5718 Campus Group 4: Lab 0 Mininet Walkthrough

Shaopu Zhang, Yibo Peng, Kaifa Lu

0. Weblink to group repository:

- https://github.com/Abe1pp/Computer-Communicate-24-summer/

1. Briefly define/explain the following terms:

- ***Linux shell & root shell***: Linux shell refers to an interface that allows users to interact with the Linux operating system by executing commands whose prompt usually ends with a "$", while root shell refers to a shell session with full administrative privileges and its command shell usually ends with a "#".
- ***git***: A distributed version control system used to manage and track changes in source code during software development.
- ***sudo***: sudo stands for "superuser do", which allows a permitted user to run programs and execute command prompts with the security privileges of another user, typically acting as the role of the superuser.
- ***apt***: A package management system used by Debian and its derivatives like Ubuntu to handle the installation, update, and removal of software packages.
- ***ifconfig***: ifconfig stands for "interface configuration", typically used to configure, manage, and query network interface parameters from the command line in the Linux operating system.
- ***arp command & arp protocol***: arp command is used to view or modify the ARP (Address Resolution Protocol) cache that stores IP addresses and their resolved Ethernet addresses, while arp protocol is used within a LAN (Local Area Network) to associate the physical address (like MAC address) with a given network (IP) address.
- ***network interface***: A physical or virtual component that connects a computer to a private or public network, typically equipped with a specific network address, which enables a computer to communicate with other devices over a network like LAN or the internet.
- ***Linux***: A family of open-source operating systems based on Unix or Linux kernel, which is widely used for servers, desktops, embedded systems, and supercomputers.
- ***Command Line Interface (CLI)***: A text-based interface used to interact with software or operating systems by typing and executing commands in the terminal or console for performing different tasks.
- ***ping***: A diagnostic tool used to test the accessibility of devices on a network by sending ICMP "echo request" packets and waiting for replies, which measures the round-trip time for packets sent and received from the origin host to the destination computer.

2. Part 1: Everyday Mininet Usage

a. Under "Interact with Hosts and Switches"

i. Sketch the default topology. Label the kernel switch, reference controller, and hosts, and show the links between hosts.

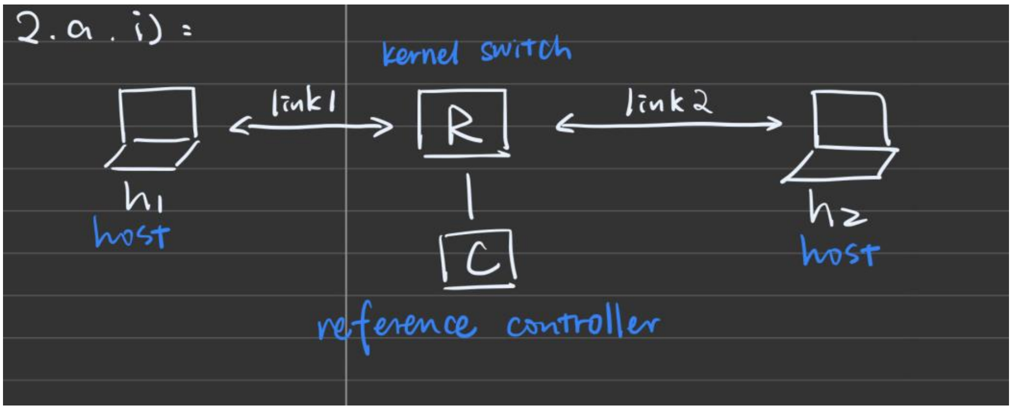
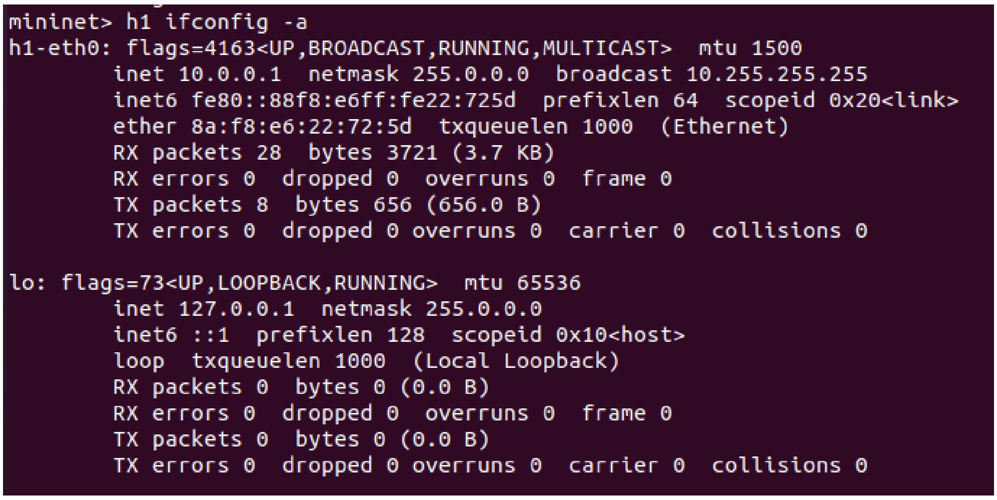


