MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

DEPARTMENT OF ICT

Lab Report No : 04

Course Code : ICT-3208

Course Title : Network Planning and Designing Lab

Lab Report Name : Mininet Lab

Submitted by	***Submitted to***
Name: Ashik Mahmud ID : IT-17009 Session : 2016-2017 3rd Year 2nd Semester	Nazrul Islam Assistant Professor, Department of ICT,MBSTU Santosh,Tangail-1902

Objective: In this lab we will learn about installation process of Mininet in Linux. After completion of installation .Apply some mininet command from Mininet Workthrough.

Installation Process

1. $sudo apt-get install git

```
smmhossain@samim ~ $ sudo apt-get install git
[sudo] password for smmhossain:
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.17.1-1ubuntu0.7).
The following packages were automatically installed and are no longer required:
  efibootmgr gyp javascript-common libfwup1 libhttp-parser2.7.1 libjs-async
  libjs-inherits libjs-jquery libjs-node-uuid libjs-underscore libllvm9
  libssl1.0-dev libuv1 libuv1-dev node-abbrev node-ansi node-ansi-color-table
  node-archy node-async node-balanced-match node-block-stream
  node-brace-expansion node-builtin-modules node-combined-stream
  node-concat-map node-cookie-jar node-delayed-stream node-forever-agent
  node-form-data node-fs.realpath node-fstream node-fstream-ignore
  node-github-url-from-git node-glob node-graceful-fs node-gyp
  node-hosted-git-info node-inflight node-inherits node-ini
  node-is-builtin-module node-isexe node-json-stringify-safe node-lockfile
  node-lru-cache node-mime node-minimatch node-mkdirp node-mute-stream
  node-node-uuid node-nopt node-normalize-package-data node-npmlog node-once
  node-osenv node-path-is-absolute node-pseudomap node-qs node-read
  node-read-package-json node-request node-retry node-rimraf node-semver
  node-sha node-slide node-spdx-correct node-spdx-expression-parse
  node-spdx-license-ids node-tar node-tunnel-agent node-underscore
  node-validate-npm-package-license node-which node-wrappy node-yallist
```

2. $sudo mn

```
smmhossain@samim ~ $ sudo mn
*** 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>
```

3. 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
```

4. mininet>nodes

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

5. mininet>net

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

6. mininet> h1 ifconfig -a

```
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::685b:22ff:fec2:9df8  prefixlen 64  scopeid 0x20<link>
        ether 6a:5b:22:c2:9d:f8  txqueuelen 1000  (Ethernet)
        RX packets 45  bytes 5512 (5.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 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

mininet>
```

7.mininet> s1 ifconfig -a

```
mininet> s1 ifconfig -a
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 58:8a:5a:2c:90:3b  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 28373  bytes 2474289 (2.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28373  bytes 2474289 (2.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ovs-system: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 42:07:c4:7c:11:aa  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
```

8.mininet> h1 ps -a

```
mininet> h1 ps -a
  PID TTY          TIME CMD
  725 tty2     00:17:53 chrome
 1278 tty2     00:01:56 chrome
 1384 tty2     00:00:00 deja-dup-monito
 3349 tty1     00:00:00 gnome-session-b
 3355 tty1     00:00:37 gnome-shell
 3412 tty1     00:00:00 Xwayland
 3521 tty1     00:00:00 ibus-daemon
 3525 tty1     00:00:00 ibus-dconf
 3528 tty1     00:00:00 ibus-x11
 4436 tty1     00:00:00 gsd-xsettings
 4437 tty1     00:00:00 gsd-a11y-settin
 4439 tty1     00:00:00 gsd-clipboard
 4440 tty1     00:00:02 gsd-color
 4462 tty1     00:00:00 gsd-datetime
 4463 tty1     00:00:00 gsd-housekeepin
 4467 tty1     00:00:00 gsd-keyboard
 4470 tty1     00:00:00 gsd-media-keys
 4471 tty1     00:00:00 gsd-mouse
 4479 tty1     00:00:00 gsd-power
 4482 tty1     00:00:00 gsd-print-notif
 4485 tty1     00:00:00 gsd-rfkill
 4492 tty1     00:00:00 gsd-screensaver
```

9. mininet> s1 ps -a

