

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 networks¹. 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.

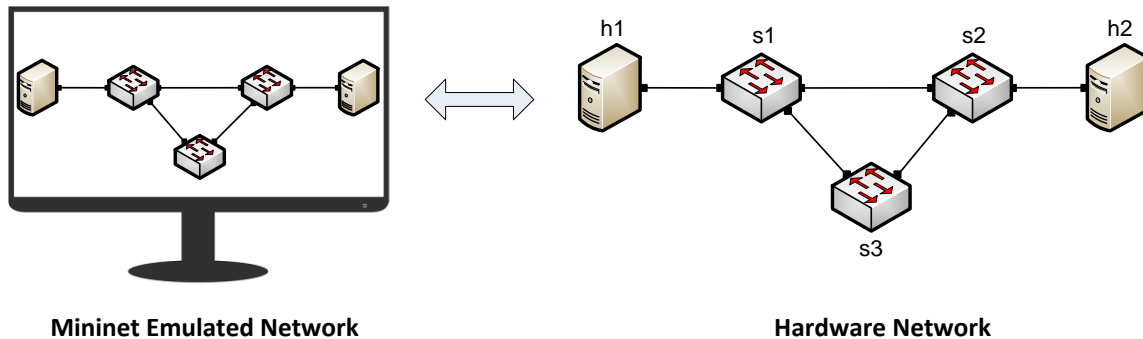


Figure 1. Hardware network vs. Mininet emulated network.

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 than 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 switch⁴. 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.

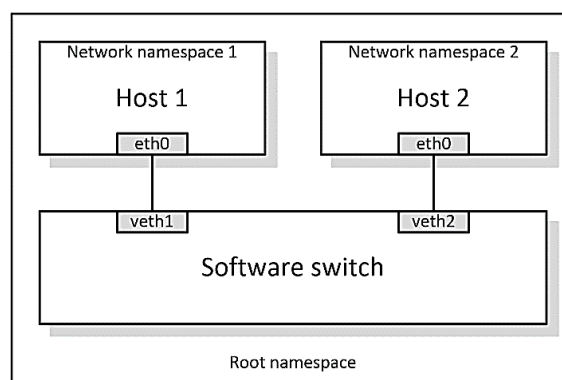


Figure 2. Network namespaces and virtual Ethernet links.

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
```

```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
sdn@admin:~$ sudo mn
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet>

```

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.

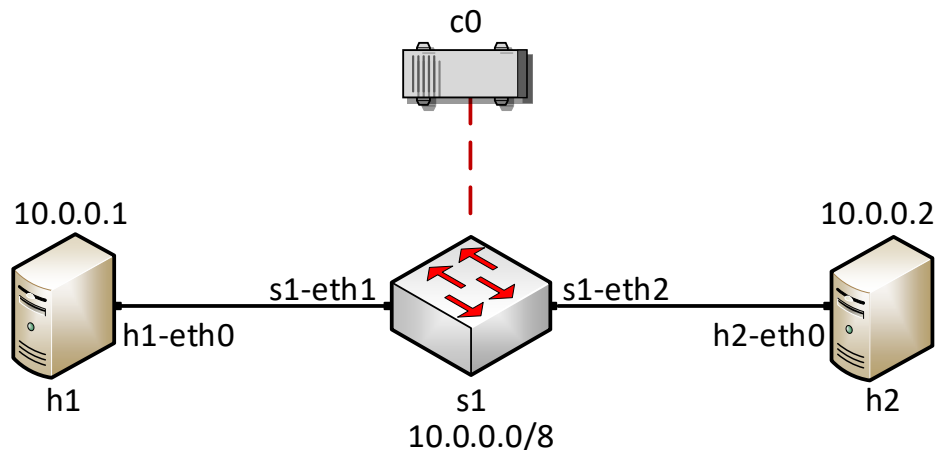


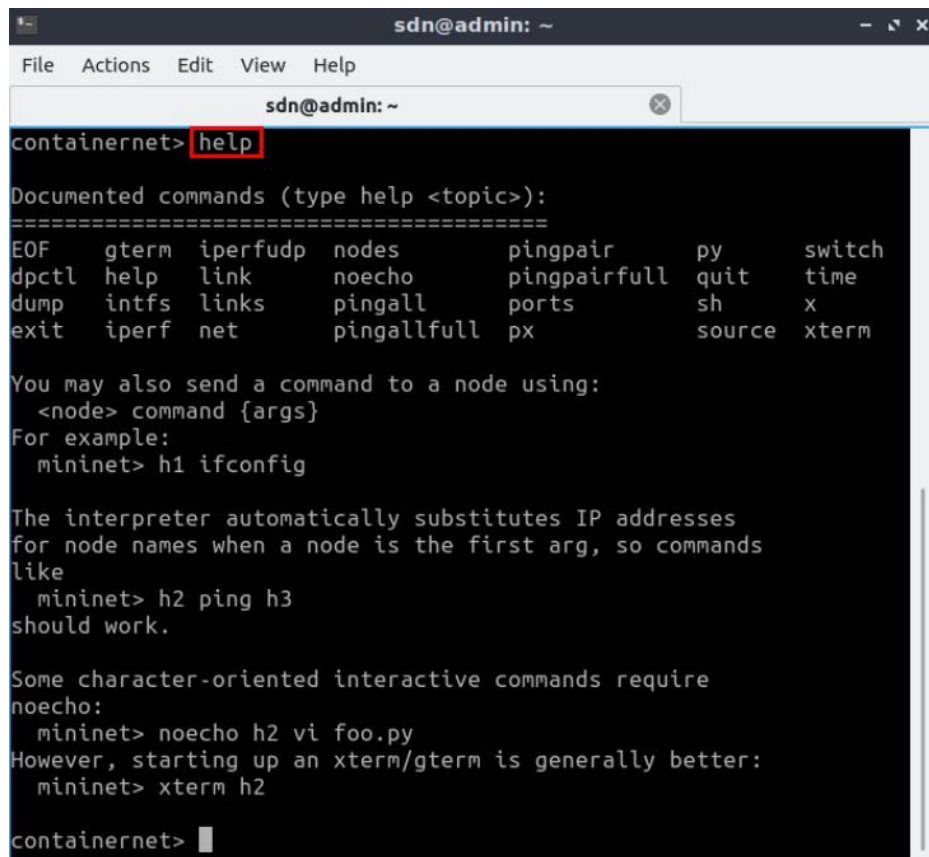
Figure 5. Mininet's default minimal topology.

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
```



```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
containernet> help

Documented commands (type help <topic>):
=====
EOF      gterm  iperfudp  nodes      pingpair    py      switch
dpctl    help   link      noecho     pingpairfull  quit    time
dump     intfs  links     pingall    ports       sh      x
exit     iperf  net       pingallfull px          source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

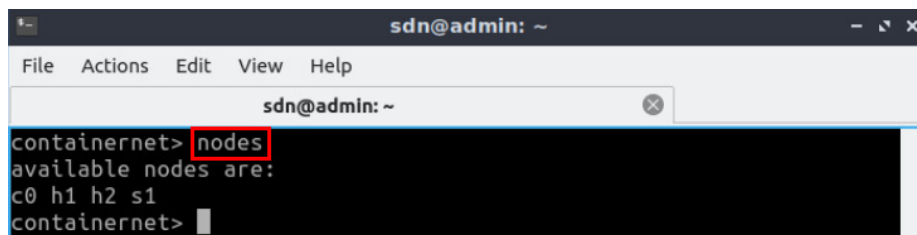
containernet>

```

Figure 6. Mininet's `help` command.

