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

The switching functions are traditionally classified into three categories. Each category is capable to communicate with its peers in a vertical and horizontal way. Therefore, it was more practice to represent each of these categories as a layer or *plane*. The communication between peers happens in the same plane, and cross-category communication occurs in a dimension between planes[11].

Consider Figure 1. The *data plane* is responsible of managing the major part of the packets handled by the switch. The ports in the data plane manage the reception and transmission of packets and a forwarding table is responsible to set the logic. The data plane handles operations such as packet buffering, packet scheduling, header modification, and forwarding. For example, if the header of a data packet contains information that matches the forwarding table, the data plane may perform some header field modification in order to be forwarded offloading the intervention of the other two planes. Sometimes some packets are handled by the control plane due to its intrinsic information is not allocated into the forwarding table. This may happen when there is a control protocol that requires a processing in the *control plane*.

The *control plane* has assigned several roles. A key functionality is to store the information in the forwarding table, so that, the data plane can independently process the highest percentage of the traffic as possible. The control plane also handles different control protocols that may modify the forwarding table, depending on the configuration and type of switch. These control protocols actively manage the topology of the network.

The third plane depicted in Figure 1 is the *management plane*. This plane allows network administrators to configure and to monitor the switch. Network administrators are interested on extracting information from control and data plane as appropriate in order to perform actions that may affect those two planes. The interface used by network administrators is usually based on network management system to communicate with the management plane in a switch[11].

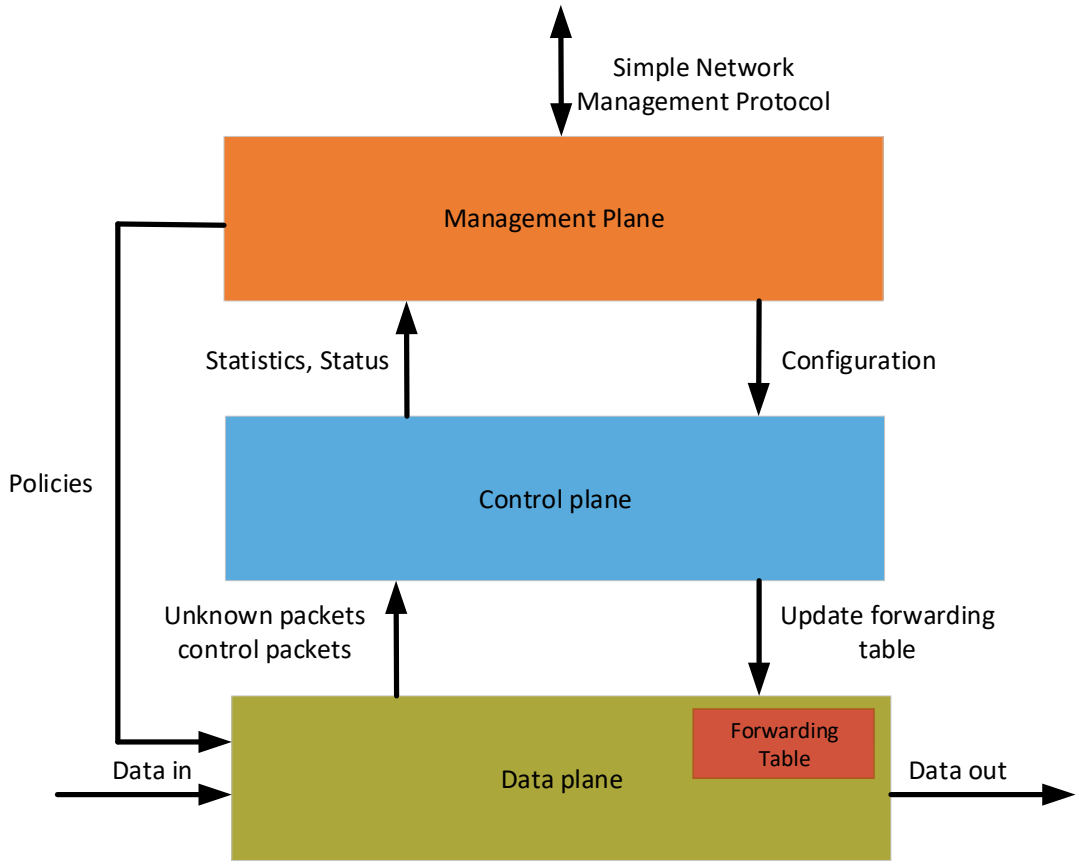

Figure 1. Roles of the control, data and management planes[3].

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.

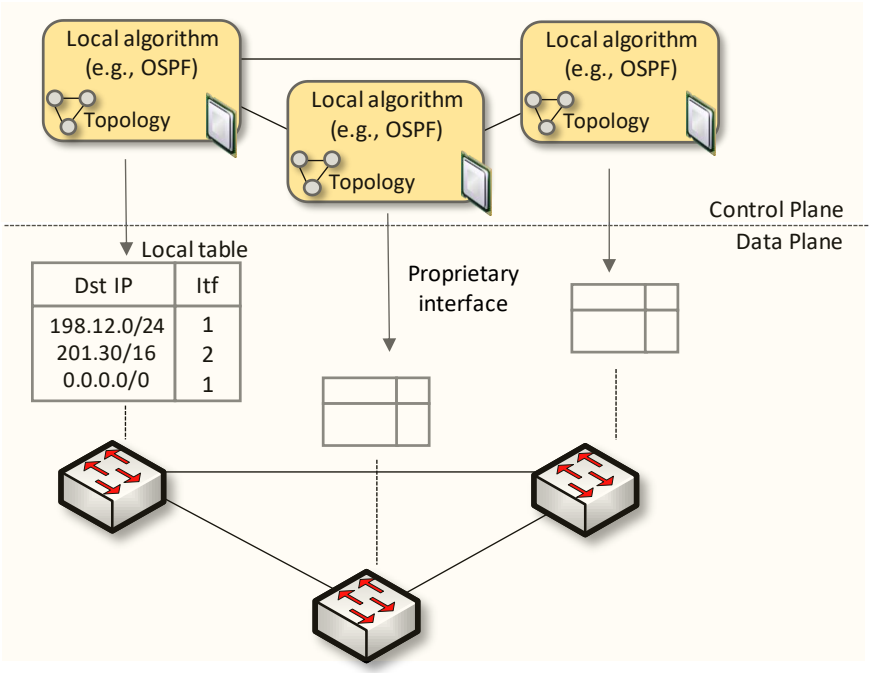

Figure 2. The control plane and the data plane are coupled in the same legacy device.

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].

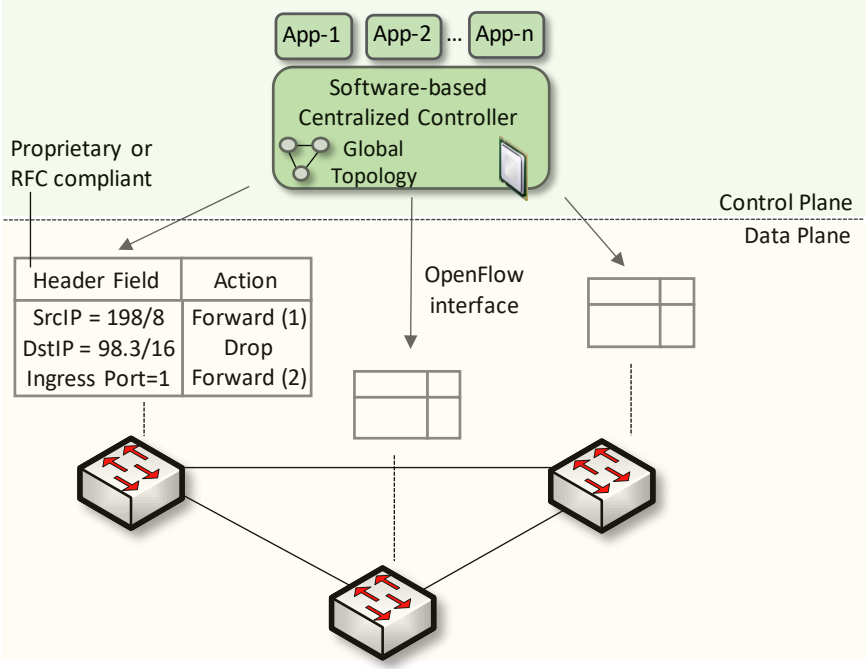

Figure 3. The control plane is embedded in a centralized server and it is decoupled from data plane devices.

