

# Project 2 OpenFlow Based Stateless firewall

Student Name: Mohammed Ragab

Email: maragab@asu.edu

Submission Date: 8<sup>th</sup> November 2023

Class Name and Term: CSE548 Fall 2023

**Video link:** <https://youtu.be/C9x4DCE4aEE>

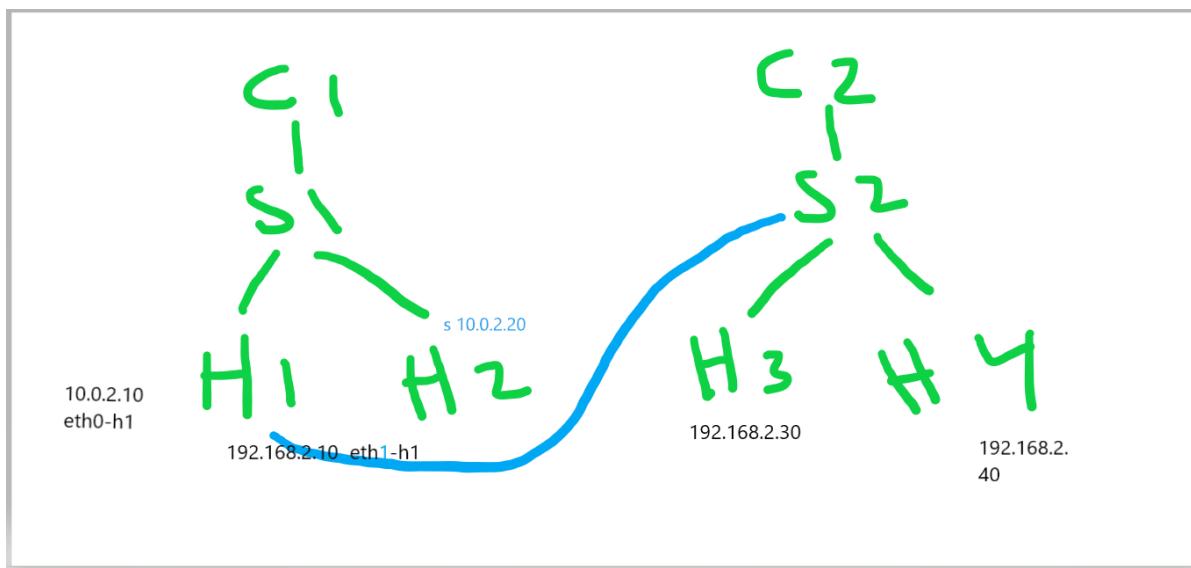
## I. PROJECT OVERVIEW

I have created a software-defined network firewall using OpenFlow protocol using POX controller and I created a virtual network using Mininet tool and I created a python script the uses the Mininet python APIs. The network has 4 hosts, 2 switches and 2 controllers. The links between these nodes are specified as in the assigned instructions and as shown in my video.

in the terminal window on Mininet (CLI), I added the linked between switches and the controllers as shown in my video

## II. NETWORK SETUP

- Here is my layout of the network with the assigned IP addresses for each host,



## III. SOFTWARE

- Mininet
- POX
- Containernet
- Ping
- Tcpdump
- Python3
- VScode as text editor
- Linux ubuntu
- Xterm terminal

## IV. PROJECT DESCRIPTION

# OVS lab

## add internet gateway interface to bridge 0

```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sat 03:52
root@ubuntu: ~
File Edit View Search Terminal Tabs Help root@ubuntu: ~
root@ubuntu:~# ovs-vsctl del-br br0
root@ubuntu:~# ovs-vsctl show
b0159eb6-dfd1-4693-b7a2-f90817a738ac
  ovs_version: "2.9.8"
root@ubuntu:~# ovs-vsctl add-br br0
root@ubuntu:~# ovs-vsctl show
b0159eb6-dfd1-4693-b7a2-f90817a738ac
  Bridge "br0"
    Port "br0"
      Interface "br0"
        type: internal
        ovs_version: "2.9.8"
root@ubuntu:~# ifconfig br0 up
root@ubuntu:~# ovs-vsctl add-port br0 enp0s3
root@ubuntu:~# ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fdd8:2918:9b30:6a00:508:3720:d7d4:2d6a  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::409b:efff:fef6:5347  prefixlen 64  scopeid 0x20<link>
        inet6 fdd8:2918:9b30:6a00:a00:27ff:fe71:9eca  prefixlen 64  scopeid 0x0<global>
        ether 08:00:27:71:9e:ca  txqueuelen 1000  (Ethernet)
          RX packets 15  bytes 1060 (1.0 KB)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 46  bytes 6596 (6.5 KB)
          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:66:62:f0:ef  txqueuelen 0  (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

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.3  netmask 255.255.255.0  broadcast 192.168.1.255
```

## Check bridge setup

Please edit the highlighted portion.

3

The screenshot shows a terminal window titled "Terminal" running on an Ubuntu system. The terminal displays the output of several commands:

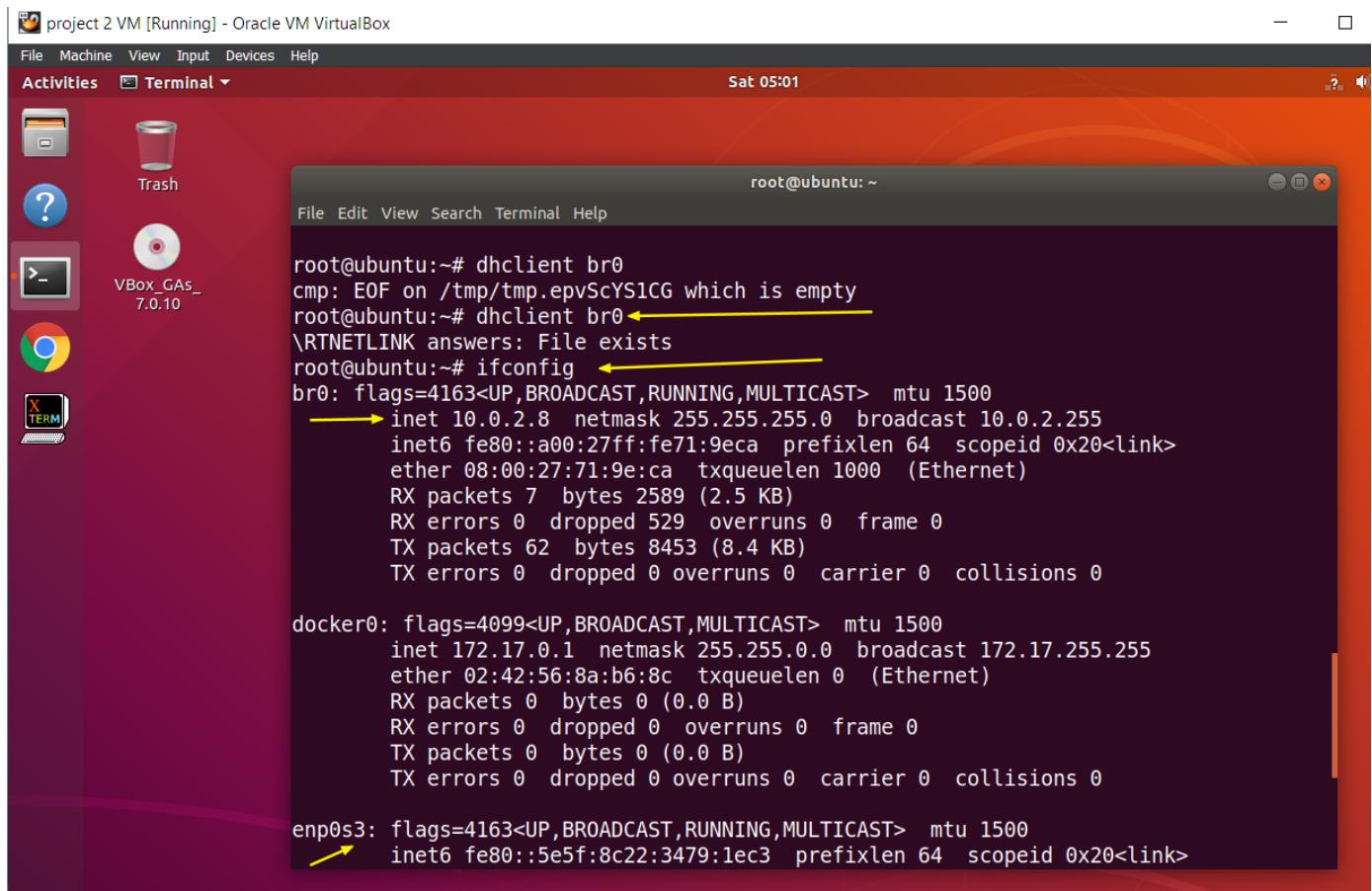
- `ifconfig` output for interfaces `inet6 fe80::7c5c:63bf` and lo. The inet6 fe80::7c5c:63bf` interface has a MTU of 65536 and is connected via Ethernet. The lo interface has a MTU of 65536 and is a Local Loopback.`
- `ifconfig br0` output for interface `br0`. It shows a MTU of 1500 and is connected via Ethernet. RX and TX statistics are shown.
- `ovs-vsctl del-br br1` command to delete bridge `br1`.
- `ovs-vsctl show` command to show Open vSwitch configuration. It lists bridge `br0`, port `br0`, and interface `br0` (type: internal).
- `ovs_version: "2.9.8"` command to show the version of the Open vSwitch library.

Red arrows highlight the command `ifconfig br0`, the command `ovs-vsctl show`, and the interface entry for `br0` in the `ovs-vsctl show` output.

assign IP to br0 using dhclient command

Please edit the highlighted portion.

4



A screenshot of an Ubuntu desktop environment within Oracle VM VirtualBox. The terminal window shows the root user's command-line interface. The user has run several commands related to network interfaces:

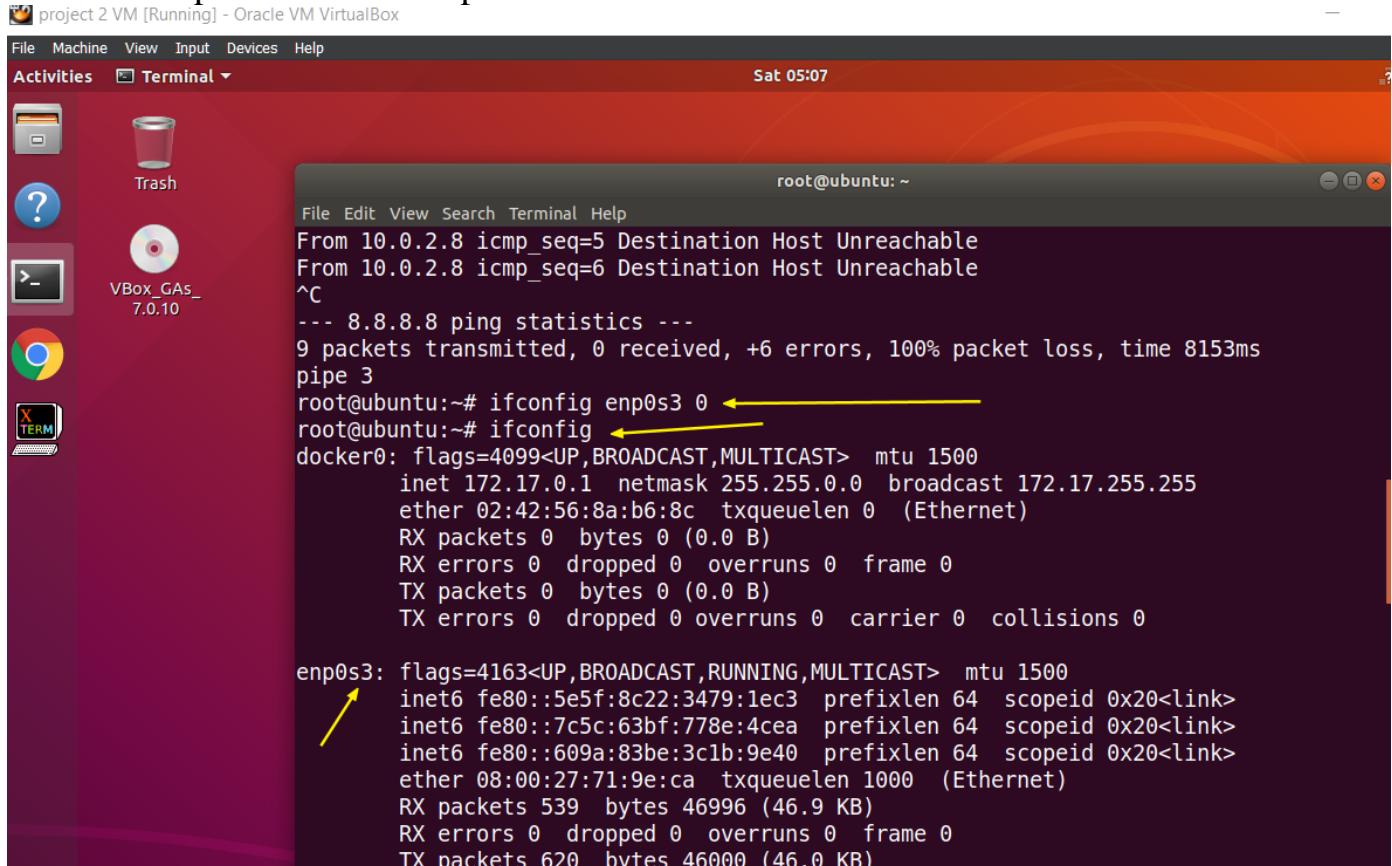
```
root@ubuntu:~# dhclient br0
cmp: EOF on /tmp/tmp.epvScYS1CG which is empty
root@ubuntu:~# dhclient br0
\RTNETLINK answers: File exists
root@ubuntu:~# ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.8 netmask 255.255.255.0 broadcast 10.0.2.255
                inet6 fe80::a00:27ff:fe71:9eca prefixlen 64 scopeid 0x20<link>
                  ether 08:00:27:71:9e:ca txqueuelen 1000 (Ethernet)
                    RX packets 7 bytes 2589 (2.5 KB)
                    RX errors 0 dropped 529 overruns 0 frame 0
                    TX packets 62 bytes 8453 (8.4 KB)
                    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
          ether 02:42:56:8a:b6:8c txqueuelen 0 (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

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet6 fe80::5e5f:8c22:3479:1ec3 prefixlen 64 scopeid 0x20<link>
```

The output shows three network interfaces: br0, docker0, and enp0s3. The br0 interface is currently configured with an IP address of 10.0.2.8. The docker0 interface is a bridge interface. The enp0s3 interface is a physical Ethernet interface.

remove the ip address from enp0s3 device



A screenshot of an Ubuntu desktop environment within Oracle VM VirtualBox. The terminal window shows the root user's command-line interface. The user has run several commands related to network interfaces:

```
From 10.0.2.8 icmp_seq=5 Destination Host Unreachable
From 10.0.2.8 icmp_seq=6 Destination Host Unreachable
^C
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 0 received, +6 errors, 100% packet loss, time 8153ms
pipe 3
root@ubuntu:~# ifconfig enp0s3 0
root@ubuntu:~# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
          ether 02:42:56:8a:b6:8c txqueuelen 0 (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

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet6 fe80::5e5f:8c22:3479:1ec3 prefixlen 64 scopeid 0x20<link>
        inet6 fe80::7c5c:63bf:778e:4cea prefixlen 64 scopeid 0x20<link>
        inet6 fe80::609a:83be:3c1b:9e40 prefixlen 64 scopeid 0x20<link>
          ether 08:00:27:71:9e:ca txqueuelen 1000 (Ethernet)
            RX packets 539 bytes 46996 (46.9 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 620 bytes 46000 (46.0 KB)
```

The output shows the enp0s3 interface now has no IP address assigned, indicated by the '0' parameter in the ifconfig command. The other interfaces (br0, docker0) remain configured.

