Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 01

Course code: ICT-3208

Course title: Computer Network Lab

Date of Performance:01-12-20

Date of Submission:18-12-20

Submitted by

Name: Moniruzzaman

ID:IT-18007

3rd year 2ndsemester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No: 01

Experiment Name : Basic mininet commands

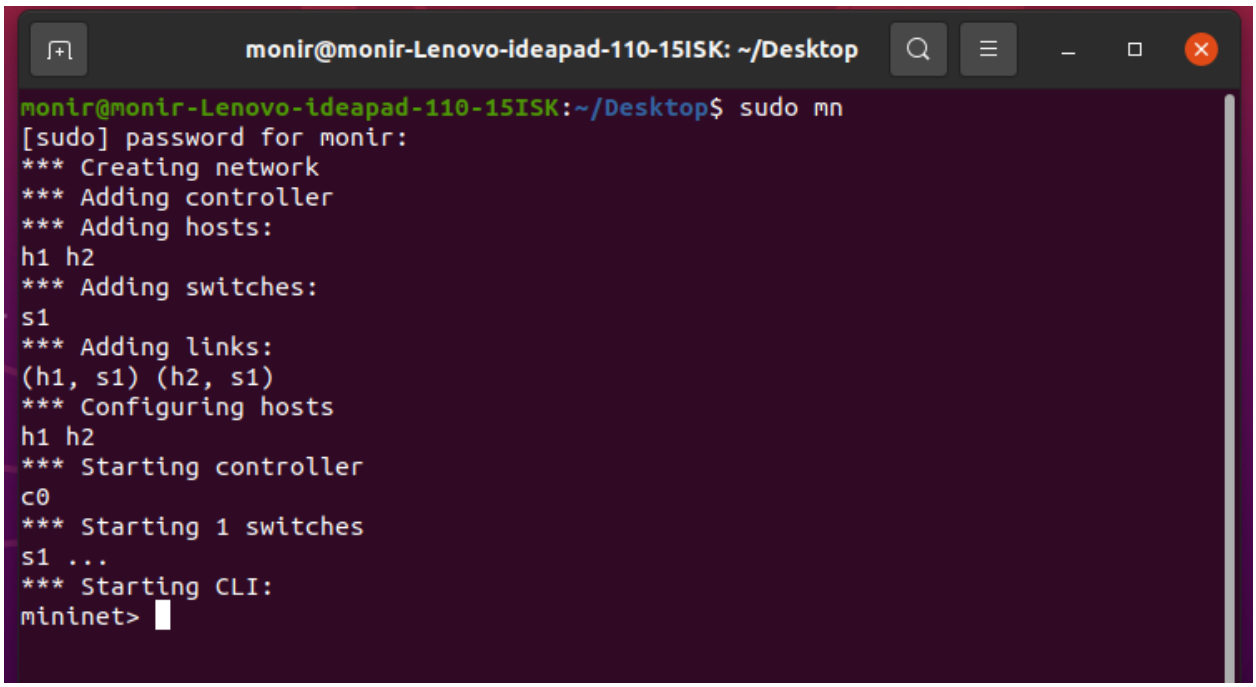
Create Virtual Network :

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).

Interact with Hosts and Switches :

Start a minimal topology and enter the CLI :

$ sudo mn



When issuing the sudo mn command, Mininet initializes the topology and launches its command line interface which looks like this:

mininet >

Again Display Mininet CLI commands:

mininet> help

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
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>
```

To display the available nodes, type the following command:

mininet> nodes

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

Display links: **mininet>**

net

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

Dump information about all nodes: **mininet> dump**

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=11348>
<Host h2: h2-eth0:10.0.0.2 pid=11350>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=11355>
<Controller c0: 127.0.0.1:6653 pid=11341>
mininet>
```

If the first string typed into the Mininet CLI is a host, switch or controller name, the command is executed on that node. Run a command on a host process: **mininet> h1 ifconfig -a**

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
mininet> s1 ifconfig -a
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether fc:45:96:91:48:9f  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 801  bytes 73615 (73.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 801  bytes 73615 (73.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 7e:64:b6:56:fd:15  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 2a:ac:08:fd:d2:4c  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 20  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::cd5:cdff:fe8c:a6b9  prefixlen 64  scopeid 0x20<link>
        ether 0e:d5:cd:8c:a6:b9  txqueuelen 1000  (Ethernet)
```

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 **Test 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 command shown below. **mininet> h1 ping 10.0.0.2**

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop

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=14.8 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.736 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.059 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.117 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.122 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=0.118 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.134 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.117 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.122 ms
64 bytes from 10.0.0.2: icmp_seq=18 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=21 ttl=64 time=0.119 ms
64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=0.038 ms
64 bytes from 10.0.0.2: icmp_seq=23 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=24 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=25 ttl=64 time=0.117 ms
64 bytes from 10.0.0.2: icmp_seq=26 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=27 ttl=64 time=0.121 ms
64 bytes from 10.0.0.2: icmp_seq=28 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=29 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=30 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=31 ttl=64 time=0.115 ms
```

This command tests the connectivity between host h1 and host h2. To stop the test, press Ctrl+c .

Stop the emulation by typing the following command: **mininet> exit**

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 12.111 seconds
monir@monir-Lenovo-ideapad-110-15ISK:~/Desktop$
```

Mininet crashes for some reason, clean it up by the following command: **$ sudo mn -c**

```
monir@monir-Lenovo-ideapad-110-15ISK: ~/Desktop
monir@monir-Lenovo-ideapad-110-15ISK:~/Desktop$ sudo mn -c
[sudo] password for monir:
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd
ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflo
wd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/nul
l
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.
monir@monir-Lenovo-ideapad-110-15ISK:~/Desktop$
```

Discussion :

Mininet is a network emulator which creates realistic virtual network . From this lab how to install mininet successfully . And I have also learn the basic command and procedure of mininet from this lab.