

EE450: Computer Networks

Lab 1: Introduction to Mininet

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.

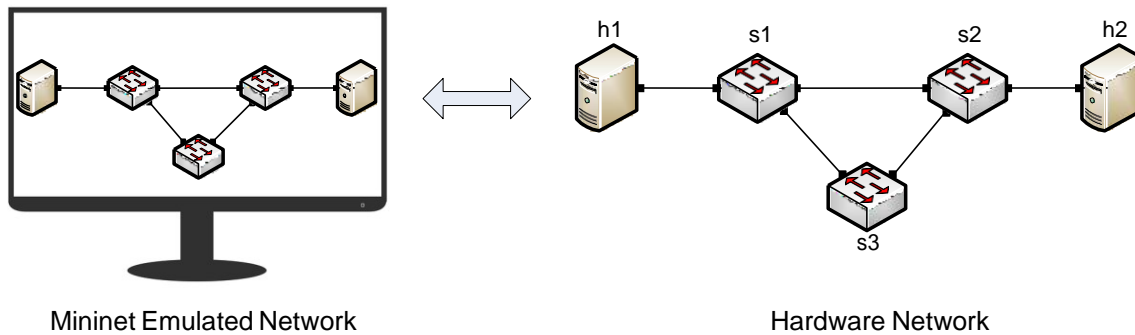


Figure 1. Hardware network vs. Mininet emulated network.

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.

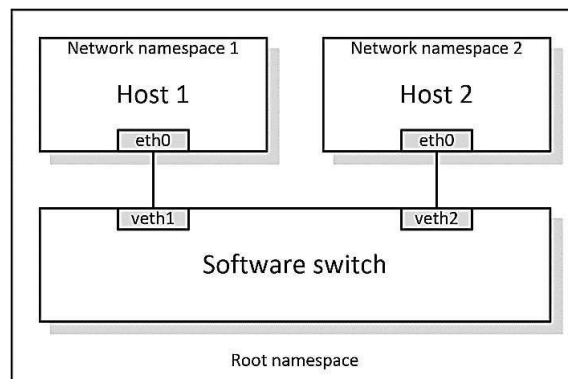


Figure 2. Network namespaces and virtual Ethernet links.

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.

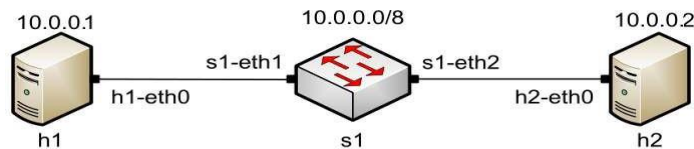


Figure 5. Mininet's default minimal topology.

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: ~
File Actions Edit View Help
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     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

mininet>
  
```

Figure 6. Mininet's `help` command.

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

```

admin@admin-pc: ~
File Actions Edit View Help
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>

```

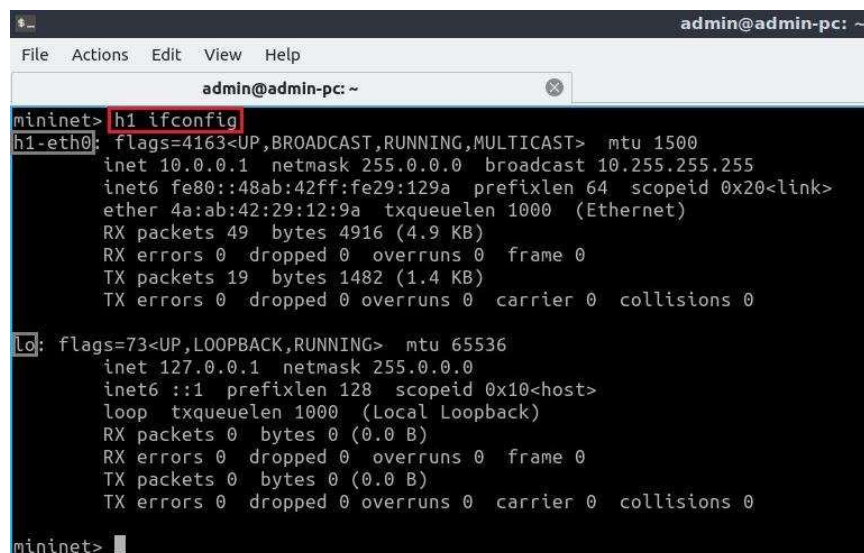
Figure 8. Mininet's `net` command.

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 interfaces *s1-eth1*.
 - c. connects to *h2-eth0* through interfaces *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>

```

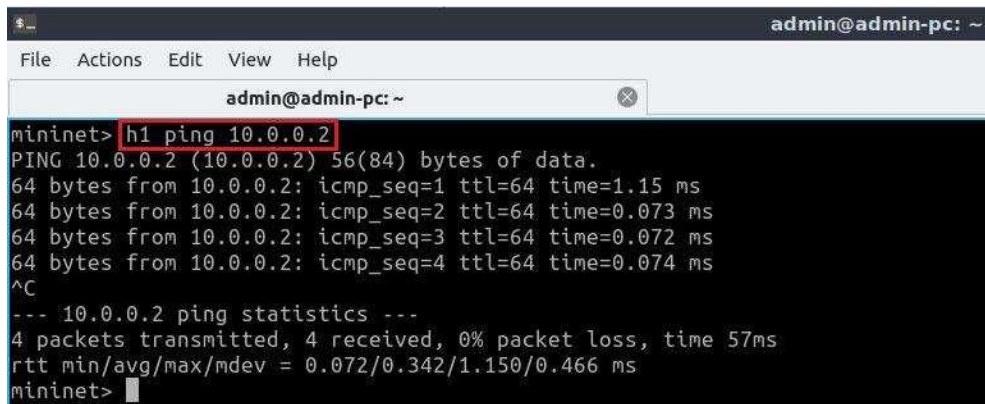
Figure 9. Output of `h1 ifconfig` command.

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.