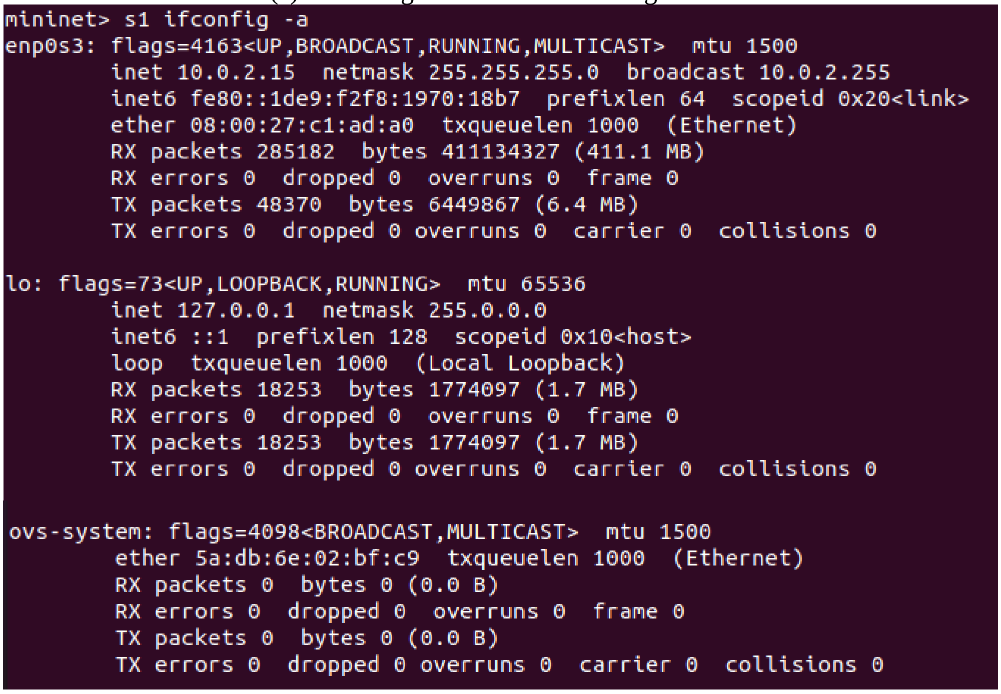
Fig. 1. The sketch of the default topology in Mininet

ii. Take screenshots of the output of the ifconfig commands.

