



UNIVERSITY OF
SOUTH CAROLINA

NETWORK TOOLS AND PROTOCOLS

Lab 1: Introduction to Mininet

Document Version: **06-14-2019**



Award 1829698

“CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput
Networks for Big Science Data Transfers”

Contents

| | |
|---|----|
| Overview | 3 |
| Objectives..... | 3 |
| Lab settings | 3 |
| Lab roadmap | 3 |
| 1 Introduction to Mininet | 3 |
| 2 Invoking Mininet using the CLI..... | 5 |
| 2.1 Invoking Mininet using the default topology..... | 5 |
| 2.2 Testing connectivity | 8 |
| 3 Building and emulating a network in Mininet using the GUI | 9 |
| 3.1 Building the network topology..... | 9 |
| 3.2 Testing connectivity | 11 |
| 3.3 Automatic assignment of IP addresses | 13 |
| 3.4 Saving and loading a Mininet topology..... | 15 |
| References | 16 |

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.

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.

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: Invoking Mininet using the CLI.
3. Section 3: Building and emulating a network in Mininet using the GUI.

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.

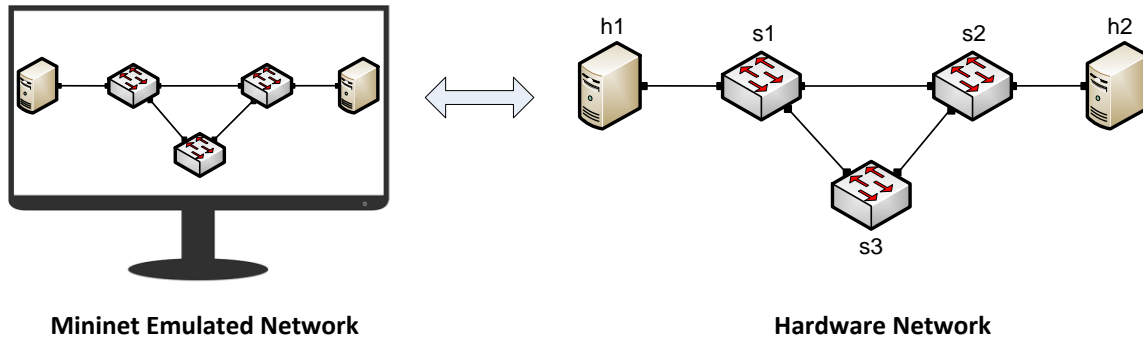


Figure 1. Hardware network vs. Mininet emulated network.

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 few 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 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.

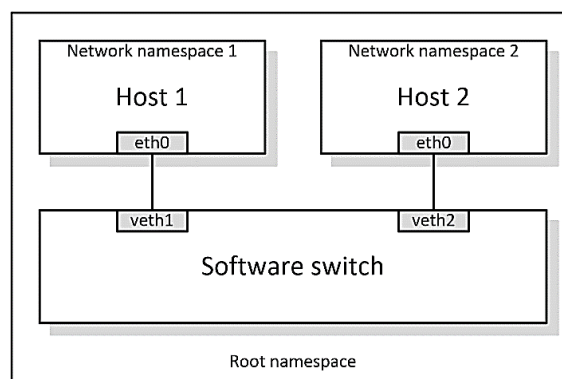


Figure 2. Network namespaces and virtual Ethernet links.

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. In a simulator environment on the other hand, applications and protocol implementations need to be ported to run within the simulator before they can be used⁴.

2 Invoking Mininet using the CLI

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

2.1 Invoking 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 `sudo mn` at the CLI. When prompted for a password, type `password` and hit enter. Note that the password will not be visible as you type it.

```
admin@admin-pc: ~
File Actions Edit View Help
admin@admin-pc: ~
admin@admin-pc:~$ sudo mn
[sudo] password for admin:
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

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.

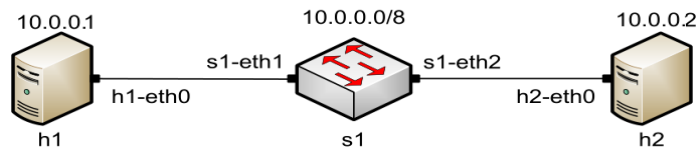


Figure 5. Mininet’s default minimal topology.

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 command `help` in the Mininet CLI:

```

admin@admin-pc: ~
mininet> help
Documented commands (type help <topic>):
=====
EOF      gterm  iperfudp  nodes      pingpair    py        switch
dpctl   help   link      noecho     pingpairfull  quit     time
dump    intfz  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

mininet>
    
```

Figure 6. Mininet’s `help` command.

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

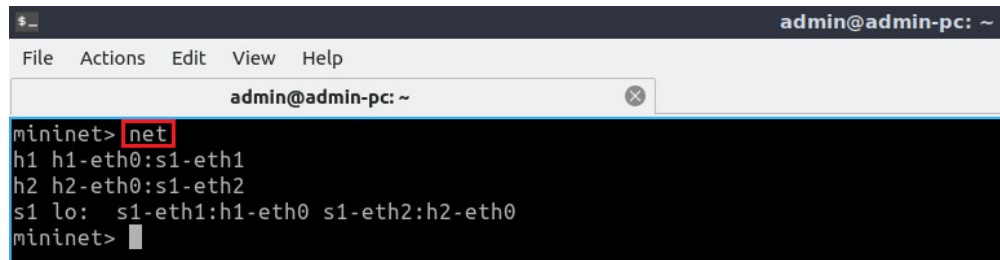
```

admin@admin-pc: ~
mininet> nodes
available nodes are:
h1 h2 s1
mininet>
    
```

Figure 7. Mininet’s `nodes` command.

The output of this command shows that there are 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 `net` in the Mininet CLI to see the available links.



```

admin@admin-pc: ~
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
mininet>

```

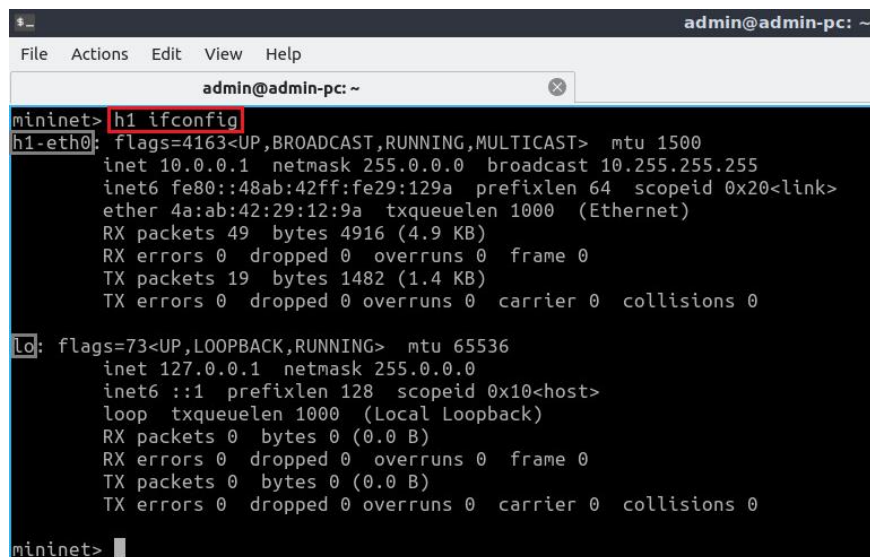
Figure 8. Mininet's `net` command.

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*.

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

Step 6. Issue the command `h1 ifconfig`.