Please edit the highlighted portion.

5

set the default gateway as br0 and ping google DNS successfully

The screenshot shows a terminal window titled "root@ubuntu:~". The terminal output is as follows:

```
root@ubuntu:~# sudo ip route del default dev <interface>
-bash: syntax error near unexpected token `newline'
root@ubuntu:~#
root@ubuntu:~# ip route del default dev enp0s3 ←
root@ubuntu:~# ip route show ←
default via 10.0.2.1 dev br0
10.0.2.0/24 dev br0 proto kernel scope link src 10.0.2.8
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.8 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
root@ubuntu:~# ping 8.8.8.8 ←
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=108 time=62.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=61.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=108 time=59.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=108 time=58.9 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=108 time=59.8 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 58.902/60.633/62.972/1.537 ms
root@ubuntu:~#
```

Yellow arrows highlight the command "ip route del default dev enp0s3", the output "default via 10.0.2.1 dev br0", the ping command "ping 8.8.8.8", and the output "0% packet loss".

show that the br0 is the default gateway

The screenshot shows a terminal window titled "ubuntu@ubuntu:~". The terminal output is as follows:

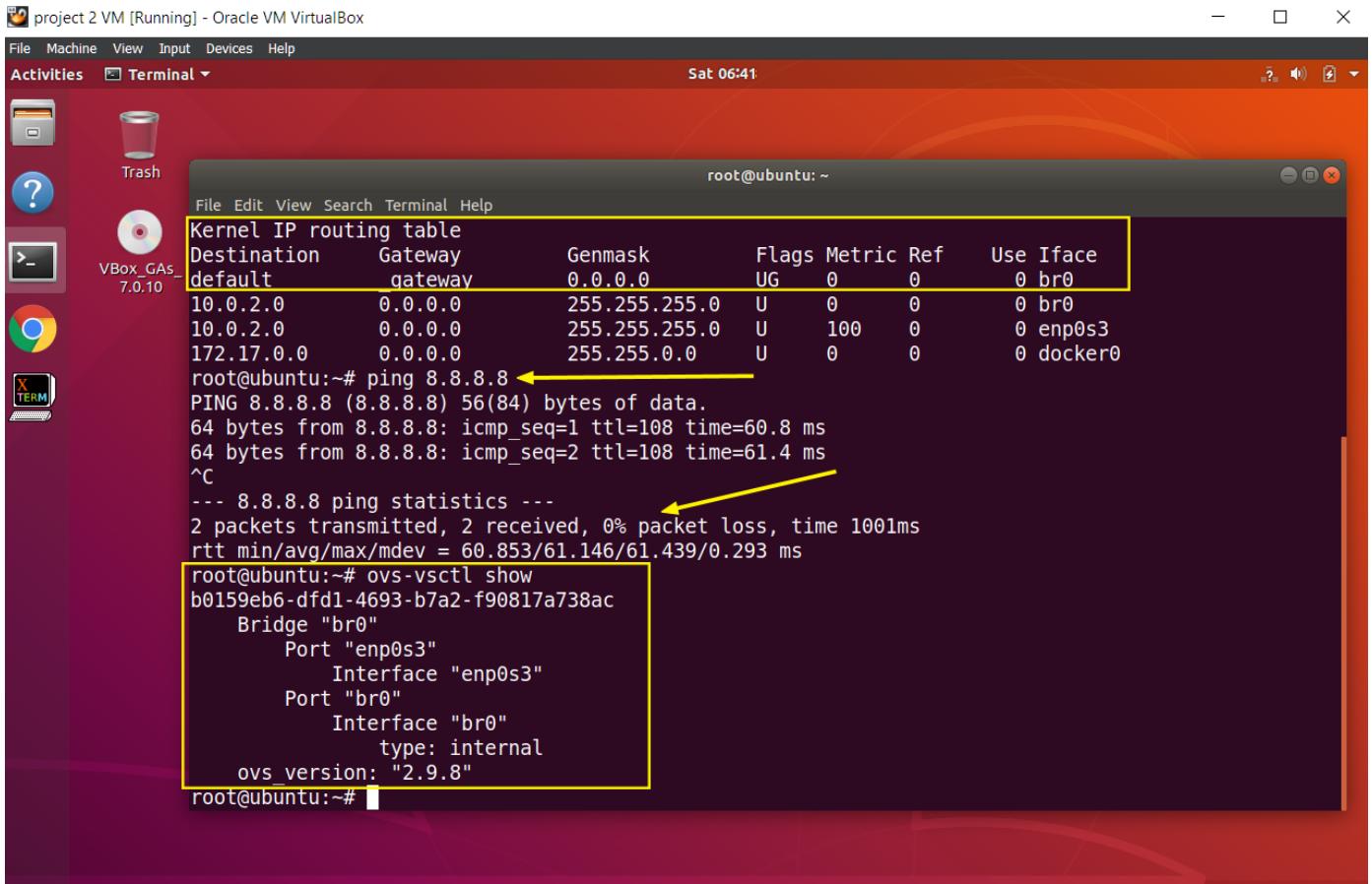
```
ubuntu@ubuntu:~$ route -n ←
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1       0.0.0.0       UG    0      0        0 br0
0.0.0.0         10.0.2.1       0.0.0.0       UG    20100   0        0 enp0s3
10.0.2.0        0.0.0.0        255.255.255.0  U     0      0        0 br0
10.0.2.0        0.0.0.0        255.255.255.0  U     100    0        0 enp0s3
172.17.0.0      0.0.0.0        255.255.0.0    U     0      0        0 docker0
ubuntu@ubuntu:~$ route del -net 0.0.0.0 gw 10.0.2.1 netmask 0.0.0.0 dev enp0s3
SIOCDELRT: Operation not permitted
ubuntu@ubuntu:~$ sudo route del -net 0.0.0.0 gw 10.0.2.1 netmask 0.0.0.0 dev enp0s3 ←
[sudo] password for ubuntu:
ubuntu@ubuntu:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.1       0.0.0.0       UG    0      0        0 br0
10.0.2.0        0.0.0.0        255.255.255.0  U     0      0        0 br0
10.0.2.0        0.0.0.0        255.255.255.0  U     100    0        0 enp0s3
172.17.0.0      0.0.0.0        255.255.0.0    U     0      0        0 docker0
ubuntu@ubuntu:~$
```

A yellow arrow highlights the command "sudo route del -net 0.0.0.0 gw 10.0.2.1 netmask 0.0.0.0 dev enp0s3".

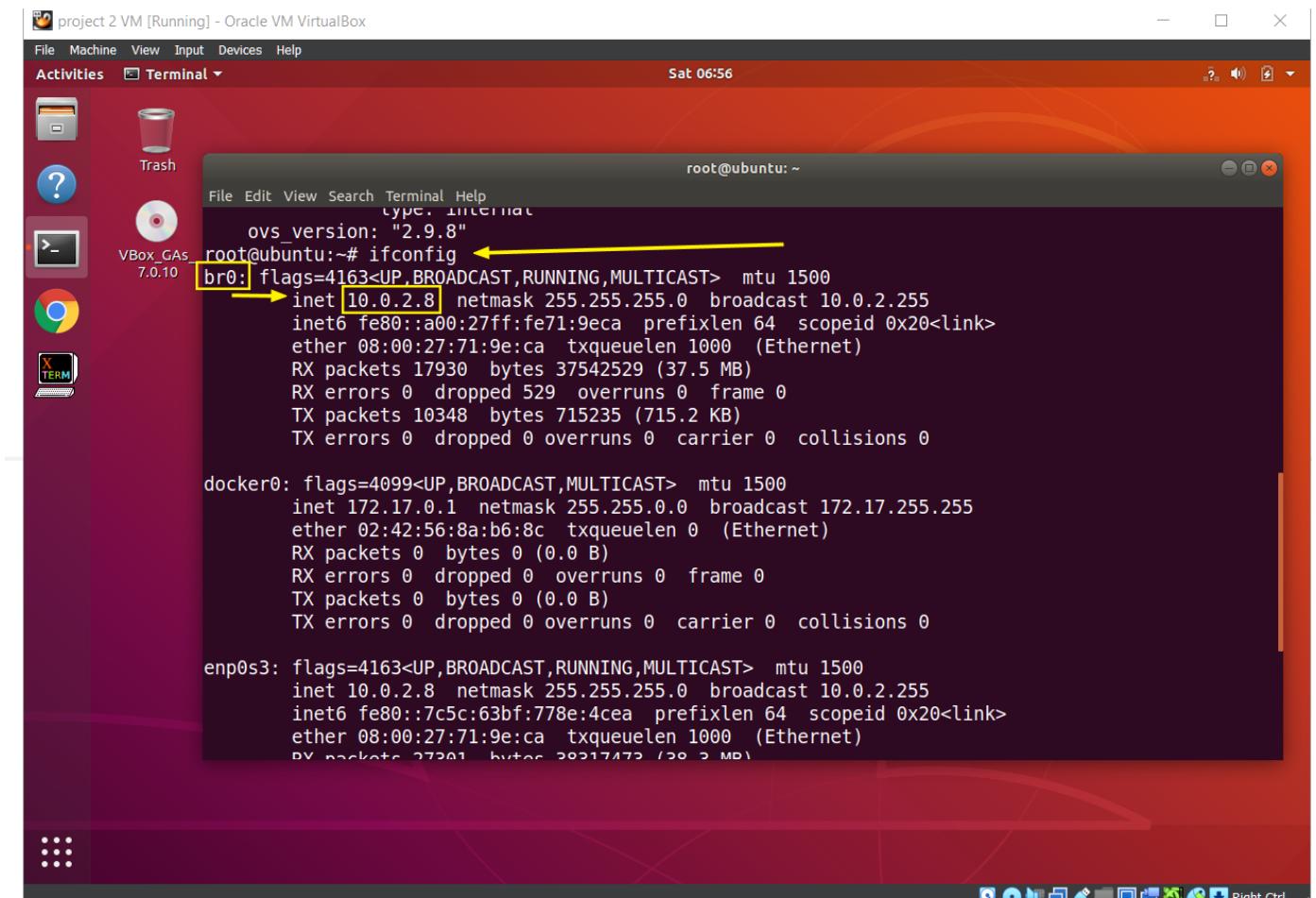
show br0 and its interface and can ping google dns successfully and default route is the br0

Please edit the highlighted portion.

6



show the ip address of br0



# POX lab

## run POX

The screenshot shows a desktop environment with a window titled "project 2 VM [Running] - Oracle VM VirtualBox". Inside, there are three terminal windows and one network monitor window.

- Terminal 1:** "Node: h1" (root shell)
- Terminal 2:** "Node: h2" (root shell)
- Terminal 3:** "Node: h3" (root shell)
- Network Monitor:** Monitors traffic on interface h4. It shows several ICMP echo requests and replies between nodes h1, h2, and h3.

```

root@ubuntu:~/pox
Args are:
--short    Only summarize docs
--no-args  Don't show parameter info

Parameters for default launcher:
Name          Default
-----
no-args       False
short         False

Note: This can be invoked with parameters not listed here.
root@ubuntu:~/pox# ./pox.py -verbose forwarding.hub ←
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
INFO:forwarding.hub:Proactive hub running.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.6.9/Apr 18 2020 01:56:04)
DEBUG:core:Platform is Linux-5.3.0-53-generic-x86_64-with-Ubuntu-18.04-bionic
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-00-01
DEBUG:openflow.of_01: connection aborted
^[]

h 64
11:33:55.827811 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5071, seq 9, length 64
11:33:56.828514 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5071, seq 10, length 64
11:33:56.828553 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5071, seq 10, length 64

```

## run mininet with the specified topology

```

ubuntu@ubuntu:~$ sudo mn --topo single,3 --mac --arp --switch ovsk --controller=remote
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> xterm h1 h2 h3
containernet>

```

Node: h2

```

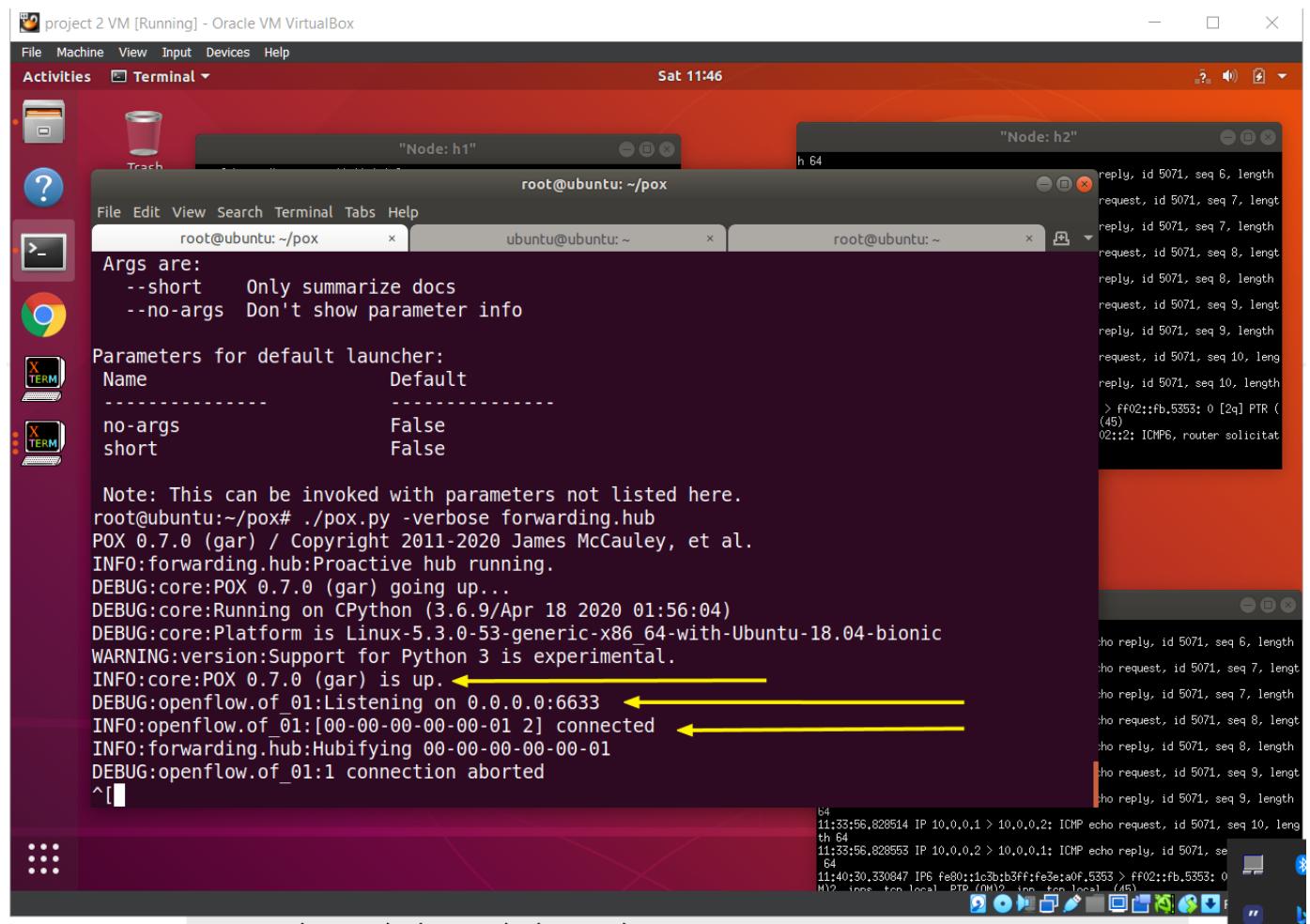
h 64
64
11:33:56.828514 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5071, seq 10, length 64
11:33:56.828553 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 5071, seq 11, length 64
11:40:30.330847 IP6 fe80::1c3b:b3ff:fe3e:a0f.5353 > ff02::fb.5353: 0 [45]
ICMP6 echo request, id 5071, seq 10, length 64
11:40:30.330847 IP6 fe80::1c3b:b3ff:fe3e:a0f.5353 > ff02::fb.5353: 0 [45]
ICMP6 echo reply, id 5071, seq 11, length 64

```

pox controller is up and running on the mininet network

Please edit the highlighted portion.