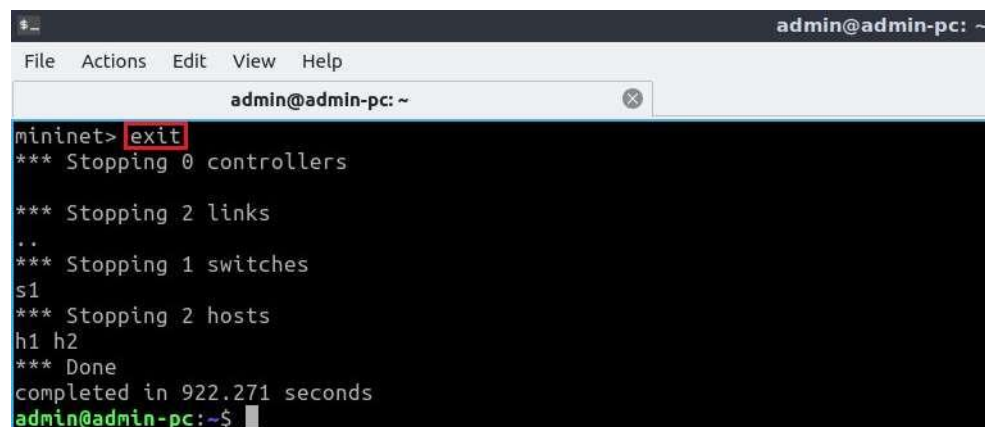
```

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.

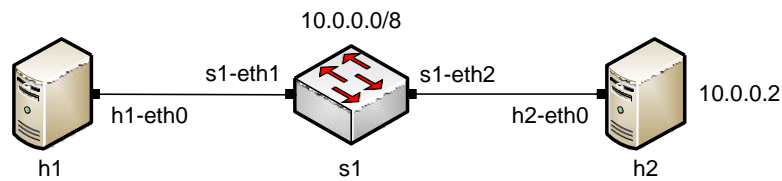


Figure 12. Lab topology.

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.

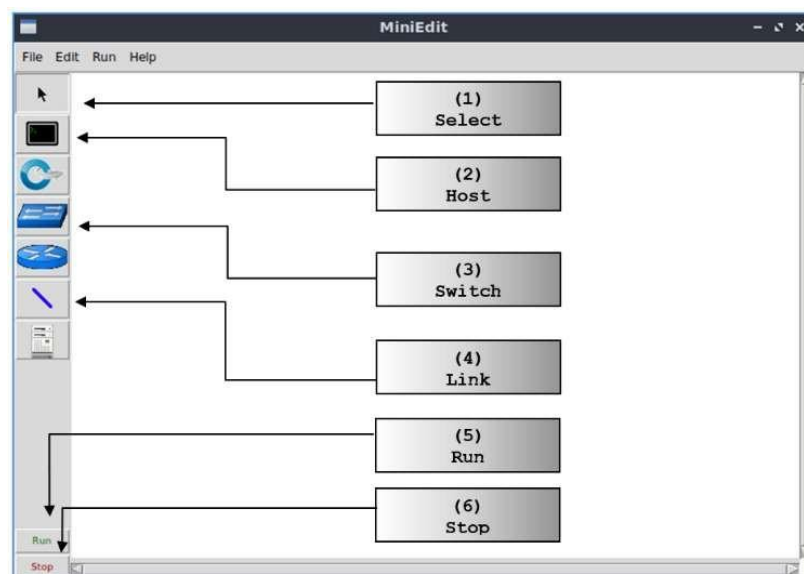


Figure 14. MiniEdit Graphical User Interface (GUI).

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.

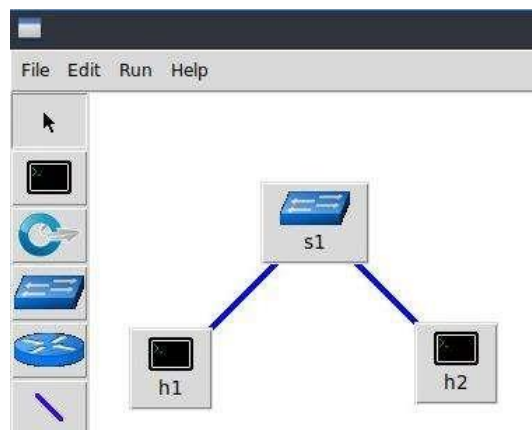


Figure 15. MiniEdit's topology.

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.

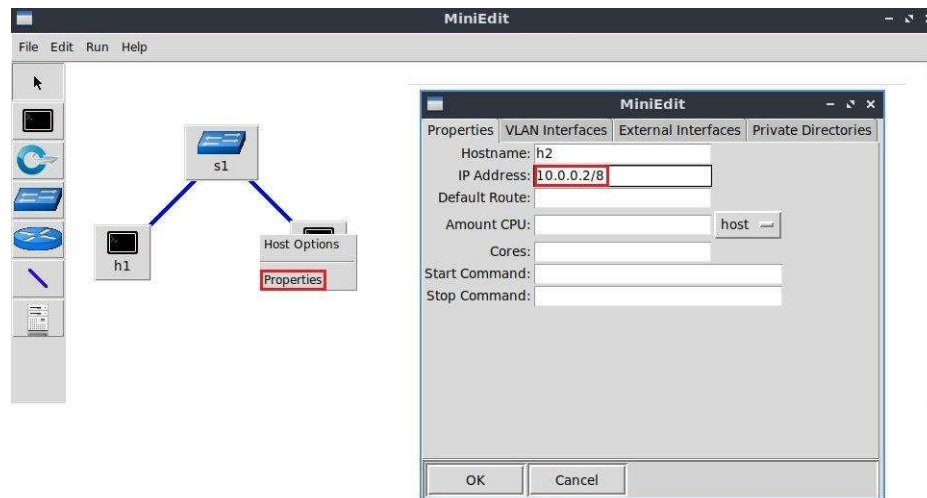


Figure 16. Configuration of a host's properties.

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.

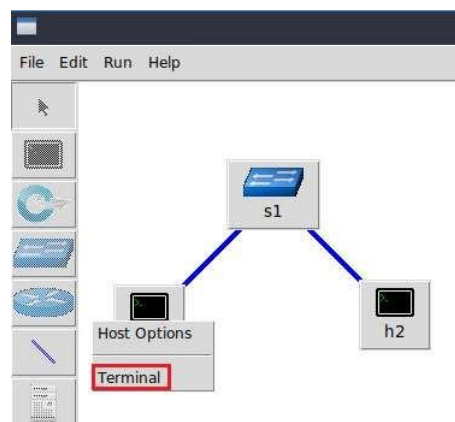


Figure 18. Opening a terminal on host h1.

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

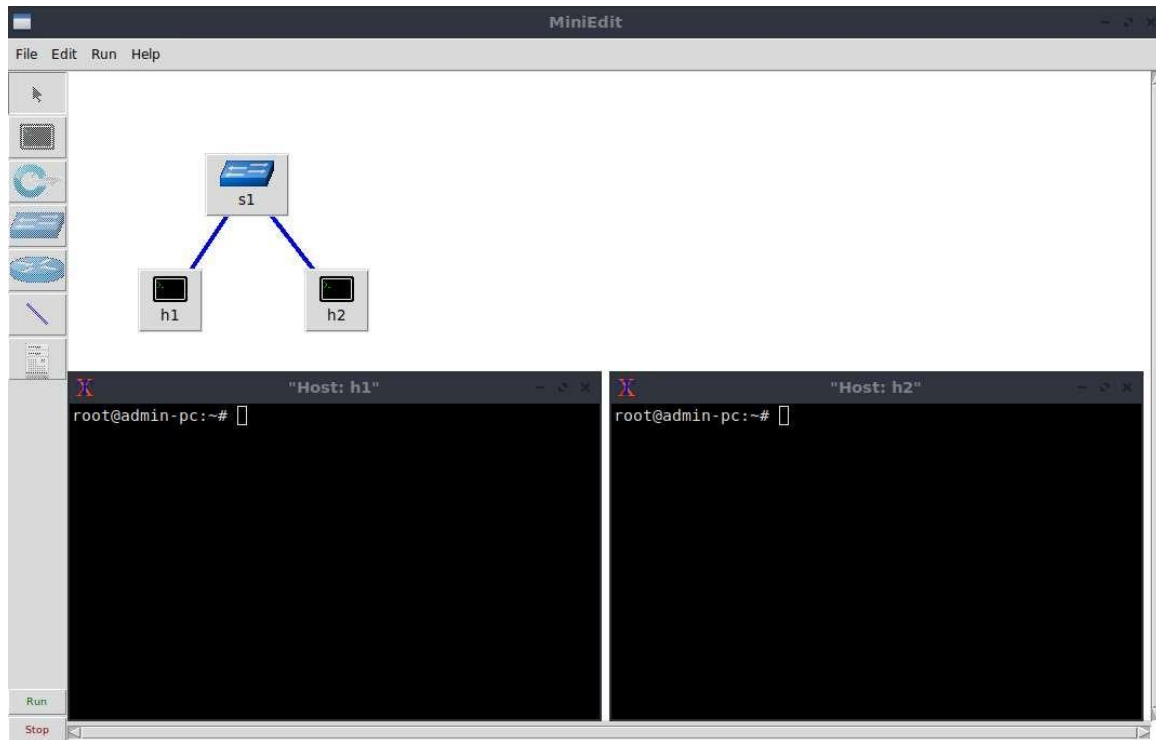


Figure 19. Terminals at host h1 and host h2.

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.

```

"Host: h1"
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

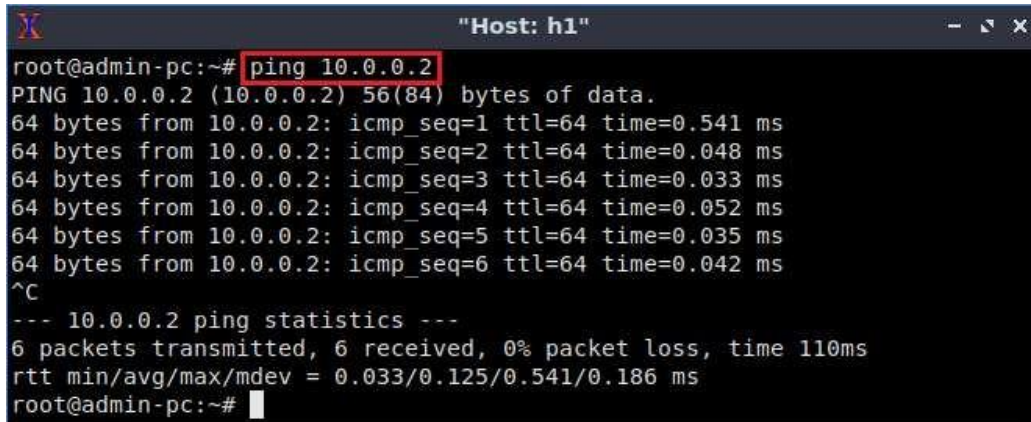
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped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

