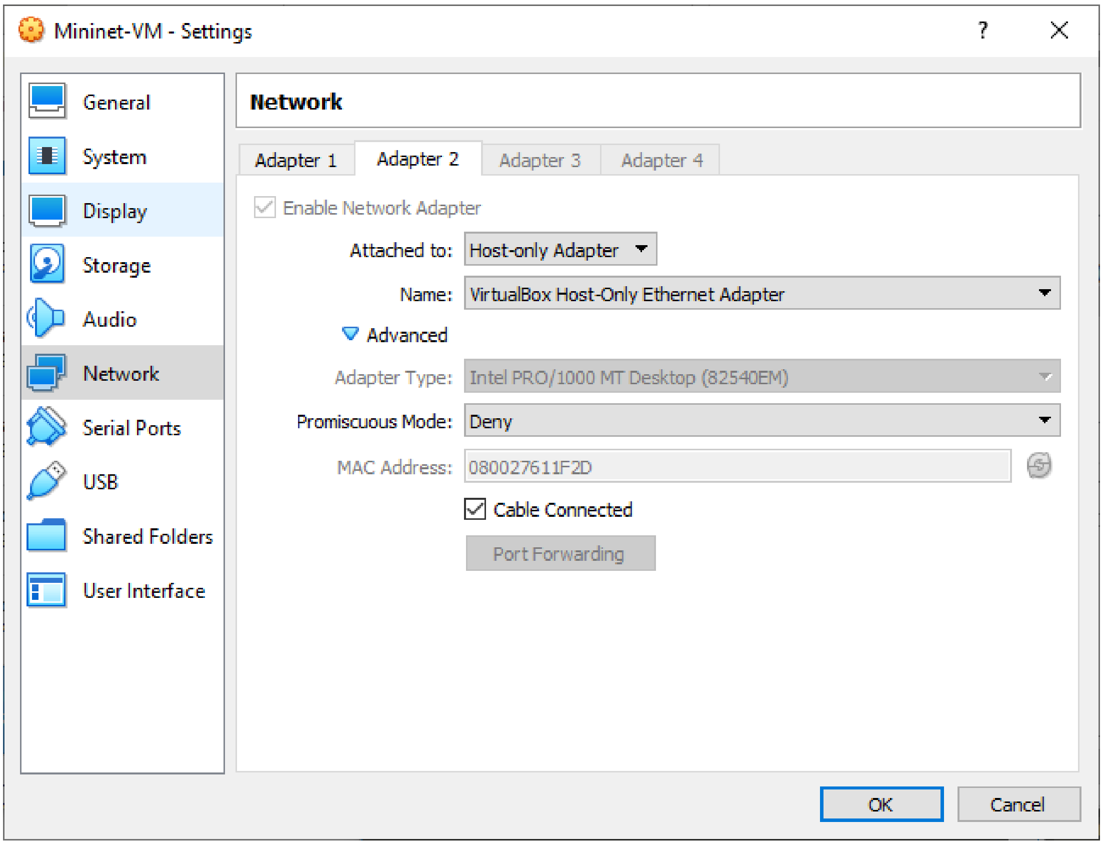
1. Prerequisites

1. Operational Host Only Network Adapter.

Enable Host Only Network Adapter in VirtualBox through VM Settings > Network. Apply settings while VM is turned off, otherwise settings will be greyed out as shown.



☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 080027611F2D

☑ Cable Connected

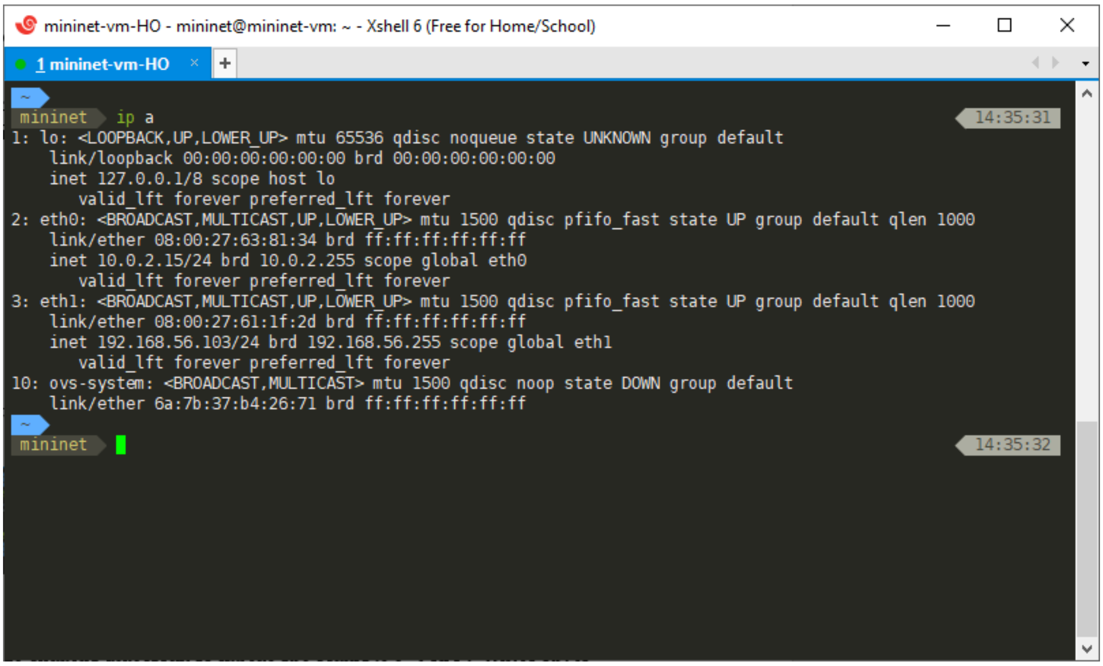
Fig. 1.1: Adding Host Only Network Adapter as Adapter 2.

