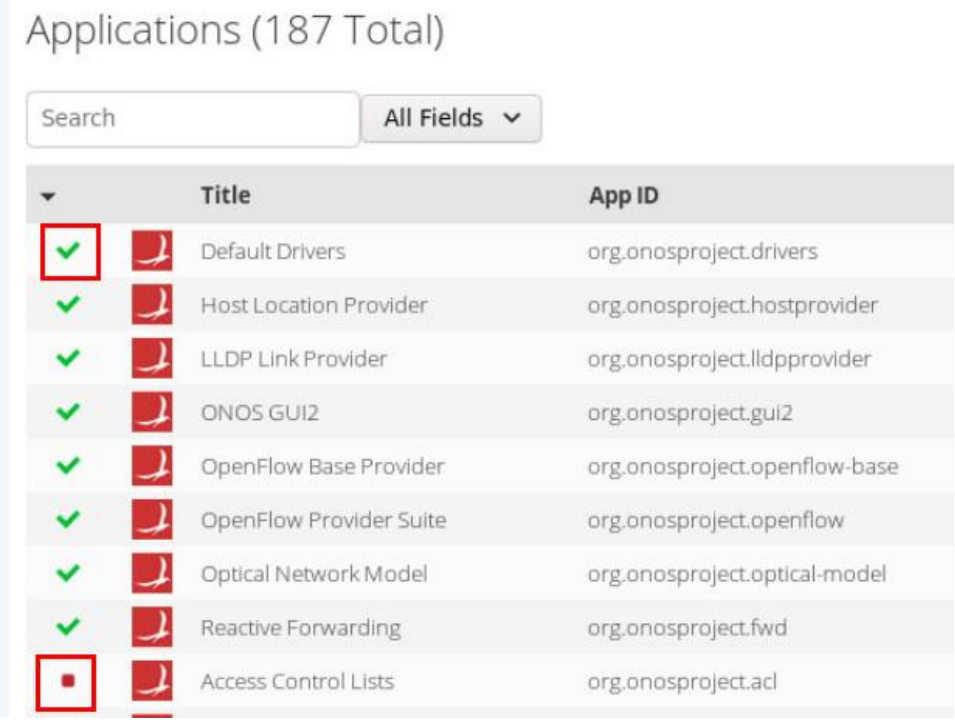

Figure 45. Opening applications.

Consider Figure 45. A list of all the available applications is displayed. The activated applications are checked in green (e.g., Default Drivers application), whereas the deactivated applications are marked with a red square (e.g., Access Control Lists application). Note that the activated applications are a result of activating the OpenFlow and the forwarding application.

Step 8. To deactivate an application, you can click on the application, hit the deactivation button (gray square on the right-hand side of the window), then confirm the operation. The steps are shown in the below figure, where we deactivate the forwarding application (reactive forwarding).

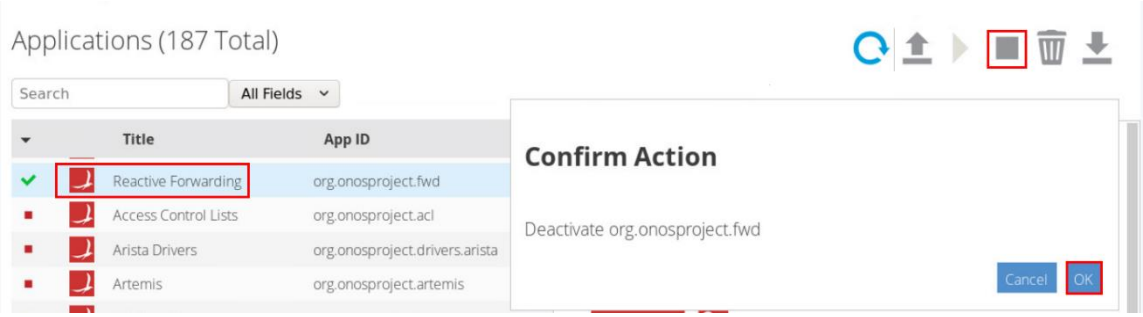

Figure 46. Opening applications.

Step 9. To activate an application, you can click on the application, hit the activation button (gray arrow on the right-hand side of the window), then confirm the operation. The steps are shown in the below figure, where we activate the forwarding application (reactive forwarding).

Figure 47. Opening applications.

Displaying flow tables using GUI

This concludes Lab 4. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. "*ONOS: towards an open, distributed SDN OS*," In Proceedings of the third workshop on Hot topics in software defined networking, pp. 1-6, 2014.
2. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "*OpenFlow: enabling innovation in campus networks*." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.
3. D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig, *"Software-defined networking: A comprehensive survey*," Proceedings of the IEEE, 103(1), pp.14-76, 2014.
4. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach"*. Morgan Kaufmann, 2016.
5. Mininet walkthrough, [Online]. Available: http://mininet.org.

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 5: Configuring VXLAN to Provide Network Traffic Isolation

Document Version: 05-25-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab presents Virtual eXtensible Local Area Network (VXLAN), a network virtualization scheme that provides a solution for the scalability problems associated with datacenter and large cloud computing deployments. This lab aims to configure VXLAN to isolate network traffic within an emulated environment. Additionally, the user will inspect the packets to identify the fields corresponding to VXLAN network identifiers.

Objectives

By the end of this lab, the user will:

1. Understand the concept of VXLAN.
2. Assign IP addresses to a router interface.
3. Configure a routing protocol.
4. Emulate servers by using docker containers.
5. Push flow tables to configure VXLAN in a switch.
6. Isolate network traffic by using VXLAN.
7. Visualize VXLAN network identifiers by using Wireshark.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Configuring OSPF router r1 and router r2.
4. Section 4: Configuring VXLAN.
5. Section 5: Verifying configuration.

1 Introduction

Data centers operate by hosting services for multiple tenants such as data servers and cloud computing services[7]. Those services require on-demand elastic provisioning of computing resources for multi-tenant environments. Such feature is supported by network virtualization, which provide an efficient way to host multiple tenants in the same server and traffic isolation, to avoid a tenant to have access to the data of another tenant.

Isolating network traffic could be done via Layer 2 or Layer 3 networks. For Layer 2 networks, VLANs are often used to segregate traffic. In such scenario a tenant could be identified by its own VLAN. However, VLAN-based network isolation suffers a limitation of 4094 VLANs, which is inadequate considering the high demand of cloud services. Additionally, a tenant could require multiple VLANs, which exacerbates the issue.

On the other hand, layer 3 networks do not provide an extensive solution for multi-tenant networks as well. Two tenants might use the same set of Layer 3 addresses within their networks, which requires the cloud provider to provide isolation in some other form. Further, requiring all tenants to use IP excludes customers relying on direct Layer 2 or non-IP Layer 3 protocols for inter VM communication.

1.1 VXLAN architecture

Hypervisor-based overlay networks is a novel use of Software-defined Network (SDN) capabilities[3]. This concept does not modify the physical network, which means that networking devices and their configurations remains unchanged. Hypervisor-based virtualized networks are built above such network[5]. The system at the edge of the network works as an interface to these virtual networks. In these networks, many details of the physical network from the devices that connect to the overlays are hidden.

VXLAN (Virtual eXtensible Local Area Network) addresses the above requirements of Layer 2 and Layer 3 data center network infrastructure in the presence of Virtual Machines (VMs) in a multi-tenant environment. VXLAN runs over the existing networking infrastructure and provides a means to increase the number of devices on a Layer 2 network. In summary, VXLAN is a Layer 2 overlay scheme build on the top of a Layer 3 network. Each overlay is unique within the tenant domain and is known as VXLAN segment. The communication is restricted just between VMs within the same VXLAN segment. Each VXLAN segment is identified by a 24-bit segment ID, called the VXLAN Network Identifier (VNI). This allows up to 16 million (2^{24}) VXLAN segments to coexist within the same administrative domain.

Consider Figure 1. VXLAN could be considered as a tunneling scheme to overlay Layer 2 networks on top of Layer 3 networks[6]. The tunnels are stateless, so each segment is encapsulated according to a set of rules. The end point of the tunnel (VXLAN Tunnel End Point or VTEP) is located within the hypervisor on the server that hosts the VM. The traffic is isolated according to the VNI and the end-hosts can communicate as they are located within the same network.

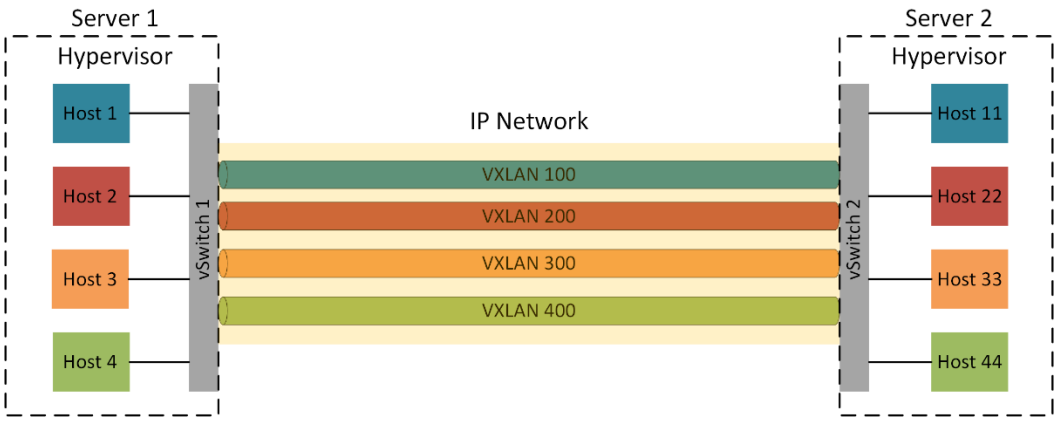

Figure 1. VXLAN overview.

1.2 VXLAN packet format

Figure 2 illustrates the format of a VXLAN packet[5]. The outer header contains the MAC and IP addresses appropriate for sending a unicast packet to the destination switch, acting as a virtual tunnel end point. The VXLAN header follows the outer header and contains a VXLAN Network Identifier of 24 bits in length, sufficient for about 16 million networks.

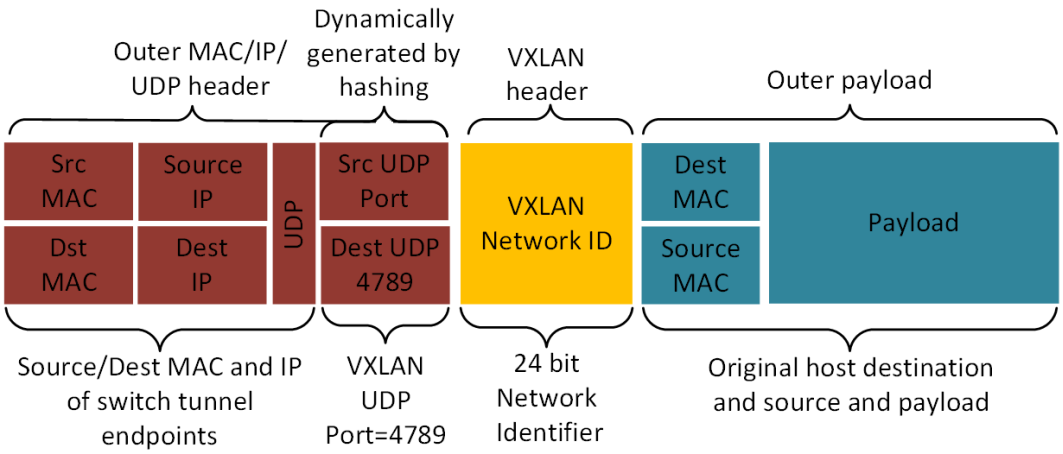

Figure 2. VXLAN packet format.

2 Lab topology

Consider Figure 3. The topology consists of four end-hosts, two switches and two routers. The end hosts and switches are running inside Server 1 and Server 2. Those servers are implemented by Docker[8] containers which run Mininet instances. Router r1 and router r2 are supported by Free-range Routing (FRR) engine.

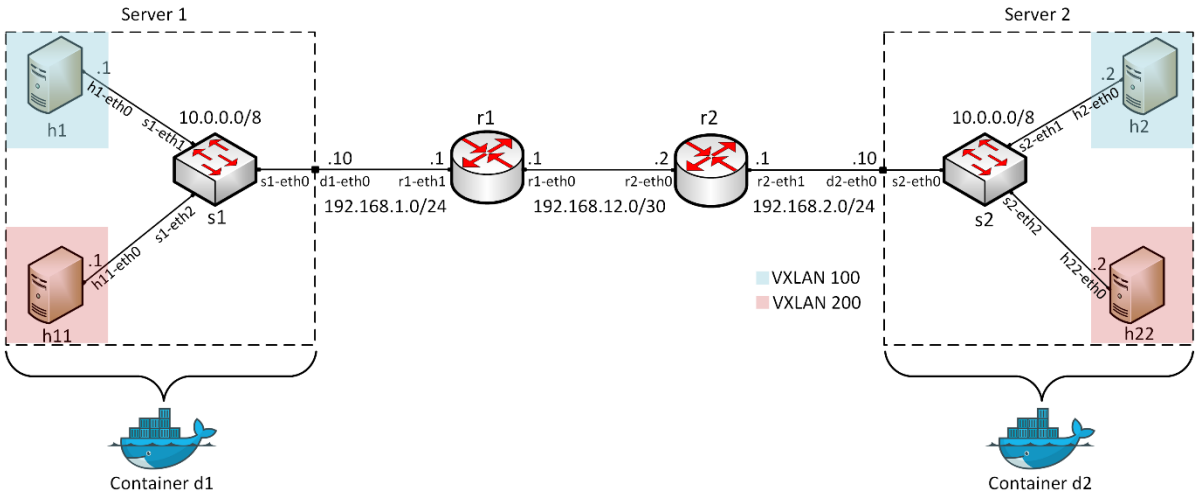

Figure 3. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet
r1	r1-eth0	192.168.12.1	/30
	r1-eth1	192.168.1.1	/24
r2	r2-eth0	192.168.12.2	/30
	r2-eth1	192.168.2.1	/24
h1	h1-eth0	10.0.0.1	/8
h11	h11-eth0	10.0.0.1	/8
h2	h2-eth0	10.0.0.2	/8
h22	h22-eth0	10.0.0.2	/8
d1	d1-eth0	192.168.1.10	/24
d2	d2-eth0	192.168.2.10	/24

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), save the topology and export a layer 2 model.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

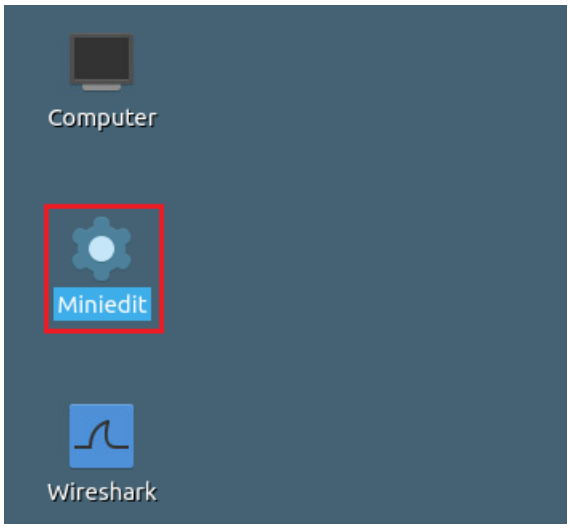

Figure 4. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab5.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab5* and click on *Open*.

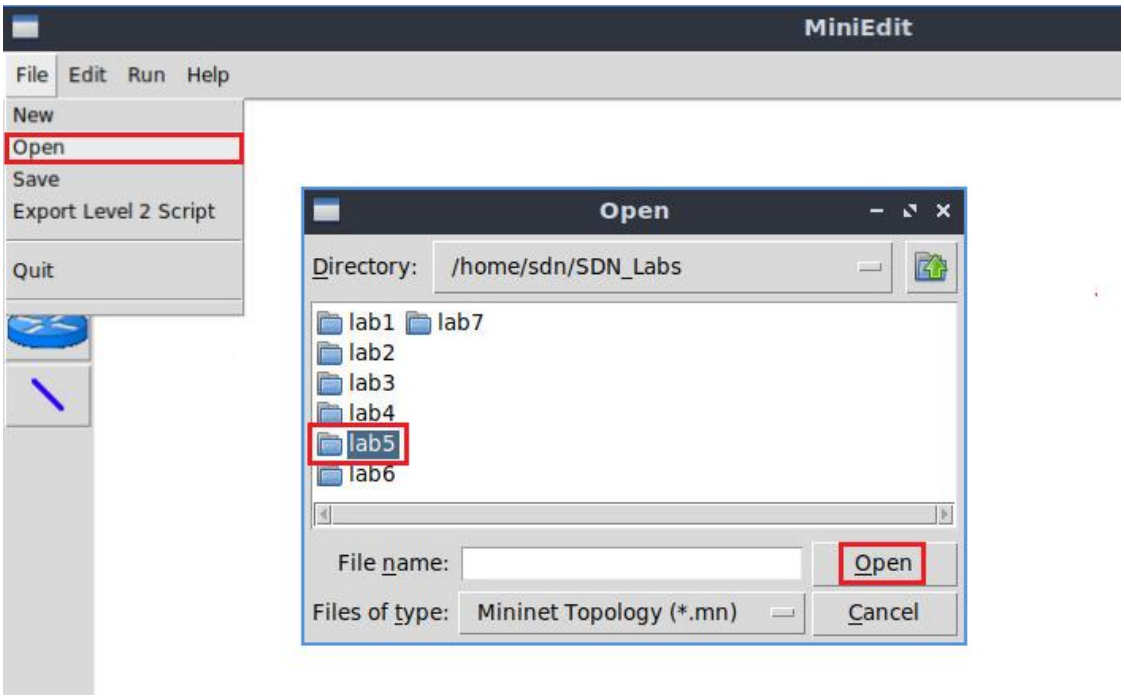

Figure 5. Opening topology.

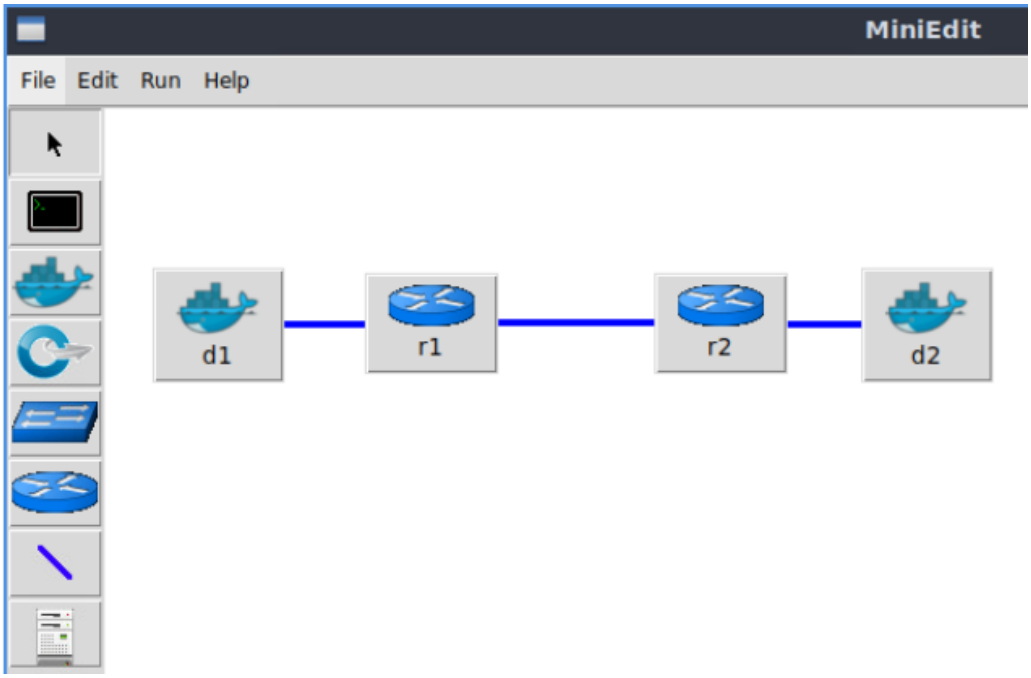

Figure 6. MiniEdit's topology.

2.3 Load the configuration file

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open Linux terminal.

Figure 7. Opening Linux terminal.

Step 2. Click on the Linux terminal and navigate into *SDN_Labs/lab5* directory by issuing the following command. This folder contains a configuration file and the script responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab5
```

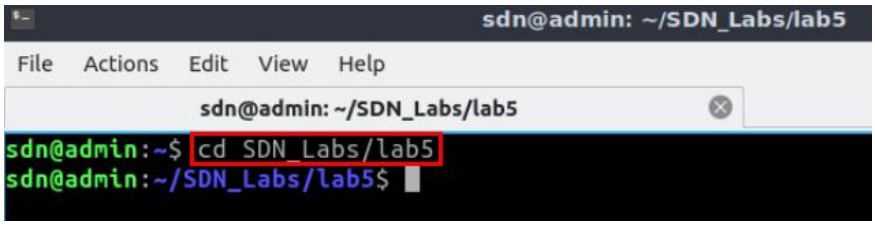

Figure 8. Entering the *SDN_Labs/lab5* directory.

Step 3. To execute the shell script, type the following command. The argument of the program corresponds to the configuration zip file that will be loaded in all the routers in the topology.

```
./config_loader.sh lab5_conf.zip
```

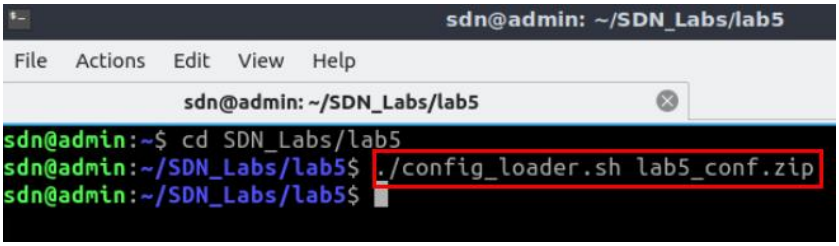

Figure 9. Executing the shell script to load the configuration.

Step 4. Type the following command to exit the Linux terminal.

```
exit
```

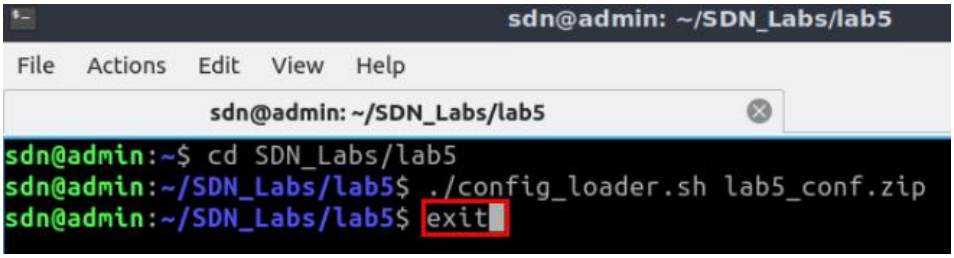

Figure 10. Exiting from the terminal.

2.4 Run the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

Figure 11. Starting the emulation.

Step 2. Issue the following command on Mininet terminal to display the interface names and connections.

```
links
```

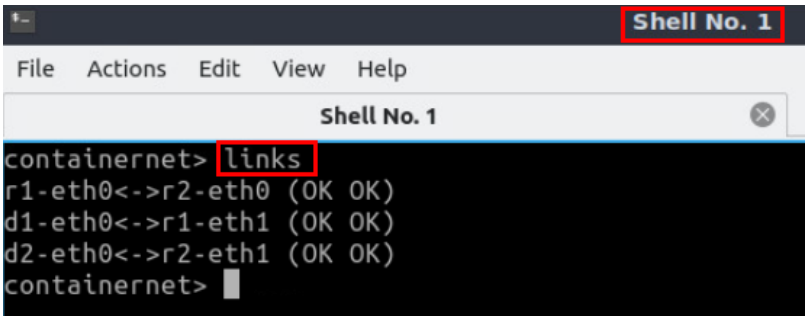

Figure 12. Displaying network interfaces.

In Figure 12, the link displayed within the gray box indicates that interface *eth0* of router r1 connects to interface *eth0* of router r2 (i.e., *r1-eth0<->r2-eth0*).

2.5 Verify the configuration

You will verify the IP addresses listed in Table 2 and inspect the routing table of routers r1, r2, and r3.

Step 1. In order to verify router r1, hold right-click on router r1 and select *Terminal*.

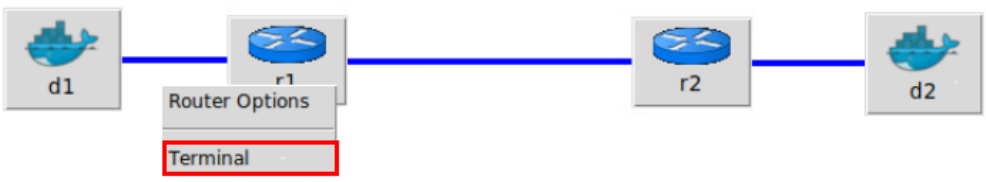

Figure 13. Opening a terminal on router r1.

Step 2. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you

do not enable zebra daemon initially. In order to start the zebra, type the following command:

```
zebra
```

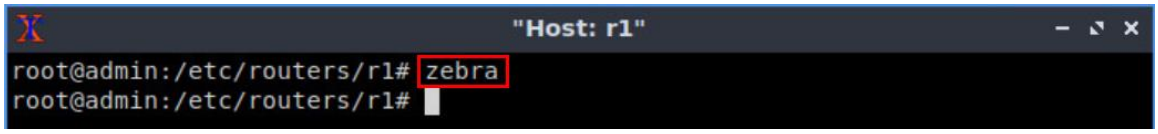

Figure 14. Starting zebra daemon.

Step 3. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command:

```
vtysh
```

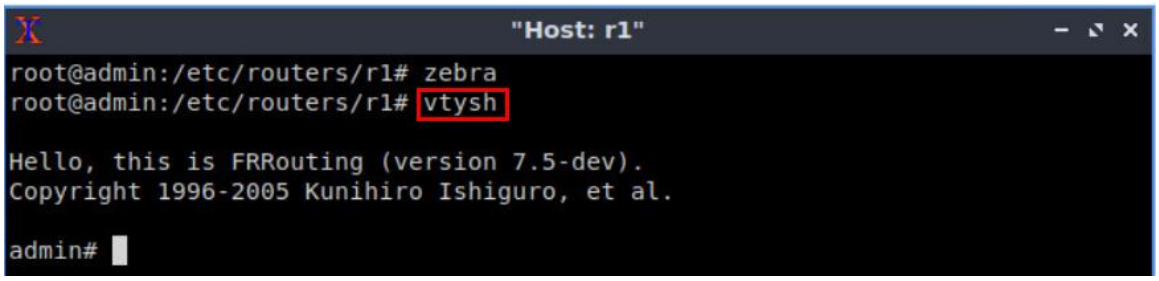

Figure 15. Starting vtysh on router r1.

Step 4. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 does not contain any route to external networks as there is no routing protocol configured yet.

```
show ip route
```

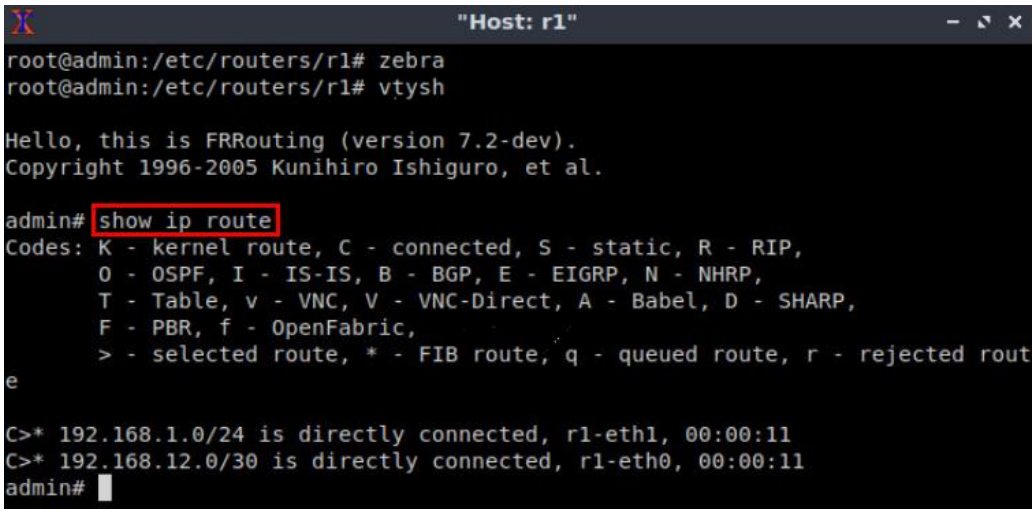

Figure 16. Displaying routing table of router r1.

The output in the figure above shows that the networks 192.168.1.0/24 and 192.168.12.0/30 are directly connected through the interfaces *r1-eth1* and *r1-eth0*, respectively.

Step 5. Hold right-click on router r2 and select *Terminal*.

Figure 17. Opening a terminal on router r2.

Step 6. Router r2 is configured similarly to router r1 but, with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r2 terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r2.

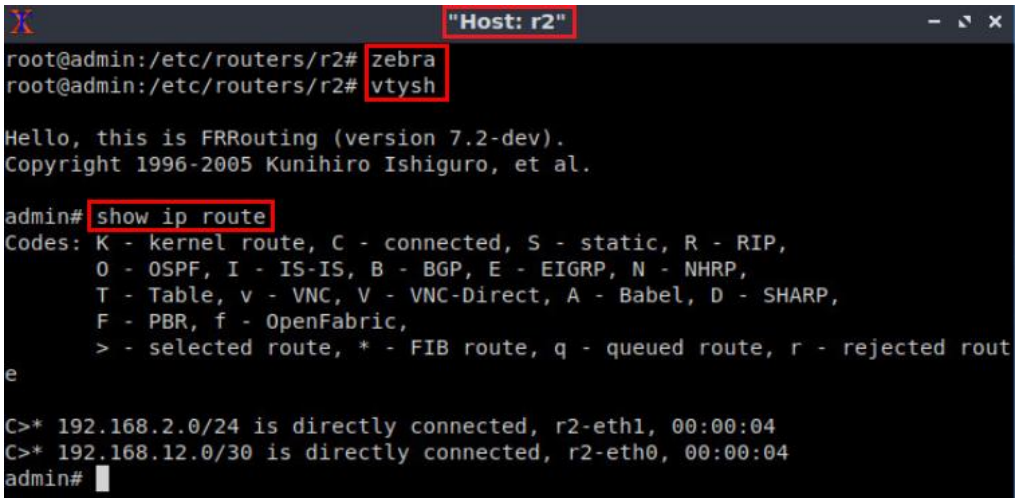

Figure 18. Displaying routing table of router r2.

3 Configuring OSPF on routers r1 and r2

In this section, you will configure OSPF routing protocol in router r1 and router r2. First, you will enable the OSPF daemon on routers r3 and r4. Second, you will a establish single area OSPF, which is classified as area 0 or backbone area. Finally, you will advertise all the connected networks.

Step 1. To configure OSPF routing protocol, you need to enable the OSPF daemon first. In router r1, type the following command to exit the vtysh session.

```
exit
```

Figure 19. Exiting the vtysh session.

Step 2. Type the following command on router r1 terminal to enable OSPF daemon.

```
ospfd
```

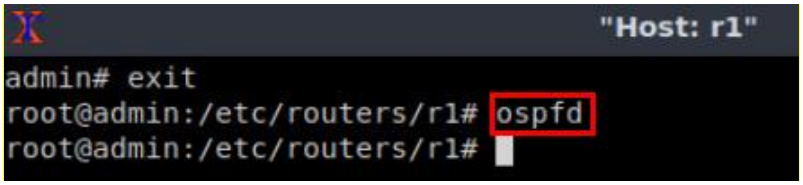

Figure 20. Starting OSPF daemon.

Step 3. In order to enter to router r1 terminal, issue the following command.

```
vtysh
```

Figure 21. Star Starting vtyshon router r1.

Step 4. To enable router r1 configuration mode, issue the following command:

```
configure terminal
```

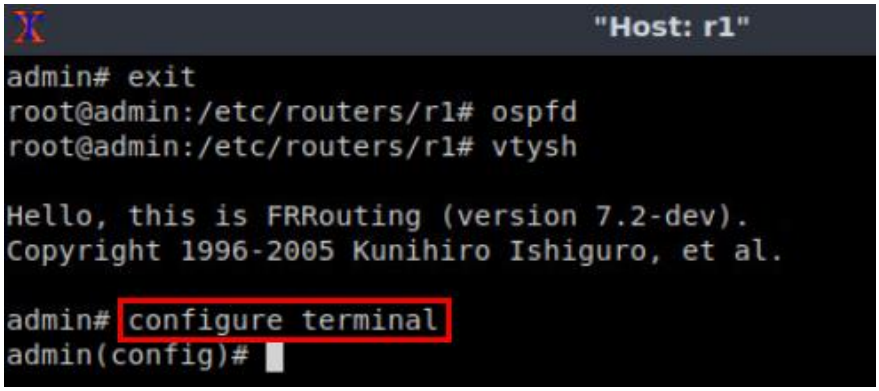

Figure 22. Enabling configuration mode on router r1.

Step 5. In order to configure OSPF routing protocol, type the command shown below. This command enables OSPF configuration mode where you advertise the networks directly connected to router r1.

```
router ospf
```

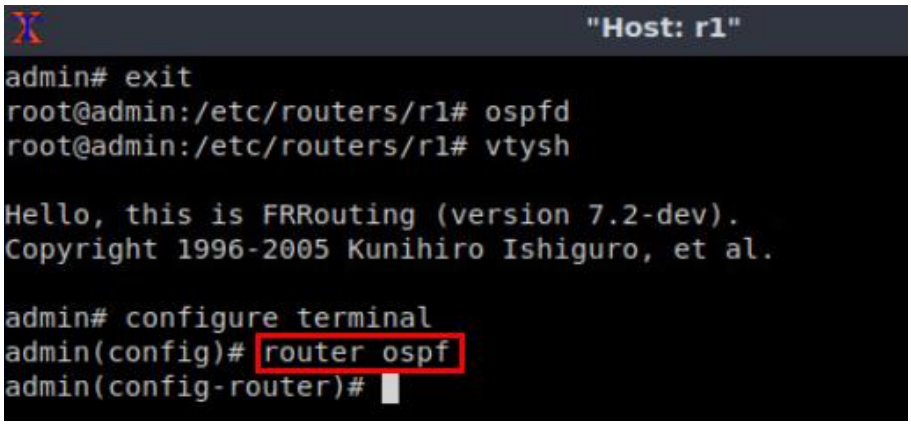

Figure 23. Configuring OSPF on router r1.

Step 6. In this step, you will enable all the interfaces of router r1 to participate in the OSPF routing process, i.e., all the attached networks will be advertised to OSPF neighbors. The advertised networks are associated with area 0. To advertise all connected networks in the same command, the network 0.0.0.0/0 will be used. This network address matches all IP addresses.

```
network 0.0.0.0/0 area 0
```

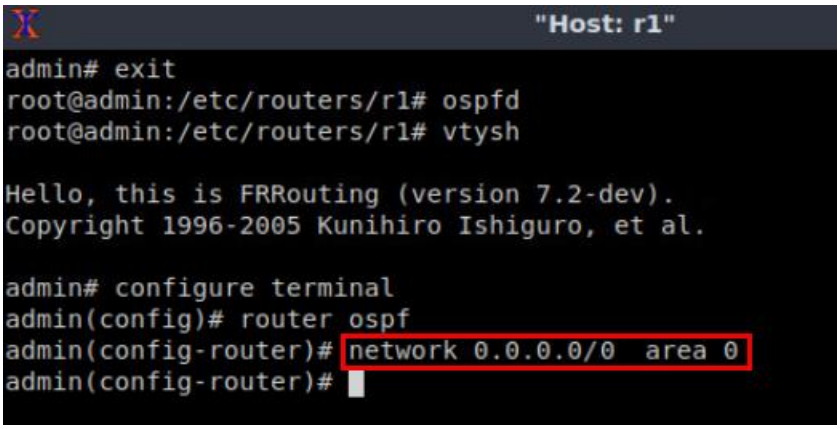

Figure 24. Enabling all the interfaces of router r1 to participate in the OSPF routing process.

Step 7. Type the following command to exit from the configuration mode.

```
end
```

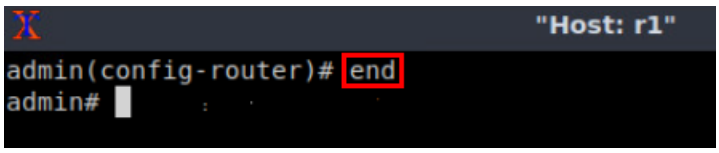

Figure 25. Exiting from the configuration mode.

Step 8. Router r2 is configured similarly to router r1. Those steps are summarized in the following figure. To proceed, on route r2 terminal, issue the commands depicted below.

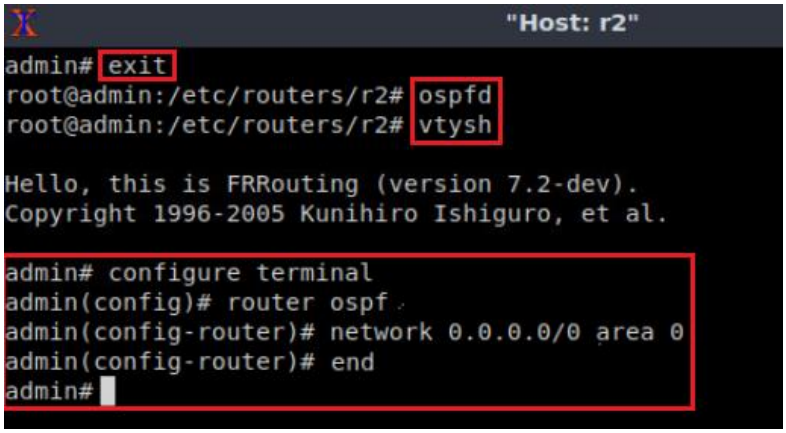

Figure 26. Exiting from the configuration mode.

Step 9. Type the following command to verify the routing table of router r1.

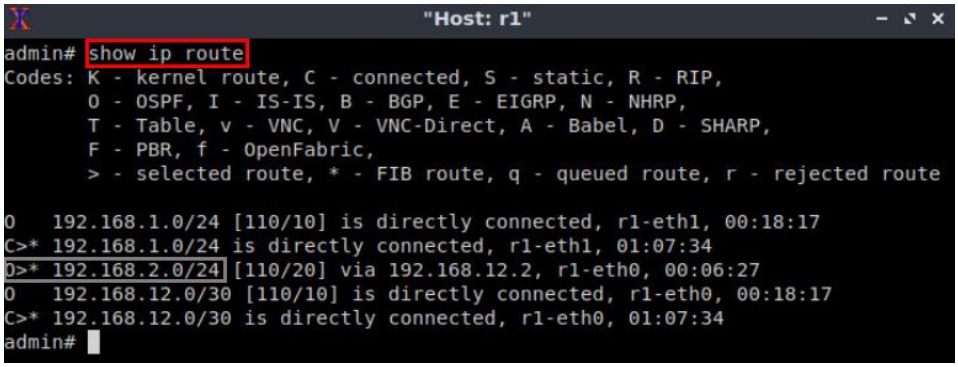

Figure 27. Verifying the routing table of router r1.

Consider Figure 27. The network 192.168.2.0/24 is learned via OSPF (O>*) and it is reachable via the next hop 192.168.12.2 (route r2).

4 Configuring VXLAN

In this section, the user will start the networks within the containers d1 and d2. Both containers run a Mininet topology as depicted in Figure 3. In container d1, the topology consists in two hosts (h1 and h11) connected to a switch (s1). Similarly, container d2 runs a topology with two hosts (h2 and h22) connected to a switch (s2). The end-hosts within the containers will be isolated by using VXLAN.

Note that the containers d1 and d2 emulates a multitenant environment. Multi-tenancy is a mode of operation where multiple independent instances such as end-hosts (see Figure 3) of a tenant operate in a shared environment, while ensuring logical segmentation between the instances. A tenant could be a business entity, user group,

applications, or cloud services. The tenant instances such as h1, h11, h2 and h22 are logically isolated but physically operate on the same fabric.

4.1 Run Mininet instances within the containers

The following section shows the steps to run a Mininet topology within the containers and how to navigate through the configuration files.

Step 1. In container d1 terminal, type the following python script to start a Mininet instance that consists of two hosts connected to a switch.

```
python start_server1.py
```

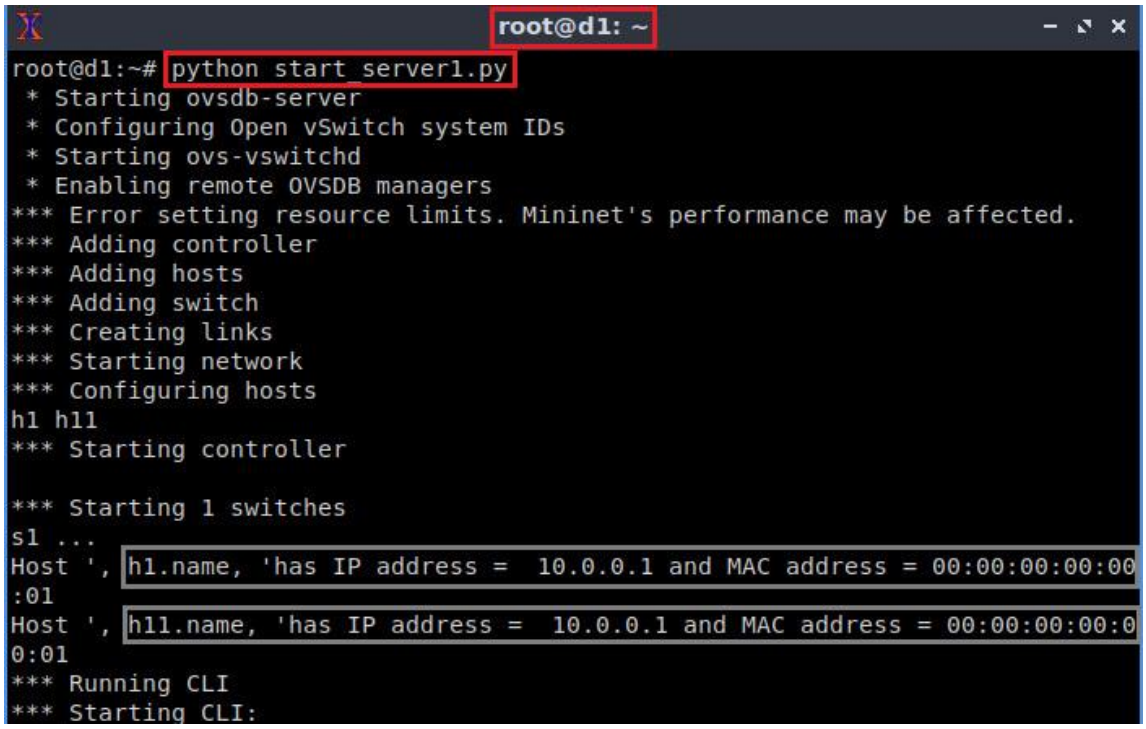

Figure 28. Starting a Mininet instance within container d1.

The figure above starts a Mininet instance in the container d1. Also, the information about the hosts are summarized after starting switch s1.

Notice that host h1 and host h11 have the same IP addresses and MAC addresses. These hosts will be isolated b using VXLAN.

Step 2. In container d1, run the following command to verify the devices in the topology:

```
links
```

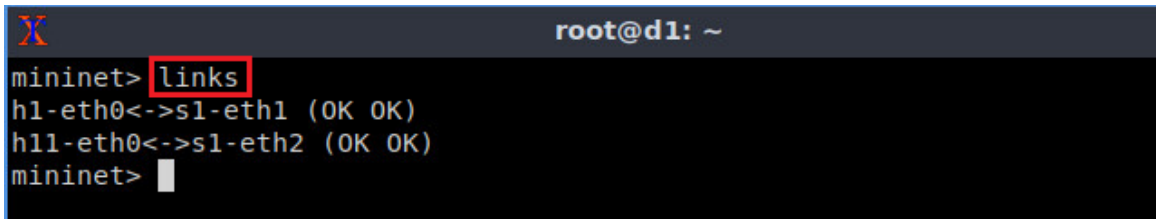

Figure 29. Verifying the links between the devices in container d1.

The figure above shows that the host h1 and switch s1 are connected via the interface pair *h1-eth0<->s1-eth1*. Similarly, host h11 is connected to the switch s1 (*h11-eth0<->s1-eth2*).

Step 3. Similarly, in container d2 terminal, type the following python script to start a Mininet instance that consists in two hosts connected to a switch as well.

```
python start_server2.py
```

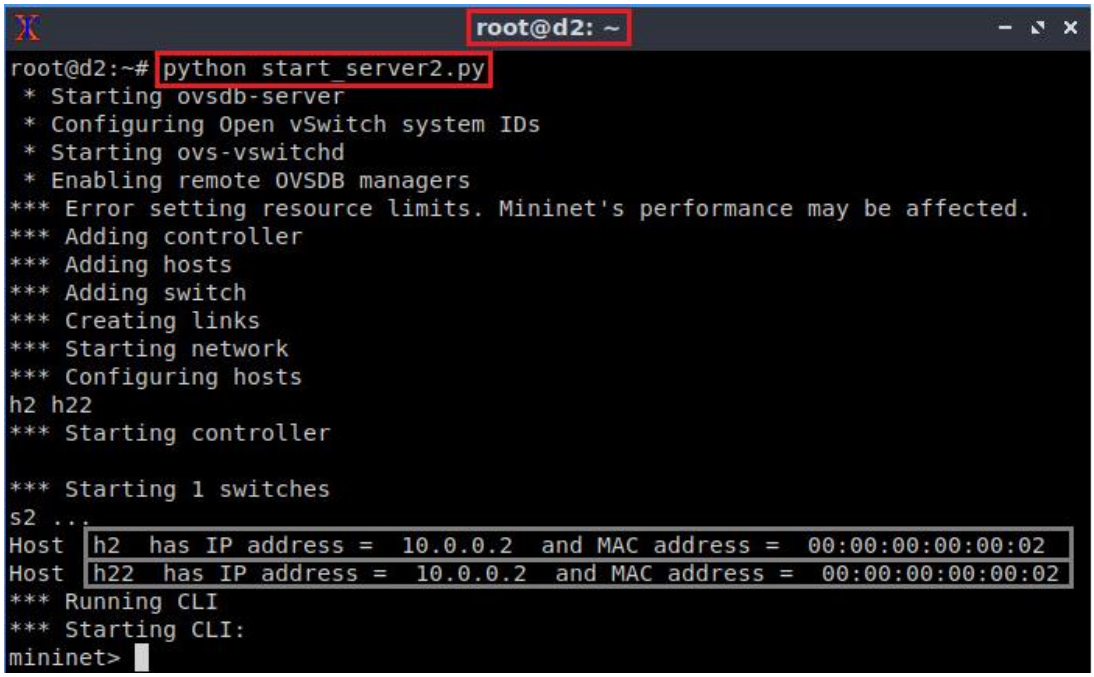

Figure 30. Starting a Mininet instance within container d2.

The figure above starts a Mininet instance in the container d2. Also, the information about the hosts are summarized after starting switch s2.

Notice that host h2 and host h22 have the same IP addresses and MAC addresses. These hosts will be isolated b using VXLAN.

Step 4. In container d2 terminal, run the following command to verify the devices in the topology:

```
links
```

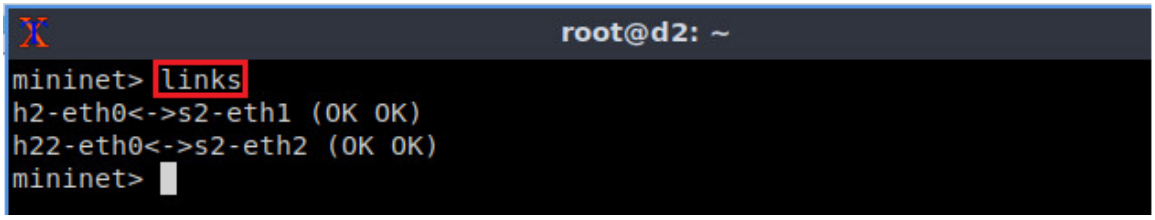

Figure 31. Verifying the links between the devices in container d2.

The figure above shows that the host h2 and switch s2 are connected via the interface pair *h2-eth0<->s2-eth1*. Similarly, host h22 is connected to the switch s2 (*h22-eth0<->s2-eth2*).

4.2 Adding entries to the switches' flow tables

In this section you will add entries to the flow tables of switch s1 and switch s2. These entries are added to a table that is responsible for traffic processing. In this lab, the flow tables specify the VXLAN tags and the actions to forward the packets to their right destination.

The main purpose of configuring VXLAN in this lab is to isolate the traffic from h1 to h2 and from h11 to h22.

Step 1. To visualize the entries to be added to the flow table of switch s1, in container d1, type the following command:

```
sh cat flow1.txt | nl
```

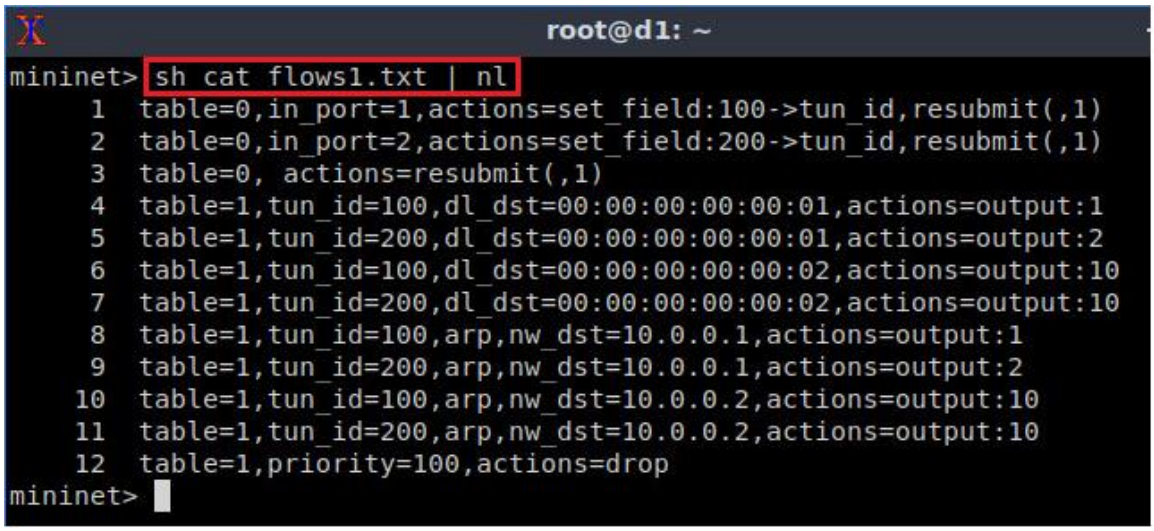

Figure 32. Flow table in container d1.

Step 2. In container d1**,** Issue the following command to add entries to the flow table of switch s1.