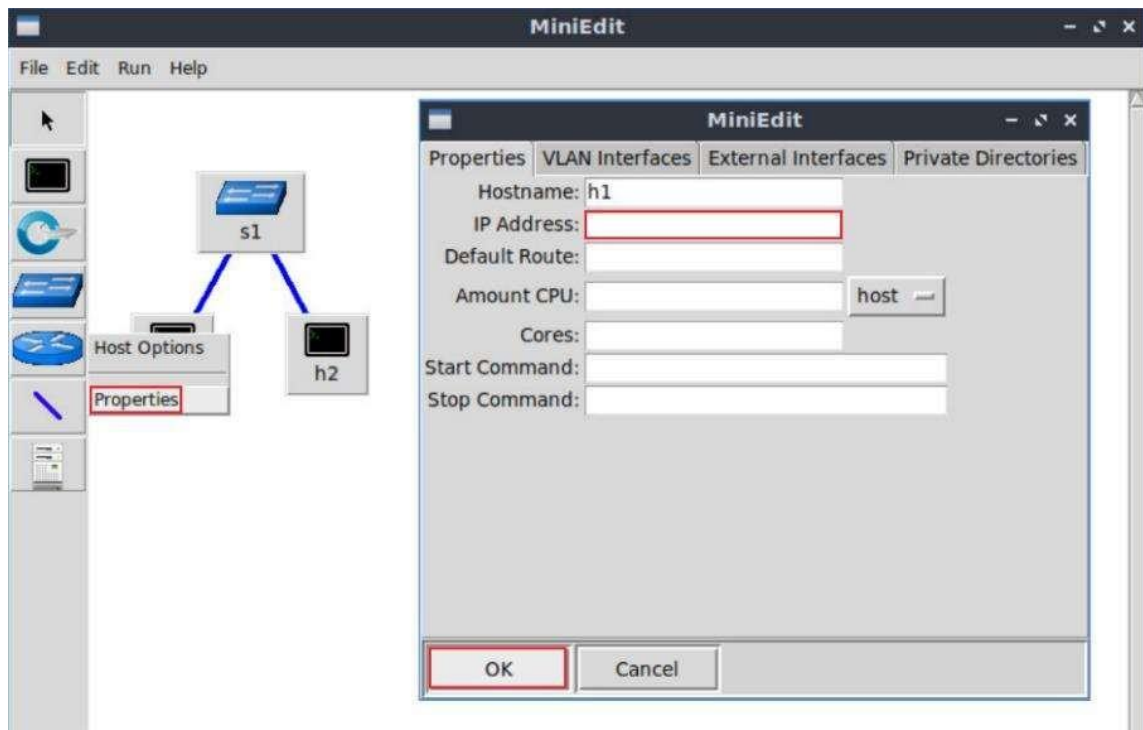


Figure 23. Host h1 properties.

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.

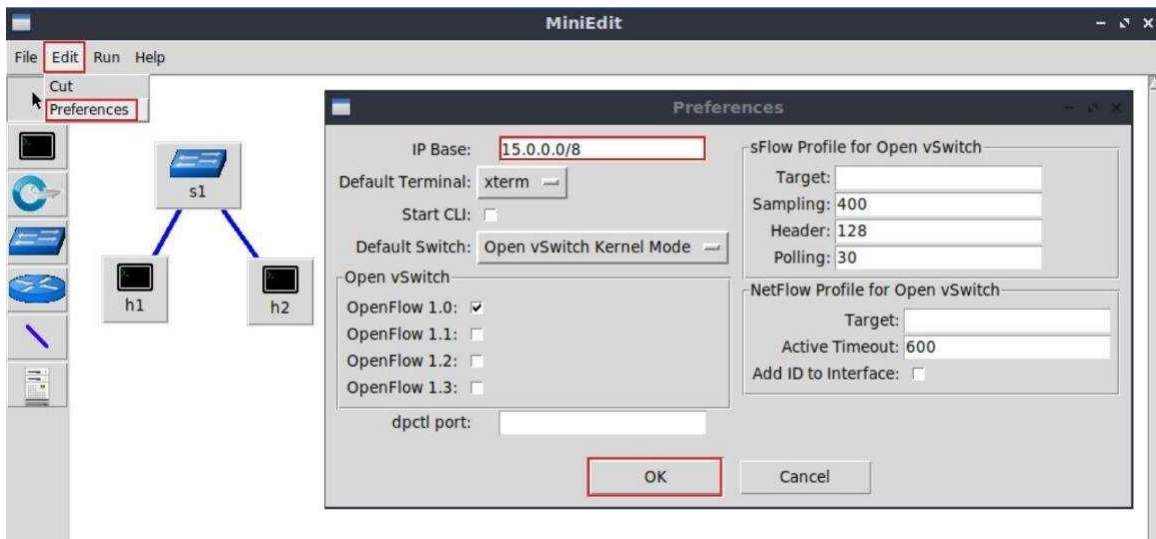


Figure 24. Modification of the IP Base (network address and prefix length).

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.

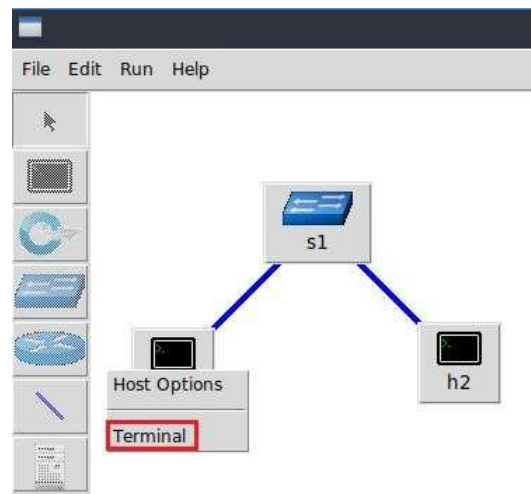


Figure 25. Opening a terminal on host h1.

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.

```

root@admin-pc:~# ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 15.0.0.1 netmask 255.0.0.0 broadcast 15.255.255.255
    inet6 fe80::5c52:56ff:febc:848b prefixlen 64 scopeid 0x20<link>
    ether 5e:52:56:bc:84:8b txqueuelen 1000 (Ethernet)
    RX packets 24 bytes 2851 (2.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7 bytes 586 (586.0 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-pc:~#

```

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.

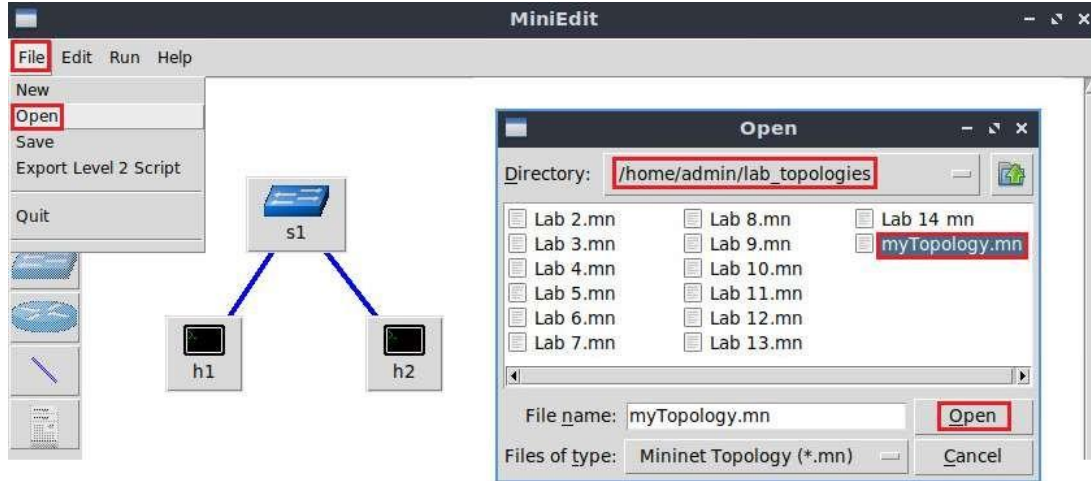


Figure 27. Saving the topology.

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.

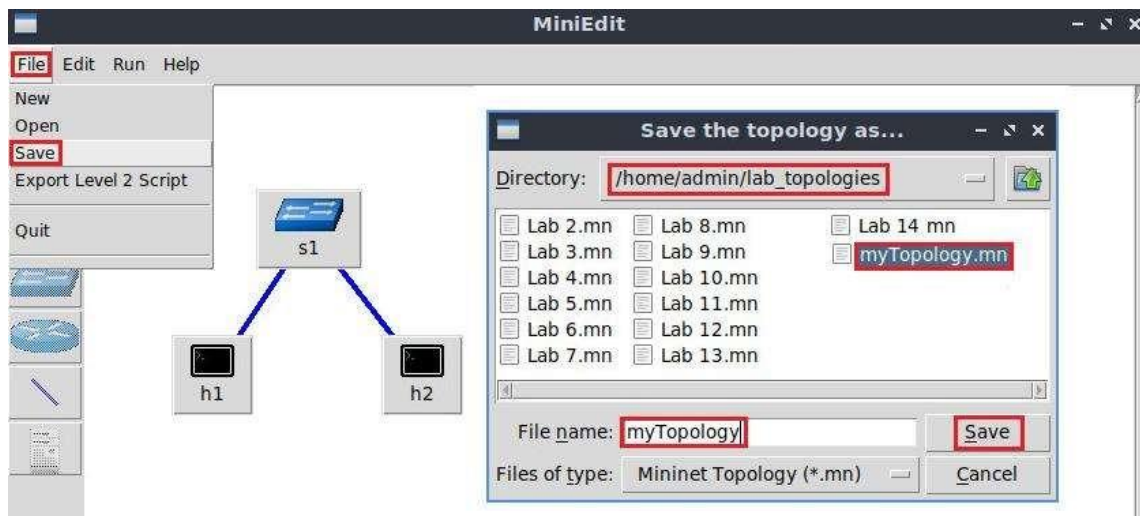


Figure 28. Opening a topology.