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.

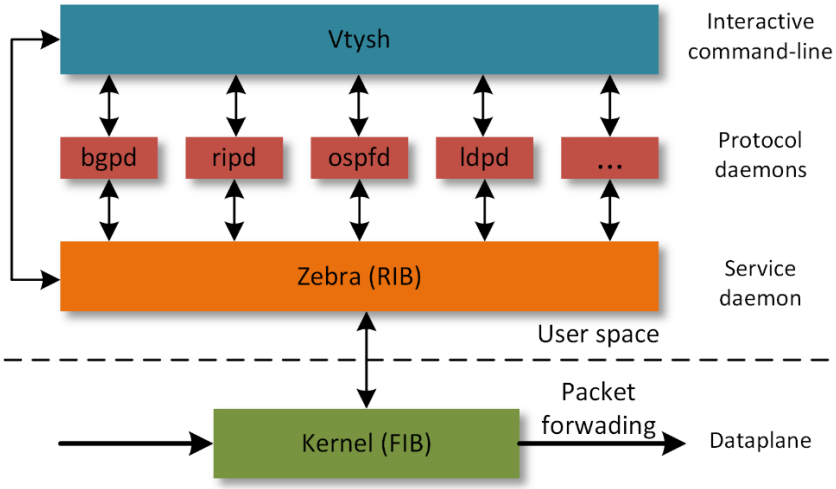

Figure 4. FRR architecture.

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.

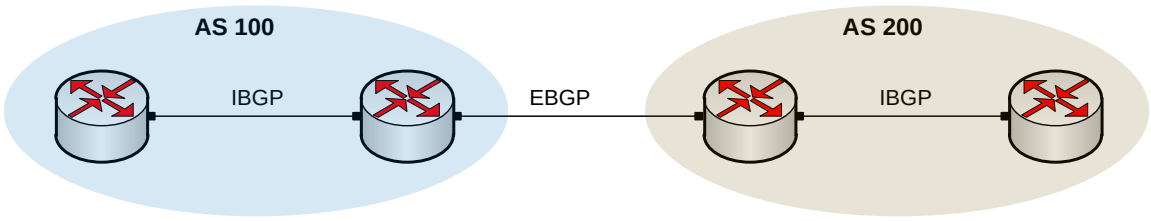

Figure 5. Routers that exchange information within the same AS use IBGP, while routers that exchange information between different ASes use EBGP.

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.

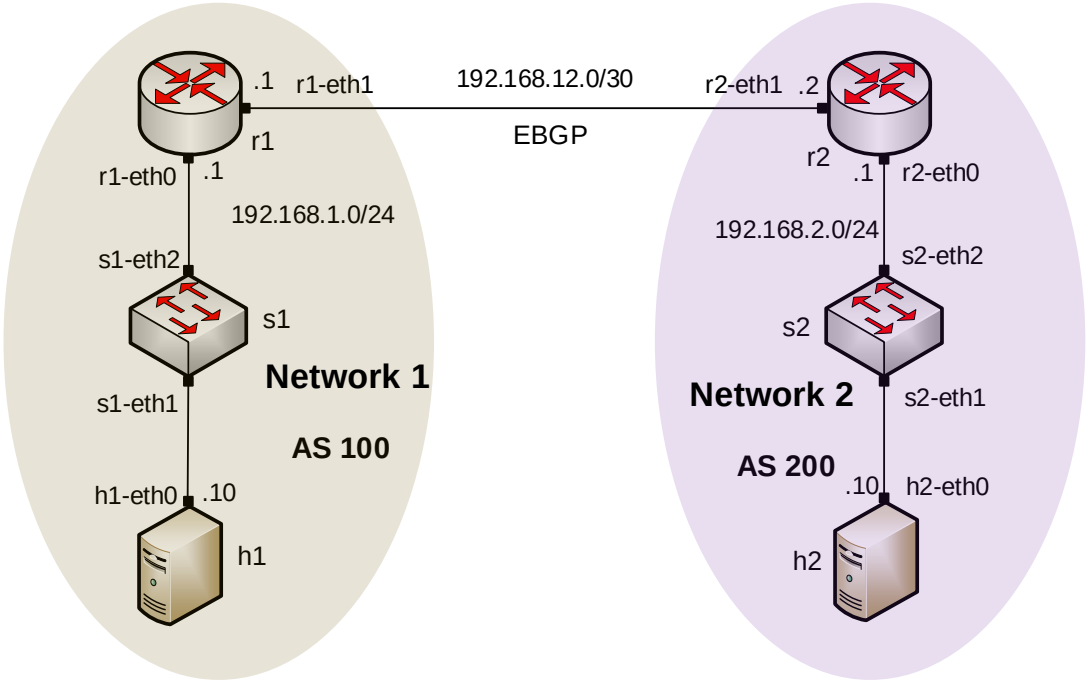

Figure 6. Lab topology.

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`.

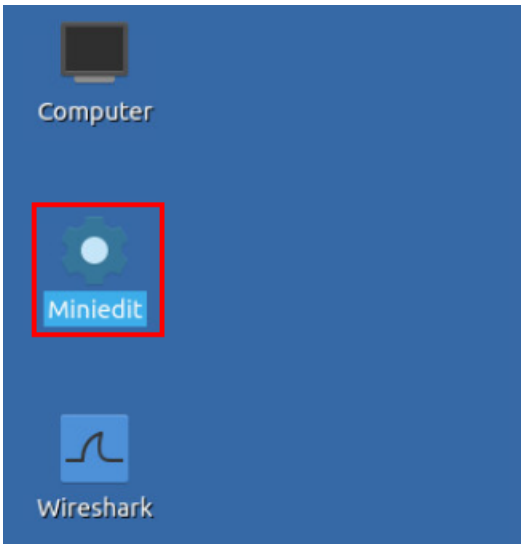

Figure 7. MiniEdit shortcut.

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*.

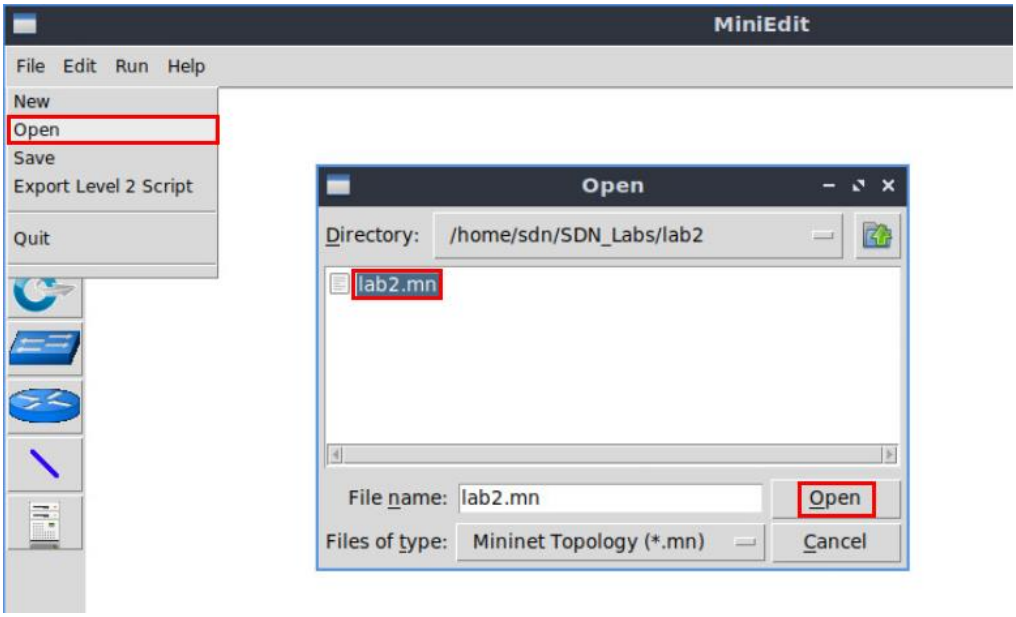

Figure 8. MiniEdit's open dialog.