```
sh ovs-ofctl add-flows s1 flows1.txt
```

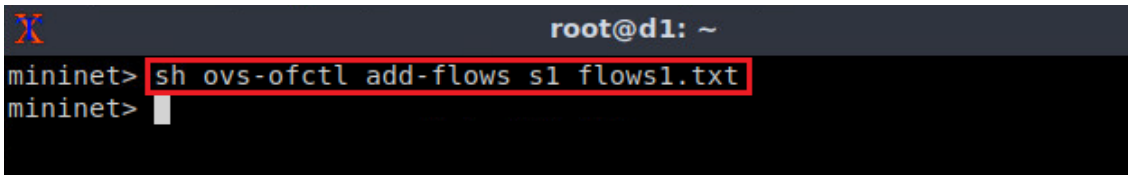

Figure 33. Adding flow entries to switch s1.

Step 3. In this step, you will configure a VXLAN tunnel endpoint (VTEP) that will enable outgoing traffic from switch s1 to the outer network. A script is written to facilitate this process. To execute the script, type the following command.

```
sh ./vxlan_cmd1.cmd
```

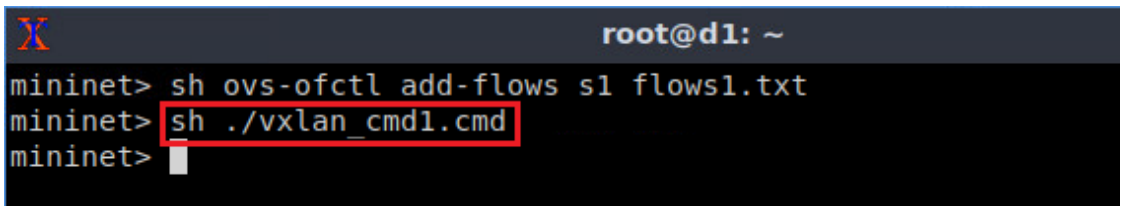

Figure 34. Enabling outgoing traffic in switch s1.

VTEP is the device responsible for encapsulating and de-encapsulating layer 2 traffic. This device is the connection between the overlay and the underlay network. In this case, the VTEP is configured to provide connectivity between the switches and the containers' egress interfaces.

Step 4. In container d2, issue the following command to add entries to the flow table of switch s2.

```
sh ovs-ofctl add-flows s2 flows2.txt
```

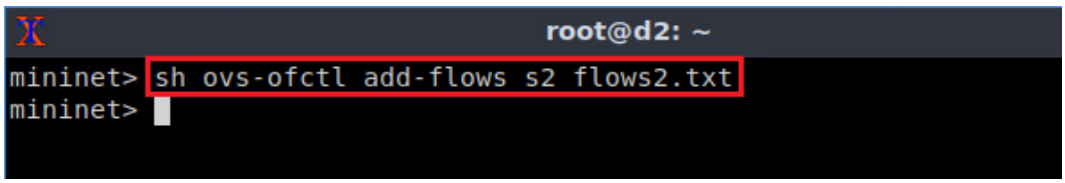

Figure 35. Adding flow entries to switch s2.

Step 5. Similarly, in container d2, type the command below to configure a VXLAN tunnel endpoint (VTEP) in order that enables outgoing traffic from switch s2 to the outer network, issue the following command:

```
sh ./vxlan_cmd2.cmd
```

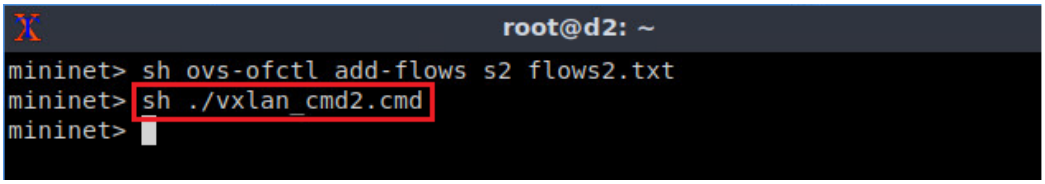

Figure 36. Enabling outgoing traffic in switch s2.

5 Verifying configuration

In this section, the user will verify that the VXLAN tags were applied accordingly. Notice that the traffic between h1 and h2 has the VXLAN tag 100 and, the traffic between h11 and h22 corresponds to the VXLAN tag 200. This tag is known as the VXLAN Network Identifier (VNI). The VNI is used to identify VXLAN traffic.

5.1 Performing connectivity test between end-hosts

The following steps aim to verify the connectivity between end-hosts. This means that there should be connectivity between h1 and h2, also between h11 and h22.

Step 1. In container d1 terminal, issue the following command to verify the connectivity between host h1 and host h11. Notice that `h1` specifies host 1 as the source.

```
h1 ping 10.0.0.2
```

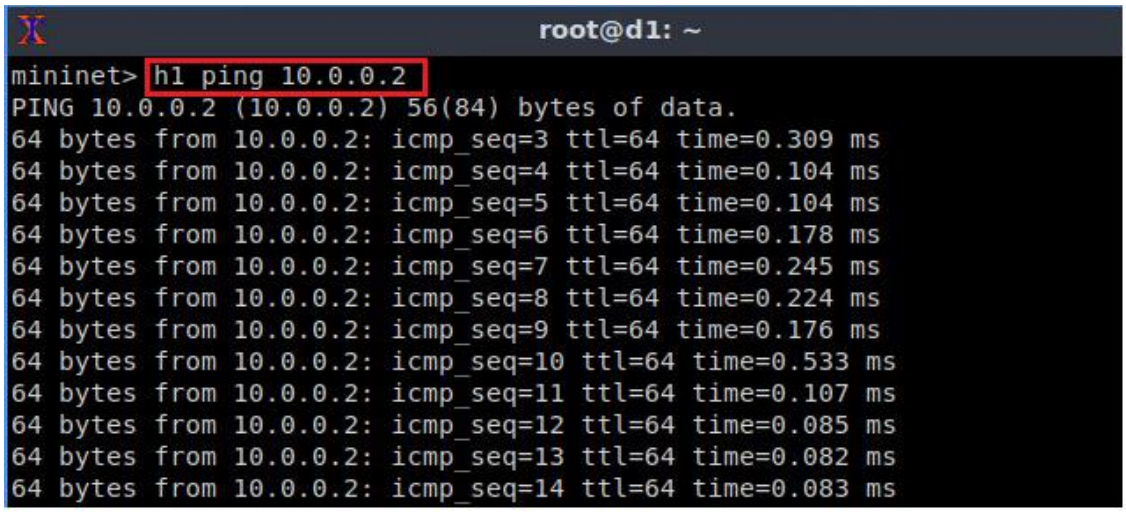

Figure 37. Performing a connectivity test between host h1 and host h2.

Consider Figure 37. The results show a successful connectivity test.

Step 2. In container d2 terminal, issue the command shown below to disable the network interface of host h2.

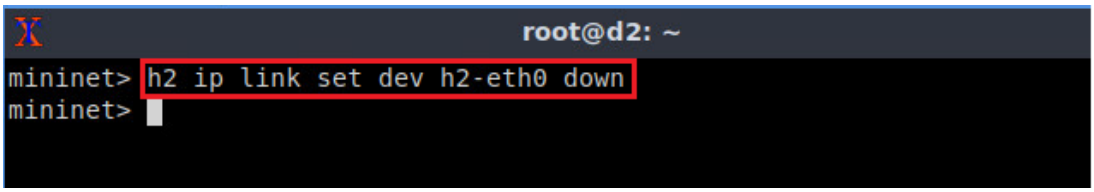

Figure 38. Disabling h2 network interface.

Step 3. Click on container d1 terminal. The user will verify that the connectivity is lost. Press `Ctrl+c` to stop the test.

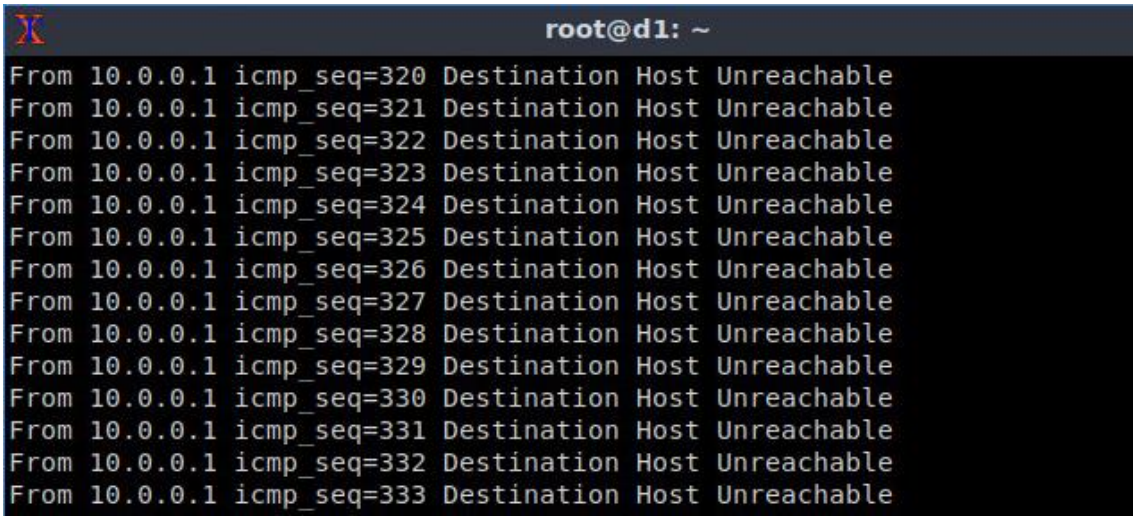

Figure 39. Verifying connectivity between host h1 and host h2.

Step 4. In container d1 terminal, issue the following command to test the connectivity between host h11 and host h22. Notice that `h11` specifies host 11 as the source.

```
h11 ping 10.0.0.2
```

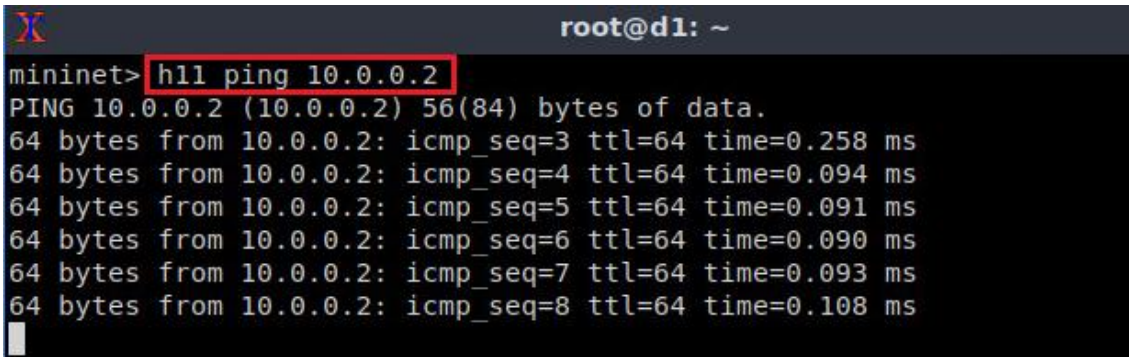

Figure 40. Performing a connectivity test between host h11 and host h22.

The results will display a successful connectivity test. Do not stop the connectivity test.

5.2 Verifying VXLAN network identifiers using Wireshark

The following steps show how to verify VXLAN network identifiers using Wireshark network analyzer. The identifiers are used by the switch to isolate network traffic.

Step 1. Click on router r1 terminal and issue the following command to exit the vtysh session.

```
exit
```

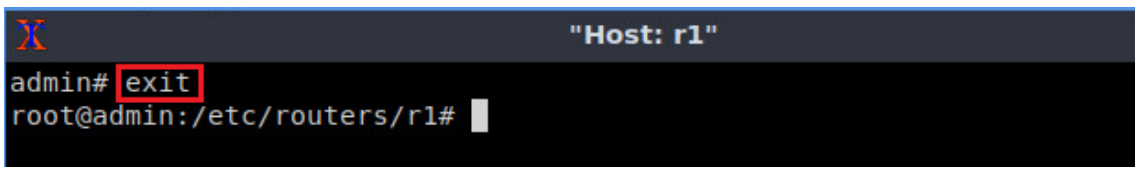

Figure 41. Exiting from vtysh.

Step 2. In router r1 terminal, start Wireshark dissector by issuing the following command. A new window will emerge.

```
wireshark
```

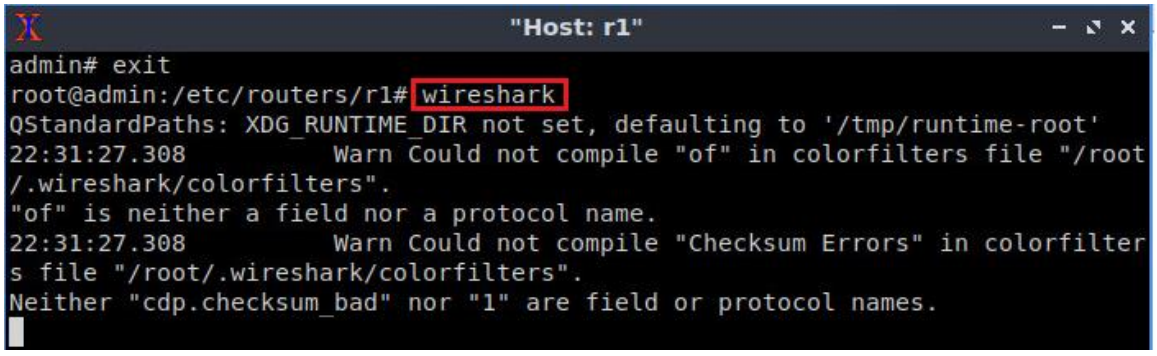

Figure 42. Starting Wireshark network analyzer.

After executing the above command on router r1 terminal, Wireshark window will open, where you monitor different interfaces related to router r1.

Step 3. Click on interface *r1-eth0* then on the icon located on upper left-hand side to start capturing packets on this interface.

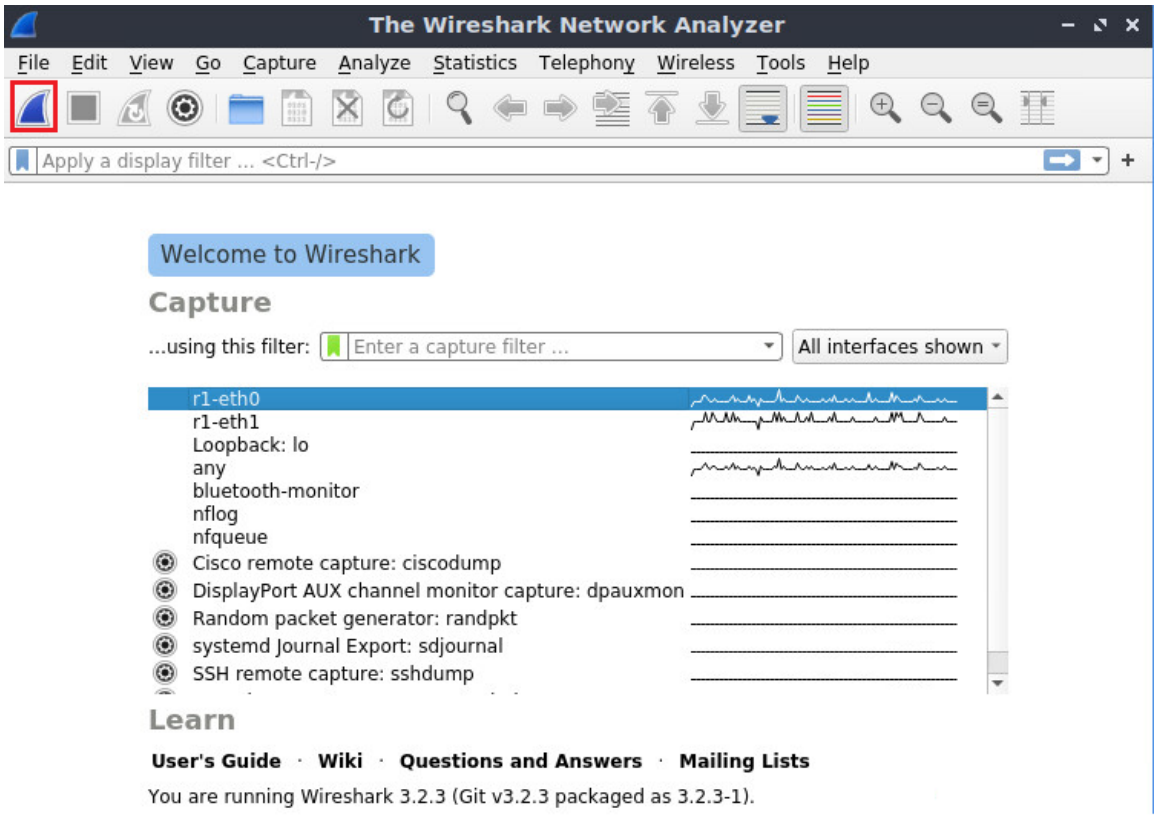

Figure 43. Starting packet capturing on interface r1-eth0.

Step 4. In the filter box located in upper left-hand side, type *vxlan* in order to filter the packets that contains VXLAN tags.

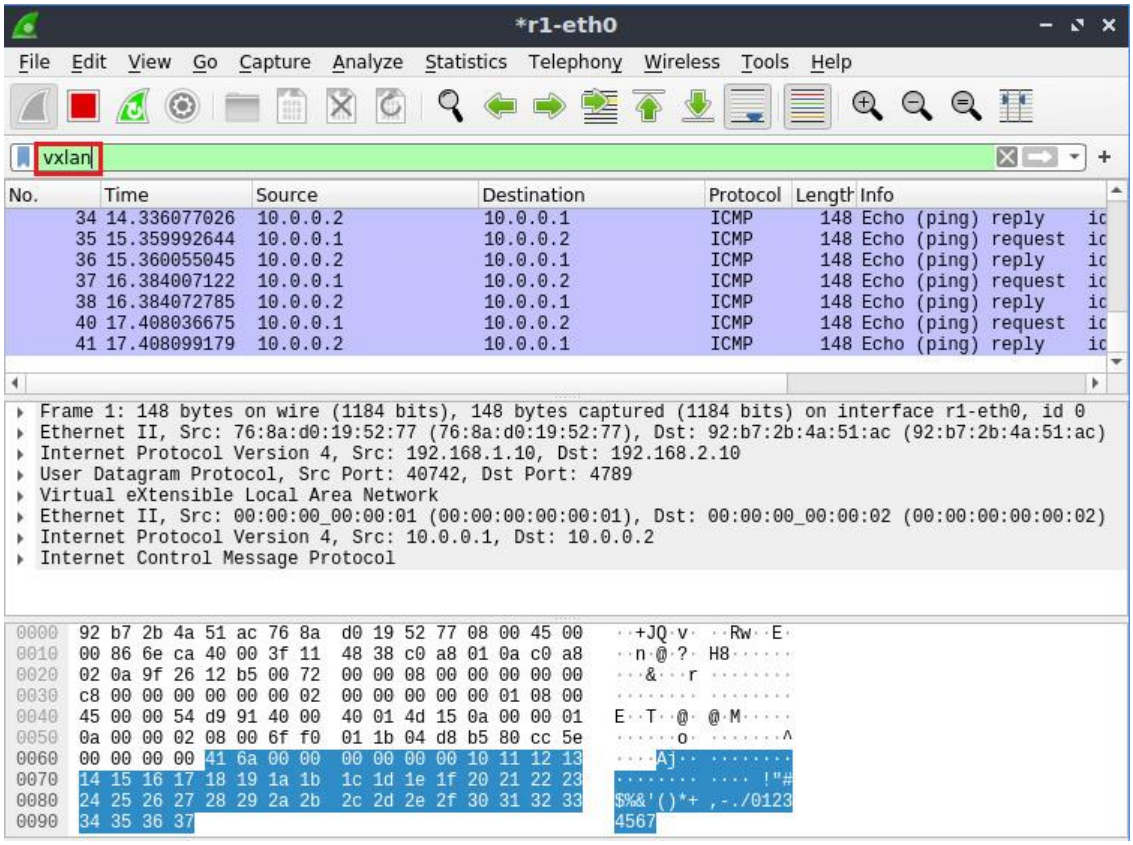

Figure 44. Filtering network traffic.

Step 5. Click on the arrow located on leftmost of the field called *Virtual eXtensible Local Area Network.* A list will be displayed. Verify that the *VXLAN Network Identifier* is 200. Notice that such tag corresponds to the traffic from h11 to h22.

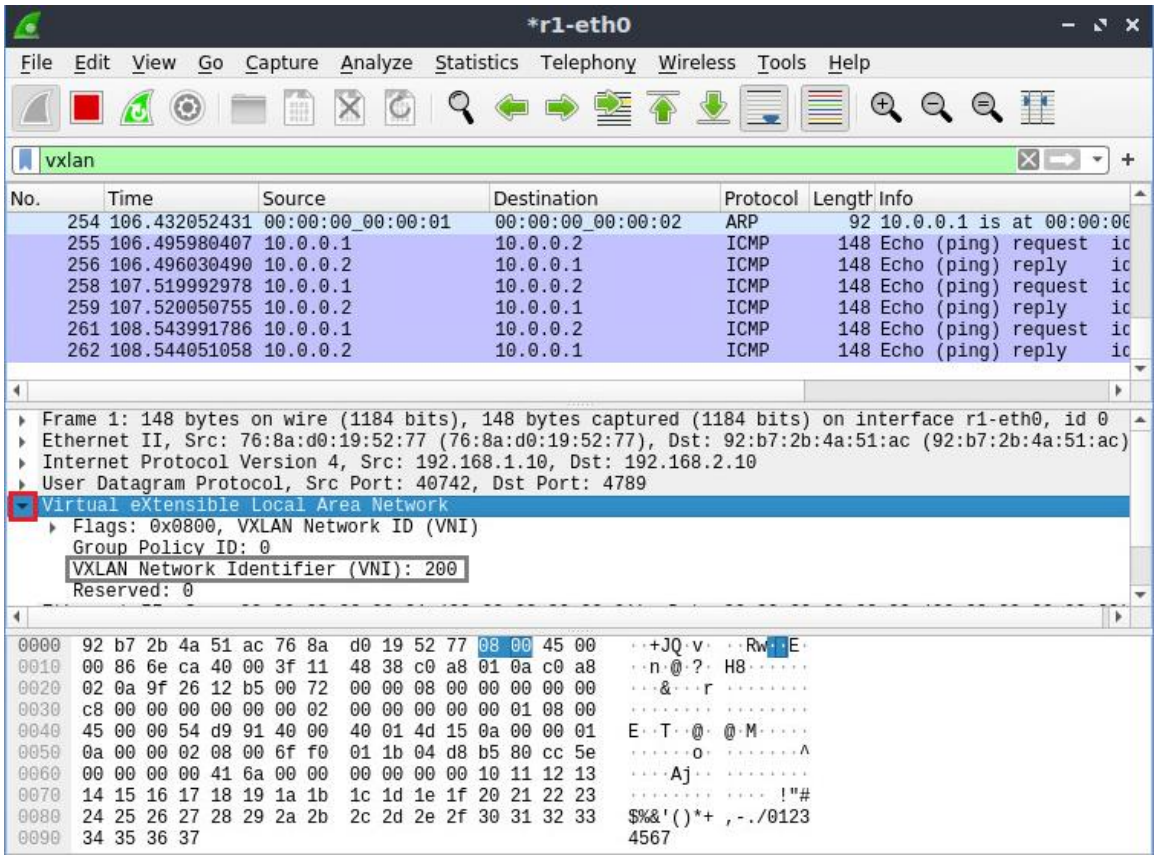

Figure 45. Verifying VXLAN network identifier.

Step 6. To stop packet capturing, click on the red button located on the upper left-hand side.

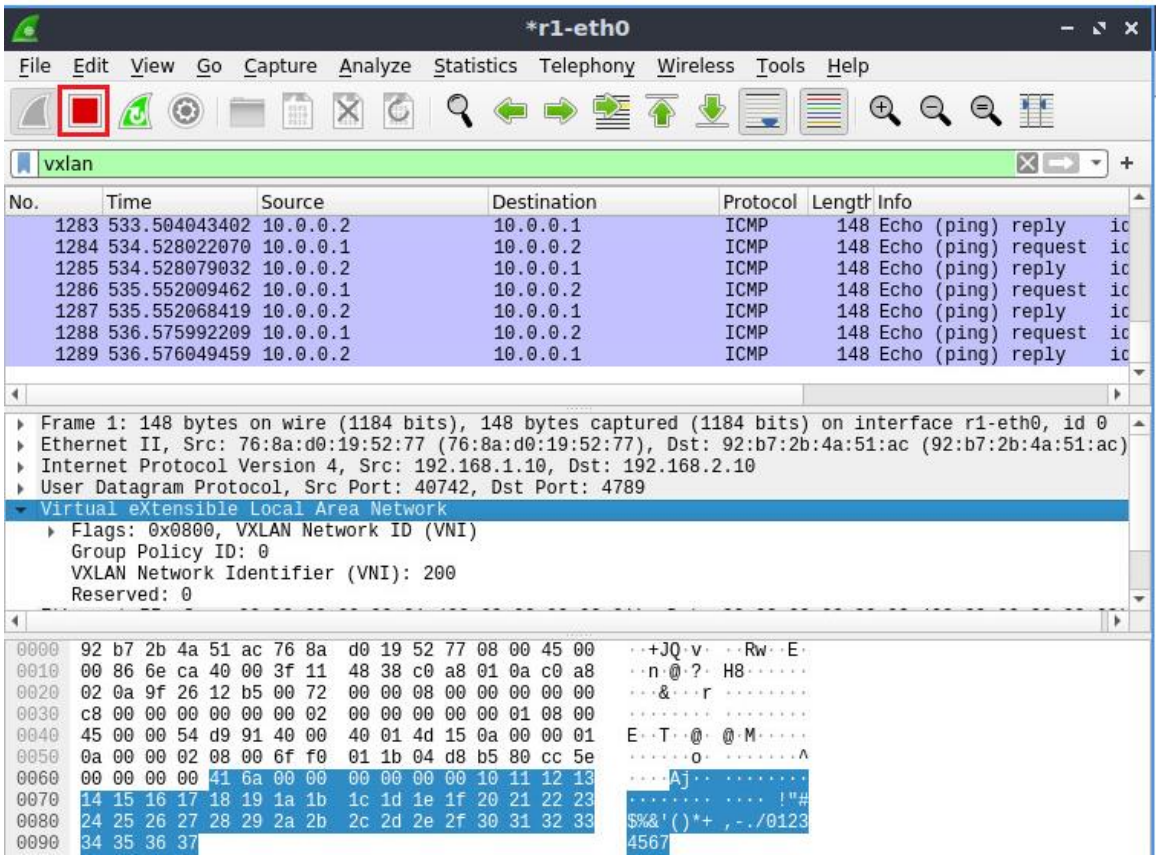

Figure 46. Stopping packet capturing.

Step 7. In container d1, press `ctrl+c` to stop the test.

Step 8. In container d2 terminal, re-enable the network interface in host h2 by issuing the following command:

```
h2 ip link set dev h2-eth0 up
```

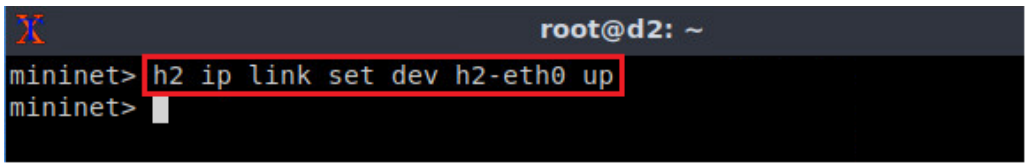

Figure 47. Enabling interface h2-eth0.

Step 9. Perform a connectivity test between h1 and h2 by issuing the following command:

```
h1 ping 10.0.0.2
```

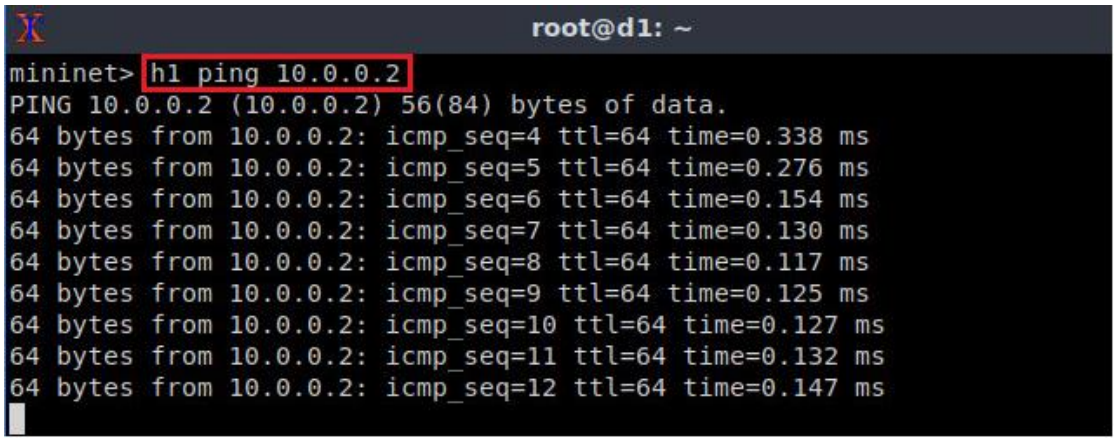

Figure 48. Performing a connectivity test between host h1 and host h2.

Consider Figure 48. The results show a successful connectivity test.

Step 10. In Wireshark window, start packet capturing by clicking on the button located on upper left-hand side.

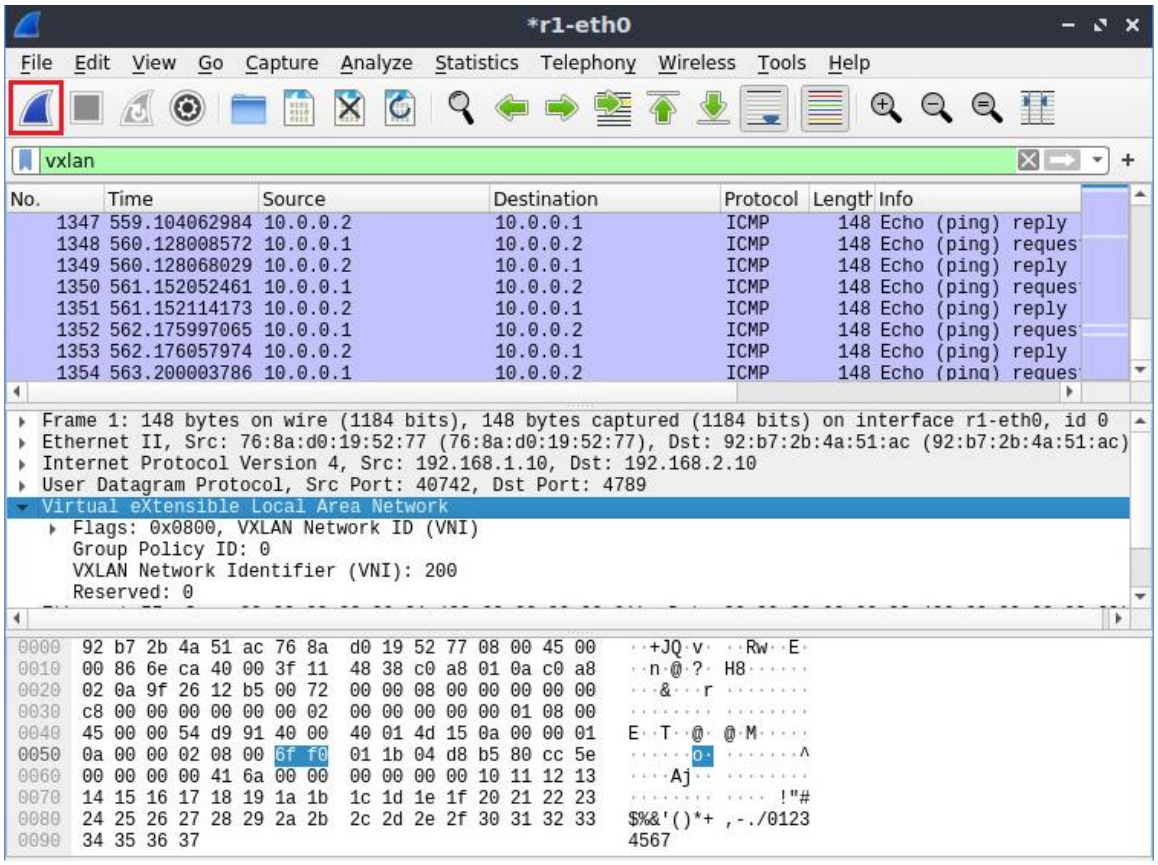

Figure 49. Starting packet capturing.

Step 11. A notification window will be prompted. Click on *Continue without Saving* to proceed.

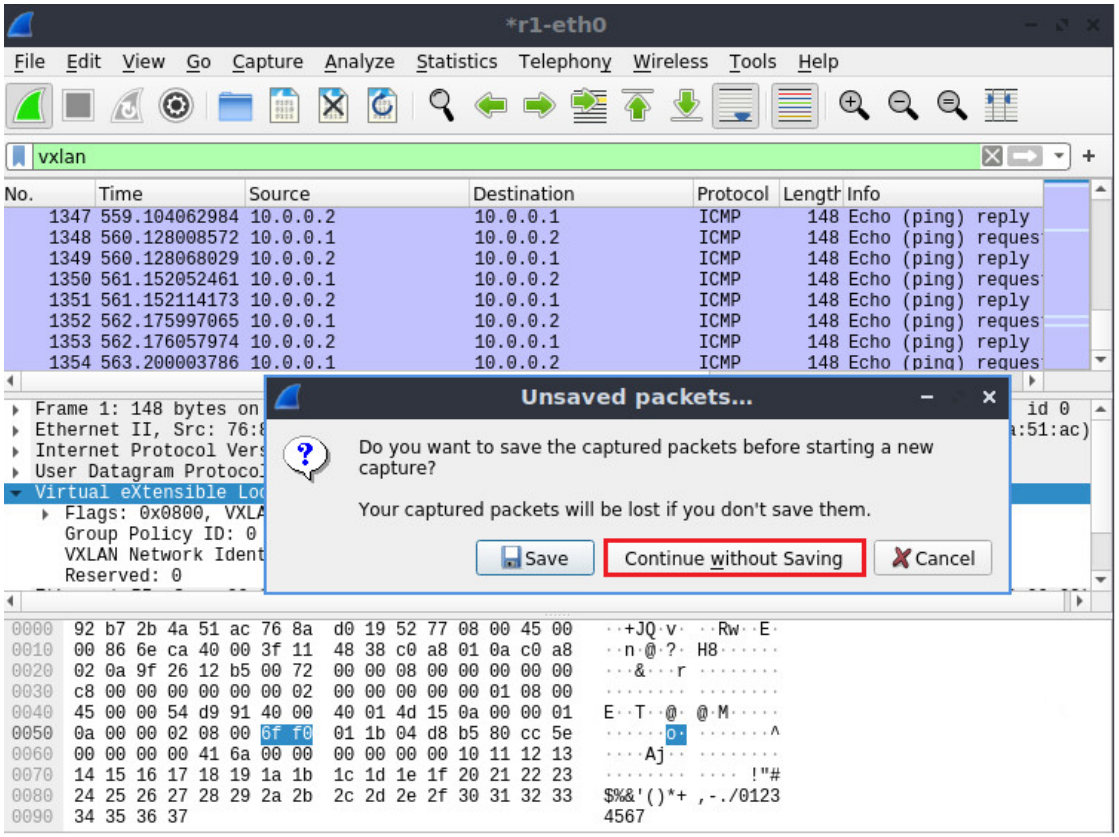

Figure 50. Closing without saving previous packet capture.

Step 12. Verify that the VXLAN Network Identifier is 100. Notice that such tag corresponds to the traffic from h1 to h2.

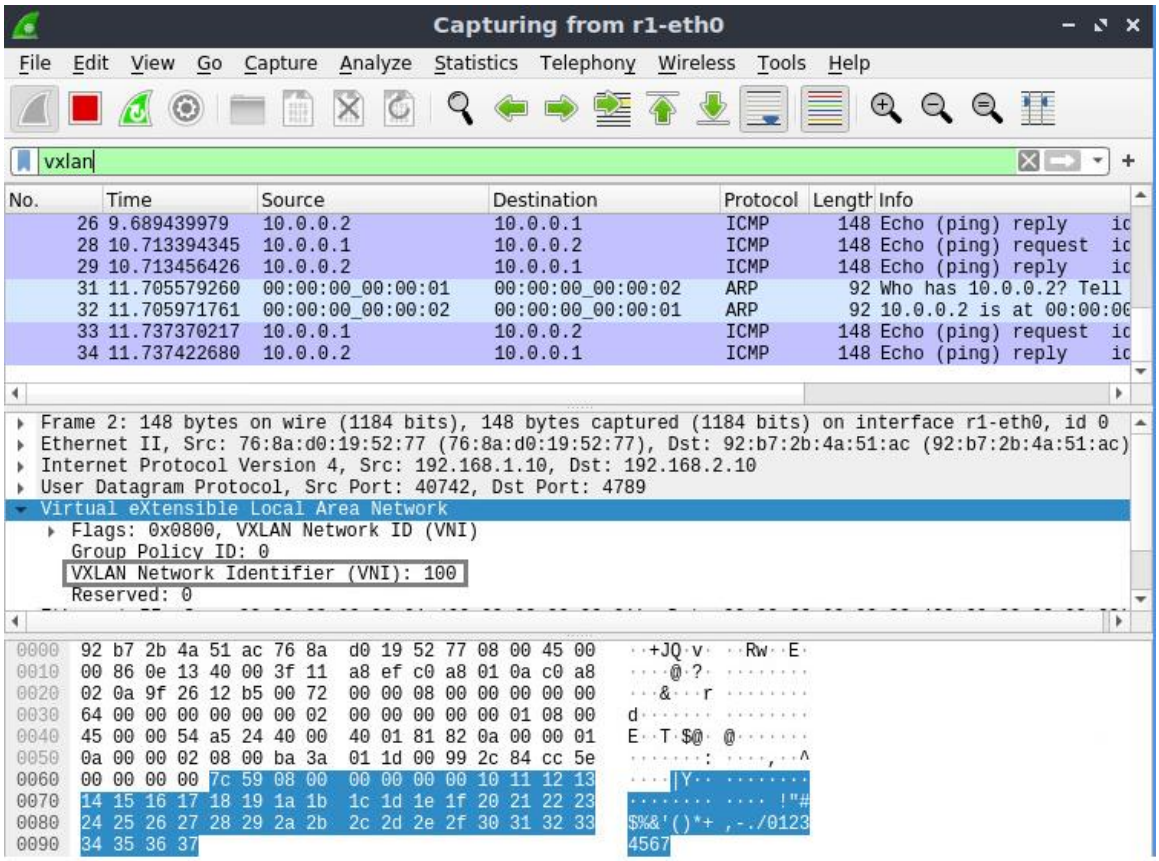

Figure 51. Verifying VXLAN network identifier.

This concludes Lab 5. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. Mininet walkthrough, [Online]. Available: http://mininet.org.
2. Peuster, Manuel, Johannes Kampmeyer, and Holger Karl. "*Containernet 2.0: A rapid prototyping platform for hybrid service function chains.*" *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018.
3. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "*OpenFlow: enabling innovation in campus networks*." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.
4. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach*". Morgan Kaufmann, 2016.
5. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, "*ONOS: towards an open, distributed SDN OS*," In Proceedings of the third workshop on Hot topics in software defined networking, pp. 1-6, 2014.
6. Mahalingam, Mallik, et al. "*Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*." *RFC* 7348 (2014): 1-22.
7. Juniper Networks, "*Understanding EVPN with VXLAN Data Plane Encapsulation*", [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/concept/evpn-vxlan-data-plane-encapsulation.html.

8. Qu, Xiaorong, Weiguo Hao, and Yuanbin Yin. "*L3 gateway for VXLAN.*" U.S. Patent No. 8,923,155. 30 Dec. 2014.
9. Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." *Linux journal* 2014.239 (2014): 2.
10. Linux foundation collaborative projects, "*FRRouting: what's in your router*", [Online]. https://frrouting.org/

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 6: Introduction to OpenFlow

Document Version: 28-05-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab is an introduction to OpenFlow, which defines both the communications protocol between the Software Defined Networking (SDN) data plane and the SDN control plane, and part of the behavior of the data plane. In this lab, you will use the ovs-ofctl command line utility to administer OpenFlow switches, such as inserting/deleting flows. The focus in this lab is to understand and inspect the OpenFlow messages exchanged between the control plane and the data plane.

Objectives

By the end of this lab, the user will:

1. Understand SDN and its components.
2. Understand OpenFlow.
3. Configure OpenFlow switches using ovs-ofctl.
4. Configure ONOS controller
5. Use Wireshark network analyzer to capture OpenFlow packets.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Monitoring and administering OpenFlow switches.
4. Section 4: Capturing OpenFlow packets.

1 Introduction

1.1 Data, control and management planes

The various switching functions are traditionally segregated into three separate categories. Since each category may be capable of horizontal communication with peer elements in adjacent entities in a topology, and also capable of vertical communication with the other categories, it has become common to represent each of these categories as a layer or *plane*. Peer communication occurs in the same plane, and cross-category messaging occurs in the third dimension, between planes[3].

Consider Figure 1. The vast majority of packets handled by the switch are only managed by the *data plane*. The data plane consists of the various ports that are used for the reception and transmission of packets and a forwarding table with its associated logic. The data plane assumes responsibility for packet buffering, packet scheduling, header modification, and forwarding. If an arriving data packet's header information is found in the forwarding table, it may be subject to some header field modification and then will be forwarded without any intervention of the other two planes[3].

Not all packets can be handled in that way, sometimes simply because their information is not yet entered into the table, or because they belong to a control protocol that must be processed by the *control plane.* The control plane, (see Figure 1), is involved in many activities. Its principal role is to keep current the information in the forwarding table so that the data plane can independently handle as high a percentage of the traffic as possible. The control plane is responsible for processing a number of different control protocols that may affect the forwarding table, depending on the configuration and type of switch. These control protocols are jointly responsible for managing the active topology of the network[3].

The third plane depicted in Figure 1 is the *management plane*. Network administrators configure and monitor the switch through this plane, which in turn extracts information from or modifies data in the control and data plane as appropriate. The network administrators use some form of network management system to communicate with management plane in a switch[3].

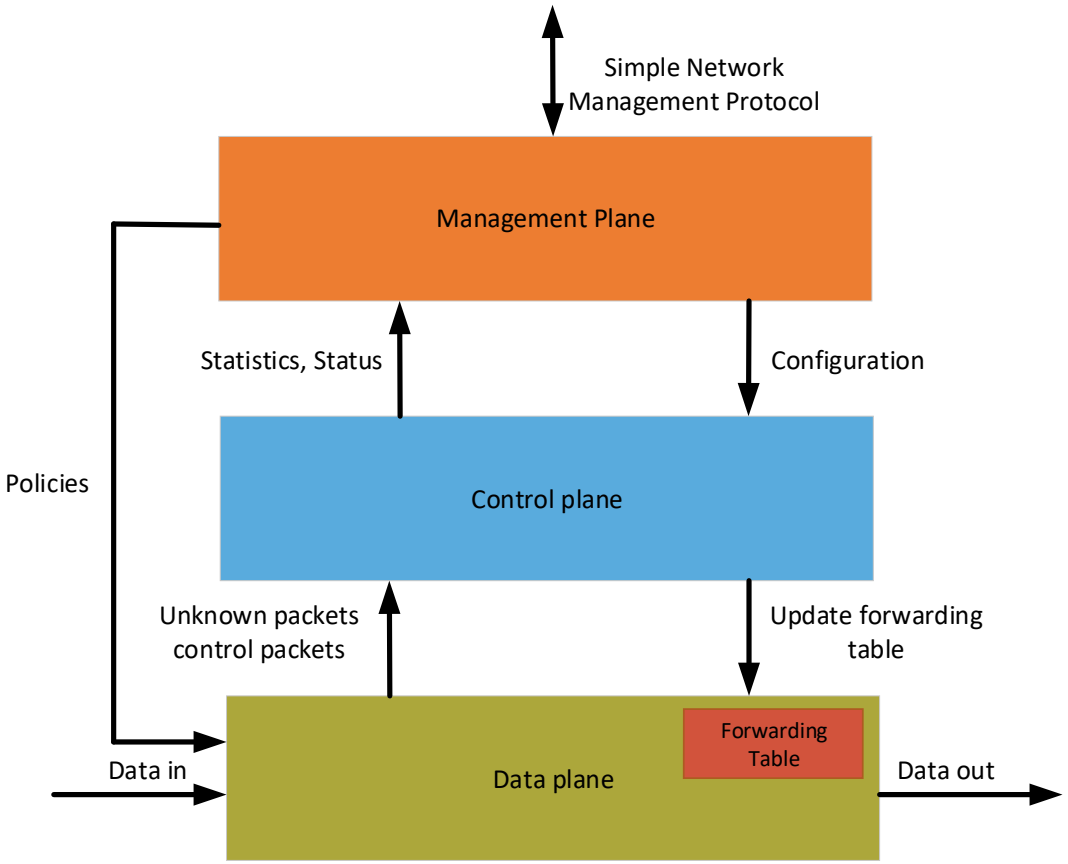

Figure 1. Roles of the control, data and management planes[3].

1.2 OpenFlow Overview

OpenFlow defines both the communication protocol between the SDN data plane and the SDN control plane, as well as part of the behavior of the data plane. It does not describe the behavior of the controller itself. There are other approaches to SDN, but today OpenFlow is the only nonproprietary, general-purpose protocol for programming the forwarding plane of SDN switches[3].

Consider Figure 2. In a basic component of OpenFlow system, there is always an OpenFlow controller that communicates to one or more OpenFlow switches. The OpenFlow protocol defines the specific messages and message formats exchanged between the controller (control plane) and the device (data plane). The OpenFlow behavior specifies how the device should react in various situations and how it should respond to commands from the controller.

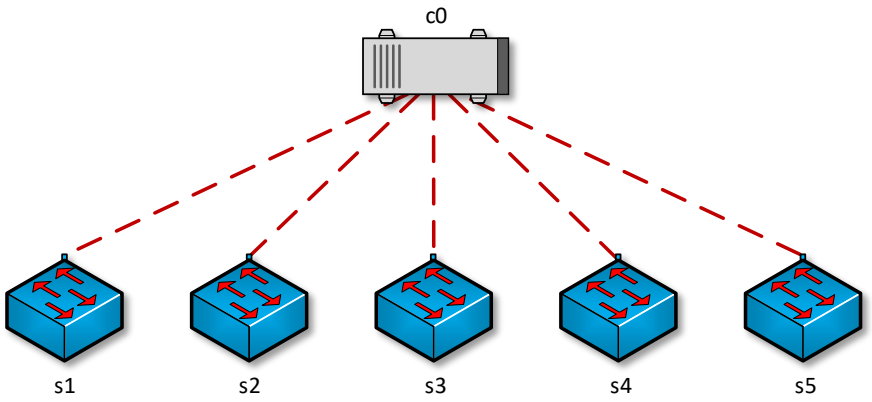

Figure 2. OpenFlow components.

1.3 OpenFlow components

In a packet switch, the core function is to take packets that arrive on one port and forward them through another port. OpenFlow switches perform this operation using the packet-matching function with the flow table. Thus, once a packet arrives to the switch, the latter will look up in its flow table and check if there is a match. Consequently, the switch will decide which action to take based on the flow table. The action could be:

- Forward the packet out a local port
- Drop the packet
- Pass the packet to the controller.

The basic functions of an OpenFlow switch and its relationship to a controller is depicted in Figure 3. When the data plane doesn't have a match to the incoming packet, it sends a PACKET_IN message to the controller. The control plane runs routing and switching protocols and other logic to determine what the forwarding tables and logic in the data plane should be. Consequently, when the controller has a data packet to forward out through the switch, it uses the OpenFlow PACKET_OUT message. All the communication between OpenFlow controller and data plane are defined by the OpenFlow protocol.

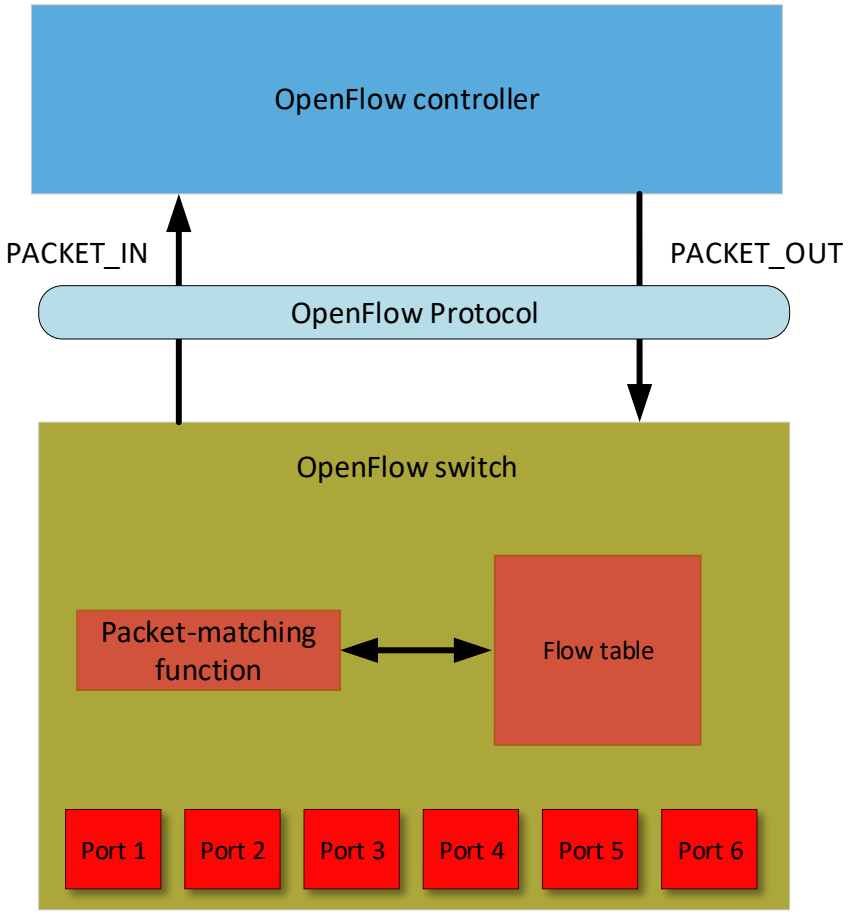

Figure 3. OpenFlow switch.

2 Lab topology

Consider Figure 4. The topology consists of two end-hosts, a switch and a controller. The blue device is an OpenFlow switch and it is directly connected to the controller c0.

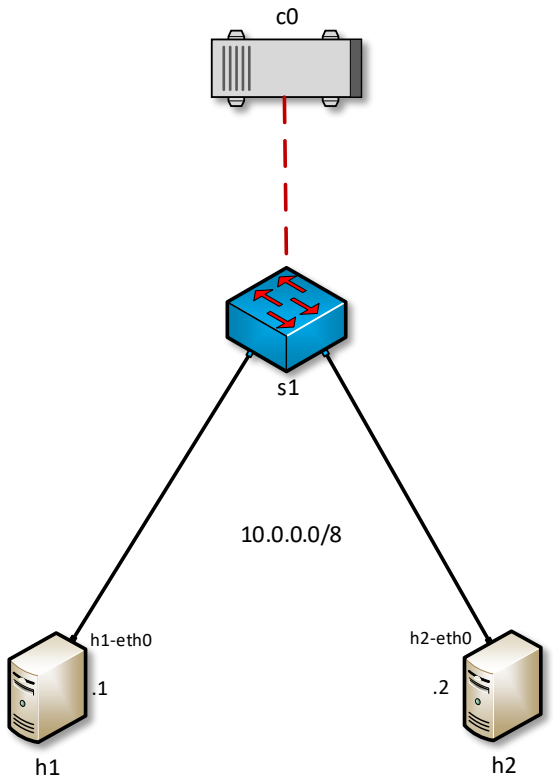

Figure 4. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet
h1	h1-eth0	10.0.0.1	/8
h2	h2-eth0	10.0.0.2	/8
c0	n/a	127.0.0.1	/32

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), saving the topology, and exporting a layer 2 model.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

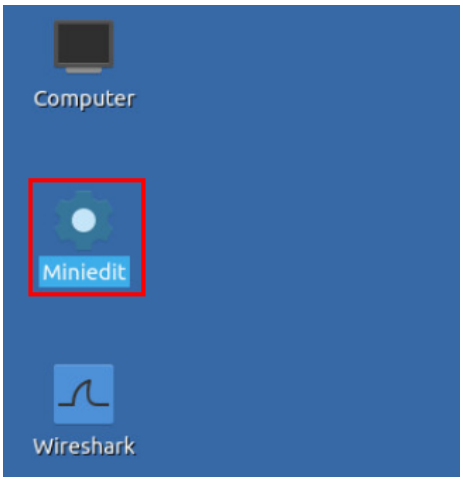

Figure 5. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab6.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab6* and click on *Open*.

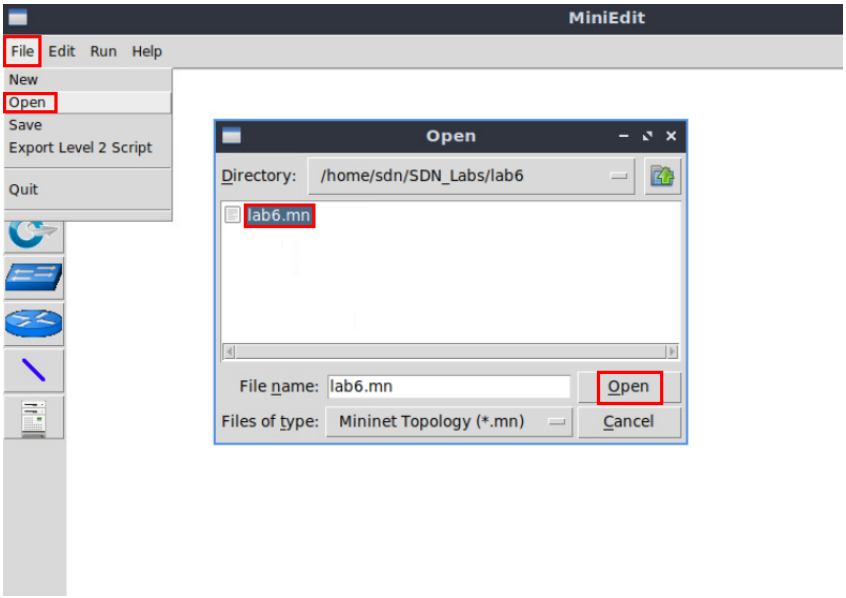

Figure 6. Opening topology.

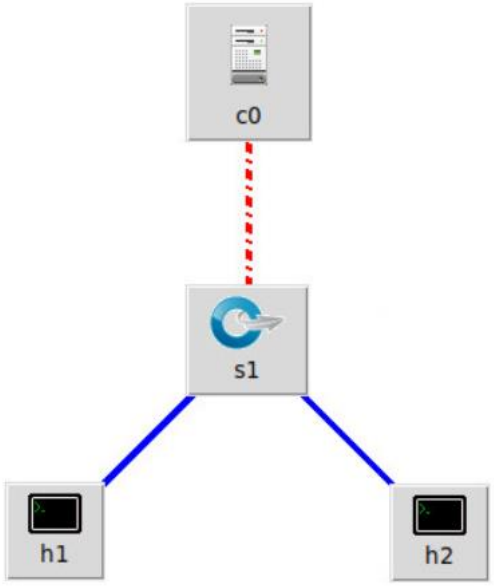

Figure 7. MiniEdit's topology.

Step 3. Click on the *Run* button to start the emulation. The emulation will start and the buttons of the MiniEdit panel will gray out, indicating that they are currently disabled.

Figure 8. Starting the emulation.

3 Monitoring and administering OpenFlow switches

In this section, you will use *ovs-ofctl* command line tool to monitor and administer OpenFlow switches. This tool can show the current state of an OpenFlow switch, including its features, configuration, and table entries.

Step 1. Open Linux terminal by clicking on the shortcut depicted below.

Figure 9. Opening Linux terminal.

Step 2. Issue the command below to execute programs with the security privileges of the superuser (root). When prompted for a password, type `password`.