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>.

EE450: Computer Networks

Lab 2: Introduction to iPerf3

Contents

Overview	3
Objectives	3
Lab settings	3
Lab roadmap	3
1. Introduction to iPerf	3
2. Lab topology	4
2.1 Starting host h1 and host h2	6
3. Using iPerf3 (client and server commands)	6
3.1 Starting client and server	7
3.2 Setting transmitting time period	8
3.3 Setting time interval	9
3.4 Changing the number of bytes to transmit	10
3.5 Specifying the transport-layer protocol	11
3.6 Changing port number	13
3.7 Export results to JSON file	13
3.8 Handle one client	14
4. Plotting iPerf3 results	15
References	17

Overview

This lab briefly introduces iPerf3 and explains how it can be used to measure and test network throughput in a designed network topology. It demonstrates how to invoke both client-side and server-side options from the command line utility.

Objectives

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

1. Understand throughput and how it differs from bandwidth in network systems.
2. Create iPerf3 tests with various settings on a designed network topology.
3. Understand and analyze iPerf3's test output.
4. Visualize iPerf3's output using a custom plotting script.

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 iPerf3.
2. Section 2: Lab topology.
3. Section 3: Using iPerf3 (client and server commands).
4. Section 4: Plotting iPerf3's results.

1 Introduction to iPerf

Bandwidth is a physical property of a transmission media that depends on factors such as the construction and length of wire or fiber. To network engineers, bandwidth is the maximum data rate of a channel, a quantity measured in bits per second (bps)¹. Having a high-bandwidth link does not always guarantee high network performance. In fact, several factors may affect the performance such as latency, packet loss, jitter, and others.

In the context of a communication session between two end devices along a network path, *throughput* is the rate in bps at which the sending process can deliver bits to the receiving process. Because other sessions will be sharing the bandwidth along the network path, and because these other sessions will recur, the available throughput can fluctuate with time². Note, however, that sometimes the terms throughput and bandwidth are used interchangeably.

iPerf3 is a real-time network throughput measurement tool. It is an open source, cross-platform client-server application that can be used to measure the throughput between the two end devices. A typical iPerf3 output contains a timestamped report of the amount of data transferred and the throughput measured.

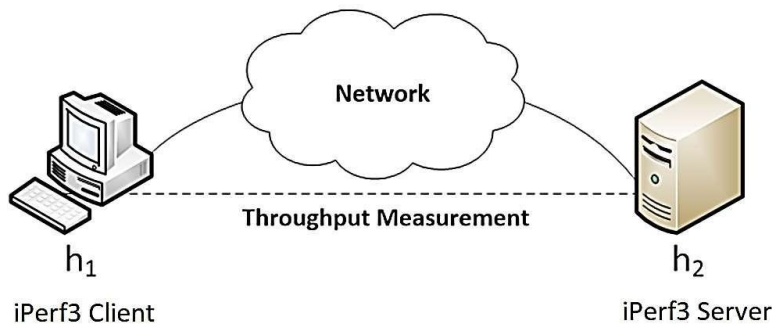


Figure 1. Throughput measurement with iPerf3.

Measuring throughput is particularly useful when experiencing network bandwidth issues such as delay, packet loss, etc. iPerf3 can operate on Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Stream Control Transmission Protocol (SCTP).

In iPerf3, the user can set *client* and *server* configurations via options and parameters and can create data flows to measure the throughput between the two end hosts in a unidirectional or bidirectional way. iPerf3 outputs a timestamped report of the amount of data transferred and the throughput measured³.

2 Lab topology

Let's get started with creating a simple Mininet topology using MiniEdit. The topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

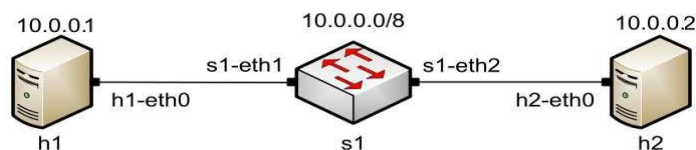


Figure 2. Mininet's default minimal topology.

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 3. MiniEdit shortcut.

Step 2. On MiniEdit's menu bar, click on *File* then *Open* to load the lab's topology. Locate the *Lab 2.mn* topology file in the default directory, */home/admin/lab_topologies*, and click on *Open*.

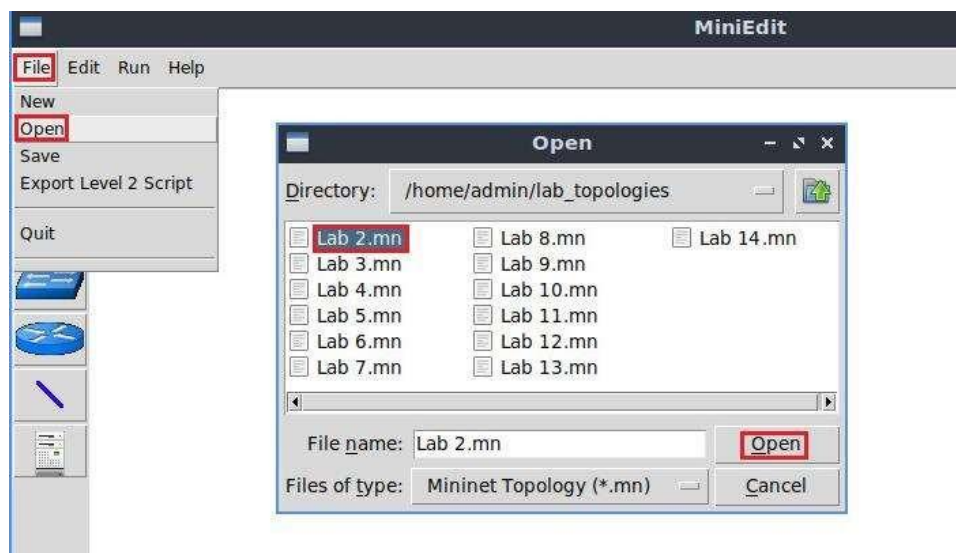


Figure 4. MiniEdit's *Open* dialog.

Step 3. Before starting the measurements between host h1 and host h2, the network must be started. Click on the *Run* button located at the bottom left of MiniEdit's window to start the emulation.



Figure 5. Running the emulation.

The above topology uses 10.0.0.0/8 which is the default network assigned by Mininet.

2.1 Starting host h1 and host h2

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

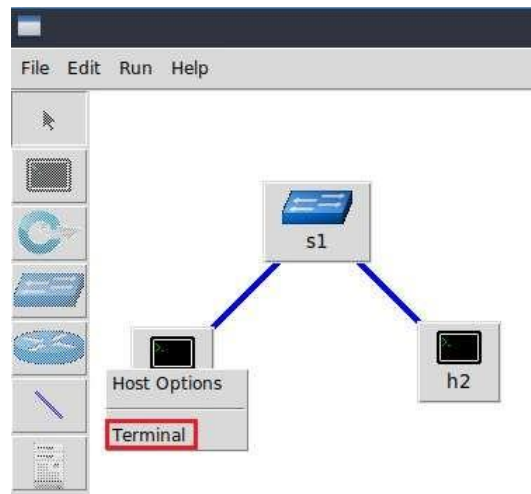


Figure 6. Opening a terminal on host h1.