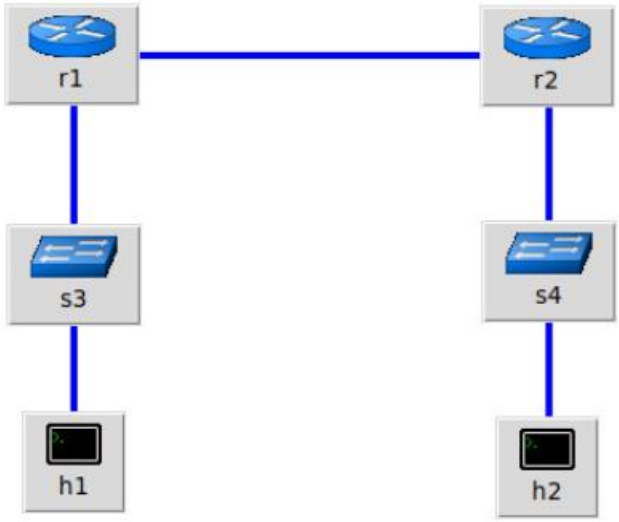

Figure 9. Mininet's topology.

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
```

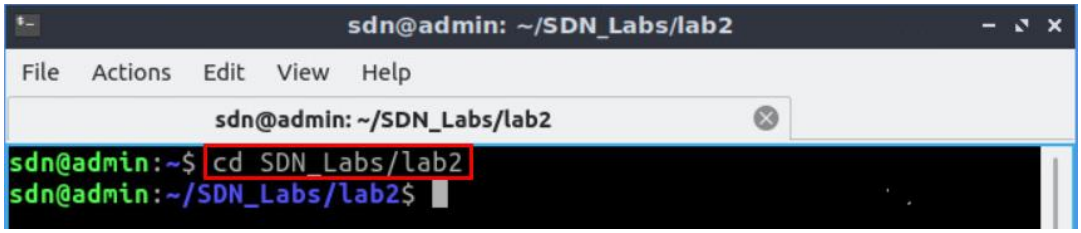

Figure 11. Entering the *SDN_Labs/lab2* directory.

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
```

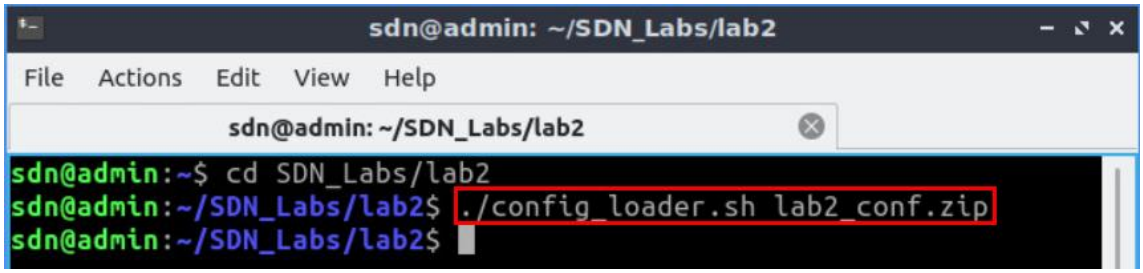

Figure 12. Executing the shell script to load the configuration.

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

```
exit
```

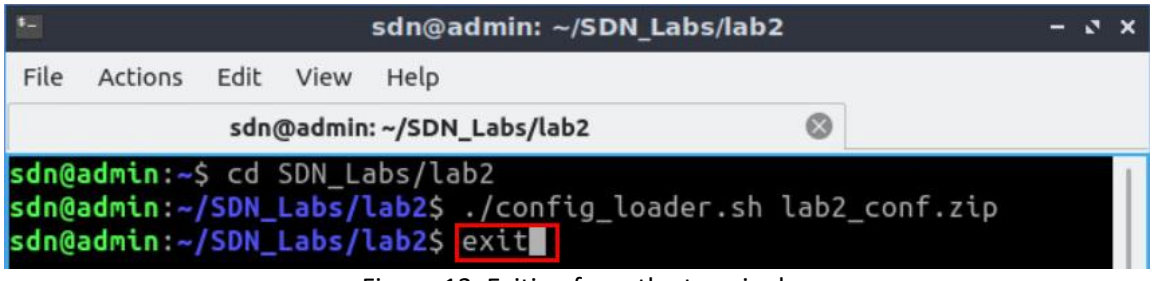

Figure 13. Exiting from the terminal.

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
```

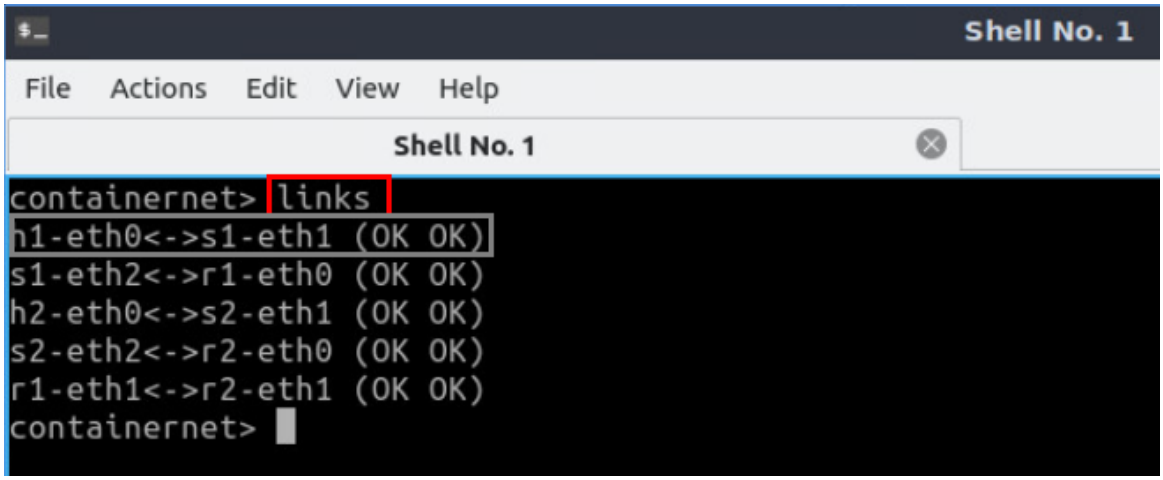

Figure 15. Displaying network interfaces.

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.

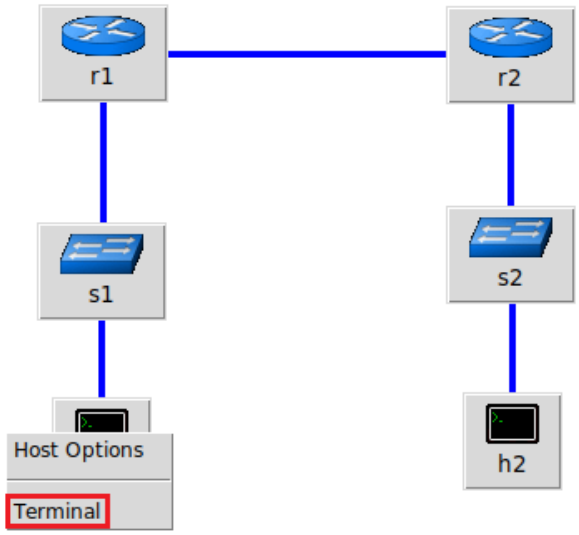

Figure 16. Opening a terminal on host h1.

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
```

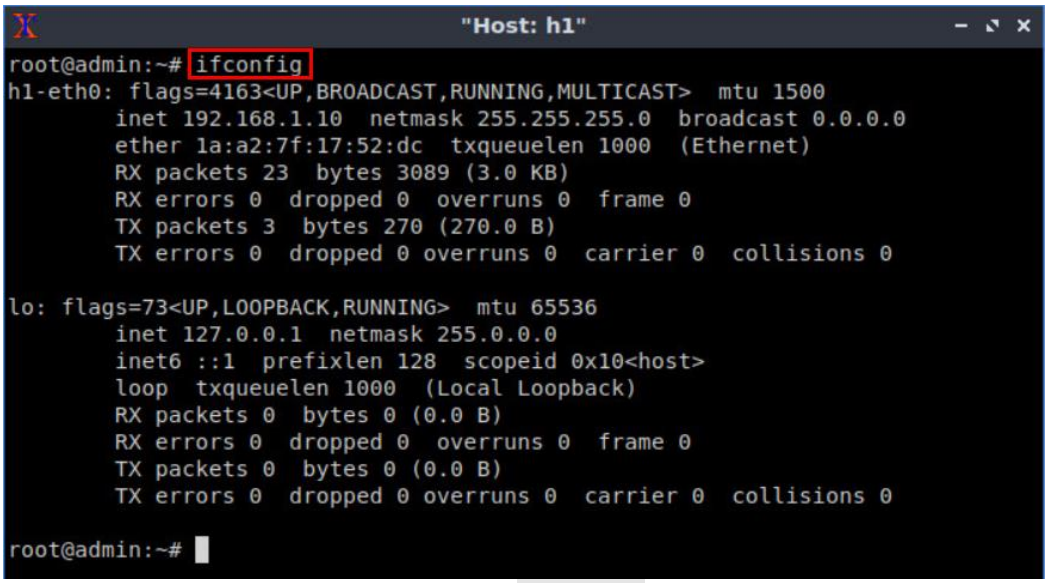

Figure 17. Output of `ifconfig` command.