```
sudo su
```

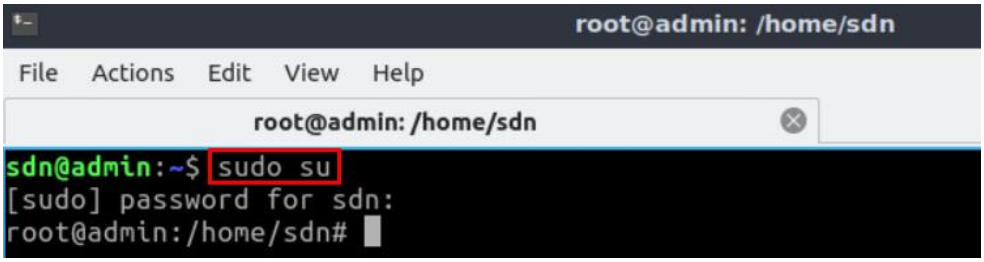

Figure 10. Switching to root mode.

Step 3. Issue the command below to connect to switch s1 and show its information.

```
ovs-ofctl show s1
```

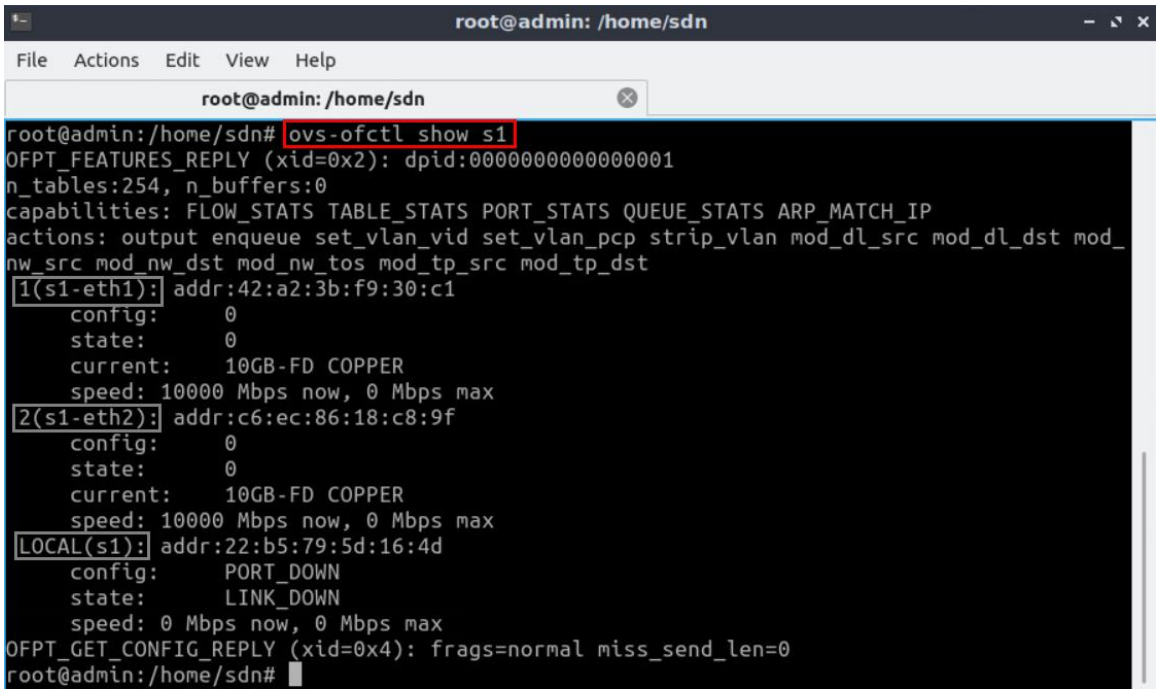

Figure 11. Showing switch s1 information.

Consider Figure 11. Switch s1 has three interfaces. Each interface displays the Media Access Control (MAC) address (`addr`) along with other information, such as the current state of the switch.

Step 4. Issue the command below to print the flow entries of switch s1.

```
ovs-ofctl dump-flows s1
```

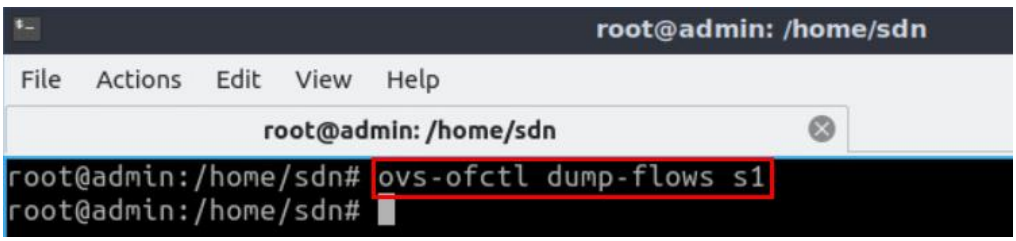

Figure 12. Showing the flow entries of switch s1.

Consider Figure 12. No output was shown in response to the above command. This is because initially the switch has no flow entries.

Step 5. Hold right-click on host h1 and select Terminal.

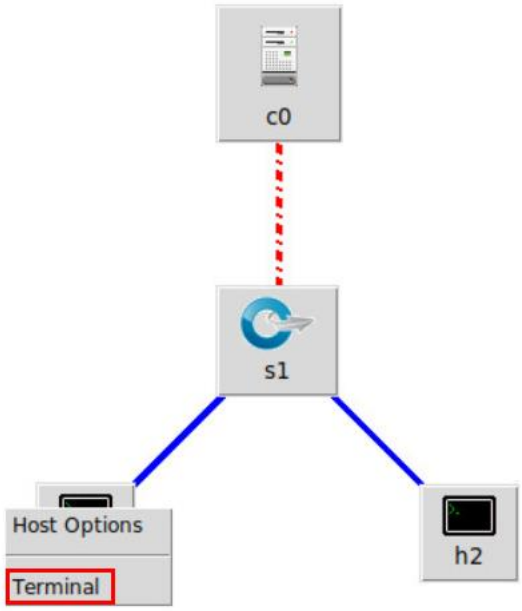

Figure 13. Opening host h1 terminal.

Step 6. Run a connectivity test by issuing the command shown below. The `ping` command is used to verify the connectivity between two ends. It must be followed by the IP address of the destination host, which is 10.0.0.2 (host h2) in this case. To stop the test, press `Ctrl+c`.

```
ping 10.0.0.2
```

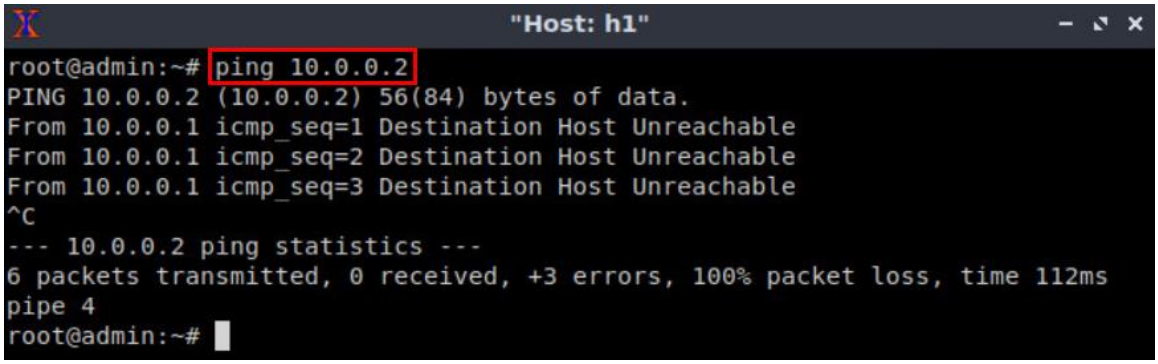

Figure 14. Pinging host h2 from host h1.

Consider Figure 14. The connectivity test is unsuccessful since switch s1 flow table is empty. Incoming traffic to the switch will not match any rule, and hence no action will be taken. Therefore, switch s1 doesn't know what to do with incoming traffic, leading to ping failure.

Step 7. Open Linux terminal.

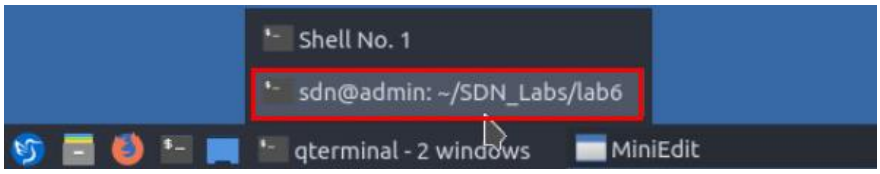

Figure 15. Opening Linux terminal.

Step 8. Issue the below command to manually install a flow into switch s1. The inserted flow forwards incoming packets at port 1 (`in_port=1`) to port 2 (`actions=output:2`).

```
ovs-ofctl add-flow s1 in_port=1,actions=output:2
```

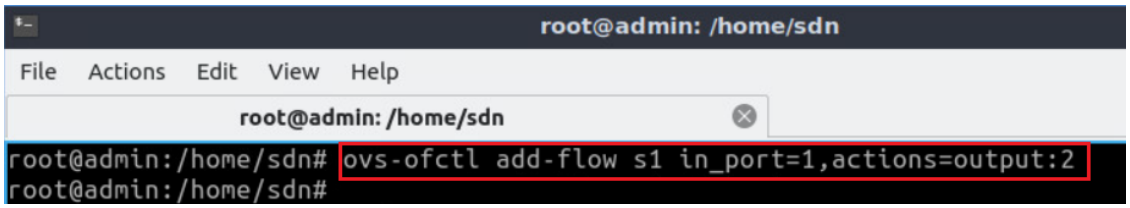

Figure 16. Adding a flow entry to switch s1.

Step 9. Issue the below command to manually install a flow into switch s1. The inserted flow forwards incoming packets at port 2 (`in_port=2`) to port 1 (`actions=output:1`).

```
ovs-ofctl add-flow s1 in_port=2,actions=output:1
```

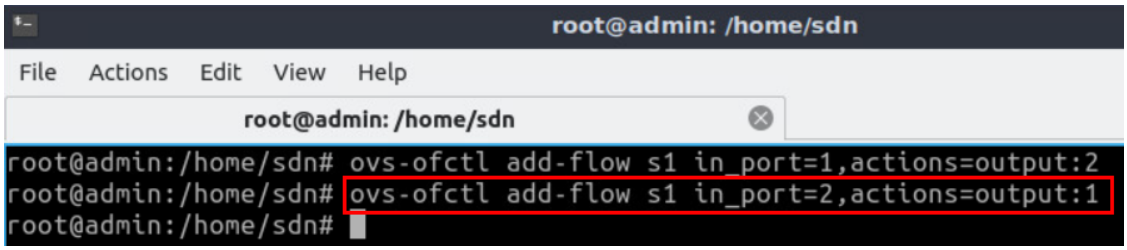

Figure 17. Adding a flow entry to switch s1.

Step 10. Issue the command below to print the flow entries of switch s1.

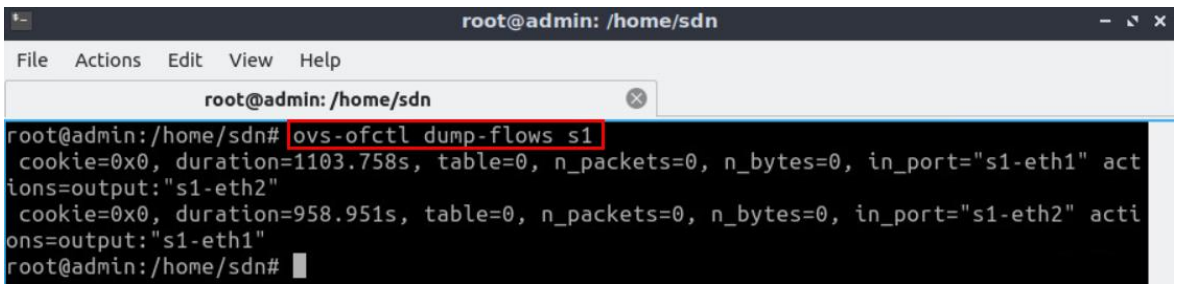

Figure 18. Showing the flow entries of switch s1.

Step 11. In host h1, run a connectivity test with host h2 by issuing the following command.

```
ping 10.0.0.2
```

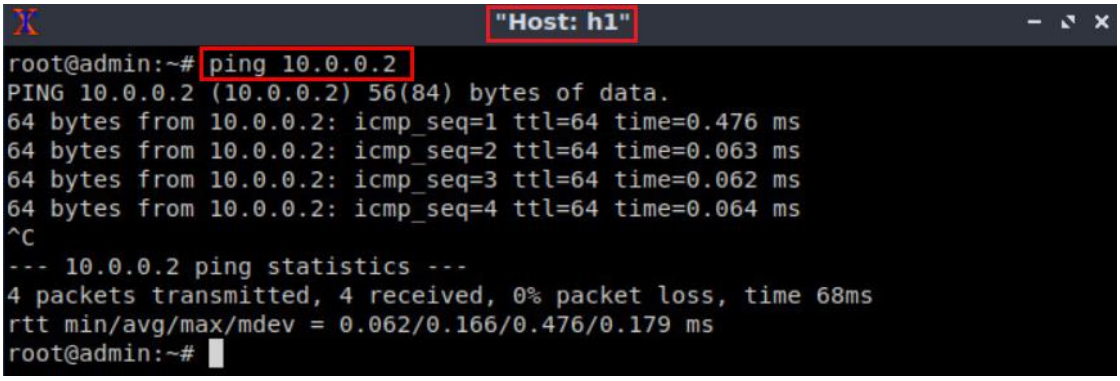

Figure 19. Pinging host h2 from host h1.

Step 12. In addition to adding flow entries to the switches using ovs-ofctl, you can also delete entries as well as deleting the whole flow table. Issue the following command on Linux terminal to delete the flow table of switch s1.

```
ping 10.0.0.2
```

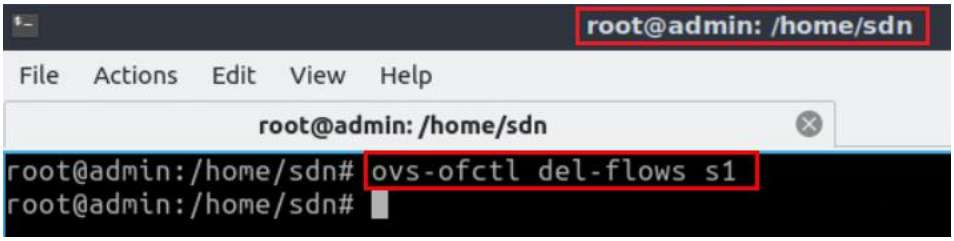

Figure 20. Deleting the flow table of switch s1.

4 Capturing OpenFlow packets

In this section, you will start Wireshark, navigate through some of its features, and learn how to monitor network traffic. Additionally, you will enable ONOS controller and capture OpenFlow packets.

4.1 Starting Wireshark

In this section, you will use Wireshark, the defacto network protocol analyzer, to monitor the network and inspect OpenFlow packets that are being transmitted between the controller and the data plane (switch).

Step 1. In Linux terminal, issue the following command to launch Wireshark.

```
wireshark &
```

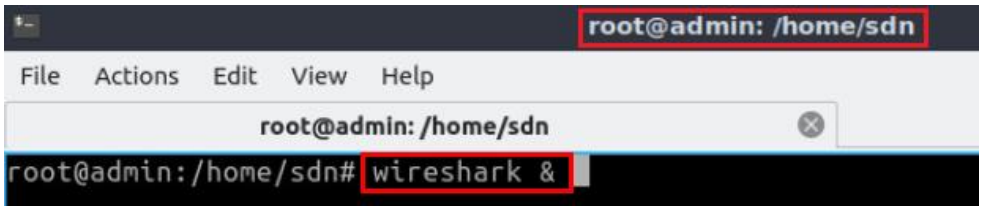

Figure 21. Starting Wireshark.

Wireshark window depicted in Figure 22 will appear after executing the above command.

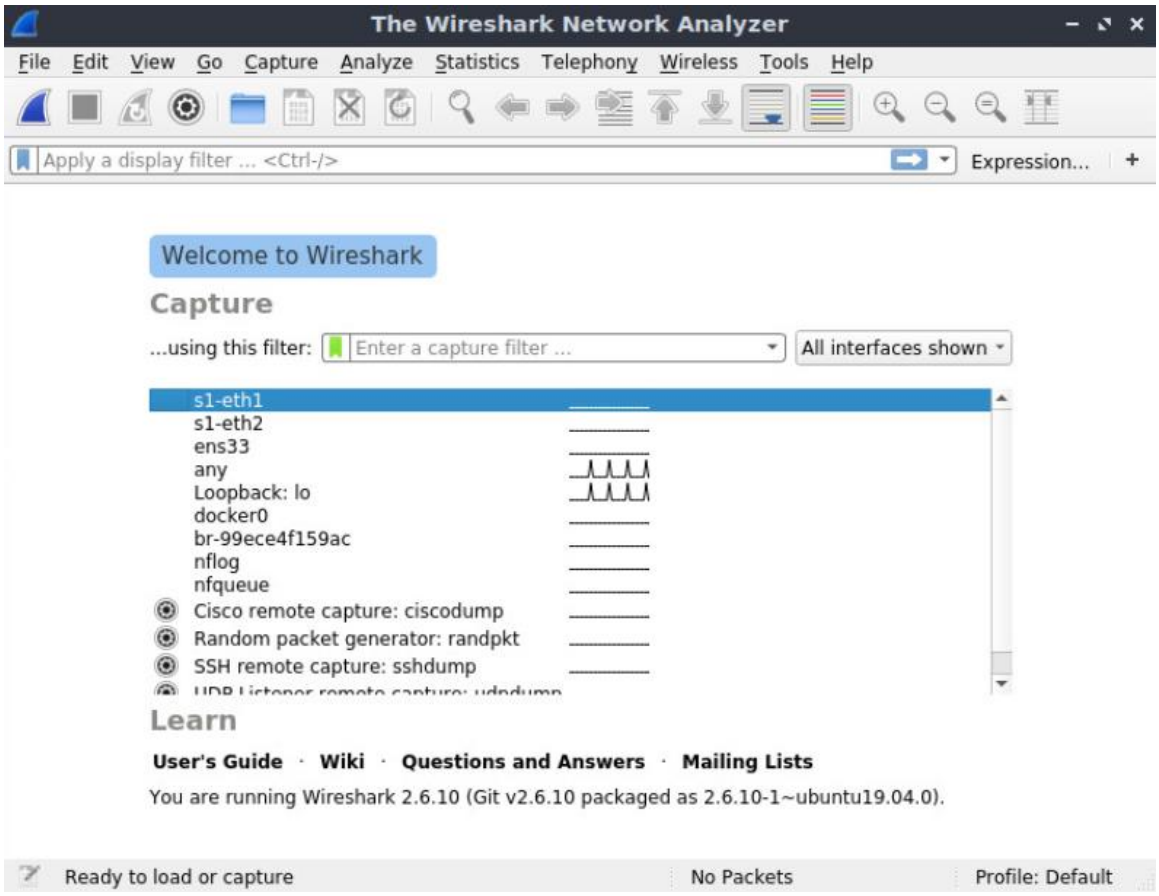

Figure 22. Wireshark window.

Step 2. In the opened Wireshark window, you will see a list of interfaces that Wireshark can capture network traffic on, such as s1-eth1, s1-eth2. Click on 'Loopback: lo' then start capturing the packets by clicking the 'shark fin' icon on the top left of the Window.

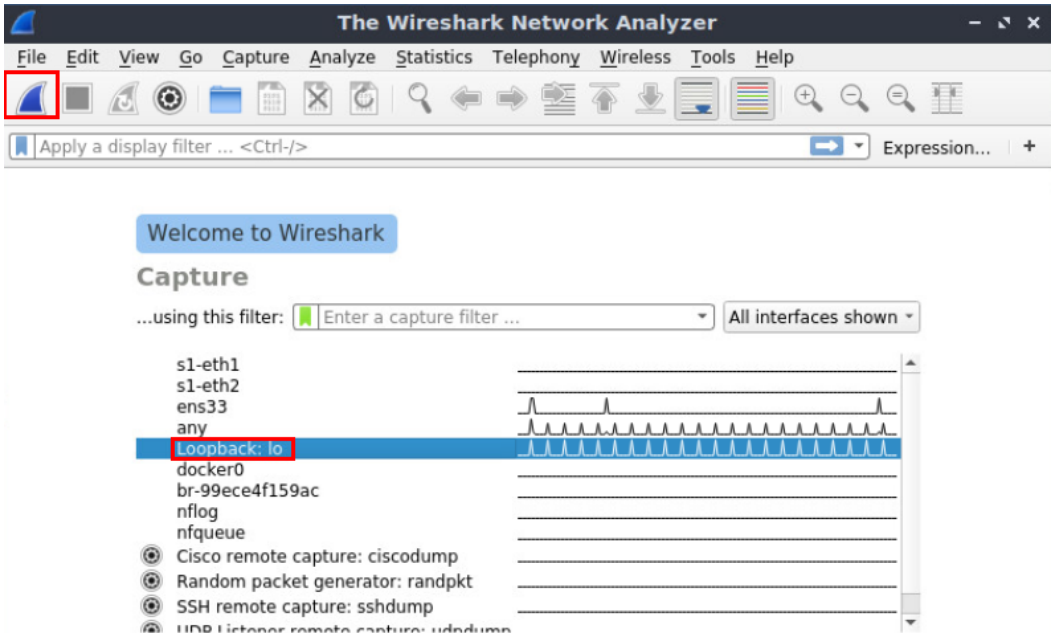

Figure 23. Start capturing packets in Wireshark.

Step 3. When you start capturing packets, you will notice that Wireshark is divided into three sections. The first section displays the captured packets including their number, time they were captured, source and destination IP addresses, protocol, length, and information of the packet. The second section contains detailed information about every capture packet (each selected packet will have its own information). The third section contains the real data that was captured in the packet.

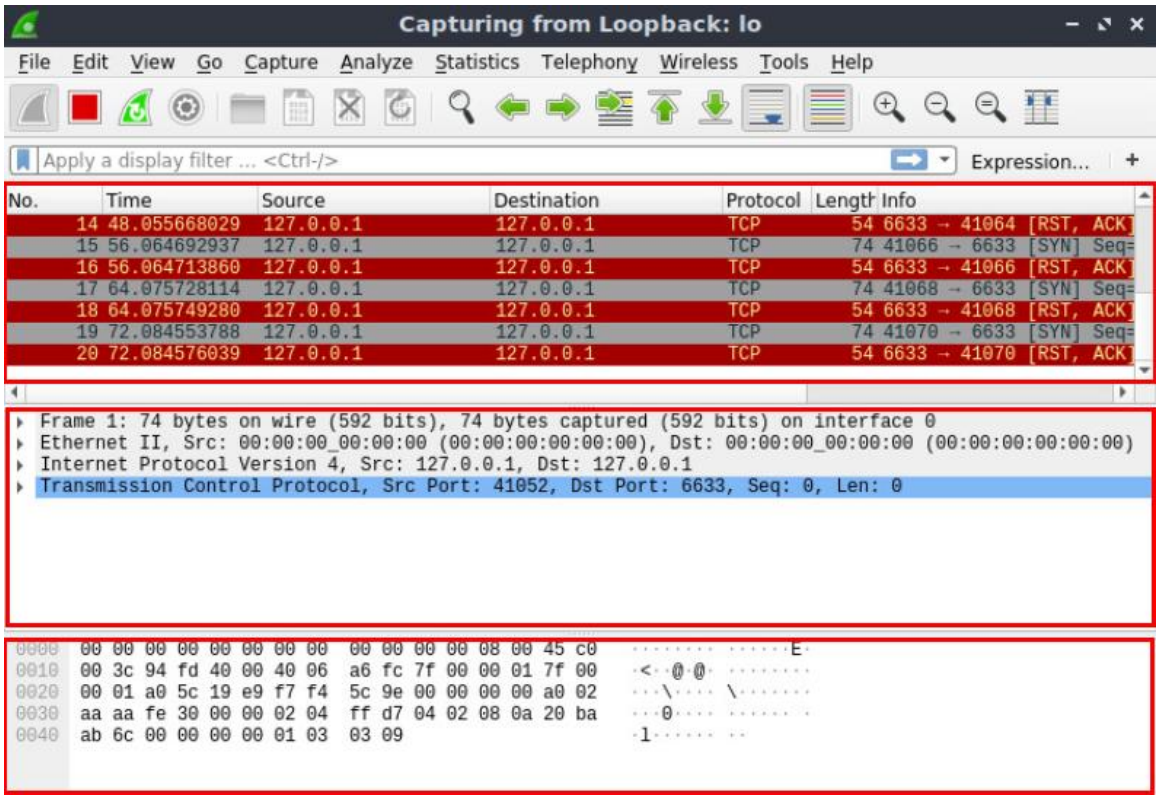

Figure 24. Capturing from Loopback: lo interface.

Step 4. Wireshark supports filters, i.e., you can apply filters to display a specific set of capture packets. To show OpenFlow packets only (protocol: OpenFlow), write the following expression in Wireshark filter text box, then hit Enter.

```
openflow_v1
```

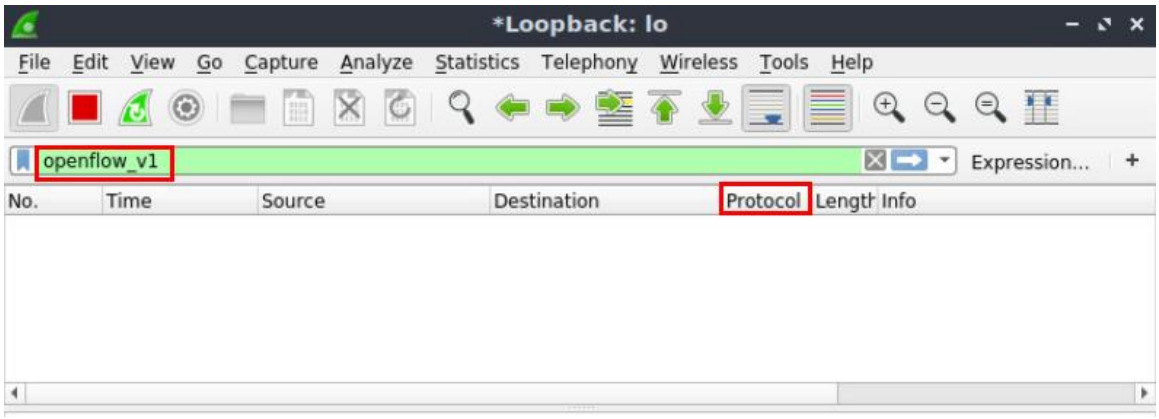

Figure 25. Capturing only OpenFlow packets in Wireshark.

Consider Figure 25. The applied filter in Wireshark displays packets with protocol type OpenFlow only, specifically. No packets appear in the packet capturing section since we haven't enabled the controller that exchanges OpenFlow packets with the data plane devices.

4.2 Starting ONOS controller

In this section, you will start ONOS controller and activate basic ONOS applications, such as OpenFlow application. The latter triggers the exchange of OpenFlow packets between the data plane (switch s1) and the control plane (c0). Thus, allowing the controller to discover the topology and insert flow entries into switch s1. Using Wireshark, you will capture the exchanged OpenFlow packets and understand their main types.

Step 1. In Linux terminal, where Wireshark was launched, issue the following command to exit the superuser mode.

```
exit
```

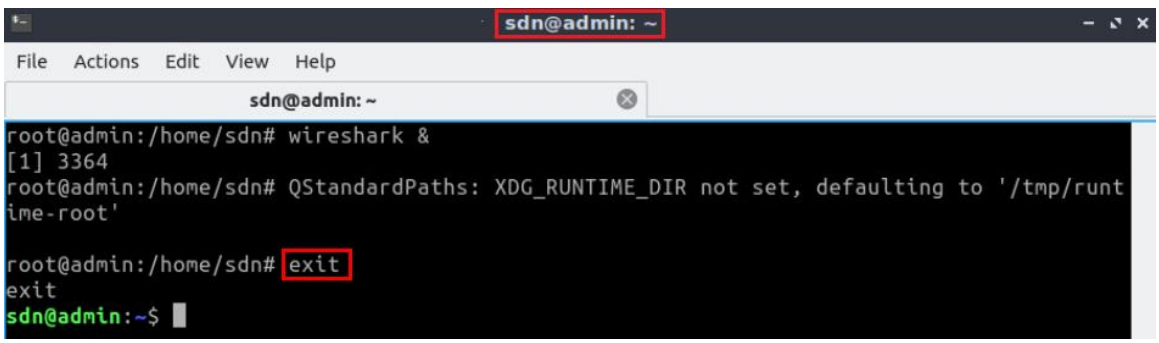

Figure 26. Exiting superuser mode.

Step 2. Navigate into *SDN_Labs/lab6* directory by issuing the following command. This folder contains the script responsible for starting ONOS. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab6
```

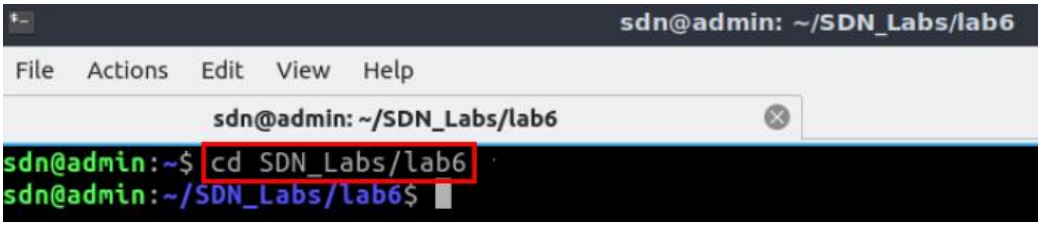

Figure 27. Entering the *SDN_Labs/lab6* directory.

Step 3. A script was written to run ONOS and enter its Command Line Interface (CLI). In order to run the script, issue the following command.

```
./run_onos.sh
```

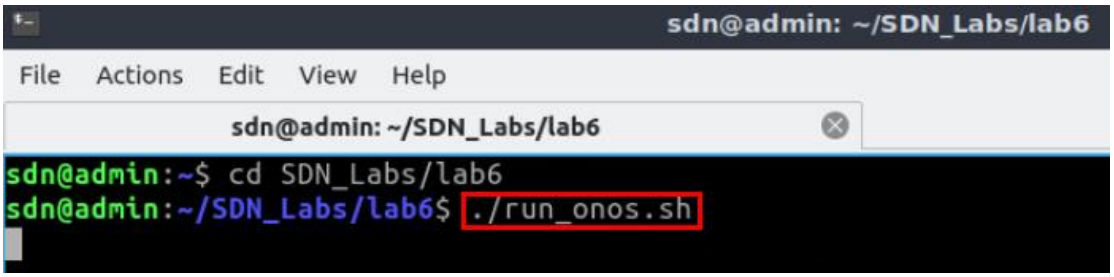

Figure 28. Starting ONOS.

Once the script finishes executing and ONOS is ready, you will be able to execute commands on ONOS CLI as shown in the figure below. Note that this script may take a couple of minutes.

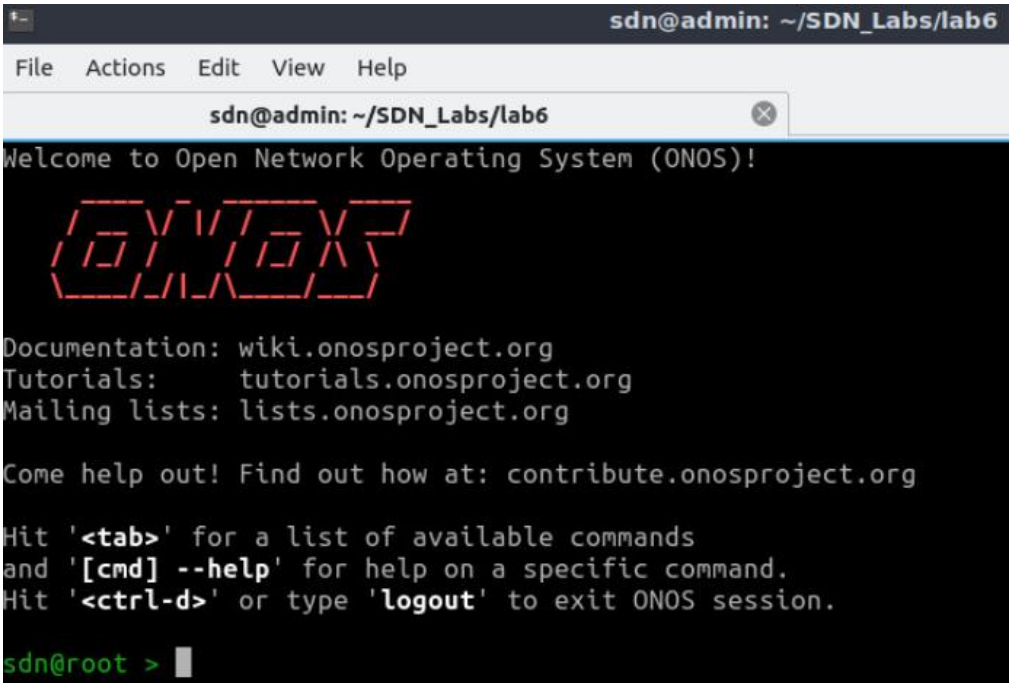

Figure 29. ONOS CLI.

Step 4. In ONOS terminal, issue the following command to activate the OpenFlow application. This application allows ONOS controller to discover the hosts, devices, and links in the current topology.

```
app activate org.onosproject.openflow
```

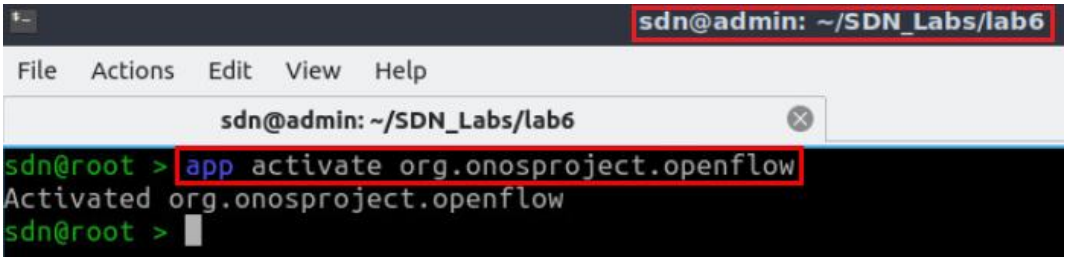

Figure 30. Activating OpenFlow application.

Step 5. After activating the OpenFlow application, you should see a number of OpenFlow messages displayed in Wireshark as shown in the below figure.

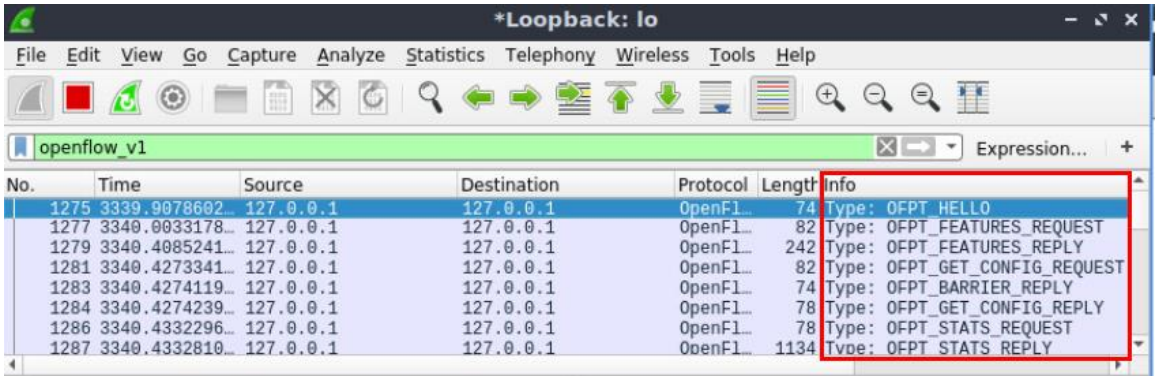

Figure 31. Capturing OpenFlow packets using Wireshark.

Consider Figure 31. The exchanged OpenFlow messages include:

- **Hello message** (from the controller to the switch): the controller sends its version number to the switch.
- **Hello message** (from the switch to the controller): the switch replies with its supported version number.
- **Features request** (from the controller to the switch): the controller asks to see which ports are available.
- **Features reply** (from the switch to the controller): the switch replies with a list of ports, port speeds, and supported tables and actions.
- **Set Config** (from the controller to the switch): the controller asks the switch to send flow expirations.
- **Port status** (from the switch to the controller):

4.3 Capturing PACKET_IN and PACKET_OUT messages

In this section, you will capture more OpenFlow messages exchanged between the controller and the switch after activating ONOS forwarding application. The latter inserts flow entries into the flow table of the switches allowing them to handle IP packets.

Step 1. To enable the forwarding application, type the command shown below. This command activates the forwarding application.

```
app activate org.onosproject.fwd
```

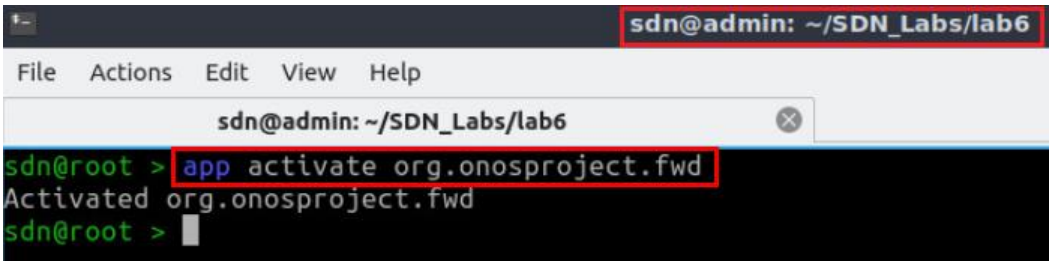

Figure 32. Activating OpenFlow application.

Step 2. On Linux terminal, click on *File>New Tab* to open an additional tab in Linux terminal.

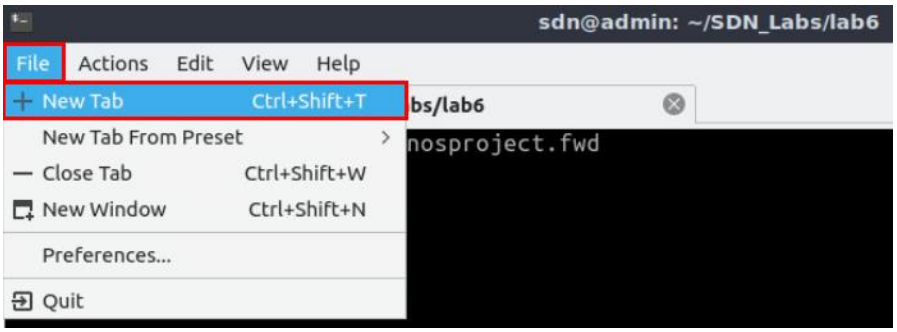

Figure 33. Opening an additional tab.

Step 3. Issue the command below to execute programs with the security privileges of the superuser (root). When prompted for a password, type `password`.

```
sudo su
```

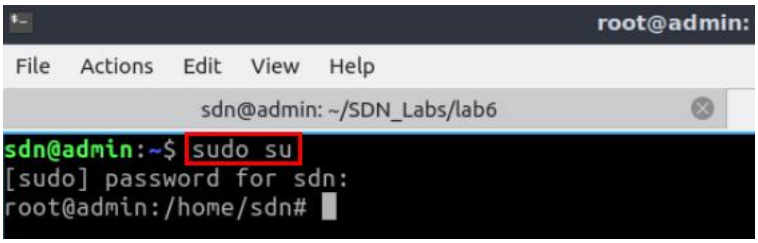

Figure 34. Switching to root mode.