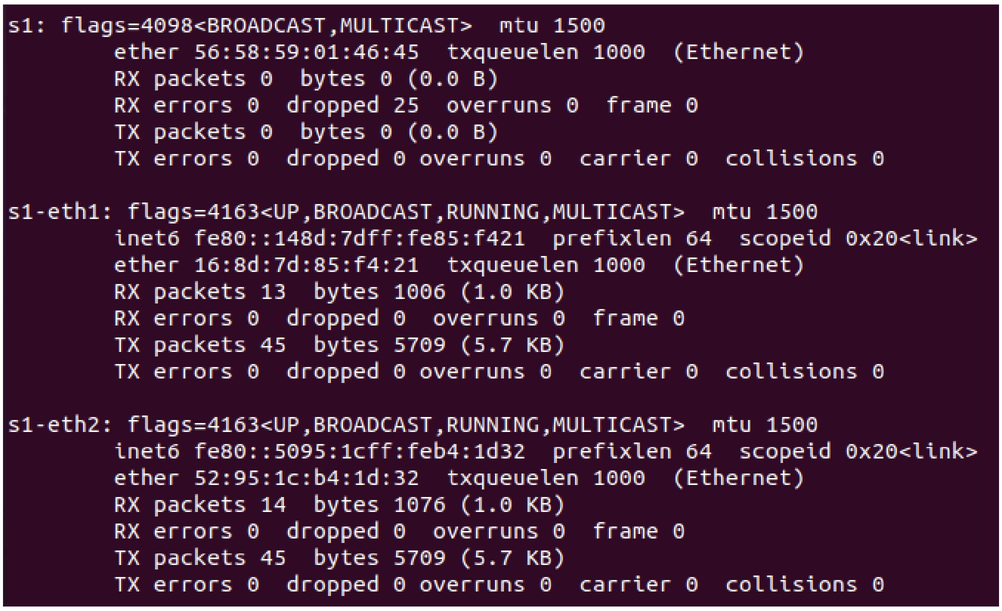


```
mininet> h1 ifconfig -a
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::88f8:e6ff:fe22:725d  prefixlen 64  scopeid 0x20<link>
        ether 8a:f8:e6:22:72:5d  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 3721 (3.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 656 (656.0 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
```

(a) Executing command: h1 ifconfig -a



```
mininet> s1 ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::1de9:f2f8:1970:18b7  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c1:ad:a0  txqueuelen 1000  (Ethernet)
        RX packets 285182  bytes 411134327 (411.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 48370  bytes 6449867 (6.4 MB)
        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 18253  bytes 1774097 (1.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18253  bytes 1774097 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 5a:db:6e:02:bf:c9  txqueuelen 1000  (Ethernet)
        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
```

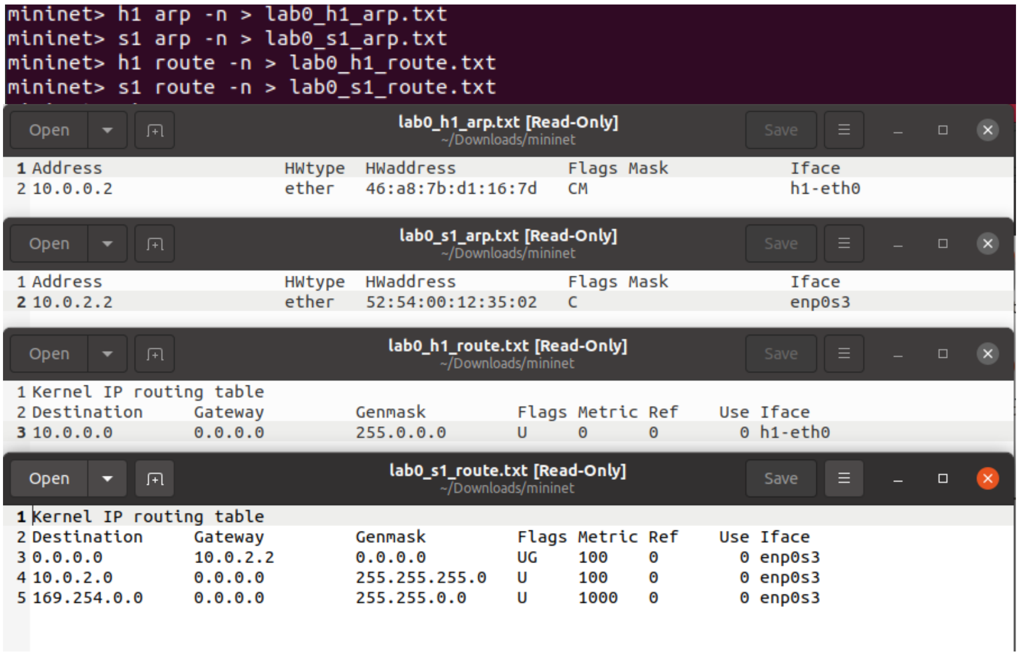


```
s1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 56:58:59:01:46:45  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 25  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::148d:7dff:fe85:f421  prefixlen 64  scopeid 0x20<link>
        ether 16:8d:7d:85:f4:21  txqueuelen 1000  (Ethernet)
        RX packets 13  bytes 1006 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45  bytes 5709 (5.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::5095:1cff:feb4:1d32  prefixlen 64  scopeid 0x20<link>
        ether 52:95:1c:b4:1d:32  txqueuelen 1000  (Ethernet)
        RX packets 14  bytes 1076 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45  bytes 5709 (5.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(b) Executing command: s1 ifconfig -a

Fig. 2. The output of the ifconfig commands

iii. Run arp and route on both s1 and h1. Take a screenshot of the output and discuss how this demonstrates that the hosts have an isolated network state.