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
```

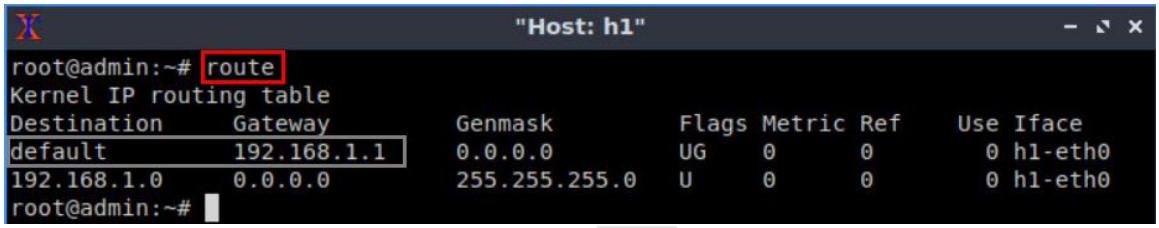

Figure 18. Output of `route` command.

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*.

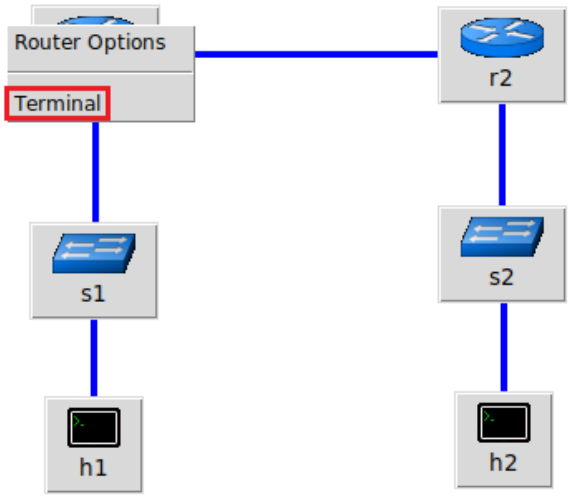

Figure 19. Opening a terminal on router r1.

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
```

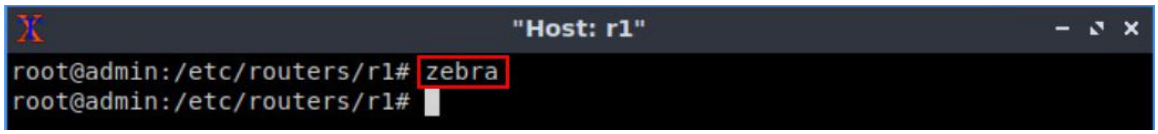

Figure 20. Starting zebra daemon.

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
```

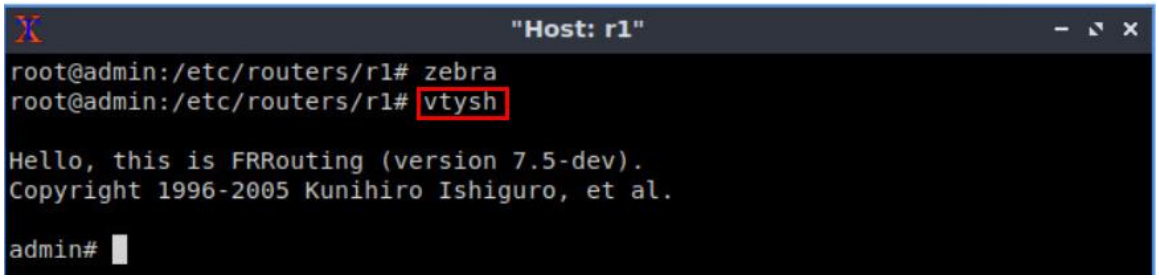

Figure 21. Starting vtysh on router r1.

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
```

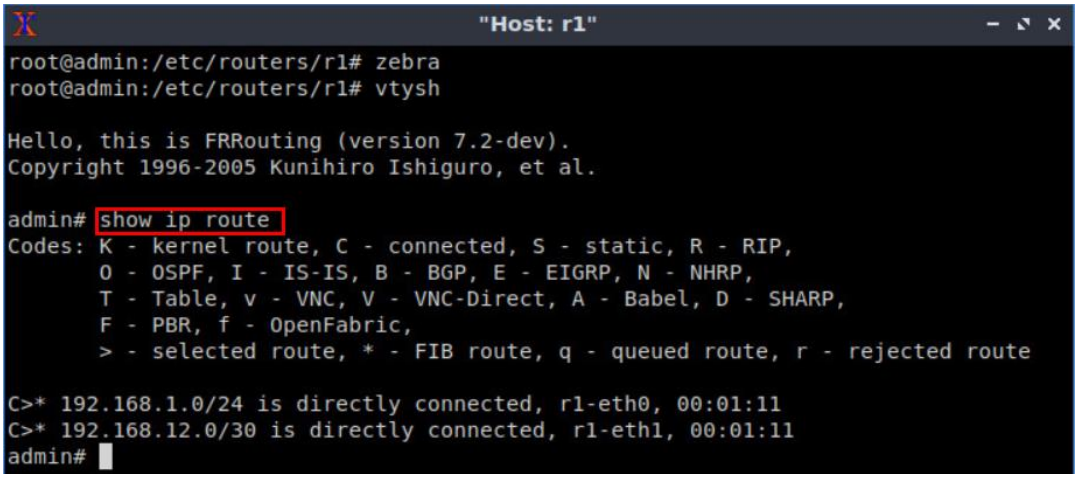

Figure 22. Displaying routing table of router r1.

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.

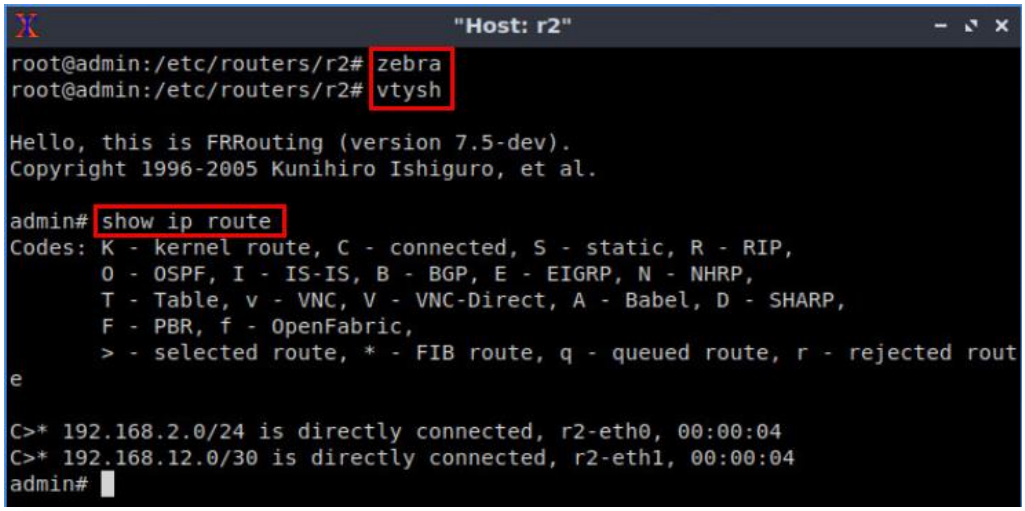

Figure 23. Displaying routing table of router r2.