9



start the forwarding.l2 pairs component correctly

The screenshot shows a Linux desktop environment with several windows open. In the top left, there's a window titled "project 2 VM [Running] - Oracle VM VirtualBox". Below it is a dock with icons for a file browser, terminal, and other applications. Two terminal windows are visible: one titled "Node: h1" and another titled "Node: h2". The "Node: h1" window shows the command-line interface for a POX controller, with several log messages. One message highlights the command `./pox.py -verbose forwarding.l2_pairs`. Another message highlights the status `INFO:core:POX 0.7.0 (gar) is up.`. A third message highlights the listening port `DEBUG:openflow.of_01:Listening on 0.0.0.0:6633`. A fourth message highlights the connection `INFO:openflow.of_01:[00-00-00-00-00-01 1] connected`. A yellow arrow points from the highlighted command in the "Node: h1" terminal to the corresponding log entry. Another yellow arrow points from the "INFO:core:POX 0.7.0 (gar) is up." message to the "Node: h2" terminal window, which displays a series of ICMP echo requests and replies. The "Node: h2" window has a title bar showing "h 64" and a status bar showing "Sat 11:52".

```

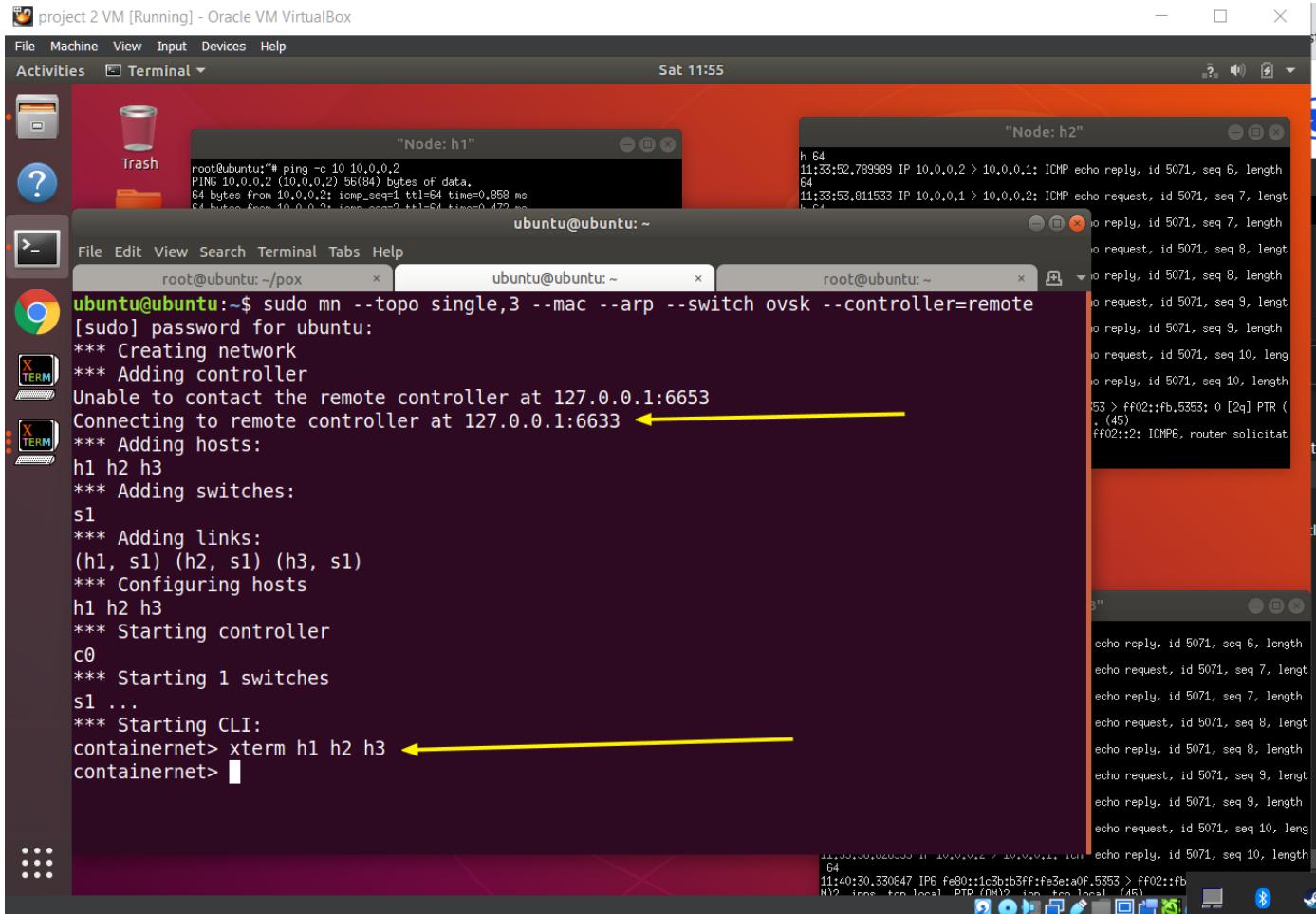
root@ubuntu:~/pox
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-00-01
DEBUG:openflow.of_01:1 connection aborted
^[[CINFO:core:Going down...
INFO:openflow.of_01:[00-00-00-00-00-01 2] disconnected
INFO:core:Down.
ERROR:openflow.of_01:Exception reading connection None
Traceback (most recent call last):
  File "/root/pox/pox/openflow/of_01.py", line 1084, in run
    rlist, wlist, elist = yield Select(sockets, [], sockets, 5)
GeneratorExit
DEBUG:openflow.of_01:No longer listening for connections
root@ubuntu:~/pox# ./pox.py -verbose forwarding.l2_pairs
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
INFO:forwarding.l2_pairs:Pair-Learning switch running.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.6.9/Apr 18 2020 01:56:04)
DEBUG:core:Platform is Linux-5.3.0-53-generic-x86_64-with-Ubuntu-18.04-bionic
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected

```

mininet topology after running POX controller with forwarding.l2 pairs component correctly

Please edit the highlighted portion.

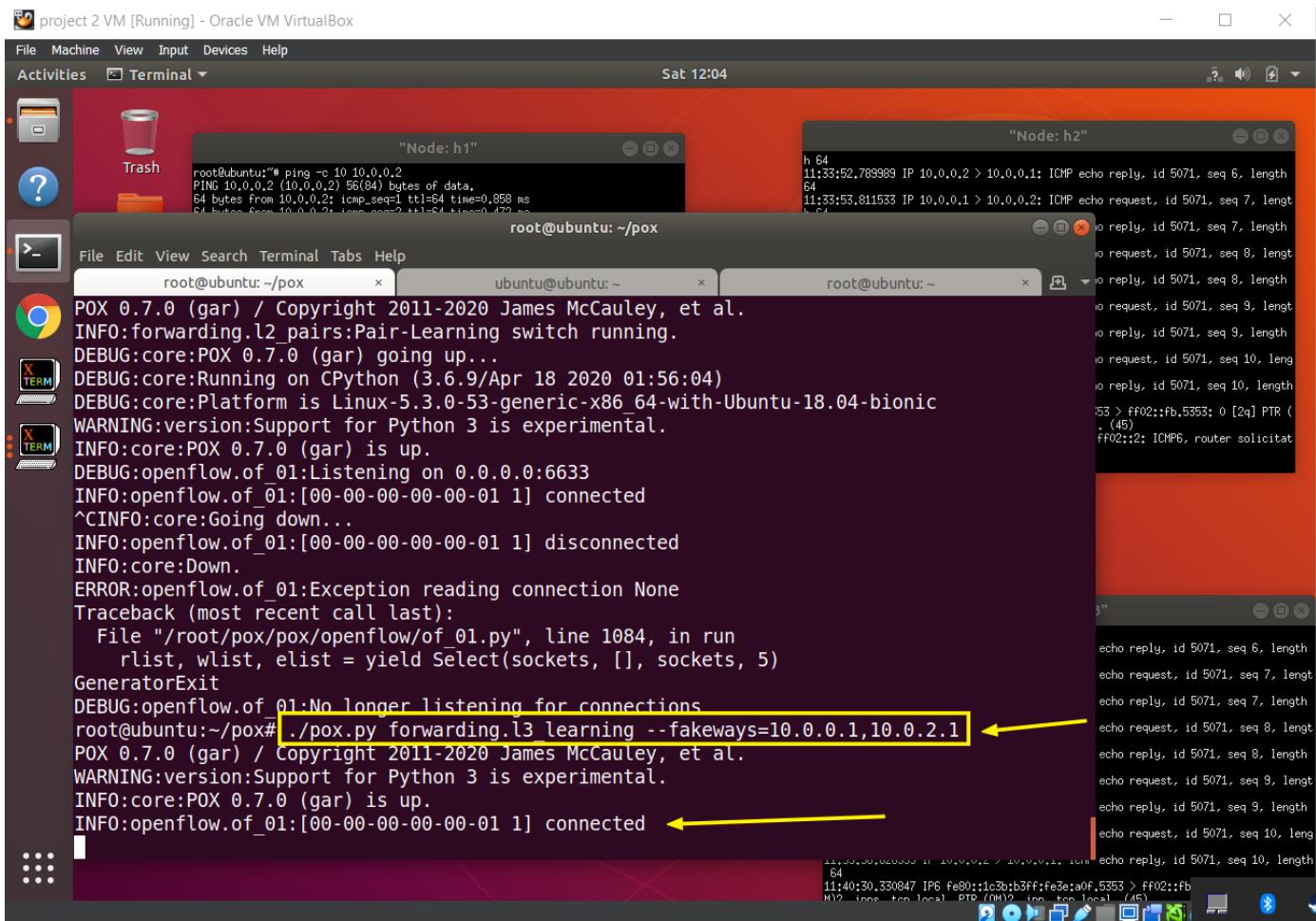
11



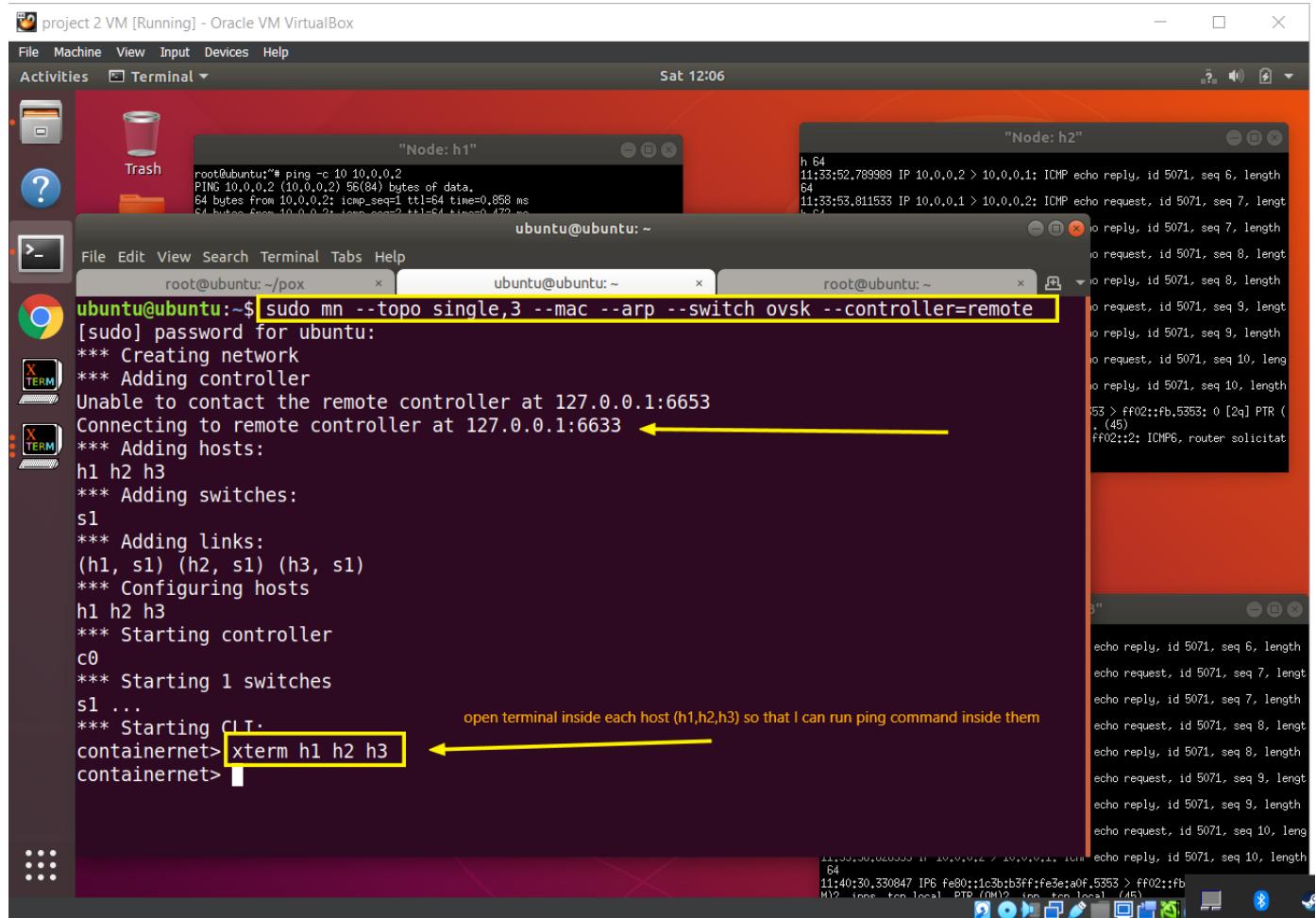
run POX with fake gateways

Please edit the highlighted portion.