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

```
nodes
```



```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
containernet> nodes
available nodes are:
c0 h1 h2 s1
containernet>

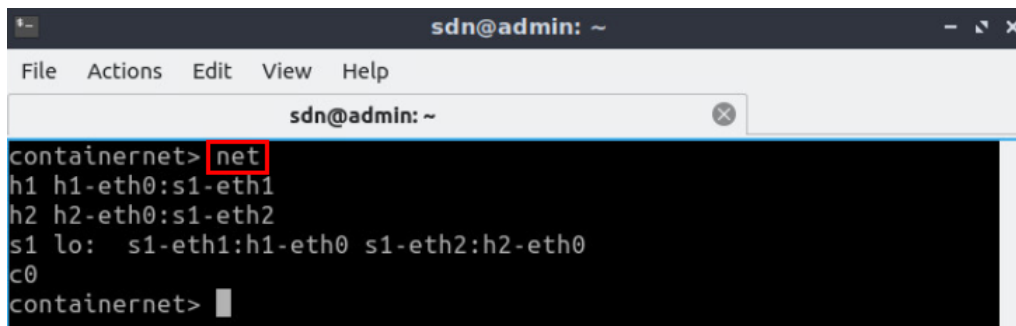
```

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
```



```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
containernetwork> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
containernetwork>

```

Figure 8. Mininet's `net` command.

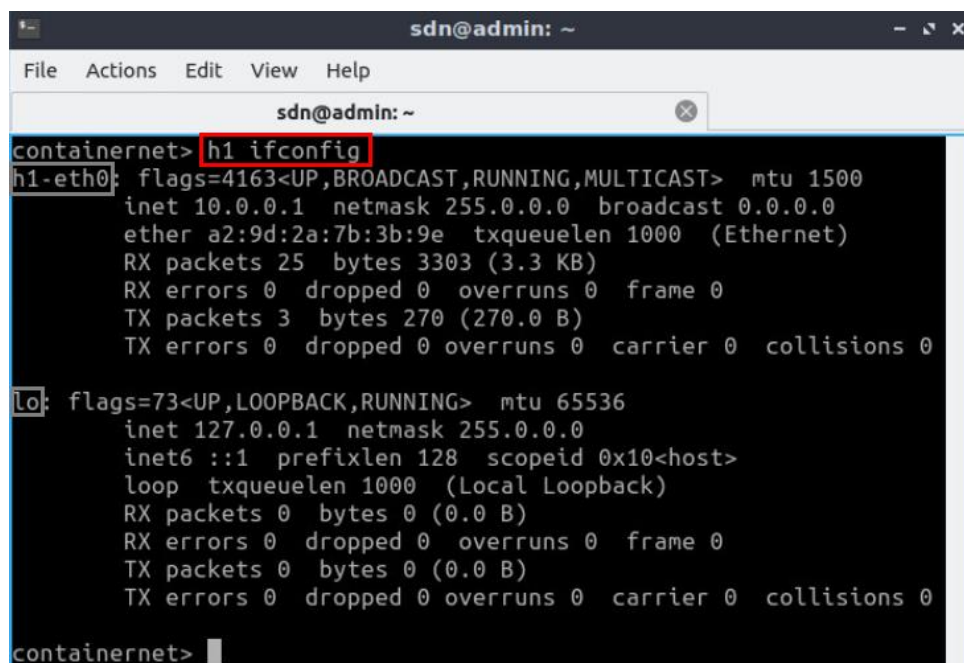
The output of this command shows that:

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
```



```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
containernetwork> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 0.0.0.0
    ether a2:9d:2a:7b:3b:9e txqueuelen 1000 (Ethernet)
    RX packets 25 bytes 3303 (3.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 270 (270.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
containernetwork>

```

Figure 9. Output of `h1 ifconfig` command.

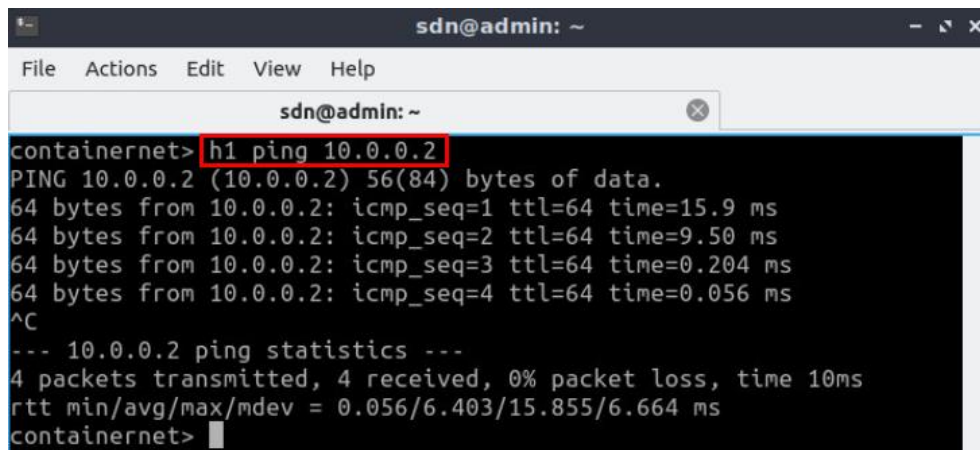
This command executes the `ifconfig` Linux command on host h1. The command shows host h1's interfaces. The display indicates that host h1 has an interface `h1-eth0` configured with IP address 10.0.0.1, and another interface `lo` configured with IP address 127.0.0.1 (loopback interface).

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
```



```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
containernet> h1 ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=15.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=9.50 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.204 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.056 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 10ms
rtt min/avg/max/mdev = 0.056/6.403/15.855/6.664 ms
containernet>

```

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

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