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
```

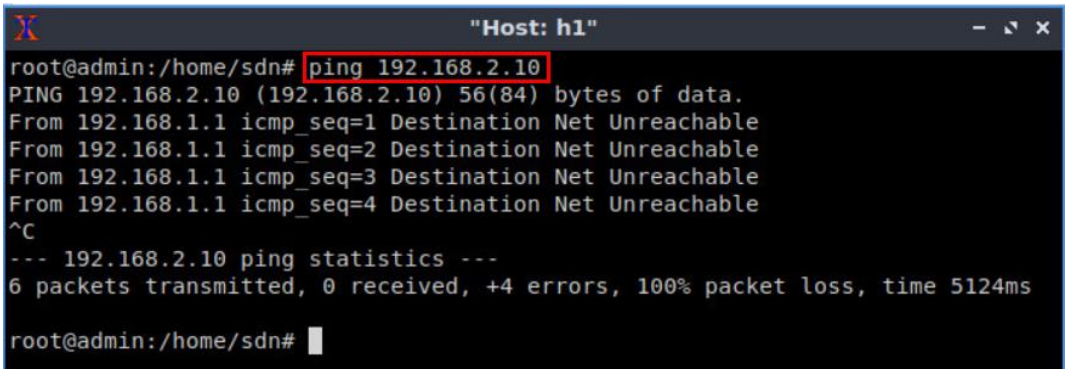

Figure 24. Connectivity test between host h1 and host h2.

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
```

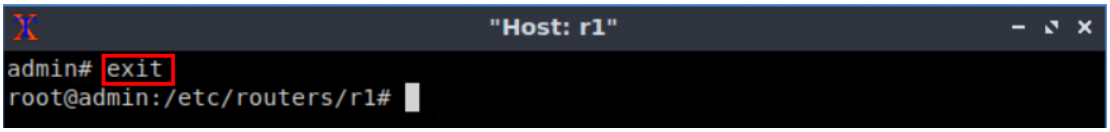

Figure 25. Exiting the vtysh session.

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
```

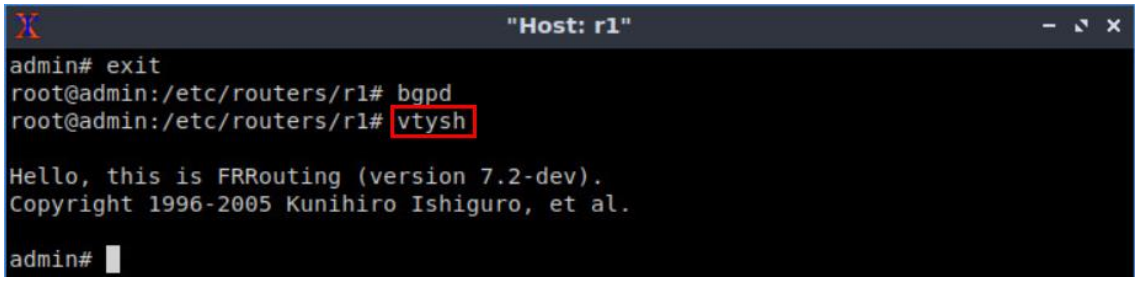

Figure 27. Starting vtysh on router r1.

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

```
configure terminal
```

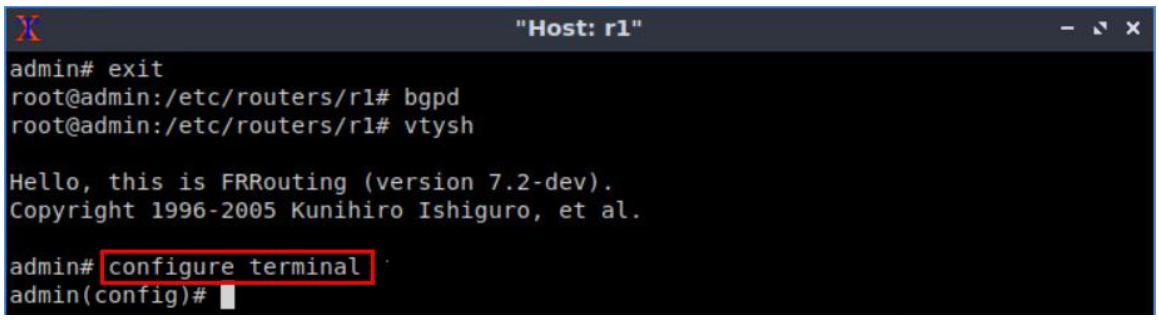

Figure 28. Enabling configuration mode on router r1.

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

```
router bgp 100
```

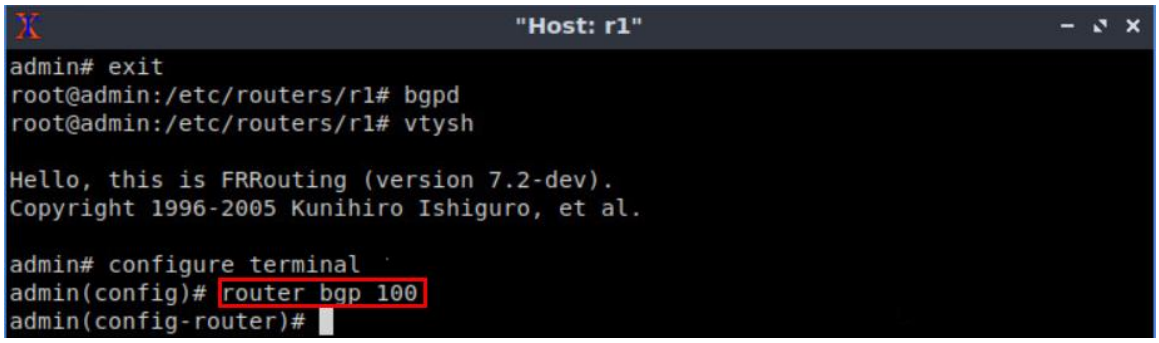

Figure 29. Configuring BGP on router r1.

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
```

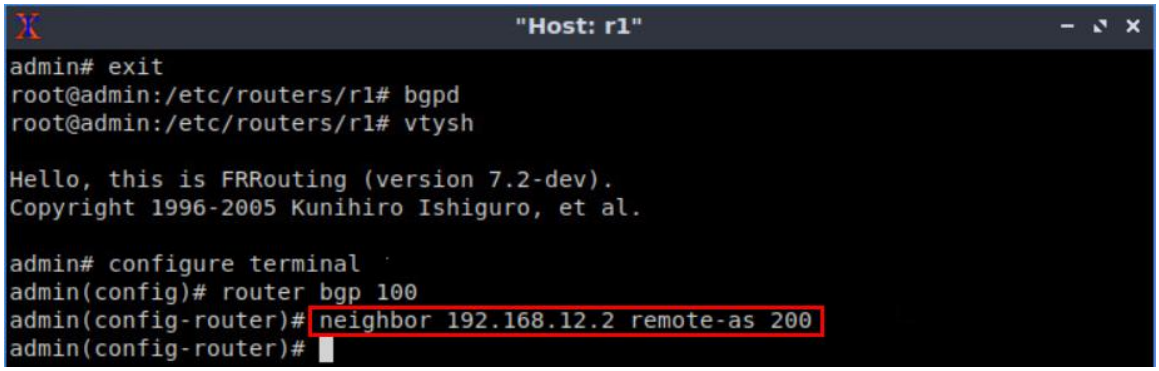

Figure 30. Assigning BGP neighbor to router r1.

