Introduction To Software-Defined Networking (Winter 2014/2015)

Institute of Computer Science

Georg-August-Universität Göttingen

Lecturer: Dr.David Koll

Author: Hari Raghavendar Rao Bandari

11334055

h.bandari@stud.uni-goettingen.de

Date: April 29, 2015

1 Exercise 8 Mininet and FlowVisor

1.1 Create your own FlowVisor topology

Using the Mininet Python API, create the FlowVisor WAN topology (which you may know from earlier exercises) in a file mini-fw-topo.py:

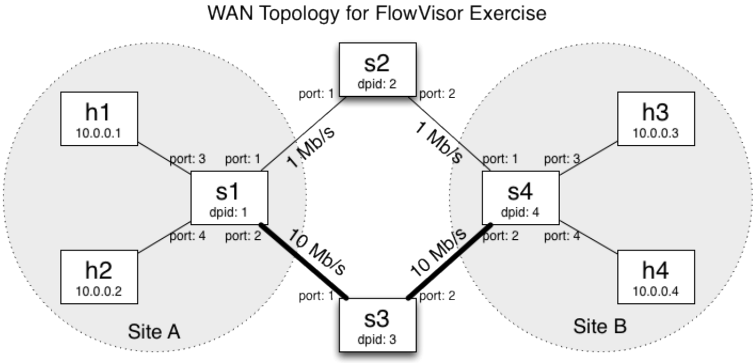


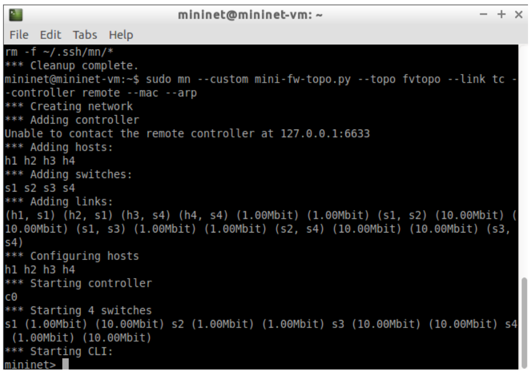
Figure 1: WAN Topology for FlowVisor[1]

To Run Diamond Topology

$ sudo mn –custom mini-fw-topo.py –topo fvtopo –link tc –controller remote –mac –arp

Above will create a network in mininet with the WAN topology. Along with it we have set static ARP entries for the Mininet hosts.

output:



```
rm -f ~/.ssh/mn/*
*** Cleanup complete.
mininet@mininet-vm:~$ sudo mn --custom mini-fw-topo.py --topo fvtopo --link tc -
-controller remote --mac --arp
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s1) (h3, s4) (h4, s4) (1.00Mbit) (1.00Mbit) (s1, s2) (10.00Mbit) (
10.00Mbit) (s1, s3) (1.00Mbit) (1.00Mbit) (s2, s4) (10.00Mbit) (10.00Mbit) (s3,
s4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 (1.00Mbit) (10.00Mbit) s2 (1.00Mbit) (1.00Mbit) s3 (10.00Mbit) (10.00Mbit) s4
 (1.00Mbit) (10.00Mbit)
*** Starting CLI:
mininet>
```

Figure 2: A network in mininet with WAN Topology [2]

1.2 Slice the Network

Now, slice your network so that it supports the following slices: In short, this slice arrangement allows traffic to be sent from h1 to h3 and h2 to h4 (and viceversa) only, even though the topology itself (i.e., without slicing) would allow sending traffic between arbitrary pairs of hosts. For slicing a network with FlowVisor in general, you need to take the following steps. First, make sure you set up the flowvisor package correctly. To configure FlowVisor, use:

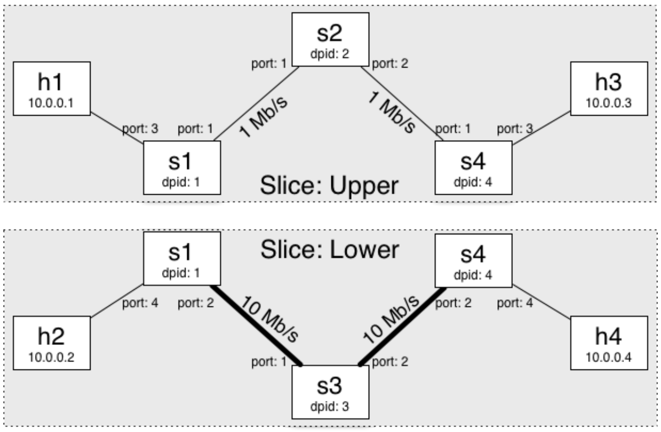


Figure 3: Simple Topology-based Slicing [3]