```
exit
```

```

sdn@admin: ~
File Actions Edit View Help
sdn@admin: ~
containernet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 1339.802 seconds
sdn@admin:~$

```

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 MiniEdit⁵ to deploy the topology illustrated below. MiniEdit is a simple GUI network editor for Mininet.

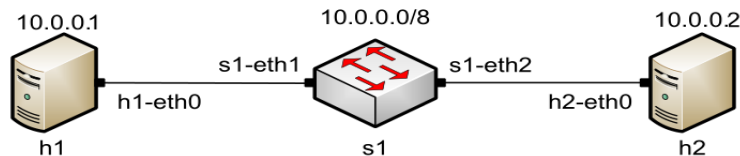


Figure 12. Lab topology.

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`.

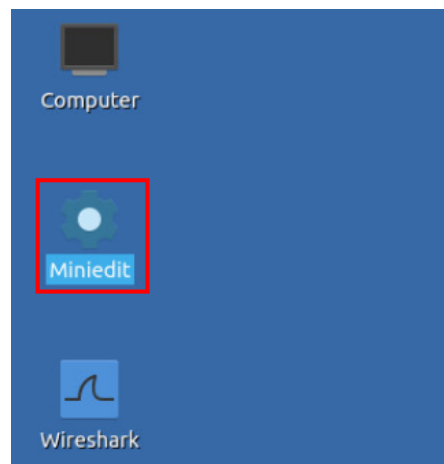


Figure 13. MiniEdit Desktop shortcut.

MiniEdit will start, as illustrated below.

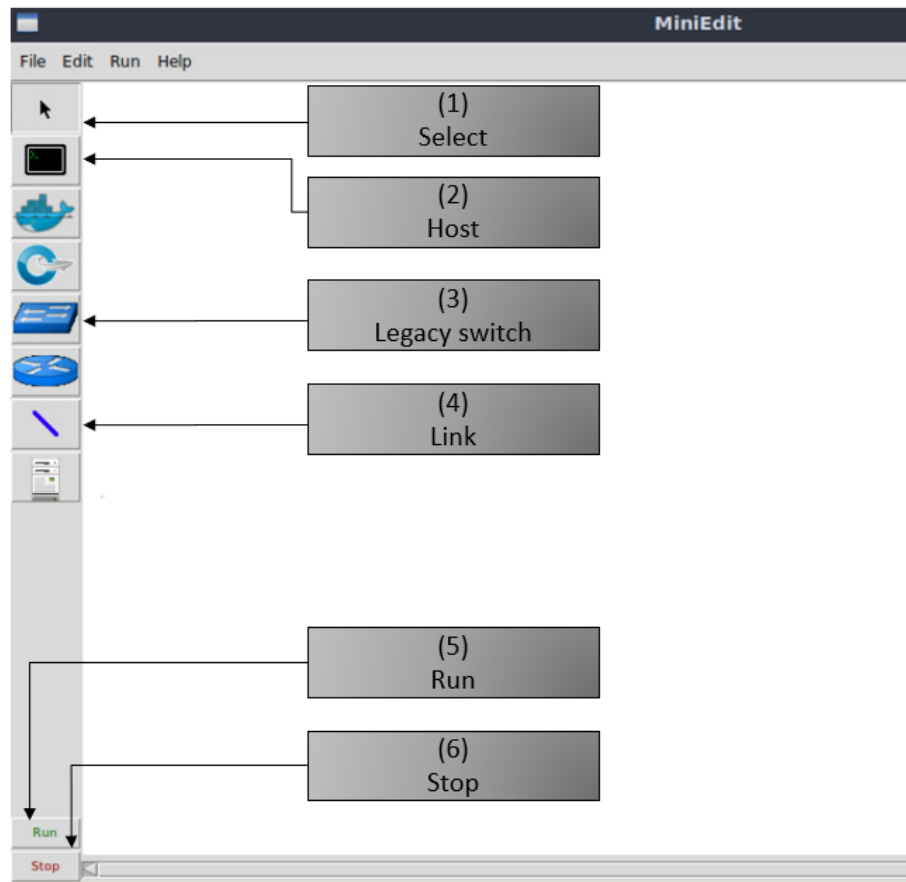


Figure 14. MiniEdit Graphical User Interface (GUI).

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.

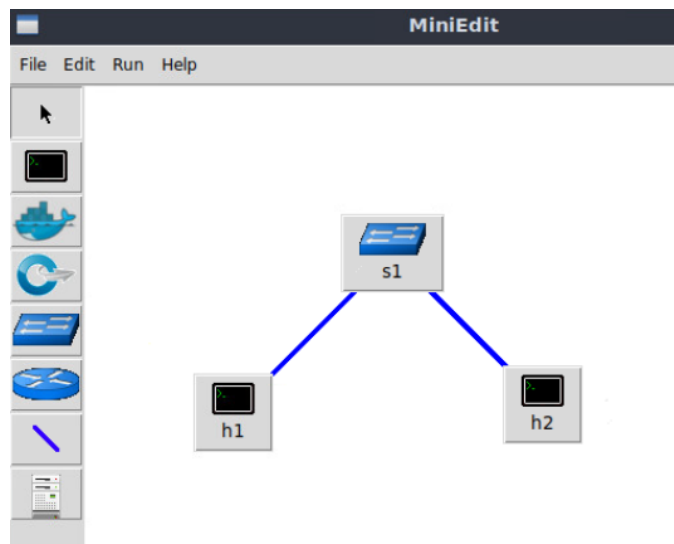


Figure 15. MiniEdit's topology.

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.

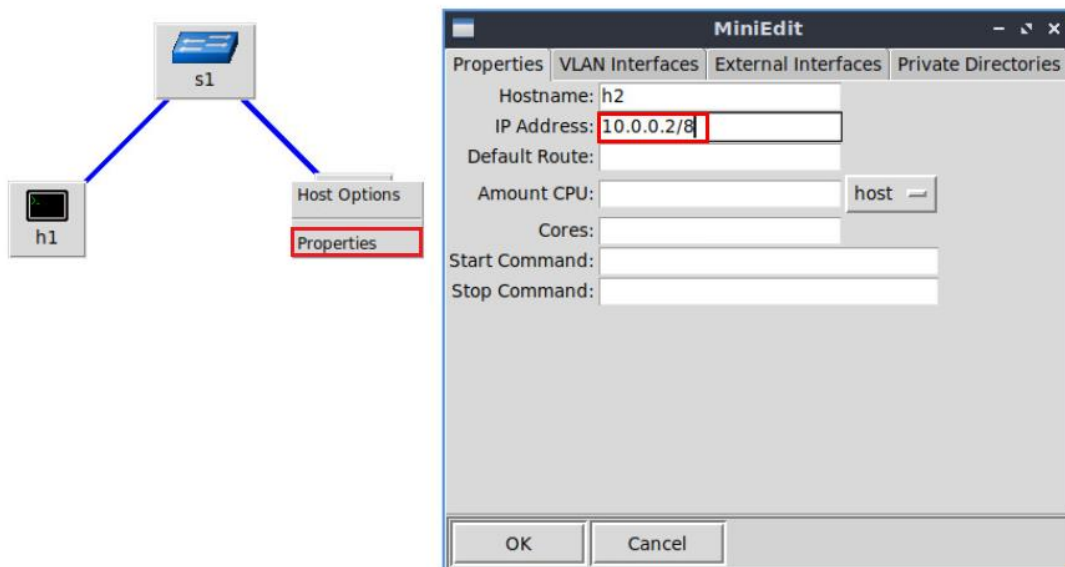


Figure 16. Configuration of a host's properties.

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.

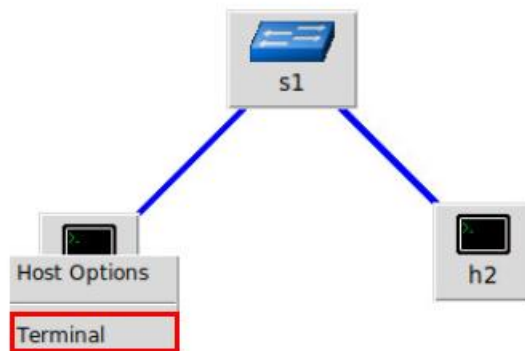


Figure 18. Opening a terminal on host h1.

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

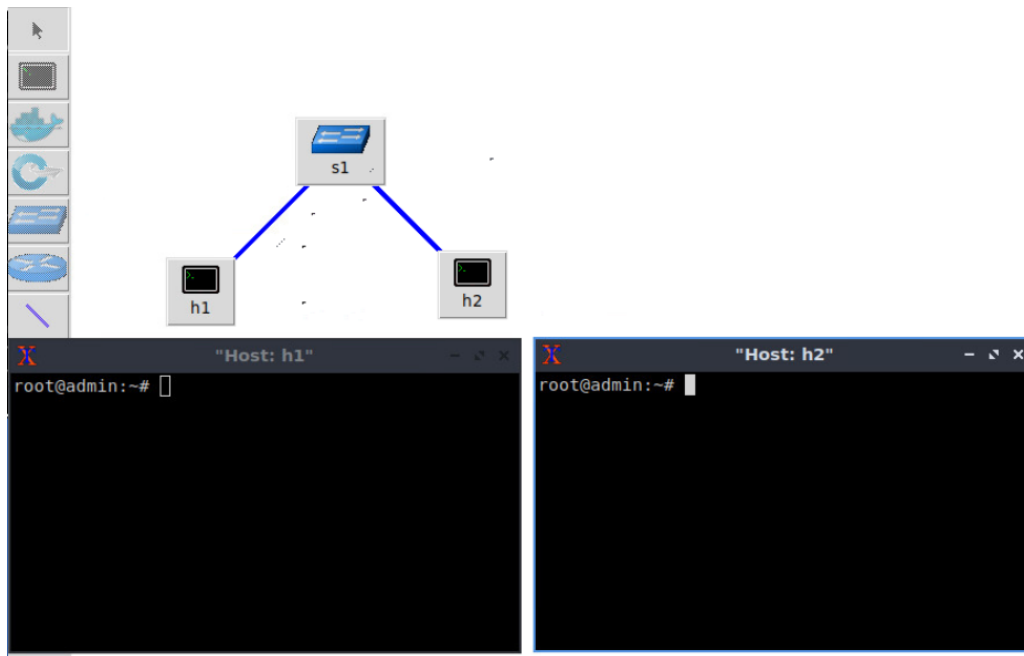


Figure 19. Terminals at host h1 and host h2.

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
```

```

"Host: h1"
root@admin:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 0.0.0.0
    ether 12:35:67:8c:4a:24 txqueuelen 1000 (Ethernet)
    RX packets 23 bytes 3089 (3.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 270 (270.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@admin:~#