Step 2. Test connectivity between the end-hosts using the `ping` command. On host h1, 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"
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=0.370 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.080 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 77ms
rtt min/avg/max/mdev = 0.078/0.152/0.370/0.126 ms
root@admin-pc:~#

```

Figure 7. Connectivity test using `ping` command.

The figure above indicates that there is connectivity between host h1 and host h2. Thus, we are ready to start the throughput measurement process.

3 Using iPerf3 (client and server commands)

Since the initial setup and configuration are done, it is time to start a simple throughput measurement. The user interacts with iPerf3 using the `iperf3` command. The basic `iperf3` syntax used on both the client and the server is as follows:

```
iperf3 [-s|-c] [ options ]
```

3.1 Starting client and server

Step 1. Hold the right-click on host h2 and select *Terminal*. This opens the terminal of host h2 and allows the execution of commands on that host.

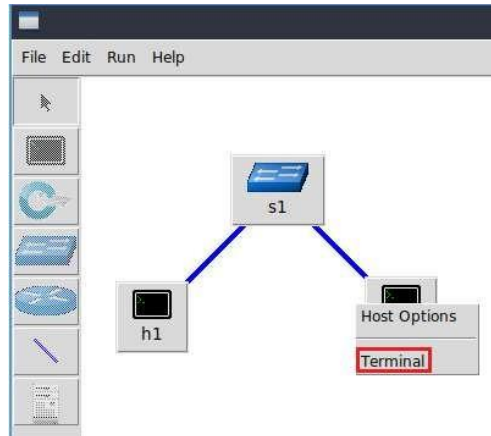


Figure 8. Opening a terminal on host h2.

Step 2. To launch iPerf3 in server mode, run the command `iperf3 -s` in host h2's terminal as shown in the figure below:

```
iperf3 -s
```



Figure 9. Host h2 running iPerf3 server.

The parameter `-s` in the command above indicates that the host is configured as a server. Now, the server is listening on port 5201 waiting for incoming connections.

Step 3. Now to launch iPerf3 in client mode, run the command `iperf3 -c 10.0.0.2` in host h1's terminal as shown in the figure below:

```
iperf3 -c 10.0.0.2
```



```

root@admin-pc:~# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 13] local 10.0.0.1 port 59414 connected to 10.0.0.2 port 5201
[ ID] Interval           Transfer     Bitrate      Retr    Cwnd
[ 13]  0.00-1.00    sec   5.18 GBytes  44.5 Gbits/sec    0   843 KBytes
[ 13]  1.00-2.00    sec   5.21 GBytes  44.7 Gbits/sec    0   1.11 MBytes
[ 13]  2.00-3.00    sec   5.20 GBytes  44.7 Gbits/sec    0   1.18 MBytes
[ 13]  3.00-4.00    sec   5.21 GBytes  44.7 Gbits/sec    0   1.24 MBytes
[ 13]  4.00-5.00    sec   5.19 GBytes  44.6 Gbits/sec    0   1.24 MBytes
[ 13]  5.00-6.00    sec   5.22 GBytes  44.8 Gbits/sec    0   1.30 MBytes
[ 13]  6.00-7.00    sec   5.24 GBytes  45.0 Gbits/sec    0   1.44 MBytes
[ 13]  7.00-8.00    sec   5.22 GBytes  44.9 Gbits/sec    0   1.44 MBytes
[ 13]  8.00-9.00    sec   5.21 GBytes  44.8 Gbits/sec    0   1.45 MBytes
[ 13]  9.00-10.00   sec   5.22 GBytes  44.8 Gbits/sec    0   1.52 MBytes
-----
[ ID] Interval           Transfer     Bitrate      Retr
[ 13]  0.00-10.00    sec   52.1 GBytes  44.8 Gbits/sec    0
[ 13]  0.00-10.04    sec   52.1 GBytes  44.6 Gbits/sec    0
sender
receiver

iperf Done.
root@admin-pc:~#

```

Figure 10. Host h1 running iPerf3 as client.

The parameter `-c` in command above indicates that host h1 is configured as a client. The parameter 10.0.0.2 is the server's (host h2) IP address. Once the test is completed, a summary report on both the client and the server is displayed containing the following data:

- *ID*: identification number of the connection.
- *Interval*: time interval to periodically report throughput. By default, the time interval is 1 second.
- *Transfer*: how much data was transferred in each time interval.
- *Bitrate*: the measured throughput in each time interval.
- *Retr*: the number of TCP segments retransmitted in each time interval. This field increases when TCP segments are lost in the network due to congestion or corruption.
- *Cwnd*: indicates the congestion windows size in each time interval. TCP uses this variable to limit the amount of data the TCP client can send before receiving the acknowledgement of the sent data.

The summarized data, which starts after the last dashed line, shows the total amount of transferred data is 52.1 Gbyte and the throughput 44.8 Gbps.

Step 4. In order to stop the server, press `Ctrl+C` in host h2's terminal. The user can see the throughput results in the server side too. The summarized data on the server is similar to that of the client side's and must be interpreted in the same way.

3.2 Setting transmitting time period

Setting the transmission time period is configured solely on the client. To change the default transmission time, apply the following steps:

Step 1. Start the iPerf3 server on host h2.

```
iperf3 -s
```



Figure 11. Host h2 running iPerf3 as server.

Step 2. Start the iPerf3 client with the `-t` option followed by the number of seconds.

```
iperf3 -c 10.0.0.2 -t 5
```

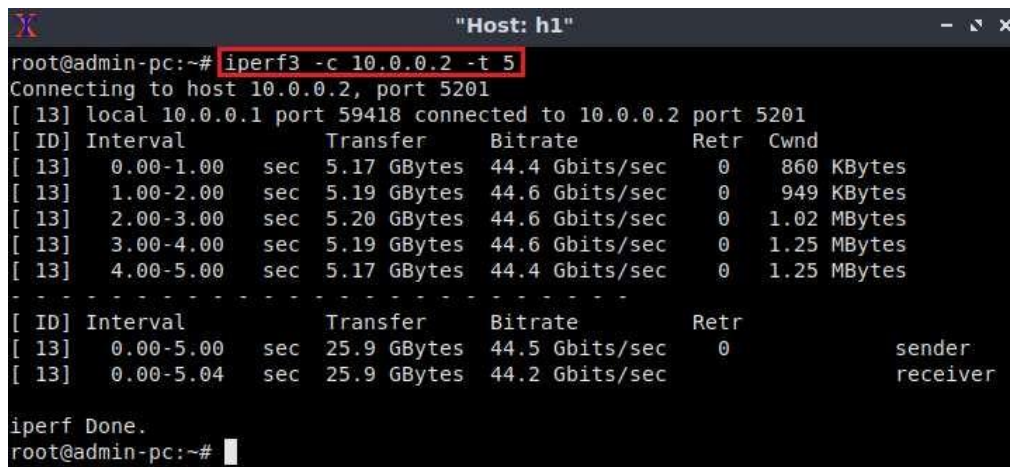


Figure 12. Host h1 transmitting for 5 seconds.

The above command starts an iPerf3 client for a 5-second time period transmitting at an average rate of 44.5 Gbps.

Step 3. In order to stop the server, press `Ctrl+c` in host h2's terminal. The user can see the throughput results in the server side too.

3.3 Setting time interval

In this test, the user will configure the client to perform a throughput test with 2-seconds reporting time interval on both the client and the server. Note the default 1-second interval period in Figure 12.

The `-i` option allows setting the reporting interval time in seconds. In this case the value should be set to 2 seconds on both the client and the server.

Step1. Setting the interval value on the server (host h2's terminal):

```
iperf3 -s -i 2
```



```

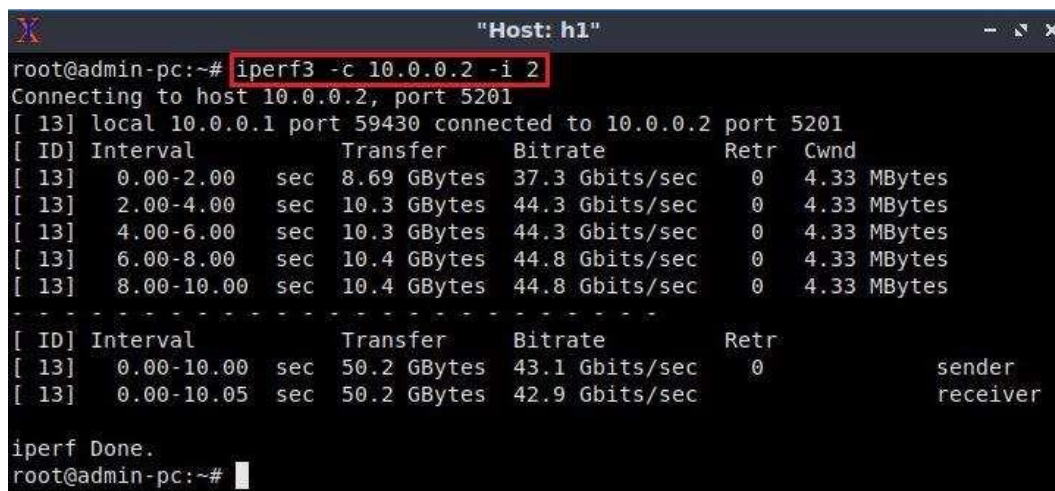
Host: h2
root@admin-pc:~# iperf3 -s -i 2
-----
Server listening on 5201
-----

```

Figure 13. Host h2 running iPerf3 as server.

Step 2. Setting the interval value on the client (host h1's terminal):