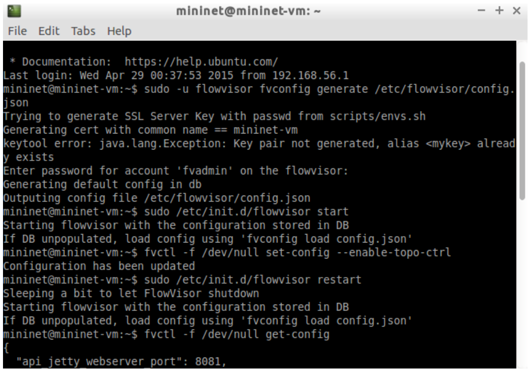
$ sudo -u flowvisor fvconfig generate /etc/flowvisor/config.json

Then, start flowvisor in a new terminal:
$ sudo /etc/init.d/flowvisor start

We have to enable topology control for flowvisor as well: $ fvctl -f /dev/null set-config –enable-topo-ctrl

Similar to ovs-ofctl, fvctl is the control channel that we will use for flowvisor. The option f refers to the flowvisor password file. Since we have set the password to be empty, we can hand it /dev/null. This part will be present in all the following fvctl calls.

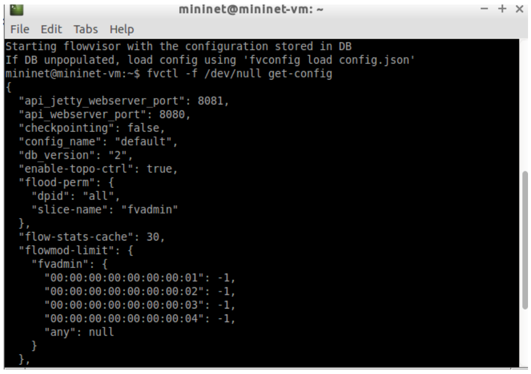
Restart flowvisor:
$ sudo /etc/init.d/flowvisor start



```
 * Documentation:  https://help.ubuntu.com/
Last login: Wed Apr 29 00:37:53 2015 from 192.168.56.1
mininet@mininet-vm:~$ sudo -u flowvisor fvconfig generate /etc/flowvisor/config.
json
Trying to generate SSL Server Key with passwd from scripts/envs.sh
Generating cert with common name == mininet-vm
keytool error: java.lang.Exception: Key pair not generated, alias <mykey> alread
y exists
Enter password for account 'fvadmin' on the flowvisor:
Generating default config in db
Outputing config file /etc/flowvisor/config.json
mininet@mininet-vm:~$ sudo /etc/init.d/flowvisor start
Starting flowvisor with the configuration stored in DB
If DB unpopulated, load config using 'fvconfig load config.json'
mininet@mininet-vm:~$ fvctl -f /dev/null set-config --enable-topo-ctrl
Configuration has been updated
mininet@mininet-vm:~$ sudo /etc/init.d/flowvisor restart
Sleeping a bit to let FlowVisor shutdown
Starting flowvisor with the configuration stored in DB
If DB unpopulated, load config using 'fvconfig load config.json'
mininet@mininet-vm:~$ fvctl -f /dev/null get-config
{
  "api_jetty_webserver_port": 8081,
```

Figure 4: Flowvisor Configuration, Start and Restart [4]

Now, have a look at the FlowVisor configuration:
$ fvctl -f /dev/null get-config



```
Starting flowvisor with the configuration stored in DB
If DB unpopulated, load config using 'fvconfig load config.json'
mininet@mininet-vm:~$ fvctl -f /dev/null get-config
{
  "api_jetty_webserver_port": 8081,
  "api_webserver_port": 8080,
  "checkpointing": false,
  "config_name": "default",
  "db_version": "2",
  "enable-topo-ctrl": true,
  "flood-perm": {
    "dpid": "all",
    "slice-name": "fvadmin"
  },
  "flow-stats-cache": 30,
  "flowmod-limit": {
    "fvadmin": {
      "00:00:00:00:00:00:00:01": -1,
      "00:00:00:00:00:00:00:02": -1,
      "00:00:00:00:00:00:00:03": -1,
      "00:00:00:00:00:00:00:04": -1,
      "any": null
    }
  },
```

