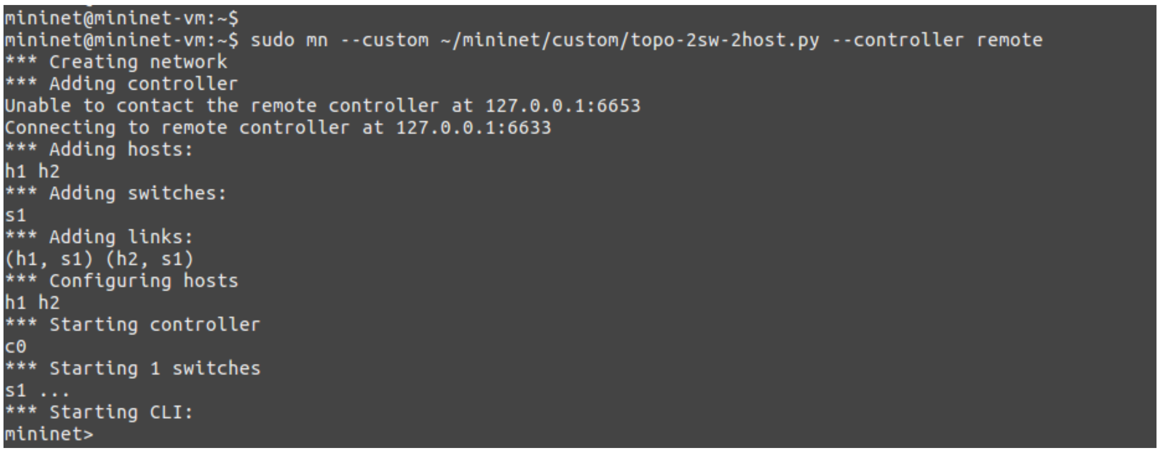
CS 204, Spring 2017
Abhishek Kumar SRIVASTAVA
Student ID: 861307778

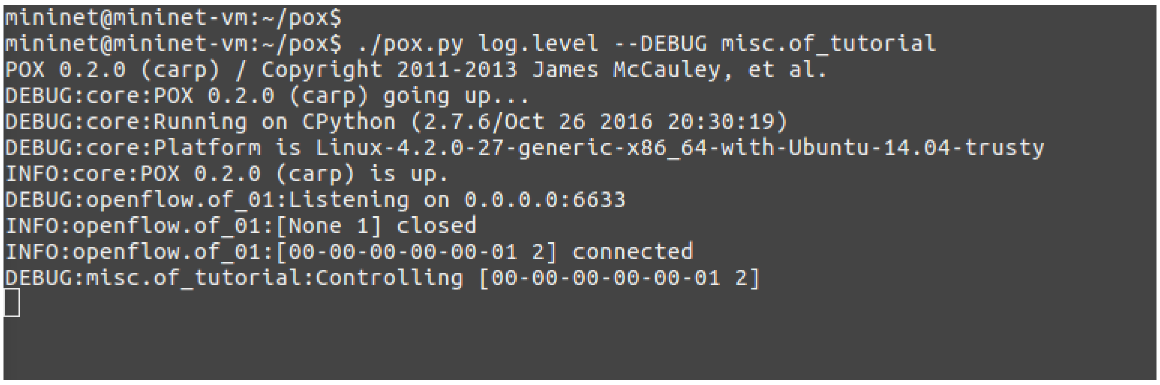
SDN Assignment Report
June 13, 2017

Solution . Setup Screenshots:



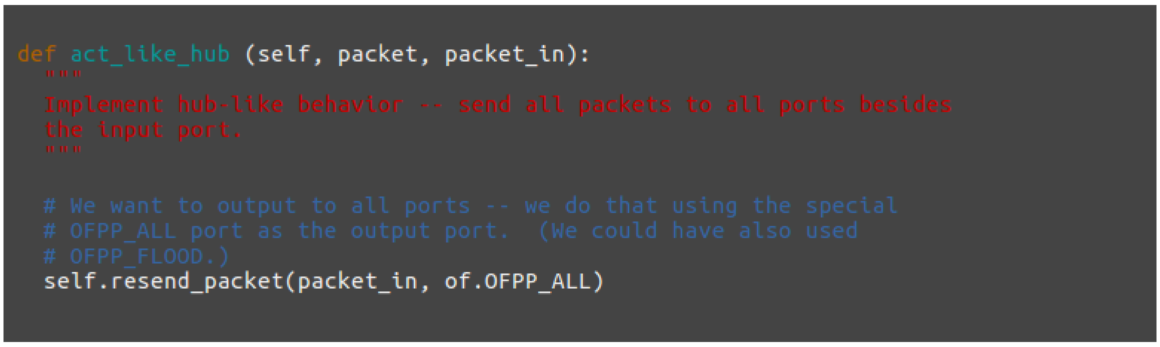
```
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Figure 1: Setting up a test topology that connects to the controller and switches.



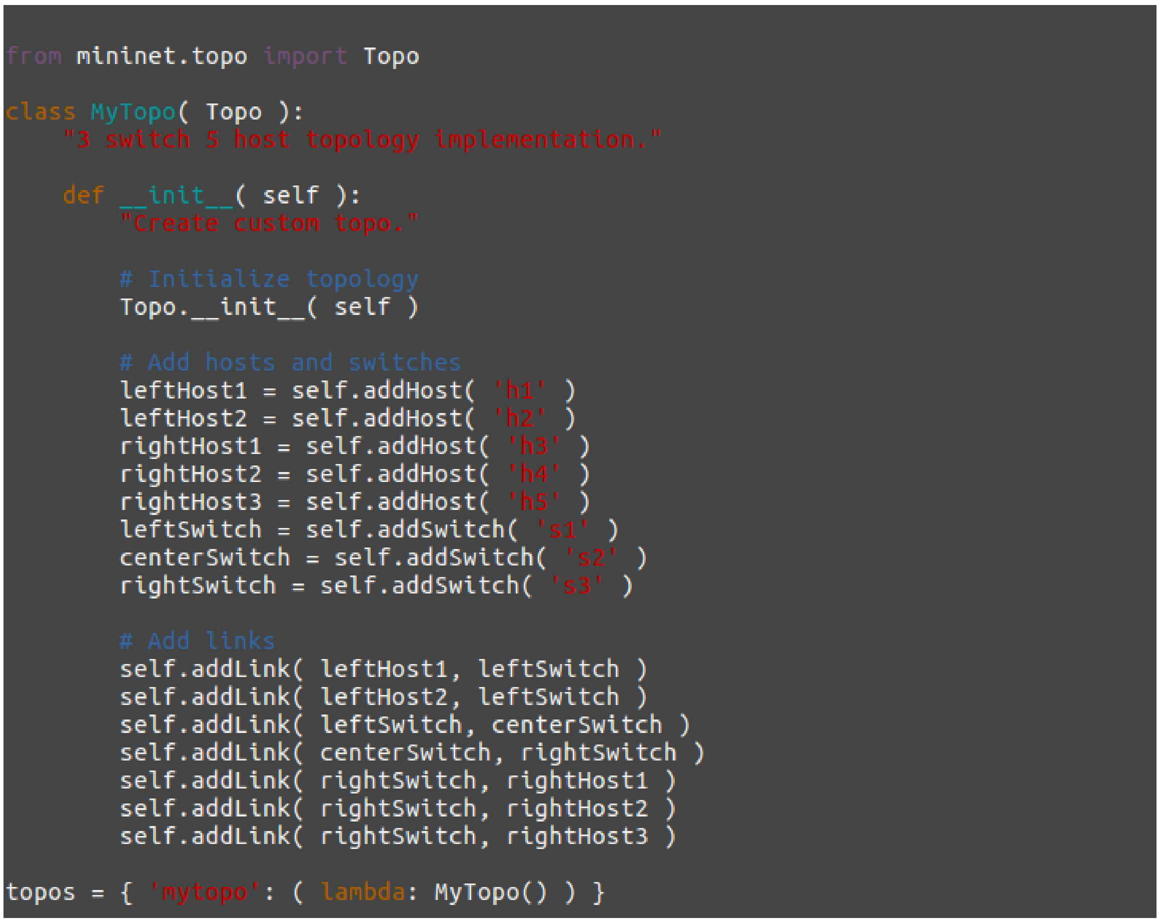
```
mininet@mininet-vm:~/pox$
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG misc.of_tutorial
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Oct 26 2016 20:30:19)
DEBUG:core:Platform is Linux-4.2.0-27-generic-x86_64-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 2]
```

Figure 2: Running the controller which connects to switches as seen in the output.



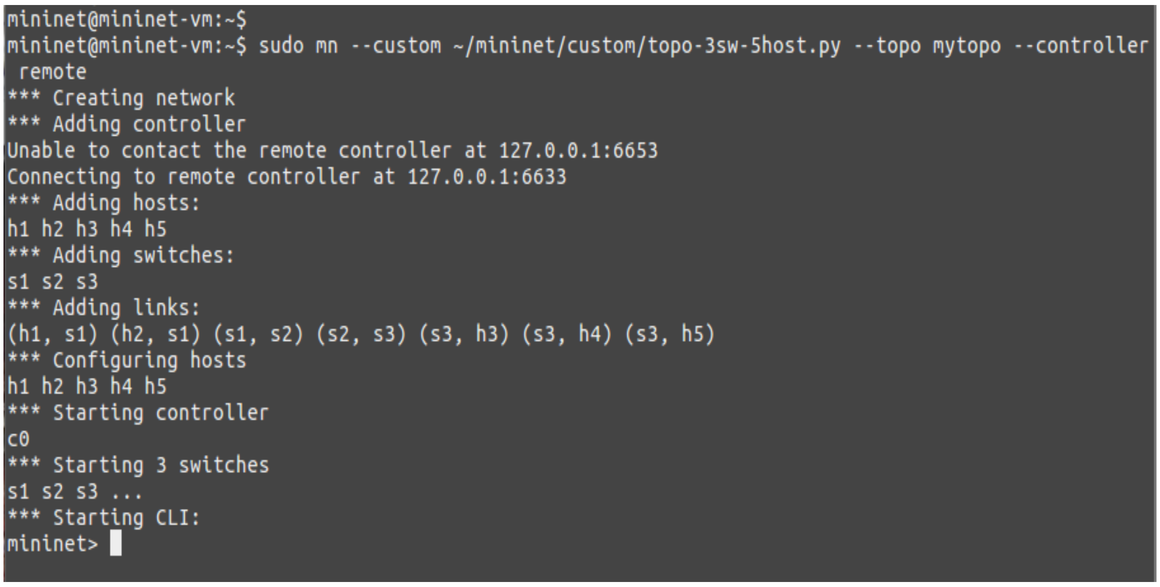
```
def act_like_hub (self, packet, packet_in):
  """
  Implement hub-like behavior -- send all packets to all ports besides
  the input port.
  """