Step 4. Issue the command below to print the flow entries of switch s1.

```
ovs-ofctl dump-flows s1
```

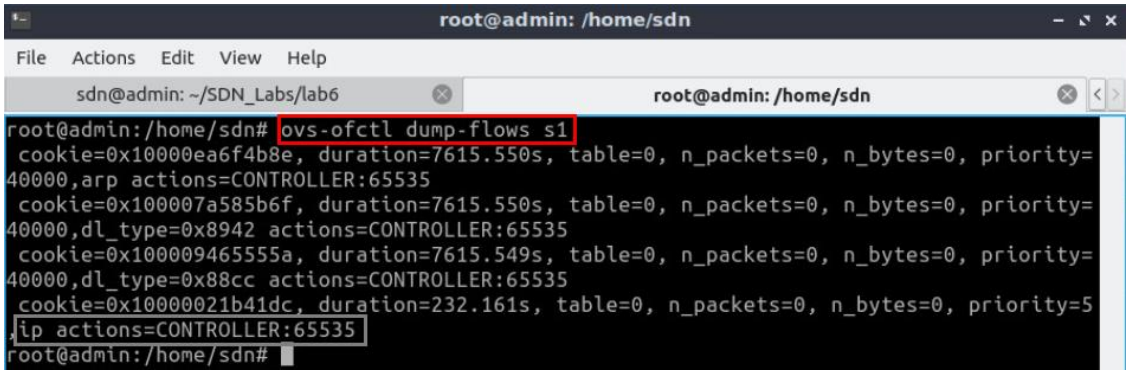

Figure 35. Showing the flow entries of switch s1.

Consider Figure 35. Instead of manually adding entries in switch s1 flow table, ONOS controller inserted the above rules to discover the topology, as well as to manage incoming IP packets by forwarding them to the controller (`ip actions=CONTROLLER:65535`).

Step 5. In host h1, ping host h2 and observe the captured packets in Wireshark. To do this, write the following command.

```
ping 10.0.0.2
```

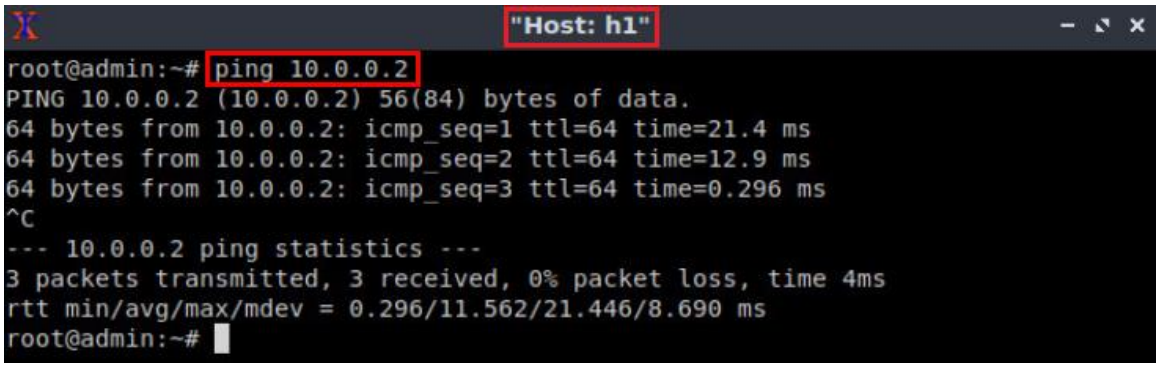

Figure 36. Pinging host h2 from host h1.

Step 6. Go to Wireshark window and inspect the exchanged OpenFlow packets.

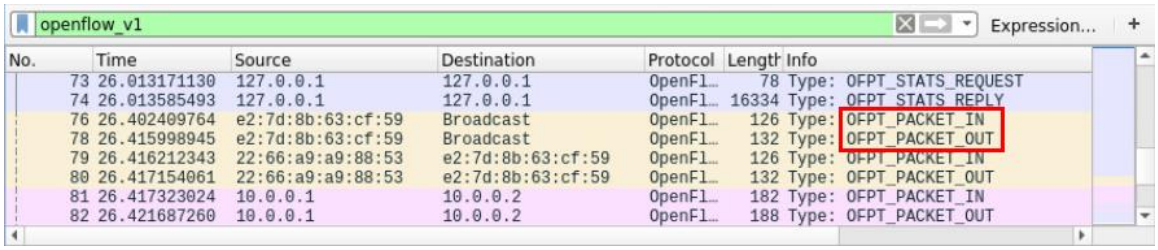

Figure 37. Capturing OpenFlow packets using Wireshark.

Consider Figure 37. During the pinging process from host h1 to host h2, you will notice a number of OpenFlow packets of the following types:

- `PACKET_IN`: the switch sends this message to the controller when a packet is received and didn't match any entry in the switch's flow table.
- `PACKET_OUT`: the controller sends a packet out one or more switch ports.

Other OpenFlow packet types include:

- `OFPT_STATS_REQUEST`: the controller sends this message type to query datapath's current state
- `OFPT_STATS_REPLY`: the switch responds to the request sent by the controller (OFPT_STATS_REQUEST)

Step 7. Stop capturing packets in Wireshark by clicking the red icon.

Figure 38. Stop Capturing Wireshark packets.

This concludes Lab 6. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. Mininet walkthrough, [Online]. Available: http://mininet.org.
2. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "*OpenFlow: enabling innovation in campus networks*." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.
3. P. Goransson, C. Black, T. Culver. *"Software defined networks: a comprehensive approach"*. Morgan Kaufmann, 2016.
4. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, "*ONOS: towards an open, distributed SDN OS*," In Proceedings of the third workshop on Hot topics in software defined networking, pp. 1-6, 2014.

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 7: Routing within an SDN network

Document Version: 08-08-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab is an introduction to integrating Software Defined Networking (SDN) network with legacy network. The lab focuses on SDN-IP application that allows SDN network to connect to legacy network using the standard Border Gateway Protocol (BGP).

Objectives

By the end of this lab, the user should be able to:

1. Understand the concept of SDN network.
2. Understand how ONOS controller interacts with the legacy router.
3. Configure BGP on legacy router.
4. Activate SDN-IP applications.
5. Verify routing within an SDN network.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Starting ONOS controller.
4. Section 4: Integrating SDN and legacy networks.
5. Section 5: Activating Reactive-routing application and verifying the connectivity between the networks.

1 Introduction

Today's Internet is utterly dependent on routing protocols, such as BGP, so without a clean mechanism to integrate an OpenFlow/SDN and legacy/IP networks, the use of OpenFlow will remain restricted to isolated data center deployments. In this lab, you will

understand how to integrate SDN and legacy networks, and how the SDN network translates the BGP information into OpenFlow entries[1].

1.1 SDN network

Routing protocols were essential to respond to rapidly changing network conditions. However, these conditions no longer exist in modern data centers. Thus, the behavior of the routing protocols wreaks temporary havoc inside the data center and hinders the ability to process large data traffic[1].

SDN is a new paradigm that solves the aforementioned problem by creating a centralized approach, rather than a distributed one. The main concept of SDN is to separate the control plane from the data plane in order to maximize the efficiency of the data plane devices. Moving the control software off the device into a centralized server makes it capable of seeing the entire network and making decisions that are optimal given a complete understanding of the situation[1].

Consider Figure 1. The control plane is decoupled from the data plane. The former is moved into a centrally located computer resource and it controls data plane devices by pushing rules into their tables[1].

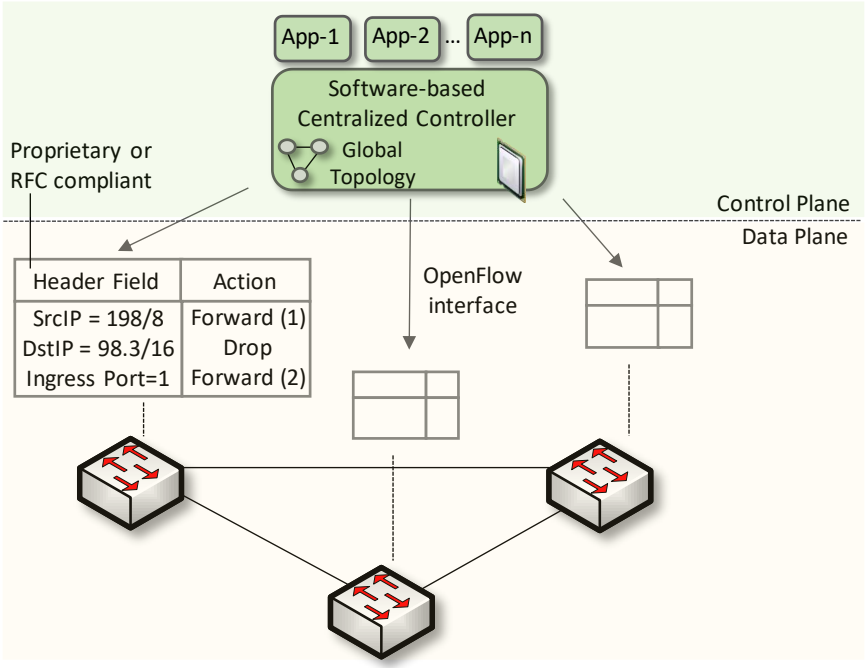

Figure 1. The control plane is embedded in a centralized server and it is decoupled from data plane devices.

1.2 Interworking SDN and legacy networks via SDN-IP application

SDN networks operate in a different manner than legacy networks, which are not going to be entirely replaced in the present time. Thus, one of the obstacles of deploying an SDN network was to integrate it with IP networks[1].

Peering between ASes on the Internet today is universally done with BGP. Therefore, a clear mechanism is needed for an SDN AS to communicate with IP ASes via BGP[1].

SDN-IP is an ONOS application that allows the SDN network to scale and connect to the rest of the Internet[3].

Consider Figure 2. Typically, an SDN network is composed of various OpenFlow switches (switches s1 and s2) connected to the controller (c0). External networks connect through their BGP routers (router r1) to the SDN data plane (switch s1), i.e., the OpenFlow switches, via an EBGP session. To communicate with external IP networks, a BGP speaker (router r2), referred to as IBGP router, must exist within the SDN network and be connected to the data plane. The SDN-IP application runs on top of the ONOS controller and it is connected to the IBGP speaker within the SDN network through an IBGP session[4]. The process that takes place is summarized as follows[4].

1. The network operator expresses the attachment points where both the BGP speakers (router r2) and external routers (router r1) are attached, as well as which BGP speaker is in charge of peering with a certain router. This operation is done using a configuration file that is loaded into ONOS prior to activating the SDN-IP application.
2. Once activated, SDN-IP application parses the configurations and puts in communication the external routers with the BGP speakers. This is translated by ONOS into flows and inserted to the switches. Additionally, the controller opens the port 2000 to communicate with the BGP speakers.
3. Assuming the BGP speakers and the external routers have been configured to peer via BGP, once the flows are installed in the switches, each external router can create EBGP sessions with the BGP speakers.
4. Routes get advertised from the external routers to the BGP speaker via EBGP.
5. Routes propagate among the BGP speakers within the SDN network, as well as to the SDN-IP application via IBGP.
6. The learned routes are advertised by the BGP speakers to other external routers.
7. SDN-IP application translates the learned routes into requests understood by ONOS. The latter translates the requests into OpenFlow entries on the switches.
8. External routers within legacy networks communicate directly through the OpenFlow data plane.

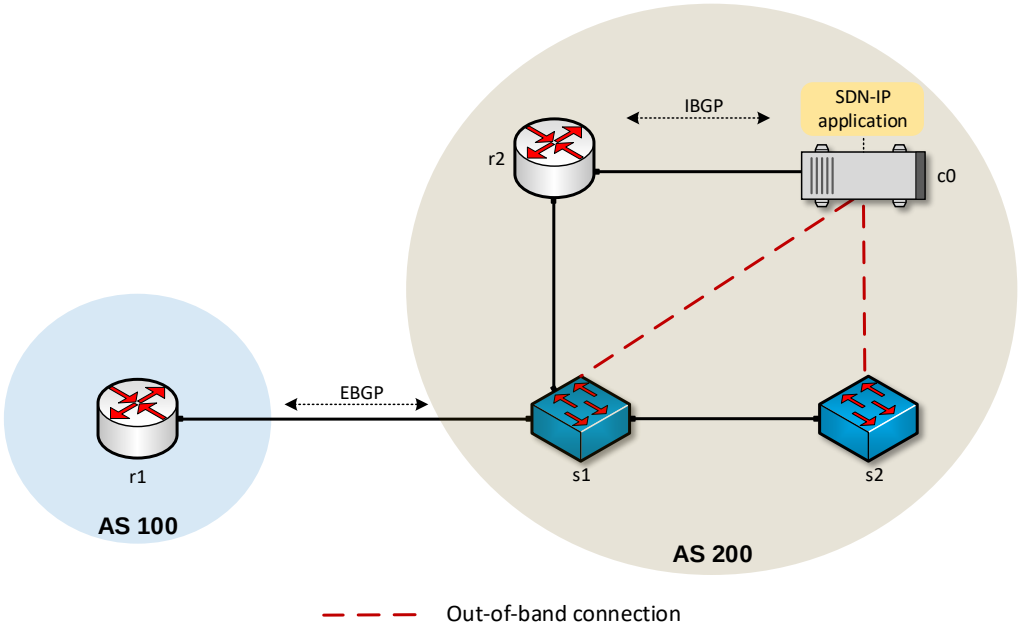

Figure 2. Integrating SDN and IP networks.

2 Lab topology

Consider Figure 3. The topology consists of one SDN network (AS 100). Router r1 is a BGP router within the SDN network. Router R1 is connected to the controller in order to propagate the BGP advertisements to the SDN-IP application running on top of ONOS controller. Router r1 and the controller are connected via the network 10.0.0.0/24.

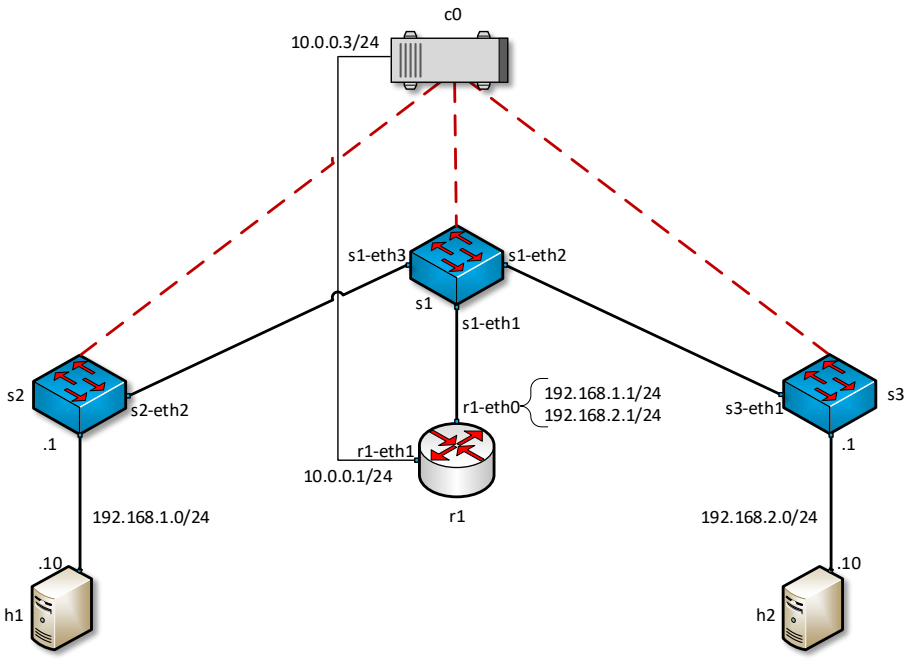

Figure 3. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet	Default gateway
Router r1	r1-eth0	192.168.1.1	/24	N/A
		192.168.2.1	/24	N/A
	r1-eth1	10.0.0.1	/24	N/A
Host h1	h1-eth0	192.168.1.10	/24	192.168.2.1
Host h2	h2-eth0	192.168.2.10	/24	192.168.2.1
c0	n/a	172.17.0.2	/32	n/a
	n/a	10.0.0.3	/24	n/a

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), saving the topologies, and exporting layer 2 models.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

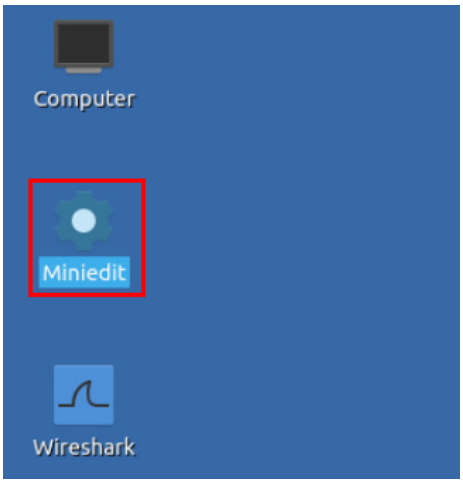

Figure 4. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab7.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab7* and click on *Open*.

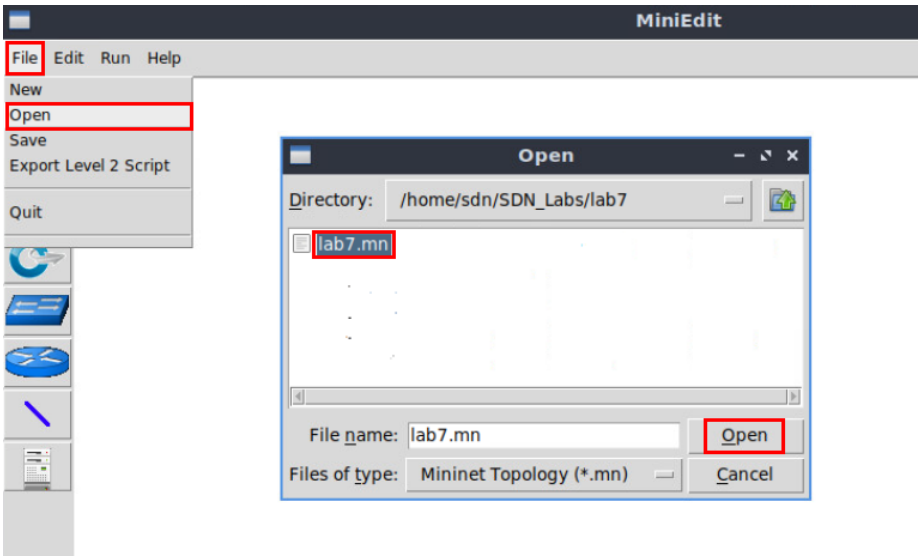

Figure 5. Opening topology.

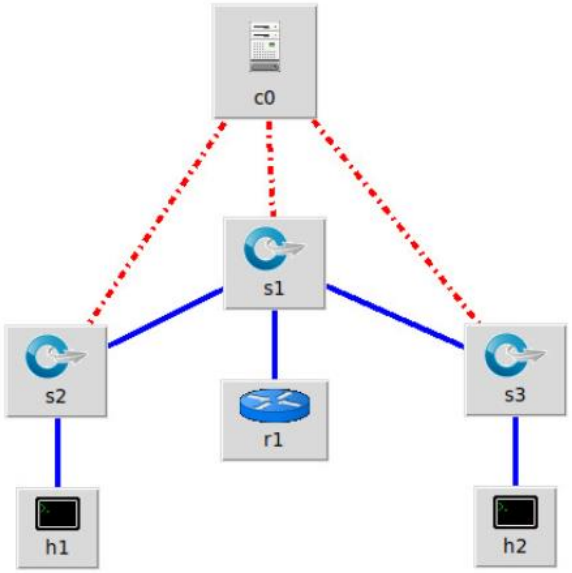

Figure 6. MiniEdit's topology.

2.3 Load the configuration file

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open Linux terminal.

Figure 7. Opening Linux terminal.

Step 2. Click on the Linux terminal and navigate into *SDN_Labs/lab7* directory by issuing the following command. This folder contains a configuration file and the script responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab7
```

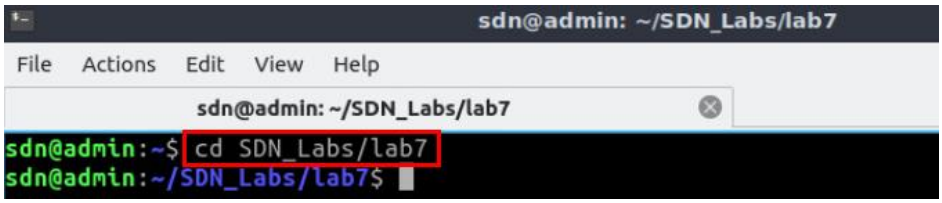

Figure 8. Entering the *SDN_Labs/lab7* directory.

Step 3. To execute the shell script, type the following command. The argument of the program corresponds to the configuration zip file that will be loaded in all the routers in the topology.

```
./config_loader.sh lab7_conf.zip
```

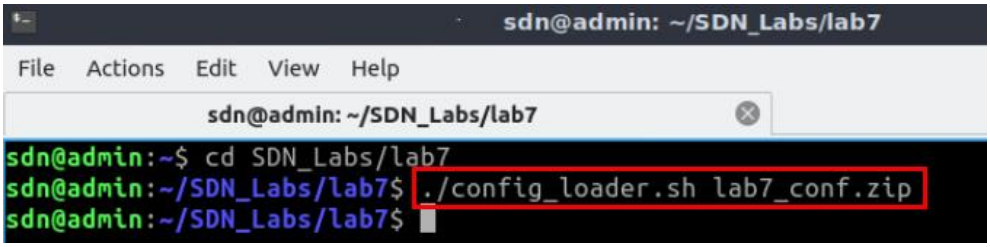

Figure 9. Executing the shell script to load the configuration.

Step 4. Type the following command to exit the Linux terminal.

```
exit
```

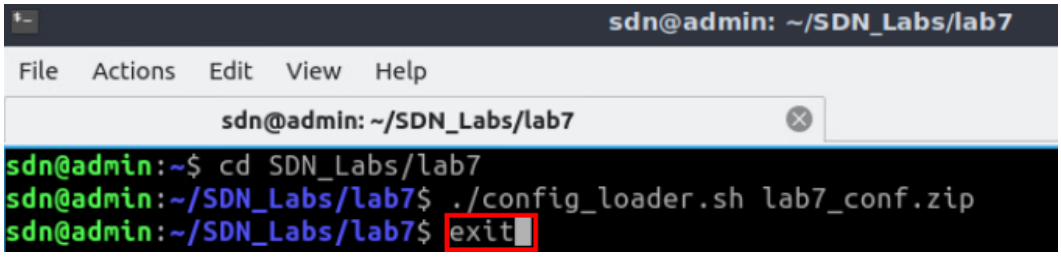

Figure 10. Exiting from the terminal.

2.4 Run the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. At this point host h1 and host h2 interfaces are configured. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

Figure 11. Starting the emulation.

Step 2. Issue the following command on Mininet terminal to display the interface names and connections.

```
links
```

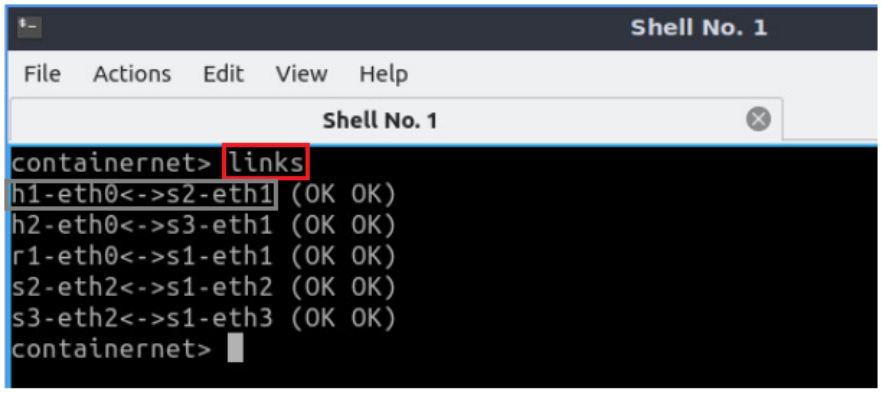

Figure 12. Displaying network interfaces.

In Figure 12, the link displayed within the gray box indicates that interface *eth0* of host h1 connects to interface *eth1* of switch s2 (i.e., *h1-eth0<->s2-eth1*).

2.5 Verify the configuration

You will verify the IP addresses listed in Table 2 and inspect the routing table of router r1.

Step 1. Hold right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on that host.

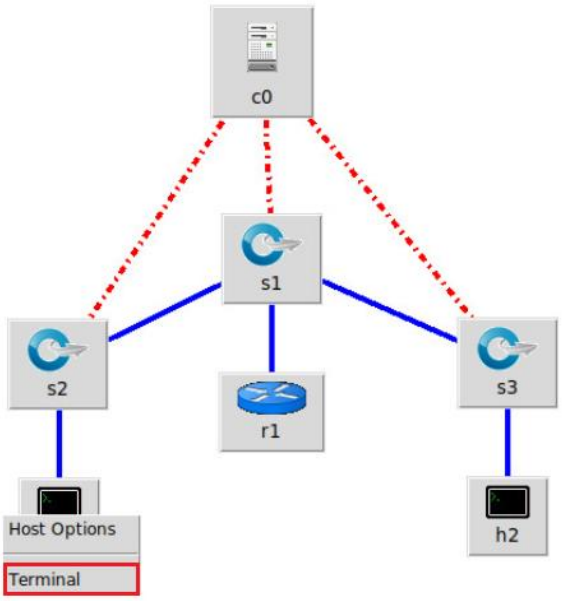

Figure 13. Opening a terminal on host h1.

Step 2. On host h1 terminal, type the command shown below to verify that the IP address was assigned successfully. You will corroborate that host h1 has two interfaces, *h1-eth0* configured with the IP address 192.168.1.10 and the subnet mask 255.255.255.0.

```
ifconfig
```

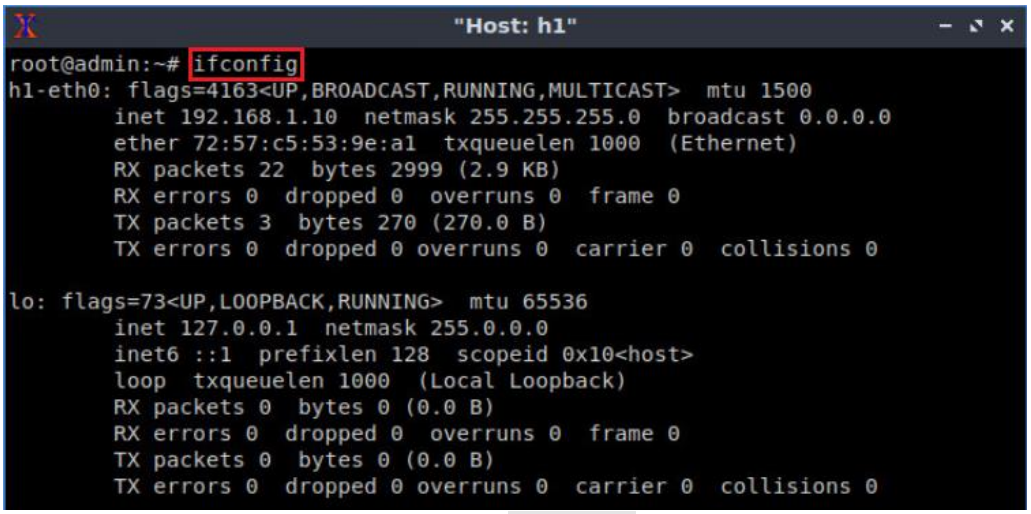

Figure 14. Output of `ifconfig` command.

Step 3. On host h1 terminal, type the command shown below to verify that the default gateway IP address is 192.168.1.1.

```
route
```

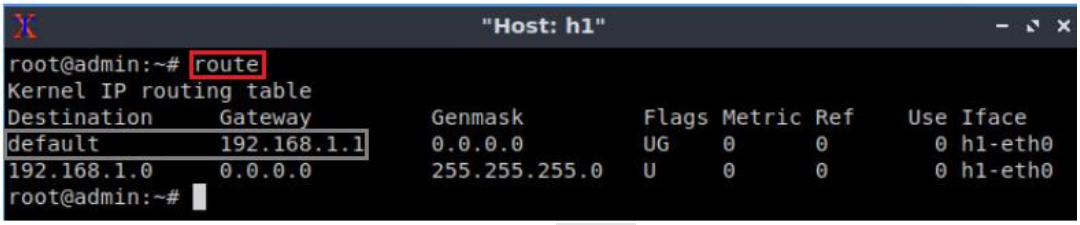

Figure 15. Output of `route` command.

Step 4. In order to verify host h2 default route, proceed similarly by repeating from step 1 to step 3 on host h2 terminal. Similar results should be observed.

Step 5. In order to verify router r1, hold right-click on router r1 and select *Terminal*.

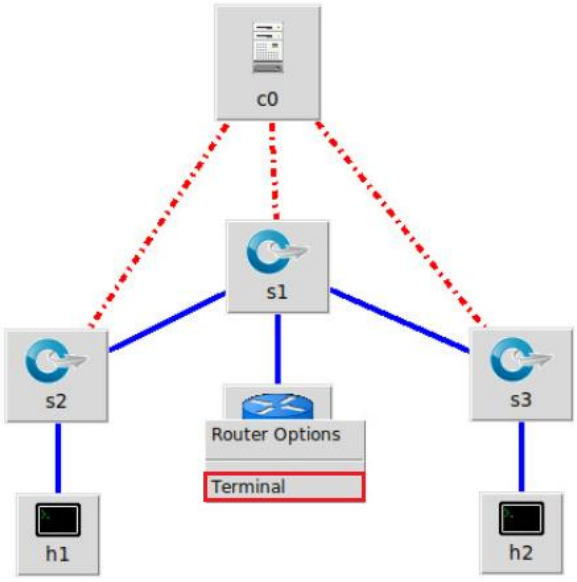

Figure 16. Opening a terminal on router r1.

Step 6. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you do not enable zebra daemon initially. In order to start the zebra, type the following command.

```
zebra
```

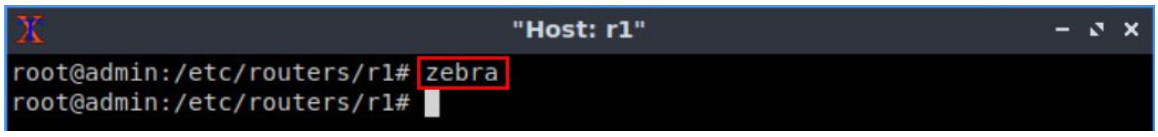

Figure 17. Starting zebra daemon.

Step 7. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command.

```
vtysh
```

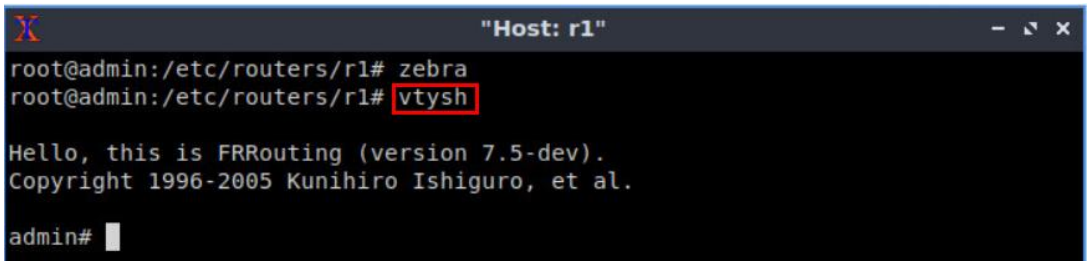

Figure 18. Starting vtysh on router r1.

Step 8. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks.

```
show ip route
```

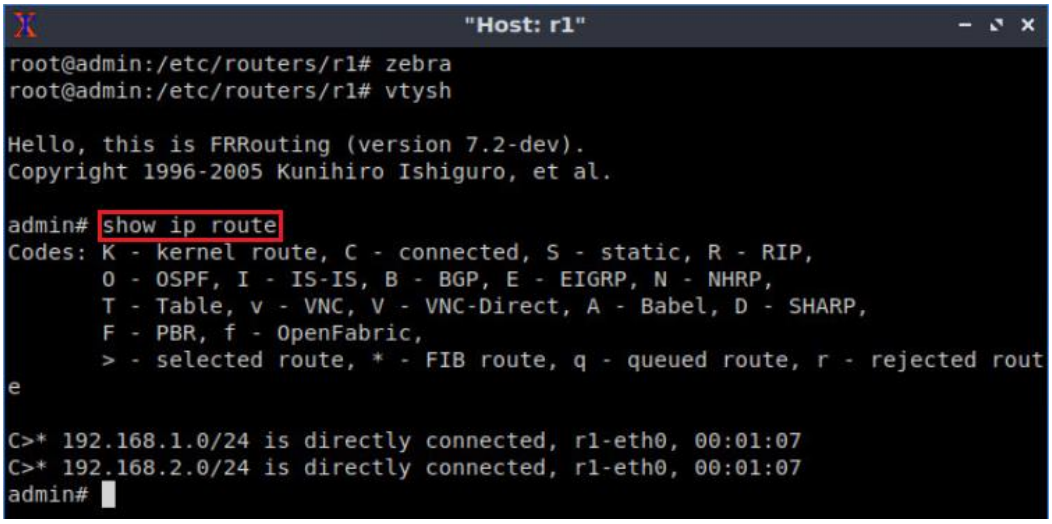

Figure 19.Displaying routing table of router r1.

The output in the figure above shows that the networks 192.168.1.0/24 and 192.168.2.0/24 are directly connected through the interface *r1-eth0*.

3 Starting ONOS controller

In the section, you will start ONOS controller and activate OpenFlow application so that the controller discovers the devices, hosts, and links in the topology.

Step 1. Go to the opened linux terminal.

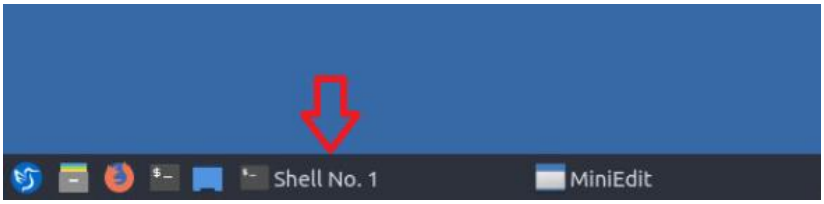

Figure 20. Opening Linux terminal.

Step 2. Click on *File>New Tab* to open an additional tab in Linux terminal. Alternatively, the user may press `Ctrl+Shift+T`.

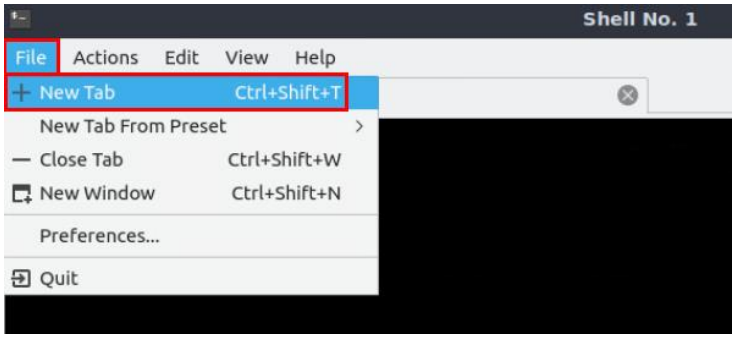

Figure 21. Opening an additional tab.

Step 3. Navigate into *SDN_Labs/lab7* directory by issuing the following command.

```
cd SDN_Labs/lab7
```

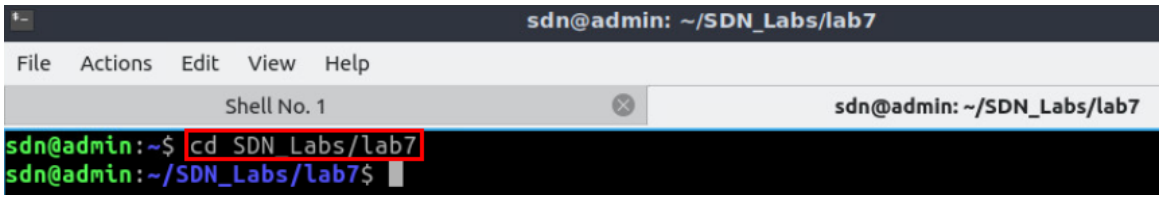

Figure 22. Entering the *SDN_Labs/lab7* directory.

Step 4. Issue the command below to execute programs with the security privileges of the superuser (root). When prompted for a password, type `password`.

```
sudo su
```

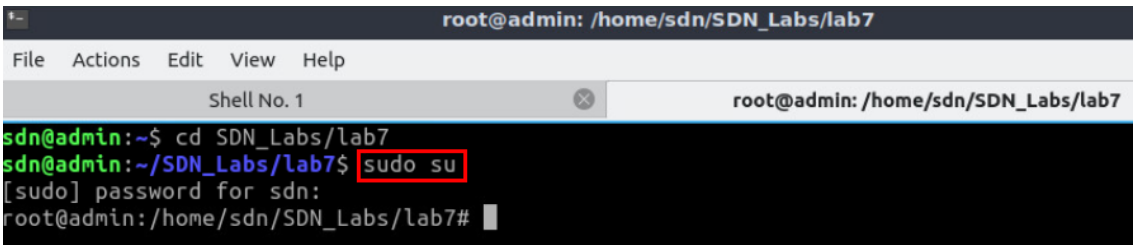

Figure 23. Switching to root mode.

Step 5. A script was written to run ONOS and enter its Command Line Interface (CLI). In order to run the script, issue the following command.

```
./run_onos.sh
```

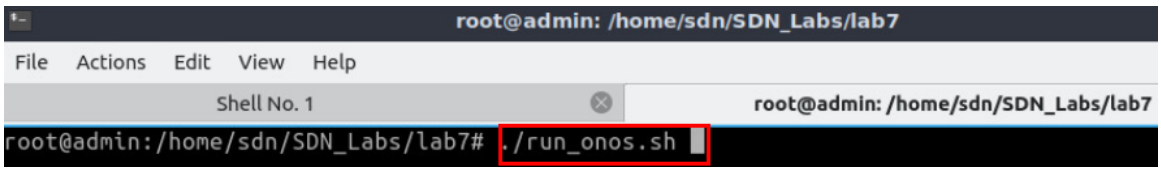

Figure 24. Running ONOS instance.

Once the script finishes executing and ONOS is ready, you will be able to execute commands on ONOS CLI as shown in the figure below. Note that this script may take a couple of minutes.

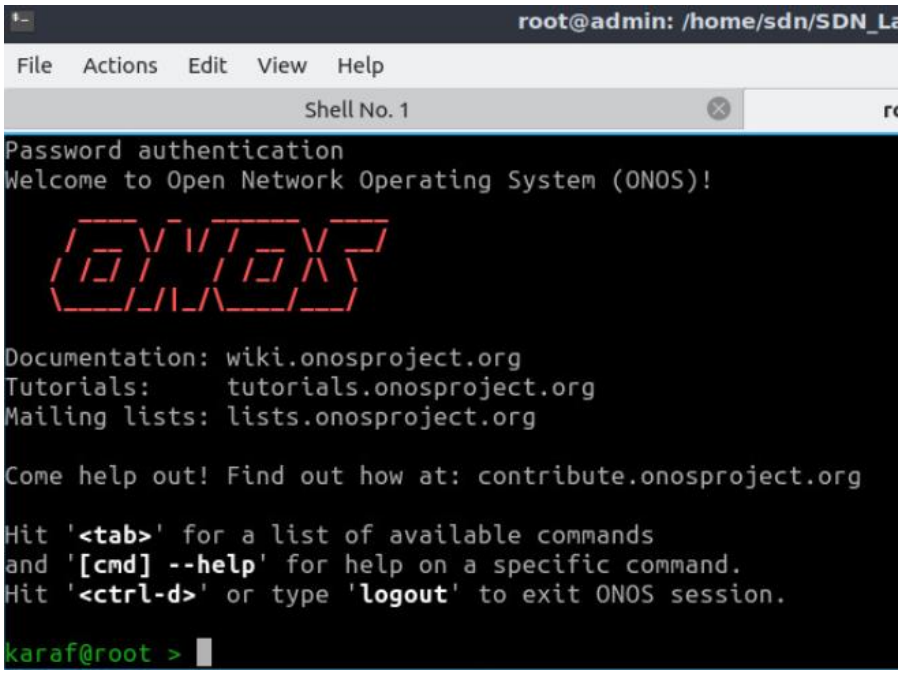

Figure 25. ONOS CLI.

Step 6. In ONOS terminal, issue the following command to activate the OpenFlow application.

```
app activate org.onosproject.openflow
```

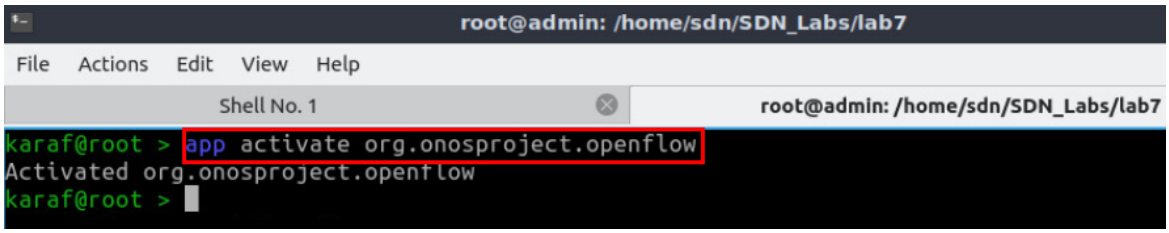

Figure 26. Activating OpenFlow application.

Note that when you activate any ONOS application, you may have to wait few seconds so that the application gives the correct output.

Step 7. To display the list of all currently known devices (OVS switches), type the following command.

```
devices
```

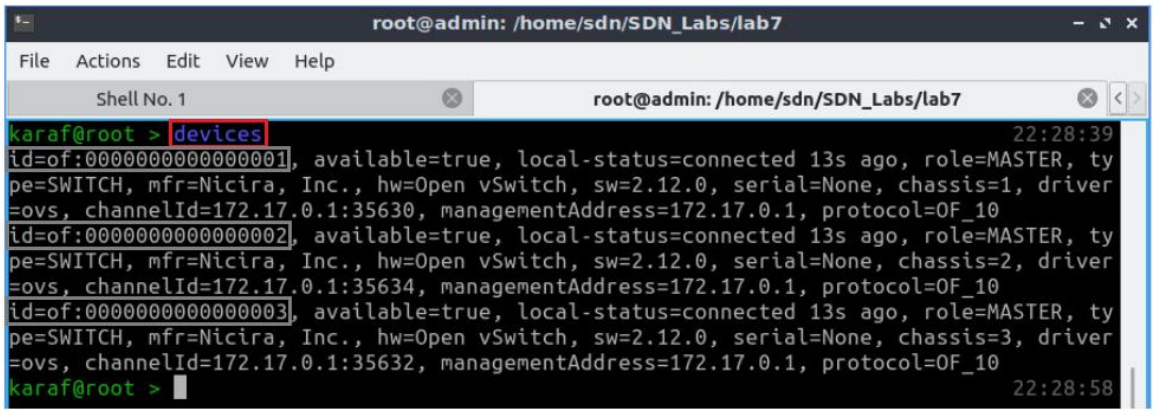

Figure 27. Displaying the current known devices (switches).

Step 8. To display the list of all currently known links, type the following command.

```
links
```

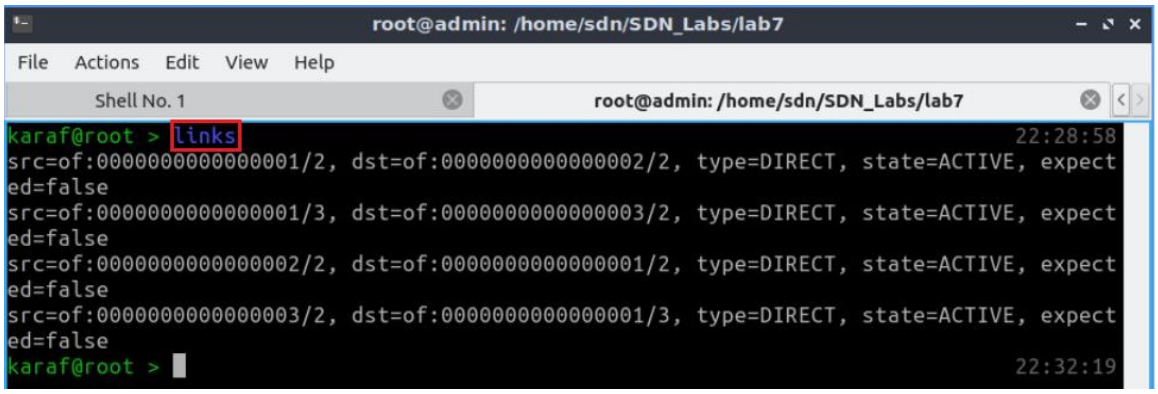

Figure 28. Displaying the current known links.

Step 9. To display the list of all currently known hosts, type the following command.

```
hosts
```

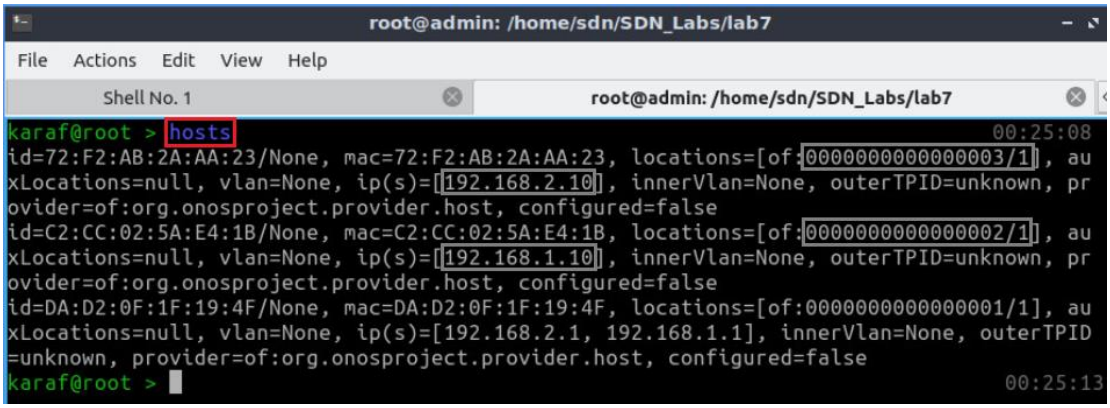

Figure 29. Displaying the current known links.

Consider Figure 29. ONOS recognizes router display the interfaces of the OpenFlow switches they are connected to (192.168.1.10, 192.168.2.10). Note that you might have to wait until ONOS discovers the two hosts in case they don't appear immediately.

4 Integrating SDN and legacy networks

In the previous sections, you configured the legacy devices, as well as started ONOS and its OpenFlow application to discover the topology. In this section, you will first execute a script that connects the IBGP speaker (router r1) with ONOS controller, thus, the two entities can communicate. Furthermore, you will activate ONOS SDN-IP application in order to interconnect the SDN network with the legacy network.

4.1 Connecting the IBGP speaker (router r1) with ONOS controller

In this section, you will execute a script that creates a peer-to-peer link connecting router r1 with ONOS.

Step 1. Go to Mininet tab in the Linux terminal.

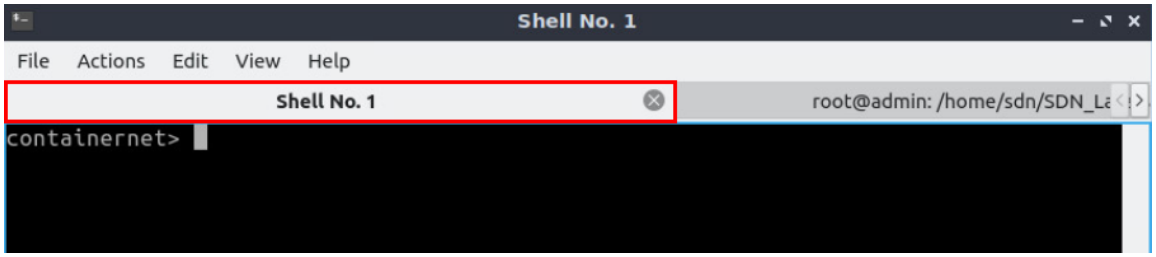

Figure 30. Opening Mininet tab.

Step 2. In order to create a point-to-point network between the IBGP speaker (router r1) and ONOS, a script was written to facilitate the process. In order to execute the script, type the following command.