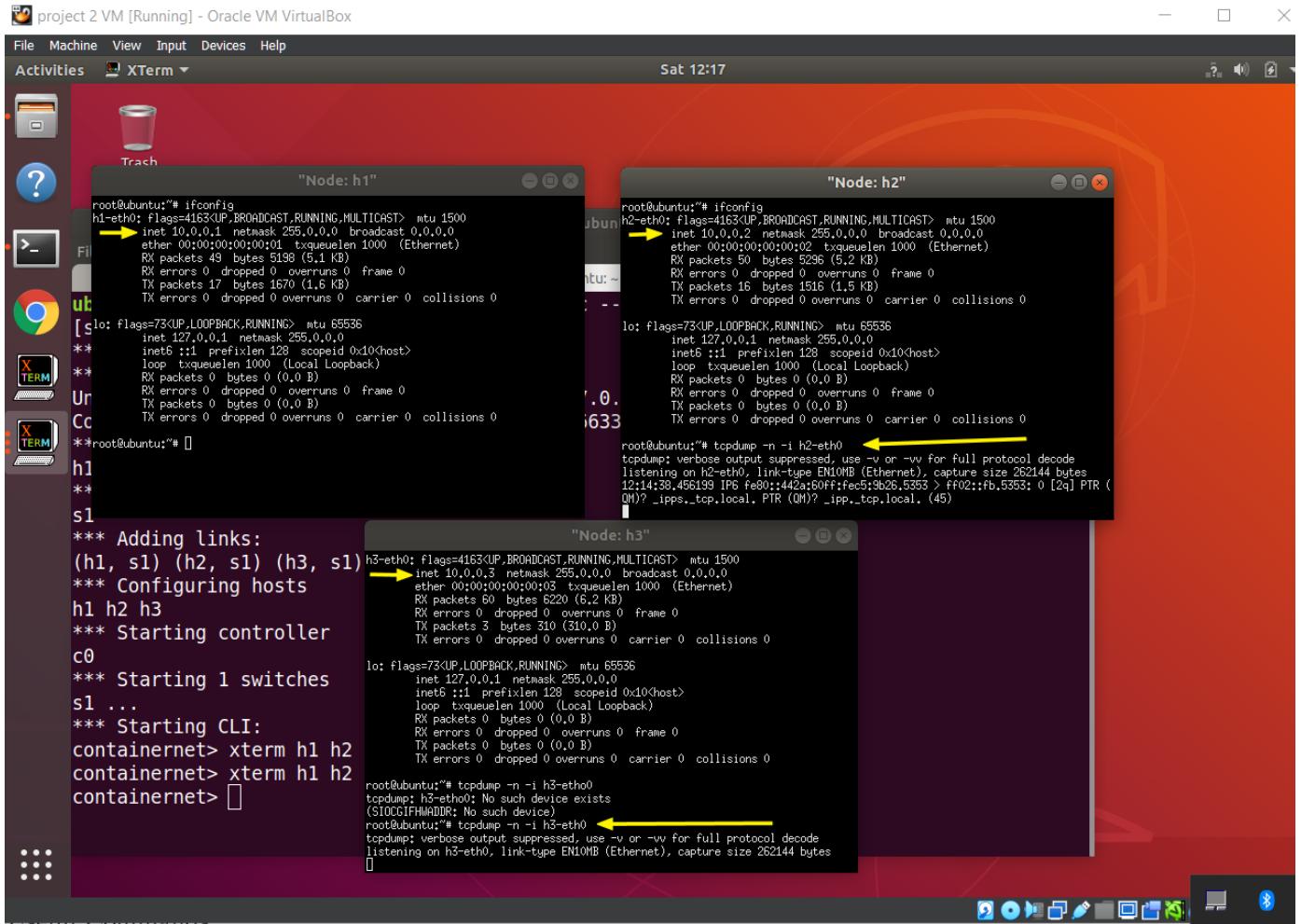
12



mininet topology after run POX with fake gateways



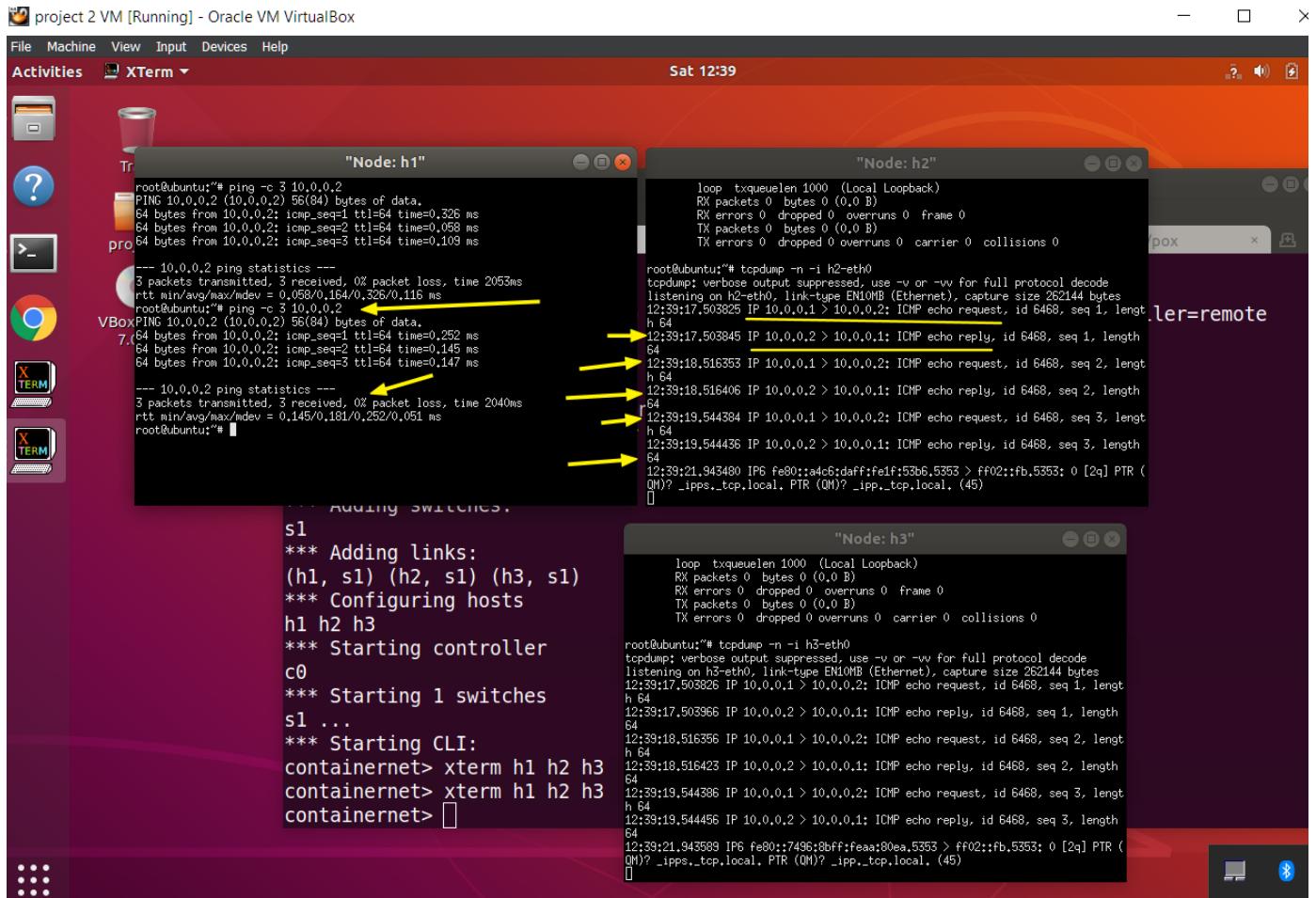
xterm of 3 hosts and their IP addresses and setting TCP dump on h2 and h3 to capture the ping request from h1 to h2



host 1 ping host 2 and TCPDump logs the ping request and reply showing the IP of host 1 and host 2

Please edit the highlighted portion.

15



## Mininet Lab

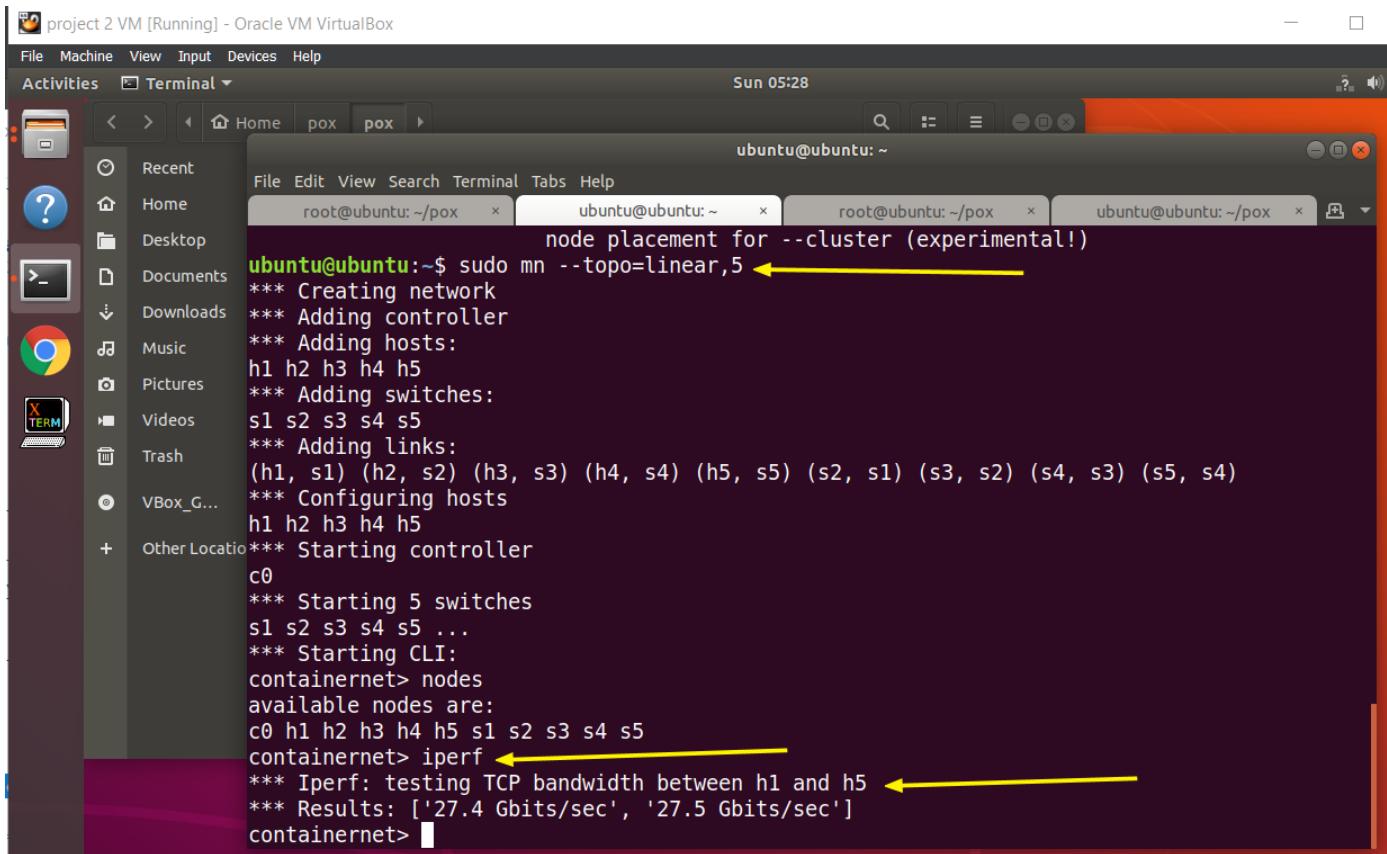
create linear topology with 5 switches and 1 host for each switch

```
ubuntu@ubuntu:~$ sudo mn --topo=linear,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
containernet>
```

## nodes of linear topology with 5 switches and 1 host for each switch

```
--cluster=server1,server2...
run on multiple servers (experimental!)
--placement=block|random
node placement for --cluster (experimental!)
ubuntu@ubuntu:~$ sudo mn --topo=linear,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
containernet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 s1 s2 s3 s4 s5
containernet>
```

## measure bandwidth of linear topology with 5 switches and 1 host for each switch



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has four tabs:

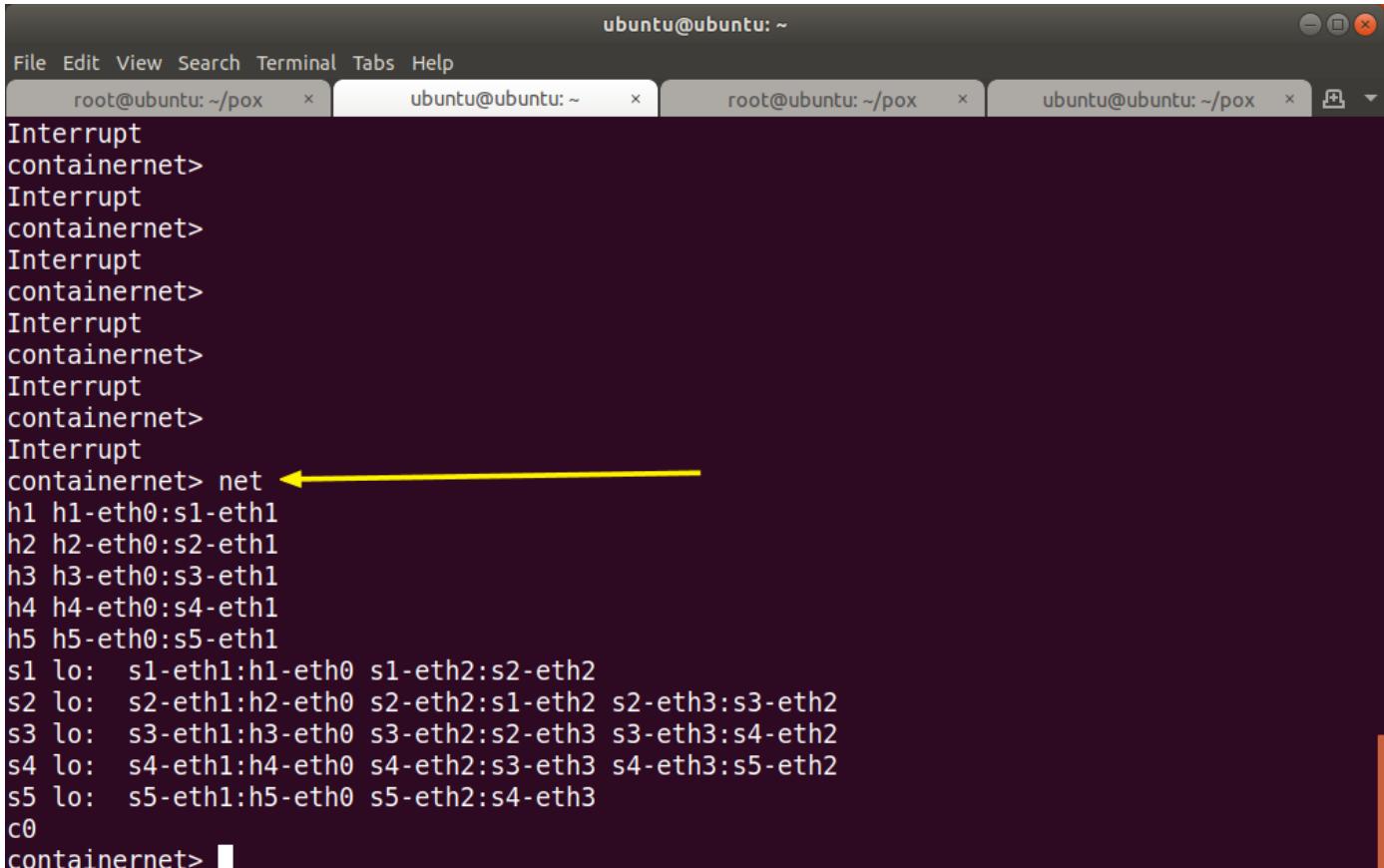
- root@ubuntu:~/pox
- ubuntu@ubuntu:~
- root@ubuntu:~/pox
- ubuntu@ubuntu:~/pox

The active tab (second from left) contains the following command and its output:

```
ubuntu@ubuntu:~$ sudo mn --topo=linear,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
containernet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 s1 s2 s3 s4 s5
containernet> iperf
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['27.4 Gbits/sec', '27.5 Gbits/sec']
containernet>
```

Yellow arrows highlight the command `sudo mn --topo=linear,5`, the `iperf` command, and the bandwidth results.