  # We want to output to all ports -- we do that using the special
  # OFPP_ALL port as the output port.  (We could have also used
  # OFPP_FLOOD.)
  self.resend_packet(packet_in, of.OFPP_ALL)
```

Figure 3: “of_tutorial” act as dumb switch(hub) and floods packets to all port except the input port.



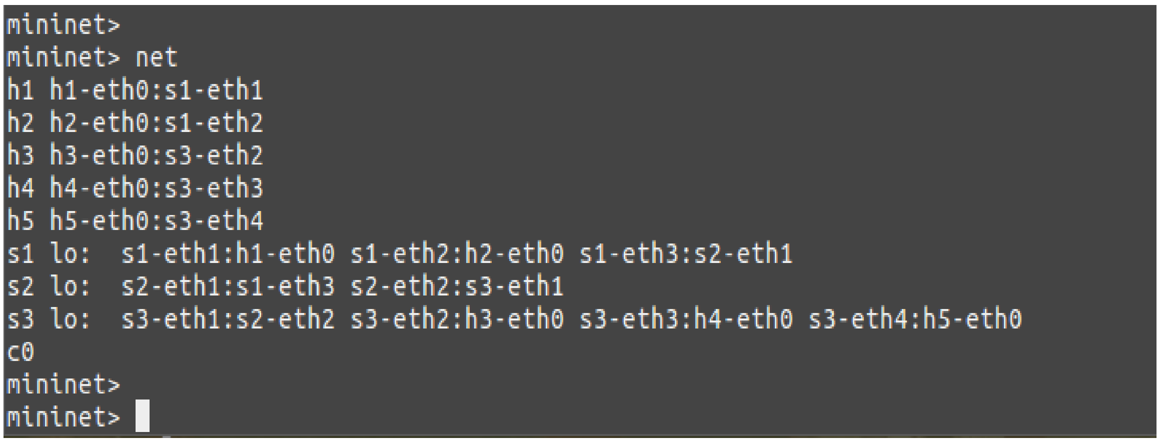
```
from mininet.topo import Topo

class MyTopo( Topo ):
    "3 switch 5 host topology implementation."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost1 = self.addHost( 'h1' )
        leftHost2 = self.addHost( 'h2' )
        rightHost1 = self.addHost( 'h3' )
        rightHost2 = self.addHost( 'h4' )
        rightHost3 = self.addHost( 'h5' )
        leftSwitch = self.addSwitch( 's1' )
        centerSwitch = self.addSwitch( 's2' )
        rightSwitch = self.addSwitch( 's3' )

        # Add links
        self.addLink( leftHost1, leftSwitch )
        self.addLink( leftHost2, leftSwitch )
        self.addLink( leftSwitch, centerSwitch )
        self.addLink( centerSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost1 )
        self.addLink( rightSwitch, rightHost2 )
        self.addLink( rightSwitch, rightHost3 )

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

Figure 4: Code to create topology of 5hosts & 3 switches and add links as specified in the problem: "topo-3sw-5host".