```
source ./SDN_Labs/lab7/create_link.sh
```

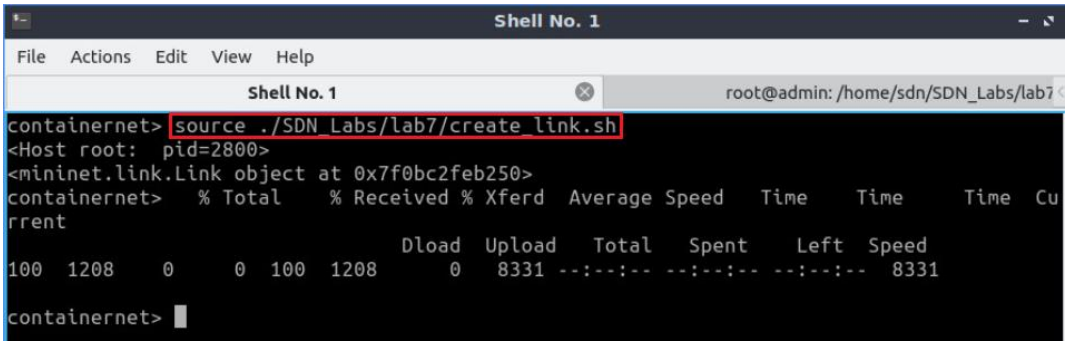

Figure 31. Creating a point-to-point network (link) between the IBGP speaker and ONOS controller.

Consider Figure 31. The script creates a point-to-point network between router r1 and ONOS. The network address of the point-to-point network is 10.0.0.0/24. Router r1 is assigned the IP address 10.0.0.1/24, whereas ONOS controller is assigned 10.0.0.3/24.

Step 3. In router r1 terminal, type the following command to exit the vtysh session.

```
exit
```

Figure 32. Creating a network between the IBGP speaker and ONOS controller.

Step 4. Now that router r1 is connected to ONOS controller, a new interface must appear in it. In order to verify the connected interface, type the following command.

```
ifconfig
```

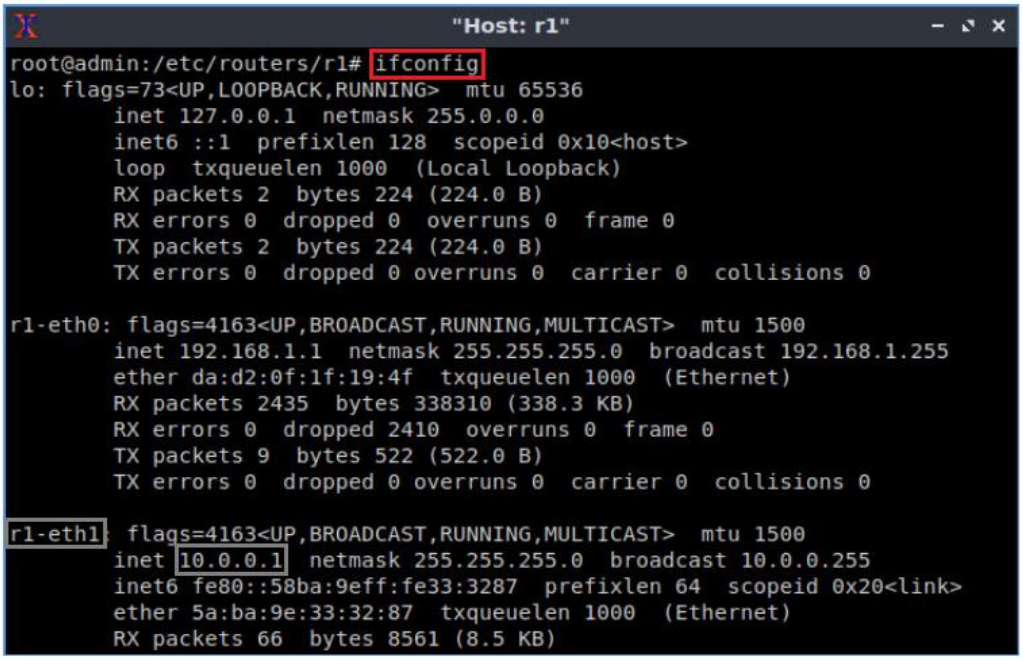

Figure 33. Listing router r1's interfaces.

Consider Figure 33. Interface *r1-eth1* is added after creating a point-to-point network between router r1 and ONOS controller. Furthermore, the interface is associated with the IP address 10.0.0.1.

Step 5. In the default directory */home/sdn/SDN_Labs /lab7*, open the file *network-cfg.json.* This JSON file informs ONOS of the existence of such devices when it is pushed.

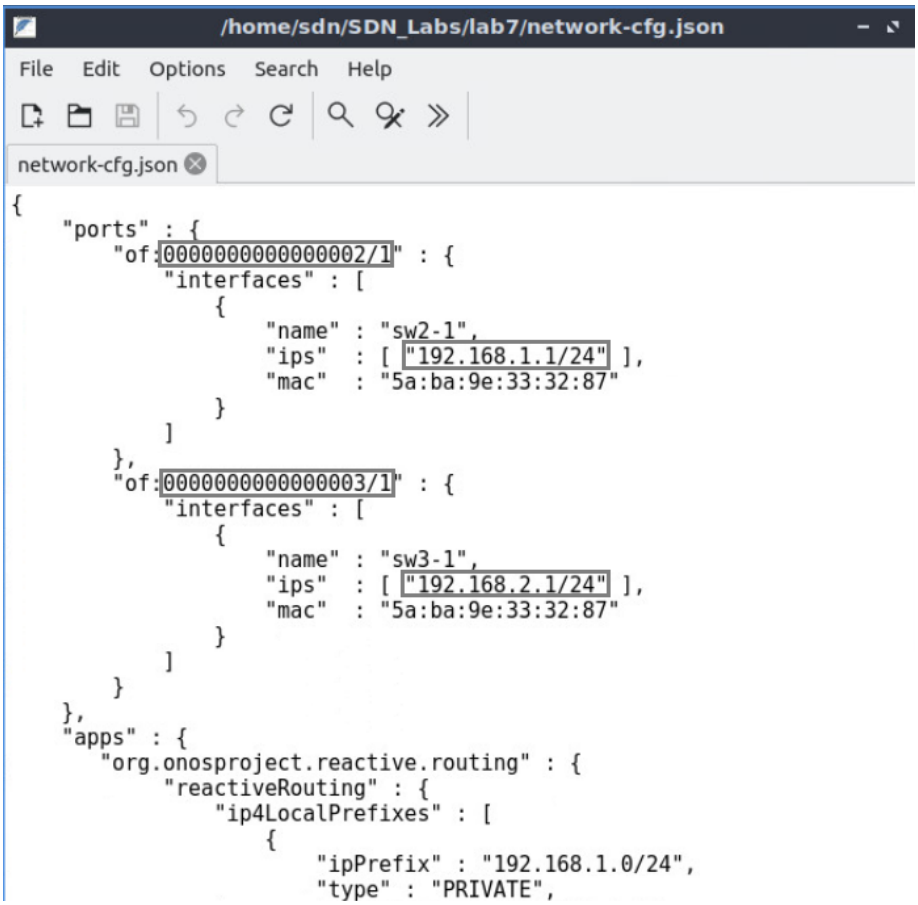

Figure 34. Opening network-cfg.json file.

Consider Figure 34. ONOS controller is now aware of two networks. IP addresses 192.168.1.1/24 and 192.168.2.1/24 belong to switches s2 and s3, respectively. MAC address, *5a:ba:9e:33:32:87* is the virtual gateway address for ONOS controller.

4.2 Configuring BGP on router r1

In this section, you will configure BGP on router r1 to peer with the ONOS controller.

Step 1. Type the following command on router r1 terminal to start BGP routing protocol.

```
bgpd
```

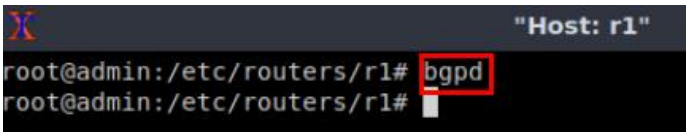

Figure 35. Starting BGP daemon.

Step 2. In order to enter to router r1 terminal, type the following command.

```
vtysh
```

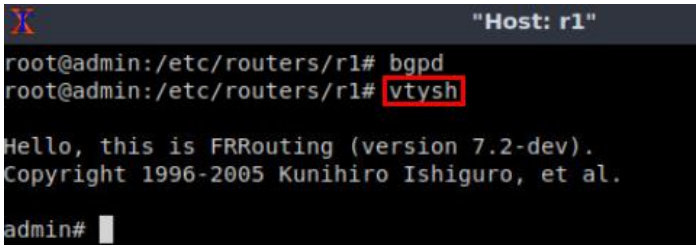

Figure 36. Starting vtysh on router r1.

Step 3. To enable router r1 configuration mode, issue the following command.

```
configure terminal
```

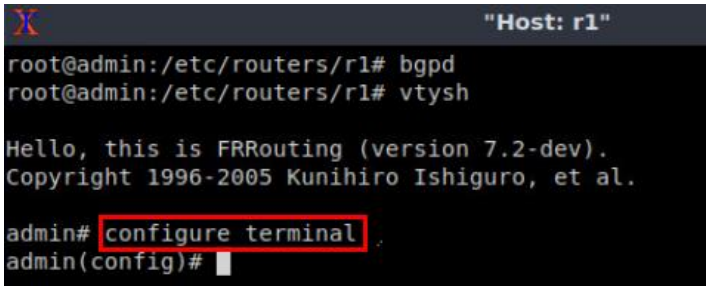

Figure 37. Enabling configuration mode on router r1.

Step 4. The ASN assigned for router r1 is 100. In order to configure BGP, type the following command.

```
router bgp 100
```

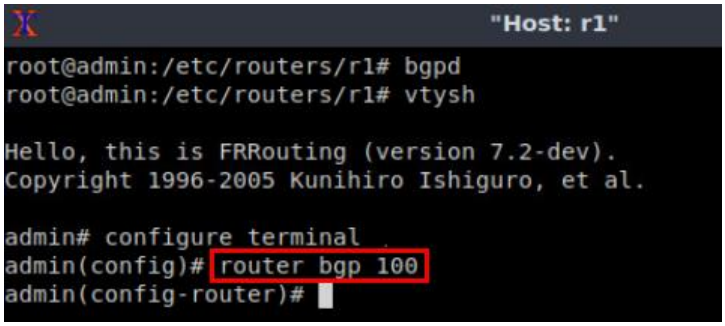

Figure 38. Configuring BGP on router r1.

Step 5. Router r1 and ONOS controller are connected using a point-to-point network (10.0.0.0/24). The IP address assigned to the controller is 10.0.0.3. As router r1 is the IBGP speaker within the SDN network, it must establish a BGP peering relationship with the controller in its network (AS 100). In order to establish BGP peering relationship with the controller, type the following command.

```
neighbor 10.0.0.3 remote-as 100
```

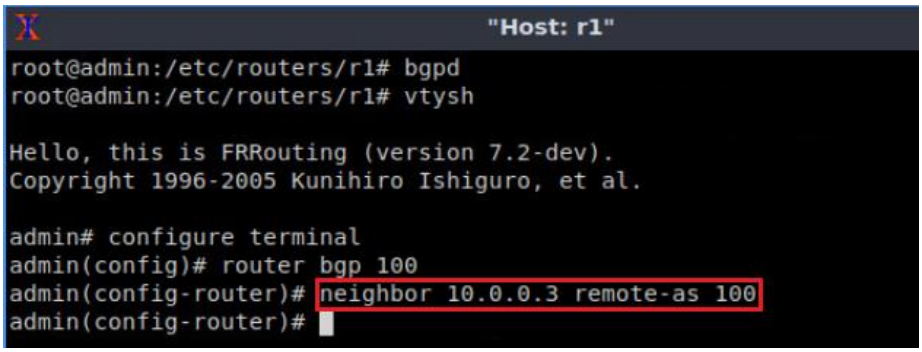

Figure 39. Assigning BGP neighbor to router r1.

Step 6. By default, ONOS listens to TCP port number 2000 for incoming BGP connections, which is not the default BGP port number 179. In order to specify the port for incoming BGP messages from ONOS, write the following command.

```
neighbor 10.0.0.3 port 2000
```

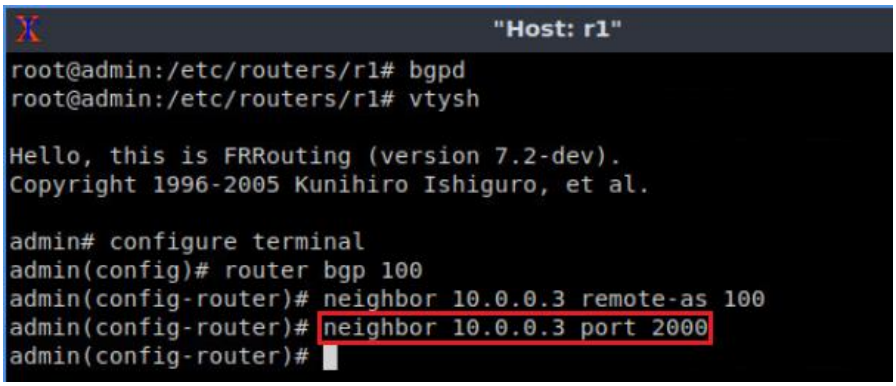

Figure 40. Changing the Listening port for BGP connections.

Step 7. Type the following command to exit from the configuration mode.

```
end
```

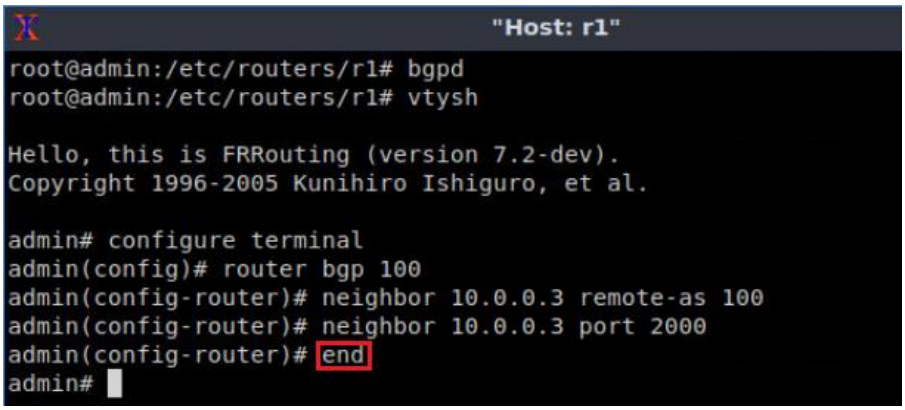

Figure 41. Exiting from configuration mode.

Step 8. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 contains a directly connected network, 10.0.0.0/24.

```
show ip route
```

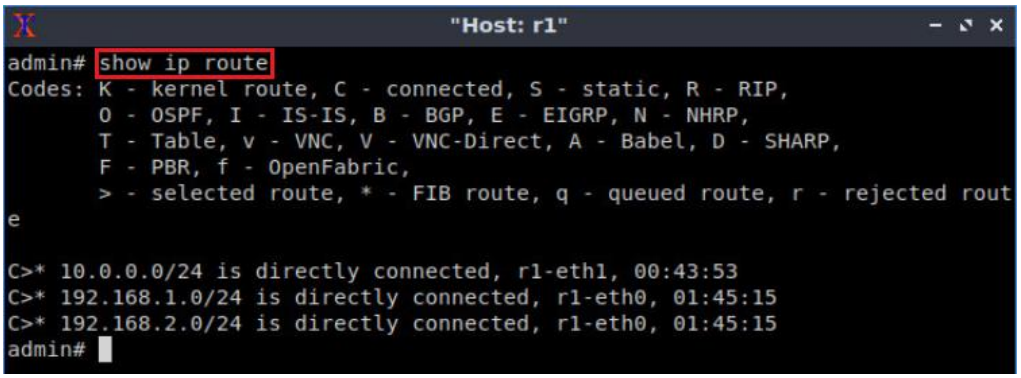

Figure 42. Displaying the routing table of router r1.

4.3 Activating SDN-IP application

In this section, you will activate the SDN-IP application and other dependencies (applications) that will interconnect the SDN network with the legacy network.

Step 1. Go to ONOS terminal.

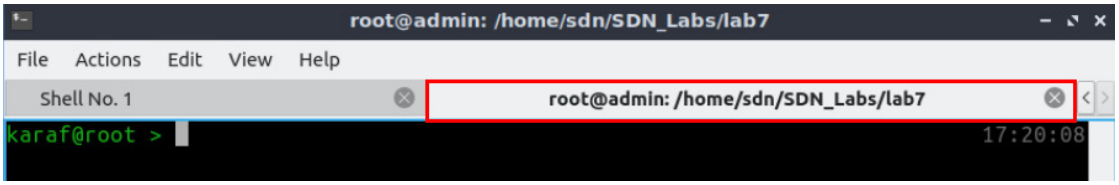

Figure 43. Opening ONOS terminal.

Step 2. Before activating the SDN-IP application you must start the *org.onosproject.config* application. The latter is an application for the network configuration. In order to activate the config application, type the following command.

```
app activate org.onosproject.config
```

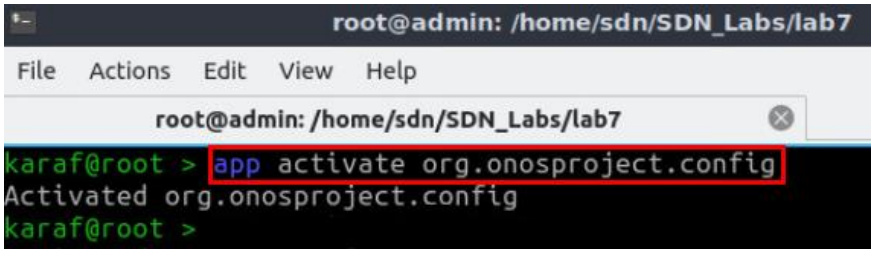

Figure 44. Activating ONOS config application.

Step 3. SDN-IP application has an additional application dependency that it relies on to ensure Address Resolution Protocol (ARP) requests are resolved properly. This is the *org.onosproject.proxyarp* application that responds to ARP requests on behalf of hosts and external routers.

```
app activate org.onosproject.proxyarp
```

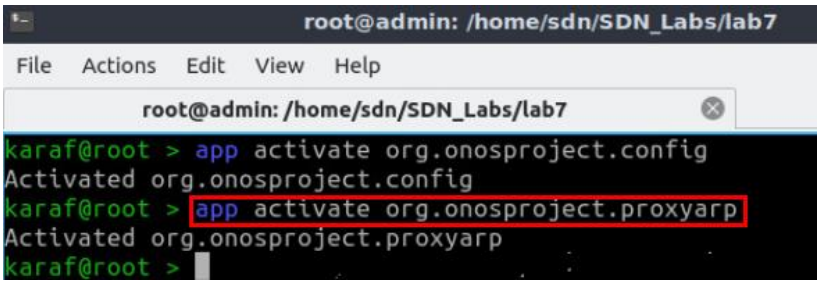

Figure 45. Activating ONOS proxyarp application.

Step 4. Once the dependencies are started, the SDN-IP application can be activated. In order to do that, type the following command.

```
app activate org.onosproject.sdnip
```

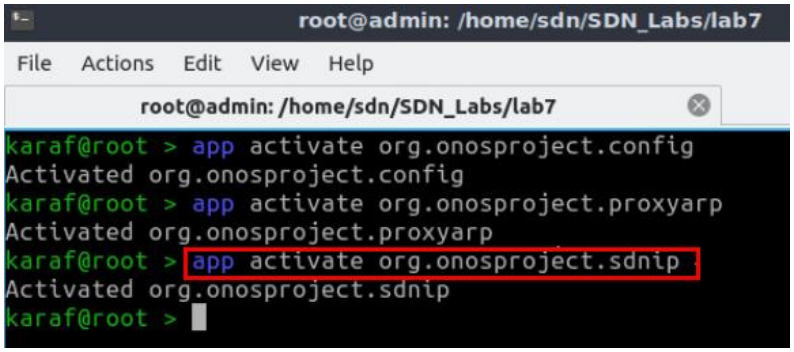

Figure 46. Activating ONOS SDN-IP application.

After activating the above applications, you might have to wait few minutes until the applications discover the topology, and exchange information in order to get correct results.

Step 5. In ONOS terminal, type the following command to show the IBGP neighbors that have connected to SDN-IP application.

```
bgp-neighbors
```

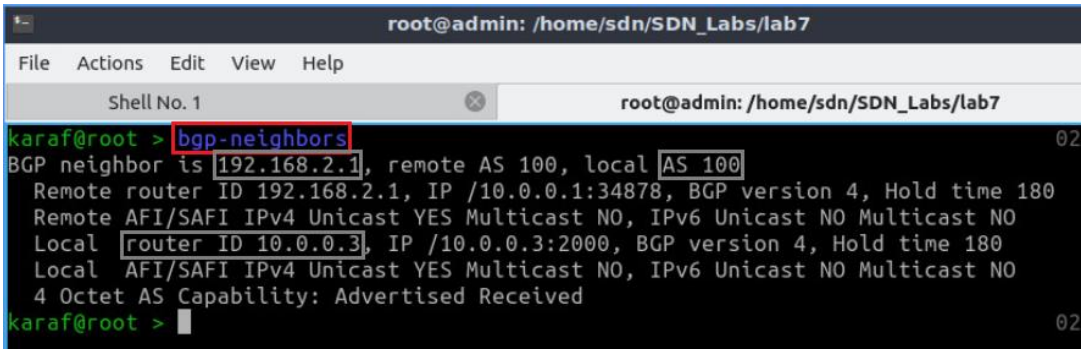

Figure 47. Viewing IBGP neighbors within the SDN network.

Consider Figure 47. The neighbor 192.168.2.1 corresponds to router r1 in AS 100. This is the internal BGP speaker in the SDN network. The local router ID that the SDN-IP application uses is 10.0.0.3.

Step 6. Test the connectivity between host h1 and host h2 using the `ping` command. On host h1, type the command specified below. To stop the test, press `Ctrl+c`. The hosts are unreachable.

```
ping 192.168.2.10
```

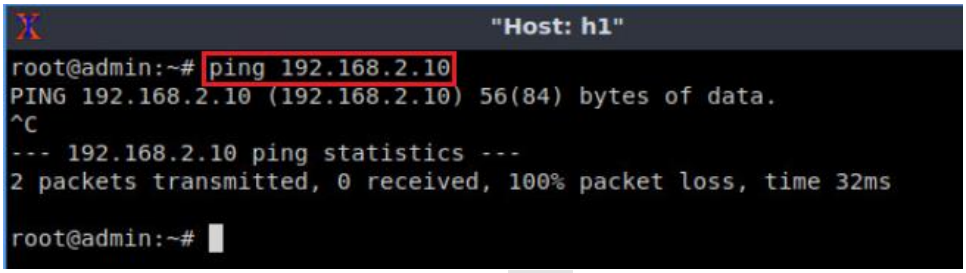

Figure 48. Output of `ping` command.

Step 7. Type the following command to verify the flows on the switch s2.

```
flows added of:0000000000000002
```

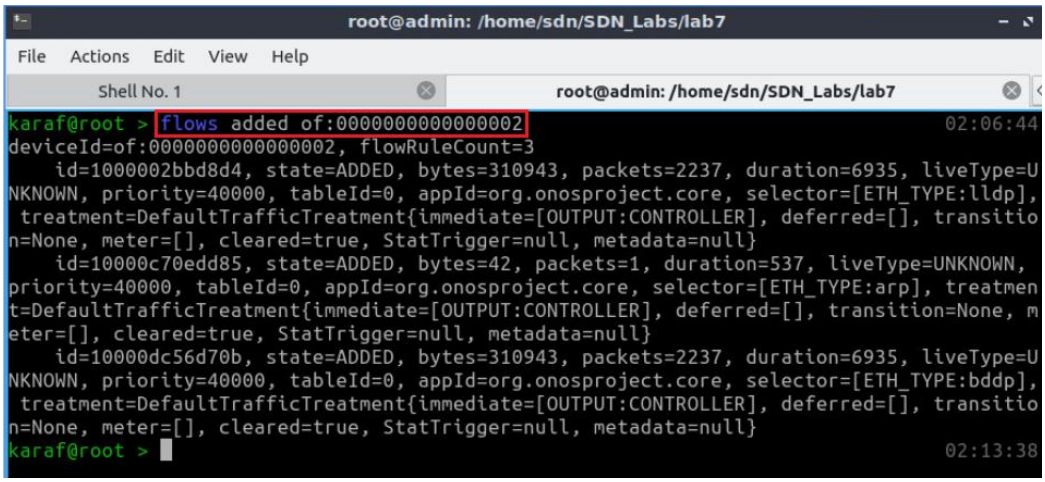

Figure 49. Verifying flows on s2.

Consider Figure 49. When ONOS receives an IPv4 or IPv6 packet-in, it will reactively compute and install the routing path for the traffic. The hosts are unreachable at this point since ipv4 flow is not present in the flow table.

5 Activating Reactive-routing application and verifying the connectivity between the networks

In this section, you will activate reactive-routing application. Additionally, you will verify the connectivity between two hosts.

Step 1. In order to activate the reactive-routing application, type the following command:

```
app activate org.onosproject.reactive-routing
```

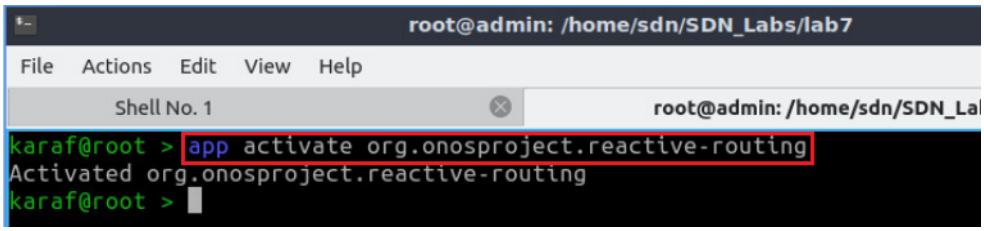

Figure 50. Activating ONOS SDN-IP application.

Step 2. Test the connectivity between host h1 and host h2 using the `ping` command. On host h1, type the command specified below. To stop the test, press `Ctrl+c`. The figure below shows a successful connectivity test.

```
ping 192.168.2.10
```

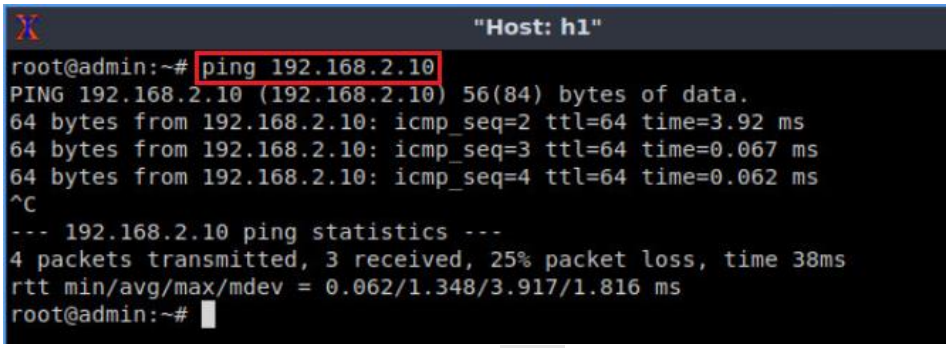

Figure 51. Output of `ping` command.

Step 3. Type the following command to verify the flows on the switch s2.

```
flows added of:0000000000000002
```

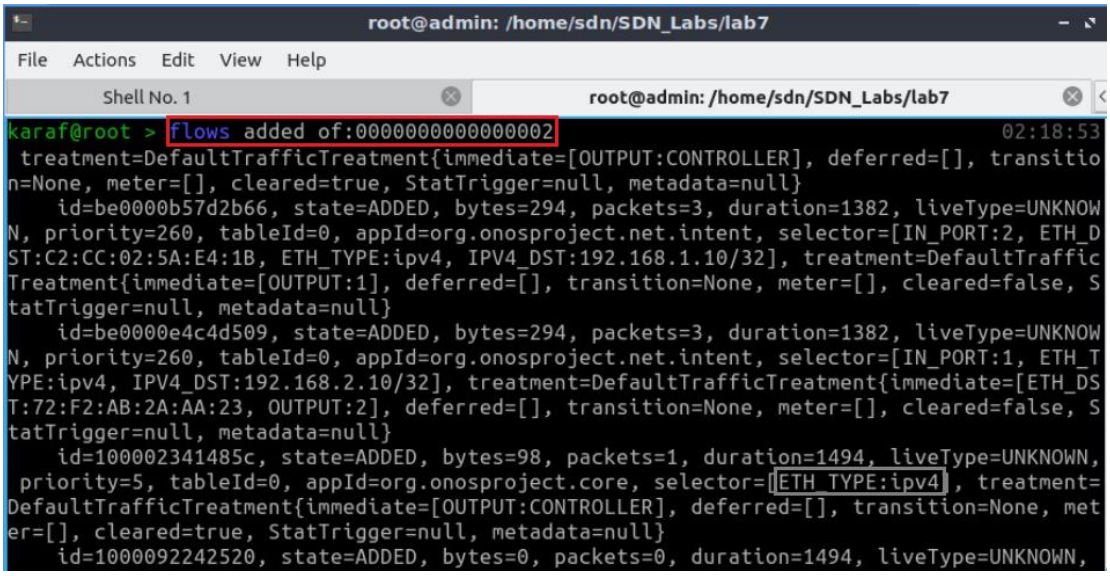

Figure 52. Verifying flows on s2.

Consider Figure 52. You will notice *ipv4* flow is added in the flow table.

Step 4. In the flow table, you can verify the reactively installed routing path for the traffic. If there any incoming flow on port 2, the destination address will be 192.168.1.10. Similarly, any incoming flow on port 1 will be delivered to the destination, 192.168.2.10.

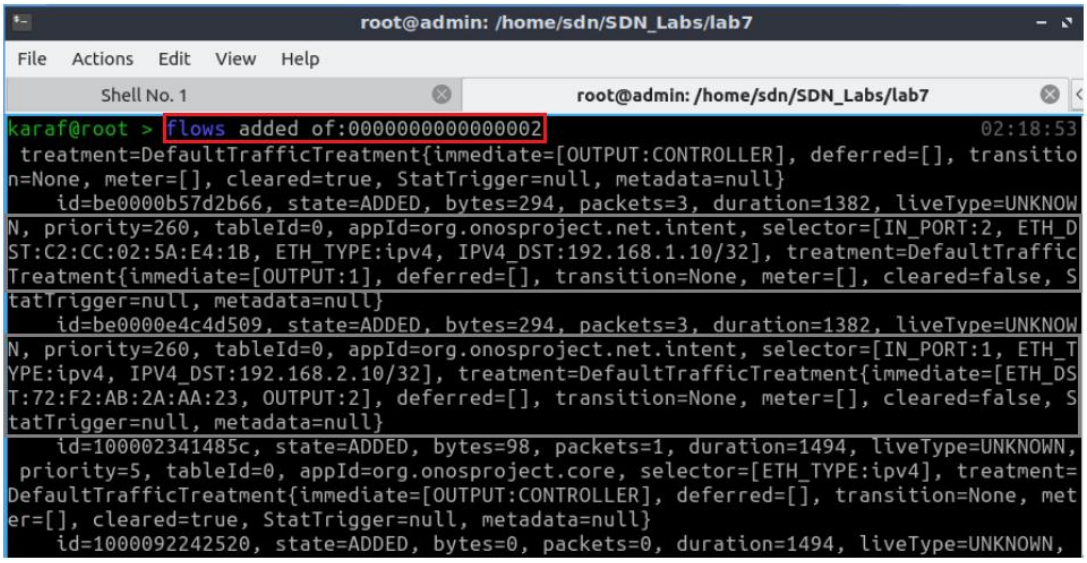

Figure 53. Verifying flows on s2.

Since the routing paths are installed reactively, each traffic only appears in the flow table only for a certain period of time (approx. 60 sec)

This concludes Lab 7. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. A. Tanenbaum, D. Wetherall, "*Computer networks*", 5th Edition, Pearson, 2012.
2. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach*". Morgan Kaufmann, 2016.
3. P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.C. Wang, J. Bi. "*Seamless interworking of SDN and IP*". In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 475-476, 2013.
4. ONOS project, "*SDN-IP*", [Online]. Available: https://wiki.onosproject.org/display/ONOS/SDN-IP.

UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 8: Interconnection between legacy networks and SDN networks

Document Version: 31-05-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab is an introduction to integrating Software Defined Networking (SDN) networks with legacy networks. The focus in this lab is to understand SDN-IP, an application that comes with ONOS controller and allows the interconnection between legacy and SDN networks. The SDN-IP application allows the SDN network to understand the exchanged Border Gateway Protocol (BGP) information and communicate with other legacy networks through this protocol.

Objectives

By the end of this lab, the user will:

1. Understand how legacy networks operate.
2. Understand how SDN networks operate.
3. Configure BGP on legacy routers.
4. Integrate SDN and legacy networks through the SDN-IP application.
5. Verify the connectivity between the SDN and legacy networks.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Configuring BGP within legacy networks.
4. Section 4: Starting ONOS controller.
5. Section 5: Integrating SDN and legacy networks.
6. Section 6: Verifying the connectivity between the networks.

1 Introduction

Today's Internet is utterly dependent on routing protocols, such as BGP, so without a clean mechanism to integrate an OpenFlow/SDN and legacy/IP networks, the use of OpenFlow will remain restricted to isolated data center deployments. In this lab, you will understand how to integrate SDN and legacy networks, and how the SDN network translates the BGP information into OpenFlow entries[1].

1.1 Legacy networks

The Internet can be viewed as a collection of networks or Autonomous Systems (ASes) that are interconnected. An AS refers to a group of connected networks under the control of a single administrative entity or domain. Traditional networks depend on routing protocols to interconnect and share routing information. Such protocols are also referred to as control protocols and they are run by each device in the network[1].

BGP is the standard exterior gateway protocol designed to exchange routing and reachability information among ASes on the Internet. BGP is relevant to network administrators of large organizations which connect to one or more Internet Service Providers (ISPs), as well as to ISPs who connect to other network providers[1].

Two routers that establish a BGP connection are referred to as BGP peers or neighbors. BGP sessions run over Transmission Control Protocol (TCP). If a BGP session is established between two neighbors in different ASes, the session is referred to as an External BGP (EBGP) session. If the session is established between two neighbors in the same AS, the session is referred to as Internal BGP (IBGP) session[1].

Figure 1 shows two legacy networks, each in an AS. Each router runs its internal local algorithm (routing protocol) to communicate with other peers. The routing protocol used between ASes is BGP.

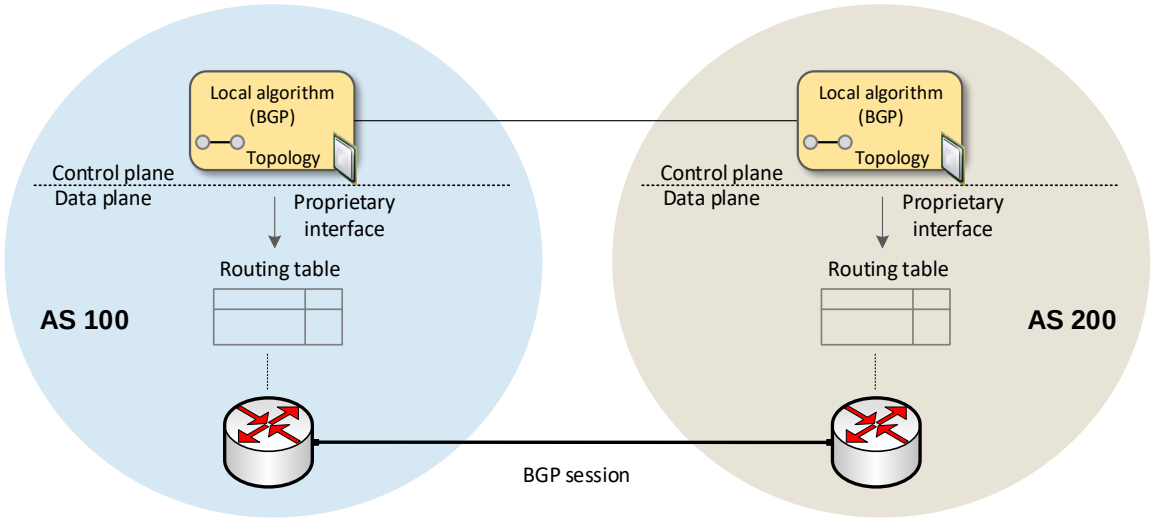

Figure 1. Legacy networks use BGP to share routing information between ASes.

1.2 SDN network

Routing protocols were essential to respond to rapidly changing network conditions. However, these conditions no longer exist in modern data centers. Thus, the behavior of the routing protocols wreaks temporary havoc inside the data center and hinders the ability to process large data traffic[1].

SDN is a new paradigm that solves the aforementioned problem by creating a centralized approach, rather than a distributed one. The main concept of SDN is to separate the control plane from the data plane in order to maximize the efficiency of the data plane devices. Moving the control software off the device into a centralized server makes it capable of seeing the entire network and making decisions that are optimal given a complete understanding of the situation[1].

Consider Figure 2. The control plane is decoupled from the data plane. The former is moved into a centrally located computer resource and it controls data plane devices by pushing rules into their tables[1].

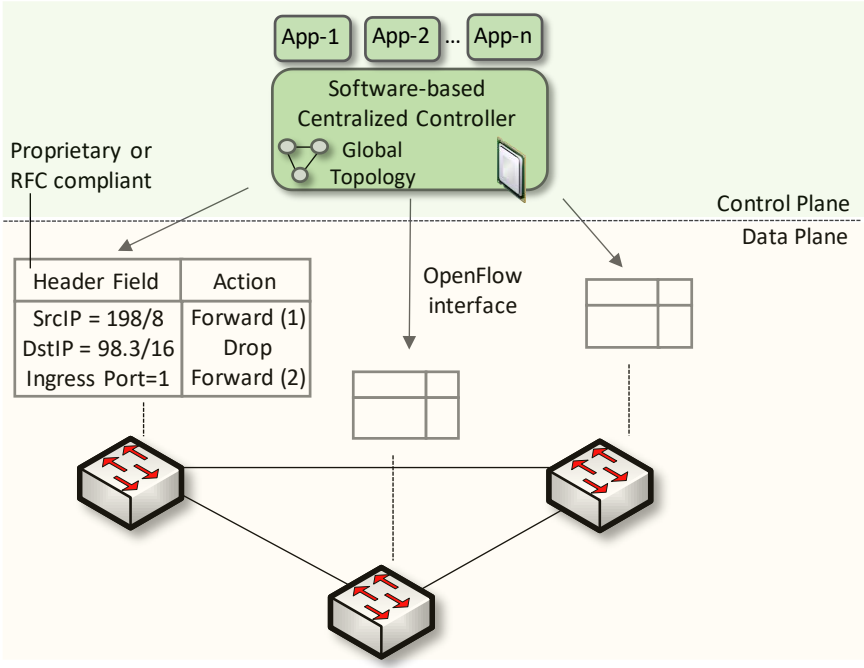

Figure 2. The control plane is embedded in a centralized server and it is decoupled from data plane devices.

1.3 Interworking SDN and legacy networks via SDN-IP application

SDN networks operate in a different manner than legacy networks, which are not going to be entirely replaced in the present time. Thus, one of the obstacles of deploying an SDN network was to integrate it with IP networks[1].

Peering between ASes on the Internet today is universally done with BGP. Therefore, a clear mechanism is needed for an SDN AS to communicate with IP ASes via BGP[1].

SDN-IP is an ONOS application that allows the SDN network to scale and connect to the rest of the Internet[3].

Consider Figure 3. Typically, an SDN network is composed of various OpenFlow switches (switches s1 and s2) connected to the controller (c0). External networks connect through their BGP routers (router r1) to the SDN data plane (switch s1), i.e., the OpenFlow switches, via an EBGP session. To communicate with external IP networks, a BGP speaker (router r2), referred to as IBGP router, must exist within the SDN network and be connected to the data plane. The SDN-IP application runs on top of the ONOS controller and it is connected to the IBGP speaker within the SDN network through an IBGP session[4]. The process that takes place is summarized as follows[4].

1. The network operator expresses the attachment points where both the BGP speakers (router r2) and external routers (router r1) are attached, as well as which BGP speaker is in charge of peering with a certain router. This operation is done using a configuration file that is loaded into ONOS prior to activating the SDN-IP application.
2. Once activated, SDN-IP application parses the configurations and puts in communication the external routers with the BGP speakers. This is translated by ONOS into flows and inserted to the switches. Additionally, the controller opens the port 2000 to communicate with the BGP speakers.
3. Assuming the BGP speakers and the external routers have been configured to peer via BGP, once the flows are installed in the switches, each external router can create EBGP sessions with the BGP speakers.
4. Routes get advertised from the external routers to the BGP speaker via EBGP.
5. Routes propagate among the BGP speakers within the SDN network, as well as to the SDN-IP application via IBGP.
6. The learned routes are advertised by the BGP speakers to other external routers.
7. SDN-IP application translates the learned routes into requests understood by ONOS. The latter translates the requests into OpenFlow entries on the switches.
8. External routers within legacy networks communicate directly through the OpenFlow data plane.

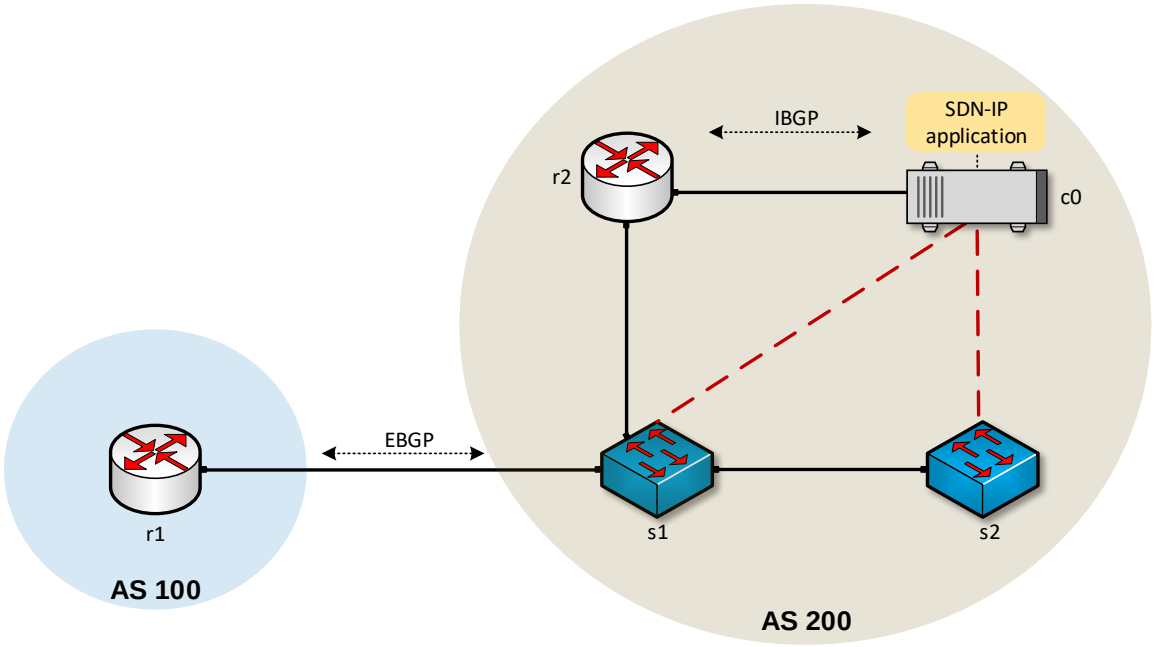

Figure 3. Integrating SDN and IP networks.

2 Lab topology

Consider Figure 4. The topology consists of two IP networks (AS 200 and AS 300) and one SDN network (AS 100). IP networks connect to the SDN network through their BGP routers. Router r1 is a BGP router within the SDN network. It communicates with EBGP routers r2 and r3 via the network addresses 192.168.12.0/30 and 192.168.13.0/30, respectively. Furthermore, router r1 is connected to the controller in order to propagate the BGP advertisements to the SDN-IP application running on top of ONOS controller. Router r1 and the controller are connected via the network 10.0.0.0/24.

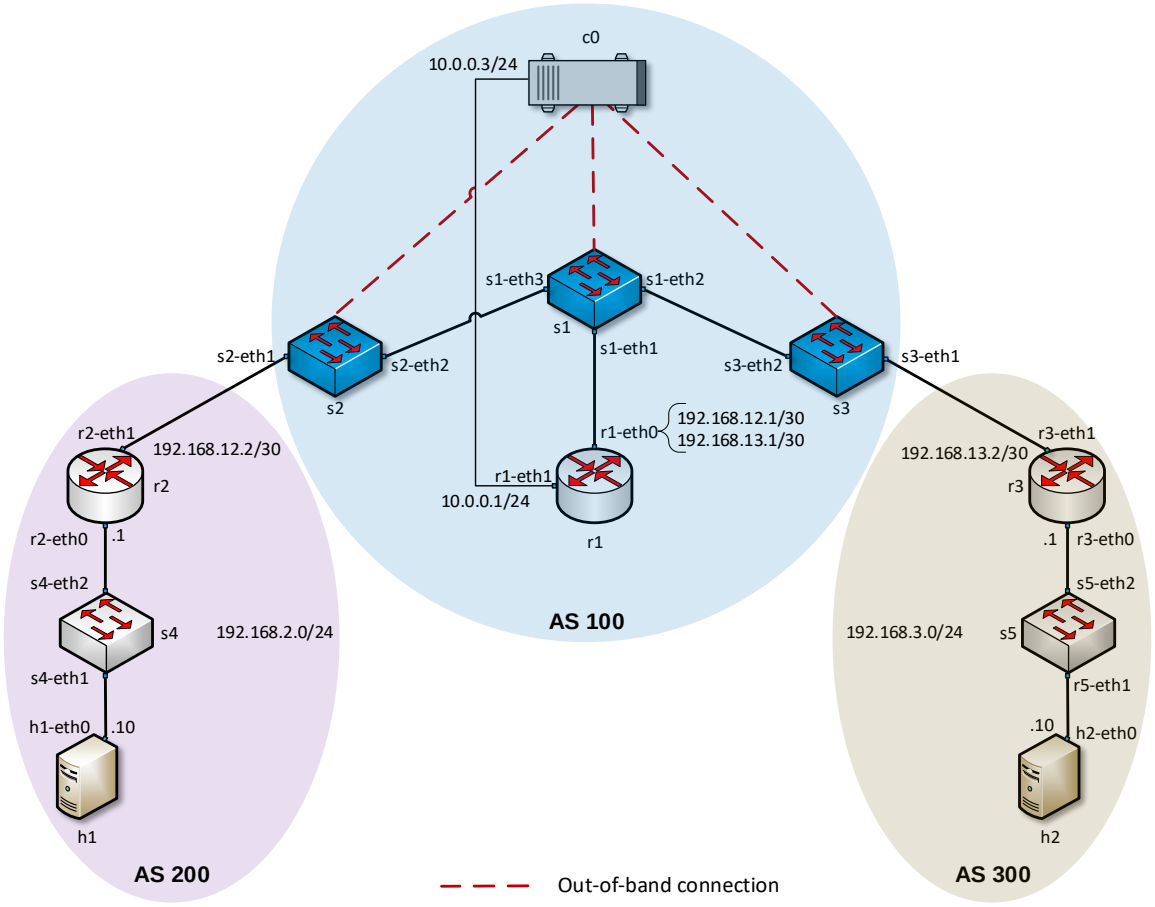

Figure 4. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet	Default gateway
Router r1	r1-eth0	192.168.12.1	/30	N/A
		192.168.13.1	/30	N/A
	r1-eth1	10.0.0.1	/24	N/A
Router r2	r2-eth0	192.168.2.1	/24	N/A
	r2-eth1	192.168.12.2	/30	N/A
Router r3	r3-eth0	192.168.3.1	/24	N/A
	r3-eth1	192.168.13.2	/30	N/A
Host h1	h1-eth0	192.168.2.10	/24	192.168.2.1
Host h2	h2-eth0	192.168.3.10	/24	192.168.3.1

c0	n/a	127.0.0.1	/32	n/a
	n/a	10.0.0.3	/24	n/a

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), saving the topologies, and exporting layer 2 models.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

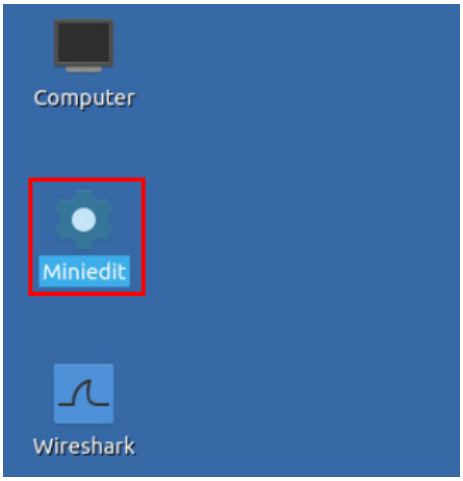

Figure 5. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab8.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab8* and click on *Open*.

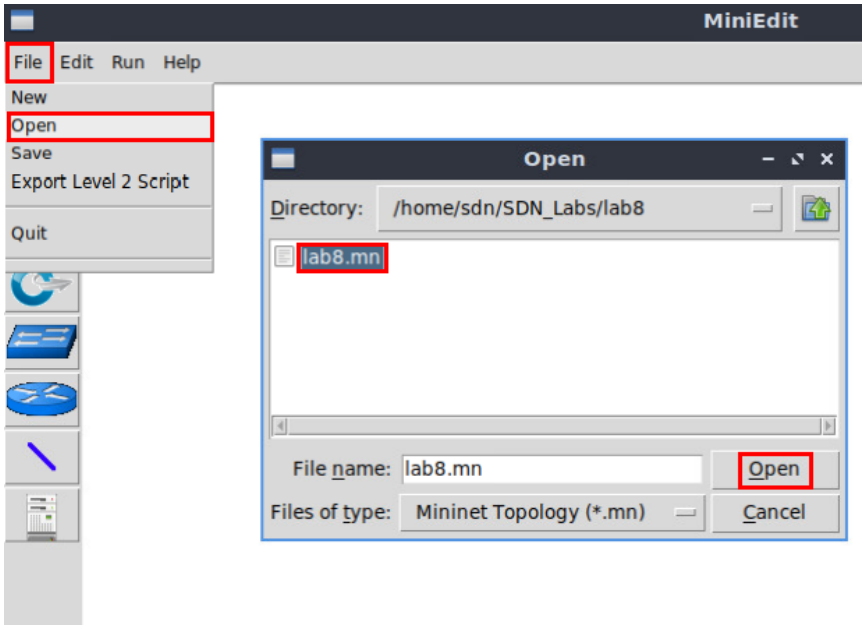

Figure 6. Opening topology.

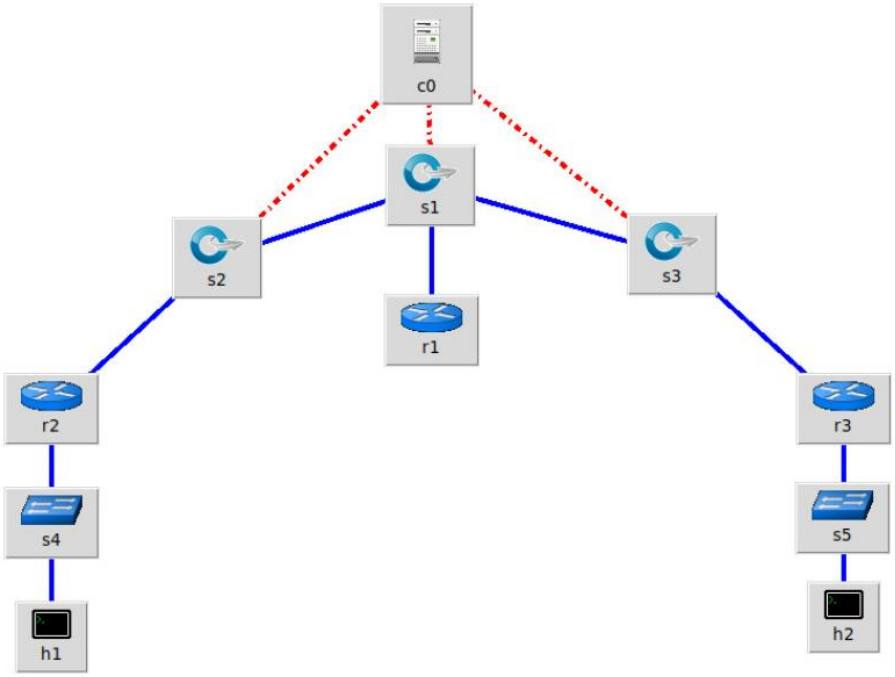

Figure 7. MiniEdit's topology.

2.3 Load the configuration file

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open Linux terminal.

Figure 8. Opening Linux terminal.

Step 2. Click on the Linux terminal and navigate into *SDN_Labs/lab8* directory by issuing the following command. This folder contains a configuration file and the script responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab8
```

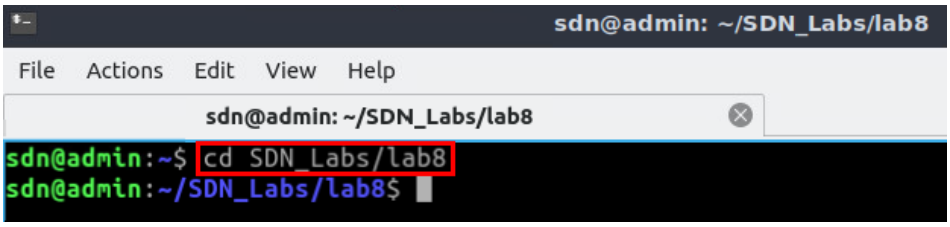

Figure 9. Entering the *SDN_Labs/lab8* directory.