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

```
end
```

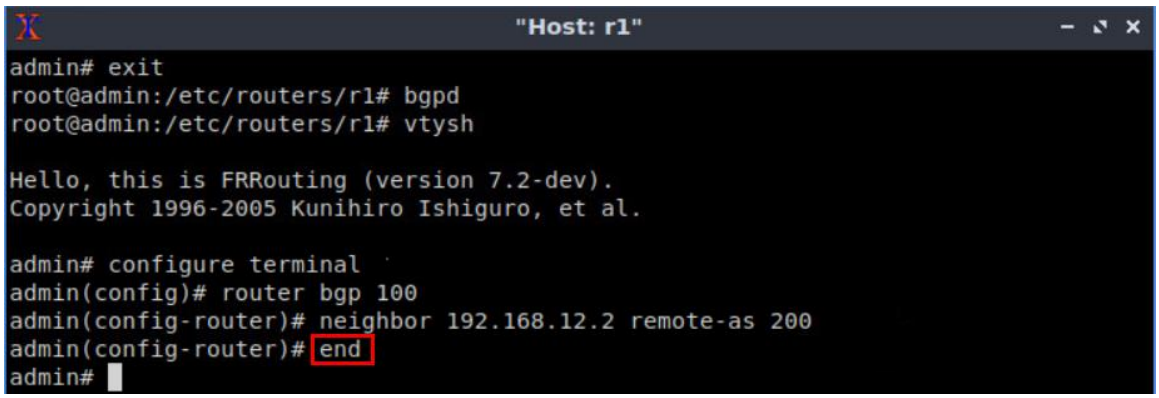

Figure 31. Exiting from configuration mode.

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
```

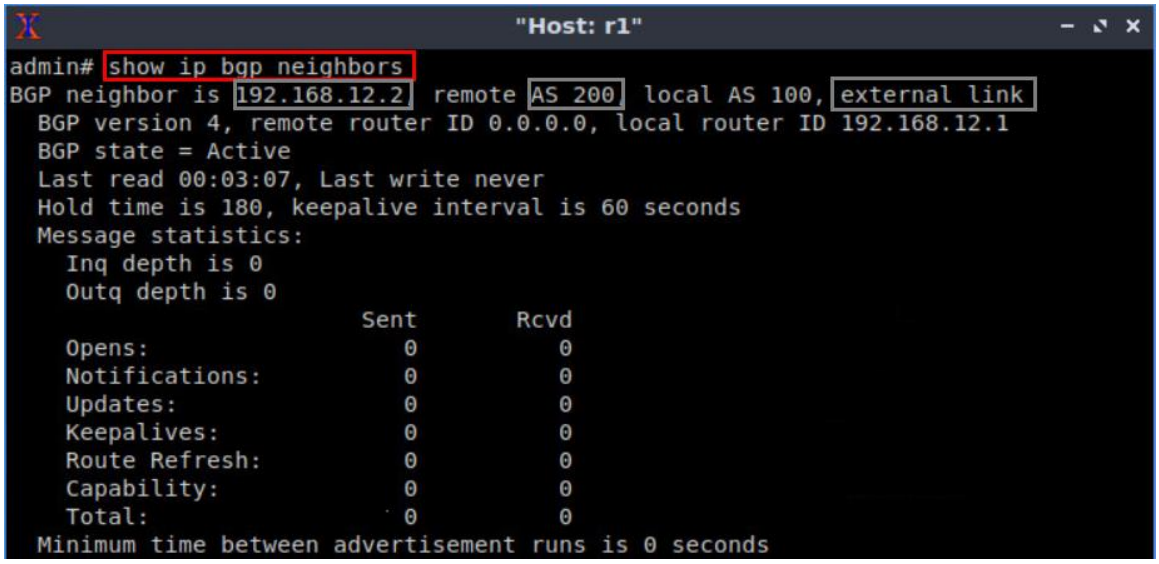

Figure 32. Verifying BGP neighbors on router r1.

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.

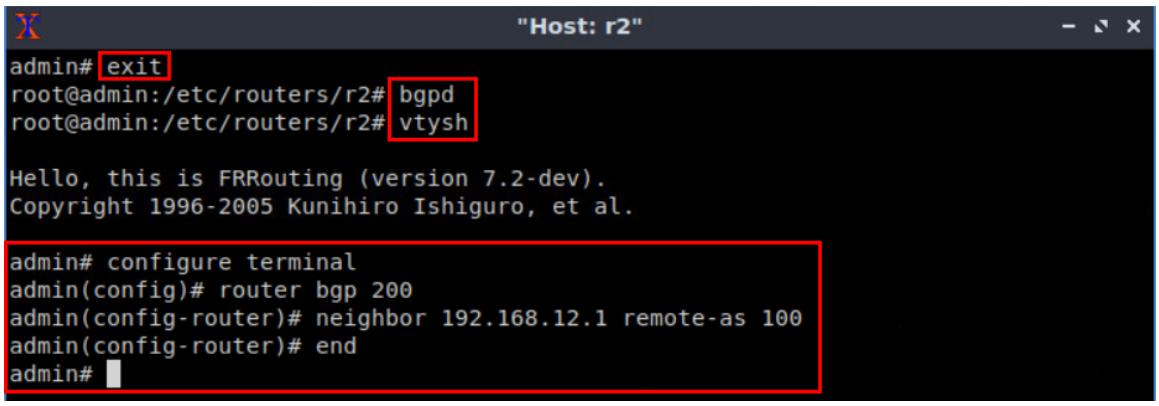

Figure 33. Assigning BGP neighbor to router r2.

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
```

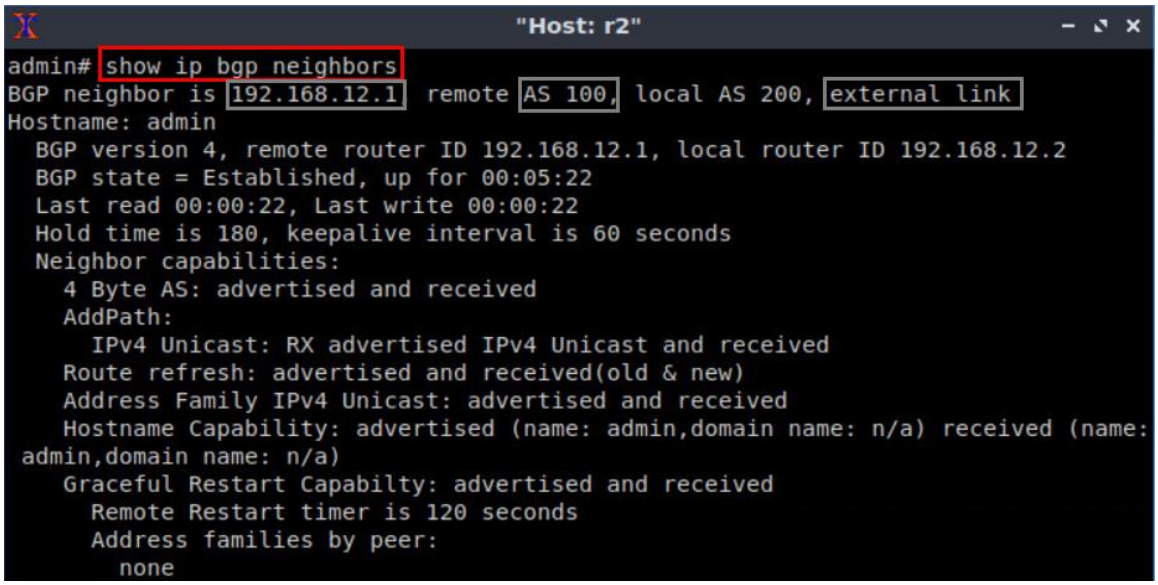

Figure 34. Verifying BGP neighbors on router r2.

