Mawlana Bhashani Science and Technology University

Lab-Report

Report No :
Lab report name : Warm up exercises.
Course code : ICT-3208
Course title : Network Planning and Designing Lab
Date of Performance :
Date of Submission :

Submitted by

Name: Iqbal Hossen
ID: IT-18041
3rd year 2nd semester
Session: 2017-18
Dept. of ICT
MBSTU.

Submitted To

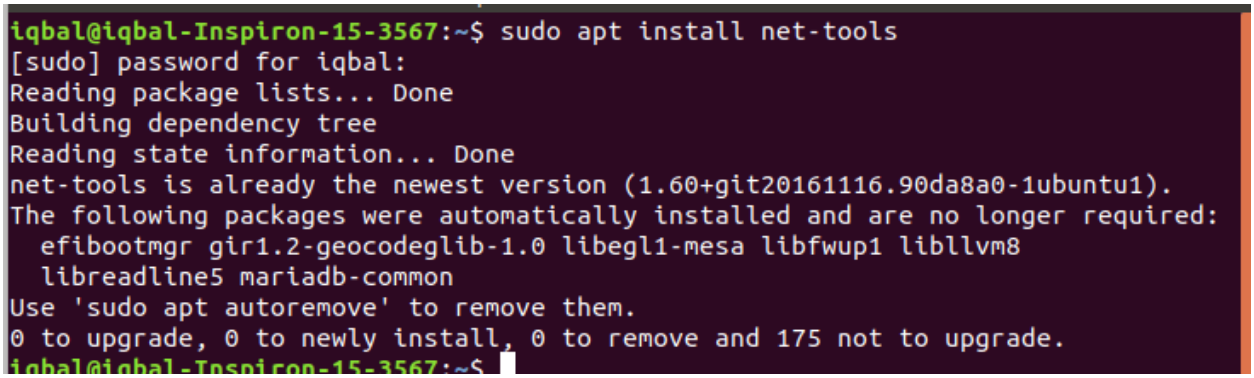
Nazrul Islam
Assistant Professor
Dept. of ICT
MBSTU.

Experiment No:01

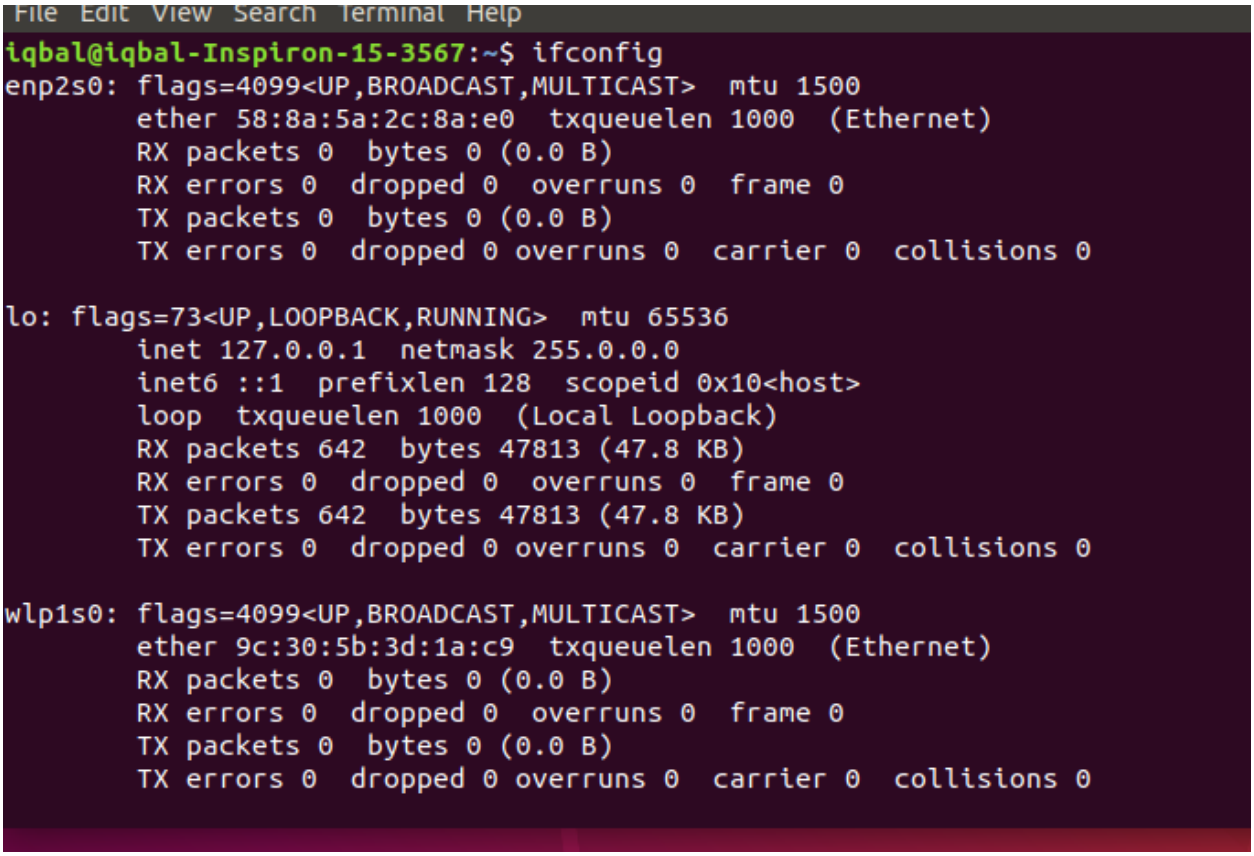
Experiment Name: Warm up exercises on Introduction to mininet and command.