Step 3. To execute the shell script, type the following command. The argument of the program corresponds to the configuration zip file that will be loaded in all the routers in the topology.

```
./config_loader.sh lab8_conf.zip
```

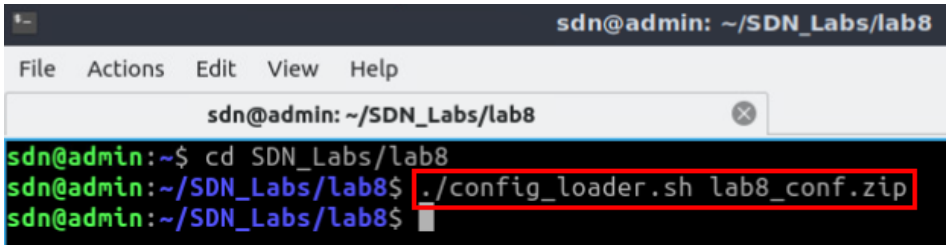

Figure 10. Executing the shell script to load the configuration.

Step 4. Type the following command to exit the Linux terminal.

```
exit
```

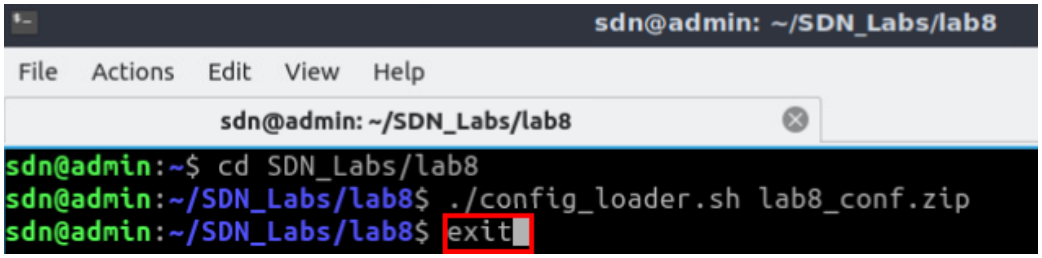

Figure 11. Exiting from the terminal.

2.4 Run the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. At this point host h1 and host h2 interfaces are configured. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

Figure 12. Starting the emulation.

Step 2. Issue the following command on Mininet terminal to display the interface names and connections.

```
links
```

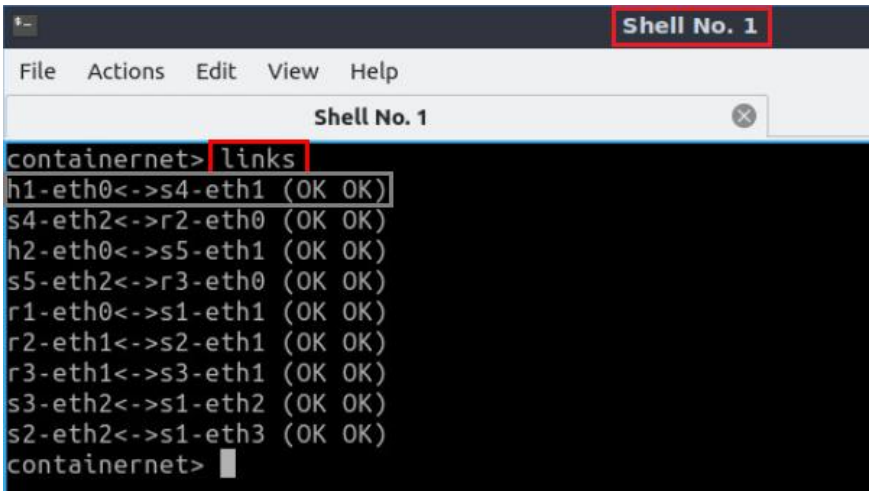

Figure 13. Displaying network interfaces.

In Figure 11, the link displayed within the gray box indicates that interface *eth1* of switch s4 connects to interface *eth0* of host h1 (i.e., *s4-eth1<->h1-eth0*).

2.5 Verify the configuration

You will verify the IP addresses listed in Table 2 and inspect the routing table of routers r1, r2, and r3.

Step 1. Hold right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands on that host.

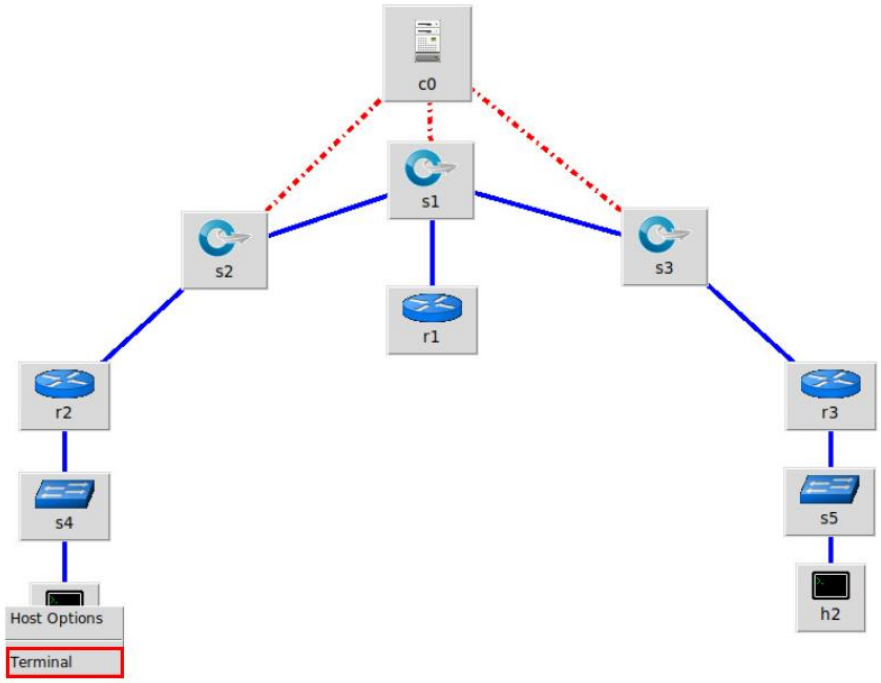

Figure 14. Opening a terminal on host h1.

Step 2. On host h1 terminal, type the command shown below to verify that the IP address was assigned successfully. You will corroborate that host h1 has two interfaces, *h1-eth0* configured with the IP address 192.168.2.10 and the subnet mask 255.255.255.0.

```
ifconfig
```

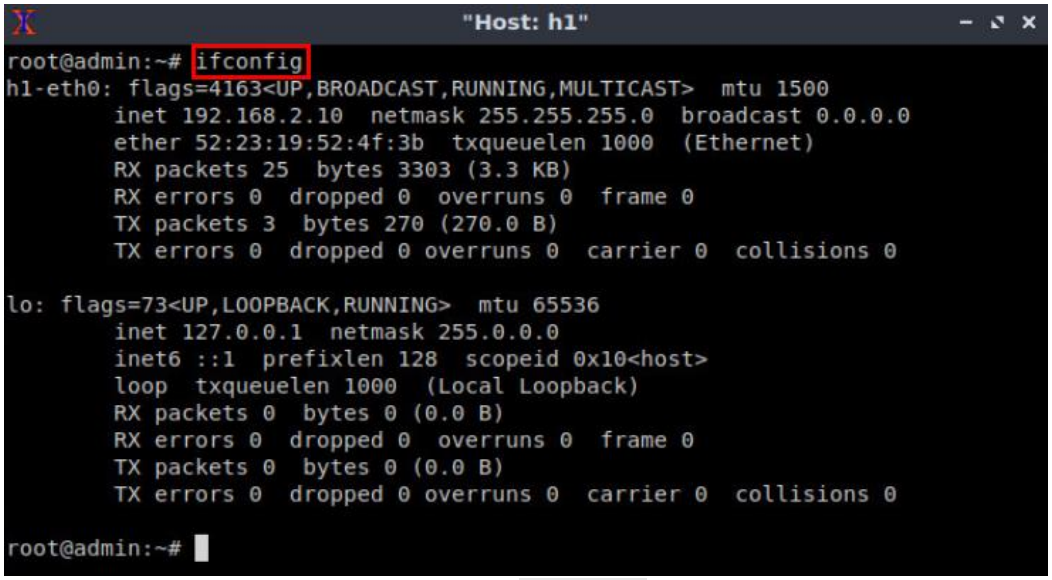

Figure 15. Output of `ifconfig` command.

Step 3. On host h1 terminal, type the command shown below to verify that the default gateway IP address is 192.168.2.1.

```
route
```

Figure 16. Output of `route` command.

Step 4. In order to verify host h2 default route, proceed similarly by repeating from step 1 to step 3 on host h2 terminal. Similar results should be observed.

Step 5. In order to verify router r1, hold right-click on router r1 and select *Terminal*.

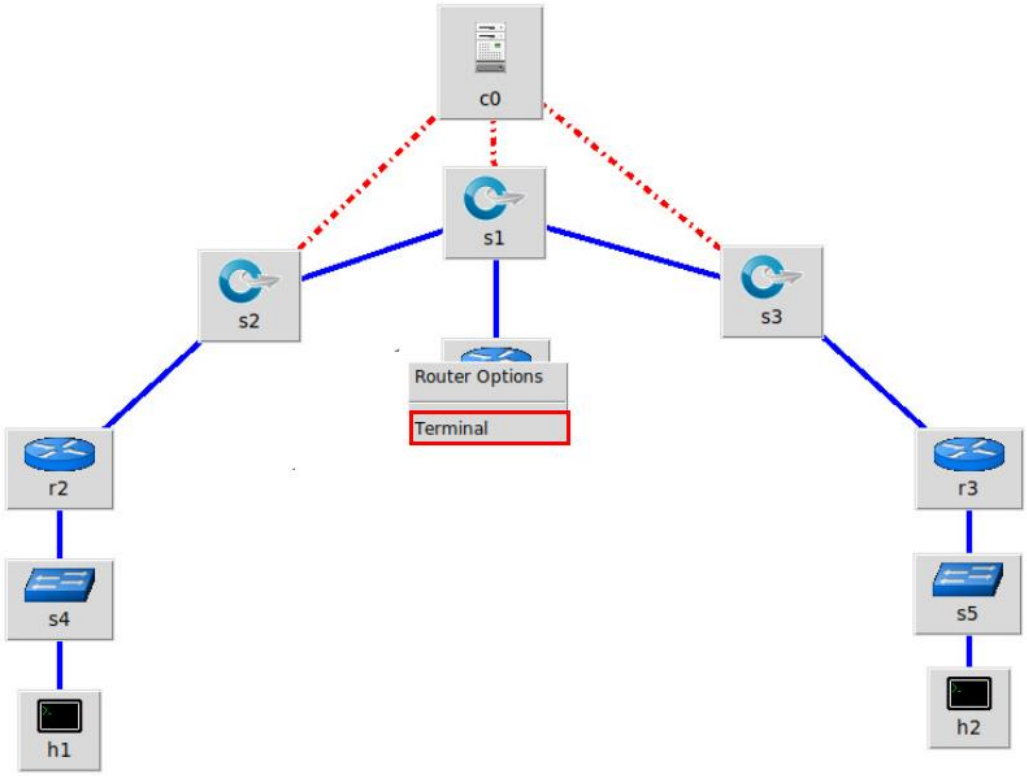

Figure 17. Opening a terminal on router r1.

Step 6. In this step, you will start zebra daemon, which is a multi-server routing software that provides TCP/IP based routing protocols. The configuration will not be working if you do not enable zebra daemon initially. In order to start the zebra, type the following command.

```
zebra
```

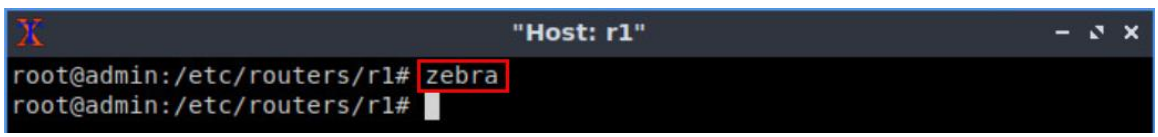

Figure 18. Starting zebra daemon.

Step 7. After initializing zebra, vtysh should be started in order to provide all the CLI commands defined by the daemons. To proceed, issue the following command.

```
vtysh
```

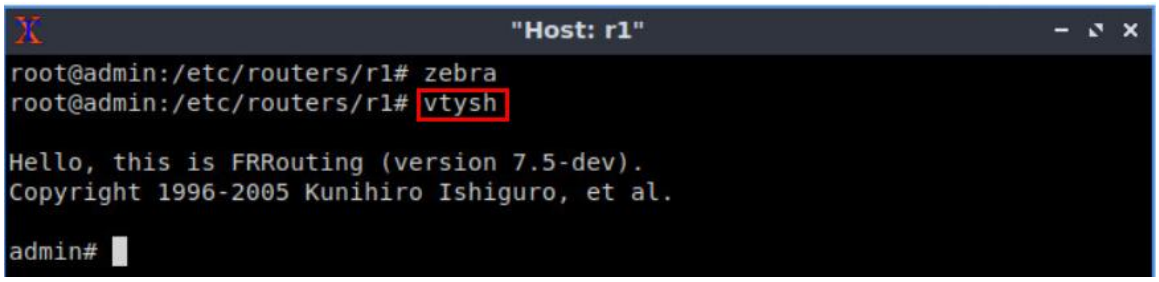

Figure 19. Starting vtysh on router r1.

Step 8. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 does not contain any route to the network of router r2 (192.168.2.0/24) or router r3 (192.168.3.0/24) as there is no routing protocol configured yet.

```
show ip route
```

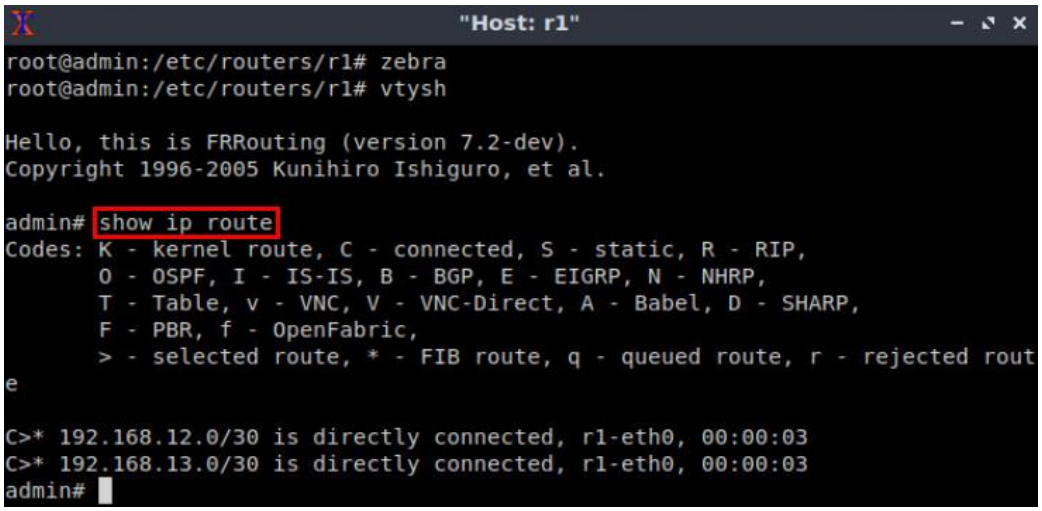

Figure 20.Displaying routing table of router r1.

The output in the figure above shows that the networks 192.168.12.0/24 and 192.168.13.0/30 are directly connected through the interface *r1-eth0*.

Step 9. Hold right-click on router r2 and select *Terminal*.

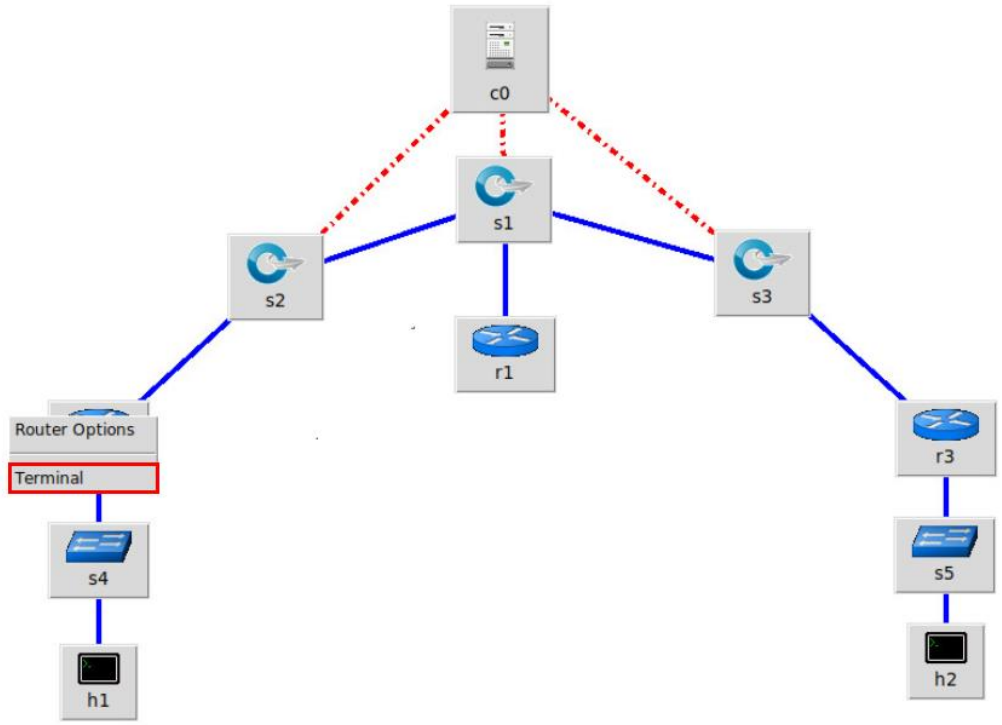

Figure 21. Opening a terminal on router r2.

Step 10. Router r2 is configured similarly to router r1 but, with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r2 terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r2.

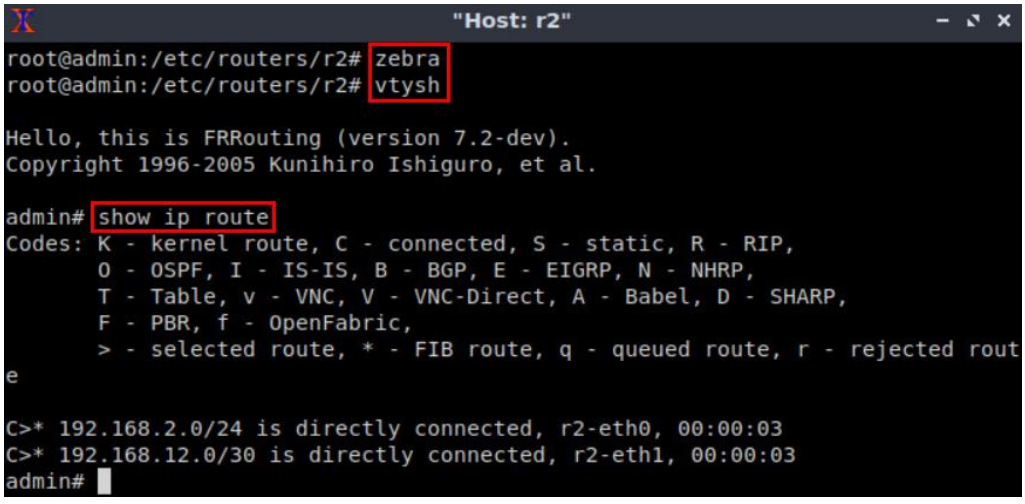

Figure 22. Displaying routing table of router r2.

Step 11. Router r3 is configured similarly to router r1 but, with different IP addresses (see Table 2). Those steps are summarized in the following figure. To proceed, in router r3 terminal issue the commands depicted below. At the end, you will verify all the directly connected networks of router r3.

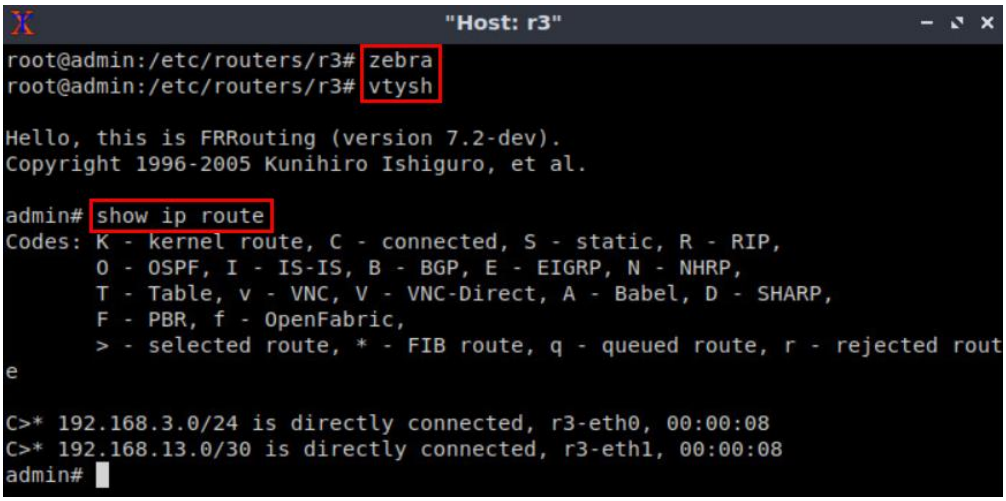

Figure 23. Displaying routing table of router r3.

3 Configuring BGP within legacy networks

In the previous section you used a script to assign the IP addresses to all devices' interfaces. In this section you will configure BGP routing protocol on the legacy networks (routers r2 and r3), the standard protocol used to connect ASes. First, you will initialize the daemon that enables BGP configuration. Then, you need to assign BGP neighbors to allow BGP peering to the remote neighbor. Additionally, you will advertise the local networks so that they are advertised to EBGP neighbors.

Step 1. To configure BGP routing protocol, you need to enable the BGP daemon first. In router r2, type the following command to exit the vtysh session.

```
exit
```

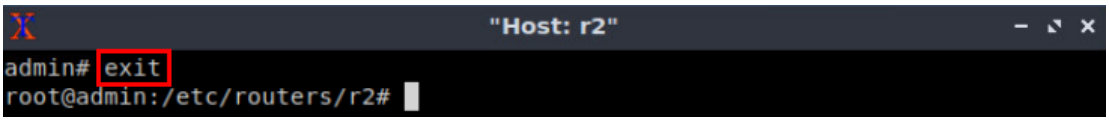

Figure 24. Exiting the vtysh session.

Step 2. Type the following command on router r2 terminal to start BGP routing protocol.

```
bgpd
```

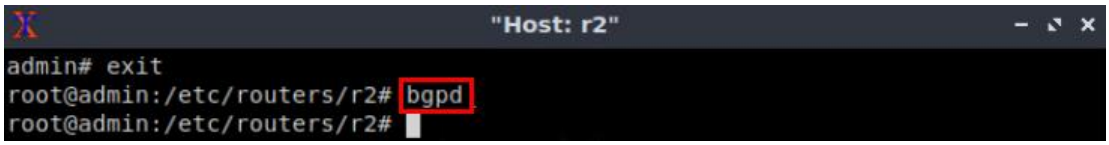

Figure 25. Starting BGP daemon.

Step 3. In order to enter to router r2 terminal, type the following command.

```
vtysh
```

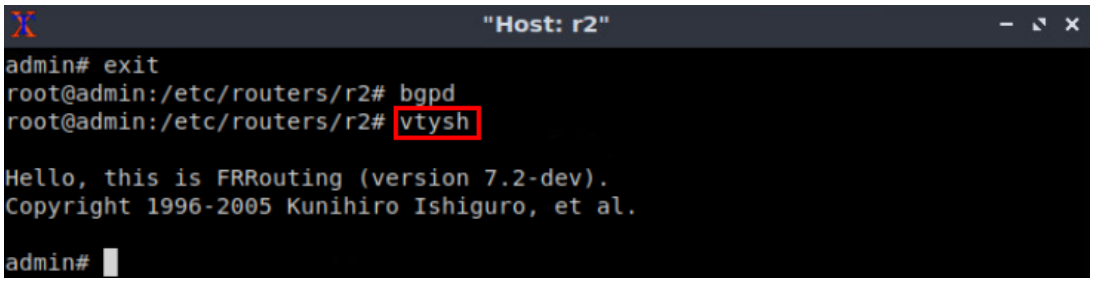

Figure 26. Starting vtysh on router r2.

Step 4. To enable router r2 configuration mode, issue the following command.

```
configure terminal
```

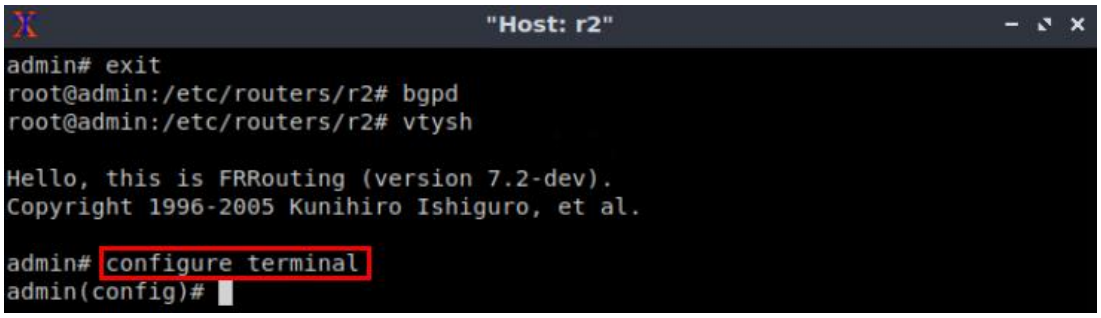

Figure 27. Enabling configuration mode on router r1.

Step 5. The Autonomous System Number (ASN) assigned for router r2 is 200. In order to configure BGP, type the following command.

```
router bgp 200
```

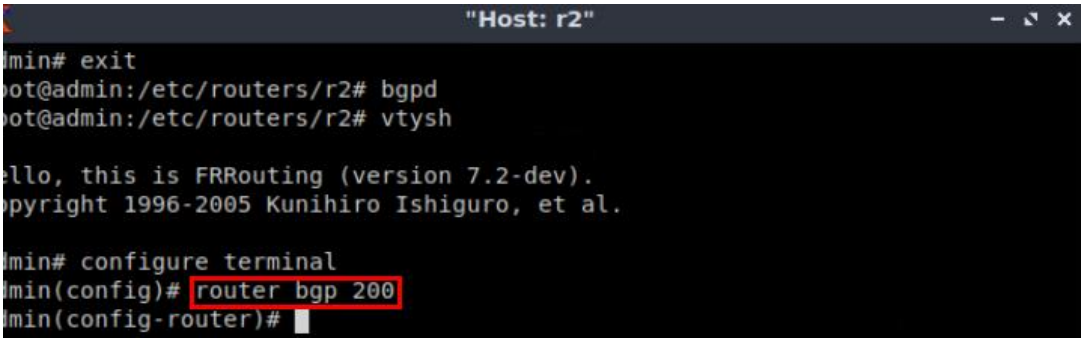

Figure 28. Configuring BGP on router r2.

Step 6. To configure a BGP neighbor to router r2 (AS 200), type the command shown below. This command specifies the neighbor IP address (192.168.12.1) and ASN of the remote BGP peer (AS 100).

```
neighbor 192.168.12.1 remote-as 100
```

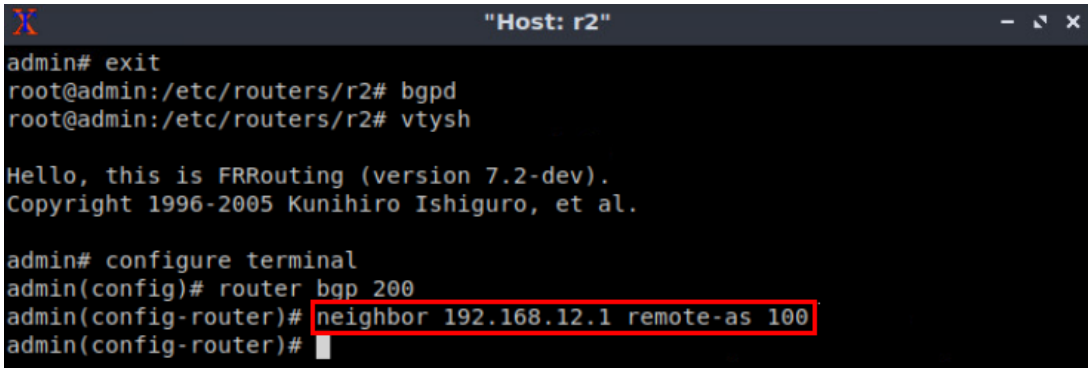

Figure 29. Assigning BGP neighbor to router r1.

Step 7. Issue the following command so that router r2 advertises the network 192.168.2.0/24.

```
network 192.168.2.0/24
```

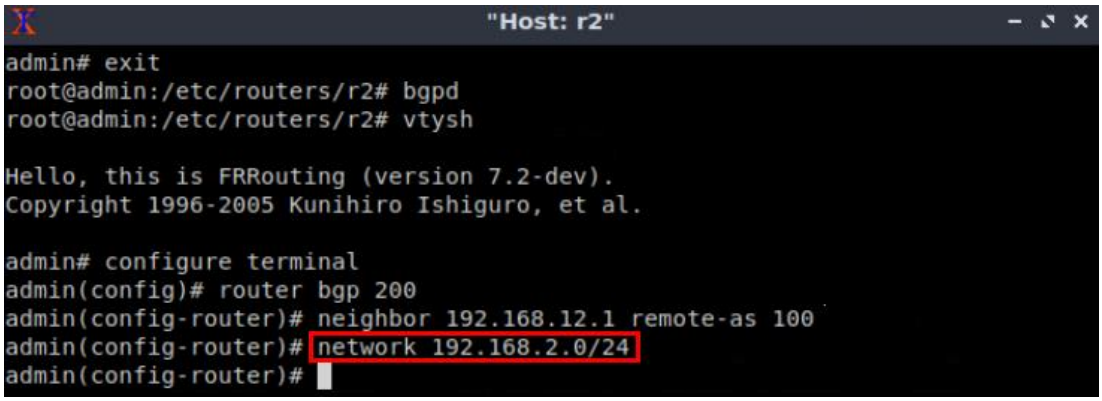

Figure 30. Advertising the network connected to router r2.

Step 8. Type the following command to exit from the configuration mode.

```
end
```

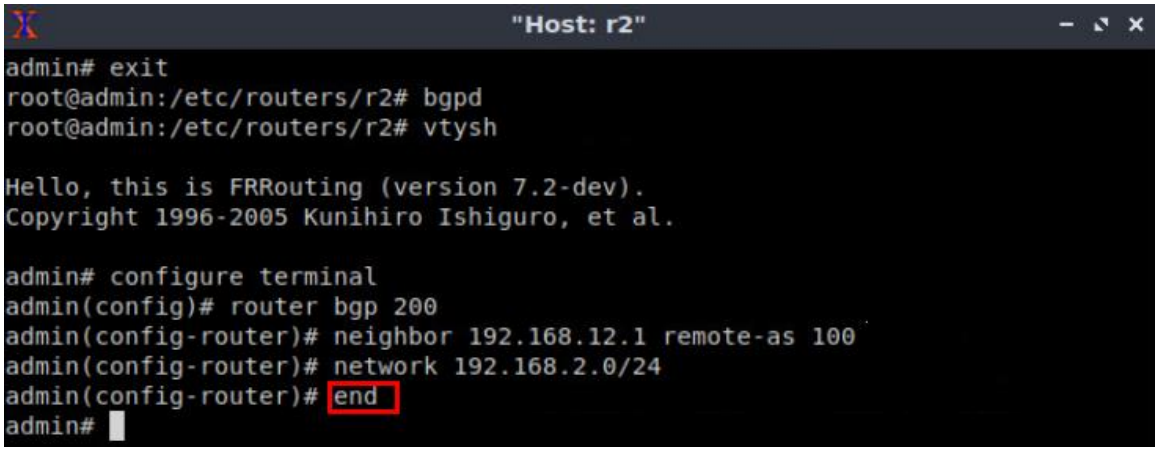

Figure 31. Exiting from configuration mode.

Step 9. Type the following command to verify BGP neighbors. You will verify that the neighbor IP address is 192.168.12.1. The corresponding ASN is 100.

```
show ip bgp neighbors
```

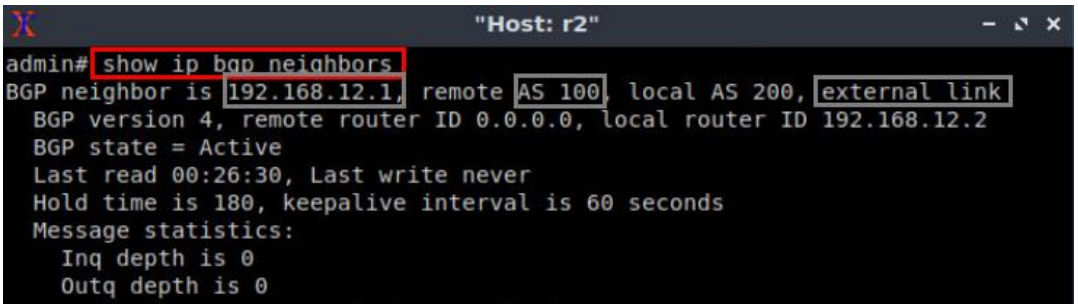

Figure 32. Verifying BGP neighbors on router r2.

Step 10. The configuration of BGP on router r3 is similarly configured as router r2. Router r3 lies within AS 300, it establishes BGP neighbor relationship with router r1 (192.168.13.1) in AS 100, and advertises the network 192.168.3.0/24. The configuration of BGP on router r3 is depicted in the below figure.

```
end
```

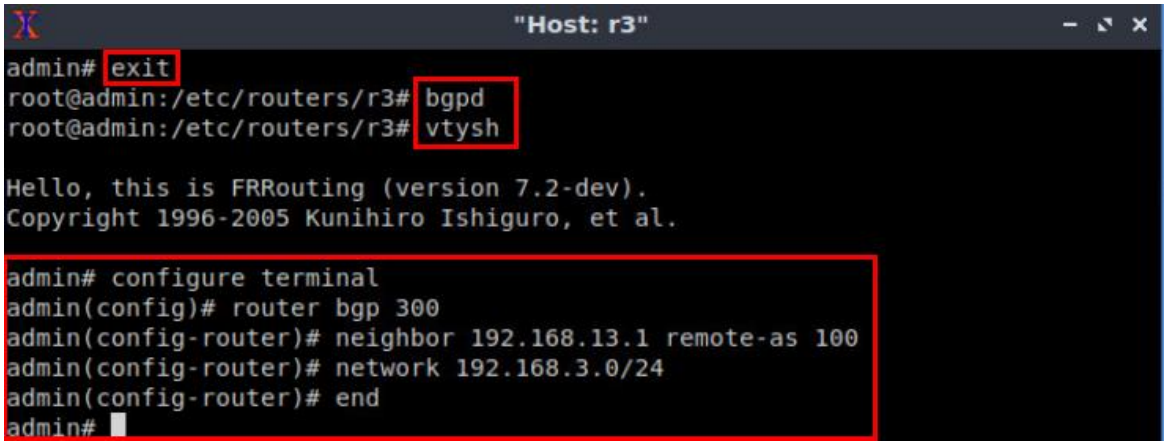

Figure 33. BGP configuration on router r3.

Step 11. To verify BGP neighbors of router r3, type the following command.

```
show ip bgp neighbors
```

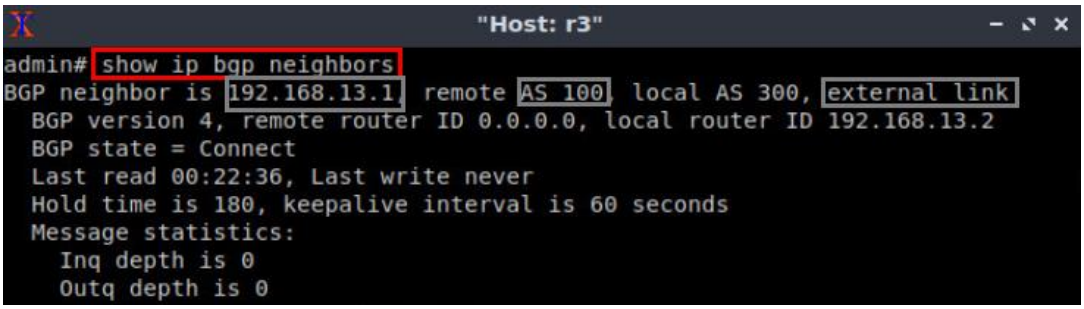

Figure 34. Verifying BGP neighbors on router r3.

4 Starting ONOS controller

In the section, you will start ONOS controller and activate OpenFlow application so that the controller discovers the devices, hosts, and links in the topology.

Step 1. Go to the opened linux terminal.

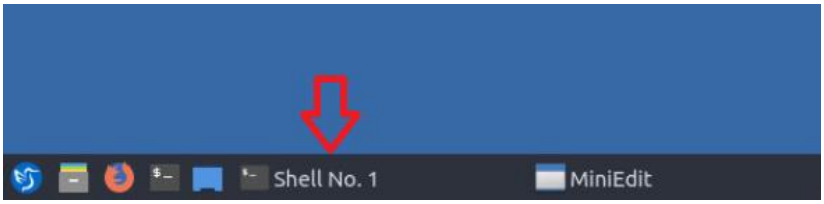

Figure 35. Opening Linux terminal.

Step 2. Click on *File>New Tab* to open an additional tab in Linux terminal. Alternatively, the user may press `Ctrl+Shift+T`.

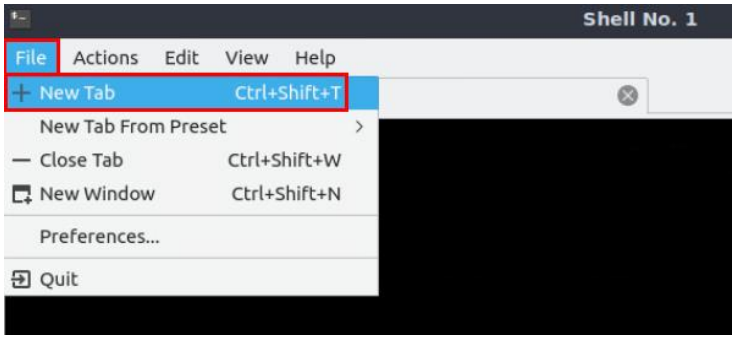

Figure 36. Opening an additional tab.

Step 3. Navigate into *SDN_Labs/lab8* directory by issuing the following command.

```
cd SDN_Labs/lab8
```

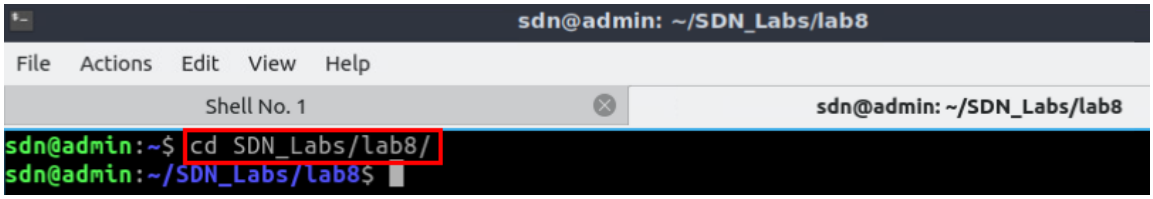

Figure 37. Entering the *SDN_Labs/lab8* directory.

Step 4. Issue the command below to execute programs with the security privileges of the superuser (root). When prompted for a password, type `password`.

```
sudo su
```

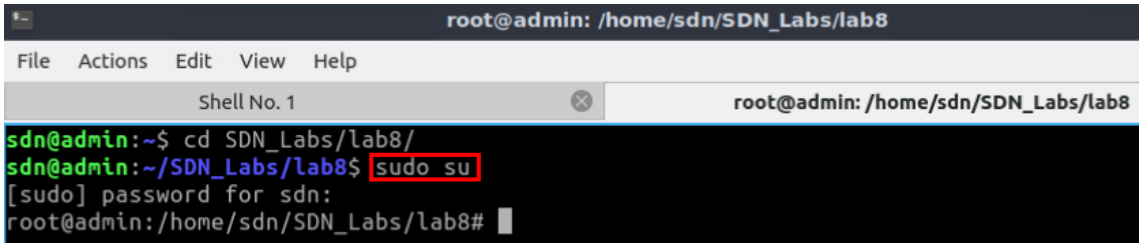

Figure 38. Switching to root mode.

Step 5. A script was written to run ONOS and enter its Command Line Interface (CLI). In order to run the script, issue the following command.

```
./run_onos
```

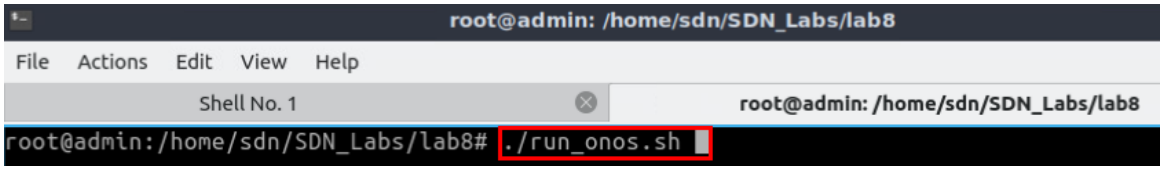

Figure 39. Running ONOS instance.

Once the script finishes executing and ONOS is ready, you will be able to execute commands on ONOS CLI as shown in the figure below. Note that this script may take a couple of minutes.

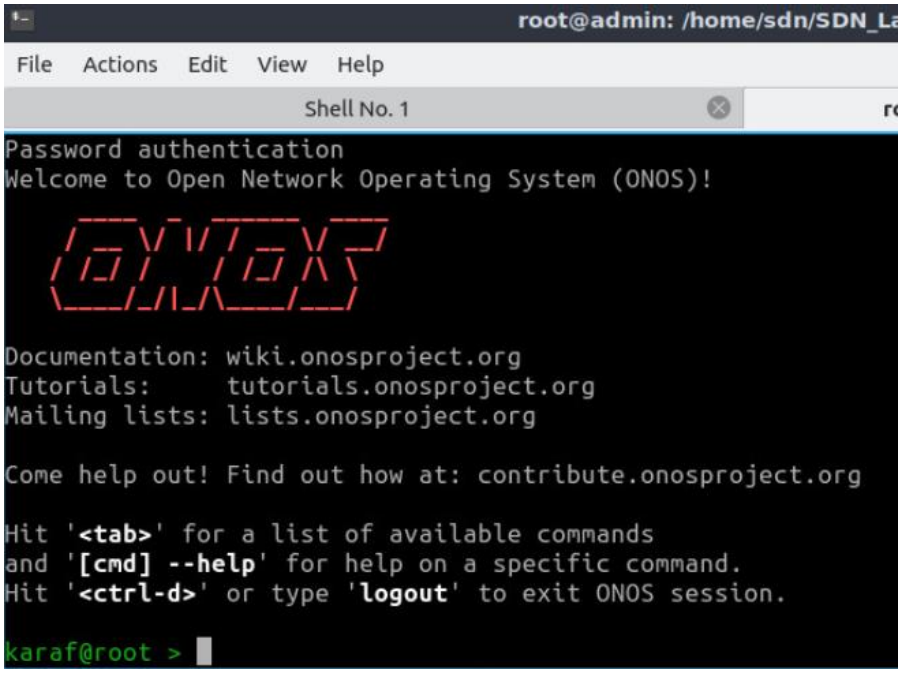

Figure 40. ONOS CLI.

Step 6. In ONOS terminal, issue the following command to activate the OpenFlow application.

```
app activate org.onosproject.openflow
```

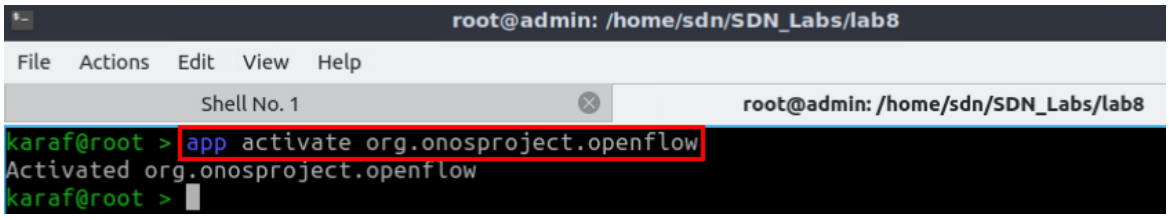

Figure 41. Activating OpenFlow application.