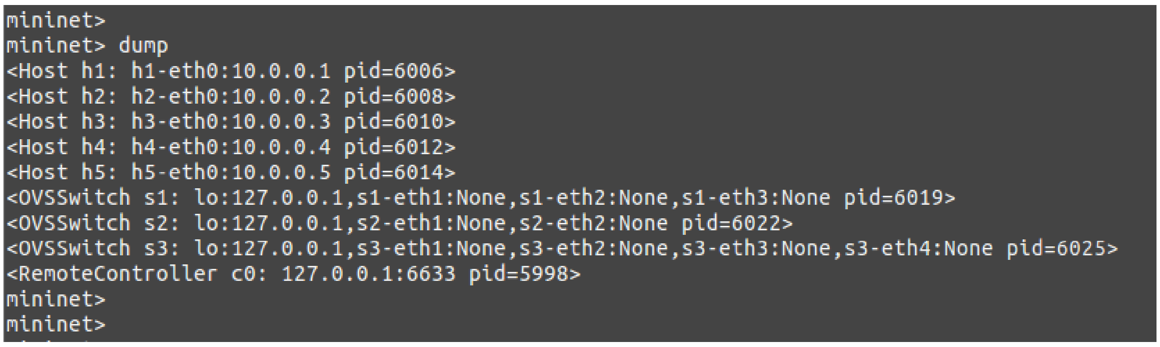
```
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo-3sw-5host.py --topo mytopo --controller
 remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s1) (s1, s2) (s2, s3) (s3, h3) (s3, h4) (s3, h5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Figure 5: Running "topo-3sw-5host" network topology.



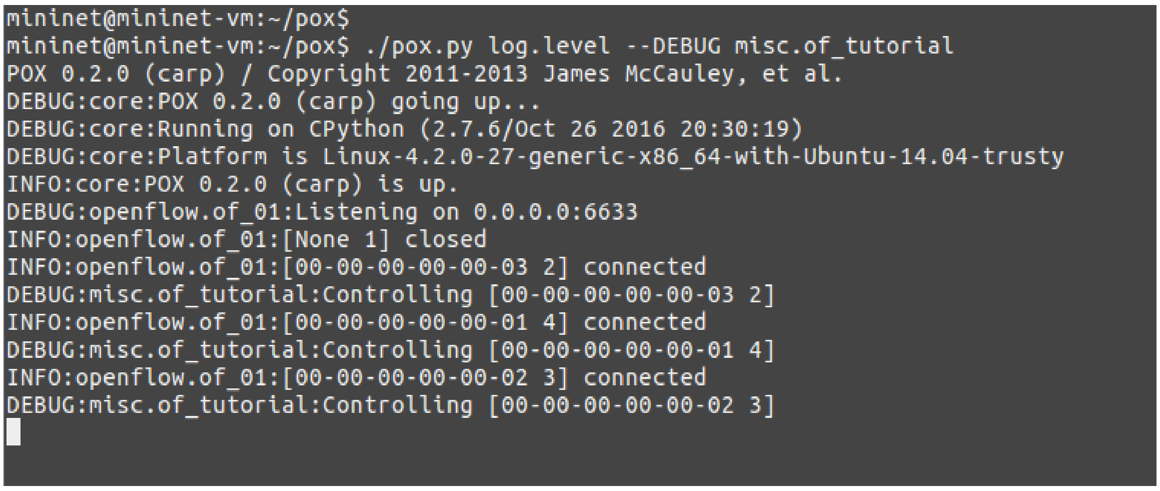
```
mininet>
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s3-eth2
h4 h4-eth0:s3-eth3
h5 h5-eth0:s3-eth4
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:s2-eth1
s2 lo:  s2-eth1:s1-eth3 s2-eth2:s3-eth1
s3 lo:  s3-eth1:s2-eth2 s3-eth2:h3-eth0 s3-eth3:h4-eth0 s3-eth4:h5-eth0
c0
mininet>
mininet>
```

Figure 6: Showing connections for the "topo-3sw-5host" topology.



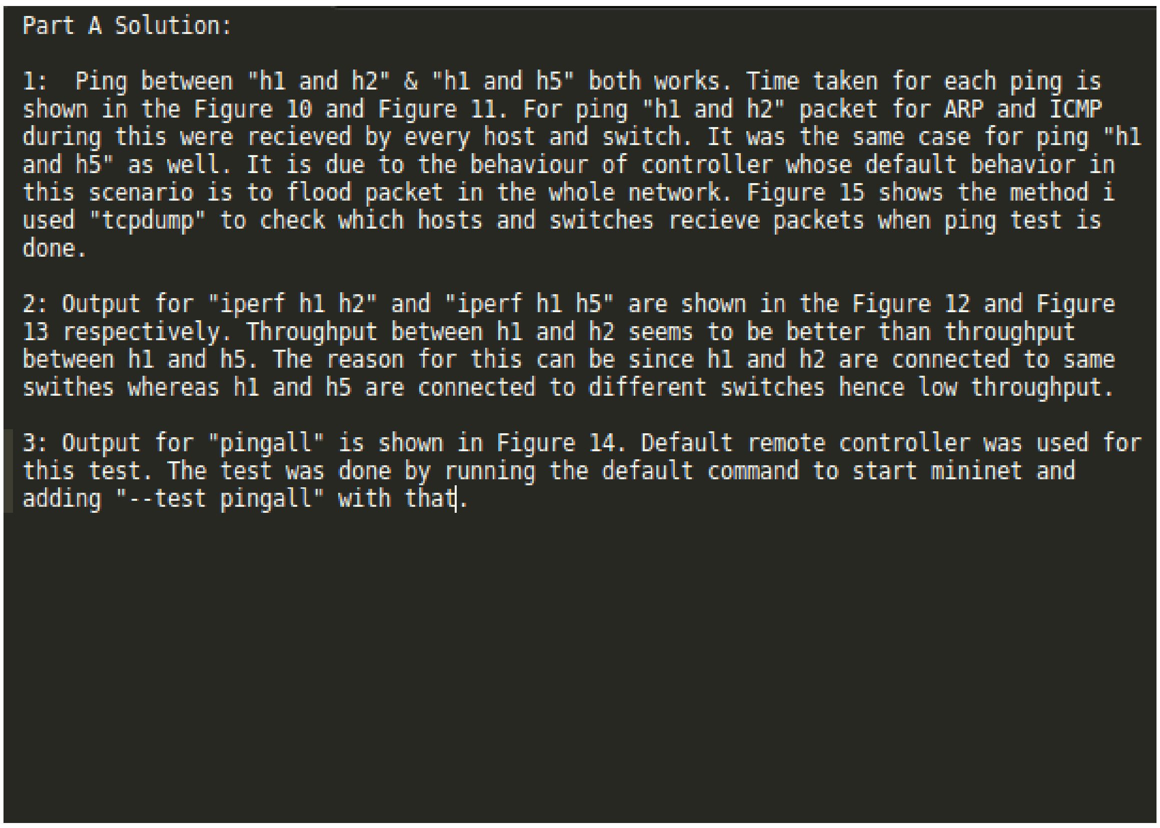
```
mininet>
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=6006>
<Host h2: h2-eth0:10.0.0.2 pid=6008>
<Host h3: h3-eth0:10.0.0.3 pid=6010>
<Host h4: h4-eth0:10.0.0.4 pid=6012>
<Host h5: h5-eth0:10.0.0.5 pid=6014>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=6019>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=6022>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None pid=6025>
<RemoteController c0: 127.0.0.1:6633 pid=5998>
mininet>
mininet>
```

Figure 7: Ip Addresses of switches and hosts.



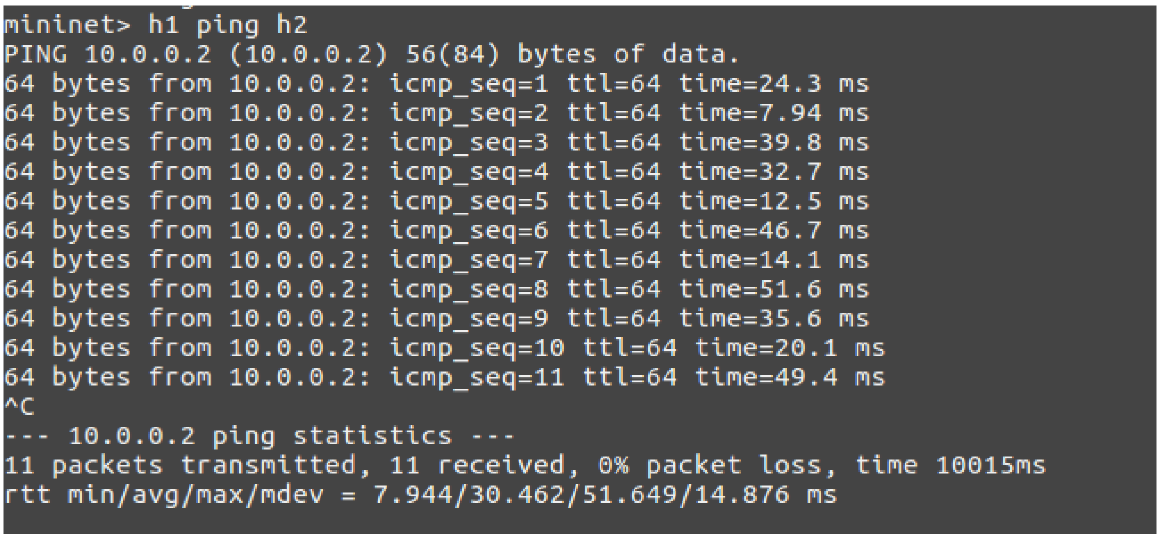
```
mininet@mininet-vm:~/pox$
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG misc.of_tutorial
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Oct 26 2016 20:30:19)
DEBUG:core:Platform is Linux-4.2.0-27-generic-x86_64-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-03 2]
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 4]
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 3]
```

Figure 8: Output of running remote controller "of_tutorial". All 3 switches able to connect to the controller.



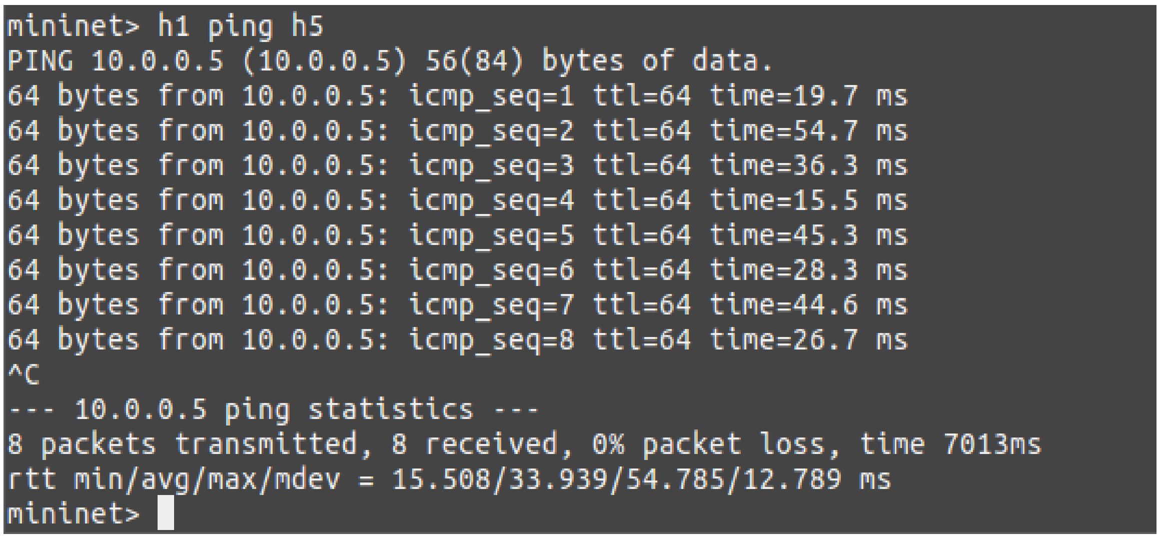
```
Part A Solution:

1:  Ping between "h1 and h2" & "h1 and h5" both works. Time taken for each ping is
shown in the Figure 10 and Figure 11. For ping "h1 and h2" packet for ARP and ICMP
during this were recieved by every host and switch. It was the same case for ping "h1
and h5" as well. It is due to the behaviour of controller whose default behavior in
this scenario is to flood packet in the whole network. Figure 15 shows the method i
used "tcpdump" to check which hosts and switches recieve packets when ping test is
done.

2: Output for "iperf h1 h2" and "iperf h1 h5" are shown in the Figure 12 and Figure
13 respectively. Throughput between h1 and h2 seems to be better than throughput
between h1 and h5. The reason for this can be since h1 and h2 are connected to same
swithes whereas h1 and h5 are connected to different switches hence low throughput.

3: Output for "pingall" is shown in Figure 14. Default remote controller was used for
this test. The test was done by running the default command to start mininet and
adding "--test pingall" with that.
```

Figure 9: Solution A.



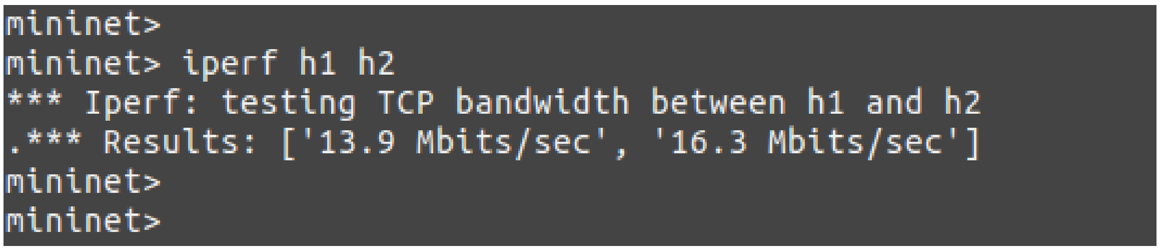
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=24.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=7.94 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=39.8 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=32.7 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=12.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=46.7 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=14.1 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=51.6 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=35.6 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=20.1 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=49.4 ms
^C
--- 10.0.0.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10015ms
rtt min/avg/max/mdev = 7.944/30.462/51.649/14.876 ms
```

Figure 10: Output of "h1 ping h2" command for default behavior of controller.



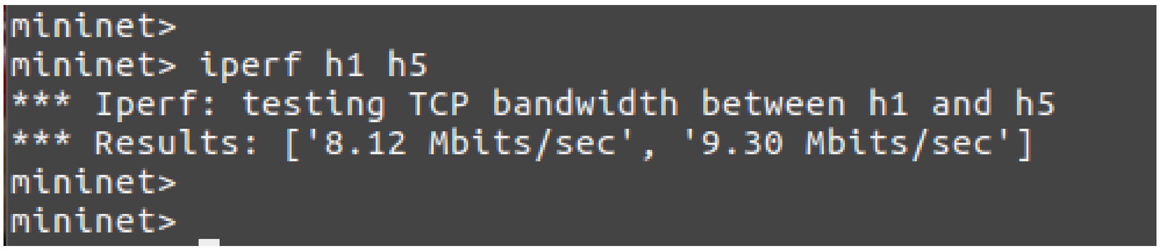
```
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=19.7 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=54.7 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=36.3 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=15.5 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=45.3 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=28.3 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=44.6 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=26.7 ms
^C
--- 10.0.0.5 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7013ms
rtt min/avg/max/mdev = 15.508/33.939/54.785/12.789 ms
mininet>
```

Figure 11: Output of “h1 ping h5” command for default behavior of controller.