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
```

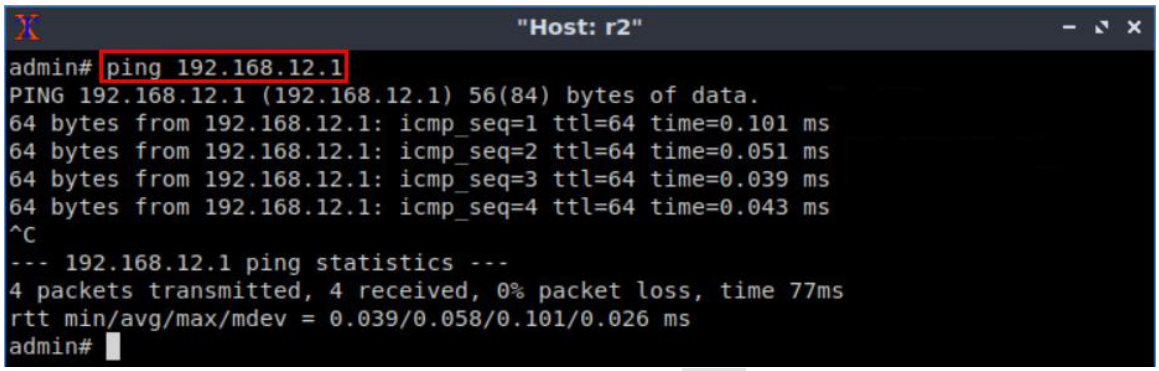

Figure 35. Connectivity test using `ping` command.

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
```

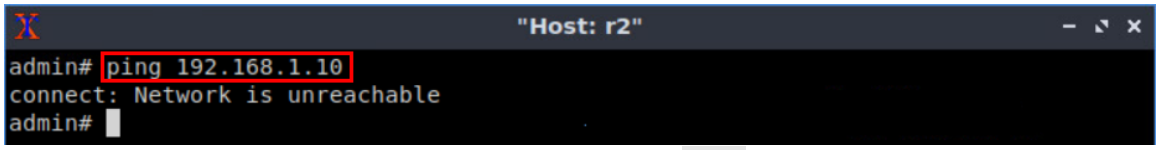

Figure 36. Connectivity test using `ping` command.

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
```

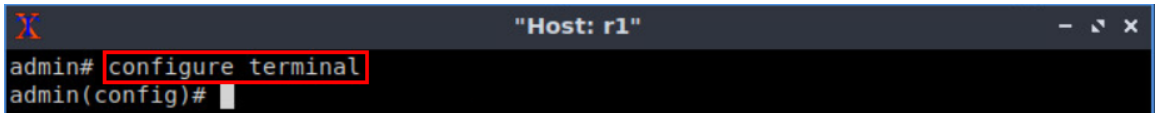

Figure 37. Enabling configuration mode on router r1.

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
```

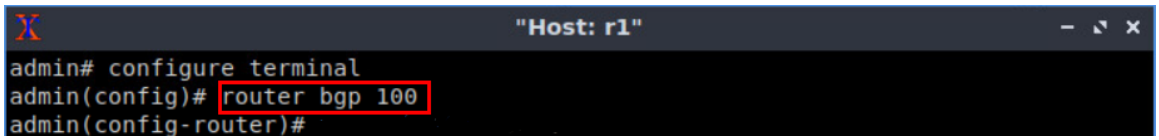

Figure 38. Entering to BGP configuration mode.

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
```

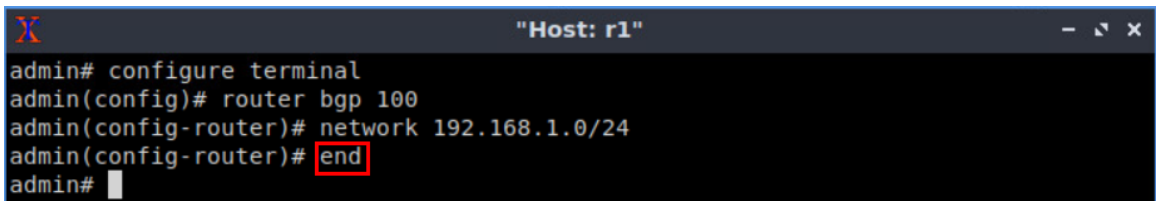

Figure 40. Exiting from configuration mode.

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

```
show ip bgp
```

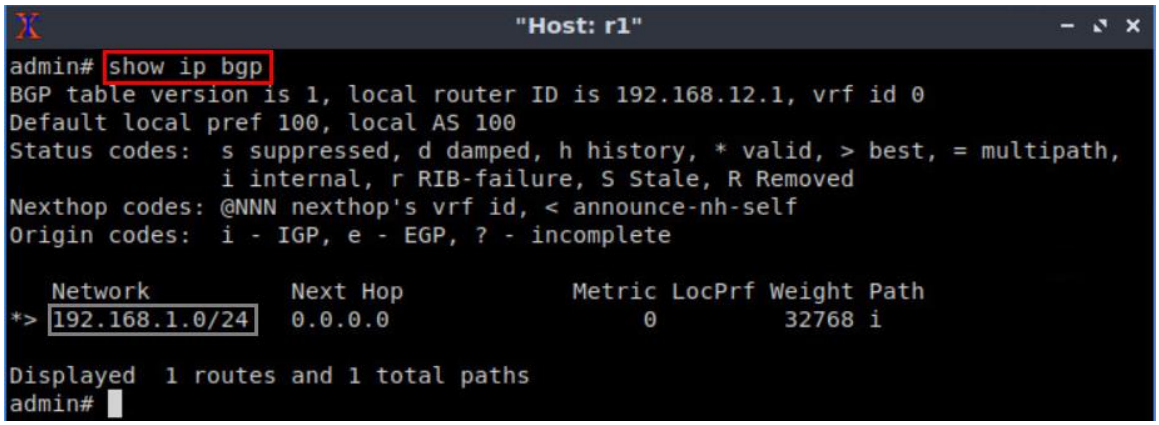

Figure 41. Verifying BGP networks on router r1.

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
```

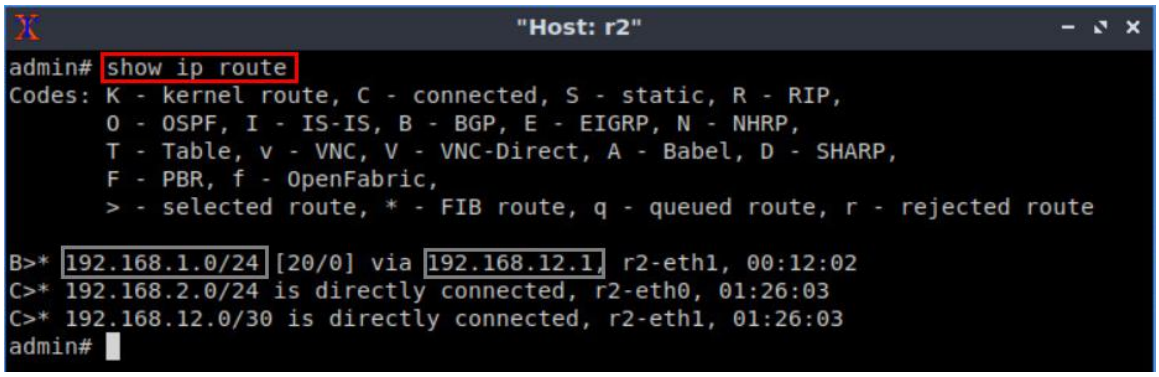

Figure 42. Verifying routing table of router r2.

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
```

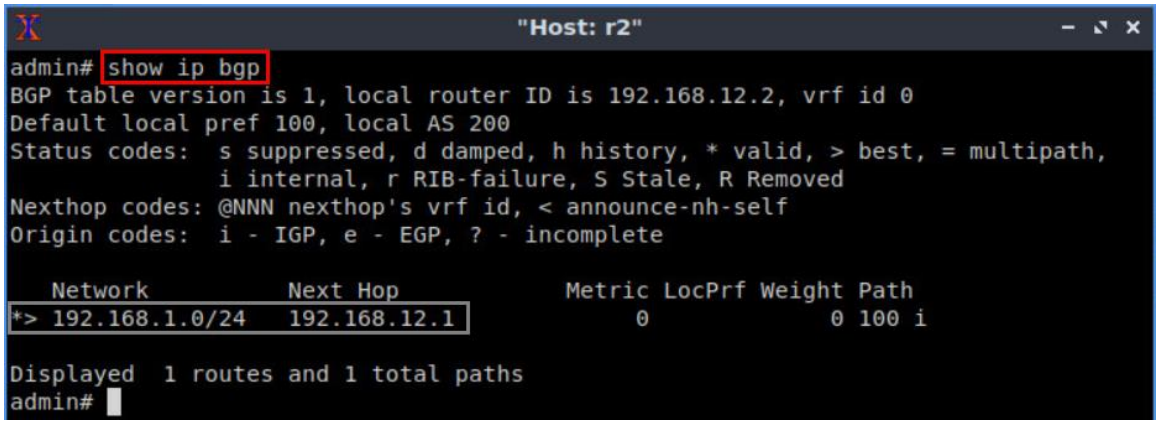

Figure 43. Verifying BGP table of router r2.

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.

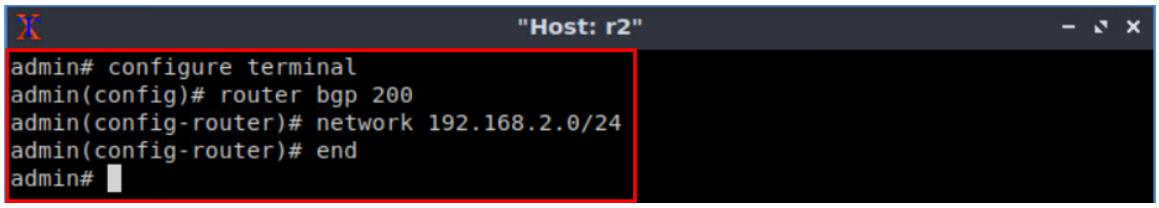

Figure 44. Advertising the network connected to router r2.

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
```