```
mininet> h1 arp -n > lab0_h1_arp.txt
mininet> s1 arp -n > lab0_s1_arp.txt
mininet> h1 route -n > lab0_h1_route.txt
mininet> s1 route -n > lab0_s1_route.txt
```

Open
lab0_h1_arp.txt [Read-Only]
~/Downloads/mininet
Save

```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.2                 ether   46:a8:7b:d1:16:7d   CM                    h1-eth0
```

Open
lab0_s1_arp.txt [Read-Only]
~/Downloads/mininet
Save

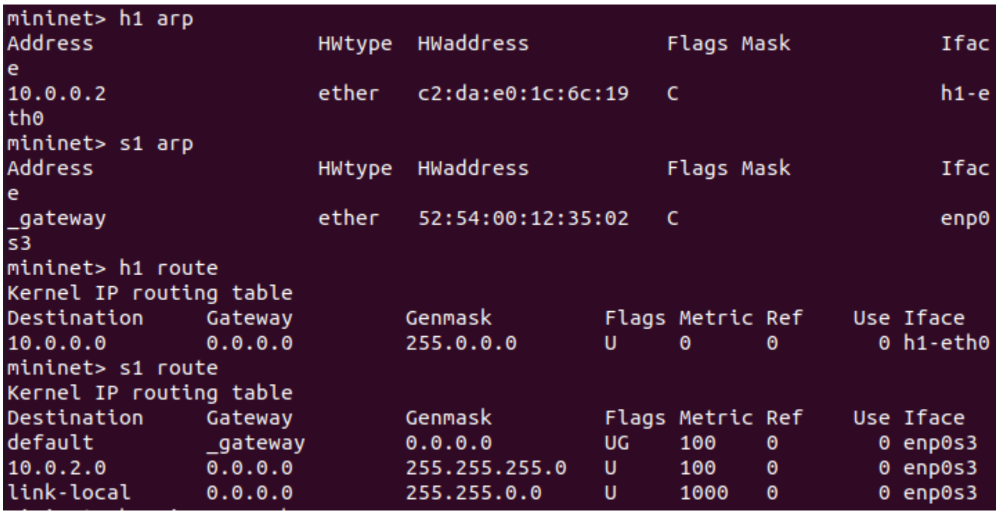
```
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.2                 ether   52:54:00:12:35:02   C                     enp0s3
```

Open
lab0_h1_route.txt [Read-Only]
~/Downloads/mininet
Save

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 h1-eth0
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```

(a) Output shown in the .txt file