```

admin@admin-pc: ~
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
inet6 fe80::48ab:42ff:fe29:129a prefixlen 64 scopeid 0x20<link>
ether 4a:ab:42:29:12:9a txqueuelen 1000 (Ethernet)
RX packets 49 bytes 4916 (4.9 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 19 bytes 1482 (1.4 KB)
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
mininet>

```

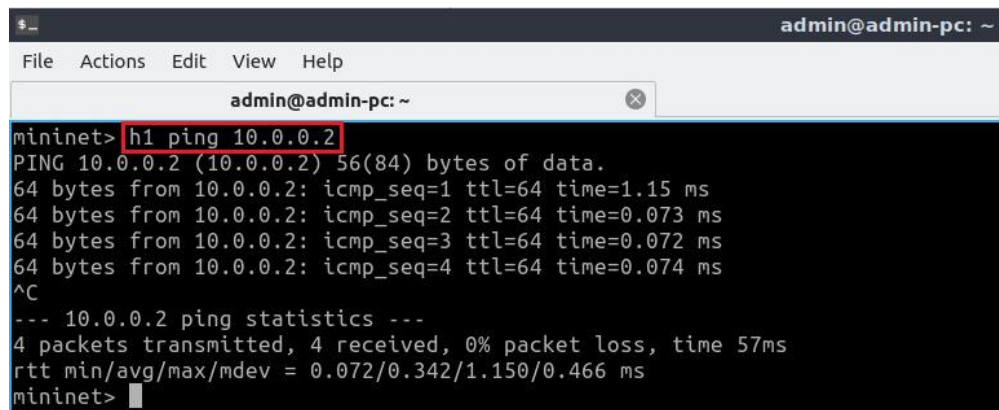
Figure 9. Output of `h1 ifconfig` command.

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 Testing 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 `h1 ping 10.0.0.2`. 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.

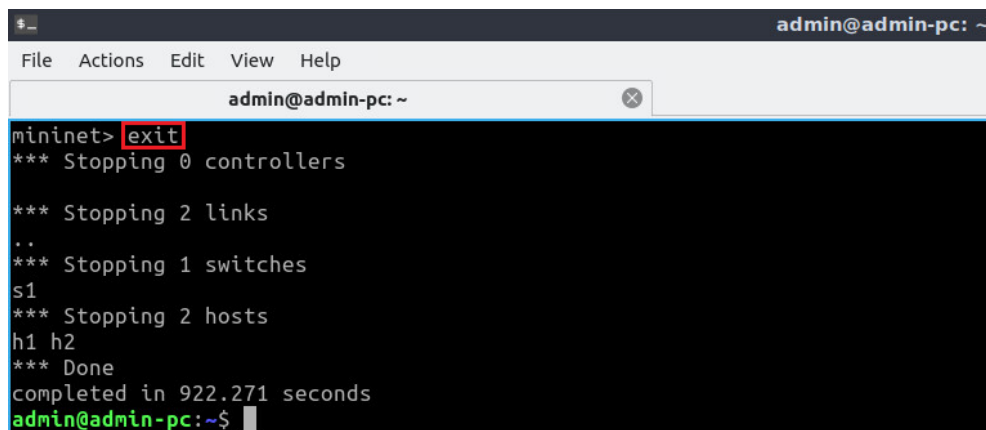


```

admin@admin-pc: ~
mininet> 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=1.15 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.072 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 57ms
rtt min/avg/max/mdev = 0.072/0.342/1.150/0.466 ms
mininet>
  
```

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

Step 2. Stop the emulation by typing `exit`.