## net links of linear topology with 5 switches and 1 host for each switch



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has four tabs:

- root@ubuntu:~/pox
- ubuntu@ubuntu:~
- root@ubuntu:~/pox
- ubuntu@ubuntu:~/pox

The active tab (second from left) contains the following command and its output:

```
containernet>
Interrupt
containernet>
Interrupt
containernet>
Interrupt
containernet>
Interrupt
containernet>
Interrupt
containernet>
Interrupt
containernet>
containernet> net
```

A yellow arrow points to the command `net`.

Below the command, the output lists network interfaces for hosts h1 through h5 and switches s1 through s5:

```
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
h5 h5-eth0:s5-eth1
s1 lo: s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo: s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo: s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo: s4-eth1:h4-eth0 s4-eth2:s3-eth3 s4-eth3:s5-eth2
s5 lo: s5-eth1:h5-eth0 s5-eth2:s4-eth3
c0
```

A yellow arrow points to the first interface entry for host h1.

**measure bandwidth of single topology with 1 switch and 4 host**

The screenshot shows a terminal window titled "project 2 VM [Running] - Oracle VM VirtualBox". The terminal has four tabs open:

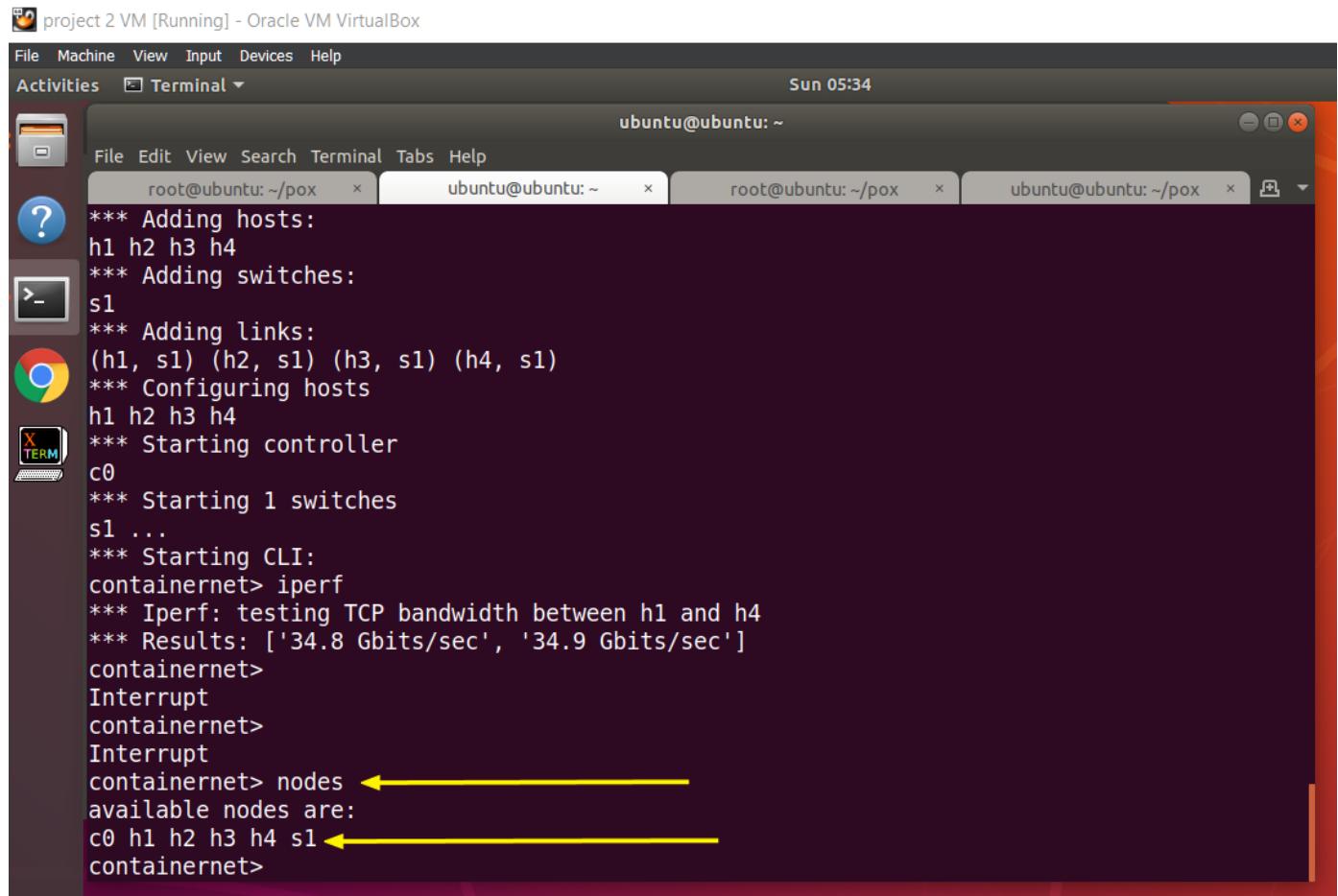
- root@ubuntu:~/pox
- ubuntu@ubuntu:~
- root@ubuntu:~/pox
- ubuntu@ubuntu:~/pox

The main terminal window displays the following output:

```
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
completed in 629.068 seconds
ubuntu@ubuntu:~$ sudo mn --topo single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['34.8 Gbits/sec', '34.9 Gbits/sec']
containernet>
```

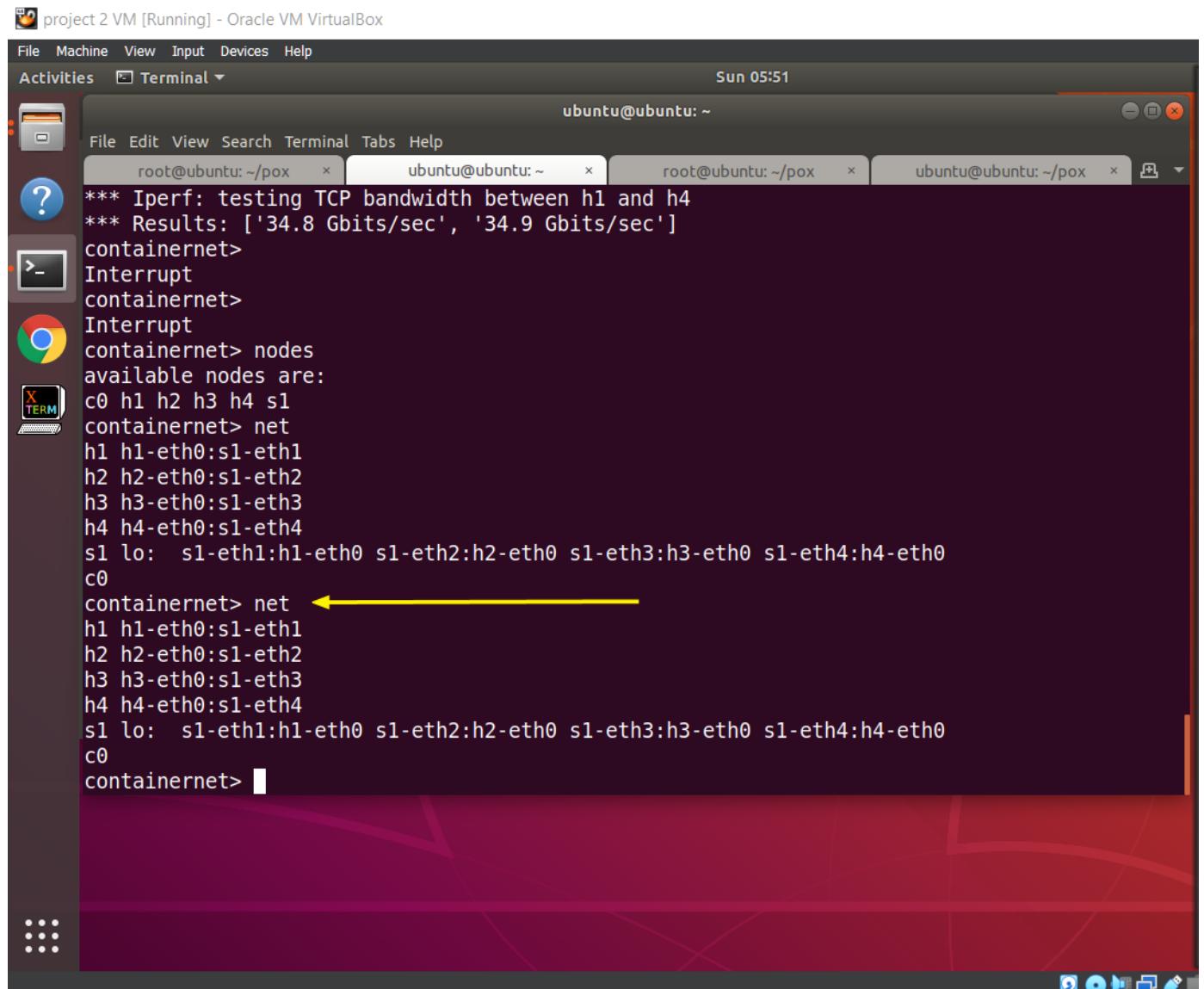
Three yellow arrows point to the command "sudo mn --topo single,4", the command "iperf", and the results "['34.8 Gbits/sec', '34.9 Gbits/sec']".

**nodes of single topology with 1 switch and 4 host**



```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sun 05:34
ubuntu@ubuntu: ~
File Edit View Search Terminal Tabs Help
root@ubuntu: ~/pox x ubuntu@ubuntu: ~ x root@ubuntu: ~/pox x ubuntu@ubuntu: ~/pox x
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['34.8 Gbits/sec', '34.9 Gbits/sec']
containernet>
Interrupt
containernet>
Interrupt
containernet> nodes ←
available nodes are: ←
c0 h1 h2 h3 h4 s1 ←
containernet>
```

net links of single topology with 1 switch and 4 host



project 2 VM [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal Sun 05:51

ubuntu@ubuntu: ~

File Edit View Search Terminal Tabs Help

root@ubuntu: ~/pox x ubuntu@ubuntu: ~ x root@ubuntu: ~/pox x ubuntu@ubuntu: ~/pox x

```
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['34.8 Gbits/sec', '34.9 Gbits/sec']
containernet>
Interrupt
containernet>
Interrupt
containernet> nodes
available nodes are:
c0 h1 h2 h3 h4 s1
containernet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0
c0
containernet> net ←
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0
c0
containernet> █
```

**create a network topology with depth 2 and 8 fanout**

```

project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
ubuntu@ubuntu: ~
root@ubuntu:~/pox x ubuntu@ubuntu: ~ x root@ubuntu:~/pox x ubuntu@ubuntu:~/pox x
Sun 06:04

ubuntu@ubuntu:~$ sudo mn --topo tree,depth=2,fanout=8
[sudo] password for ubuntu:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s1, s5) (s1, s6) (s1, s7) (s1, s8) (s1, s9) (s2, h1) (s2, h2) (s2, h3) (s2, h4) (s2, h5) (s2, h6) (s2, h7) (s2, h8) (s3, h9) (s3, h10) (s3, h11) (s3, h12) (s3, h13) (s3, h14) (s3, h15) (s3, h16) (s4, h17) (s4, h18) (s4, h19) (s4, h20) (s4, h21) (s4, h22) (s4, h23) (s4, h24) (s5, h25) (s5, h26) (s5, h27) (s5, h28) (s5, h29) (s5, h30) (s5, h31) (s5, h32) (s6, h33) (s6, h34) (s6, h35) (s6, h36) (s6, h37) (s6, h38) (s6, h39) (s6, h40) (s7, h41) (s7, h42) (s7, h43) (s7, h44) (s7, h45) (s7, h46) (s7, h47) (s7, h48) (s8, h49) (s8, h50) (s8, h51) (s8, h52) (s8, h53) (s8, h54) (s8, h55) (s8, h56) (s9, h57) (s9, h58) (s9, h59) (s9, h60) (s9, h61) (s9, h62) (s9, h63) (s9, h64)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Starting controller