```
mininet> h1 arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.0.2                 ether   c2:da:e0:1c:6c:19   C                     h1-e
th0
mininet> s1 arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
_gateway                 ether   52:54:00:12:35:02   C                     enp0
s3
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.0.0.0       U     0      0        0 h1-eth0
mininet> s1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
link-local      0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
```

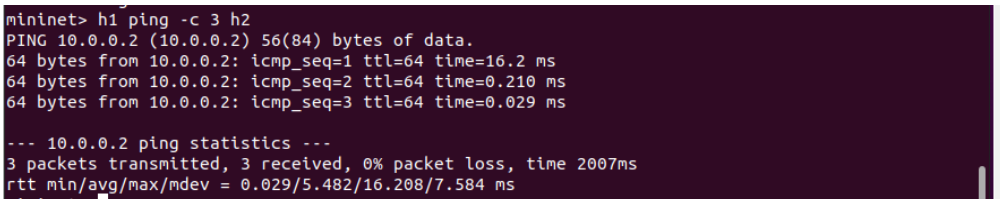
(b) Output shown in the command line

Fig. 3. The output of arp and route on both s1 and h1

Ans: As shown in Fig. 3, h1 only has 1 route indicating to the address "10.0.0.0". Thus, it can only connect to routers with IP addresses within the "10.x.x.x" range, which is its own subnet, as there is no default gateway config.

b. ***Under "Test Connectivity Between Hosts"***

i. Take screenshots of the output of the ping commands.

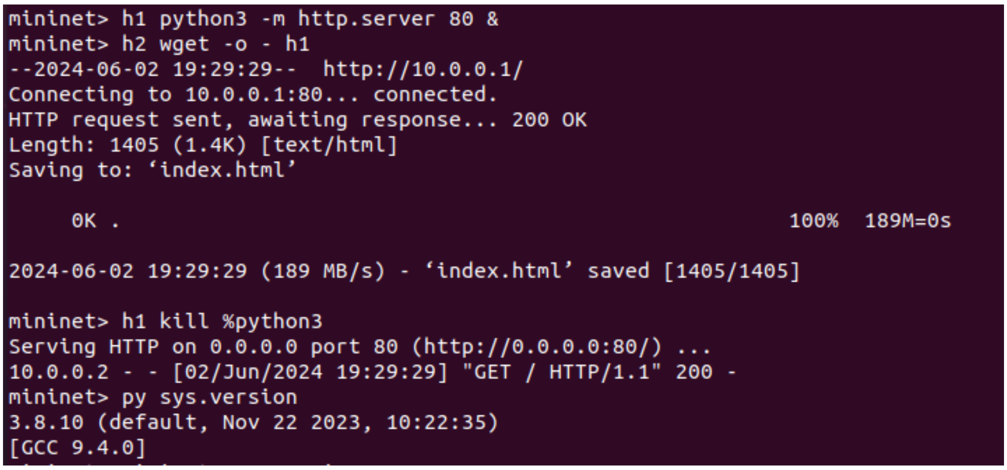


```
mininet> h1 ping -c 3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=16.2 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.210 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.029 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.029/5.482/16.208/7.584 ms
```

Fig. 4. The output of the ping commands

c. ***Under "Run a simple web server and client"***

i. Make sure the Python version you are using is the right one for your version of Mininet. Use the method provided in this section and provide screenshots of the output.



```
mininet> h1 python3 -m http.server 80 &
mininet> h2 wget -o - h1
--2024-06-02 19:29:29--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1405 (1.4K) [text/html]
Saving to: 'index.html'

     0K .                                                     100%  189M=0s

2024-06-02 19:29:29 (189 MB/s) - 'index.html' saved [1405/1405]