```
mininet> s1 ps -a
  PID TTY          TIME CMD
  725 tty2     00:17:53 chrome
 1278 tty2     00:01:57 chrome
 1384 tty2     00:00:00 deja-dup-monito
 3349 tty1     00:00:00 gnome-session-b
 3355 tty1     00:00:37 gnome-shell
 3412 tty1     00:00:00 Xwayland
 3521 tty1     00:00:00 ibus-daemon
 3525 tty1     00:00:00 ibus-dconf
 3528 tty1     00:00:00 ibus-x11
 4436 tty1     00:00:00 gsd-xsettings
 4437 tty1     00:00:00 gsd-a11y-settin
 4439 tty1     00:00:00 gsd-clipboard
 4440 tty1     00:00:02 gsd-color
 4462 tty1     00:00:00 gsd-datetime
 4463 tty1     00:00:00 gsd-housekeepin
 4467 tty1     00:00:00 gsd-keyboard
 4470 tty1     00:00:00 gsd-media-keys
 4471 tty1     00:00:00 gsd-mouse
 4479 tty1     00:00:00 gsd-power
 4482 tty1     00:00:00 gsd-print-notif
 4485 tty1     00:00:00 gsd-rfkill
```

10. mininet> h1 ping -c 1 h1

```
mininet> h1 ping -c 1 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=1.21 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.210/1.210/1.210/0.000 ms
mininet>
```

11. mininet>pingall

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

12. mininet> exit

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

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 624.014 seconds
```

13. $sudo mn -c

```
smmhossain@samim ~ $ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd
 ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openfl
owd 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
```

14. $sudo mn –test pingpair

```
smmhossain@samim ~ $ sudo mn --test pingpair
*** 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 ...
*** Waiting for switches to connect
s1
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 0.275 seconds
```

15. $sudo mn –test iperf

```
smmhossain@samim ~ $ sudo mn --test iperf
*** 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 ...
*** Waiting for switches to connect
s1
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['32.7 Gbits/sec', '32.7 Gbits/sec']
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 5.967 seconds
```

16. $ sudo mn --test pingall --topo single,3

```
smmhossain@samim ~ $ sudo mn --test pingall --topo single, 3
Usage: mn [options]
(type mn -h for details)

The mn utility creates Mininet network from the command line. It can create
parametrized topologies, invoke the Mininet CLI, and run tests.

Options:
  -h, --help            show this help message and exit
  --switch=SWITCH       default|ivs|lxbr|ovs|ovsbr|ovsk|user[,param=value...]
                        ovs=OVSSwitch default=OVSSwitch ovsk=OVSSwitch
                        lxbr=LinuxBridge user=UserSwitch ivs=IVSSwitch
                        ovsbr=OVSBridge
  --host=HOST           cfs|proc|rt[,param=value...]
                        rt=CPULimitedHost{'sched': 'rt'} proc=Host
                        cfs=CPULimitedHost{'sched': 'cfs'}
  --controller=CONTROLLER
                        default|none|nox|ovsc|ref|remote|ryu[,param=value...]
                        ovsc=OVSController none=NullController
                        remote=RemoteController default=DefaultController
                        nox=NOX ryu=Ryu ref=Controller
  --link=LINK           default|ovs|tc|tcu[,param=value...] default=Link
                        ovs=OVSLink tcu=TCULink tc=TCLink
  --topo=TOPO           linear|minimal|reversed|single|torus|tree[,param=value
                        ...] linear=LinearTopo torus=TorusTopo tree=TreeTopo
                        single=SingleSwitchTopo
```

17. $sudo mn --test pingall --topo linear,4

```
smmhossain@samim ~ $ sudo mn --test pingall --topo linear,4
*** 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 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller

*** Starting 4 switches
s1 s2 s3 s4 ...
*** Waiting for switches to connect
s1 s2 s3 s4
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 0 controllers

*** Stopping 7 links
.......
*** Stopping 4 switches
s1 s2 s3 s4
```

18. $ sudo mn --link tc,bw=10,delay=10ms

```
smmhossain@samim ~ $ sudo mn --link tc,bw=10,delay=10ms
*** 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:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay) (1
0.00Mbit 10ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...(10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet>
```

19. $ sudo mn -v debug

```
smmhossain@samim ~ $ sudo mn -v debug
*** errRun: ['which', 'controller']
  1*** errRun: ['which', 'ovs-controller']
  1*** errRun: ['which', 'test-controller']
  1*** errRun: ['which', 'ovs-testcontroller']
  1*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** errRun: ['grep', '-c', 'processor', '/proc/cpuinfo']
4
  0*** Setting resource limits
*** Creating network
*** Adding controller
*** Adding hosts:
*** errRun: ['which', 'mnexec']
/usr/bin/mnexec
  0*** errRun: ['which', 'ifconfig']
/sbin/ifconfig
  0*** h1 : ('unset HISTFILE; stty -echo; set +m',)
unset HISTFILE; stty -echo; set +m
h1 *** h2 : ('unset HISTFILE; stty -echo; set +m',)
unset HISTFILE; stty -echo; set +m
h2
*** Adding switches:
*** errRun: ['which', 'ovs-vsctl']
/usr/bin/ovs-vsctl
  0*** errRun: ['ovs-vsctl', '-t', '1', 'show']
50e2088d-4fec-4010-9515-819f2c85561a
```