```

admin@admin-pc: ~
mininet> exit
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 922.271 seconds
admin@admin-pc:~$
  
```

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 Building and emulating 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.

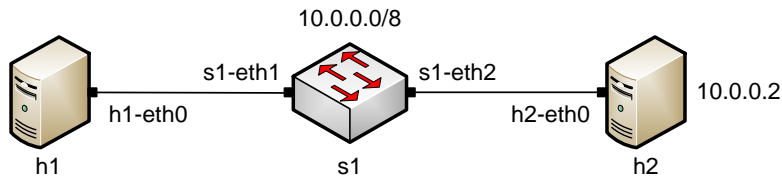


Figure 12. Lab topology.

3.1 Building 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`.

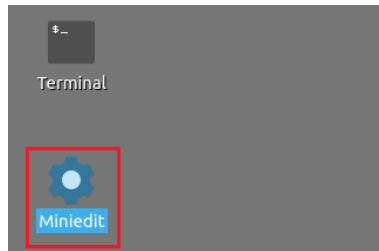


Figure 13. MiniEdit Desktop shortcut.

MiniEdit will start, as illustrated below.

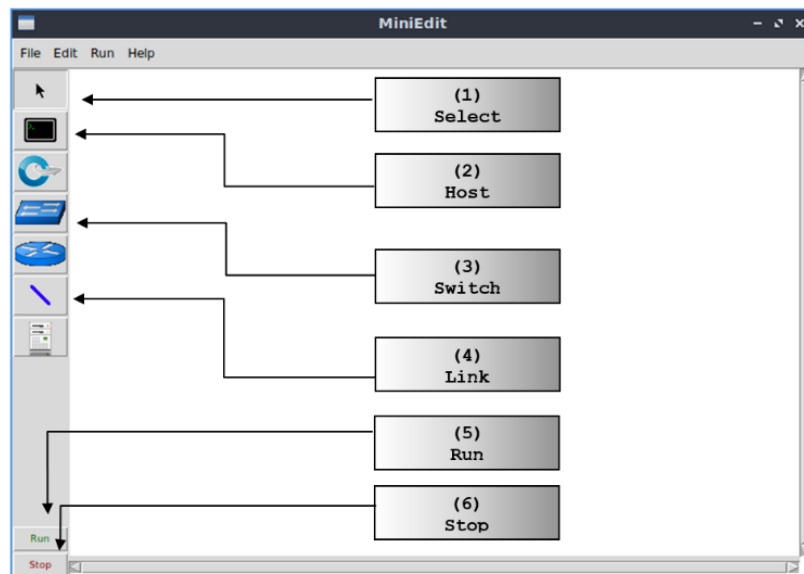


Figure 14. MiniEdit Graphical User Interface (GUI).

The main buttons 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. *Switch*: allows addition of a new 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 of Figure 12, two hosts and one switch must be deployed. Deploy these devices in MiniEdit, as shown below.