mininet> h1 kill %python3
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.0.0.2 - - [02/Jun/2024 19:29:29] "GET / HTTP/1.1" 200 -
mininet> py sys.version
3.8.10 (default, Nov 22 2023, 10:22:35)
[GCC 9.4.0]
```

Fig. 5. The output of python version

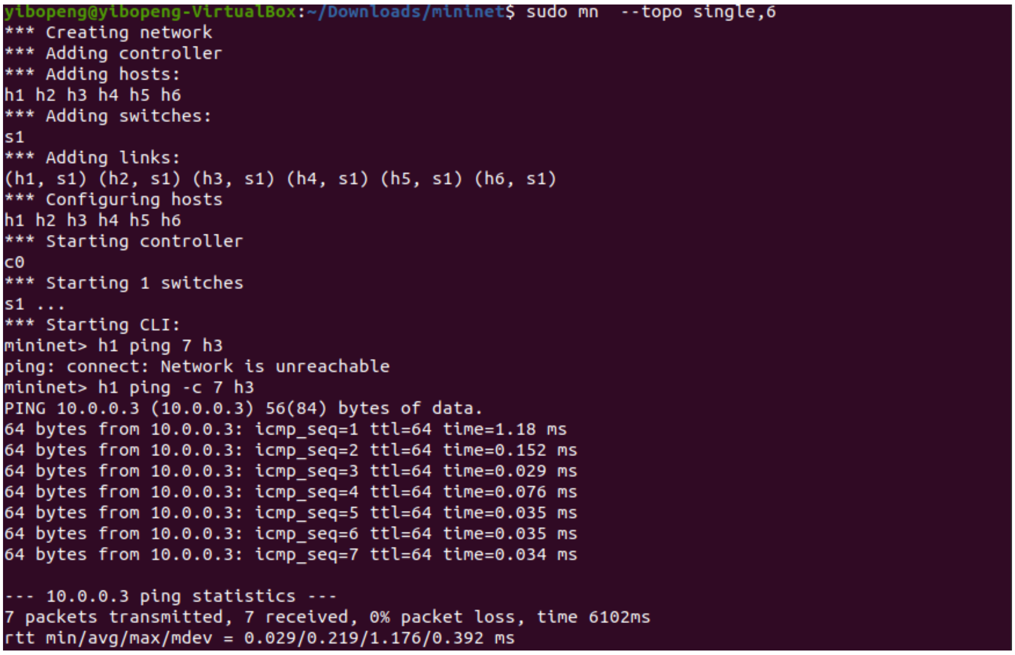
ii. Provide in your lab report the version of Mininet that you are running and the version of Python that you are running.

Ans: python: 3.8.10 & Mininet: 2.3.1b4

3. Part 2: Advanced Startup Options

a. Under "Changing Topology Size and Type"

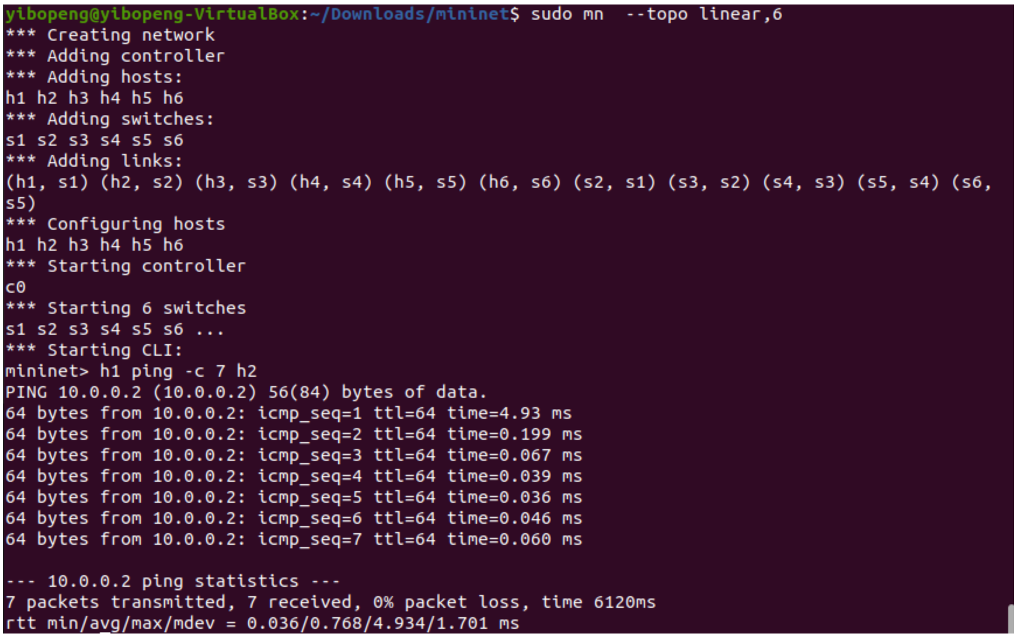
i. Set up a topology with one switch and 6 hosts. Run a ping from h1 to h3. Take screenshots of the commands and outputs.



```
yibopeng@yibopeng-VirtualBox:~/Downloads/mininet$ sudo mn  --topo single,6
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping 7 h3
ping: connect: Network is unreachable
mininet> h1 ping -c 7 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=1.18 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.152 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.029 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.076 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.035 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.034 ms

--- 10.0.0.3 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6102ms
rtt min/avg/max/mdev = 0.029/0.219/1.176/0.392 ms
```

Fig. 6. Commands and outputs of the topology with one switch and six hosts

ii. Set up a linear topology with 6 switches, each with 1 host. Run a ping from h1 to h3. Take screenshots of the commands and outputs.



```
yibopeng@yibopeng-VirtualBox:~/Downloads/mininet$ sudo mn  --topo linear,6
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (s2, s1) (s3, s2) (s4, s3) (s5, s4) (s6,
s5)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> h1 ping -c 7 h2
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=4.93 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.199 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.039 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.036 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.060 ms