2. Connection (SSH session) to VM via Host Only Network Adapter.

First, enable the interface associated with Host Only Network Adapter and assign it an IP address through the locally running DHCP sever on guest machine.

```
sudo ifconfig eth1 up
sudo dhclient eth1
```

Then, SSH to the VM via above interface.


mininet-vm-HO - mininet@mininet-vm: ~ - Xshell 6 (Free for Home/School)
1 mininet-vm-HO

```
~
mininet  ip a                                                                                    14:35:31
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:63:81:34 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:61:1f:2d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global eth1
       valid_lft forever preferred_lft forever
10: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
    link/ether 6a:7b:37:b4:26:71 brd ff:ff:ff:ff:ff:ff
~
mininet                                                                                          14:35:32
```

Fig. 1.2: SSH connection established via eth1 interface.

3. Implementation of the Topology in Mininet.

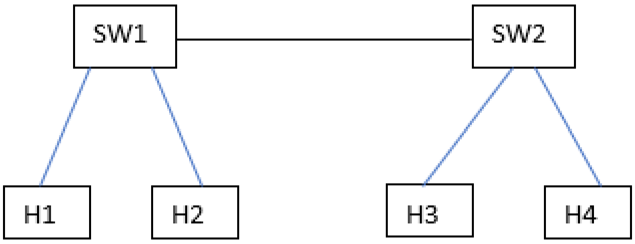
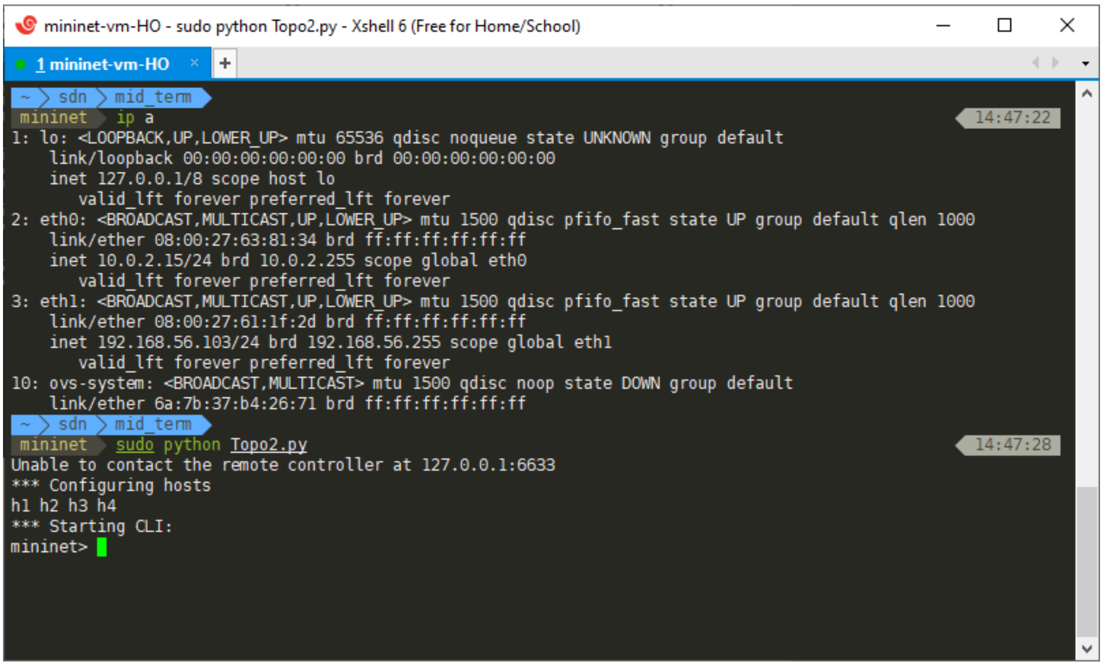


Fig. 3.1: Topology to be implemented in Mininet.