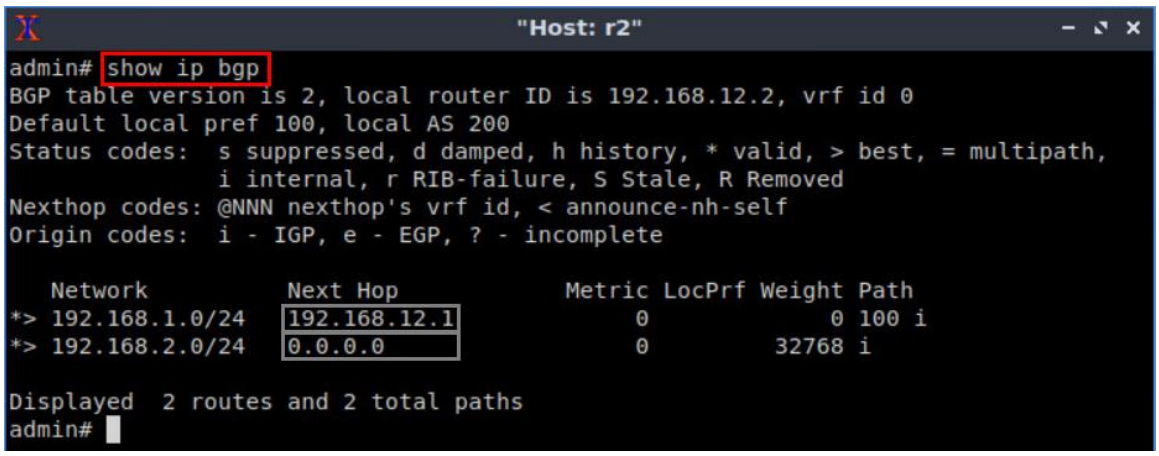

Figure 45. Verifying BGP table of router r2.

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
```

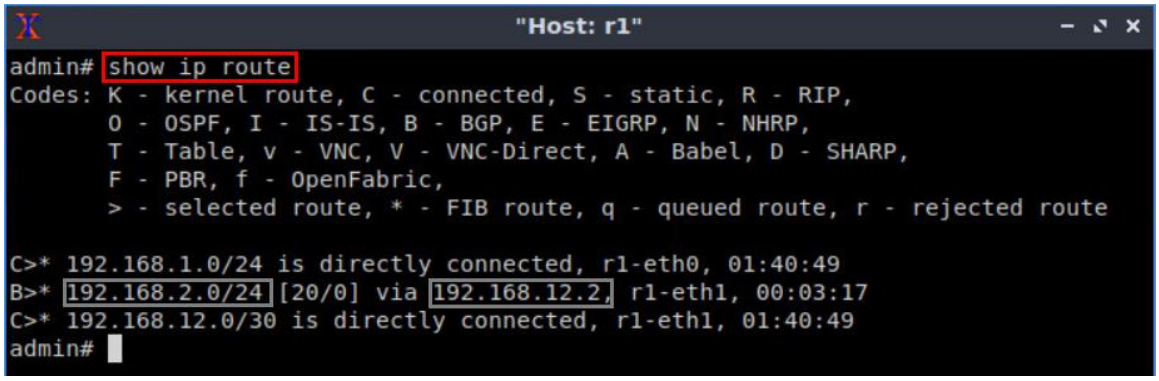

Figure 46. Verifying routing table of router r1.

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
```

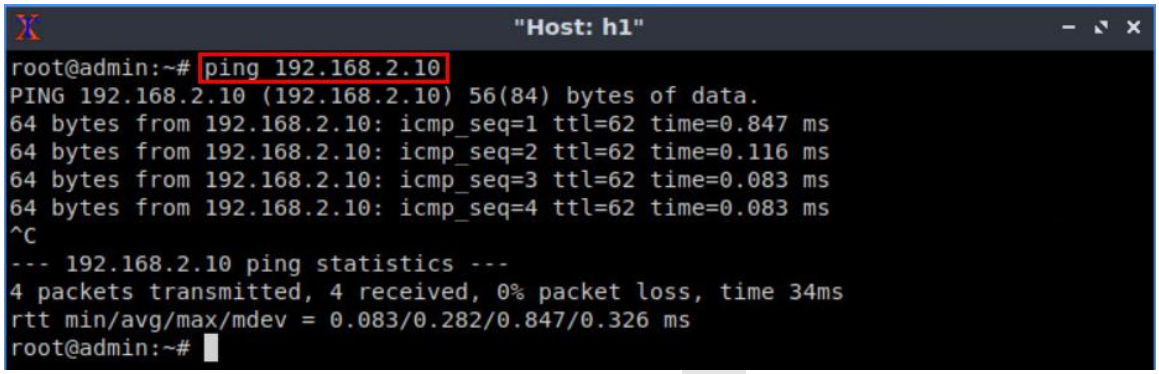

Figure 47. Connectivity test using `ping` command.

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

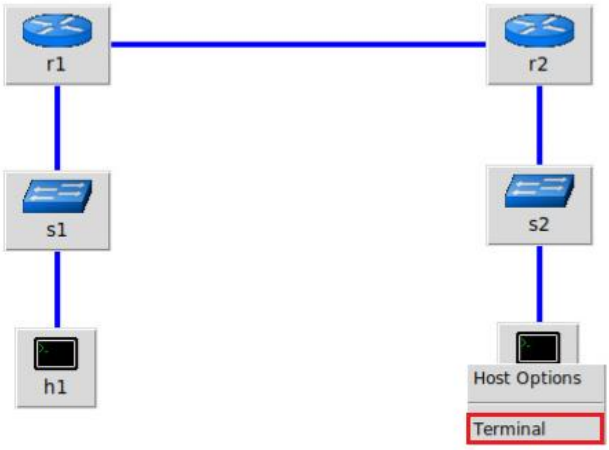

Figure 48. Opening host h2 terminal.

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
```

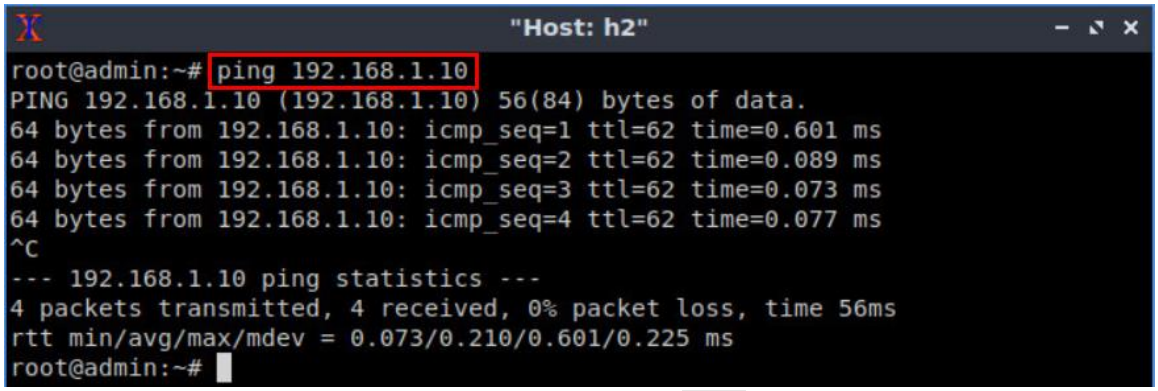

Figure 49. Connectivity test using `ping` command.

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