```

host 1 can ping host 64 successfully

```

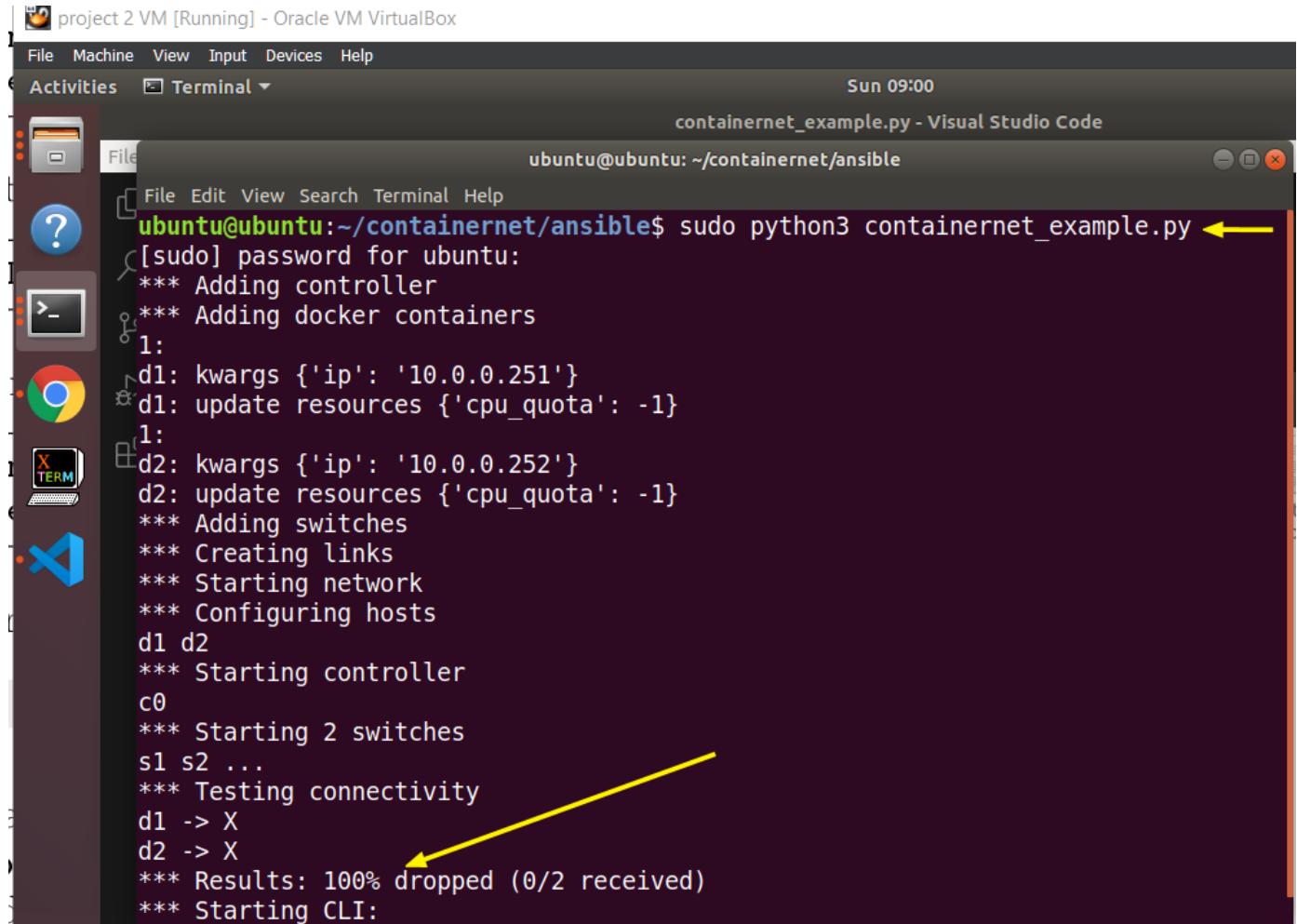
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
ubuntu@ubuntu: ~
root@ubuntu:~/pox x ubuntu@ubuntu: ~ x root@ubuntu:~/pox x ubuntu@ubuntu:~/pox x
Sun 06:06

root@ubuntu:~$ 2, h4) (s2, h5) (s2, h6) (s2, h7) (s2, h8) (s3, h9) (s3, h10) (s3, h11) (s3, h12) (s3, h13) (s3, h14) (s3, h15) (s3, h16) (s4, h17) (s4, h18) (s4, h19) (s4, h20) (s4, h21) (s4, h22) (s4, h23) (s4, h24) (s5, h25) (s5, h26) (s5, h27) (s5, h28) (s5, h29) (s5, h30) (s5, h31) (s5, h32) (s6, h33) (s6, h34) (s6, h35) (s6, h36) (s6, h37) (s6, h38) (s6, h39) (s6, h40) (s7, h41) (s7, h42) (s7, h43) (s7, h44) (s7, h45) (s7, h46) (s7, h47) (s7, h48) (s8, h49) (s8, h50) (s8, h51) (s8, h52) (s8, h53) (s8, h54) (s8, h55) (s8, h56) (s9, h57) (s9, h58) (s9, h59) (s9, h60) (s9, h61) (s9, h62) (s9, h63) (s9, h64)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h62 h63 h64
*** Starting controller
c0
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
containernet> h1 ping -c 2 h64
PING 10.0.0.64 (10.0.0.64) 56(84) bytes of data.
64 bytes from 10.0.0.64: icmp_seq=1 ttl=64 time=127 ms
64 bytes from 10.0.0.64: icmp_seq=2 ttl=64 time=0.705 ms
--- 10.0.0.64 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss time 1002ms
rtt min/avg/max/mdev = 0.705/63.872/127.039/63.167 ms
containernet>

```

# Containernet lab

d1 (host 1 cannot ping d2 (host 2) because there is no link between them



```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sun 09:00
containernet_example.py - Visual Studio Code
ubuntu@ubuntu: ~/containernet/ansible
File Edit View Search Terminal Help
ubuntu@ubuntu:~/containernet/ansible$ sudo python3 containernet_example.py ←
[sudo] password for ubuntu:
*** Adding controller
*** Adding docker containers
1:
d1: kwargs {'ip': '10.0.0.251'}
d1: update resources {'cpu_quota': -1}
1:
d2: kwargs {'ip': '10.0.0.252'}
d2: update resources {'cpu_quota': -1}
*** Adding switches
*** Creating links
*** Starting network
*** Configuring hosts
d1 d2
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Testing connectivity
d1 -> X
d2 -> X
*** Results: 100% dropped (0/2 received)
*** Starting CLI:
```

host 1 can ping host 2) after adding link between the 2 switches and set the bandwidth and TCLink between the 2 switches

Please edit the highlighted portion.

23

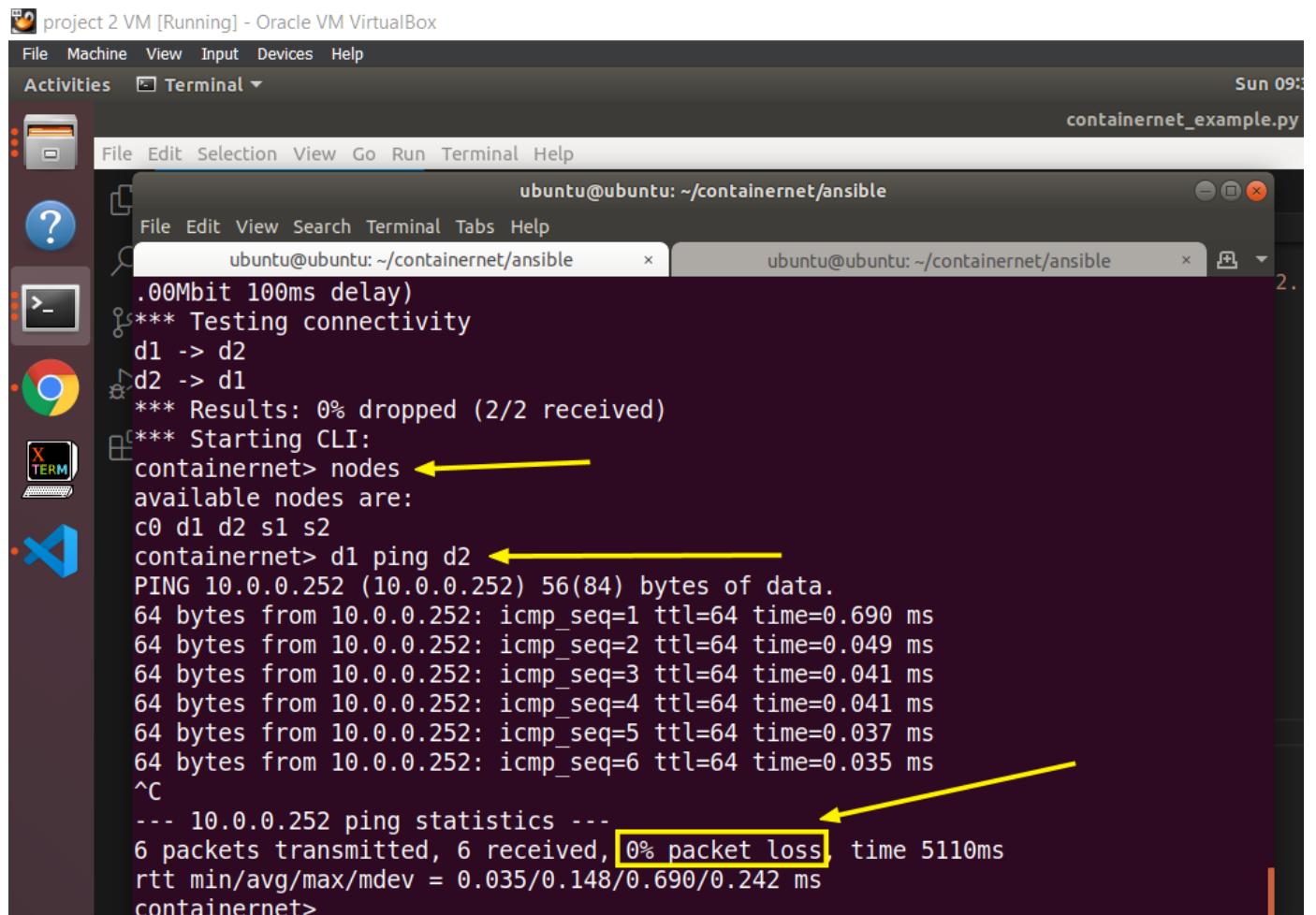
```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
File Edit View Search Terminal Help
*** Stopping network*** Stopping 1 controllers
c0
*** Stopping 3 links
...
s1 s2
*** Stopping 2 switches
d1 d2
*** Done
*** Removing NAT rules of 0 SAPs
ubuntu@ubuntu:~/containernet/ansible$ sudo python3 containernet_example.py
*** Adding controller
*** Adding docker containers
1:
d1: kargs {'ip': '10.0.0.251'}
d1: update resources {'cpu_quota': -1}
1:
d2: kargs {'ip': '10.0.0.252'}
d2: update resources {'cpu_quota': -1}
*** Adding switches
*** Creating links
*** Added link between d1 and d2
*** Added link between S1 and S2
(1.00Mbit 100ms delay) (1.00Mbit 100ms delay) (1.00Mbit 100ms delay) *** Added `TCLink` between s1 and s2 with 100ms delay and 1Mbps bandwidth
*** Starting network
*** Configuring hosts
d1 d2
*** Starting controller
c0
*** Starting 2 switches
s1 (1.00Mbit 100ms delay) s2 (1.00Mbit 100ms delay) ... (1.00Mbit 100ms delay) (1.00Mbit 100ms delay)
*** Testing connectivity
d1 -> d2
d2 -> d1
*** Results: 0% dropped (2/2 received)
*** Starting CLI:
containernet>
```

run sudo mn test --pingall

```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
File Edit View Search Terminal Help
ubuntu@ubuntu:~/containernet/ansible$ sudo mn test --pingall
Usage: mn [options]
(type mn -h for details)

mn: error: no such option: --pingall
ubuntu@ubuntu:~/containernet/ansible$ sudo mn --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.373 seconds
ubuntu@ubuntu:~/containernet/ansible$ ^C
ubuntu@ubuntu:~/containernet/ansible$
```

## d1 ping successfully d2



```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Sun 09:53
containernet_example.py

File Edit Selection View Go Run Terminal Help
ubuntu@ubuntu: ~/containernet/ansible
ubuntu@ubuntu: ~/containernet/ansible 2.
.00Mbit 100ms delay)
*** Testing connectivity
d1 -> d2
d2 -> d1
*** Results: 0% dropped (2/2 received)
*** Starting CLI:
containernet> nodes ←
available nodes are:
c0 d1 d2 s1 s2
containernet> d1 ping d2 ←
PING 10.0.0.252 (10.0.0.252) 56(84) bytes of data.
64 bytes from 10.0.0.252: icmp_seq=1 ttl=64 time=0.690 ms
64 bytes from 10.0.0.252: icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from 10.0.0.252: icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from 10.0.0.252: icmp_seq=4 ttl=64 time=0.041 ms
64 bytes from 10.0.0.252: icmp_seq=5 ttl=64 time=0.037 ms
64 bytes from 10.0.0.252: icmp_seq=6 ttl=64 time=0.035 ms
^C
--- 10.0.0.252 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5110ms
rtt min/avg/max/mdev = 0.035/0.148/0.690/0.242 ms
containernet>
```

added the TLink between s1 and s2 python code

project 2 VM [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Visual Studio Code Sun 09:55

containernet\_example.py - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
  containernet_example.py x
home > ubuntu > containernet > ansible > containernet_example.py
  21  NEXT, you are adding the switches and link the host1 to switch1 and host2 to switch2.
  22  ...
  23  info('*** Adding switches\n')
  24  s1 = net.addSwitch('s1')
  25  s2 = net.addSwitch('s2')
  26  info('*** Creating links\n')
  27  net.addLink(d1, s1)
  28  net.addLink(s2, d2)
  29  net.addLink(d1,d2) # adding a link between d1 (host1) and d2(host2)
  30  info('*** Added link between d1 and d2\n')
  31  net.addLink(s1,s2 )
  32  info('*** Added link between S1 and S2\n')
  33
  34 # Add a `TCLink` between s1 and s2 with 100ms delay and 1Mbps bandwidth ←
  35 net.addLink(s1, s2, cls=TCLink, delay="100ms", bw=1)
  36 info('*** Added `TCLink` between s1 and s2 with 100ms delay and 1Mbps bandwidth\n')
  37
  38 # Next, you will define a function to start the mininet topology
  39
  40 info('*** Starting network\n')
  41 net.start()
  42 # You can pre-identify set a ping request while starting the topology
  43 info('*** Testing connectivity\n')
  44 net.ping([d1, d2])
  45
  46 # Starting the CLI (Computer Telephony Integration ) in the end
  47 CLI(net)
  48
  49 # Stop the network when catch the stop request
  50 info('*** Stopping network')
  51 net.stop()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● ubuntu@ubuntu:~/containernet/ansible$ nautilus ~/containernet/ansible
○ ubuntu@ubuntu:~/containernet/ansible$ []
  ⌂ 0 △ 0 ⌂ 0

```

## Main lab project 2

for more information , please watch my video that demonstrate the demo

<https://www.youtube.com/watch?v=C9x4DCE4aEE>

setup Mininet network and adding the links between containers and check the links between nodes

Please edit the highlighted portion.

26

A screenshot of a Linux desktop environment (Ubuntu) running in Oracle VM VirtualBox. The desktop has several icons in the dock, including a browser, terminal, file manager, and system monitor. A terminal window titled 'pox' is open, showing the command-line interface for a Mininet network. The user is executing commands to add controllers, switches, hosts, and links, and then configuring IP addresses and routes. Yellow arrows highlight specific command lines: 'sudo python3 mininet\_2switches\_2controller\_4hosts.py', 'py net.addLink(c1,s1)', 'py net.addLink(c2,s2)', 'links', and 'inet 10.0.2.10'. The terminal also shows the resulting network topology and host configurations.