To implement the topology (Fig.3.1) in Mininet, simply execute the **Topo2.py** script as shown.

```
sudo python Topo2.py
```


mininet-vm-HO - sudo python Topo2.py - Xshell 6 (Free for Home/School)
1 mininet-vm-HO
+

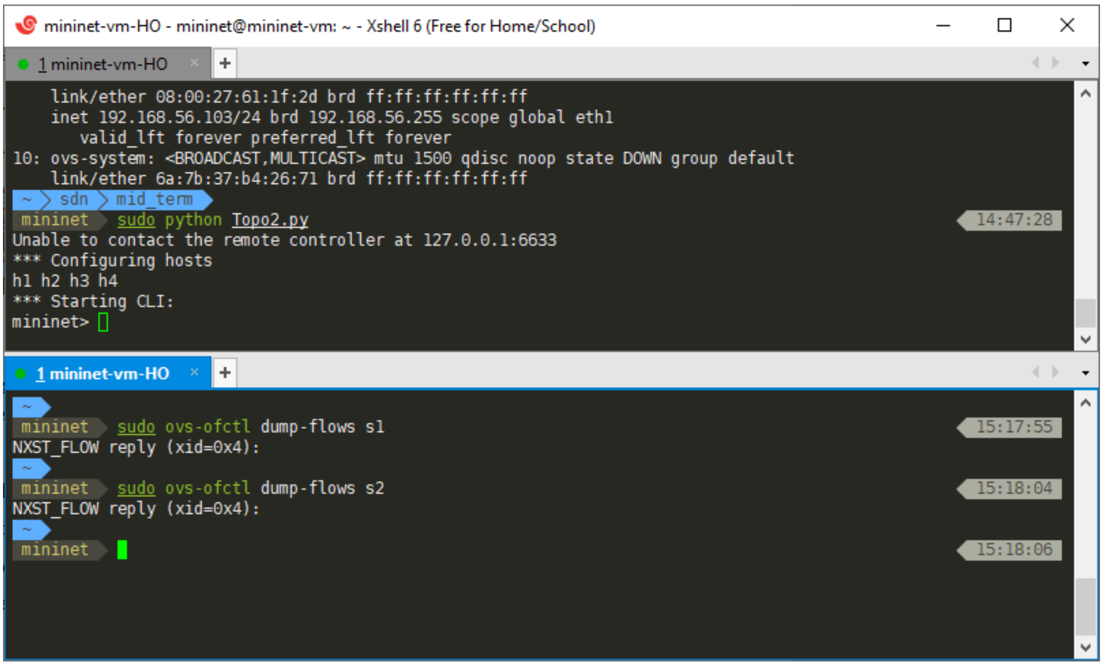
```
~ 〉 sdn 〉 mid_term
mininet   ip a                                                                                14:47:22
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:63:81:34 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:61:1f:2d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global eth1
       valid_lft forever preferred_lft forever
10: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
    link/ether 6a:7b:37:b4:26:71 brd ff:ff:ff:ff:ff:ff
~ 〉 sdn 〉 mid_term
mininet   sudo python Topo2.py                                                                14:47:28
Unable to contact the remote controller at 127.0.0.1:6633
*** Configuring hosts
h1 h2 h3 h4
*** Starting CLI:
mininet>
```

Fig. 1.3: Creating the topology in Mininet.

2. Part A - Installing OpenFlow Rules Directly on Switches

1. Checking switch forwarding tables to make sure no forwarding rules are initially available.

```
sudo ovs-ofctl dump-flows s1
sudo ovs-ofctl dump-flows s2
```



mininet-vm-HO - mininet@mininet-vm: ~ - Xshell 6 (Free for Home/School)
1 mininet-vm-HO
1 mininet-vm-HO

```
    link/ether 08:00:27:61:1f:2d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global eth1
       valid_lft forever preferred_lft forever
10: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
    link/ether 6a:7b:37:b4:26:71 brd ff:ff:ff:ff:ff:ff
 ~  sdn  mid_term
 mininet  sudo python Topo2.py                                                    14:47:28
Unable to contact the remote controller at 127.0.0.1:6633
*** Configuring hosts
h1 h2 h3 h4
*** Starting CLI:
mininet>
```