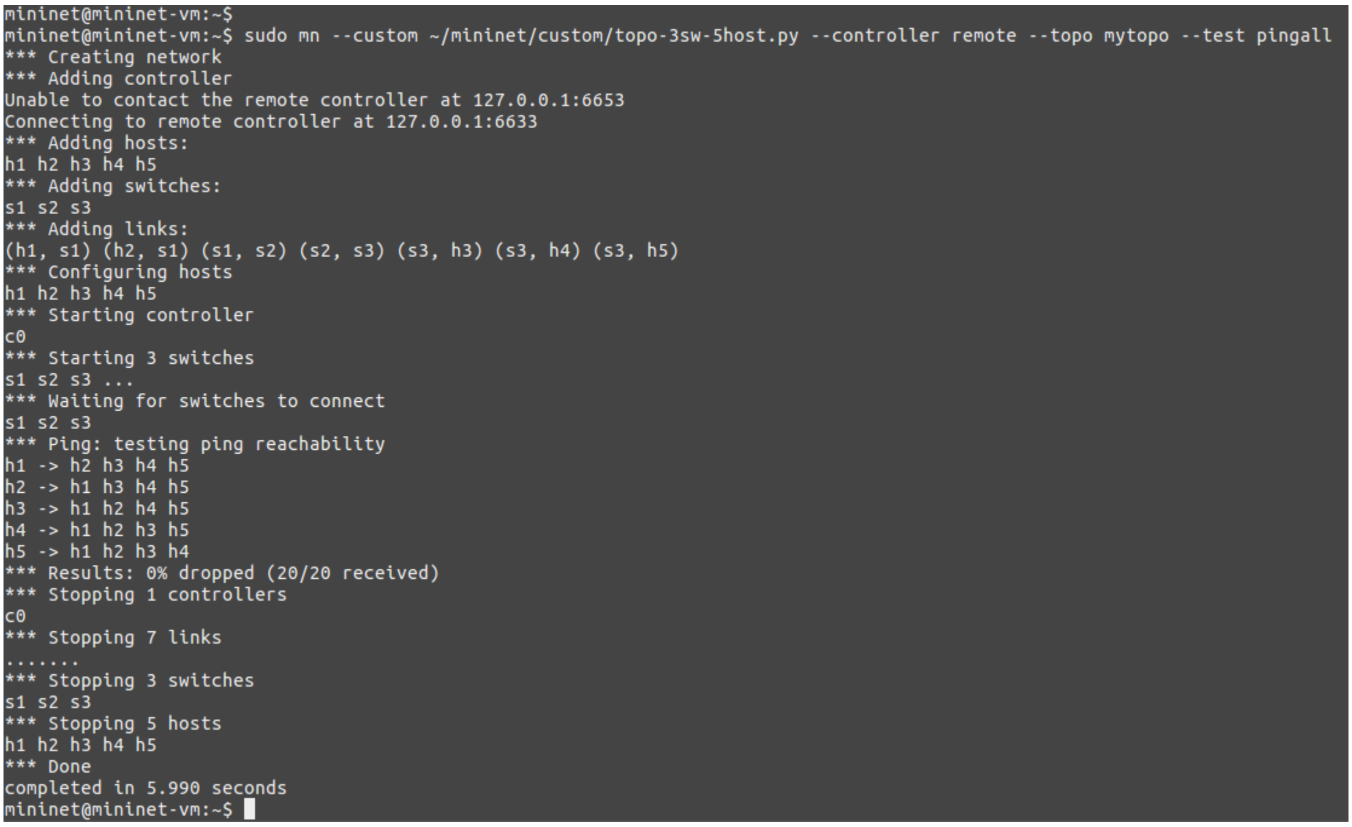
```
mininet>
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['13.9 Mbits/sec', '16.3 Mbits/sec']
mininet>
mininet>
```

Figure 12: Output of “iperf h1 h2” command for default behavior of controller.



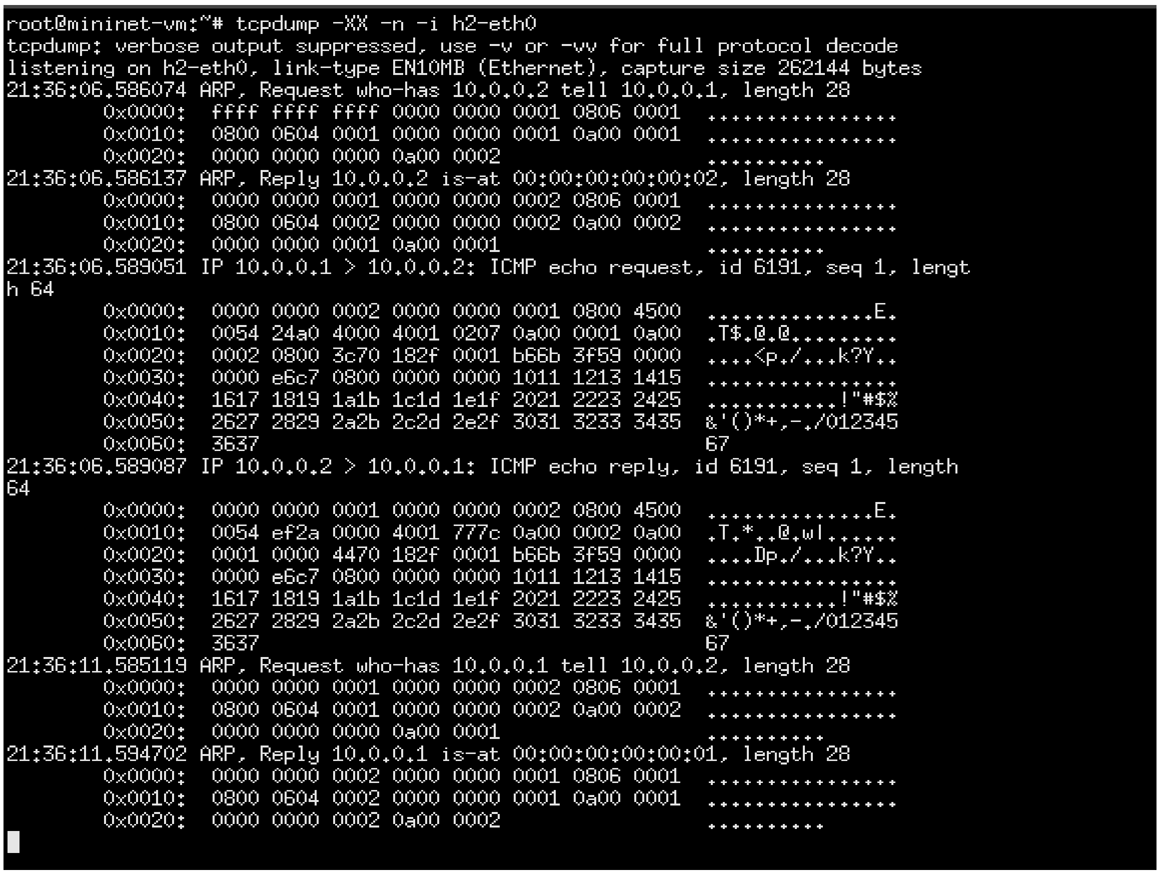
```
mininet>
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['8.12 Mbits/sec', '9.30 Mbits/sec']
mininet>
mininet>
```

Figure 13: Output of “iperf h1 h5” command for default behavior of controller.



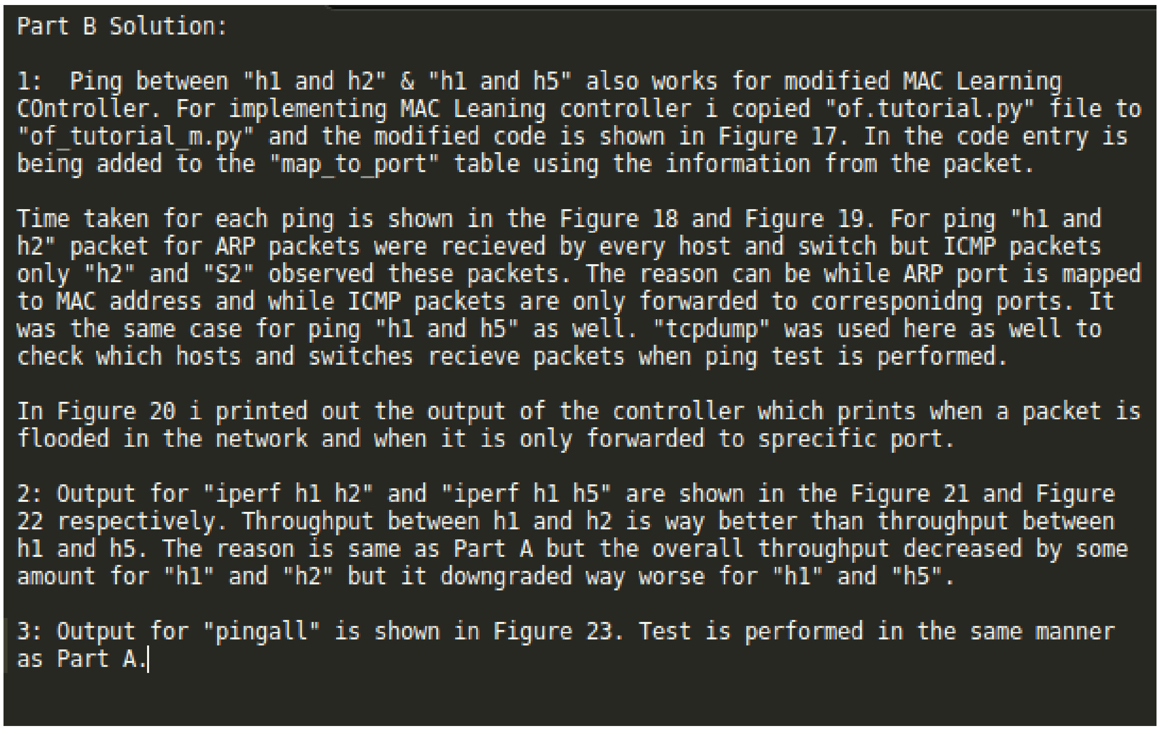
```
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo-3sw-5host.py --controller remote --topo mytopo --test pingall
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s1) (s1, s2) (s2, s3) (s3, h3) (s3, h4) (s3, h5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Waiting for switches to connect
s1 s2 s3
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
completed in 5.990 seconds
mininet@mininet-vm:~$
```