```

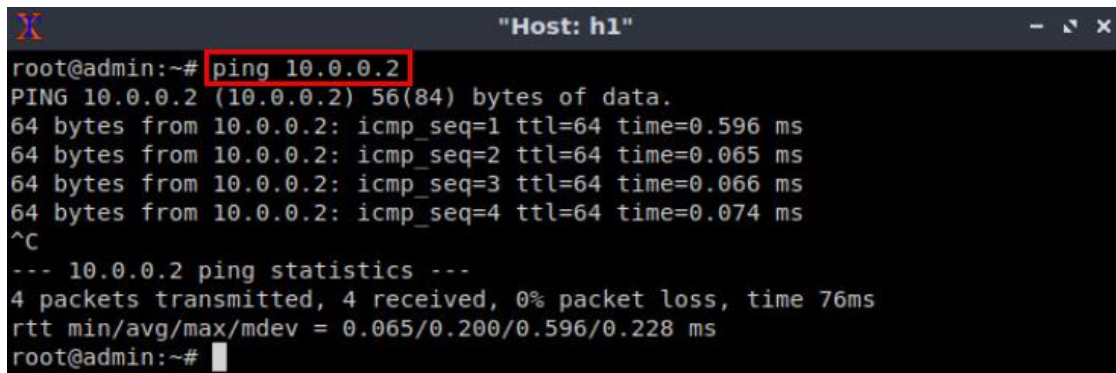
Figure 20. Output of `ifconfig` command on host h1.

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
```



```

root@admin:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.596 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.066 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.074 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 76ms
rtt min/avg/max/mdev = 0.065/0.200/0.596/0.228 ms
root@admin:~#

```

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.

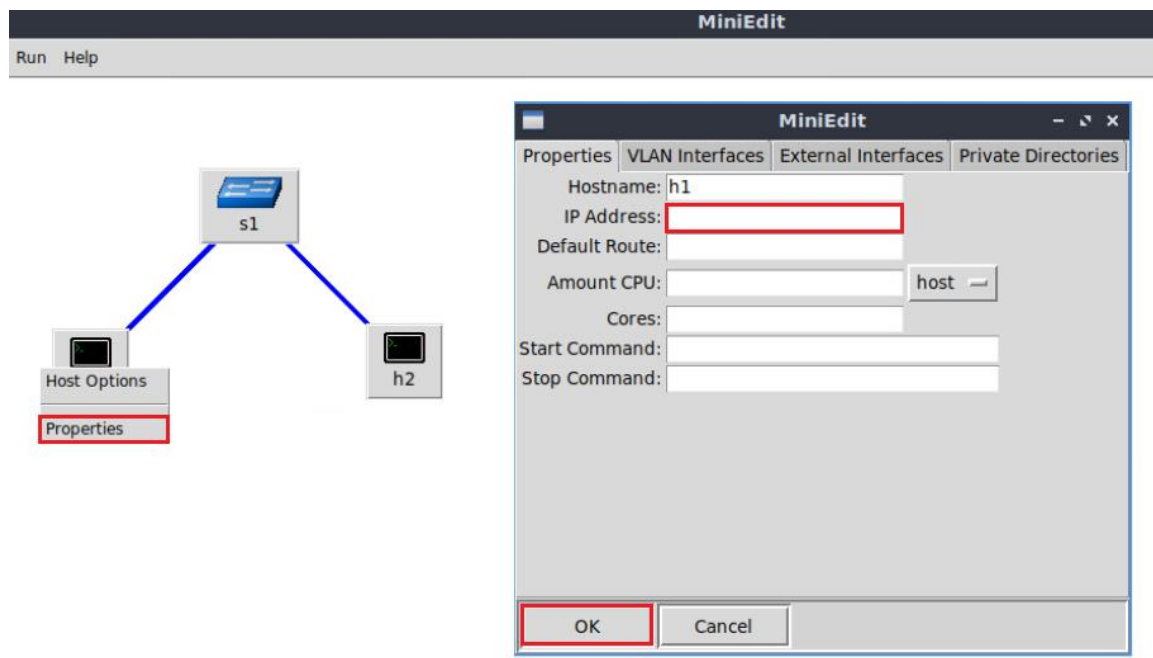


Figure 23. Host h1 properties.

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.

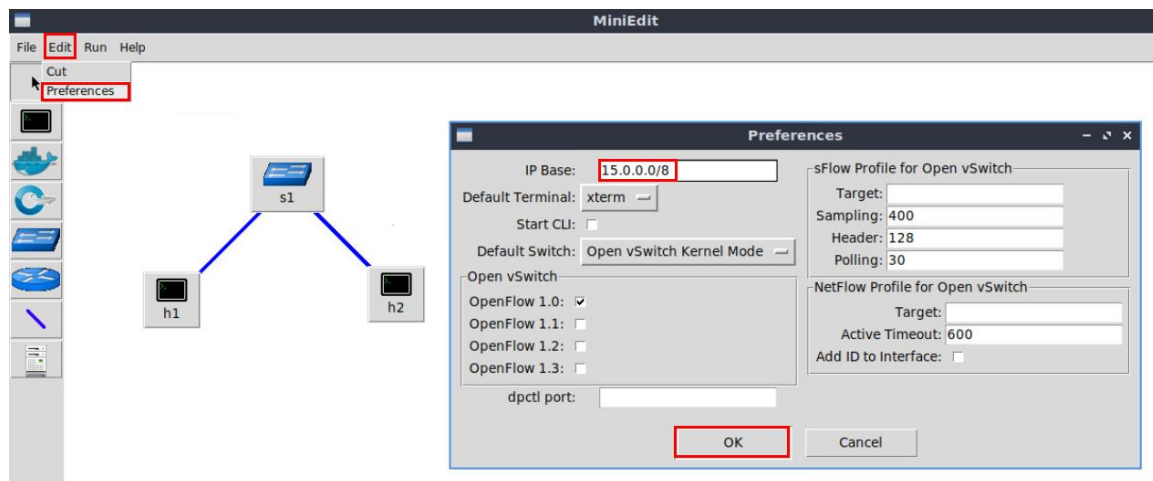


Figure 24. Modification of the IP Base (network address and prefix length).

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.

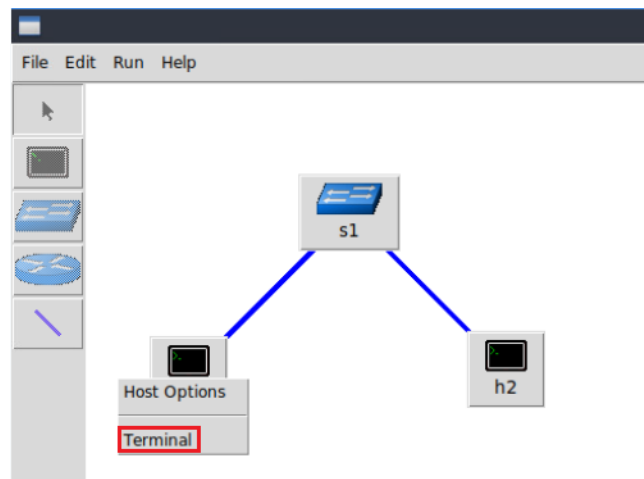


Figure 25. Opening a terminal on host h1.

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
```