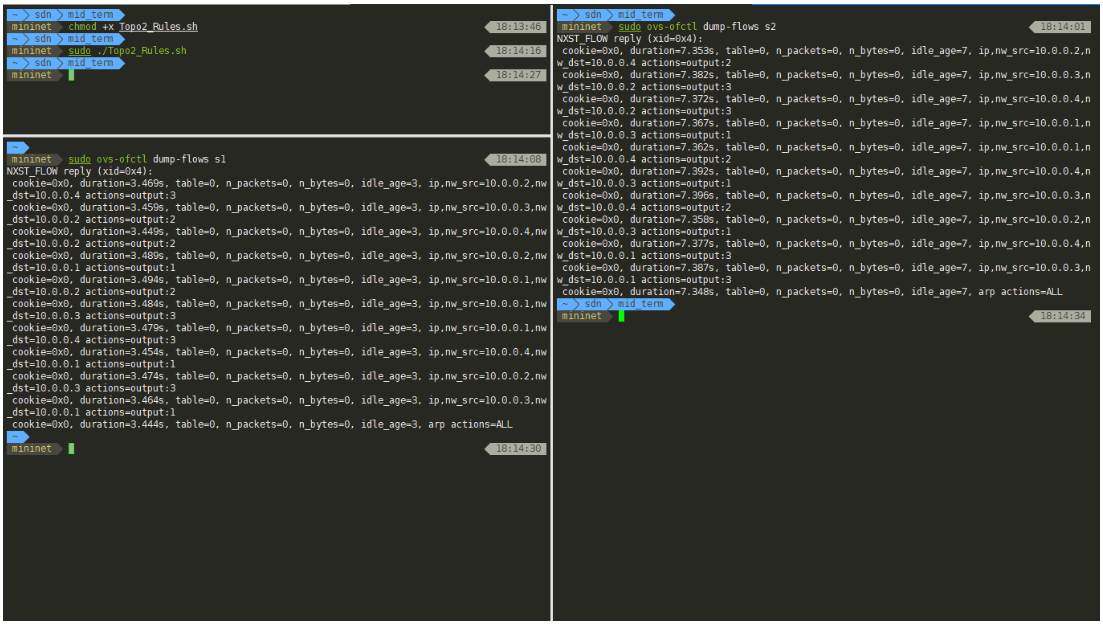
```
 ~
 mininet  sudo ovs-ofctl dump-flows s1                                            15:17:55
NXST_FLOW reply (xid=0x4):
 ~
 mininet  sudo ovs-ofctl dump-flows s2                                            15:18:04
NXST_FLOW reply (xid=0x4):
 ~
 mininet                                                                          15:18:06
```

Fig. 2.1: Making sure no rules are available in switches.

2. Write forwarding rules directly to switches.

Instead of typing in forwarding rules for each switch, automate the menial task by executing the script **Topo2_Rules.sh** as follows.

```
chmod +x Topo2_Rules.sh
sudo ./Topo2_Rules.sh
```



```
~ > sdn > mid_term
mininet > chmod +x Topo2_Rules.sh                                                          18:13:46
~ > sdn > mid_term
mininet > sudo ./Topo2_Rules.sh                                                            18:14:16
~ > sdn > mid_term
mininet >                                                                                  18:14:27
```

```
~
mininet > sudo ovs-ofctl dump-flows s1                                                     18:14:08
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=3.469s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.2,nw
_dst=10.0.0.4 actions=output:3
 cookie=0x0, duration=3.459s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.3,nw
_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=3.449s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.4,nw
_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=3.489s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.2,nw
_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=3.494s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.1,nw
_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=3.484s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.1,nw
_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=3.479s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.1,nw
_dst=10.0.0.4 actions=output:3
 cookie=0x0, duration=3.454s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.4,nw
_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=3.474s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.2,nw
_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=3.464s, table=0, n_packets=0, n_bytes=0, idle_age=3, ip,nw_src=10.0.0.3,nw
_dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=3.444s, table=0, n_packets=0, n_bytes=0, idle_age=3, arp actions=ALL
~
mininet >                                                                                  18:14:30
```

```
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s2                                                     18:14:01
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=7.353s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.2,n
w_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=7.382s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.3,n
w_dst=10.0.0.2 actions=output:3
 cookie=0x0, duration=7.372s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.4,n
w_dst=10.0.0.2 actions=output:3
 cookie=0x0, duration=7.367s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.1,n
w_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=7.362s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.1,n
w_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=7.392s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.4,n
w_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=7.396s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.3,n
w_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=7.358s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.2,n
w_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=7.377s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.4,n
w_dst=10.0.0.1 actions=output:3
 cookie=0x0, duration=7.387s, table=0, n_packets=0, n_bytes=0, idle_age=7, ip,nw_src=10.0.0.3,n
w_dst=10.0.0.1 actions=output:3
 cookie=0x0, duration=7.348s, table=0, n_packets=0, n_bytes=0, idle_age=7, arp actions=ALL
~ > sdn > mid_term
mininet >                                                                                  18:14:34
```