```
iperf3 -c 10.0.0.2 -i 2
```



```

Host: h1
root@admin-pc:~# iperf3 -c 10.0.0.2 -i 2
Connecting to host 10.0.0.2, port 5201
[ 13] local 10.0.0.1 port 59430 connected to 10.0.0.2 port 5201
[ ID] Interval      Transfer    Bitrate      Retr  Cwnd
[ 13] 0.00-2.00      sec 8.69 GBytes 37.3 Gbits/sec  0    4.33 MBytes
[ 13] 2.00-4.00      sec 10.3 GBytes 44.3 Gbits/sec  0    4.33 MBytes
[ 13] 4.00-6.00      sec 10.3 GBytes 44.3 Gbits/sec  0    4.33 MBytes
[ 13] 6.00-8.00      sec 10.4 GBytes 44.8 Gbits/sec  0    4.33 MBytes
[ 13] 8.00-10.00     sec 10.4 GBytes 44.8 Gbits/sec  0    4.33 MBytes
-----
[ ID] Interval      Transfer    Bitrate      Retr
[ 13] 0.00-10.00     sec 50.2 GBytes 43.1 Gbits/sec  0
[ 13] 0.00-10.05     sec 50.2 GBytes 42.9 Gbits/sec
iperf Done.
root@admin-pc:~#

```

Figure 14. Host h1 and host h2 reporting every 2 seconds.

Note that the `-i` option can be specified differently on the client and the server. For example, if the `-i` option is specified with the value 3 on the client only, then the client will be reporting every 3 seconds while the server will be reporting every second (the default `-i` value).

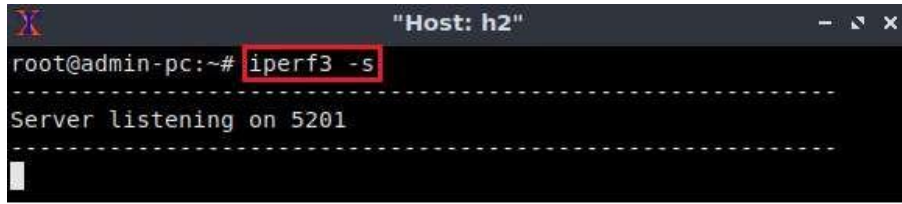
Step 3. In order to stop the server, press `Ctrl+c` in host h2's terminal. The user can see the throughput results in the server side too.

3.4 Changing the number of bytes to transmit

In this test, the client is configured to send a specific amount of data by setting the number of bytes to transmit. By default, iPerf3 performs the throughput measurement for 10 seconds. However, with this configuration, the client will keep sending packets until all the bytes specified by the user were sent.

Step 1. Type the following command on host h2's terminal to start the iPerf3 server.

```
iperf3 -s
```



```

root@admin-pc:~# iperf3 -s
-----
Server listening on 5201
-----

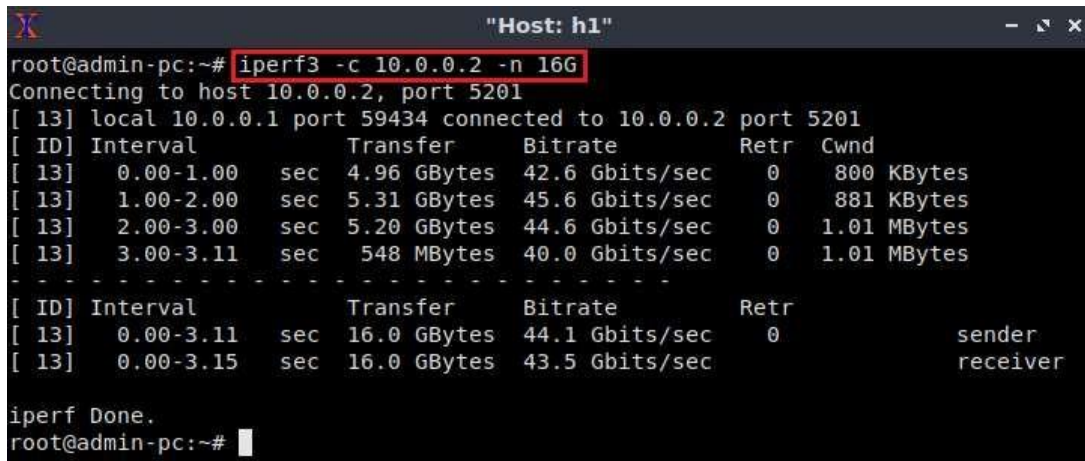
```

Figure 15. Host h2 running iPerf3 as server.

Step 2. This configuration is only set on the client (host h1's terminal) using the `-n` option as follows:

```
iperf3 -c 10.0.0.2 -n 16G
```

The `-n` option in the above command indicates the amount of data to transmit: 16 Gbytes. The user can specify other scale values, for example, `16M` is used to send 16 Mbytes.



```

root@admin-pc:~# iperf3 -c 10.0.0.2 -n 16G
Connecting to host 10.0.0.2, port 5201
[ 13] local 10.0.0.1 port 59434 connected to 10.0.0.2 port 5201
[ ID] Interval           Transfer     Bitrate      Retr    Cwnd
[ 13]  0.00-1.00       sec   4.96 GBytes  42.6 Gbits/sec    0     800 KBytes
[ 13]  1.00-2.00       sec   5.31 GBytes  45.6 Gbits/sec    0     881 KBytes
[ 13]  2.00-3.00       sec   5.20 GBytes  44.6 Gbits/sec    0    1.01 MBytes
[ 13]  3.00-3.11       sec   548 MBytes  40.0 Gbits/sec    0    1.01 MBytes
-----
[ ID] Interval           Transfer     Bitrate      Retr
[ 13]  0.00-3.11       sec   16.0 GBytes  44.1 Gbits/sec    0
[ 13]  0.00-3.15       sec   16.0 GBytes  43.5 Gbits/sec    0
iperf Done.
root@admin-pc:~#

```

Figure 16. Host h1 sending 16 Gbps of data.

Note the total time spent for sending the 16 Gbytes of data is 3.11 seconds and not the default transmitting time used by iPerf3 (10 seconds).

Step 3. In order to stop the server, press `Ctrl+c` in host h2's terminal. The user can see the throughput results in the server side too.

3.5 Specifying the transport-layer protocol

So far, the throughput measurements were conducted on the TCP protocol, which is the default configuration protocol. In order to change the protocol to UDP, the user must invoke the option `-u` on the client side. Similarly, the option `--sctp` is used for the SCTP protocol. iPerf3 automatically detects the transport-layer protocol on the server side.

Step 1. Start the iPerf3 server on host h2.

```
iperf3 -s
```



```

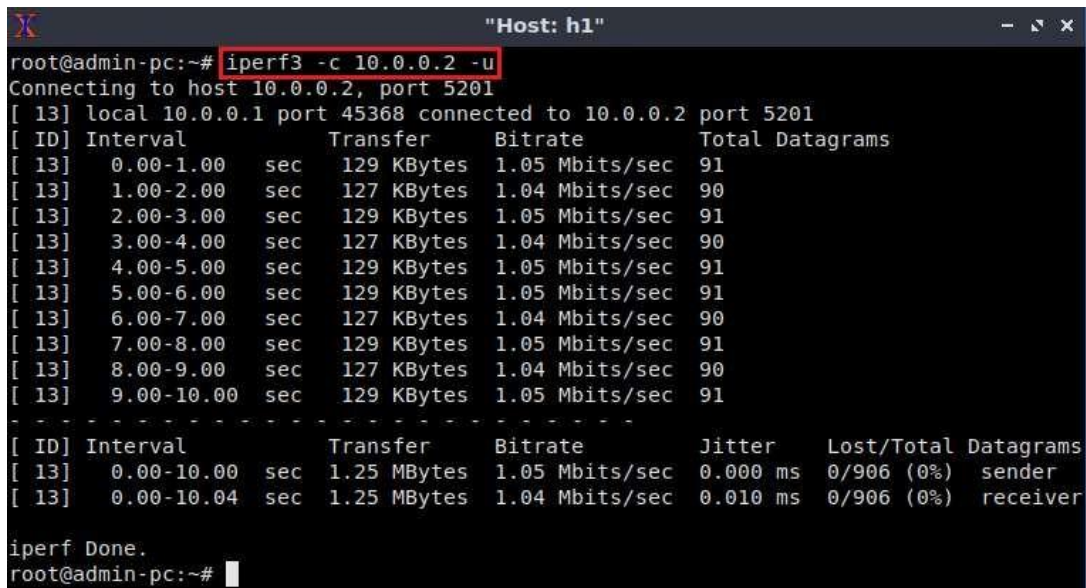
Host: h2
root@admin-pc:~# iperf3 -s
Server listening on 5201

```

Figure 17. Host h2 running iPerf3 as server.

Step 2. Specify UDP as the transport-layer protocol using the `-u` option as follows.