The image shows a terminal window titled '"Host: h1"'. The prompt is 'root@admin:~#'. The command 'ifconfig' has been entered and is highlighted with a red box. The output shows details for the 'h1-eth0' interface and the loopback interface 'lo'. The 'h1-eth0' interface has IP address 15.0.0.1, netmask 255.0.0.0, and broadcast 0.0.0.0. The 'lo' interface has IP address 127.0.0.1, netmask 255.0.0.0, and is a local loopback interface.

```
root@admin:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 15.0.0.1 netmask 255.0.0.0 broadcast 0.0.0.0
    ether 3a:5c:0b:d2:8a:1f txqueuelen 1000 (Ethernet)
    RX packets 14 bytes 1950 (1.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 270 (270.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@admin:~#
```

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.

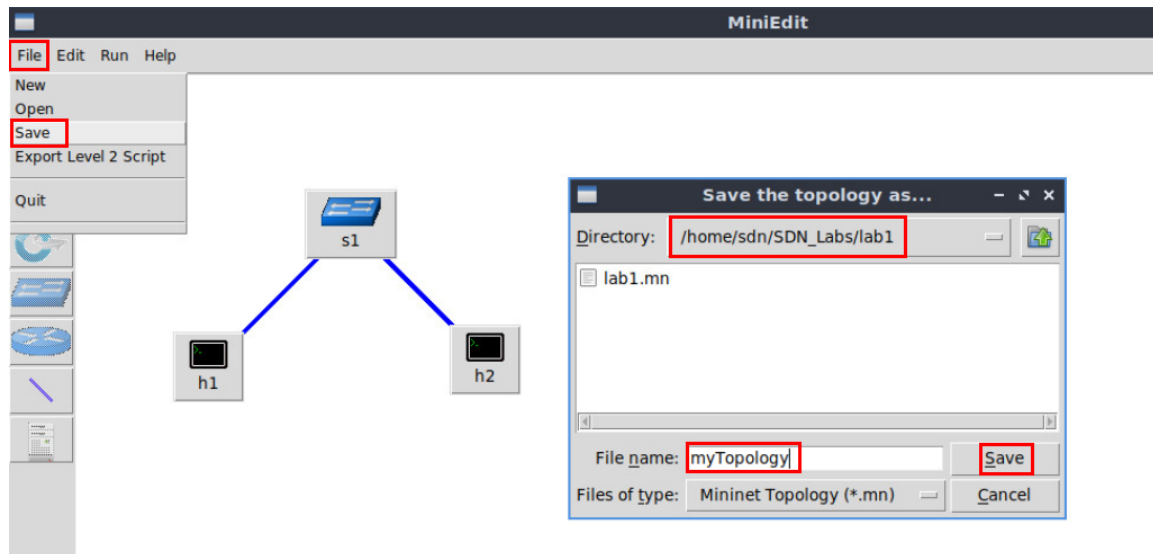


Figure 28. Saving the topology.

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.

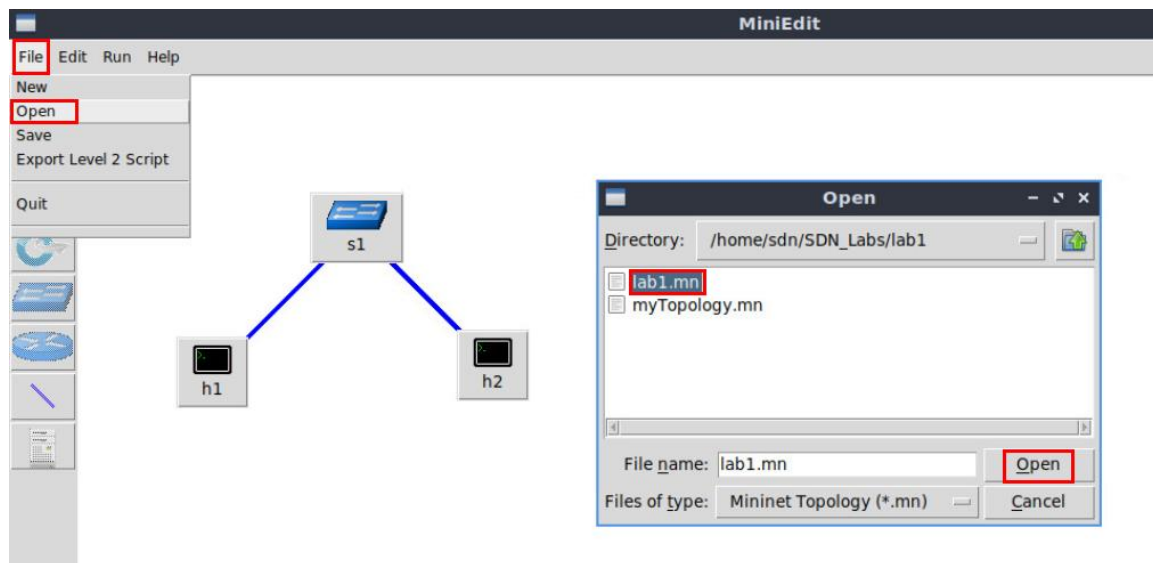


Figure 29. Opening a topology.

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.

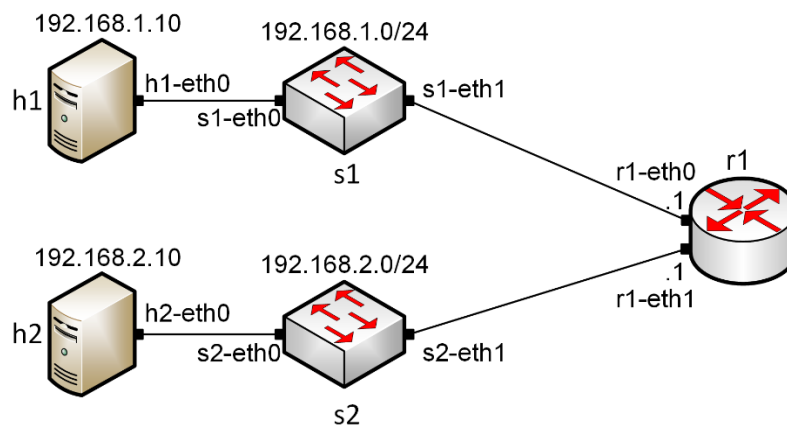


Figure 30. Topology.