```
ubuntu@ubuntu:~/containernet/ansible$ sudo python3 mininet_2switches_2controller_4hosts.py
*** Adding controllers
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding switches
*** Adding hosts
*** Creating links
*** Added link between c1 and s1
*** Added link between c2 and s2
*** Starting network
*** Configuring hosts
h1 h2 h3 h4
*** Running CLI
*** Starting CLI:
containernet> py net.addLink(c1,s1)
<mininet.link.Link object at 0x7f9a4b16d470>
containernet> py net.addLink(c2,s2)
<mininet.link.Link object at 0x7f9a4b16d668>
containernet> h1 ifconfig h1-eth0 10.0.2.10
containernet> h2 ifconfig h2-eth0 10.0.2.20
containernet> h3 ifconfig h3-eth0 192.168.2.30
containernet> h4 ifconfig h4-eth0 192.168.2.40
containernet> h1 ip addr add 192.168.2.10/24 dev h1-eth1
containernet> net
h1 h1-eth0:s1-eth1 h1-eth1:s2-eth3
h2 h2-eth0:s1-eth2
h3 h3-eth0:s2-eth1
h4 h4-eth0:s2-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:c1-eth0
s2 lo: s2-eth1:h3-eth0 s2-eth2:h4-eth0 s2-eth3:h1-eth1 s2-eth4:c2-eth0
c1 c1-eth0:s1-eth3
c2 c2-eth0:s2-eth4
containernet> links
s1-eth1<->h1-eth0 (OK OK)
s1-eth2<->h2-eth0 (OK OK)
s2-eth1<->h3-eth0 (OK OK)
s2-eth2<->h4-eth0 (OK OK)
```

## assign IP address for host 1 and check it

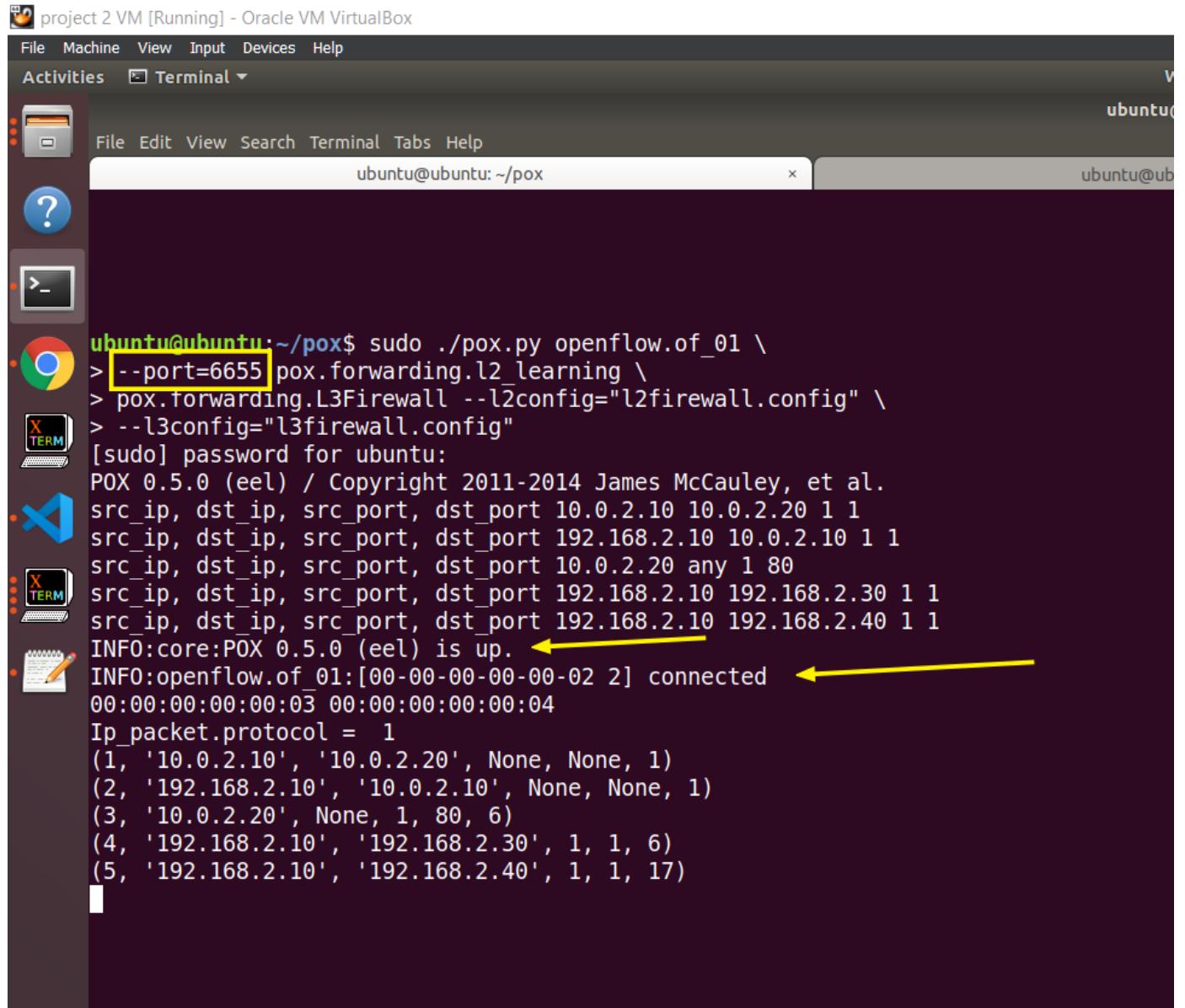
A screenshot of a Linux desktop environment (Ubuntu) running in Oracle VM VirtualBox. The desktop has icons for a browser, terminal, file manager, and system monitor. A terminal window titled 'pox' is open, showing the configuration of host 1's network interface. The user runs 'ifconfig' to check the interface status and 'ip' to set the IP address. Yellow arrows highlight the 'xterm h1 h2 h3 h4' command, the 'inet 10.0.2.10' line in the 'ifconfig' output, and the 'inet 192.168.2.10' line in the 'ip' output. The terminal also shows the resulting configuration of the host 1 interface.

```
ubuntu@ubuntu:~/containernet/ansible$ xterm h1 h2 h3 h4
containernet> xterm h1 h2 h3 h4
containernet> py h1.IP
<bound method Node.IP of <Host h1: h1-eth0:10.0.0.1,h1-eth1:None pid=2746> >
containernet> h1 ifconfig h1-eth1 192.168.2.10
containernet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.10 netmask 255.0.0.0 broadcast 10.255.255.255
        ether 62:c1:52:c3:d0:32 txqueuelen 1000 (Ethernet)
            RX packets 67 bytes 9484 (9.4 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 31 bytes 2714 (2.7 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
h1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.10 netmask 255.255.255.0 broadcast 0.0.0.0
        inet6 fe80::1cae:c2ff:fe66:5f3b prefixlen 64 scopeid 0x20<link>
            ether 1a:ae:c2:66:5f:3b txqueuelen 1000 (Ethernet)
            RX packets 79 bytes 10476 (10.4 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 25 bytes 1930 (1.9 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 15 bytes 1288 (1.2 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 15 bytes 1288 (1.2 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

## controller 1 port 6633 is up and running

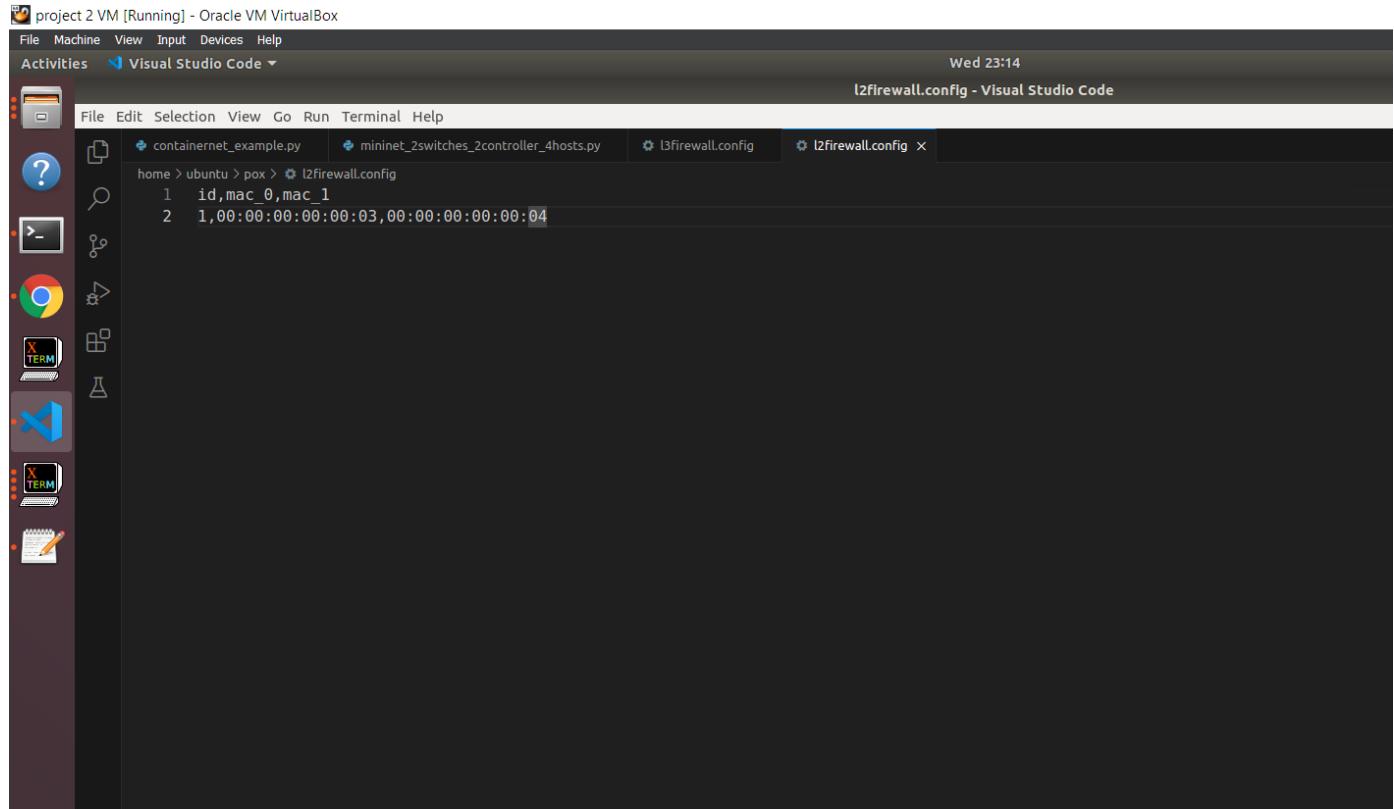
```
ubuntu@ubuntu:~/pox$ sudo ./pox.py openflow.of_01 \
> --port=6633 box.forwarding.l2_learning \
> pox.forwarding.L3Firewall --l2config="l2firewall.config" \
> --l3config="l3firewall.config"
[sudo] password for ubuntu:
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
src_ip, dst_ip, src_port, dst_port 10.0.2.10 10.0.2.20 1 1
src_ip, dst_ip, src_port, dst_port 192.168.2.10 10.0.2.10 1 1
src_ip, dst_ip, src_port, dst_port 10.0.2.20 any 1 80
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.30 1 1
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.40 1 1
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
00:00:00:00:03 00:00:00:00:04
Ip_packet.protocol = 1
(1, '10.0.2.10', '10.0.2.20', None, None, 1)
(2, '192.168.2.10', '10.0.2.10', None, None, 1)
(3, '10.0.2.20', None, 1, 80, 6)
(4, '192.168.2.10', '192.168.2.30', 1, 1, 6)
(5, '192.168.2.10', '192.168.2.40', 1, 1, 17)
^[
```

## controller 2 port 6655 is up and running



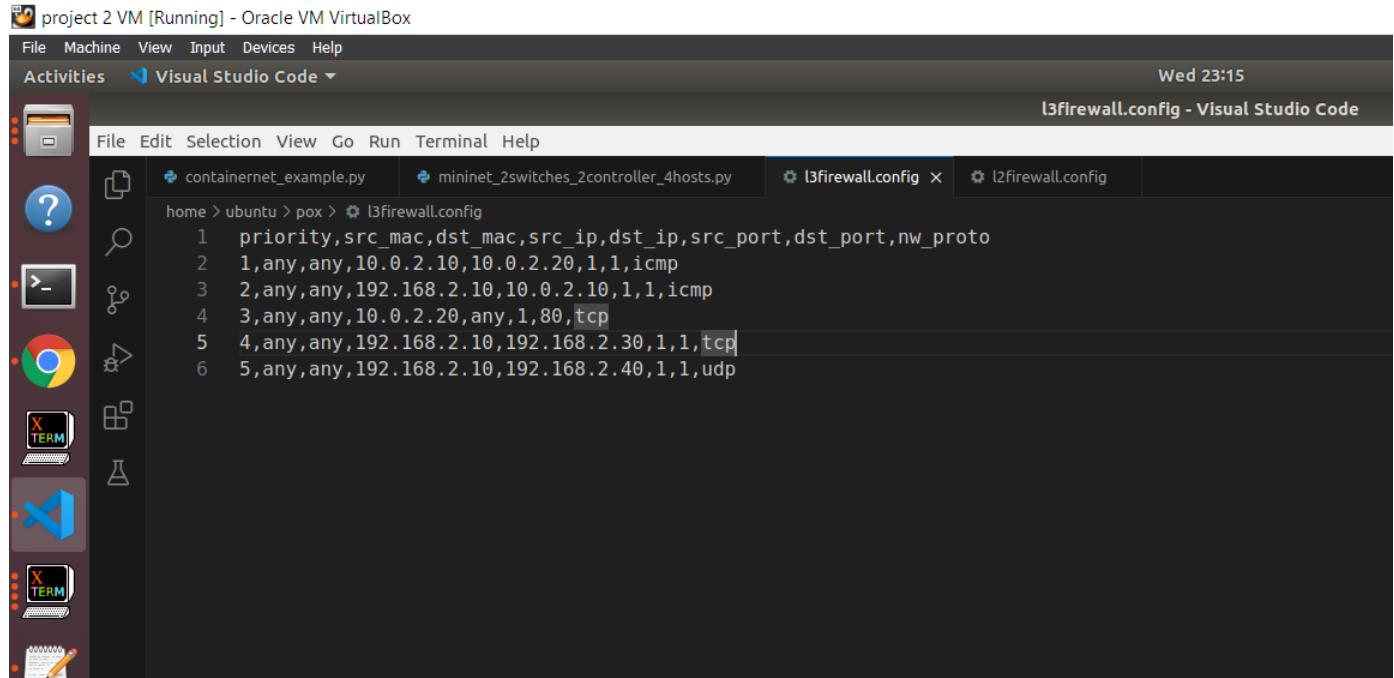
```
ubuntu@ubuntu:~/pox$ sudo ./pox.py openflow.of_01 \
> --port=6655 pox.forwarding.l2_learning \
> pox.forwarding.L3Firewall --l2config="l2firewall.config" \
> --l3config="l3firewall.config"
[sudo] password for ubuntu:
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
src_ip, dst_ip, src_port, dst_port 10.0.2.10 10.0.2.20 1 1
src_ip, dst_ip, src_port, dst_port 192.168.2.10 10.0.2.10 1 1
src_ip, dst_ip, src_port, dst_port 10.0.2.20 any 1 80
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.30 1 1
src_ip, dst_ip, src_port, dst_port 192.168.2.10 192.168.2.40 1 1
INFO:core:POX 0.5.0 (eel) is up. ←
INFO:openflow.of_01:[00-00-00-00-00-02 2] connected ←
00:00:00:00:00:03 00:00:00:00:00:04
Ip_packet.protocol = 1
(1, '10.0.2.10', '10.0.2.20', None, None, 1)
(2, '192.168.2.10', '10.0.2.10', None, None, 1)
(3, '10.0.2.20', None, 1, 80, 6)
(4, '192.168.2.10', '192.168.2.30', 1, 1, 6)
(5, '192.168.2.10', '192.168.2.40', 1, 1, 17)
```