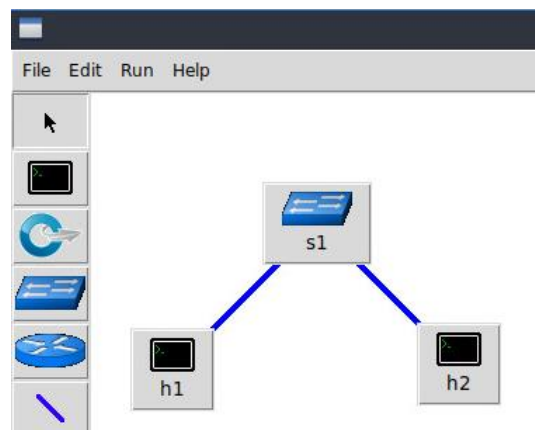


Figure 15. MiniEdit's topology.

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 at 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.

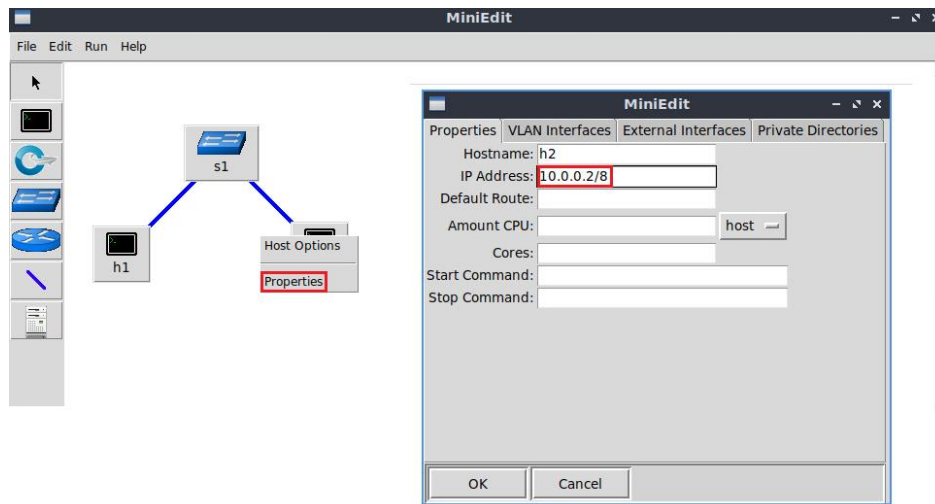


Figure 16. Configuration of a host's properties.

3.2 Testing 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.

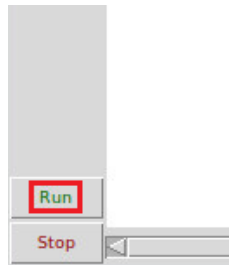


Figure 17. Starting the emulation.

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.

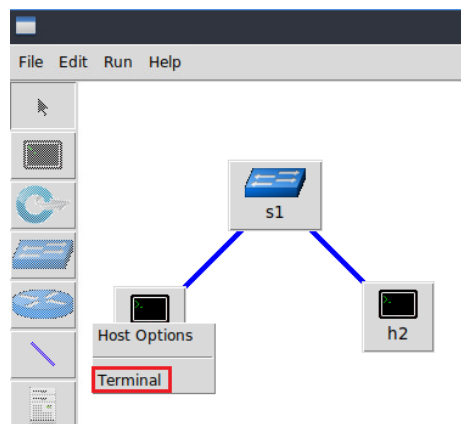


Figure 18. Opening a terminal on host h1.

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

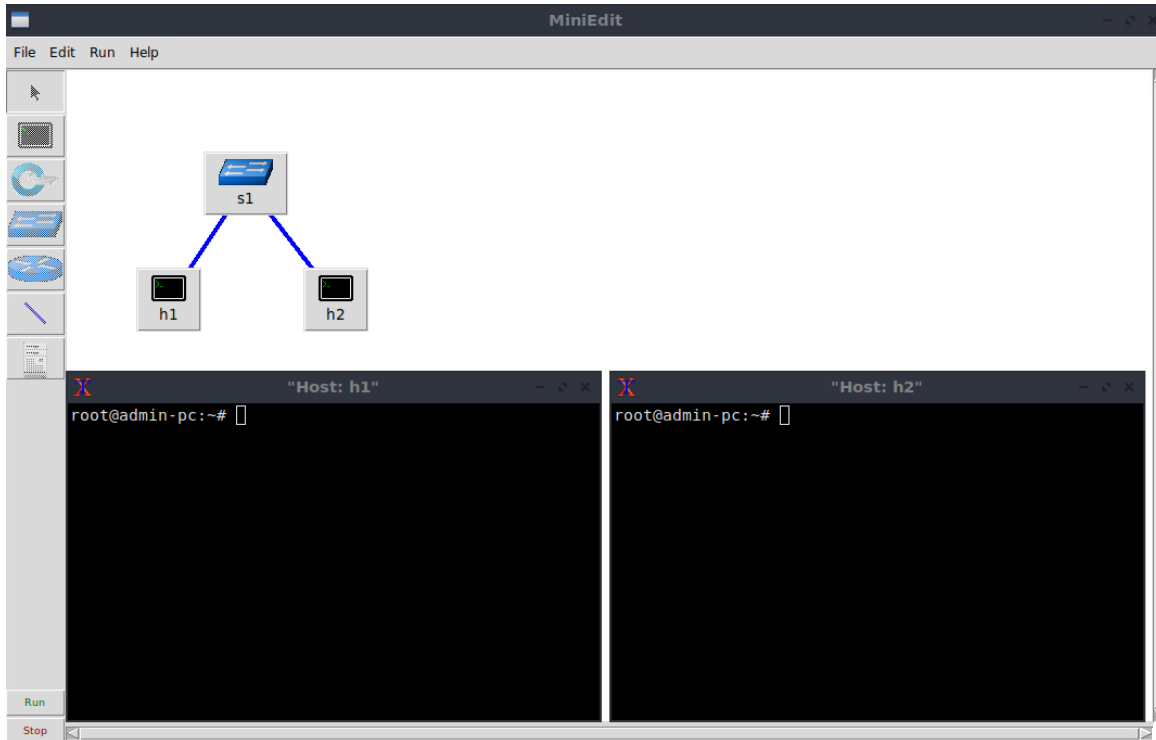


Figure 19. Terminals at host h1 and host h2.

Step 3. On host h1's terminal, type the command `ifconfig` 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.