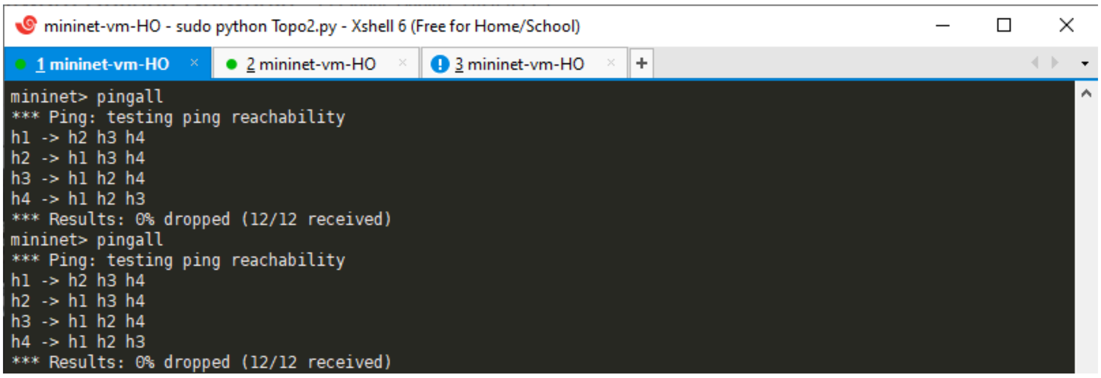
Fig. 2.2: Directly writing forwarding rules to switches with bash script.

3. Checking forwarding operation in practice.

In order to make sure the rules were written successfully, view rules in both switches again as in step 1 (Part A) above. To test forwarding in operation, exchanging ICMP and IP packets between hosts could be done.

- **Testing ICMP packet delivery between all hosts**

```
pingall # At mininet prompt
```

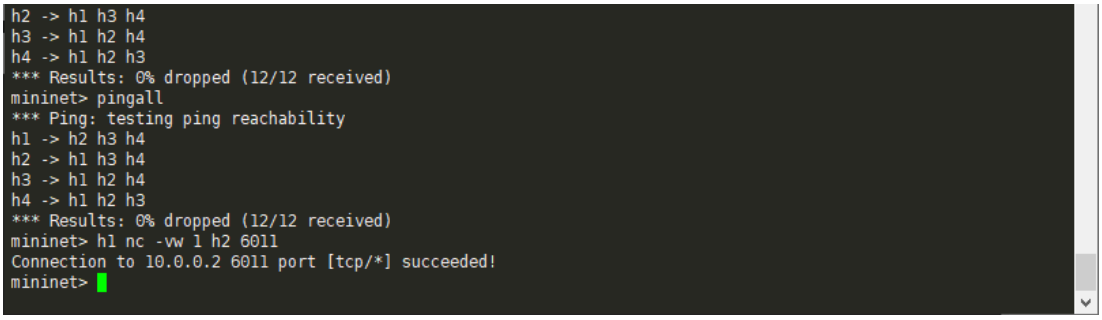

mininet-vm-HO - sudo python Topo2.py - Xshell 6 (Free for Home/School)
1 mininet-vm-HO
2 mininet-vm-HO
3 mininet-vm-HO

```
mininet> pingall
*** 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)
mininet> pingall
*** 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)
```

Fig. 2.3: Ping result between all hosts.

- **Testing TCP packet delivery from H1 to H2**

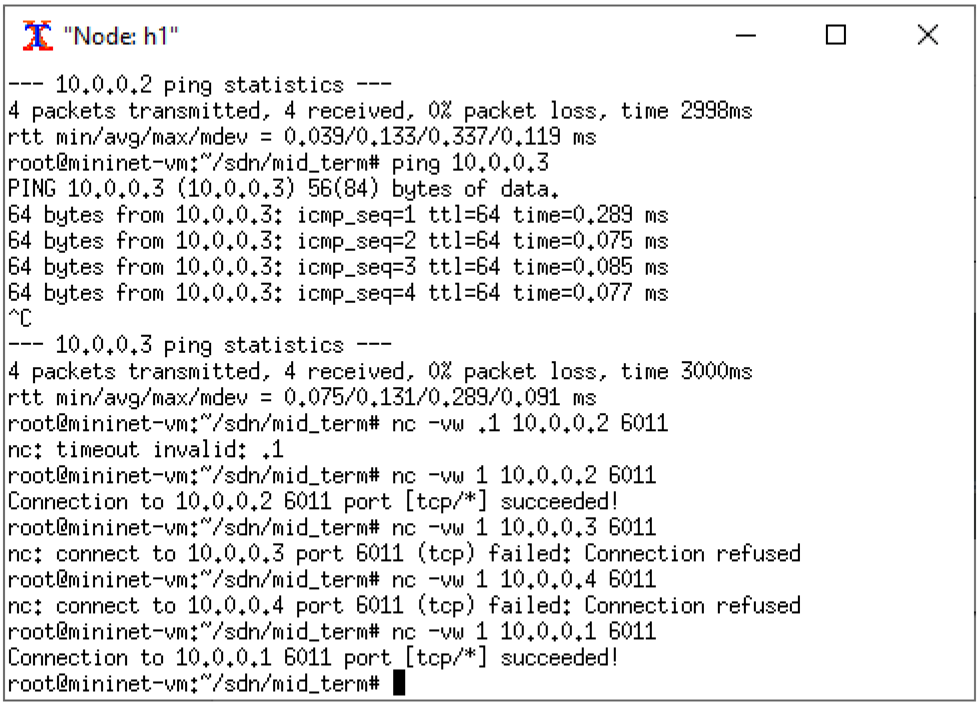
```
h1 nc -vw 1 h2 6011 # If Port 6011 is open in H2
```



```
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> pingall
*** 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)
mininet> h1 nc -vw 1 h2 6011
Connection to 10.0.0.2 6011 port [tcp/*] succeeded!
mininet> 
```

Fig. 2.4: Sending TCP packets with Netcat from H1 to H2.