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.

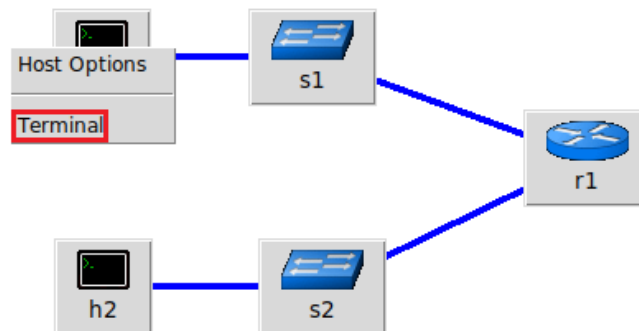


Figure 32. Opening a terminal on host h1.

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
```

```

root@admin:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.10 netmask 255.255.255.0 broadcast 0.0.0.0
    ether be:72:ce:b7:f1:aa txqueuelen 1000 (Ethernet)
    RX packets 17 bytes 2459 (2.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3 bytes 270 (270.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@admin:~#

```

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
```

```

root@admin:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.1.1 0.0.0.0 UG 0 0 0 h1-eth0
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 h1-eth0

root@admin:~#

```

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*.

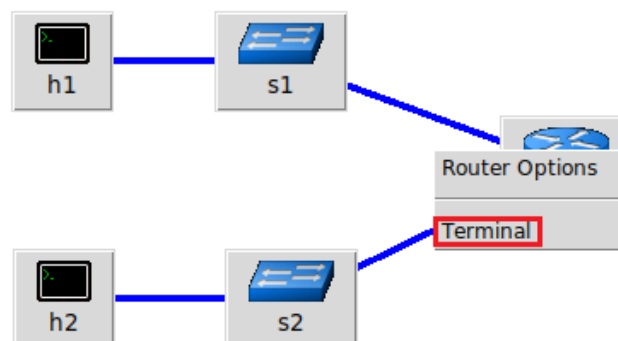


Figure 35. Opening a terminal on router r1.

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
```

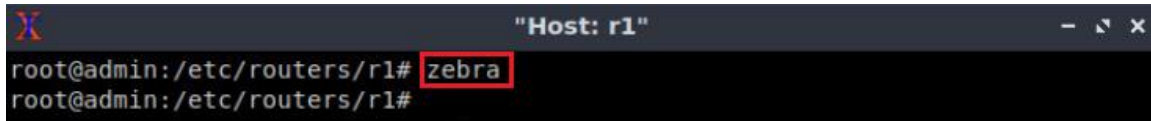
A terminal window titled "Host: r1" showing the command prompt root@admin:/etc/routers/r1#. The command 'zebra' is entered and highlighted with a red box. The prompt then changes to root@admin:/etc/routers/r1#.

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:

```
vttysh
```

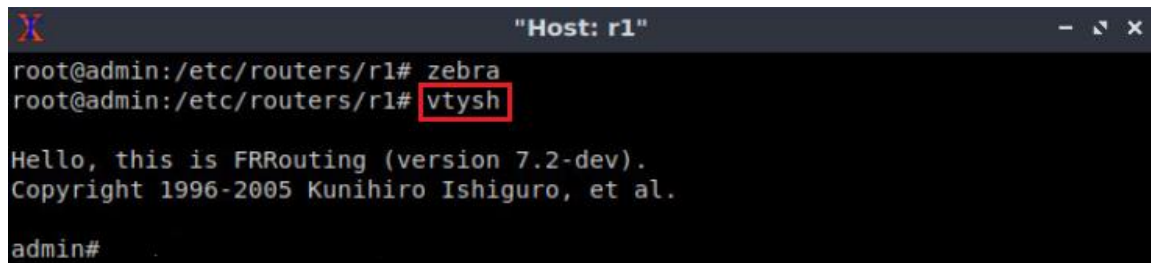
A terminal window titled "Host: r1" showing the command prompt root@admin:/etc/routers/r1#. The command 'zebra' is entered first, followed by 'vttysh', which is highlighted with a red box. The prompt then changes to admin#. Below the prompt, a message is displayed: "Hello, this is FRRouting (version 7.2-dev). Copyright 1996-2005 Kunihiro Ishiguro, et al."

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
```

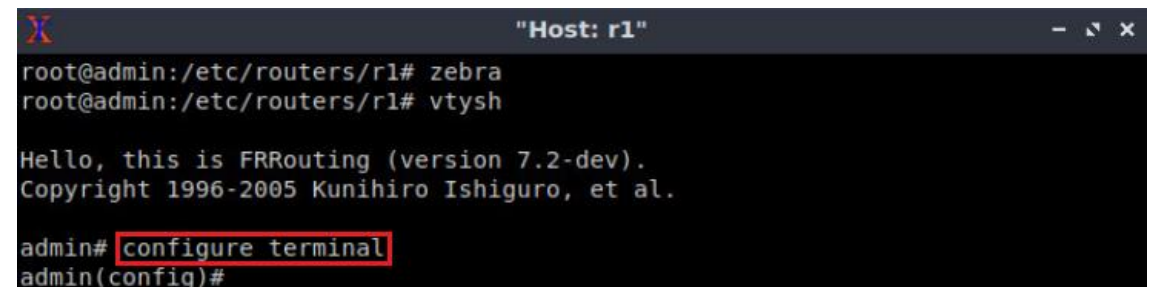
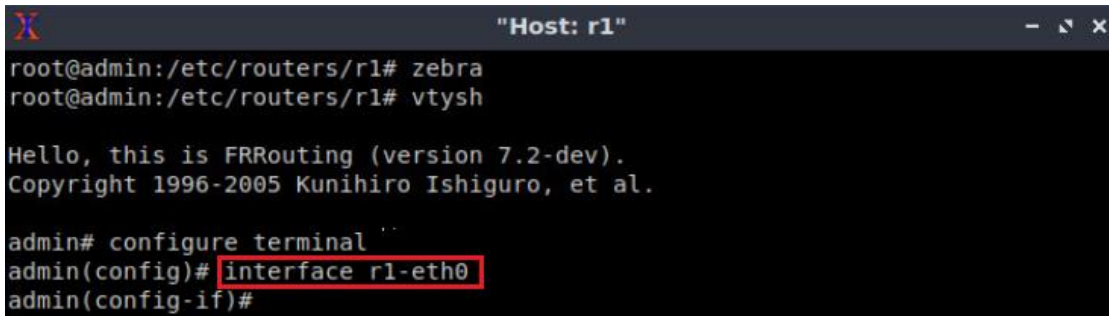
A terminal window titled "Host: r1" showing the command prompt root@admin:/etc/routers/r1#. The commands 'zebra' and 'vttysh' are entered. The prompt then changes to admin#. The command 'configure terminal' is entered and highlighted with a red box. The prompt then changes to admin(config)#.