Figure 14: "Pingall" test for default behavior of controller.



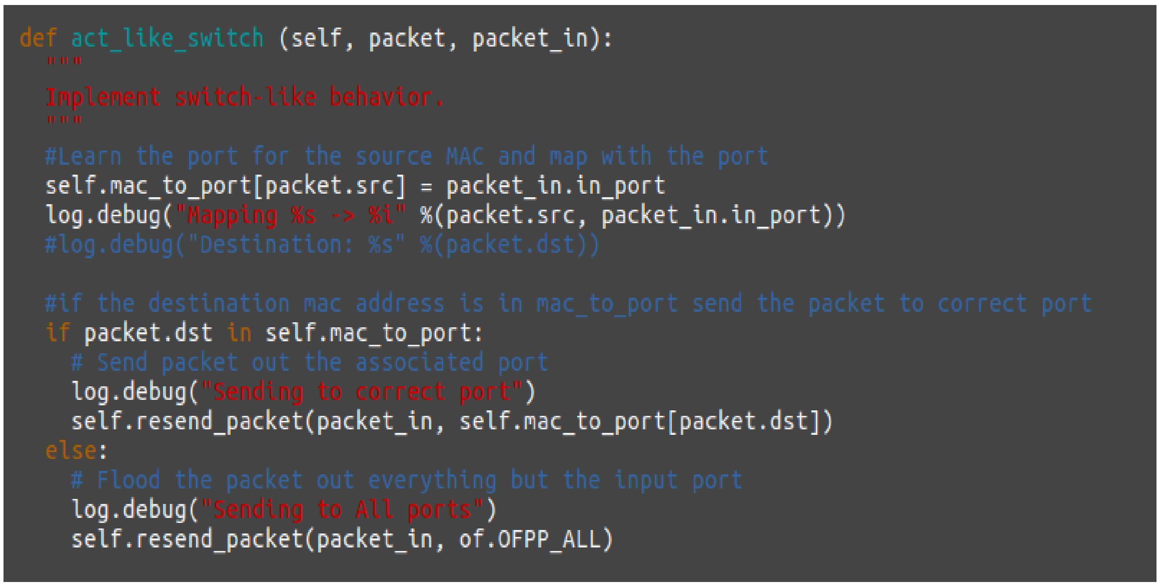
```
root@mininet-vm:~# tcpdump -XX -n -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
21:36:06.586074 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
        0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0000 0a00 0002                 ..........
21:36:06.586137 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
        0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
        0x0020:  0000 0000 0001 0a00 0001                 ..........
21:36:06.589051 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 6191, seq 1, lengt
h 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 24a0 4000 4001 0207 0a00 0001 0a00  .T$.@.@.........
        0x0020:  0002 0800 3c70 182f 0001 b66b 3f59 0000  ....<p./...k?Y..
        0x0030:  0000 e6c7 0800 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
21:36:06.589087 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 6191, seq 1, length
64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 ef2a 0000 4001 777c 0a00 0002 0a00  .T.*..@.w|......
        0x0020:  0001 0000 4470 182f 0001 b66b 3f59 0000  ....Dp./...k?Y..
        0x0030:  0000 e6c7 0800 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
21:36:11.585119 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
        0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
        0x0020:  0000 0000 0000 0a00 0001                 ..........
21:36:11.594702 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
        0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0002 0a00 0002                 ..........
```

Figure 15: "tcpdump" used to check which switches and hosts are receiving packets.



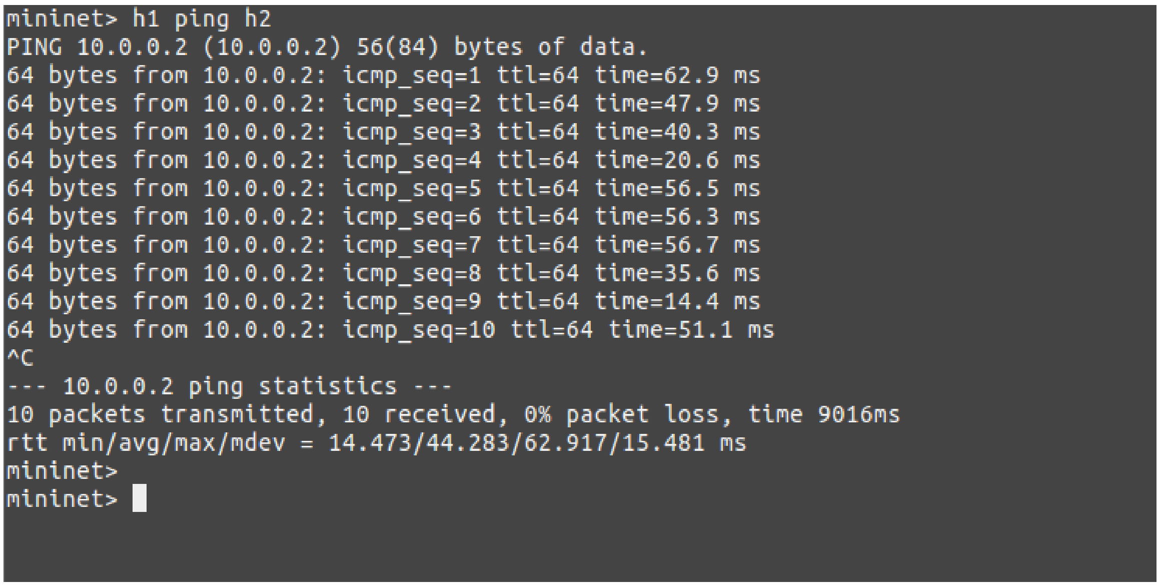
```
Part B Solution:

1:  Ping between "h1 and h2" & "h1 and h5" also works for modified MAC Learning
COntroller. For implementing MAC Leaning controller i copied "of.tutorial.py" file to
"of_tutorial_m.py" and the modified code is shown in Figure 17. In the code entry is
being added to the "map_to_port" table using the information from the packet.

Time taken for each ping is shown in the Figure 18 and Figure 19. For ping "h1 and
h2" packet for ARP packets were recieved by every host and switch but ICMP packets
only "h2" and "S2" observed these packets. The reason can be while ARP port is mapped
to MAC address and while ICMP packets are only forwarded to corresponidng ports. It
was the same case for ping "h1 and h5" as well. "tcpdump" was used here as well to
check which hosts and switches recieve packets when ping test is performed.

In Figure 20 i printed out the output of the controller which prints when a packet is
flooded in the network and when it is only forwarded to sprecific port.

2: Output for "iperf h1 h2" and "iperf h1 h5" are shown in the Figure 21 and Figure
22 respectively. Throughput between h1 and h2 is way better than throughput between
h1 and h5. The reason is same as Part A but the overall throughput decreased by some
amount for "h1" and "h2" but it downgraded way worse for "h1" and "h5".

3: Output for "pingall" is shown in Figure 23. Test is performed in the same manner
as Part A.
```

Figure 16: Solution B.