Figure 5: Flowvisor Configuration in JSON format [5]

This also has the purpose of making sure that flowvisor is actually running and that all the switches have indeed a connection to flowvisor. The configuration should show this.

a. Which part of the configuration file tells you that all four switches have connected to flowvisor?

Solution:

```
"flowmod-limit": {
    "fvadmin": {
      "00:00:00:00:00:00:00:01": -1,
      "00:00:00:00:00:00:00:02": -1,
      "00:00:00:00:00:00:00:03": -1,
      "00:00:00:00:00:00:00:04": -1,
      "any": null
    }
  }
```

In the lecture, you also got a brief overview over the major flowvisor commands. Now, make use of these commands to
b. List the currently existing slices.

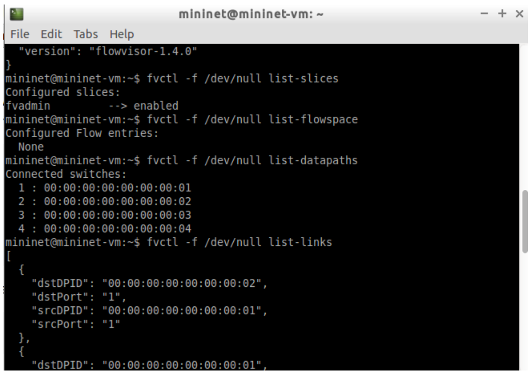
```
$ fvctl -f /dev/null list-slices
        Configured slices:\newline
             fvadmin  --> enabled
```

c. List the currently existing flowspaces.

```
$  fvctl -f /dev/null list-flowspace
Configured Flow entries:
  None
```

d. List the currently connected switches.

```
$ fvctl -f /dev/null list-datapaths
Connected switches:
  1 : 00:00:00:00:00:00:00:01
  2 : 00:00:00:00:00:00:00:02
  3 : 00:00:00:00:00:00:00:03
  4 : 00:00:00:00:00:00:00:04
```

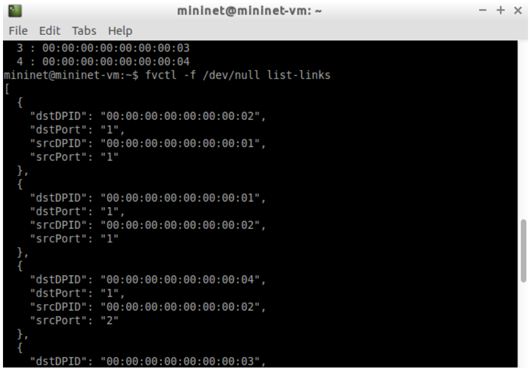


```
  "version": "flowvisor-1.4.0"
}
mininet@mininet-vm:~$ fvctl -f /dev/null list-slices
Configured slices:
fvadmin         --> enabled
mininet@mininet-vm:~$ fvctl -f /dev/null list-flowspace
Configured Flow entries:
  None
mininet@mininet-vm:~$ fvctl -f /dev/null list-datapaths
Connected switches:
  1 : 00:00:00:00:00:00:00:01
  2 : 00:00:00:00:00:00:00:02
  3 : 00:00:00:00:00:00:00:03
  4 : 00:00:00:00:00:00:00:04
mininet@mininet-vm:~$ fvctl -f /dev/null list-links
[
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:01",
```

Figure 6: Screenshot of List of slices/flowspace/datapaths [6]

e. List the currently existing links.

```
$ fvctl -f /dev/null list-links
[
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "1"
  },
```



```
  3 : 00:00:00:00:00:00:00:03
  4 : 00:00:00:00:00:00:00:04
mininet@mininet-vm:~$ fvctl -f /dev/null list-links
[
  {
    "dstDPID": "00:00:00:00:00:00:00:02",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:01",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:01",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:02",
    "srcPort": "1"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:04",
    "dstPort": "1",
    "srcDPID": "00:00:00:00:00:00:00:02",
    "srcPort": "2"
  },
  {
    "dstDPID": "00:00:00:00:00:00:00:03",
```