```
iperf3 -c 10.0.0.2 -u
```



```

Host: h1
root@admin-pc:~# iperf3 -c 10.0.0.2 -u
Connecting to host 10.0.0.2, port 5201
[ 13] local 10.0.0.1 port 45368 connected to 10.0.0.2 port 5201
[ ID] Interval          Transfer      Bitrate      Total Datagrams
[ 13] 0.00-1.00 sec      129 KBytes   1.05 Mbits/sec  91
[ 13] 1.00-2.00 sec      127 KBytes   1.04 Mbits/sec  90
[ 13] 2.00-3.00 sec      129 KBytes   1.05 Mbits/sec  91
[ 13] 3.00-4.00 sec      127 KBytes   1.04 Mbits/sec  90
[ 13] 4.00-5.00 sec      129 KBytes   1.05 Mbits/sec  91
[ 13] 5.00-6.00 sec      129 KBytes   1.05 Mbits/sec  91
[ 13] 6.00-7.00 sec      127 KBytes   1.04 Mbits/sec  90
[ 13] 7.00-8.00 sec      129 KBytes   1.05 Mbits/sec  91
[ 13] 8.00-9.00 sec      127 KBytes   1.04 Mbits/sec  90
[ 13] 9.00-10.00 sec     129 KBytes   1.05 Mbits/sec  91
-----
[ ID] Interval          Transfer      Bitrate      Jitter      Lost/Total Datagrams
[ 13] 0.00-10.00 sec     1.25 MBytes   1.05 Mbits/sec  0.000 ms    0/906 (0%) sender
[ 13] 0.00-10.04 sec     1.25 MBytes   1.04 Mbits/sec  0.010 ms    0/906 (0%) receiver
iperf Done.
root@admin-pc:~#

```

Figure 18. Host h1 sending UDP datagrams.

Once the test is completed, it will show the following summarized data:

- *ID, Interval, Transfer, Bitrate*: Same as TCP.
- *Jitter*: the difference in packet delay.
- *Lost/Total*: indicates the number of lost datagrams over the total number sent to the server (and percentage).

After the dashed lines, the summary is displayed, showing the total amount of transferred data (1.25 Mbytes) and the maximum achieved bandwidth (1.05 Mbps), over a time period of 10 seconds. The Jitter, which indicates in milliseconds (ms) the variance of time delay between data packets over a network, has a value of 0.010ms. Finally, the lost datagrams value is 0 (zero) and the total datagram which the server has received was 906, and thus, the loss rate is 0%. These values are reported on the server as well.

Step 3. In order to stop the server, press `Ctrl+c` in host h2's terminal. The user can see the throughput results in the server side too.

3.6 Changing port number

If the user wishes to measure throughput on a specific port, the `-p` option is used to configure both the client and the server to send/receive packets or datagrams on the specified port.

Step 1. Start the iPerf3 server on host h2. Use the `-p` option to specify the listening port.

```
iperf3 -s -p 3250
```

A terminal window titled "Host: h2" showing the command `iperf3 -s -p 3250` being executed. The output shows "Server listening on 3250".

Figure 19. Host h2 running iPerf3 as server on port 3250.

Step 2. Start the iPerf3 client on host h1. Use the `-p` option to specify the server's listening port.

```
iperf3 -c 10.0.0.2 -p 3250
```

A terminal window titled "Host: h1" showing the command `iperf3 -c 10.0.0.2 -p 3250` being executed. The output shows the connection to host 10.0.0.2, port 3250, and a table of throughput results.

[ID]	Interval	Transfer	Bitrate	Retr	Cwnd
[13]	0.00-1.00 sec	5.23 GBytes	44.9 Gbits/sec	0	1.02 MBytes
[13]	1.00-2.00 sec	5.17 GBytes	44.4 Gbits/sec	0	1.02 MBytes
[13]	2.00-3.00 sec	5.18 GBytes	44.5 Gbits/sec	0	1.07 MBytes
[13]	3.00-4.00 sec	5.17 GBytes	44.4 Gbits/sec	0	1.18 MBytes
[13]	4.00-5.00 sec	5.18 GBytes	44.5 Gbits/sec	0	1.51 MBytes
[13]	5.00-6.00 sec	5.21 GBytes	44.8 Gbits/sec	0	1.51 MBytes
[13]	6.00-7.00 sec	5.22 GBytes	44.8 Gbits/sec	0	1.58 MBytes
[13]	7.00-8.00 sec	5.23 GBytes	44.9 Gbits/sec	0	1.66 MBytes
[13]	8.00-9.00 sec	5.21 GBytes	44.8 Gbits/sec	0	1.83 MBytes
[13]	9.00-10.00 sec	5.26 GBytes	45.2 Gbits/sec	0	1.92 MBytes

[ID]	Interval	Transfer	Bitrate	Retr	sender	receiver
[13]	0.00-10.00 sec	52.1 GBytes	44.7 Gbits/sec	0		
[13]	0.00-10.04 sec	52.1 GBytes	44.5 Gbits/sec			

iperf Done.
root@admin-pc:~#

Figure 20. Host h2 running on port 3250.

Step 3. In order to stop the server, press `Ctrl+C` in host h2's terminal. The user can see the throughput results in the server side too.

3.7 Export results to JSON file

JSON (JavaScript Object Notation) is a lightweight data-interchange format. iPerf3 allows exporting the test results to a JSON file, which makes it easy for other applications to parse the file and interpret the results (e.g. plot the results).

Step 1. Start the iPerf3 server on host h2.

```
iperf3 -s
```



Figure 21. Host h2 running iPerf3 as server.

Step 2. Start the iPerf3 client on host h1. Specify the `-J` option to display the output in JSON format.

```
iperf3 -c 10.0.0.2 -J
```

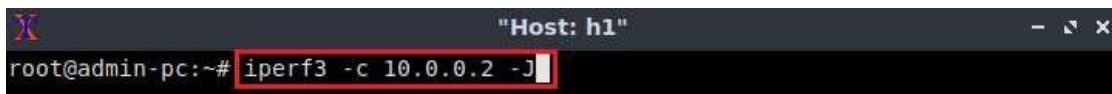


Figure 22. Host h1 using `-J` to output JSON to standard output (*stdout*).

The `-J` option outputs JSON text to the screen through standard output (*stdout*) after the test is done (10 seconds by default). It is often useful to export the output to a file that can be parsed later by other programs. This can be done by redirecting the standard output to a file using the redirection operator in Linux `>`.

```
iperf3 -c 10.0.0.2 -J > test_results.json
```

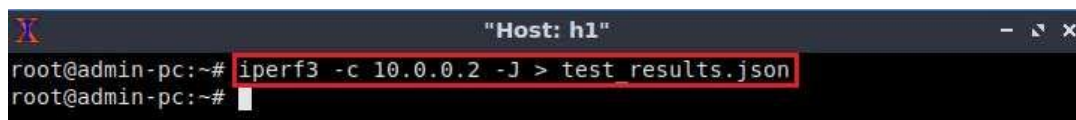


Figure 23. Host h1 using `-J` to output JSON and redirecting *stdout* to file.

After creating the JSON file, the `ls` command is used to verify that the file is created. The `cat` command can be used to display the file's contents.

Step 3. In order to stop the server, press `Ctrl+C` in host h2's terminal. The user can see the throughput results in the server side too.

3.8 Handle one client

By default, an iPerf3 server keeps listening to incoming connections. To allow the server to handle one client and then stop, the `-1` option is added to the server.

Step 1. Start the iPerf3 server on host h2. Use the `-1` option to accept only one client.

```
iperf3 -s -1
```



```

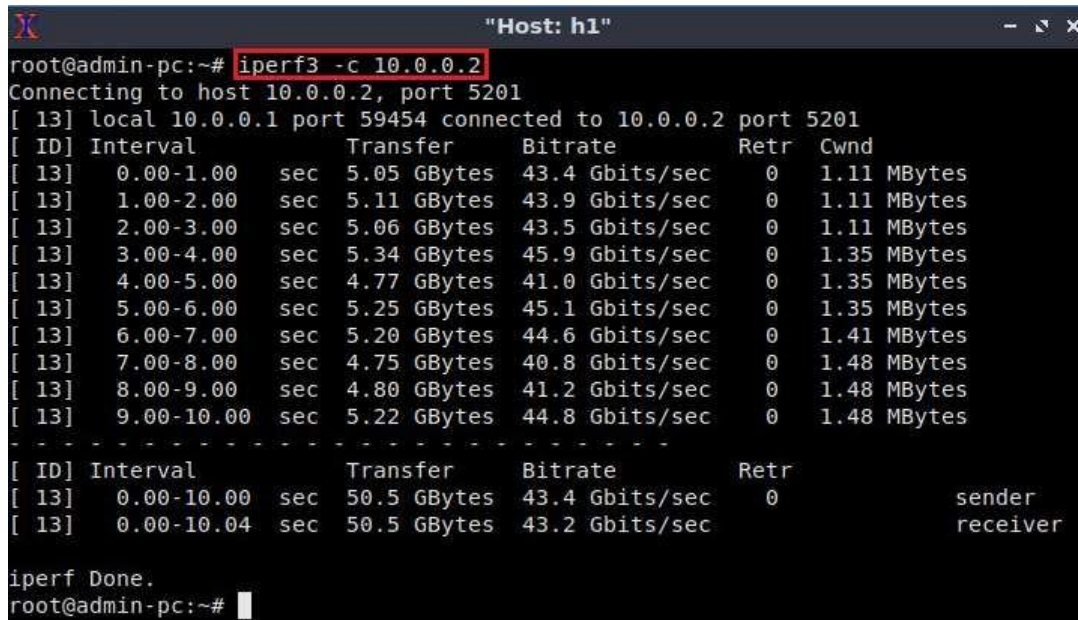
Host: h2
root@admin-pc:~# iperf3 -s -1
-----
Server listening on 5201
-----

```

Figure 24. Host h2 running a server with one connection only.

Step 2. Start the iPerf3 client on host h1.