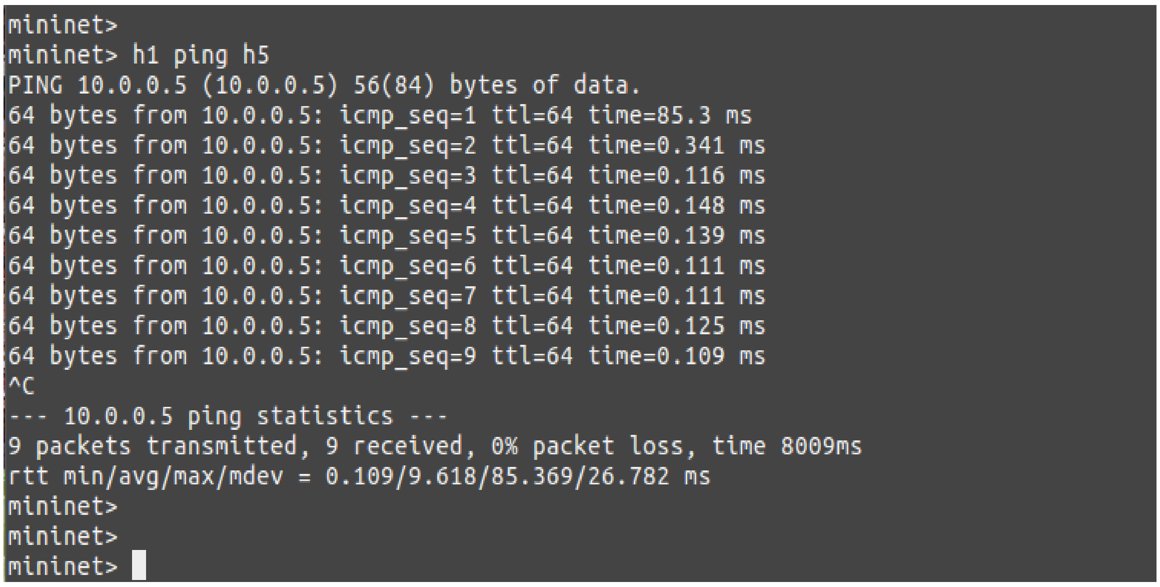
```python
def act_like_switch (self, packet, packet_in):
  """
  Implement switch-like behavior.
  """
  #Learn the port for the source MAC and map with the port
  self.mac_to_port[packet.src] = packet_in.in_port
  log.debug("Mapping %s -> %i" %(packet.src, packet_in.in_port))
  #log.debug("Destination: %s" %(packet.dst))

  #if the destination mac address is in mac_to_port send the packet to correct port
  if packet.dst in self.mac_to_port:
    # Send packet out the associated port
    log.debug("Sending to correct port")
    self.resend_packet(packet_in, self.mac_to_port[packet.dst])
  else:
    # Flood the packet out everything but the input port
    log.debug("Sending to All ports")
    self.resend_packet(packet_in, of.OFPP_ALL)
```

Figure 17: Code for MAC learning. MAC address is mapped to the port and packet is only flooded only if port is not found in map table.



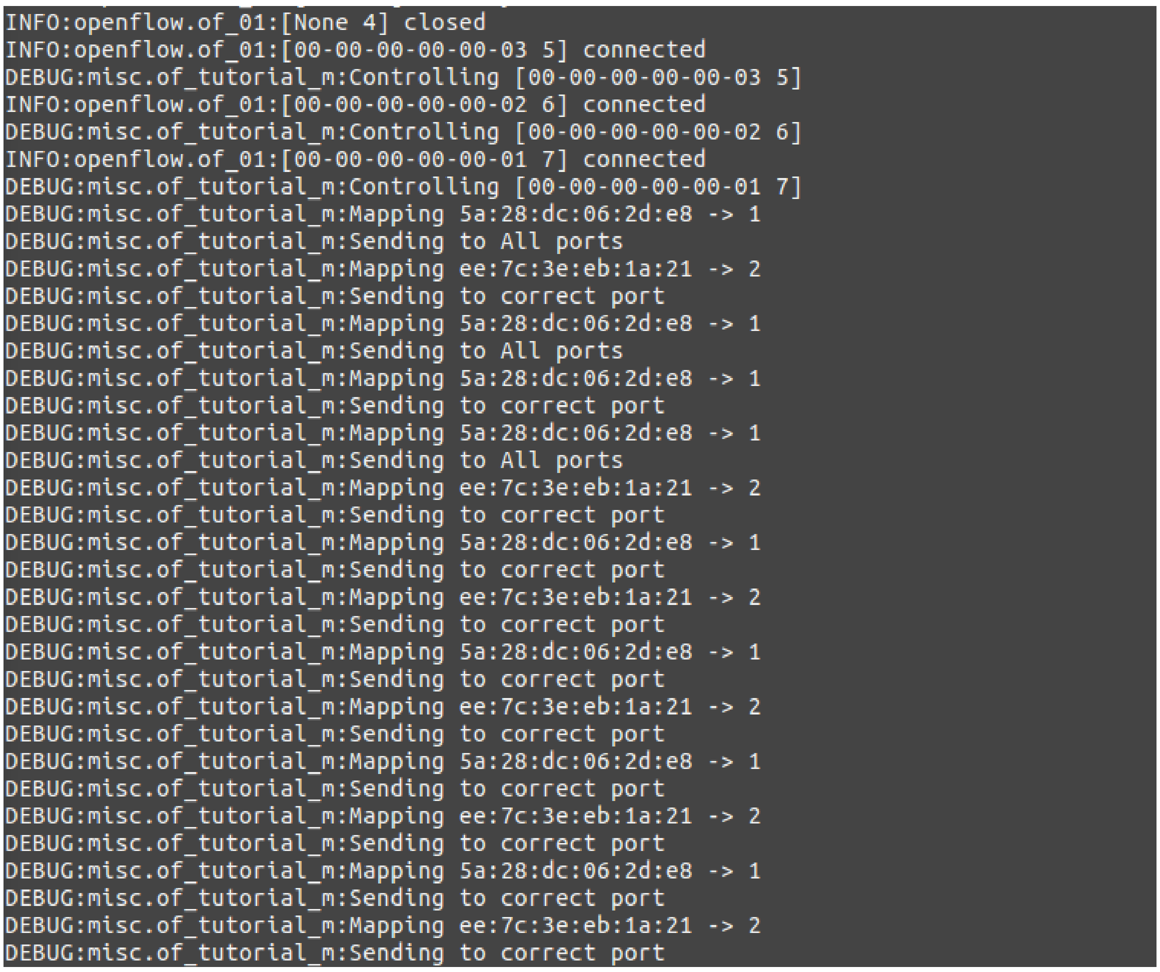
```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=62.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=47.9 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=40.3 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=20.6 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=56.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=56.3 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=56.7 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=35.6 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=14.4 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=51.1 ms
^C
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 14.473/44.283/62.917/15.481 ms
mininet>
mininet>
```

Figure 18: Output for “h1 ping h2” command for MAC learning controller.