Above process could be repeated between all hosts as needed. Alternatively, above could be done in an Xterm window if any Xserver is running on host machine.



```
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.039/0.133/0.337/0.119 ms
root@mininet-vm:~/sdn/mid_term# ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.289 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.075 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.085 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.077 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.075/0.131/0.289/0.091 ms
root@mininet-vm:~/sdn/mid_term# nc -vw .1 10.0.0.2 6011
nc: timeout invalid: .1
root@mininet-vm:~/sdn/mid_term# nc -vw 1 10.0.0.2 6011
Connection to 10.0.0.2 6011 port [tcp/*] succeeded!
root@mininet-vm:~/sdn/mid_term# nc -vw 1 10.0.0.3 6011
nc: connect to 10.0.0.3 port 6011 (tcp) failed: Connection refused
root@mininet-vm:~/sdn/mid_term# nc -vw 1 10.0.0.4 6011
nc: connect to 10.0.0.4 port 6011 (tcp) failed: Connection refused
root@mininet-vm:~/sdn/mid_term# nc -vw 1 10.0.0.1 6011
Connection to 10.0.0.1 6011 port [tcp/*] succeeded!
root@mininet-vm:~/sdn/mid_term# 
```

Fig. 2.5: Xterm window while testing interconnections.

Note that **Connection refused** proves forwarding is operational and connection is intact, otherwise **Timed out** would be have been printed.

Exit from Mininet prompt with **Ctrl+C** and clear any remaining instances with the command **sudo mn -c**.

3. Part B - Installing OpenFlow Rules Directly with Controller – Proactive Controller Operation

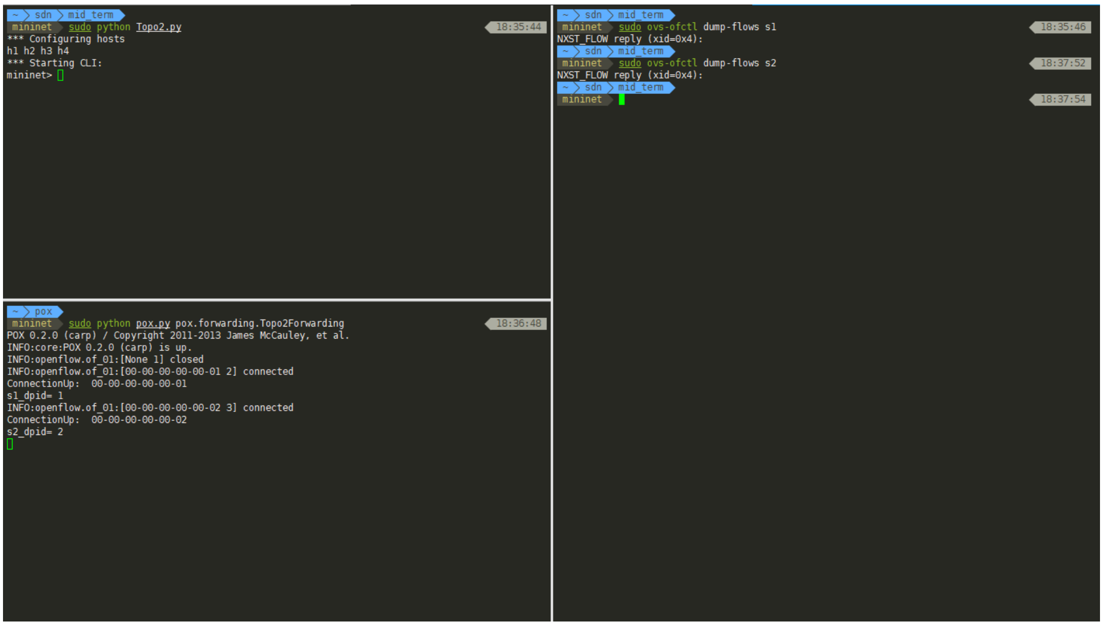
1. Execute controller script and then create the topology.

Remove any already running Mininet instances with **sudo mn -c**.

To easily execute the pox controller script on a newly spawned controller instance, copy **Topo2_Proactive.py** to **~/pox/pox/forwarding/** directory. Once there, execute **pox.py** script with **pox.forwarding.Topo2Forwarding** as an argument. Then create the topology.