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
```



```

Host: r1
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

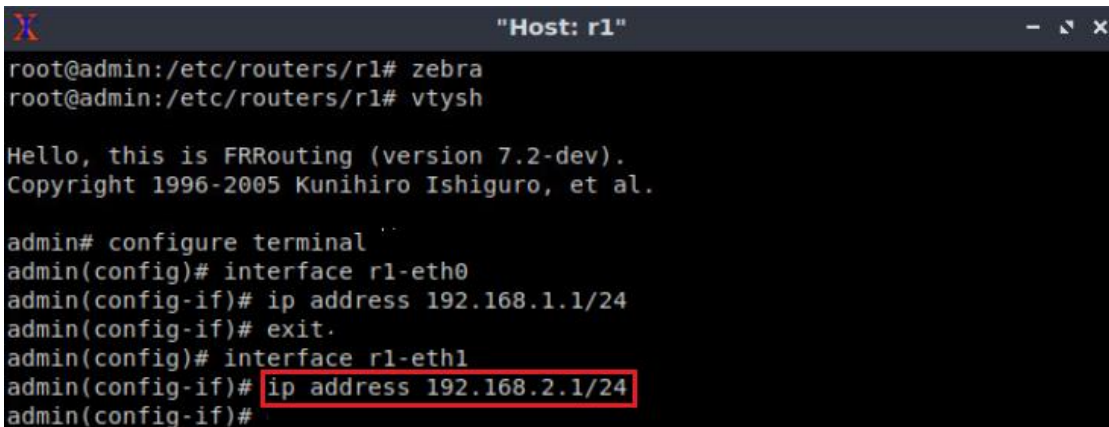
admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)#

```

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
```



```

Host: r1
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

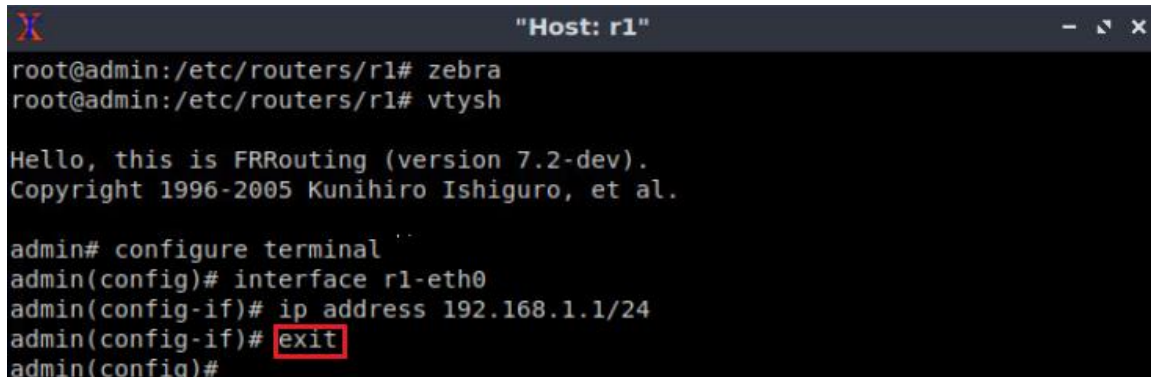
admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit
admin(config)# interface r1-eth1
admin(config-if)# ip address 192.168.2.1/24
admin(config-if)#

```

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

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

```
exit
```



```

Host: r1
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

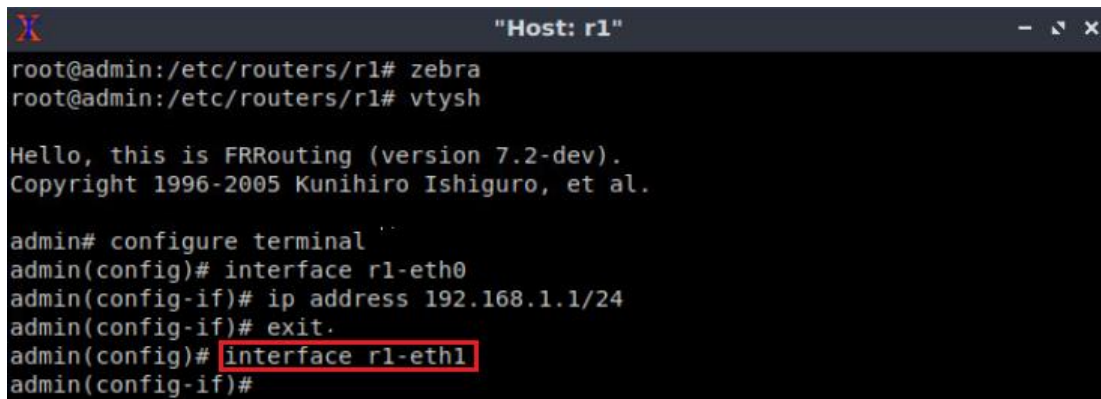
admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit
admin(config)#

```

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
```



```
Host: r1
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

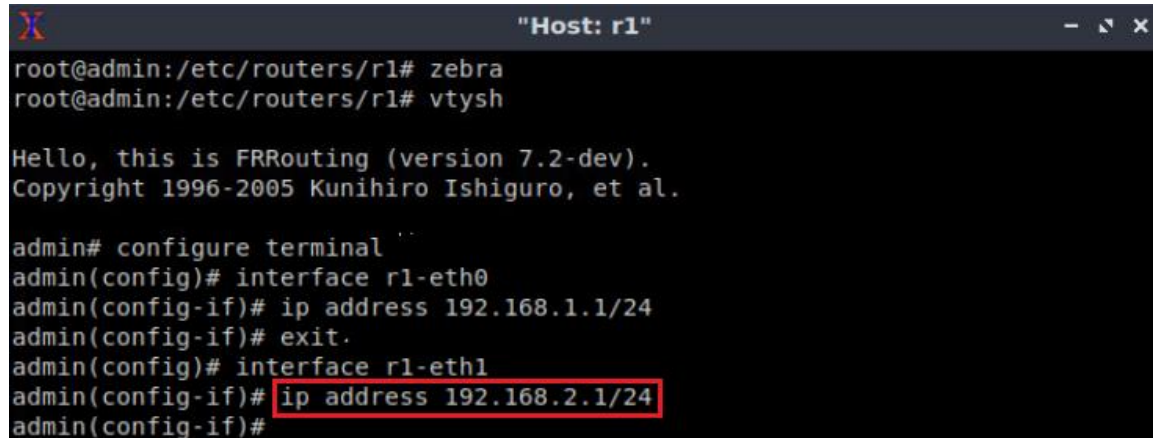
Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit.
admin(config)# interface r1-eth1
admin(config-if)#
```

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
```



```
Host: r1
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

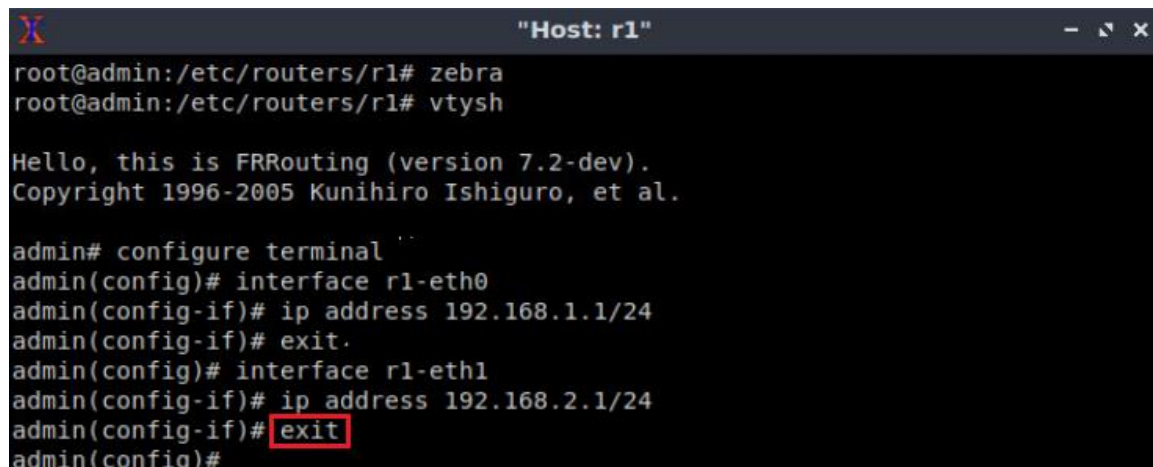
Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit.
admin(config)# interface r1-eth1
admin(config-if)# ip address 192.168.2.1/24
admin(config-if)#
```