```

root@admin-pc:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::f0d6:67ff:fe01:6041 prefixlen 64 scopeid 0x20<link>
    ether f2:d6:67:01:60:41 txqueuelen 1000 (Ethernet)
    RX packets 51 bytes 5112 (5.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 21 bytes 1678 (1.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

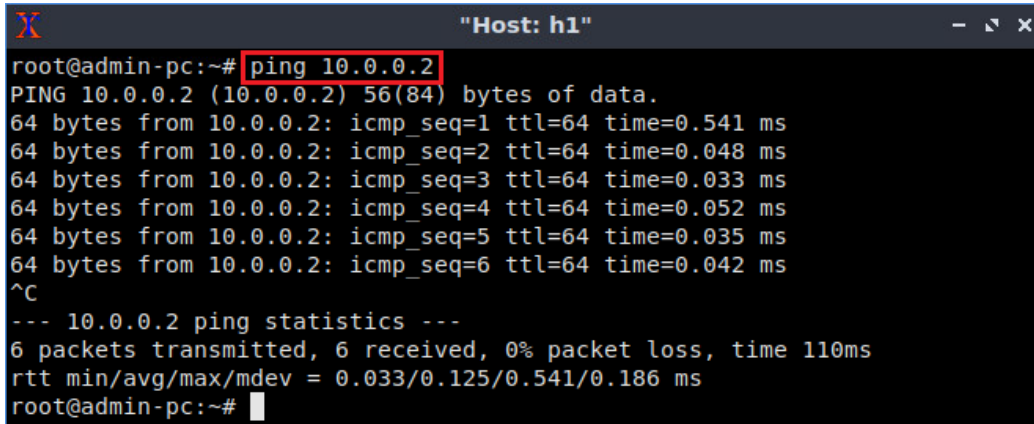
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped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

root@admin-pc:~#
    
```

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 `ping 10.0.0.2`. 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.