Make sure no forwarding rules are initially available (as in Part A, step 1).

```
cp Topo2_Proactive.py ~/pox/pox/forwarding/Topo2Forwarding.py
sudo python ~/pox/pox.py pox.forwarding.Topo2Forwarding
sudo python Topo2.py
```



```
~ > sdn > mid_term
mininet > sudo python Topo2.py                                      18:35:44
*** Configuring hosts
h1 h2 h3 h4
*** Starting CLI:
mininet> 
```

```
~ > pox
mininet > sudo python pox.py pox.forwarding.Topo2Forwarding         18:36:48
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
ConnectionUp:  00-00-00-00-00-01
s1_dpid= 1
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
ConnectionUp:  00-00-00-00-00-02
s2_dpid= 2
```

```
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s1                              18:35:46
NXST_FLOW reply (xid=0x4):
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s2                              18:37:52
NXST_FLOW reply (xid=0x4):
~ > sdn > mid_term
mininet >                                                           18:37:54
```

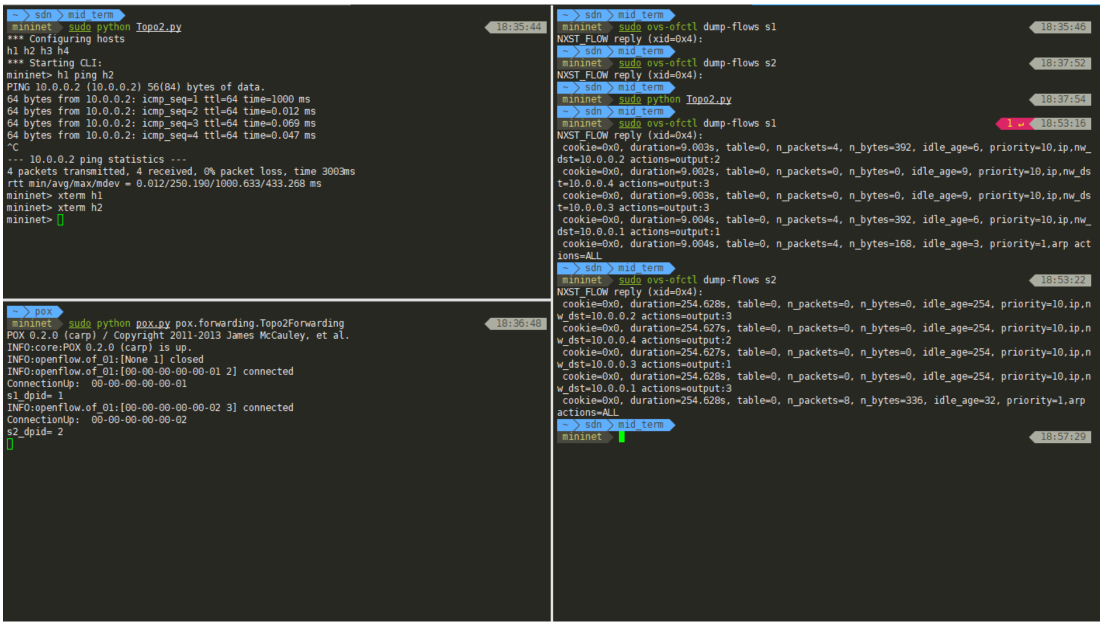
Fig. 3.1: Making sure no rules are available initially in switches.

2. Observe the switches proactively obtaining forwarding rules.

Observe how each switch is proactively obtaining forwarding rules on the very first packet incoming event regardless of interface or protocol. This behavior is expected since the controller is programmed to write all forwarding rules intended to each switch when any switch contacts controller, firing a **ConnectionUp** event, which is when a connection to a switch is started.

Writing rules proactively can reduce latency, however a spike in medium utilization every time (periodically) the rules are written (a timeout value can be introduced for rule-expiration) to switches.

Try sending ICMP and/or TCP packets between hosts (as in Part A, step 3) to see above in operation.



```
~ > sdn > mid_term
mininet > sudo python Topo2.py                                          18:35:44
*** Configuring hosts
h1 h2 h3 h4
*** Starting CLI:
mininet> h1 ping 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=1000 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.012 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.069 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.047 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.012/250.190/1000.633/433.268 ms
mininet> xterm h1
mininet> xterm h2
mininet> 
```

```
~ > pox
mininet > sudo python pox.py pox.forwarding.Topo2Forwarding             18:36:48
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
ConnectionUp:  00-00-00-00-00-01
s1_dpid= 1
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
ConnectionUp:  00-00-00-00-00-02
s2_dpid= 2
```