```
iperf3 -c 10.0.0.2
```



```

Host: h1
root@admin-pc:~# iperf3 -c 10.0.0.2
Connecting to host 10.0.0.2, port 5201
[ 13] local 10.0.0.1 port 59454 connected to 10.0.0.2 port 5201
[ ID] Interval           Transfer     Bitrate      Retr  Cwnd
[ 13]  0.00-1.00      sec   5.05 GBytes  43.4 Gbits/sec    0   1.11 MBytes
[ 13]  1.00-2.00      sec   5.11 GBytes  43.9 Gbits/sec    0   1.11 MBytes
[ 13]  2.00-3.00      sec   5.06 GBytes  43.5 Gbits/sec    0   1.11 MBytes
[ 13]  3.00-4.00      sec   5.34 GBytes  45.9 Gbits/sec    0   1.35 MBytes
[ 13]  4.00-5.00      sec   4.77 GBytes  41.0 Gbits/sec    0   1.35 MBytes
[ 13]  5.00-6.00      sec   5.25 GBytes  45.1 Gbits/sec    0   1.35 MBytes
[ 13]  6.00-7.00      sec   5.20 GBytes  44.6 Gbits/sec    0   1.41 MBytes
[ 13]  7.00-8.00      sec   4.75 GBytes  40.8 Gbits/sec    0   1.48 MBytes
[ 13]  8.00-9.00      sec   4.80 GBytes  41.2 Gbits/sec    0   1.48 MBytes
[ 13]  9.00-10.00     sec   5.22 GBytes  44.8 Gbits/sec    0   1.48 MBytes
-----
[ ID] Interval           Transfer     Bitrate      Retr
[ 13]  0.00-10.00      sec   50.5 GBytes  43.4 Gbits/sec    0
[ 13]  0.00-10.04      sec   50.5 GBytes  43.2 Gbits/sec    0
-----
iperf Done.
root@admin-pc:~#

```

Figure 25. Host h1 running an iPerf3 client.

After this test is finished, the server stops immediately.

4 Plotting iPerf3 results

In section 3.7, iPerf3's result was exported to a JSON file to be processed by other applications. A script called `plot_iperf.sh` is installed and configured on the Client's machine. It accepts a JSON file as input and generates PDF files plotting several variables produced by iPerf3.

Step 1. Start the iPerf3 server on host h2.

```
iperf3 -s
```




```

Host: h2
root@admin-pc:~# iperf3 -s
-----
Server listening on 5201
-----

```

Figure 26. Host h2 running iPerf3 as server.

Step 2. Start the iPerf3 client on host h1. Specify the `-J` option to produce the output in JSON format and redirect the output to the file `test_results.json`. Any data previously stored in this file will be replaced with current output as the `>` operator is being used here.

```
iperf3 -c 10.0.0.2 -J > test_results.json
```



```

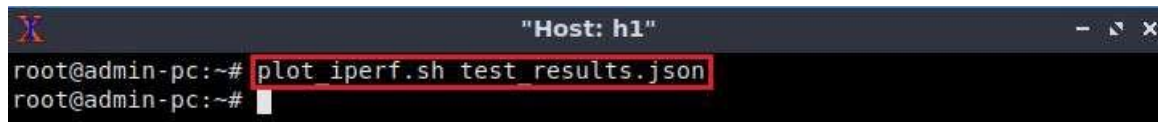
Host: h1
root@admin-pc:~# iperf3 -c 10.0.0.2 -J > test_results.json
root@admin-pc:~#

```

Figure 27. Host h1 using `-J` to output JSON and redirecting `stdout` to file.

Step 3. To generate the output for iPerf3's JSON file run the following command:

```
plot_iperf.sh test_results.json
```



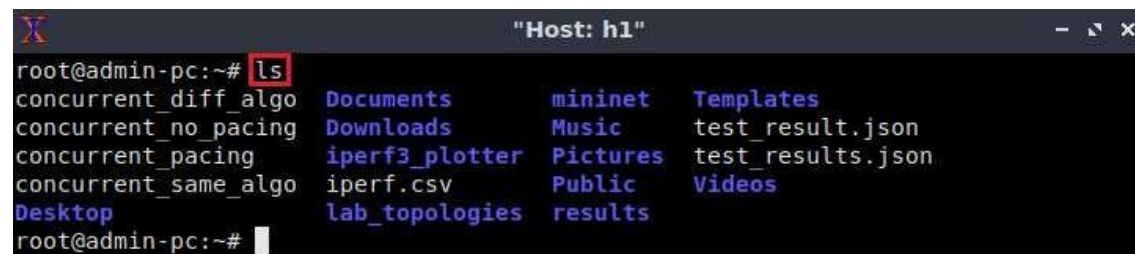
```

Host: h1
root@admin-pc:~# plot_iperf.sh test_results.json
root@admin-pc:~#

```

Figure 28. `plot_iperf.sh` script generating output results.

This plotting script generates PDF files for the following fields: congestion window (*cwnd.pdf*), retransmits (*retransmits.pdf*), Round-Trip Time (*RTT.pdf*), Round-Trip Time variance (*RTT_Var.pdf*), throughput (*throughput.pdf*), maximum transmission unit (*MTU.pdf*), bytes transferred (*bytes.pdf*). The plotting script also generates a CSV file (*1.dat*) which can be used by other applications. These files are stored in a directory *results* created in the same directory where the script was executed as shown in the figure below.



```

Host: h1
root@admin-pc:~# ls
concurrent_diff_algo  Documents      mininet      Templates
concurrent_no_pacing Downloads      Music        test_result.json
concurrent_pacing     iperf3_plotter Pictures       test_results.json
concurrent_same_algo  iperf.csv     Public       Videos
Desktop               lab_topologies results
root@admin-pc:~#

```

Figure 29. Listing the current directory's contents using the `ls` command.

Step 4. Navigate to the results folder using the `cd` command.

```
cd results/
```

```

Host: h1
root@admin-pc:~# cd results/
root@admin-pc:~/results#

```

Figure 30. Entering the results directory using the `cd` command.

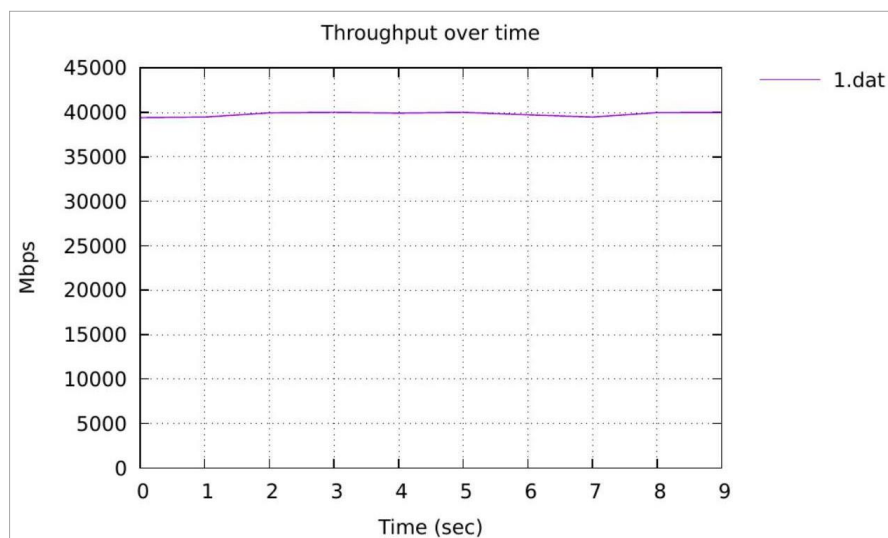
Step 5. To open any of the generated files, use the `xdg-open` command followed by the file name. For example, to open the `throughput.pdf` file, use the following command:

```
xdg-open throughput.pdf
```

```

Host: h1
root@admin-pc:~/results# xdg-open throughput.pdf

```

Figure 31. Opening the *throughput.pdf* file using `xdg-open`.Figure 32. *throughput.pdf* output.

Step 6. In order to stop the server, press `Ctrl+c` in host h2's terminal. The user can see the throughput results in the server side too.

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

References

1. A. Tanenbaum, D. Wetherall, "Computer networks," 5th Edition, Prentice Hall, 2011.
2. J. Kurose, K. Ross, "Computer networking, a top-down approach," 7th Edition, Pearson, 2017.
3. Invoking Iperf3 [Online]. Available: <https://software.es.net/iperf/invoking.html>.