Figure 7: Screenshot of List of links [7]

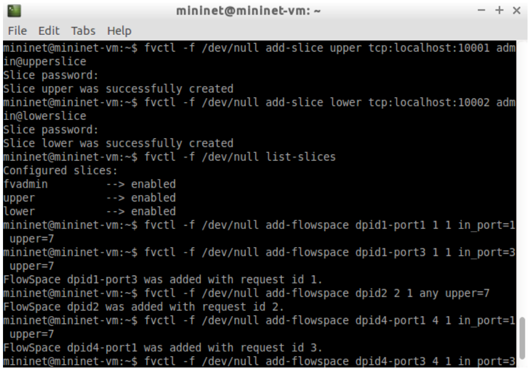
Afterwards, proceed with slicing your topology: f. Create the appropriate slices.

Create a slice named upper connecting to a controller listening on tcp:localhost:10001 by running the following command:

```
$ fvctl -f /dev/null add-slice upper tcp:localhost:10001 admin@upperslice
```

Create a slice named lower connecting to a controller listening on tcp:localhost:10002.

```
$ fvctl -f /dev/null add-slice lower tcp:localhost:10002 admin@lowerslice
$ fvctl -f /dev/null list-slices
```



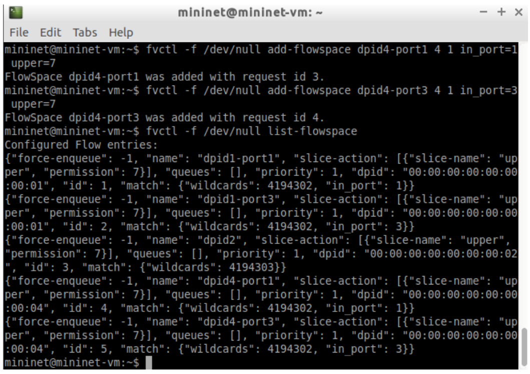
```
mininet@mininet-vm:~$ fvctl -f /dev/null add-slice upper tcp:localhost:10001 adm
in@upperslice
Slice password:
Slice upper was successfully created
mininet@mininet-vm:~$ fvctl -f /dev/null add-slice lower tcp:localhost:10002 adm
in@lowerslice
Slice password:
Slice lower was successfully created
mininet@mininet-vm:~$ fvctl -f /dev/null list-slices
Configured slices:
fvadmin         --> enabled
upper           --> enabled
lower           --> enabled
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid1-port1 1 1 in_port=1
 upper=7
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid1-port3 1 1 in_port=3
 upper=7
FlowSpace dpid1-port3 was added with request id 1.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid2 2 1 any upper=7
FlowSpace dpid2 was added with request id 2.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid4-port1 4 1 in_port=1
 upper=7
FlowSpace dpid4-port1 was added with request id 3.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid4-port3 4 1 in_port=3
```

Figure 8: Screenshot After Slicing–List of slices [8]

g. Create the appropriate flowspaces. solution: Creating flowspaces for upper Slice:

```
$ fvctl -f /dev/null add-flowspace dpid1-port1 1 1 in_port=1 upper=7

$ fvctl -f /dev/null add-flowspace dpid1-port3 1 1 in_port=3 upper=7

$ fvctl -f /dev/null add-flowspace dpid2 2 1 any upper=7

$ fvctl -f /dev/null add-flowspace dpid4-port1 4 1 in_port=1 upper=7

$ fvctl -f /dev/null add-flowspace dpid4-port3 4 1 in_port=3 upper=7
```

```
Creating flowspaces for lower slice:

$ fvctl -f /dev/null add-flowspace dpid1-port2 1 1 in_port=2 lower=7

$ fvctl -f /dev/null add-flowspace dpid1-port4 1 1 in_port=4 lower=7

$ fvctl -f /dev/null add-flowspace dpid3 3 1 any lower=7

$ fvctl -f /dev/null add-flowspace dpid4-port2 4 1 in_port=2 lower=7

$ fvctl -f /dev/null add-flowspace dpid4-port4 4 1 in_port=4 lower=7
```