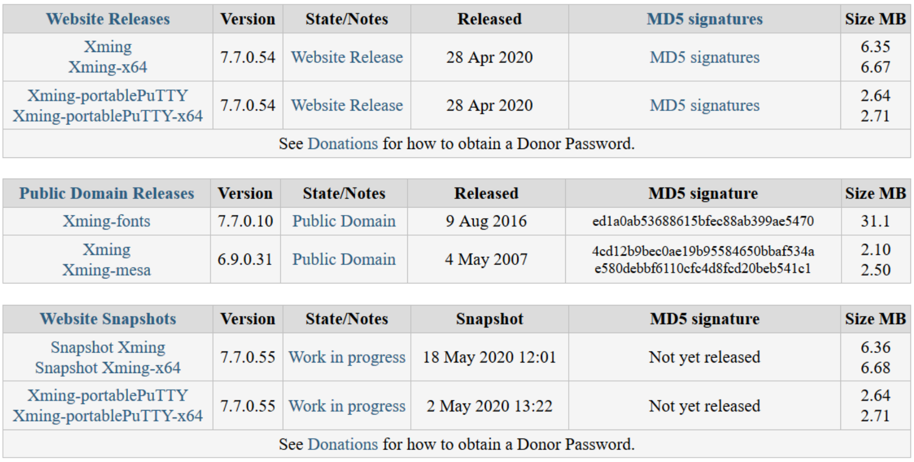
```
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s1                                  18:35:46
NXST_FLOW reply (xid=0x4):
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s2                                  18:37:52
NXST_FLOW reply (xid=0x4):
~ > sdn > mid_term
mininet > sudo python Topo2.py                                          18:37:54
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s1                             1 ↵  18:53:16
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=9.003s, table=0, n_packets=4, n_bytes=392, idle_age=6, priority=10,ip,nw_
dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=9.002s, table=0, n_packets=0, n_bytes=0, idle_age=9, priority=10,ip,nw_ds
t=10.0.0.4 actions=output:3
 cookie=0x0, duration=9.003s, table=0, n_packets=0, n_bytes=0, idle_age=9, priority=10,ip,nw_ds
t=10.0.0.3 actions=output:3
 cookie=0x0, duration=9.004s, table=0, n_packets=4, n_bytes=392, idle_age=6, priority=10,ip,nw_
dst=10.0.0.1 actions=output:1
 cookie=0x0, duration=9.004s, table=0, n_packets=4, n_bytes=168, idle_age=3, priority=1,arp act
ions=ALL
~ > sdn > mid_term
mininet > sudo ovs-ofctl dump-flows s2                                  18:53:22
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=254.628s, table=0, n_packets=0, n_bytes=0, idle_age=254, priority=10,ip,n
w_dst=10.0.0.2 actions=output:3
 cookie=0x0, duration=254.627s, table=0, n_packets=0, n_bytes=0, idle_age=254, priority=10,ip,n
w_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=254.627s, table=0, n_packets=0, n_bytes=0, idle_age=254, priority=10,ip,n
w_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=254.628s, table=0, n_packets=0, n_bytes=0, idle_age=254, priority=10,ip,n
w_dst=10.0.0.1 actions=output:3
 cookie=0x0, duration=254.628s, table=0, n_packets=8, n_bytes=336, idle_age=32, priority=1,arp
actions=ALL
~ > sdn > mid_term
mininet >                                                               18:57:29
```

Fig. 3.2: All rules for all switches are written on first **ConnectionUp** event.

4. Appendix

1. Xming X Window Server Software

An X Window System display server is provided by Xming. Secure Shell (SSH) implementations can be used with Xming in order for securely forwarding X11 sessions from one computer to another.



Website Releases	Version	State/Notes	Released	MD5 signatures	Size MB
Xming Xming-x64	7.7.0.54	Website Release	28 Apr 2020	MD5 signatures	6.35 6.67
Xming-portablePuTTY Xming-portablePuTTY-x64	7.7.0.54	Website Release	28 Apr 2020	MD5 signatures	2.64 2.71
See Donations for how to obtain a Donor Password.					

Public Domain Releases	Version	State/Notes	Released	MD5 signature	Size MB
Xming-fonts	7.7.0.10	Public Domain	9 Aug 2016	ed1a0ab53688615bfec88ab399ae5470	31.1
Xming Xming-mesa	6.9.0.31	Public Domain	4 May 2007	4cd12b9bec0ae19b95584650bbaf534a e580debbf6110cfc4d8fcd20beb541c1	2.10 2.50

Website Snapshots	Version	State/Notes	Snapshot	MD5 signature	Size MB
Snapshot Xming Snapshot Xming-x64	7.7.0.55	Work in progress	18 May 2020 12:01	Not yet released	6.36 6.68
Xming-portablePuTTY Xming-portablePuTTY-x64	7.7.0.55	Work in progress	2 May 2020 13:22	Not yet released	2.64 2.71
See Donations for how to obtain a Donor Password.					

Fig. 4.1: Xming Server package availability.