--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6120ms
rtt min/avg/max/mdev = 0.036/0.768/4.934/1.701 ms
```

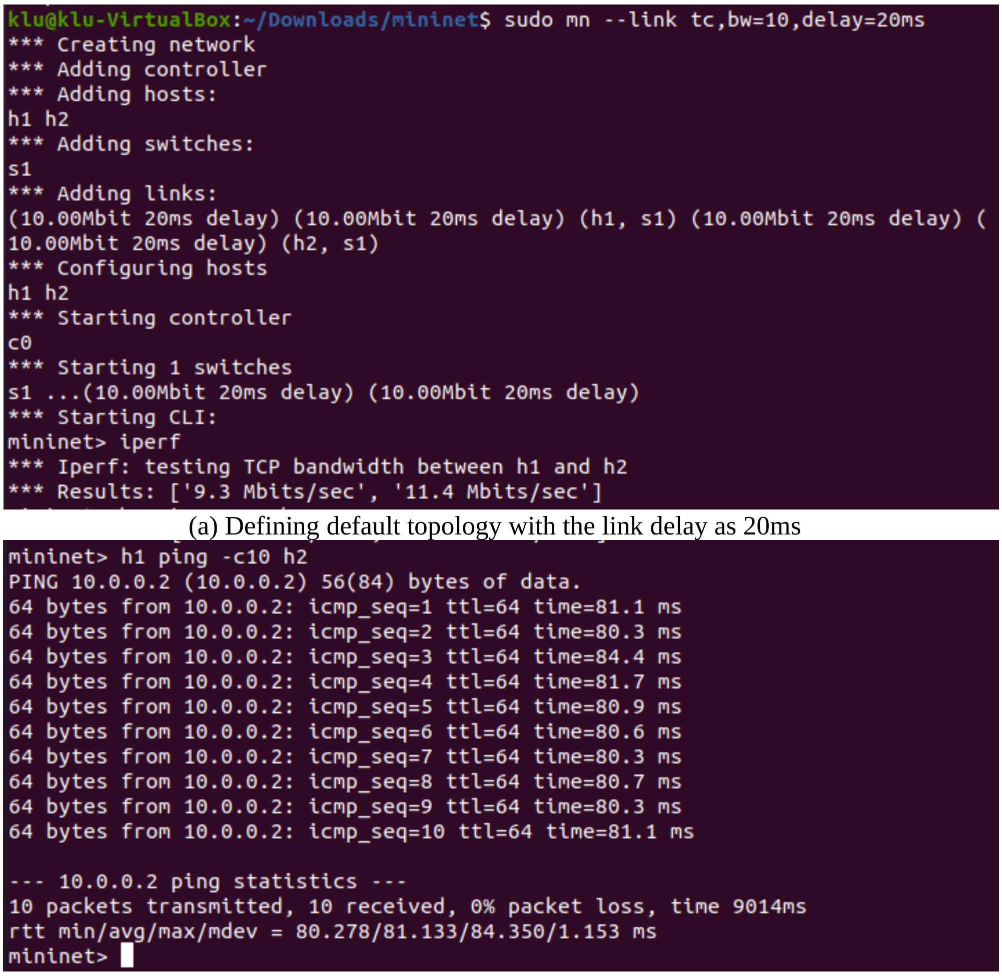
Fig. 7. Commands and outputs of the linear topology with 6 switches, each with 1 host

iii. Which topology has the longer observed latency (delay)? Explain why, referring to the differences in topology.

Ans: As shown in Fig. 6 and Fig. 7, the linear topology with 6 switches has longer observed latency, as sending a packet from h1 to h3 needs to pass three switches in the linear topology with 6 switches while only one switch in the single-switch topology. Passing more switches means more processing time in the switches and more propagation delay in the corresponding links.

b. Under "Link Variations"

i. What is the round trip time (RTT) if you set the delay for each link to 20 ms (milliseconds)?



```
klu@klu-VirtualBox:~/Downloads/mininet$ sudo mn --link tc,bw=10,delay=20ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 20ms delay) (10.00Mbit 20ms delay) (h1, s1) (10.00Mbit 20ms delay) (
10.00Mbit 20ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit 20ms delay) (10.00Mbit 20ms delay)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.3 Mbits/sec', '11.4 Mbits/sec']
```

(a) Defining default topology with the link delay as 20ms

```
mininet> h1 ping -c10 h2
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=81.1 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=80.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=84.4 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=81.7 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=80.9 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=80.6 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=80.3 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=80.7 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=80.3 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=81.1 ms