Installing net tool:

command : sudo apt install net-tools



ifconfig : stands for "**interface configuration**." It is used to view and change the configuration of the network interfaces on your system.



ifconfig -a: Displays the configuration of all interfaces, both active and inactive.

```
File  Edit  View  Search  Terminal  Help
iqbal@iqbal-Inspiron-15-3567:~$ ifconfig -a
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 58:8a:5a:2c:8a:e0  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

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 643  bytes 47959 (47.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 643  bytes 47959 (47.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp1s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 9c:30:5b:3d:1a: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
```

Installing mininet:

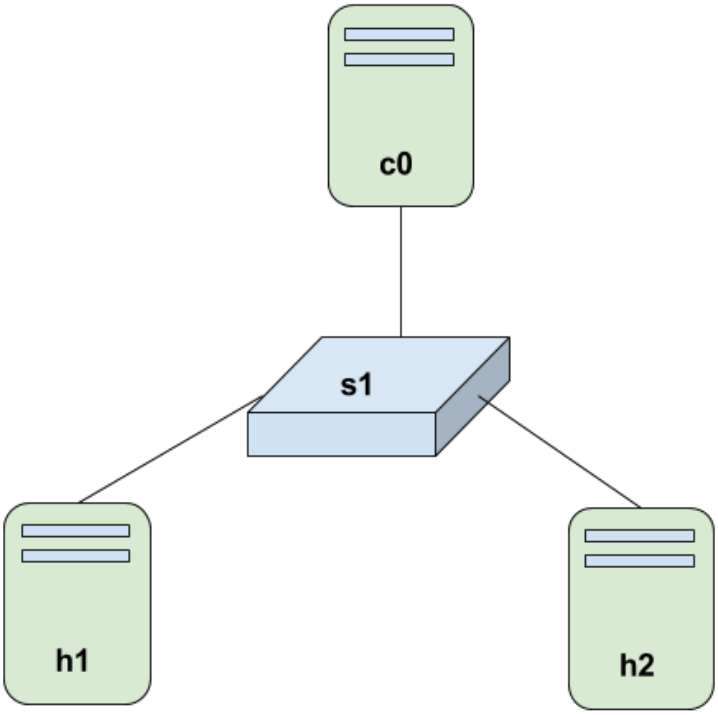
command: sudo apt-get install mininet

```
iqbal@iqbal-Inspiron-15-3567:~$ sudo apt-get install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
mininet is already the newest version (2.2.2-2ubuntu1).
The following packages were automatically installed and are no longer required:
  efibootmgr gir1.2-geocodeglib-1.0 libegl1-mesa libfwup1 libllvm8
  libreadline5 mariadb-common
Use 'sudo apt autoremove' to remove them.
0 to upgrade, 0 to newly install, 0 to remove and 172 not to upgrade.
iqbal@iqbal-Inspiron-15-3567:~$
```

Create Virtual Network

We will be using CLI(**sudo mn command**) to manage our virtual network. The default topology includes two hosts (h1,h2), OpenFlow Switch(s1) and OpenFlow controller(c0).

This command launches a simple 2 host, 1 controller, 1 switch topology. The following diagram outlines the topology.



Mininet Commands:

Help command: Run the help option to view the list of commands available:

```
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>
```

nodes command: Shows the nodes we created with the simple Mininet command:

```
mininet> nodes
available nodes are:
h1 h2 s1
mininet>
```

net command: This command shows the links of all nodes

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
```

dump command: which displays summary information about all devices:

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=8799>
<Host h2: h2-eth0:10.0.0.2 pid=8801>
<OVSBridge s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=8806>
```

py command: *Mininet Command*-Line Interface (CLI**)** *Commands*

```
mininet> py
unexpected EOF while parsing (<string>, line 0)
mininet> EOF
```

command line is py , then that command is executed with *Python***.**

X command:

```
mininet> x
usage: x node [cmd args]...
```

Time command:

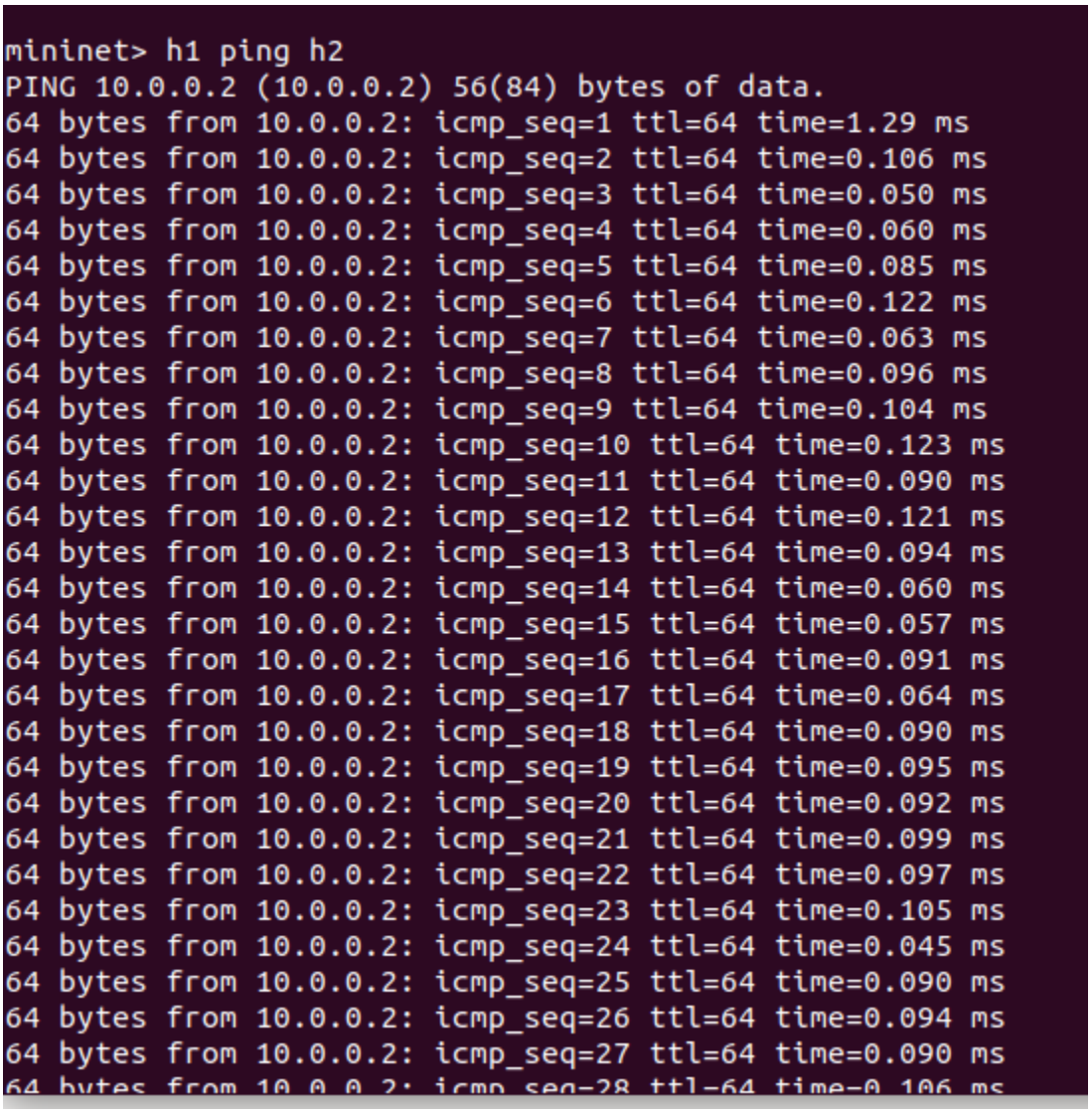
```
mininet> time
*** Elapsed time: 0.000017 secs
mininet>
```

h1 ifconfig -a: This command will display the IP address, broadcast address and MAC address of the host ***h1. The command*** h2 ifconfig -a are also as the command

```
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::687d:49ff:fefd:c910  prefixlen 64  scopeid 0x20<link>
        ether 6a:7d:49:fd:c9:10  txqueuelen 1000  (Ethernet)
        RX packets 49  bytes 5994 (5.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1006 (1.0 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
```

ping command: We can test connectivity between the host by using ping command.



Pingall command: This command will make each host in the network ping every other host in the network. In the network that we have, ***h1*** will ping ***h2***, and ***h2*** will ping ***h1.***

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

xterm command: The command to open the xterm window is:

```
mininet> xtream
*** Unknown command: xtream
mininet> xterm
usage: xterm node1 node2 ...
mininet>
```

exit command:

```
mininet> exit
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 368.451 seconds
```

EOF command:

```
mininet> EOF

*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 55.768 seconds
iqbal@iqbal-Inspiron-15-3567:~$
```

create topology : topo=single,4 that means controller , switch is one and host are 4 like h1,h2,h3,h4cleanup command: If Mininet crashes for some reason, clean it up:

```
iqbal@iqbal-Inspiron-15-3567:~$ sudo mn --topo=single,4
[sudo] password for iqbal:
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** 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

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

cleanup command: If Mininet crashes for some reason, clean it up:

```
iqbal@iqbal-Inspiron-15-3567:~$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ov
s-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd
 ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

Discussion:

By doing this lab ,we can learn about followings objectives:

1.Learn the basic commands in Mininet

2.Learn how to create basic network topologies in Mininet

3. Learn Mininet API

This walkthrough demonstrates most Mininet commands, as well as its typical usage in concert with the Wireshark dissector.