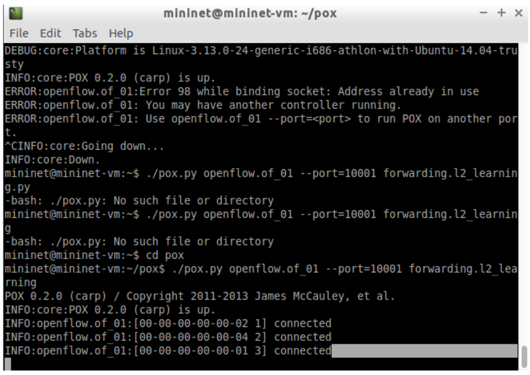
```
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid4-port1 4 1 in_port=1
 upper=7
FlowSpace dpid4-port1 was added with request id 3.
mininet@mininet-vm:~$ fvctl -f /dev/null add-flowspace dpid4-port3 4 1 in_port=3
 upper=7
FlowSpace dpid4-port3 was added with request id 4.
mininet@mininet-vm:~$ fvctl -f /dev/null list-flowspace
Configured Flow entries:
{"force-enqueue": -1, "name": "dpid1-port1", "slice-action": [{"slice-name": "up
per", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00
:00:01", "id": 1, "match": {"wildcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid1-port3", "slice-action": [{"slice-name": "up
per", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00
:00:01", "id": 2, "match": {"wildcards": 4194302, "in_port": 3}}
{"force-enqueue": -1, "name": "dpid2", "slice-action": [{"slice-name": "upper",
"permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00:00:02
", "id": 3, "match": {"wildcards": 4194303}}
{"force-enqueue": -1, "name": "dpid4-port1", "slice-action": [{"slice-name": "up
per", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00
:00:04", "id": 4, "match": {"wildcards": 4194302, "in_port": 1}}
{"force-enqueue": -1, "name": "dpid4-port3", "slice-action": [{"slice-name": "up
per", "permission": 7}], "queues": [], "priority": 1, "dpid": "00:00:00:00:00:00
:00:04", "id": 5, "match": {"wildcards": 4194302, "in_port": 3}}
mininet@mininet-vm:~$
```

Figure 9: Screenshot After Slicing–flowspaces [9]

h. (10P) Connect an instance of the POX controller to each of your slices. solution:

```
$ ./pox.py openflow.of_01 --port=10001 forwarding.l2_learning
```

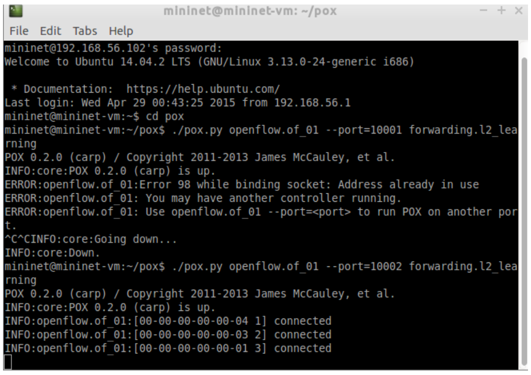
It is connected to S1, S2, S4



```
DEBUG:core:Platform is Linux-3.13.0-24-generic-i686-athlon-with-Ubuntu-14.04-tru
sty
INFO:core:POX 0.2.0 (carp) is up.
ERROR:openflow.of_01:Error 98 while binding socket: Address already in use
ERROR:openflow.of_01: You may have another controller running.
ERROR:openflow.of_01: Use openflow.of_01 --port=<port> to run POX on another por
t.
^CINFO:core:Going down...
INFO:core:Down.
mininet@mininet-vm:~$ ./pox.py openflow.of_01 --port=10001 forwarding.l2_learnin
g.py
-bash: ./pox.py: No such file or directory
mininet@mininet-vm:~$ ./pox.py openflow.of_01 --port=10001 forwarding.l2_learnin
g
-bash: ./pox.py: No such file or directory
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ ./pox.py openflow.of_01 --port=10001 forwarding.l2_lea
rning
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:[00-00-00-00-00-02 1] connected
INFO:openflow.of_01:[00-00-00-00-00-04 2] connected
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
```