```

root@admin-pc:~# 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=0.541 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.042 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 110ms
rtt min/avg/max/mdev = 0.033/0.125/0.541/0.186 ms
root@admin-pc:~#

```

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.

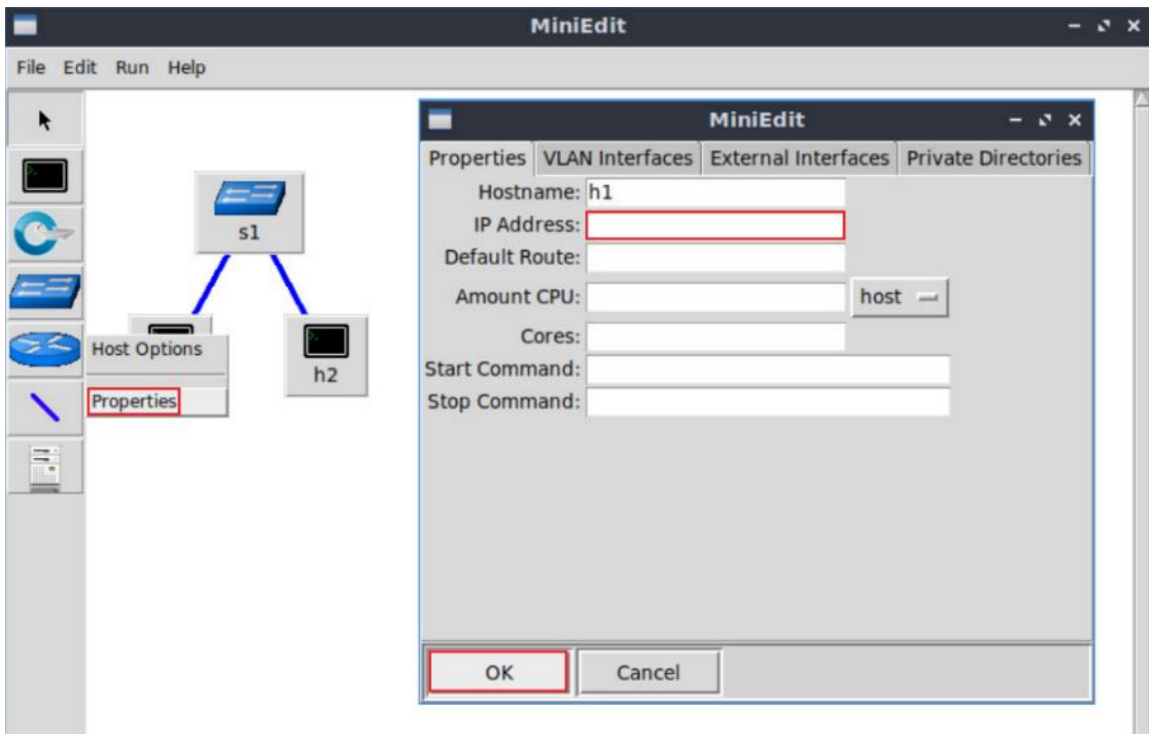


Figure 23. Host h1 properties.

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.

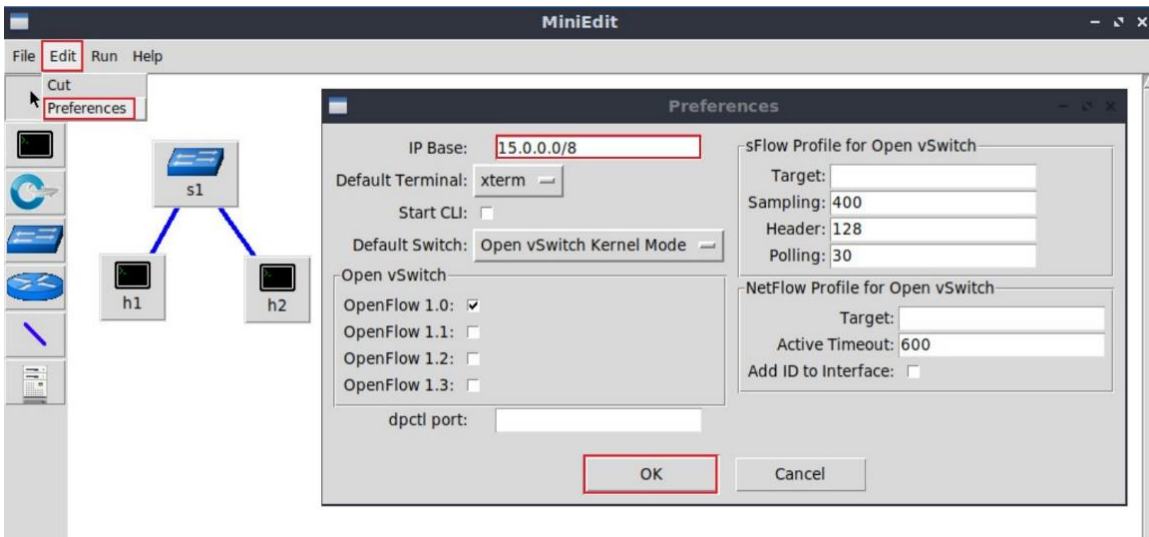


Figure 24. Modification of the IP Base (network address and prefix length).

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.

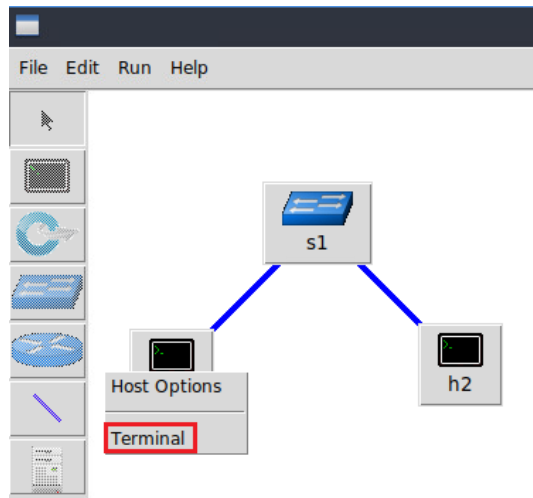


Figure 25. Opening a terminal on host h1.

Step 5. Type the command `ifconfig` 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.

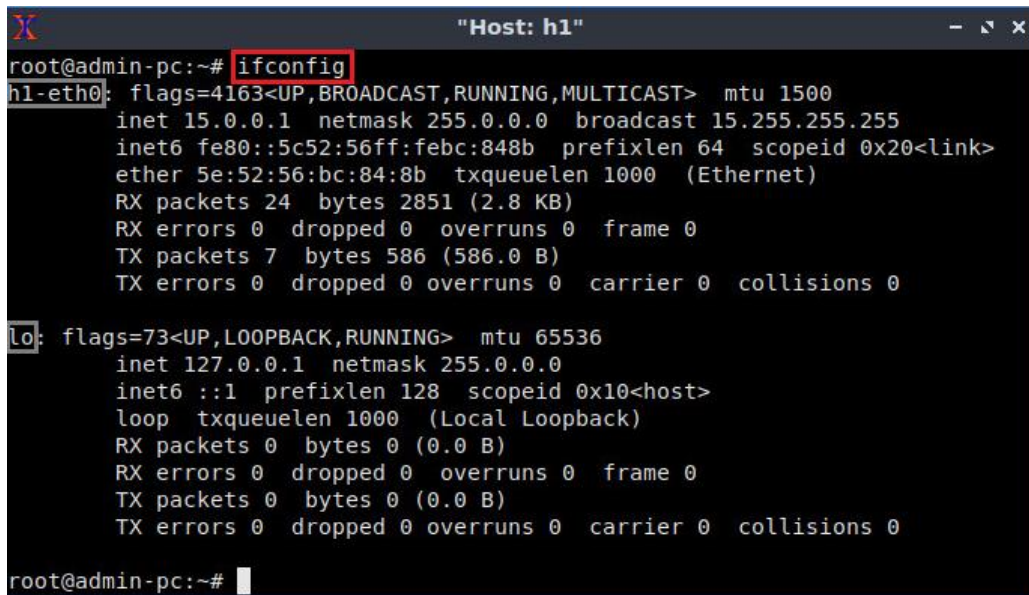