Note that when you activate any ONOS application, you may have to wait few seconds so that the application gives the correct output.

Step 7. To display the list of all currently known devices (OVS switches), type the following command.

```
devices
```

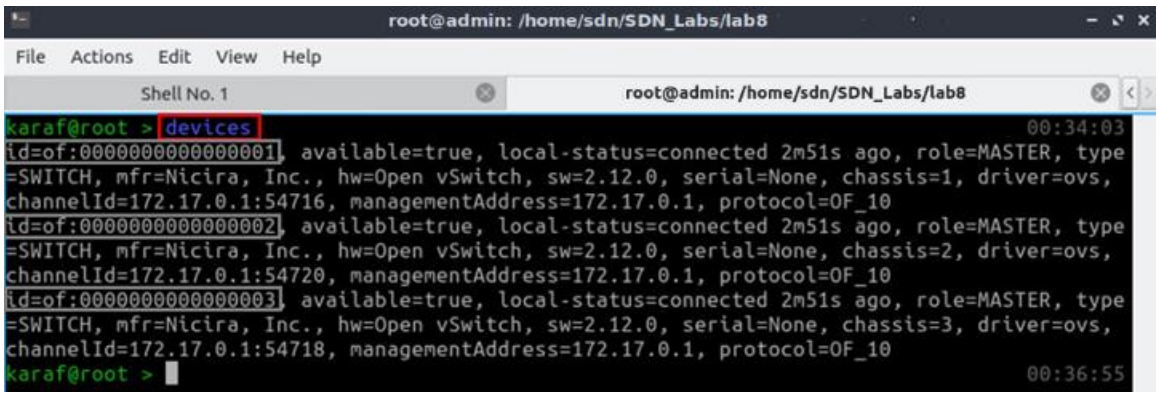

Figure 42. Displaying the current known devices (switches).

Step 8. To display the list of all currently known links, type the following command.

```
links
```

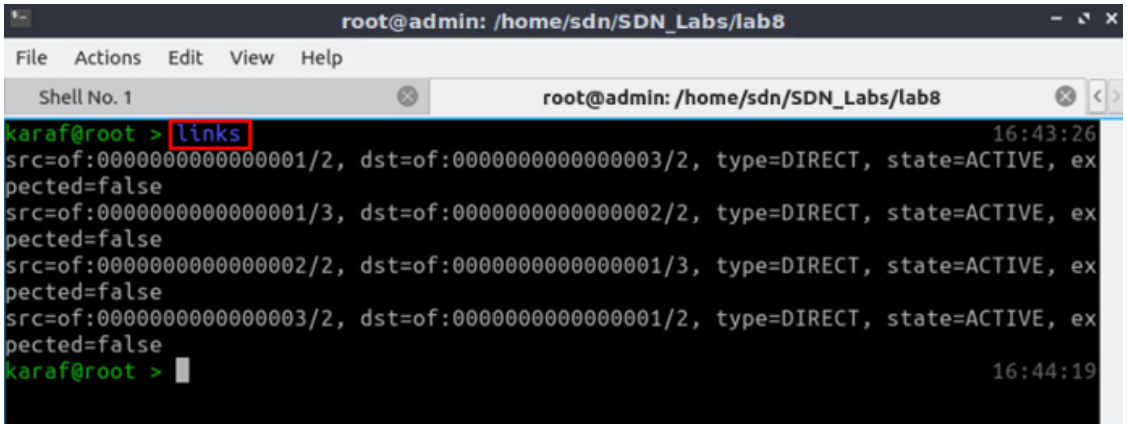

Figure 43. Displaying the current known links.

Step 9. To display the list of all currently known hosts, type the following command.

```
hosts
```

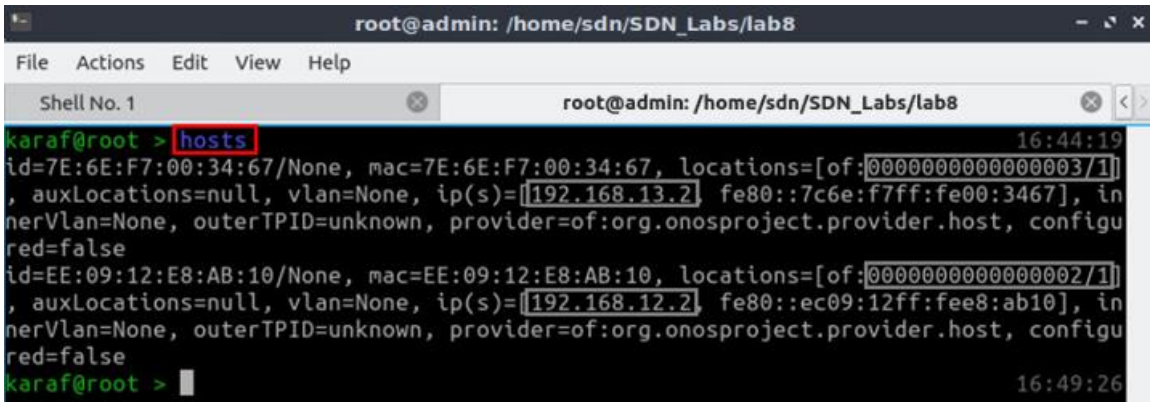

Figure 44. Displaying the current known links.

Consider Figure 44. ONOS recognizes router r2 (192.168.12.2) and router r3 (192.168.13.2) and display the interfaces of the OpenFlow switches they are connected to. Note that you might have to wait until ONOS discovers the two hosts in case they don't appear immediately.

5 Integrating SDN and legacy networks

In the previous sections, you configured the legacy devices, as well as started ONOS and its OpenFlow application to discover the topology. In this section, you will first execute a script that connects the IBGP speaker (router r1) with ONOS controller, thus, the two entities can communicate. Additionally, you will configure BGP on router r1 so that it peers with routers r2 and r3 in the external networks, as well as with ONOS. Furthermore, you will activate ONOS SDN-IP application to interconnect the three ASes.

5.1 Connecting the IBGP speaker (router r1) with ONOS controller

In this section, you will execute a script that creates a peer-to-peer link connecting router r1 with ONOS.

Step 1. Go to Mininet tab in the Linux terminal.

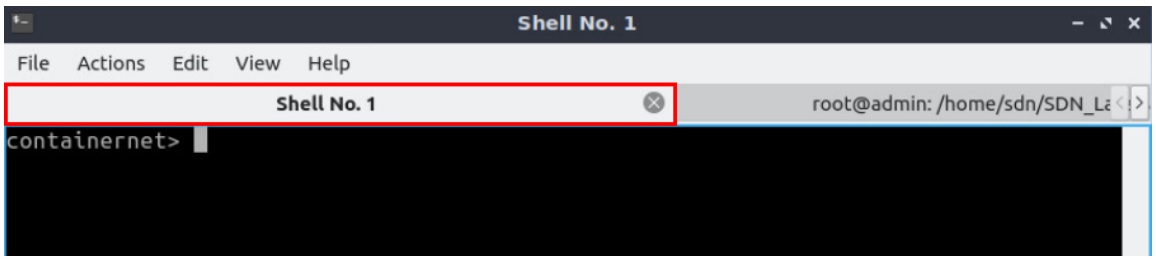

Figure 45. Opening Mininet tab.

Step 2. In order to create a point-to-point network between the IBGP speaker (router r1) and ONOS, a script was written to facilitate the process. In order to execute the script, type the following command.

```
source create_link.sh
```

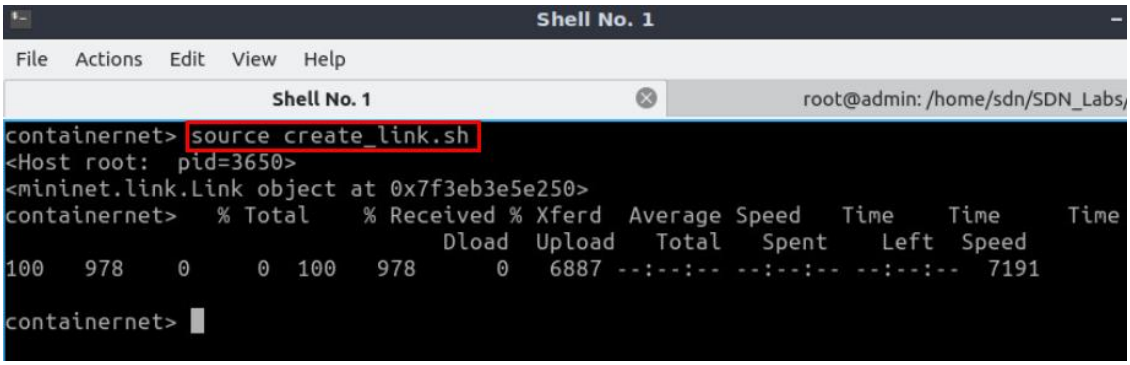

Figure 46. Creating a point-to-point network (link) between the IBGP speaker and ONOS controller.

Consider Figure 46. The script creates a point-to-point network between router r1 and ONOS. The network address of the point-to-point network is 10.0.0.0/24. Router r1 is assigned the IP address 10.0.0.1/24, whereas ONOS controller is assigned 10.0.0.3/24.

Step 3. In router r1 terminal, type the following command to exit the vtysh session.

```
exit
```

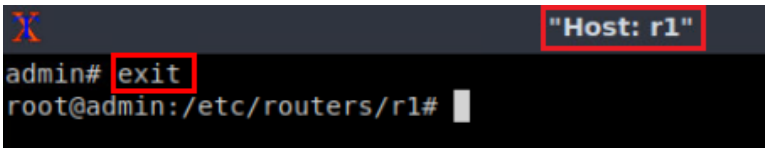

Figure 47. Creating a network between the IBGP speaker and ONOS controller.

Step 4. Now that router r1 is connected to ONOS controller, a new interface must appear in it. In order to verify the connected interface, type the following command.

```
ifconfig
```

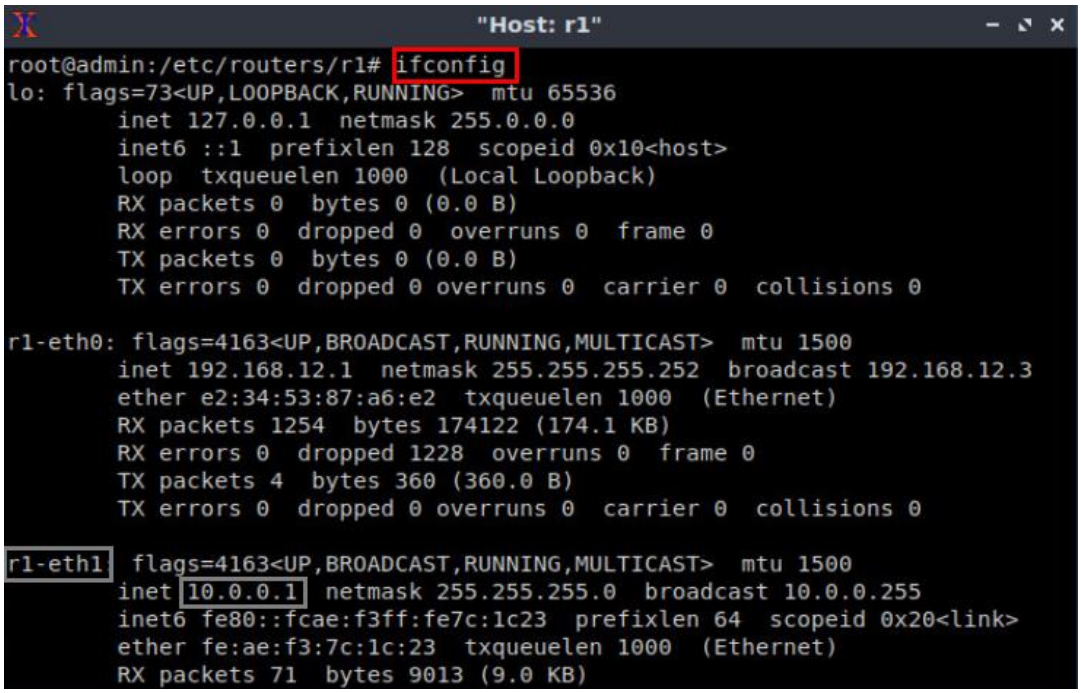

Figure 48. Listing router r1's interfaces.

Consider Figure 48. Interface *r1-eth1* is added after creating a point-to-point network between router r1 and ONOS controller. Furthermore, the interface is associated with the IP address 10.0.0.1.

5.2 Configuring BGP on router r1

In this section, you will configure BGP on router r1 to peer with routers r2 and r3, as well as with ONOS.

Step 1. Type the following command on router r1 terminal to start BGP routing protocol.

```
bgpd
```

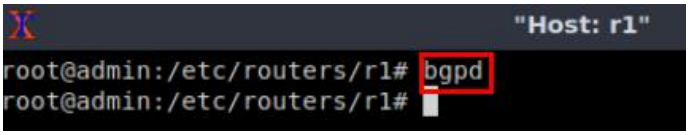

Figure 49. Starting BGP daemon.

Step 2. In order to enter to router r1 terminal, type the following command.

```
vtysh
```

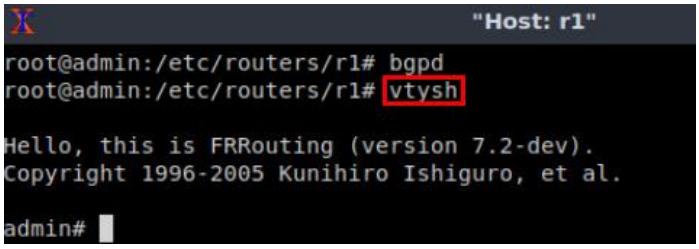

Figure 50. Starting vtysh on router r1.

Step 3. To enable router r1 configuration mode, issue the following command.

```
configure terminal
```

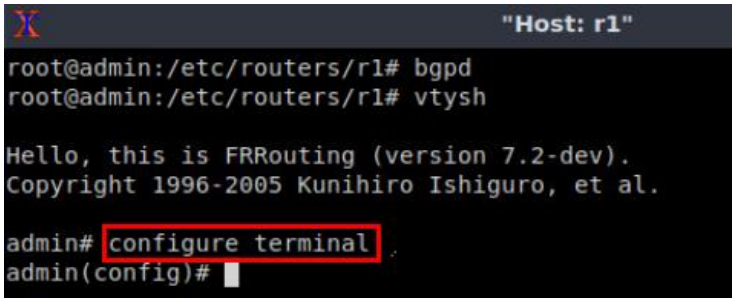

Figure 51. Enabling configuration mode on router r1.

Step 4. The ASN assigned for router r1 is 100. In order to configure BGP, type the following command.

```
router bgp 100
```

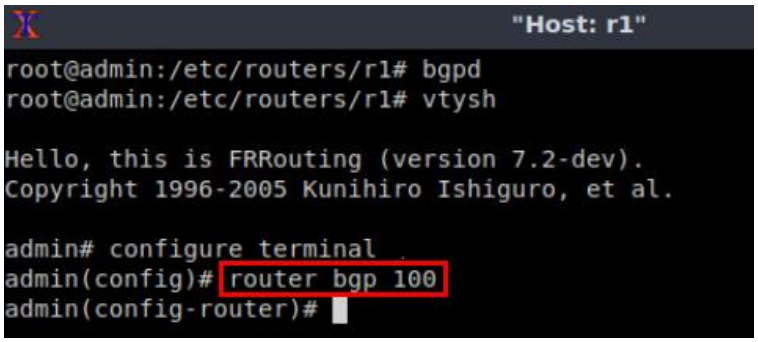

Figure 52. Configuring BGP on router r1.

Step 5. To configure a BGP neighbor to router r1 (AS 100), type the command shown below. This command specifies the neighbor IP address (192.168.12.2) and ASN of the remote BGP peer (AS 200).

```
neighbor 192.168.12.2 remote-as 200
```

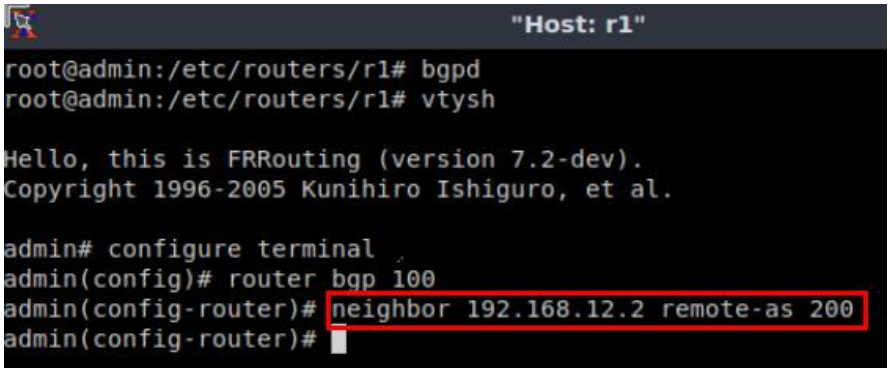

Figure 53. Assigning BGP neighbor to router r1.

Step 6. Similarly, add router r3 (192.168.13.2) in AS 300 as a BGP neighbor to router r1.

```
neighbor 192.168.12.2 remote-as 200
```

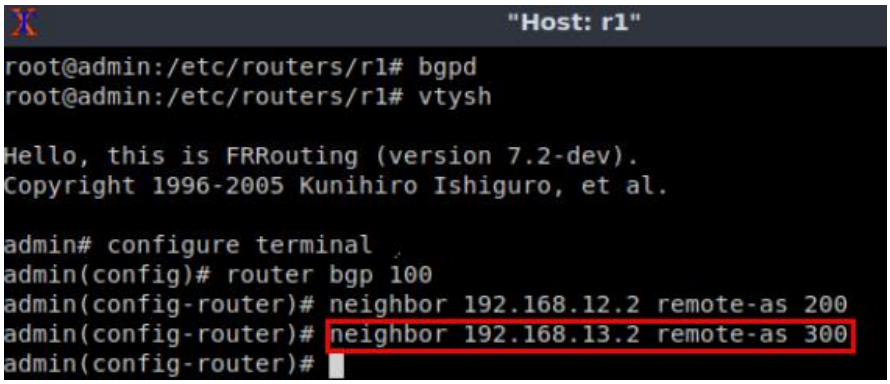

Figure 54. Assigning BGP neighbor to router r1.

Step 7. Router r1 and ONOS controller are connected using a point-to-point network (10.0.0.0/24). The IP address assigned to the controller is 10.0.0.3. As router r1 is the IBGP speaker within the SDN network, it must establish a BGP peering relationship with the controller in its network (AS 100). In order to establish BGP peering relationship with the controller, type the following command.

```
neighbor 10.0.0.3 remote-as 100
```

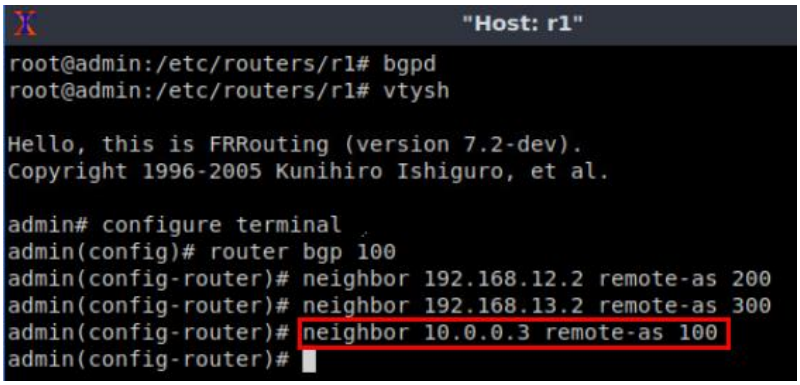

Figure 55. Assigning BGP neighbor to router r1.

Step 8. By default, ONOS listens on TCP port number 2000 for incoming BGP connections, which is not the default BGP port number 179. In order to specify the port for incoming BGP messages from ONOS, write the following command.

```
neighbor 192.168.12.2 port 2000
```

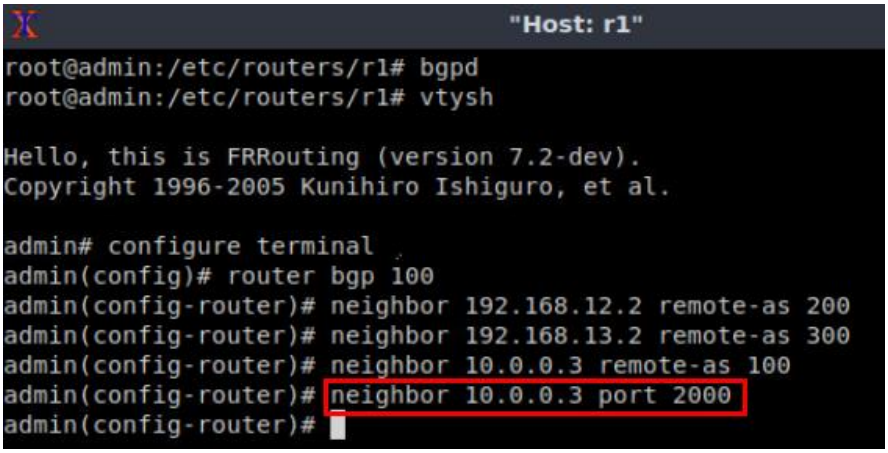

Figure 56. Changing the Listening port for BGP connections.

Step 9. Type the following command to exit from the configuration mode.

```
end
```

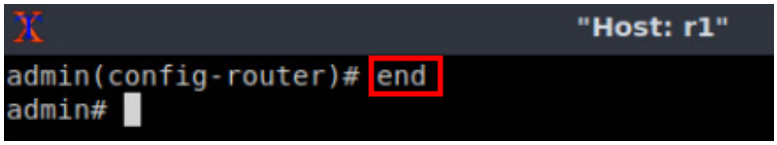

Figure 57. Exiting from configuration mode.

Step 10. Type the following command on router r1 terminal to verify the routing table of router r1. It will list all the directly connected networks. The routing table of router r1 does not contain any route to the network of router r2 (192.168.2.0/24) or router r3 (192.168.3.0/24) as there is no enabled ONOS application that deals with BGP routes.

```
end
```

Figure 58. Displaying the routing table of router r1.

5.3 Activating SDN-IP application

In this section, you will activate the SDN-IP application and other dependencies (applications) that will interconnect the SDN network with the legacy network.

Step 1. Go to ONOS terminal.

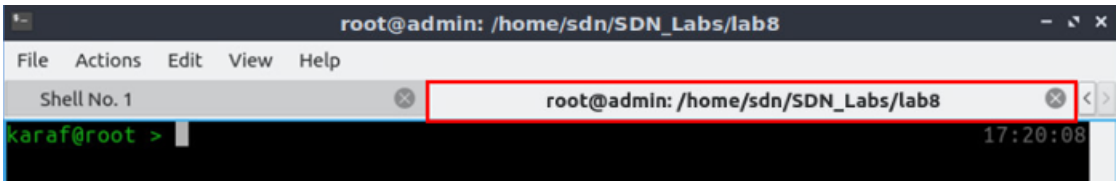

Figure 59. Opening ONOS terminal.

Step 2. Before activating the SDN-IP application you must start the *org.onosproject.config* application. The latter is an application for the network configuration. In order to activate the config application, type the following command.

```
app activate org.onosproject.config
```

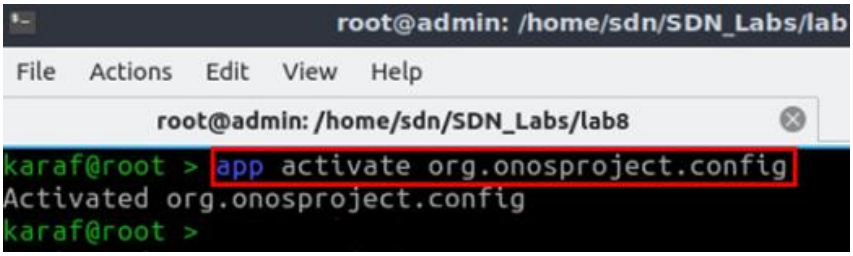

Figure 60. Activating ONOS config appication.

Step 3. SDN-IP application has an additional application dependency that it relies on to ensure Address Resolution Protocol (ARP) requests are resolved properly. This is the *org.onosproject.proxyarp* application that responds to ARP requests on behalf of hosts and external routers.

```
app activate org.onosproject.proxyarp
```

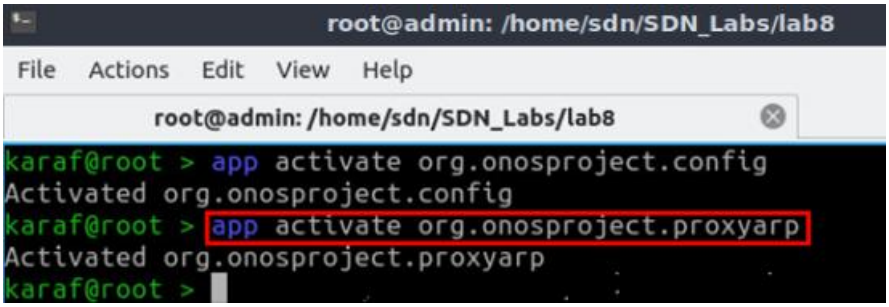

Figure 61. Activating ONOS proxyarp application.

Step 4. Once the dependencies are started, the SDN-IP application can be activated. In order to do that, type the following command.

```
app activate org.onosproject.sdnip
```

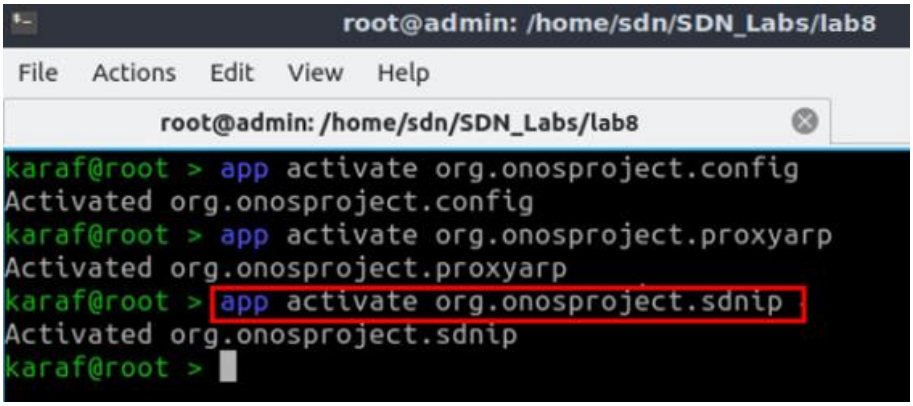

Figure 62. Activating ONOS SDN-IP application.

After activating the above applications, you might have to wait few minutes until the applications discover the topology, and exchange information in order to get correct results.

Step 5. In ONOS terminal, type the following command to show the IBGP neighbors that have connected to SDN-IP application.

```
bgp-neighbors
```

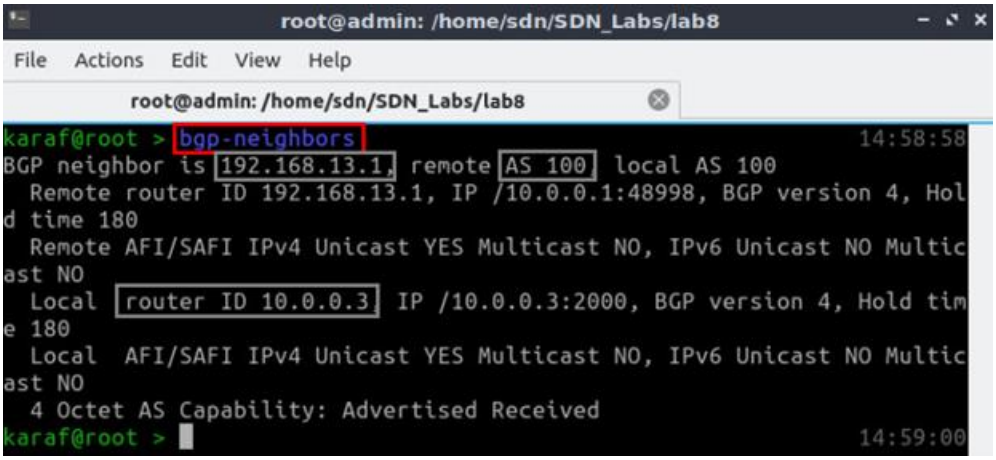

Figure 63. Viewing IBGP neighbors within the SDN network.

Consider Figure 63. The neighbor 192.168.13.1 corresponds to router r1 in AS 100. This is the internal BGP speaker in the SDN network. The local router ID that the SDN-IP application uses is 10.0.0.3.

Step 6. To show the routing table of SDN-IP, type the following command.

```
routes
```

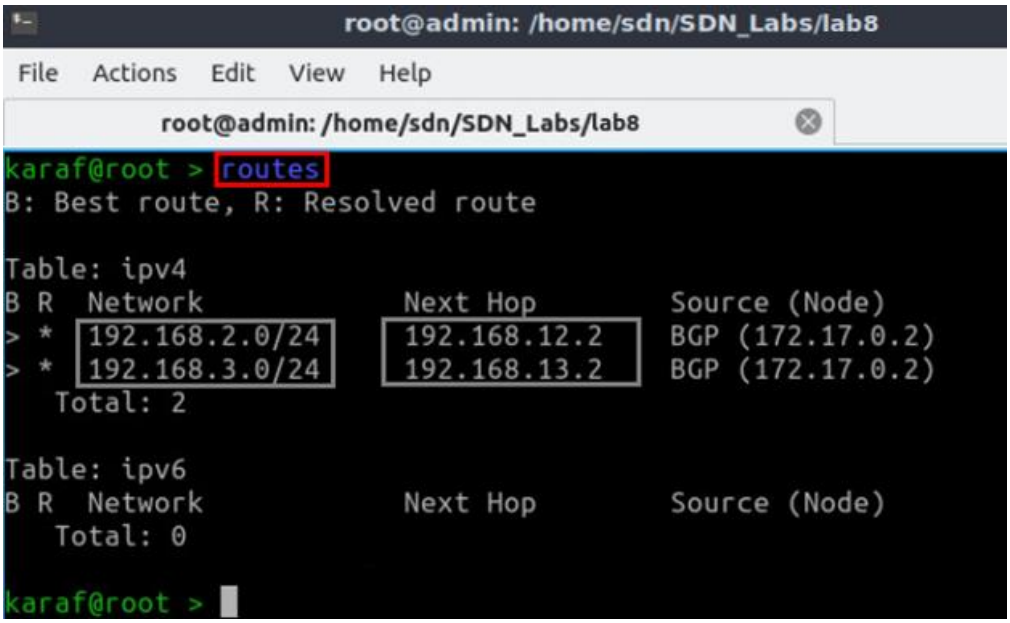

Figure 64. Showing the routing table of the SDN-IP application.

Consider Figure 64. The networks 192.168.2.0/24 and 192.168.3.0/24 are inserted in the routing table of the SDN-IP application.

6 Verifying the connectivity between the networks

Step 1. Open router r1 terminal and type the following command to show the BGP table.

```
show ip bgp
```

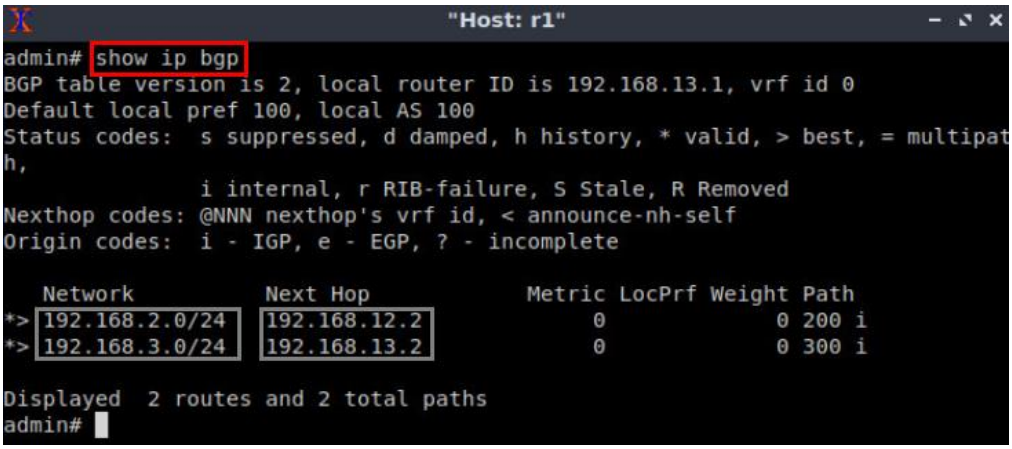

Figure 65. Showing the BGP table of router r1.

Consider Figure 65. The networks 192.168.2.0/24 and 192.168.3.0/24 are inserted in the BGP table of router r1. The next hops to reach these networks are 192.168.12.2 (router r2) and 192.168.13.2 (router r3), respectively.

Step 2. In router r1 terminal, type the following command to show the routing table.

```
show ip route
```

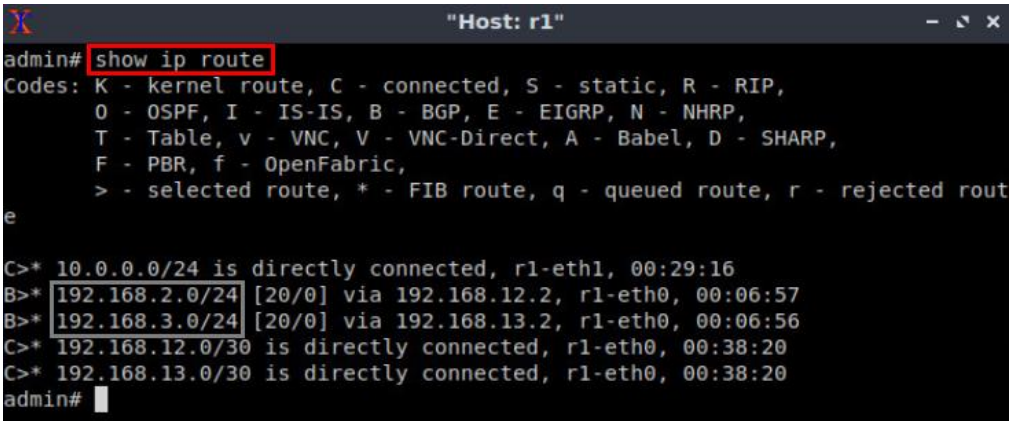

Figure 66. Showing the routing table of router r1.

Consider Figure 66. The networks 192.168.2.0/24 and 192.168.3.0/24 advertised by routers r2 and r3, respectively, are added to the routing table of router r1.

Step 3. Open router r2 terminal and type the following command to show the routing table.

```
show ip route
```

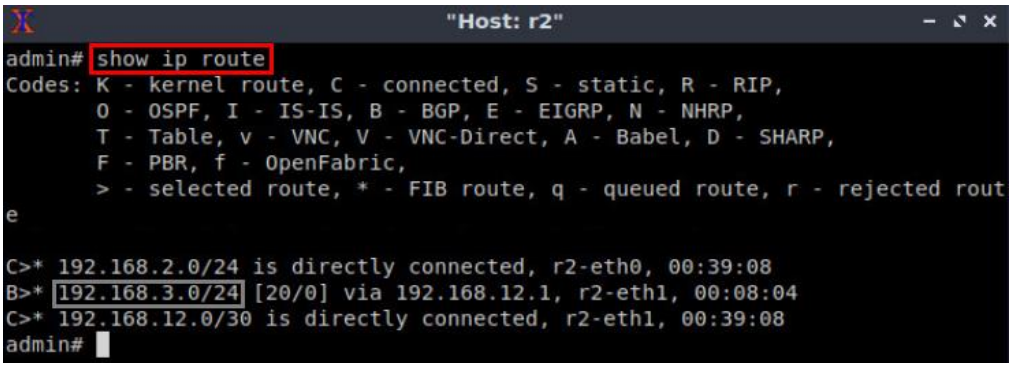

Figure 67. Showing the routing table router r2.

Consider Figure 67. The network 192.168.3.0/24 is added to the routing table of router r2, and it is reachable via 192.168.12.1 (router r1).

Step 4. Open host h1 terminal and type the following command to test the connectivity with host h2.

```
ping 192.168.3.10
```

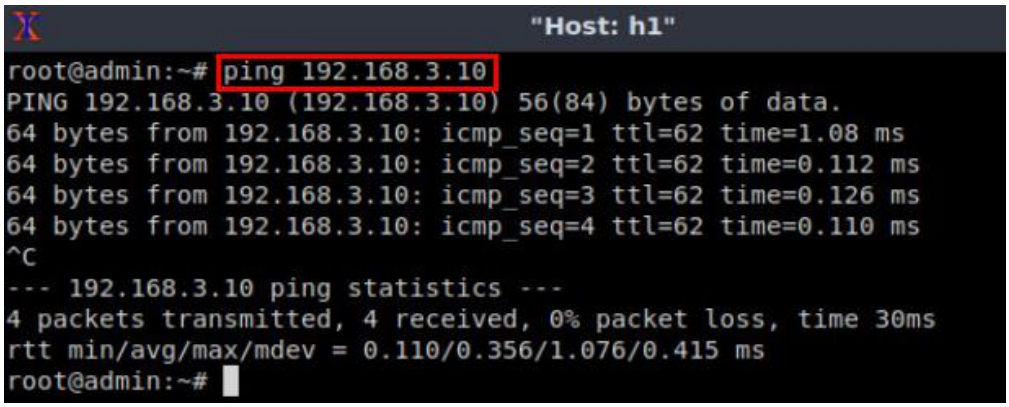

Figure 68. Pinging host h2 from host h1.

Consider Figure 68. The result of pinging host h2 from h1 shows a successful connectivity. Thus, BGP is successfully configured and integrated between legacy and SDN networks.

This concludes Lab 8. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. A. Tanenbaum, D. Wetherall, "*Computer networks*", 5th Edition, Pearson, 2012.
2. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach"*. Morgan Kaufmann, 2016.
3. P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.C. Wang, J. Bi. "*Seamless interworking of SDN and IP*". In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 475-476, 2013.
4. ONOS project, "*SDN-IP*", [Online]. Available: https://wiki.onosproject.org/display/ONOS/SDN-IP.

SOFTWARE DEFINED NETWORKING

Lab 9: Configuring Virtual Private LAN Service (VPLS)

Document Version: 08-18-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

The following lab presents Virtual Private Local Area Network Service (VPLS). A VPLS emulates a Local Area Network (LAN) and provides multipoint broadcast over layer 2 circuits between multiple endpoints. This service enables remote sites to share an Ethernet broadcast domain by connecting sites through pseudowires. In this lab, the user will configure VPLS in a simple topology using Open Flow switches.

Objectives

By the end of this lab, the user should be able to:

1. Understand the operation of VPLS.
2. Enable Open Flow switches to enable VPLS operation.
3. Use Open Network Operating System (ONOS) to perform VPLS configuration.
4. Verify end-to-end connectivity between the end-hosts attached in the same VPLS.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1. Credentials to access the Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Configuring VPLS.
4. Section 4: Verifying connectivity between end-hosts.

1 Introduction

Networks that primarily use the data link layer to interconnect devices are referred to as layer two (L2) networks. A data link layer device operates in the second layer of the Open Systems Interconnection (OSI) model. For example, a typical L2 network is an Ethernet network which englobes endpoint devices such as servers, printers, and computers are

interconnected using one or more Ethernet switches. The Ethernet switches enable L2 communication by forwarding Ethernet frames within the network.

A virtual private local area network service (VPLS) works as a L2 virtual private network (VPN) service that extends two or more remote customer networks known as VPLS sites. This service can be provided over intermediate networks often referred to as a provider network. A VPLS is delivered in a transparent manner, this implies that the remote customer sites seem to be connected in the same LAN. In summary, the VPLS enables L2 communications between customer networks through intermediate networks[1].

1.1 VPLS architecture

Consider Figure 1. A VPLS emulates a LAN and provides L2 functionalities by acting as an emulated Ethernet switch mainly within a wide area network (WAN). Once a VPLS instance is created, its main functionality is to interconnect two or more remote customer sites. Additionally, VPLS supports Virtual Local Area Networks (VLAN) and Multiprotocol Label Switching (MPLS) encapsulations. When VPLS instance is configured, it creates a full mesh of pseudowires between the Provider's Edge (PE) routers that are participating in the VPLS instance. The PE routers have the capability of replicating and forwarding broadcast and multicast frames therefore, the emulated switch has all the characteristics of an Ethernet switch[2].

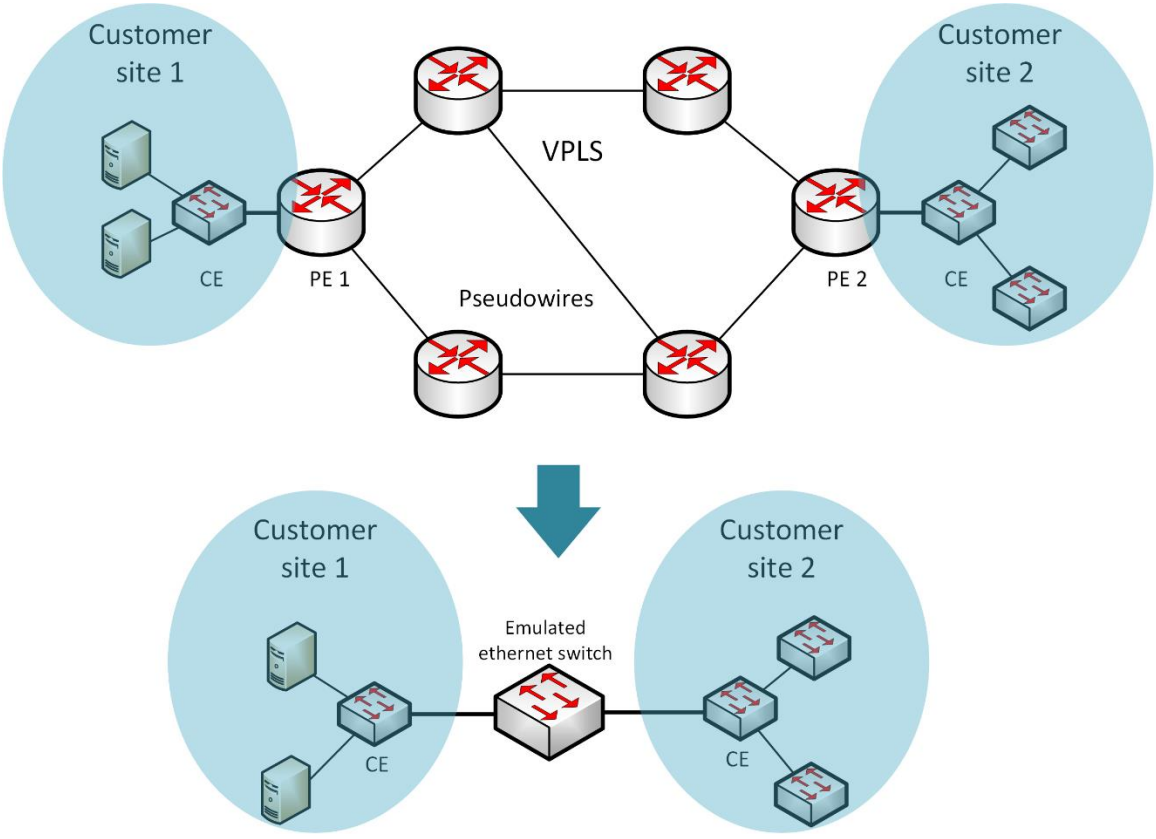

Figure 1. VPLS architecture.

VPLS is supported by an ONOS application that provides multi-point broadcast L2 circuits between multiple endpoints in an OpenFlow network. Additionally, it supports

encapsulations such as VLAN and MPLS. The application connects into overlay broadcast networks hosts connected to the OpenFlow data plane. To establish VPLS connectivity between two or more hosts, the following requirements should be fulfilled.

- At least one VPLS must be defined.
- At least two end-hosts' interfaces have to be configured in the ONOS interface configuration.
- At least two interfaces must be associated to the same VPLS.

Once the previous conditions are satisfied, hosts attached to the VPLS will be able to send and receive broadcast traffic such as the Address Resolution Protocol (ARP) request messages. This is needed to make sure that all hosts get discovered properly, before establishing unicast communication.

2 Lab topology

Consider Figure 4. The topology consists of four end-hosts, two OpenFlow switches and a controller. This topology supposes a scenario where two customers have two remote sites. Host h1 and host h3 belong to VPLS1, similarly host h2 and host h4 are in VPLS2.

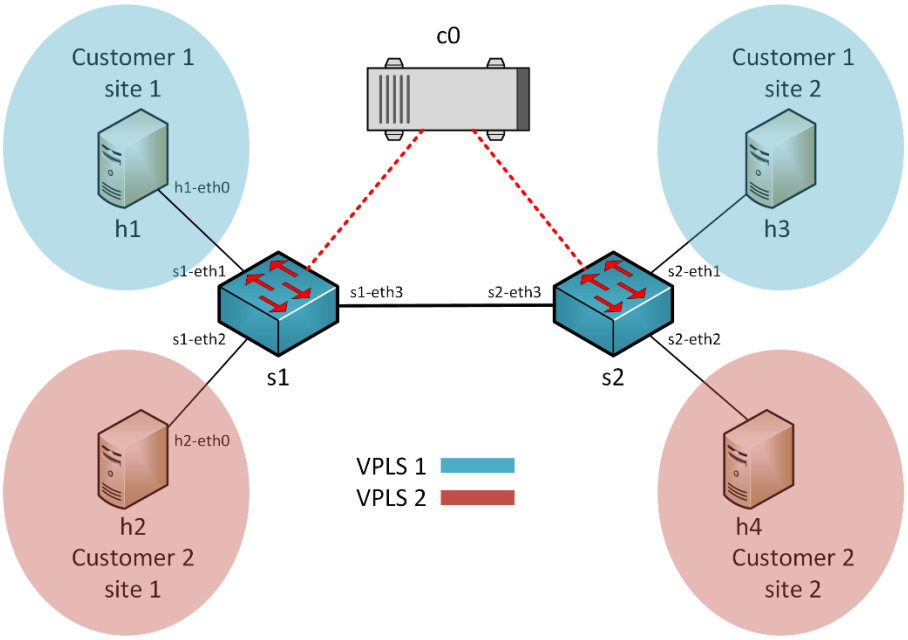

Figure 2. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet
c0	n/a	127.0.0.1	/32
h1	h1-eth0	10.0.0.1	/8
h2	h2-eth0	10.0.0.2	/8
h3	h3-eth0	10.0.0.3	/8
h4	h4-eth0	10.0.0.4	/8

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), saving topologies, and exporting layer 2 models.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

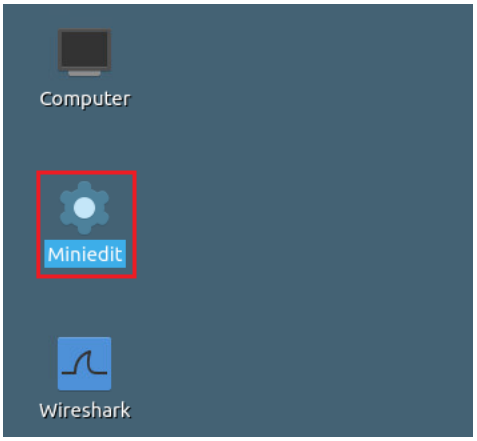

Figure 3. MiniEdit shortcut.

Step 2. Select the file *lab9.mn* and click on *Open*.

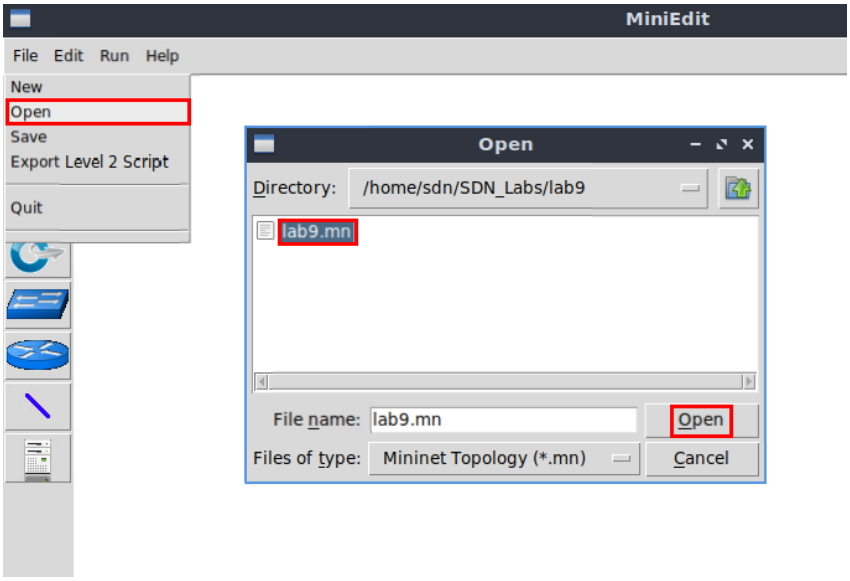

Figure 4. Opening the topology.