Figure 10: Screenshot of POX Controller [10]

It is connected to S1, S3, S4

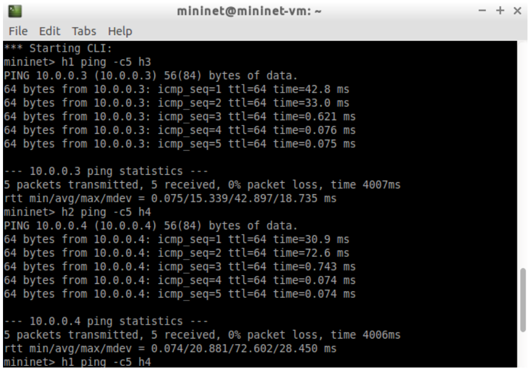


```
mininet@192.168.56.102's password:
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/
Last login: Wed Apr 29 00:43:25 2015 from 192.168.56.1
mininet@mininet-vm:~$ cd pox
mininet@mininet-vm:~/pox$ ./pox.py openflow.of_01 --port=10001 forwarding.l2_lea
rning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
ERROR:openflow.of_01:Error 98 while binding socket: Address already in use
ERROR:openflow.of_01: You may have another controller running.
ERROR:openflow.of_01: Use openflow.of_01 --port=<port> to run POX on another por
t.
^C^CINFO:core:Going down...
INFO:core:Down.
mininet@mininet-vm:~/pox$ ./pox.py openflow.of_01 --port=10002 forwarding.l2_lea
rning
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:[00-00-00-00-00-04 1] connected
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
```

Figure 11: Screenshot of POX Controller [11]

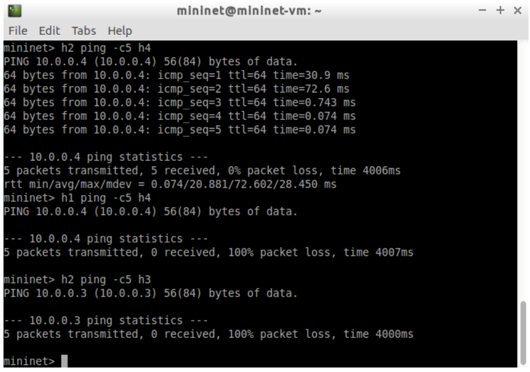
i. (10P) In Mininet, verify that your slicing works properly, i.e., h1 can reach h3 but not h2 and h4, and h2 can reach h4, but not h1 and h3.

Solution:



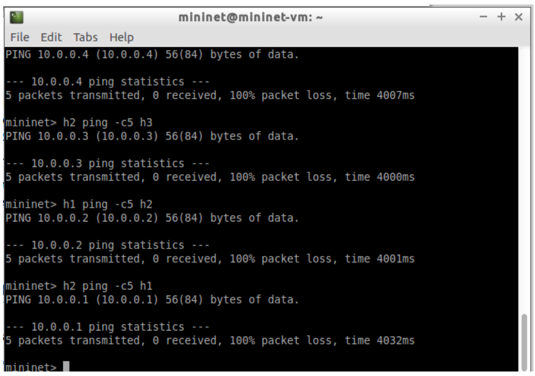
```
*** Starting CLI:
mininet> h1 ping -c5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=42.8 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=33.0 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.621 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.076 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.075 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 0.075/15.339/42.897/18.735 ms
mininet> h2 ping -c5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=30.9 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=72.6 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.743 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.074 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.074 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.074/20.881/72.602/28.450 ms
mininet> h1 ping -c5 h4
```

Figure 12: Screenshot of Mininet slicinig work properly [12]



```
mininet> h2 ping -c5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=30.9 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=72.6 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.743 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.074 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.074 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.074/20.881/72.602/28.450 ms
mininet> h1 ping -c5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4007ms

mininet> h2 ping -c5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4000ms

mininet> 
```

Figure 13: Screenshot of Mininet slicinig work properly[13]



```
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4007ms

mininet> h2 ping -c5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4000ms

mininet> h1 ping -c5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4001ms

mininet> h2 ping -c5 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

--- 10.0.0.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4032ms

mininet> 
```

Figure 14: Screenshot of Mininet slicinig work properly [14]