## L2firewall.config file



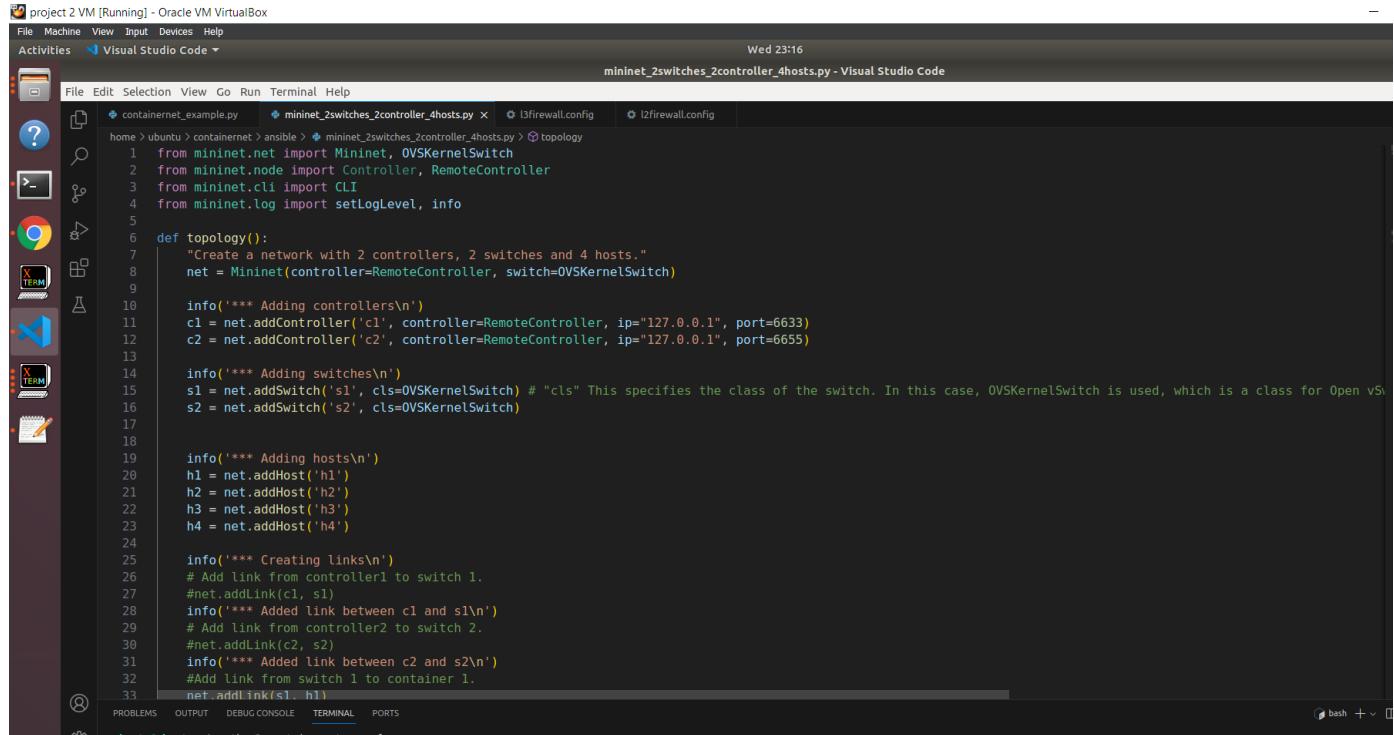
```
1 id,mac_0,mac_1
2 1,00:00:00:00:03,00:00:00:00:04
```

## L3firewall.config file



```
1 priority,src_mac,dst_mac,src_ip,dst_ip,src_port,dst_port,nw_proto
2 1,any,any,10.0.2.10,10.0.2.20,1,1,icmp
3 2,any,any,192.168.2.10,10.0.2.10,1,1,icmp
4 3,any,any,10.0.2.20,any,1,80,tcp
5 4,any,any,192.168.2.10,192.168.2.30,1,1,tcp
6 5,any,any,192.168.2.10,192.168.2.40,1,1,udp
```

## Python file for creating mininet network



The screenshot shows a Visual Studio Code interface running on a Linux host (Ubuntu) within an Oracle VM VirtualBox. The title bar indicates the project is running in a VM. The code editor displays a Python script named `mininet_2switches_2controller_4hosts.py`. The script uses the `Mininet` library to create a network topology with two controllers, two switches, and four hosts. The code includes comments explaining the creation of controllers, switches, hosts, and the links between them. The code editor has a dark theme, and the sidebar shows other files in the project like `containernet_example.py` and configuration files for firewall and topology.

```
project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Visual Studio Code
File Edit Selection View Go Run Terminal Help
File containernet_example.py mininet_2switches_2controller_4hosts.py l3firewall.config l2firewall.config
home > ubuntu > containernet > ansible > mininet_2switches_2controller_4hosts.py topology
1 from mininet.net import Mininet, OVSKernelSwitch
2 from mininet.node import Controller, RemoteController
3 from mininet.cli import CLI
4 from mininet.log import setLogLevel, info
5
6 def topology():
7     "Create a network with 2 controllers, 2 switches and 4 hosts."
8     net = Mininet(controller=RemoteController, switch=OVSKernelSwitch)
9
10    info('*** Adding controllers\n')
11    c1 = net.addController('c1', controller=RemoteController, ip="127.0.0.1", port=6633)
12    c2 = net.addController('c2', controller=RemoteController, ip="127.0.0.1", port=6655)
13
14    info('*** Adding switches\n')
15    s1 = net.addSwitch('s1', cls=OVSKernelSwitch) # "cls" This specifies the class of the switch. In this case, OVSKernelSwitch is used, which is a class for Open vSwitch
16    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
17
18    info('*** Adding hosts\n')
19    h1 = net.addHost('h1')
20    h2 = net.addHost('h2')
21    h3 = net.addHost('h3')
22    h4 = net.addHost('h4')
23
24    info('*** Creating links\n')
25    # Add link from controller1 to switch 1.
26    #net.addLink(c1, s1)
27    info('*** Added link between c1 and s1\n')
28    # Add link from controller2 to switch 2.
29    #net.addLink(c2, s2)
30    info('*** Added link between c2 and s2\n')
31    #Add link from switch 1 to container 1.
32    net.addLink(s1, h1)
33
```

The screenshot shows a running Oracle VM VirtualBox instance titled "project 2 VM [Running]". Inside the VM, a Visual Studio Code window is open, displaying a Python script. The script is a network configuration file, likely for Mininet, defining links between controllers (c1, c2) and switches (s1, s2) and hosts (h1, h2, h3, h4). It also starts the network and runs a CLI. A terminal window in the VM shows an attempt to run the script, which fails because the file does not exist.

```
# Add link from controller1 to switch 1.  
#net.addLink(c1, s1)  
info('*** Added link between c1 and s1\n')  
# Add link from controller2 to switch 2.  
#net.addLink(c2, s2)  
info('*** Added link between c2 and s2\n')  
#Add link from switch 1 to container 1.  
net.addLink(s1, h1)  
#Add link from switch 1 to container 2.  
net.addLink(s1, h2)  
#Add link from switch 2 to container 3.  
net.addLink(s2, h3)  
#Add link from switch 2 to container 4.  
net.addLink(s2, h4)  
#Add link from switch 2 to container 1.  
net.addLink(s2, h1)  
  
# net.addLink(h1, s1)  
# net.addLink(h2, s1)  
# net.addLink(h3, s2)  
# net.addLink(h4, s2)  
# net.addLink(s1, s2)  
  
info('*** Starting network\n')  
net.build()  
c1.start()  
c2.start()  
s1.start([c1])  
s2.start([c2])  
  
info('*** Running CLI\n')  
CLT(net)
```

ubuntu@ubuntu:~\$ python3 containernet\_example.py  
python3: can't open file 'containernet\_example.py': [Errno 2] No such file or directory

## Xterm terminals of 4 hosts

```

project 2 VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities XTerm
ubuntu@ubuntu: ~/pox
Wed 23:28
"Node: h1"
64 bytes from 192.168.2.30: icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from 192.168.2.30: icmp_seq=4 ttl=64 time=0.066 ms
^C
--- 192.168.2.30 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.061/0.151/0.355/0.120 ms
root@ubuntu:"/containernet/ansible# ping -I 192.168.2.10 192.1
68.2.30
> --por
PING 192.168.2.30 (192.168.2.30) from 192.168.2.10 : 56(84) byte
s of data.
> pox.f
64 bytes from 192.168.2.30: icmp_seq=1 ttl=64 time=0.272 ms
> --l3c
64 bytes from 192.168.2.30: icmp_seq=2 ttl=64 time=0.154 ms
[sudo] 64 bytes from 192.168.2.30: icmp_seq=3 ttl=64 time=0.389 ms
POX 0.5
--- 192.168.2.30 ping statistics ---
src_ip, 3 packets transmitted, 3 received 0% packet loss, time 2048ms
src_ip, rtt min/avg/max/mdev = 0.154/0.271/0.389/0.097 ms
src_ip, root@ubuntu:"/containernet/ansible# ping 8.8.8.8
src_ip, connect: Network is unreachable
src_ip, root@ubuntu:"/containernet/ansible# 
src_ip, usl_ip, src_port, dst_port 192.100.2.10 192.100.2.40
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow of 01:[00-00-00-00-00-02] connected
00:00:
"Node: h4"
22:47:02.205145 IP6 fe80::5c9e:c9ff:fe4d:4ef.mdns > ff02::fb.mdn
(1, )s: 0 [9q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ftp._tcp.local.
(2, )PTR (QM)? _webdav._tcp.local. PTR (QM)? _webdavs._tcp.local. PTR
(3, )_(QM)? _sftp-ssh._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)?
(4, )_afpovertcp._tcp.local. PTR (QM)? _nfs._tcp.local. PTR (QM)? _ipp
(5, )_tcp.local. (141)
22:51:35.489127 IP6 fe80::c864:47ff:fe22:a602.mdns > ff02::fb.mdn
(1, )s: 0 [9q] PTR (QM)? _ipps._tcp.local. PTR (QM)? _ftp._tcp.local.
PTR (QM)? _webdav._tcp.local. PTR (QM)? _webdavs._tcp.local. PTR
(2, )_(QM)? _sftp-ssh._tcp.local. PTR (QM)? _smb._tcp.local. PTR (QM)?
_afpovertcp._tcp.local. PTR (QM)? _nfs._tcp.local. PTR (QM)? _ipp
(3, )_tcp.local. (141)
22:55:56.875390 IP6 fe80::1cae:c2ff:fe66:5f3b > ip6-allrouters: I
CMP6, router solicitation, length 16

```

```

"Node: h2"
0 packets dropped by kernel
root@ubuntu:"/containernet/ansible# curl http://google.com
curl: (6) Could not resolve host: google.com
root@ubuntu:"/containernet/ansible# curl http://cnn.com
curl: (6) Could not resolve host: cnn.com
root@ubuntu:"/containernet/ansible# curl http://0.0.0.0:80/
curl: (7) Failed to connect to 0.0.0.0 port 80: Connection refused
root@ubuntu:"/containernet/ansible# ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
mtu 1500
inet 10.0.2.20 netmask 255.0.0.0 broadcast
10.255.255.255
ether ee:3d:9b:cb:11:50 txqueuelen 1000 (Ethernet)
RX packets 71 bytes 9708 (9.7 KB)
RX errors 0 dropped 0 overruns 0 frame 0

```

```

"Node: h3"
22:09:02.027251 IP 192.168.2.10 > ubuntu: ICMP echo request, id 3474, seq 1, length 64
22:09:02.027340 IP ubuntu > 192.168.2.10: ICMP echo reply, id 3474, seq 1, length 64
22:09:03.057226 IP 192.168.2.10 > ubuntu: ICMP echo request, id 3474, seq 2, length 64
22:09:03.057339 IP ubuntu > 192.168.2.10: ICMP echo reply, id 3474, seq 2, length 64
22:09:04.075798 IP 192.168.2.10 > ubuntu: ICMP echo request, id 3474, seq 3, length 64
22:09:04.076089 IP ubuntu > 192.168.2.10: ICMP echo reply, id 3474, seq 3, length 64
22:09:07.279126 ARP, Request who-has ubuntu tell 192.1
68.2.10, length 28
22:09:07.279268 ARP, Reply ubuntu is-at a6:4a:c5:d5:1c

```

## Ping from h1 to h2 ( it cannot ping)

Please edit the highlighted portion.

33

"Node: h1"

```
root@ubuntu:~/containernet/ansible# ping 10.0.2.20
PING 10.0.2.20 (10.0.2.20) 56(84) bytes of data.
^C
--- 10.0.2.20 ping statistics ---
20 packets transmitted, 0 received, 100% packet loss time 1946
6ms
```

"Node: h2"

```
root@ubuntu:~/containernet/ansible# tcp dump
Command 'tcp' not found, but there are 29 similar one
s.

root@ubuntu:~/containernet/ansible# tcpcdump
tcpcdump: verbose output suppressed, use -v or -vv for
full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), ca
pture size 262144 bytes
21:50:56.834184 ARP, Request who-has ubuntu tell 10.0
.2.10, length 28
21:50:56.834205 ARP, Reply ubuntu is-at ee:3d:9b:cb:1
1:50 (oui Unknown), length 28
21:51:13.867019 IP6 fe80::c03c:81ff:fe3d:456 > ip6-al
lroutes: ICMP6, router solicitation, length 16
21:51:29.683576 IP6 fe80::c03c:81ff:fe3d:456.mdns > f
02::fb.mdns: 0 [9q] PTR (QM)? _ipps._tcp.local. PTR
(QM)? _ftp._tcp.local. PTR (QM)? _webdav._tcp.local.
PTR (QM)? _sftp._tcp.local. PTR (QM)? _webdav._tcp.local.
PTR (QM)? _sftp._tcp.local. PTR (QM)? _sftp-ssh._t
```

"Node: h4"

```
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow of A1:[00-00-00-00-A0-02] connected
00:00:
```

"Node: h3"

```
22:09:02.027251 IP 192.168.2.10 > ubuntu: ICMP echo re
quest, id 3474, seq 1, length 64
22:09:02.027340 IP ubuntu > 192.168.2.10: ICMP echo re
ply, id 3474, seq 1, length 64
22:09:03.057226 IP 192.168.2.10 > ubuntu: ICMP echo re
quest, id 3474, seq 2, length 64
22:09:03.057339 IP ubuntu > 192.168.2.10: ICMP echo re
ply, id 3474, seq 2, length 64
22:09:04.075798 IP 192.168.2.10 > ubuntu: ICMP echo re
quest, id 3474, seq 3, length 64
22:09:04.076089 IP ubuntu > 192.168.2.10: ICMP echo re
ply, id 3474, seq 3, length 64
22:09:07.279126 ARP, Request who-has ubuntu tell 192.1
68.2.10, length 28
22:09:07.279268 ARP, Reply ubuntu is-at a6:4a:c5:d5:1c
```

h2 cannot send connection on port 80 (HTTP) as made a python server on h1

The screenshot shows four terminal windows in a Linux desktop environment:

- "Node: h1"**: Shows ping statistics to 10.0.2.10 and a command to start an HTTP server on port 80.
- "Node: h2"**: Shows curl commands failing to resolve google.com and cnn.com, and an ifconfig output for interface h2-eth0.
- "Node: h3"**: Shows a sequence of ICMP echo requests and replies between Node h3 and Node h1.
- "Node: h4"**: Shows a detailed log of IP traffic, including many ICMP echo requests and replies, and several router solicitation messages.

for more information , please watch my video that demonstrate the demo

<https://www.youtube.com/watch?v=C9x4DCE4aEE>

## V. CONCLUSION

I have learned many amazing practical things for SDN concepts and I applied them using real tools such as Mininet and POX controller and python programming language, also it was really nice to use containers as hosts and simulate a complete virtual network in a linux environment

## VI. APPENDIX B: ATTACHED FILES

Find the files on my github

<https://github.com/Mohammed-Ragab/CSE-548-advanced-network-security/tree/main/project%202%20files>

## VII. REFERENCES

- [Mininet VM Setup Notes - Mininet](#)