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

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

```
exit
```



```
Host: r1
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

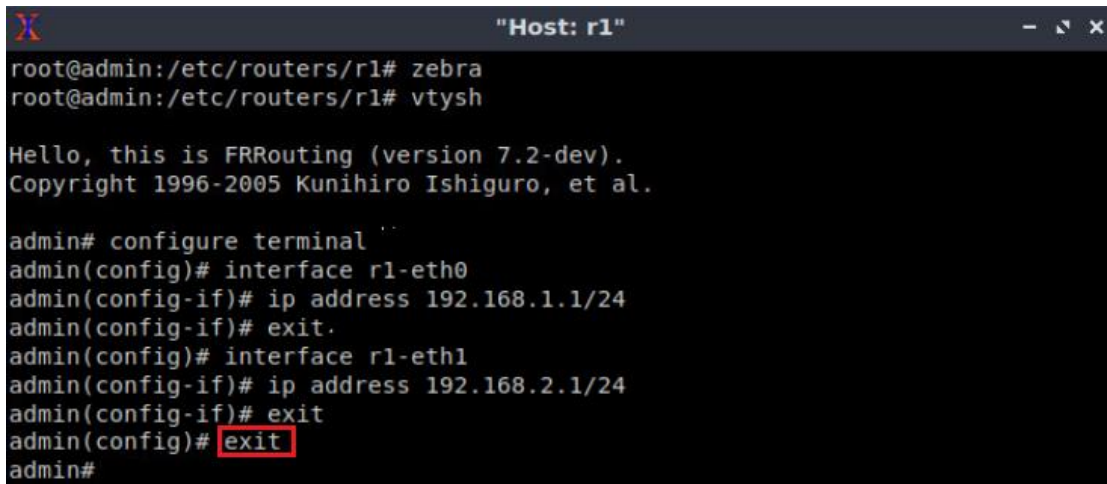
admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit.
admin(config)# interface r1-eth1
admin(config-if)# ip address 192.168.2.1/24
admin(config-if)# exit
admin(config)#
```

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
```



```

X "Host: r1"
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

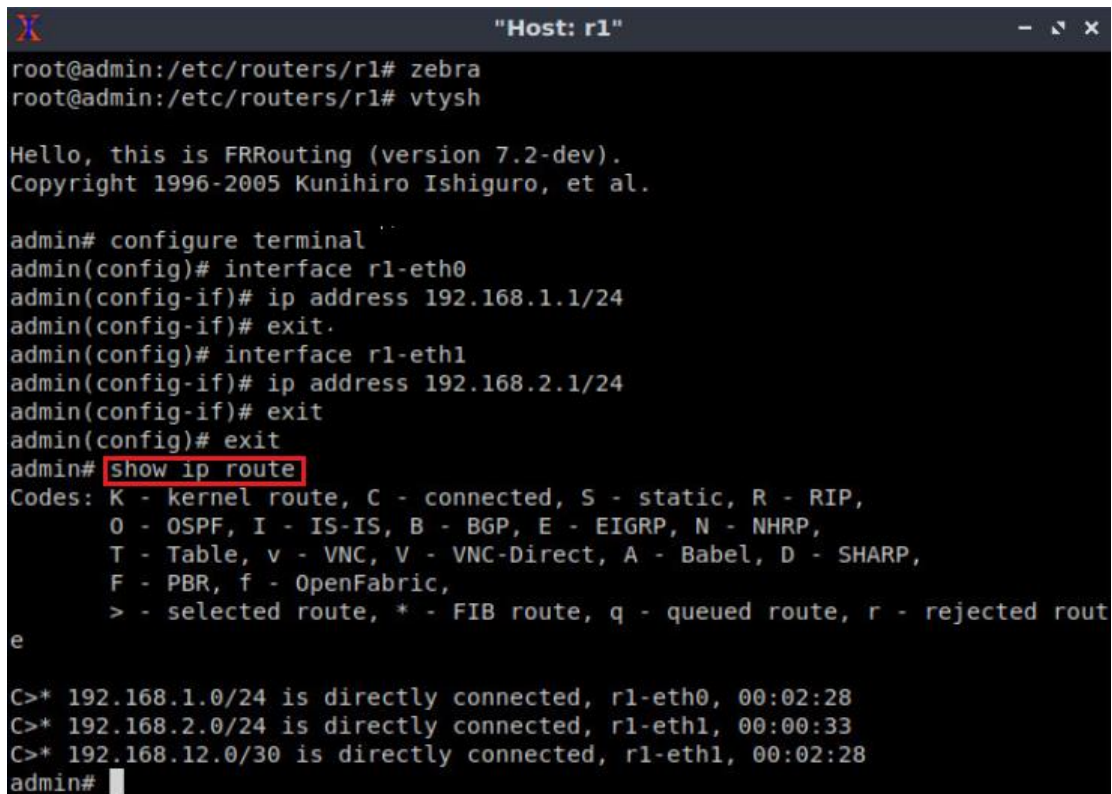
Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit
admin(config)# interface r1-eth1
admin(config-if)# ip address 192.168.2.1/24
admin(config-if)# exit
admin(config)# exit
admin#
  
```

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
```



```

X "Host: r1"
root@admin:/etc/routers/r1# zebra
root@admin:/etc/routers/r1# vtysh

Hello, this is FRRouting (version 7.2-dev).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

admin# configure terminal
admin(config)# interface r1-eth0
admin(config-if)# ip address 192.168.1.1/24
admin(config-if)# exit
admin(config)# interface r1-eth1
admin(config-if)# ip address 192.168.2.1/24
admin(config-if)# exit
admin(config)# exit
admin# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

C>* 192.168.1.0/24 is directly connected, r1-eth0, 00:02:28
C>* 192.168.2.0/24 is directly connected, r1-eth1, 00:00:33
C>* 192.168.12.0/30 is directly connected, r1-eth1, 00:02:28
admin#
  
```

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
```

```

root@admin:~# ping 192.168.2.10
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data.
64 bytes from 192.168.2.10: icmp_seq=1 ttl=63 time=0.486 ms
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=0.067 ms
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=0.058 ms
64 bytes from 192.168.2.10: icmp_seq=4 ttl=63 time=0.056 ms
^C
--- 192.168.2.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 72ms
rtt min/avg/max/mdev = 0.056/0.166/0.486/0.185 ms
root@admin:~#

```

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>.