--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 80.278/81.133/84.350/1.153 ms
mininet>
```

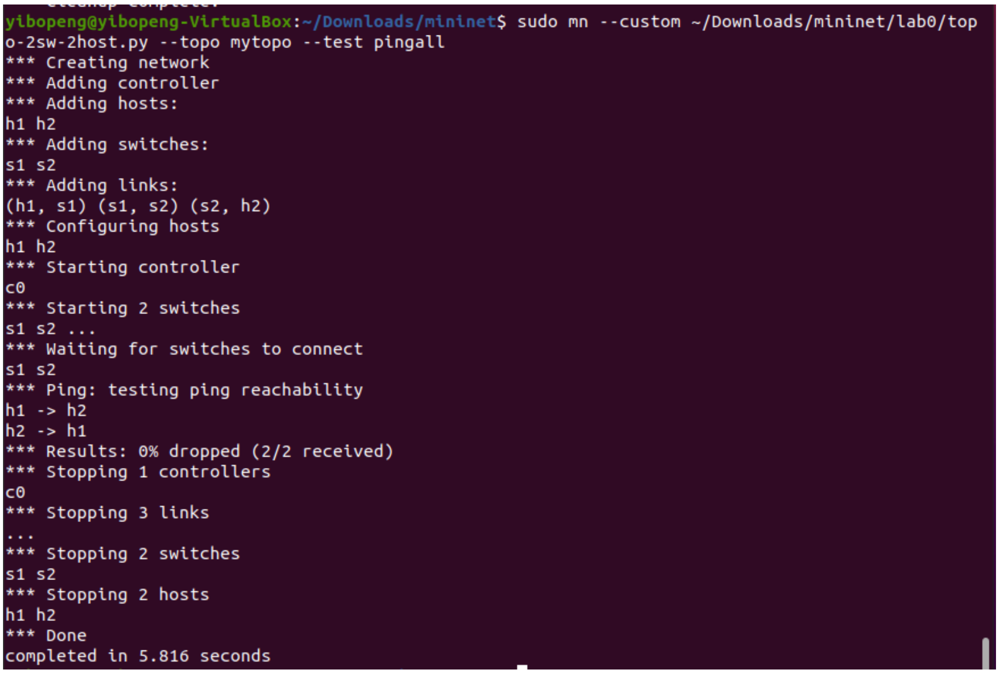
(b) Output of the round-trip time from host 1 to host 2

Fig. 8. Commands and outputs of default topology with the link delay as 20ms

Ans: The average RTT is about 81.133 milliseconds. The round trip includes both ICMP request and reply where the ICMP request traverses two links (i.e., h1 to s1 and s1 to h2) and the ICMP reply traverses two links (i.e., h2 to s1 and s1 to h1). As the delay for each link is 20 milliseconds, the delay for the round trip passing 4 links should be about 80 milliseconds.

c. ***Under "Custom Topologies"***

i. Run the command below and discuss what the topology of the network looks like in terms of switches, hosts, and direct connections/links.



```
yibopeng@yibopeng-VirtualBox:~/Downloads/mininet$ sudo mn --custom ~/Downloads/mininet/lab0/top
o-2sw-2host.py --topo mytopo --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (s1, s2) (s2, h2)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Waiting for switches to connect
s1 s2
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.816 seconds
```

Fig. 9. Commands and outputs of defining the custom topology with 2 switches and 2 hosts

Ans: MyTopo has two end hosts connected to two middle switches and each switch connects with its nearest host and the other switch, as shown in Fig. 10.

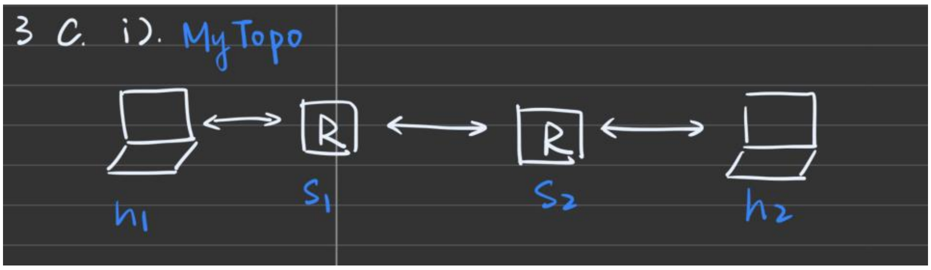


Fig. 10. Sketch of the custom typology with 2 hosts and 2 switches