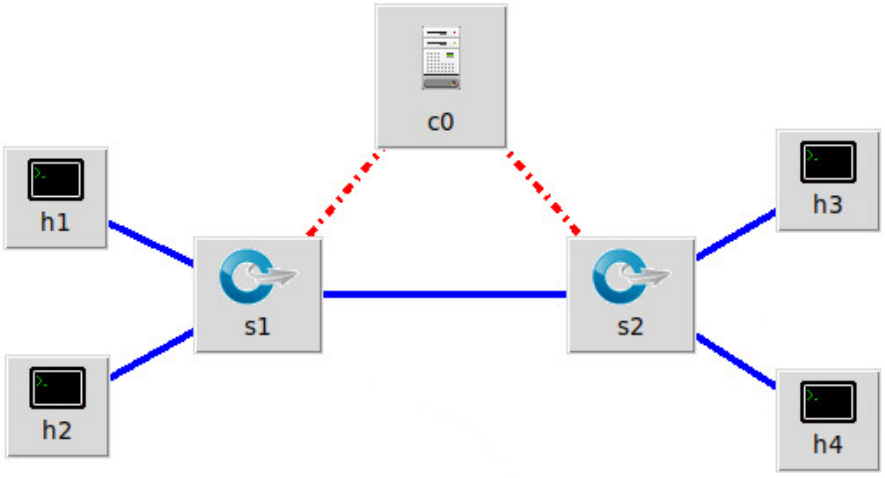

Figure 5. MiniEdit's topology.

2.3 Starting the ONOS controller

At this point the topology is loaded however, the interfaces are not configured. In order to assign IP addresses to the devices' interfaces, you will execute a script that loads the configuration to the routers and end devices.

Step 1. Click on the icon below to open the Linux terminal.

Figure 6. Opening the Linux terminal.

Step 2. Click on the Linux terminal and navigate into the *SDN_Labs/lab9* directory by issuing the following command. This folder contains a configuration file and the script

responsible for loading the configuration. The configuration file will assign the IP addresses to the routers' interfaces. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab9
```

Figure 7. Entering the *SDN_Labs/lab9* directory.

Step 3. Run ONOS and wait 1.5 minutes.

```
./run_onos.sh
```

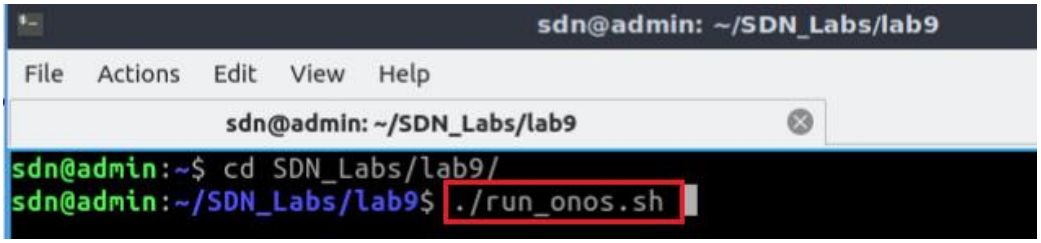

Figure 8. Executing the shell script to load the configuration.

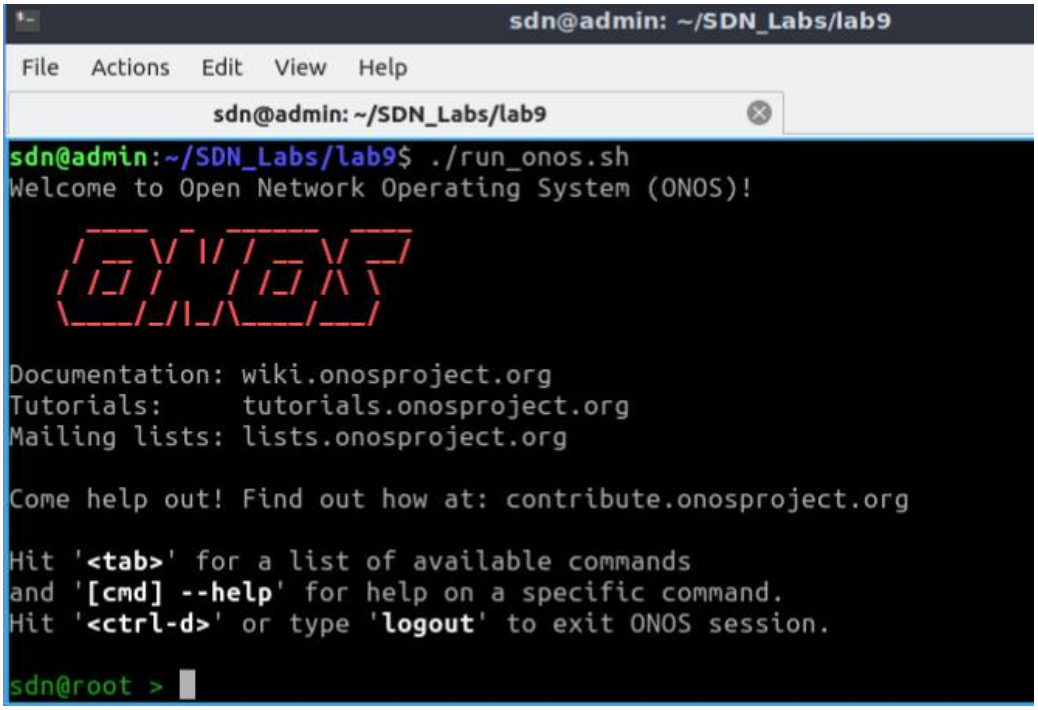

Figure 9. Starting the ONOS controller.

2.4 Running the emulation

In this section, you will run the emulation and check the links and interfaces that connect the devices in the given topology.

Step 1. At this point the interfaces of host h1 and host h2 are configured. To proceed with the emulation, click on the *Run* button located in lower left-hand side.

Figure 10. Starting the emulation.

Step 2. Open Mininet's CLI by selecting the terminal shown below:

Figure 11. Navigating into Mininet's CLI.

Step 3. Issue the following command to display the interface names and connections.

```
links
```

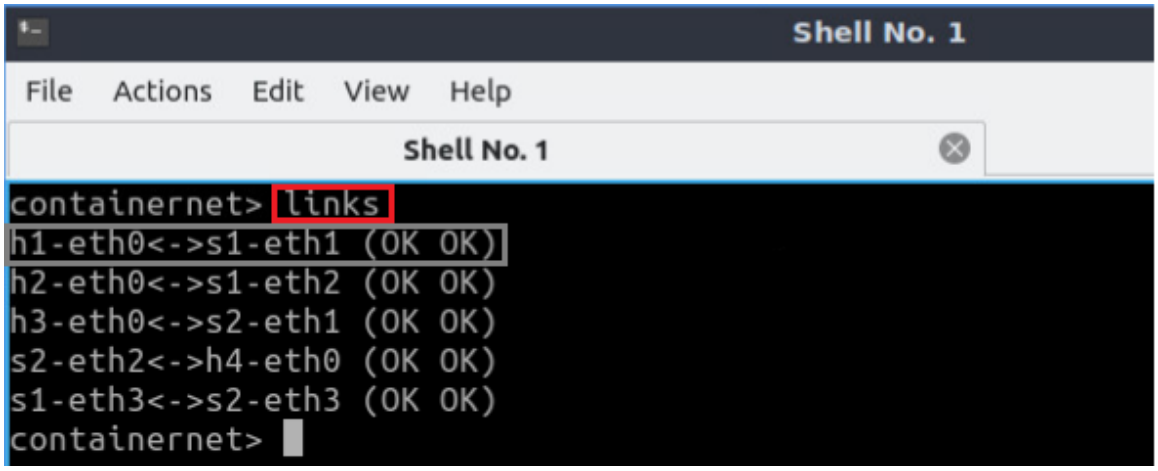

Figure 12. Displaying network interfaces.

The figure above displays the links of the devices within the network. Inside the grey box it is shown that h1 is connected to switch s1 via the interface h1-eth0 and, switch s1 is connected to host h1 through the interface s1-eth1 (i.e., *h1-eth0<->s1-eth1*).

2.5 Verifying the configuration

In this section, the user will verify the IP addresses listed in Table 2 and perform a connectivity test.

Step 1. Hold right-click on host h1 and select *Terminal*. This opens the terminal of host h1 and allows the execution of commands in that host.

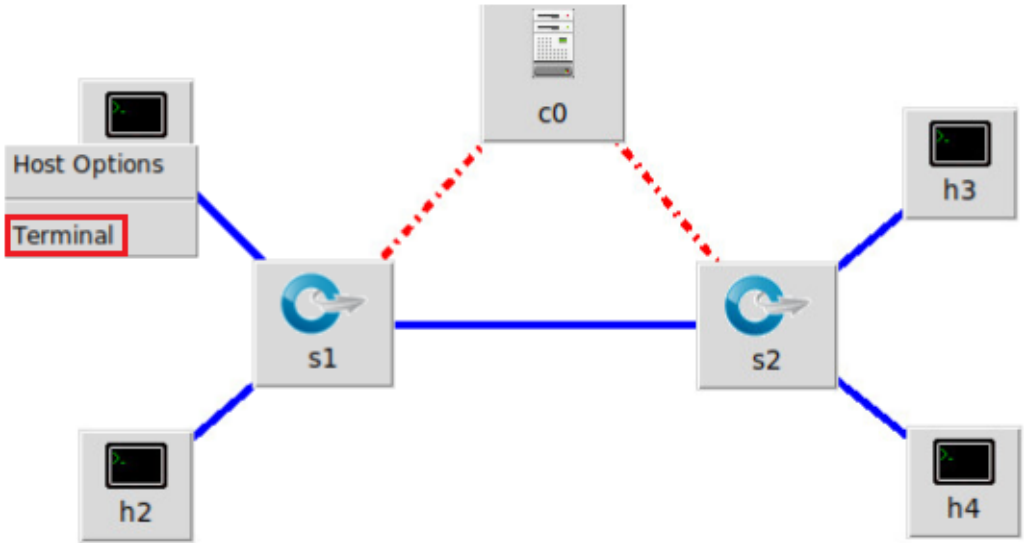

Figure 13. Opening a terminal in host h1.

Step 2. In host h1's terminal, type the command shown below to verify that the IP address was assigned successfully. The user will corroborate that host h1 has two interfaces, *h1-eth0* configured with the IP address 10.0.0.1 and the subnet mask 255.0.0.0 and the loopback interface.

```
ifconfig
```

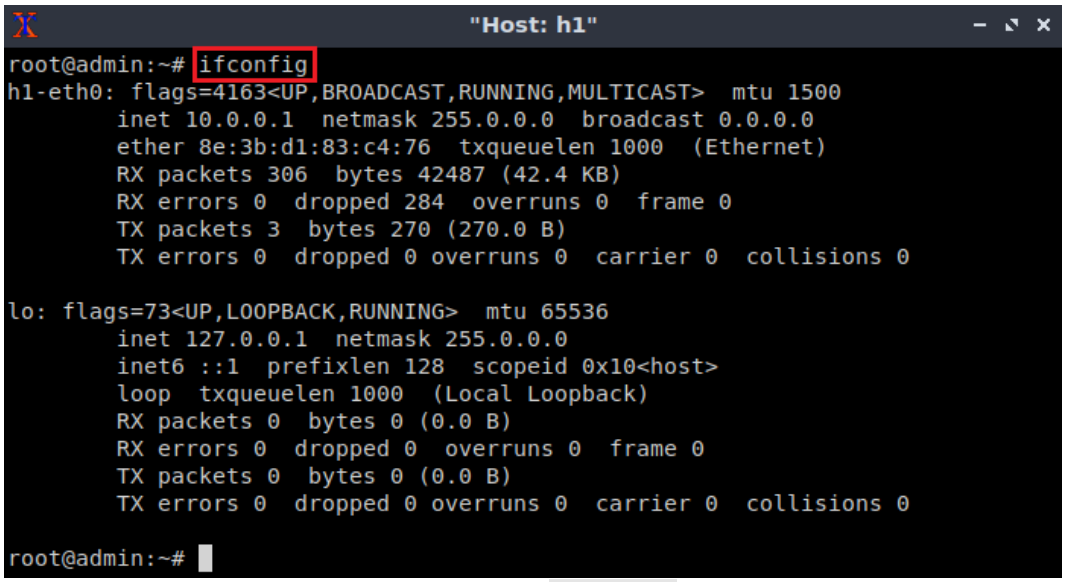

Figure 14. Output of the `ifconfig` command.

Step 3. Follow the previous step to verify the IP addresses in the other hosts are according to Table 2.

Step 4. In host h1's terminal, perform a connectivity test between h1 and h3 by typing the following command:

```
ping 10.0.0.3
```

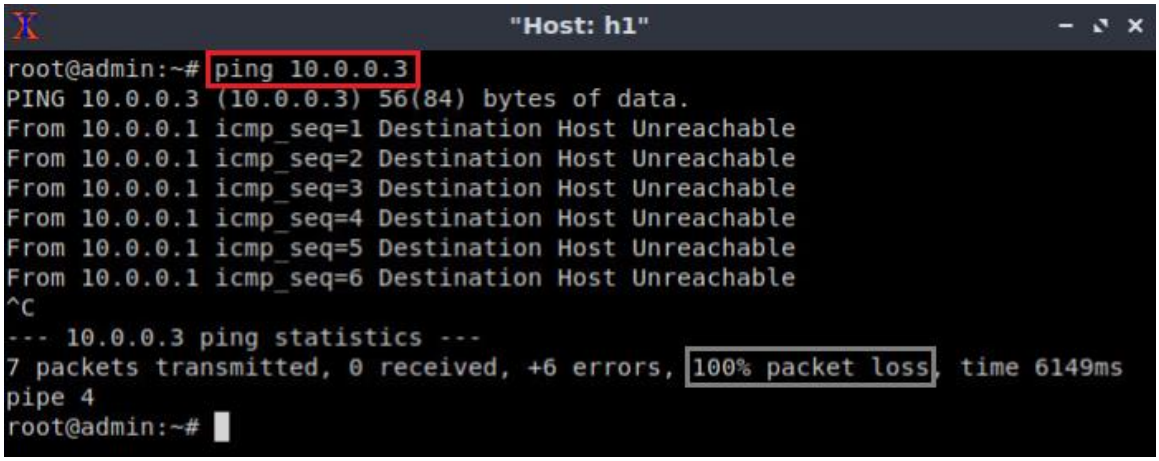

Figure 15. Performing a connectivity test in host h1.

Step 5. Press `Ctrl+c` to stop the test.

The figure above shows an unsuccessful connectivity test and as a result the test had 100% packet loss.

3 Configuring VPLS

In this section the user will configure VPLS by enabling the OpenFlow and the VPLS application in ONOS. Then, the user will define two VPLS' in order to establish a connection between the two customers' sites.

3.1 Enabling OpenFlow and VPLS applications

Step 1. Select the ONOS CLI by clicking in the Linux terminal as shown in the figure below:

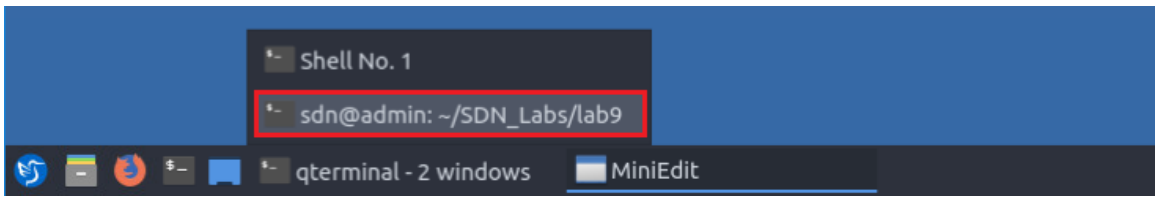

Figure 16. Navigating into the ONOS CLI.

Step 2. Activate the OpenFlow application by issuing the following command:

```
app activate org.onosproject.openflow
```

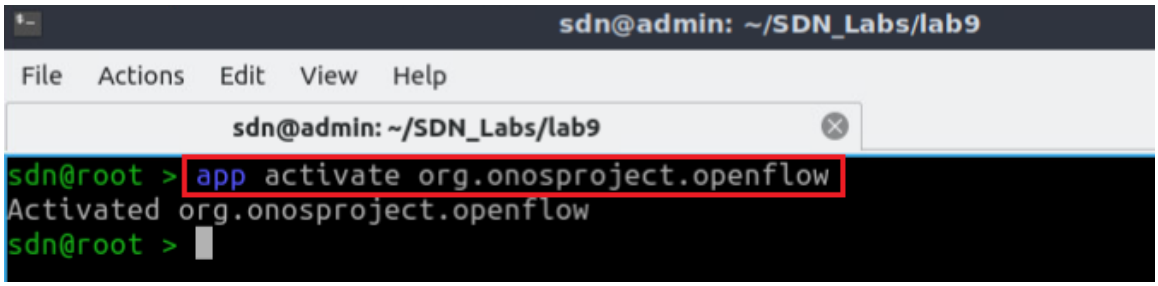

Figure 17. Activating the OpenFlow application.

Step 3. Activate the VPLS application by typing the command below:

```
app activate org.onosproject.vpls
```

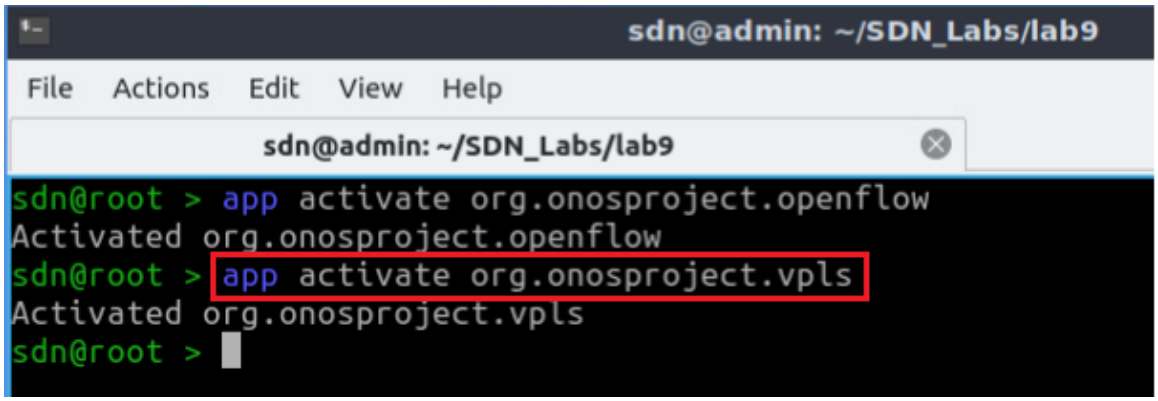

Figure 18. Activating the VPLS application.

3.2 Displaying host information

Step 1. In order to list the information regarding the hosts within the topology, type the command shown below. Due to the connectivity test performed in section 2.5, the output will show the information of host h1 (10.0.0.1) however, the information about the other hosts is missing.

```
hosts
```

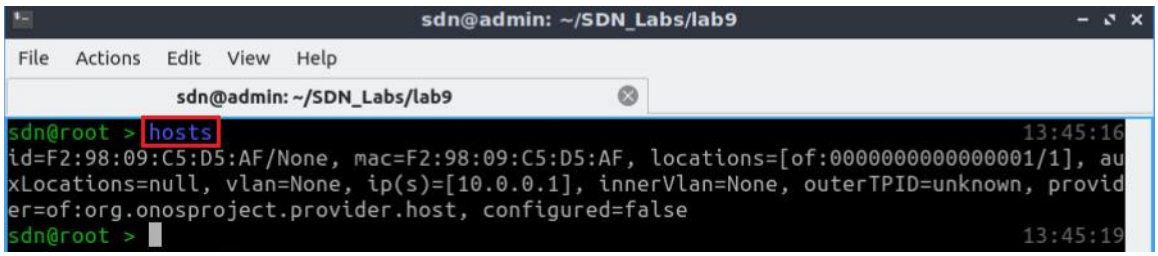

Figure 19. Displaying hosts information.

Step 2. Open Mininet's CLI by selecting the terminal shown below:

Figure 20. Navigating into Mininet's CLI.

Step 3. In the CLI, type the command shown below and wait until the test finishes. The results will show an unsuccessful (X X X) connectivity test among all end-hosts.

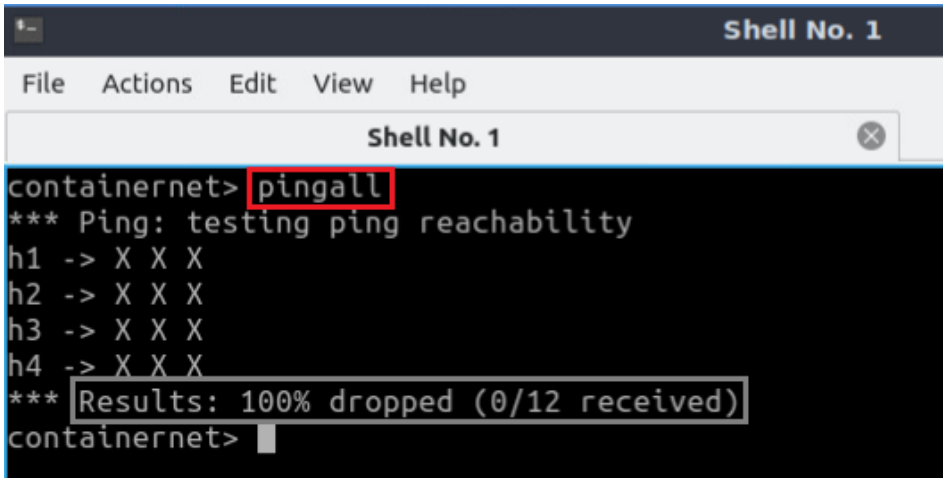

Figure 21. Performing a connectivity test between all the participants.

Step 4. Navigate back to the ONOS CLI by clicking in the Linux terminal as shown in the figure below:

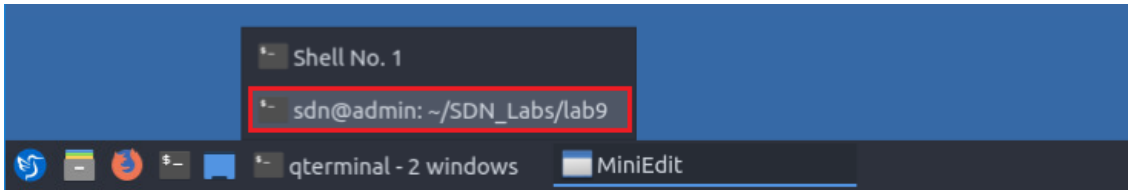

Figure 22. Navigating into ONOS CLI.

Step 5. Type the command below to display the hosts' information. The user will observe the location and the IP address of each host. The location is related to the OpenFlow switch ports. For example, the location of:0000000000000001/2 specifies an the OpenFlow switch 1 and the port number 2. In this port the user will verify that host h2 (10.0.0.2) is associated with that port.

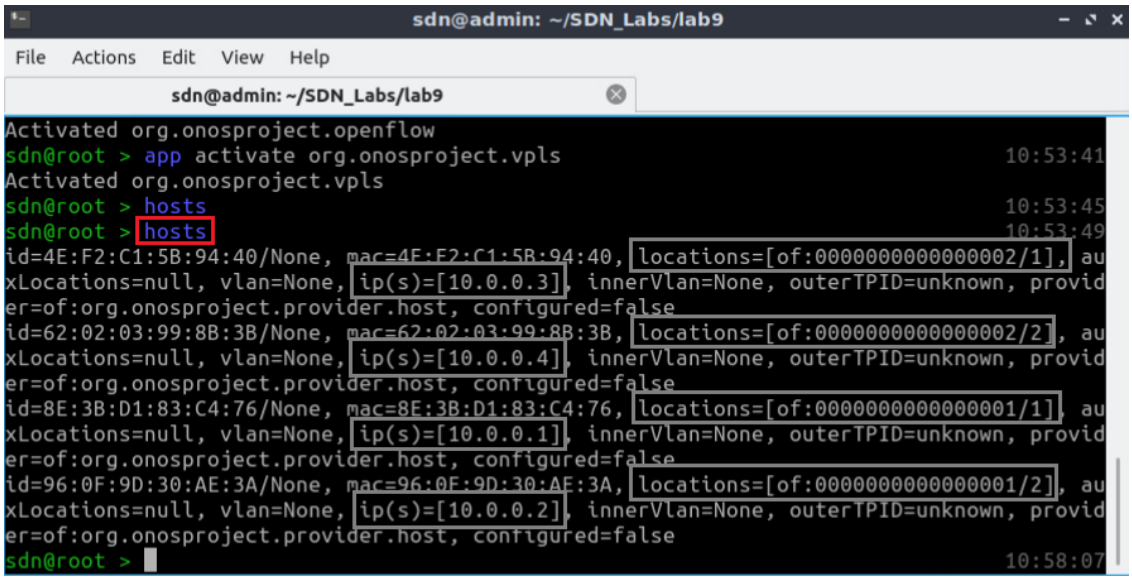

Figure 23. Displaying host information.

Table 3 summarizes the relevant information regarding each host such as the device's name, the VPLS tag that will be assigned in further steps, the IP addresses, and the OpenFlow switch datapath identifier (DPID).

Table 3. Host information.

Device	VPLS	IP Address	OF Switch DPID	VLAN
h1	VPLS1	10.0.0.1	0000000000000001/1	0
h2	VPLS2	10.0.0.2	0000000000000001/2	0
h3	VPLS1	10.0.0.3	0000000000000002/1	0
h4	VPLS2	10.0.0.4	0000000000000002/2	0

3.3 Associating OpenFlow interfaces to the end-hosts

Step 1. Considering the information contained in table 3, associate h1 to its OpenFlow port by typing the following command:

```
interface-add of:0000000000000001/1 h1
```

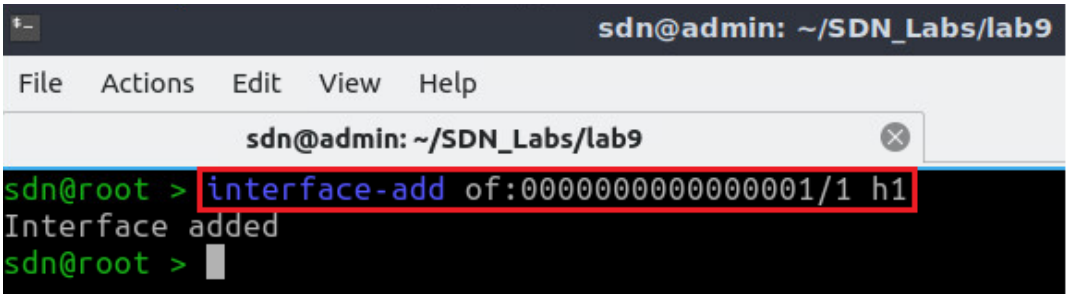

Figure 24. Adding an interface.

Step 2. Proceed similarly to associate the remaining interfaces. Those steps are summarized in the figure below:

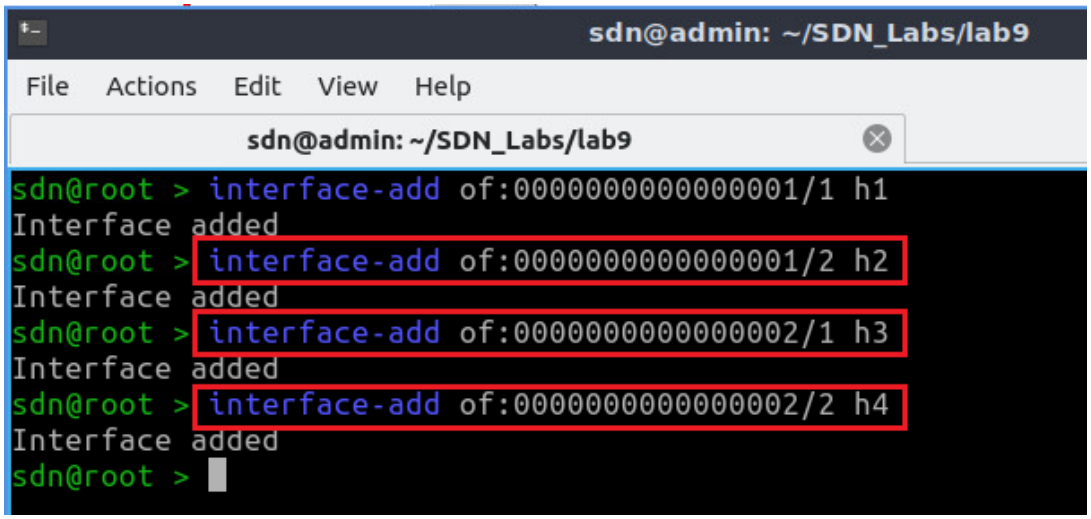

Figure 25. Adding interfaces.

3.4 Creating a VPLS

Step 1. To create a new VPLS, type the command below according to the information displayed in table 3.

```
vpls create VPLS1
```

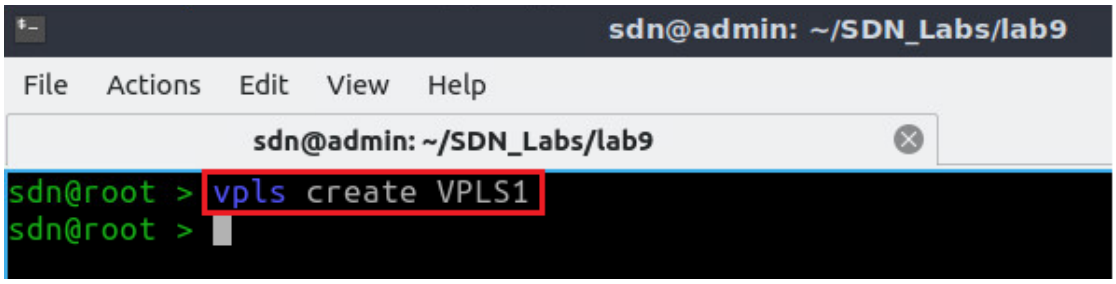

Figure 26. Creating VPLS1.

Step 2. Similarly, create another VPLS by typing the following command:

```
vpls create VPLS2
```

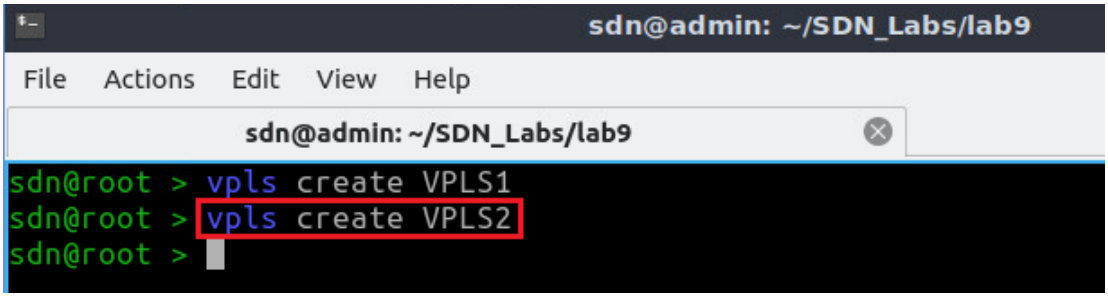

Figure 27. Creating VPLS2.

Step 3. To verify that the VPLS' were created correctly, type the following command:

```
vpls list
```

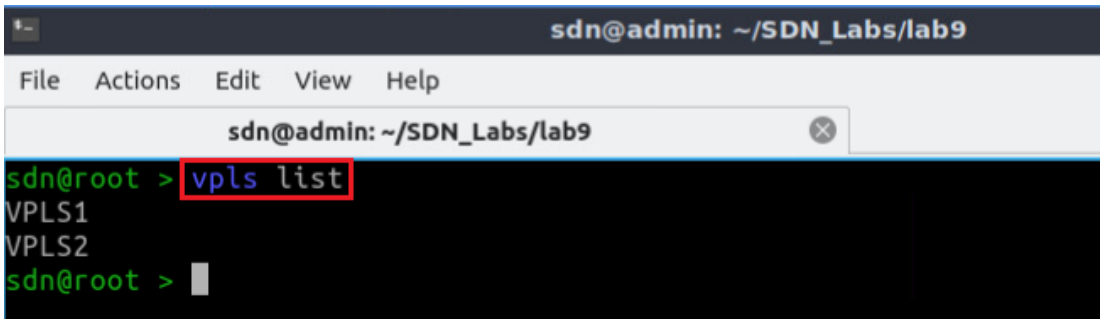

Figure 28. Listing the created VPLS'.

3.5 Adding interfaces to an existing VPLS

Step 1. To add host h1 to VPLS1 type the following command:

```
vpls add-if VPLS1 h1
```

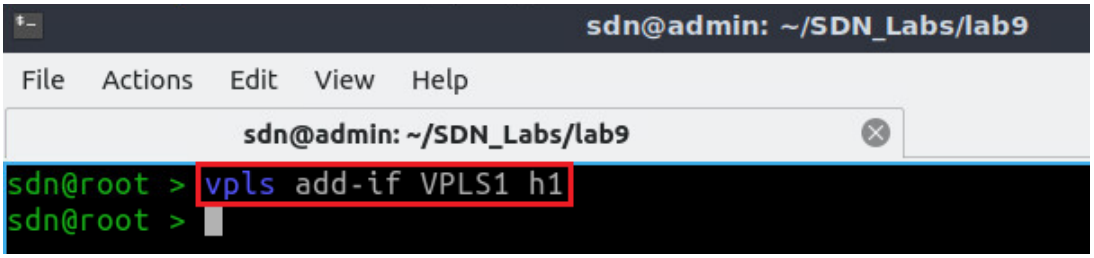

Figure 29. Assigning h1 to VPLS1.

Step 2. Add host h3 to VPLS1 by typing the command below. Now, host h1 and host h3 are in the same VPLS.

```
vpls add-if VPLS1 h3
```

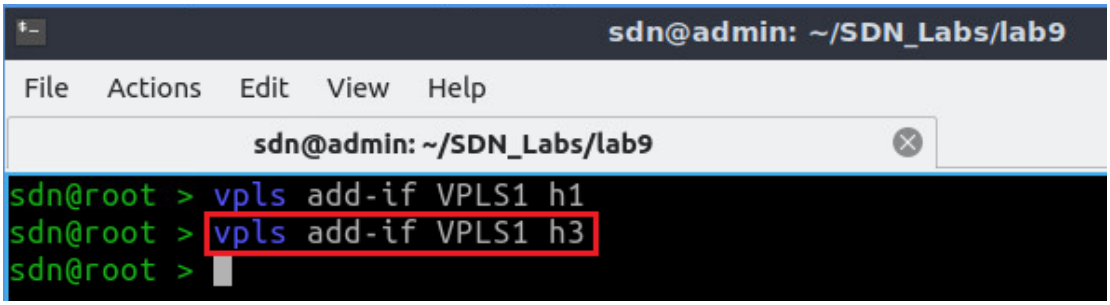

Figure 30. Assigning h3 to VPLS1.

Step 3. To add host h2 to VPLS2 type the following command:

```
vpls add-if VPLS2 h2
```

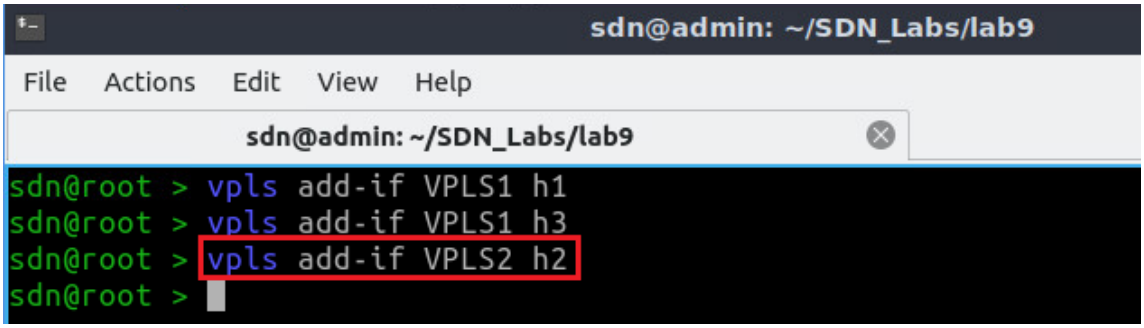

Figure 31. Assigning h2 to VPLS2.

Step 3. Similarly, add host h4 to VPLS2 by issuing the following command:

```
vpls add-if VPLS2 h4
```

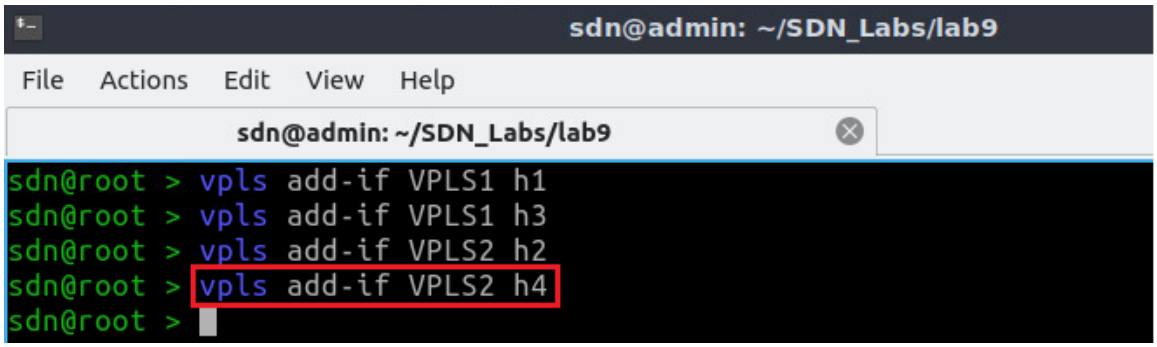

Figure 32. Assigning h4 to VPLS2.

Step 4. To verify if the configuration was applied correctly issue the command below:

```
vpls show
```

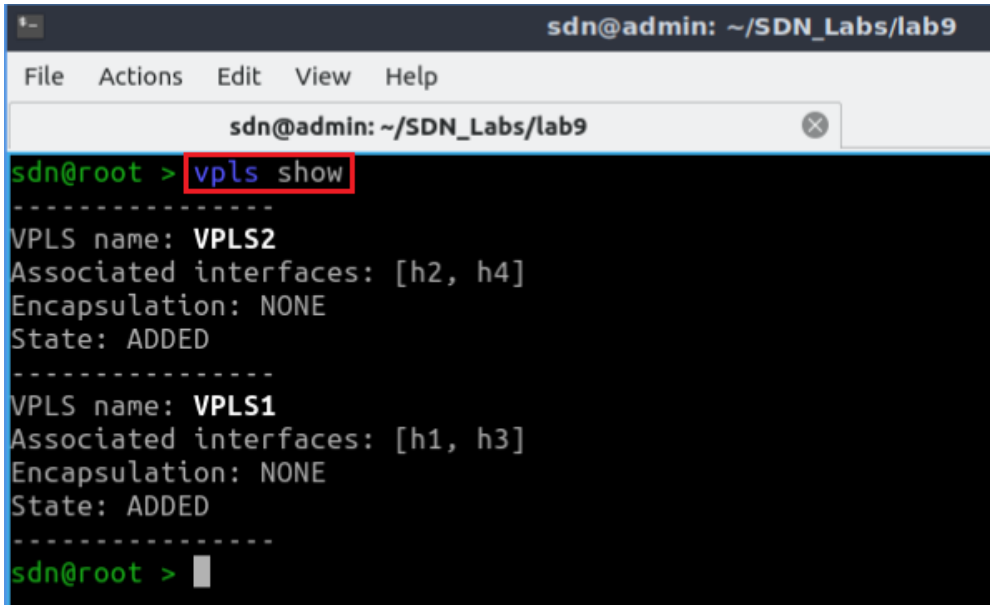

Figure 33. Verifying VPLS configuration.

The output of the figure above shows that VPLS1 has h1 and h3 as it associated interfaces. No encapsulation is used, and the state indicates that the interfaces were added

successfully. VPLS2 associated interfaces are h2 and h4 then, the configuration is the same as VPLS1.

4 Verifying connectivity between end-hosts

As soon as two or more interfaces are added to the same VPLS, intents to manage broadcast will be installed and the participants will be able to communicate among themselves.

Step 1. To verify the connectivity between host h1 and host h3, in host h1's CLI type the following command:

```
ping 10.0.0.3
```

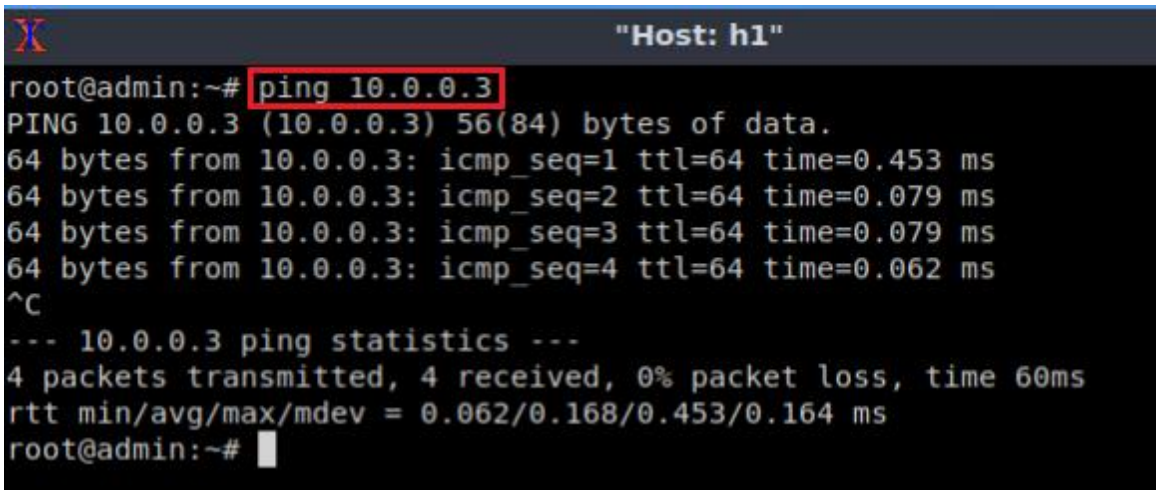

Figure 34. Connectivity test between h1 and h3

Step 2. Press `Ctrl+C` to stop the test. The figure above shows a successful connectivity test. This was possible because host h1 and host h3 are in the same VPLS (VPLS1).

Step 3. Now, in host h1 try to ping host h4 by issuing the following command:

```
ping 10.0.0.4
```

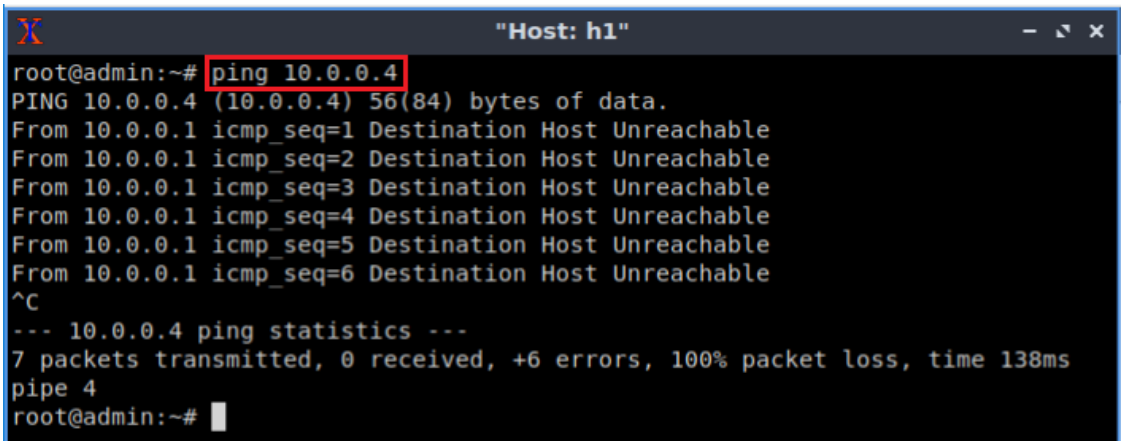

Figure 35. Connectivity test between h1 and h4.

Step 4. Press `Ctrl+c` to stop the test. The figure above shows an unsuccessful connectivity test since h1 and h4 are associated to different VPLS'.

Step 5. In host h2's terminal, perform a connectivity test between host h2 and host h4 by typing the command shown below:

```
ping 10.0.0.4
```

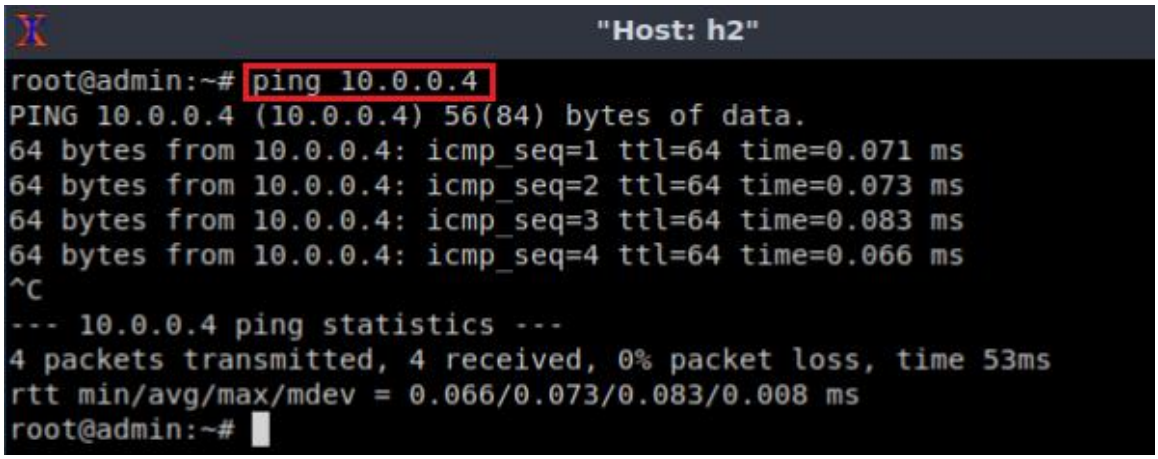

Figure 36. Connectivity test between h2 and h4.

Step 6. Press `Ctrl+c` to stop the test. The figure above shows a successful connectivity test. This was possible because host h2 and host h4 are in the same VPLS (VPLS2).

Step 7. In host h2's terminal, perform a connectivity test between host h2 and host h3 by typing the command shown below:

```
ping 10.0.0.3
```

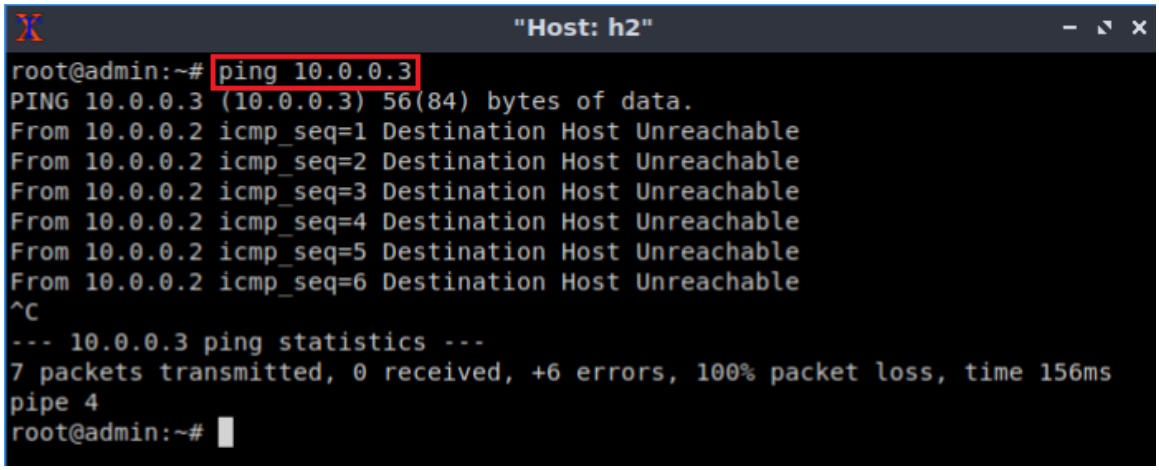

Figure 37. Connectivity test between h2 and h3.

Step 8. Press `Ctrl+c` to stop the test. The figure above shows an unsuccessful connectivity test since h2 and h3 are associated to different VPLS.

This concludes Lab 8. Stop the emulation and then exit out of MiniEdit and the Linux terminal.

References

1. Hasan, Safaa S. "*Extending VPLS support for CE lag multi-homing",* U.S. Patent 8,705,526, 2014.
2. De Ghein, Luc, "*MPLS Fundamentals"*, Cisco Press, CCIE No. 1897, 2016.
3. A. Tanenbaum, D. Wetherall, "*Computer networks*", 5th Edition, Pearson, 2012.
4. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach"*. Morgan Kaufmann, 2016.
5. P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.C. Wang, J. Bi. "*Seamless interworking of SDN and IP*", In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (pp. 475-476). 2013.
6. ONOS project. "*SDN-IP*". [Online]. Available: https://wiki.onosproject.org/display/ONOS/SDN-IP.