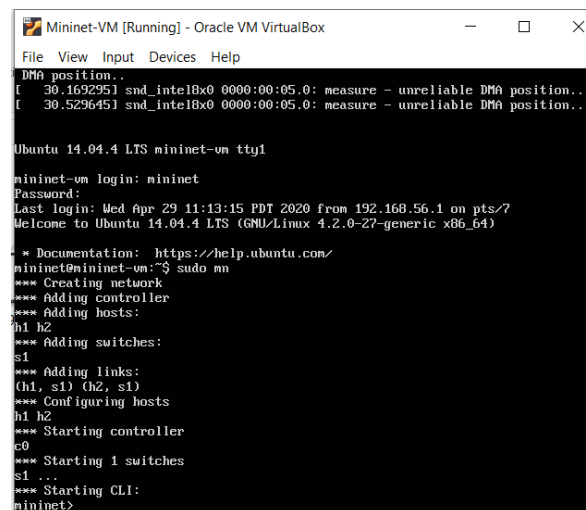


Practical 7- Defining data, control and application planes in software defined networking in Openflow.

→ Mininet is a network emulation software that allows you to launch a virtual network with switches, hosts and an SDN controller all with single command.

→ In order to start a minimal topology, following command is used:

- `sudo mn`



```

Mininet-VM [Running] - Oracle VM VirtualBox
File View Input Devices Help
DMA position..
[ 30.169295] snd_intel8x0 0000:00:05.0: measure - unreliable DMA position..
[ 30.529645] snd_intel8x0 0000:00:05.0: measure - unreliable DMA position..

Ubuntu 14.04.4 LTS mininet-vm tty1
mininet-vm login: mininet
Password:
Last login: Wed Apr 29 11:13:15 PDT 2020 from 192.168.56.1 on pts/7
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
mininet@mininet-vm:~$ sudo mn
*** 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
*** Starting CLI:
mininet>
  
```

Figure 1: A minimal topology

→ Here, mn is a launch script that executes python.

→ The topology in figure 1 consists of two hosts, one switch and one central controller. The setup is shown in figure 2.

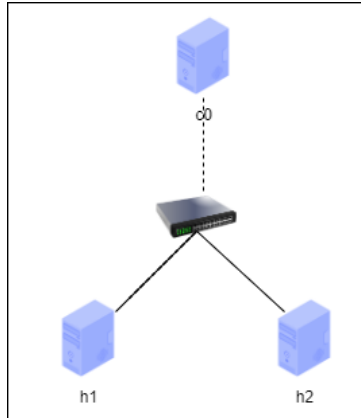


Figure 2

→ We can check out all the nodes available by simply typing the “nodes” command.

- mininet> nodes

→ We can check out the links in the network with the help of command:

- mininet> net

→ We can display all the information about the nodes with the help of:

- mininet> dump

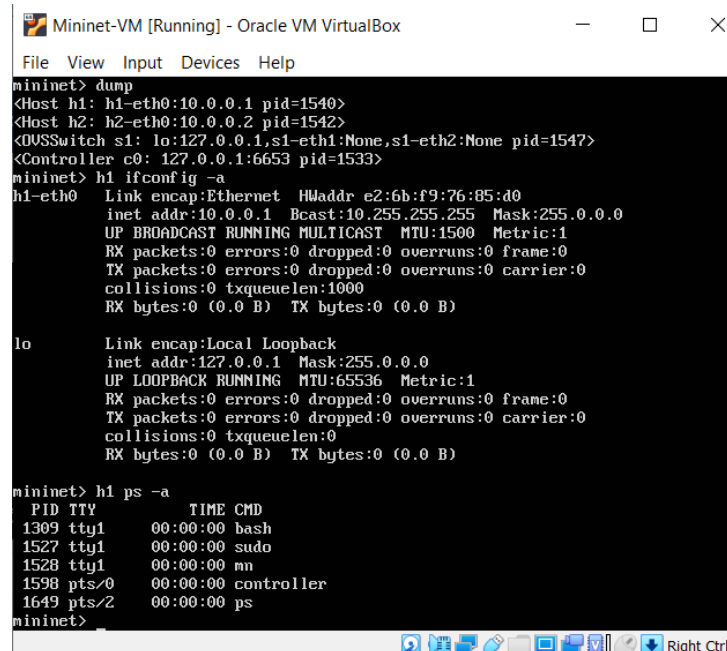
```

Mininet-VM [Running] - Oracle VM VirtualBox
File View Input Devices Help
mininet@mininet-vm:~$ sudo mn
*** 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 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 s1
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
<Host h1: h1-eth0:10.0.0.1 pid=1540>
<Host h2: h2-eth0:10.0.0.2 pid=1542>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1547>
<Controller c0: 127.0.0.1:6653 pid=1533>
mininet>
  
```

Figure 3

→ We can check the ip addresses of nodes and print out process list from a host using following commands:

- mininet> h1 ifconfig -a
- mininet> h1 ps -a



```

Mininet-VM [Running] - Oracle VM VirtualBox
File View Input Devices Help
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1540>
<Host h2: h2-eth0:10.0.0.2 pid=1542>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1547>
<Controller c0: 127.0.0.1:6653 pid=1533>
mininet> h1 ifconfig -a
h1-eth0  Link encap:Ethernet  HWaddr e2:6b:f9:76:85:d0
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

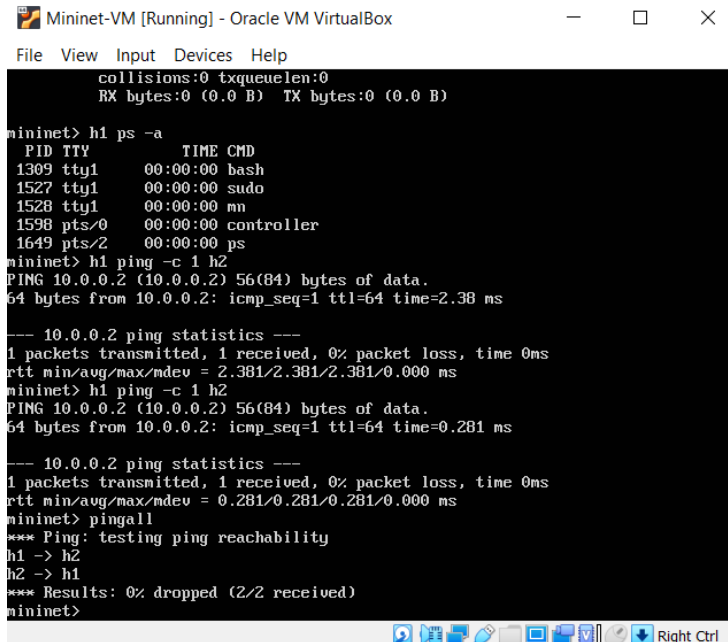
mininet> h1 ps -a
  PID TTY          TIME CMD
 1309 tty1        00:00:00 bash
 1527 tty1        00:00:00 sudo
 1528 tty1        00:00:00 mn
 1598 pts/0        00:00:00 controller
 1649 pts/2        00:00:00 ps
mininet>

```

Figure 4: Host ip address and process list

→ Now we test connectivity between hosts by writing ping command as shown in the figure 5.

→ We can observe in figure 5 that, during the first time the hosts are pinged, the time taken is 2.38ms. This is because the openflow switch sends it first packet's header and forwards to the controller for actions to perform. The controller responds with appropriate actions like in this case forwarding packet to host 2. This entry then remains cached in the switch for certain time. So, the next time packet is sent from h1 to h2 it goes directly to its destination without going to the controller.



```

Mininet-VM [Running] - Oracle VM VirtualBox
File View Input Devices Help
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

mininet> h1 ps -a
  PID TTY          TIME CMD
 1309 tty1      00:00:00 bash
 1527 tty1      00:00:00 sudo
 1528 tty1      00:00:00 mn
 1598 pts/0      00:00:00 controller
 1649 pts/2      00:00:00 ps
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=2.38 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.381/2.381/2.381/0.000 ms
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=0.281 ms

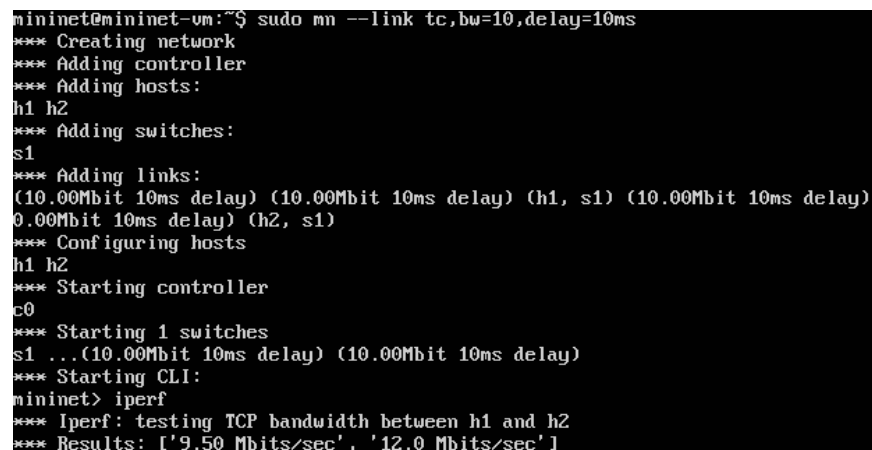
--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.281/0.281/0.281/0.000 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>

```

Figure 5: Testing connectivity between hosts

→ We can configure links between our hosts like changing bandwidth and introducing delay as follows:

- `sudo mn -link tc,bw=[bandwidth],delay=[delay_in_millisecond]`



```

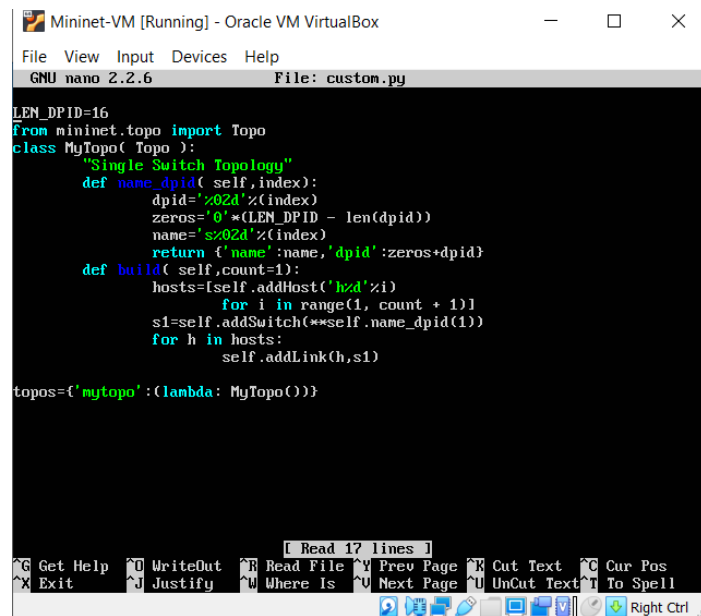
mininet@mininet-vm:~$ sudo mn --link tc,bw=10,delay=10ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay)
(10.00Mbit 10ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ... (10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.50 Mbits/sec', '12.0 Mbits/sec']

```

Figure 6: Configure link

→ With iperf command we can see tcp bandwidth between hosts as shown in figure 6.

→ Now we create our own custom topology as shown in figure 7.



```

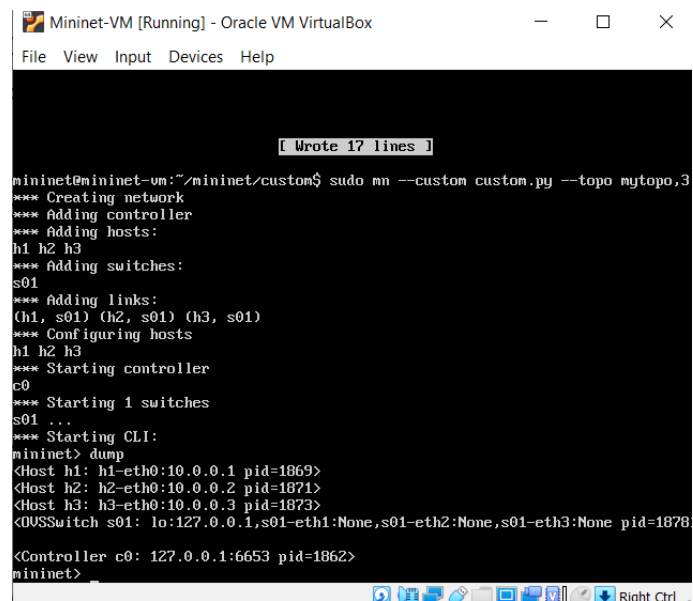
Mininet-VM [Running] - Oracle VM VirtualBox
File View Input Devices Help
GNU nano 2.2.6 File: custom.py

LEN_DPID=16
from mininet.topo import Topo
class MyTopo( Topo ):
    "Single Switch Topology"
    def name_dpid( self, index ):
        dpid='%02d'%(index)
        zeros='0'*(LEN_DPID - len(dpid))
        name='%s%02d'%(index)
        return ( name, 'dpid':zeros+dpid)
    def build( self, count=1 ):
        hosts=[self.addHost('h%d'%i)
                for i in range(1, count + 1)]
        s1=self.addSwitch(**self.name_dpid(1))
        for h in hosts:
            self.addLink(h,s1)

topos={'mytopo':(lambda: MyTopo())}
  
```

Figure 7: Custom topology

→ Now we deploy the topology as shown in figure 8.



```

Mininet-VM [Running] - Oracle VM VirtualBox
File View Input Devices Help

mininet@mininet-vm:~/mininet/custom$ sudo mn --custom custom.py --topo mytopo,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s01
*** Adding links:
(h1, s01) (h2, s01) (h3, s01)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s01 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1869>
<Host h2: h2-eth0:10.0.0.2 pid=1871>
<Host h3: h3-eth0:10.0.0.3 pid=1873>
<OUSSwitch s01: lo:127.0.0.1,s01-eth1:None,s01-eth2:None,s01-eth3:None pid=1878>
<Controller c0: 127.0.0.1:6653 pid=1862>
mininet>
  
```

Figure 8: Deploying the custom topology