Figure 26. Output of `ifconfig` command on host h1.

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.

3.4 Saving and loading 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. To save your topology, click on *File* then *Save*. Provide a name for the topology and save on your machine.

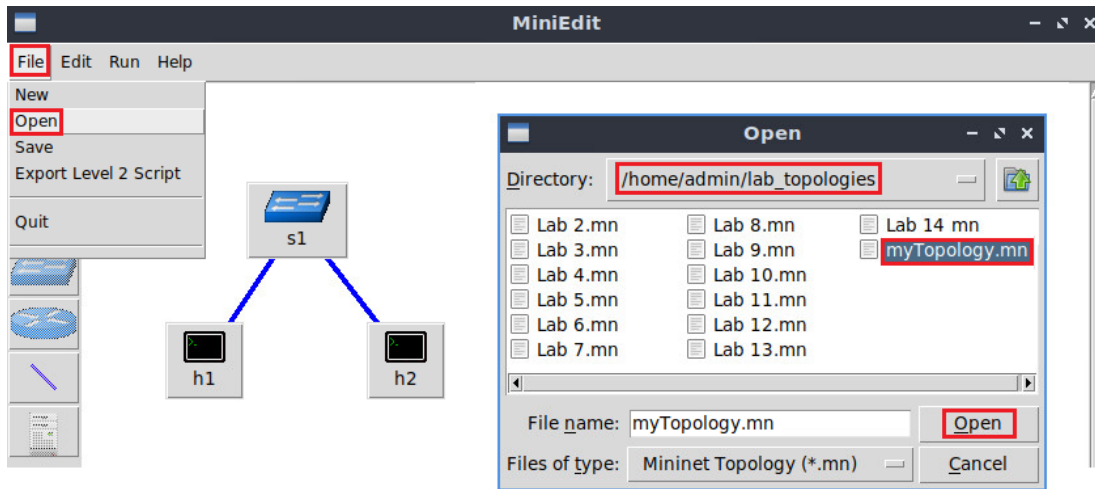


Figure 27. Saving the topology.

Step 2. To load the topology, click on *File* then *Open*. Locate the topology file and click on *Open*. The topology will be loaded again to MiniEdit.

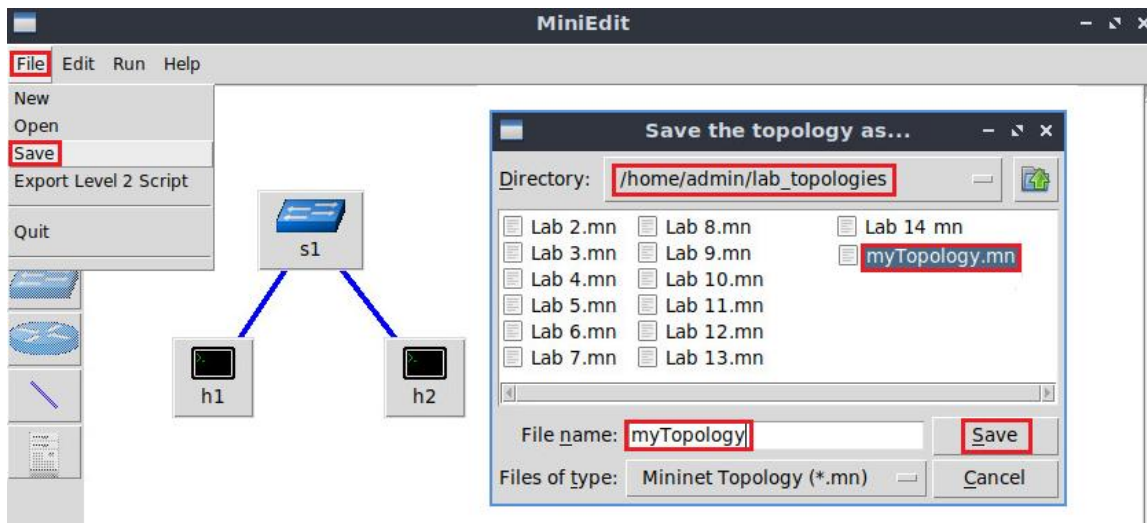


Figure 28. Opening a topology.

The upcoming labs' topologies are already built and stored in the folder */home/admin/lab_topologies* located in the Client's home directory. The *Open* dialog is used to avoid manually rebuilding each lab's topology.

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>.