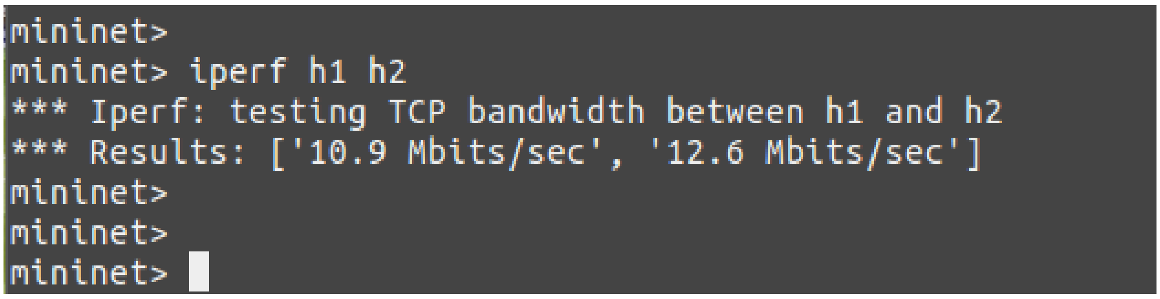
```
mininet>
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=85.3 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.341 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.116 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.148 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.139 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.111 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.111 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=0.125 ms
64 bytes from 10.0.0.5: icmp_seq=9 ttl=64 time=0.109 ms
^C
--- 10.0.0.5 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8009ms
rtt min/avg/max/mdev = 0.109/9.618/85.369/26.782 ms
mininet>
mininet>
mininet>
```

Figure 19: Output for “h1 ping h5” command for MAC learning controller.



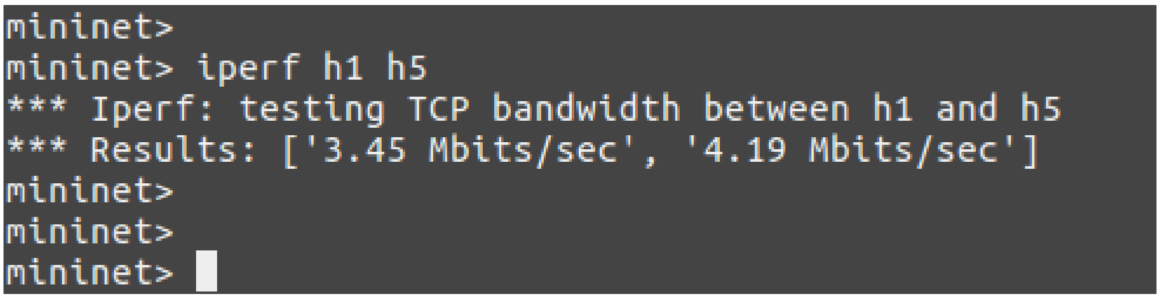
```
INFO:openflow.of_01:[None 4] closed
INFO:openflow.of_01:[00-00-00-00-00-03 5] connected
DEBUG:misc.of_tutorial_m:Controlling [00-00-00-00-00-03 5]
INFO:openflow.of_01:[00-00-00-00-00-02 6] connected
DEBUG:misc.of_tutorial_m:Controlling [00-00-00-00-00-02 6]
INFO:openflow.of_01:[00-00-00-00-00-01 7] connected
DEBUG:misc.of_tutorial_m:Controlling [00-00-00-00-00-01 7]
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping ee:7c:3e:eb:1a:21 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping ee:7c:3e:eb:1a:21 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping ee:7c:3e:eb:1a:21 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping ee:7c:3e:eb:1a:21 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping ee:7c:3e:eb:1a:21 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping 5a:28:dc:06:2d:e8 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Mapping ee:7c:3e:eb:1a:21 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
```

Figure 20: Output of MAC learning controller which shows which packet is being forwarded to correct port and which one is flooded out.



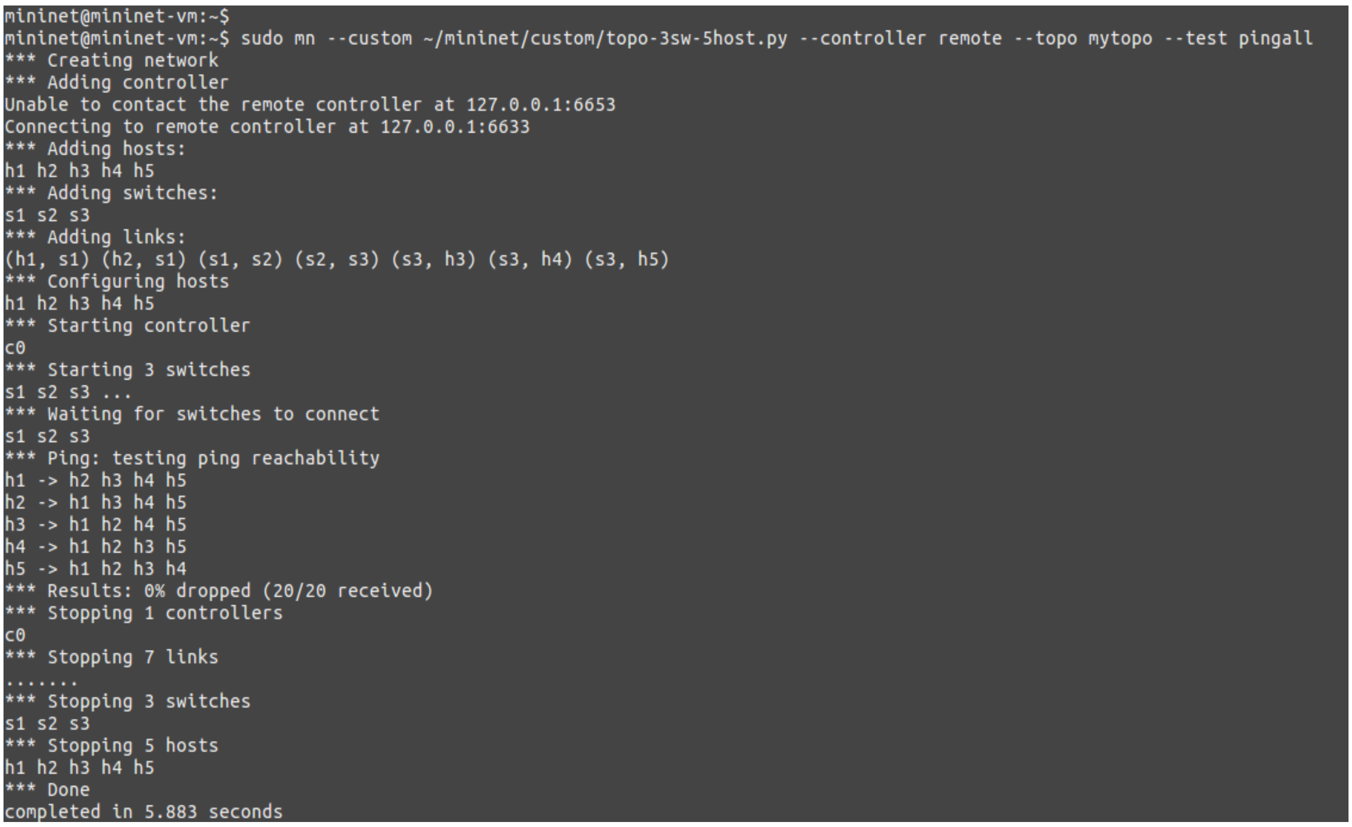
```
mininet>
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['10.9 Mbits/sec', '12.6 Mbits/sec']
mininet>
mininet>
mininet>
```

Figure 21: Throughput of h1 and h2 by running "iperf h1 h2" command.



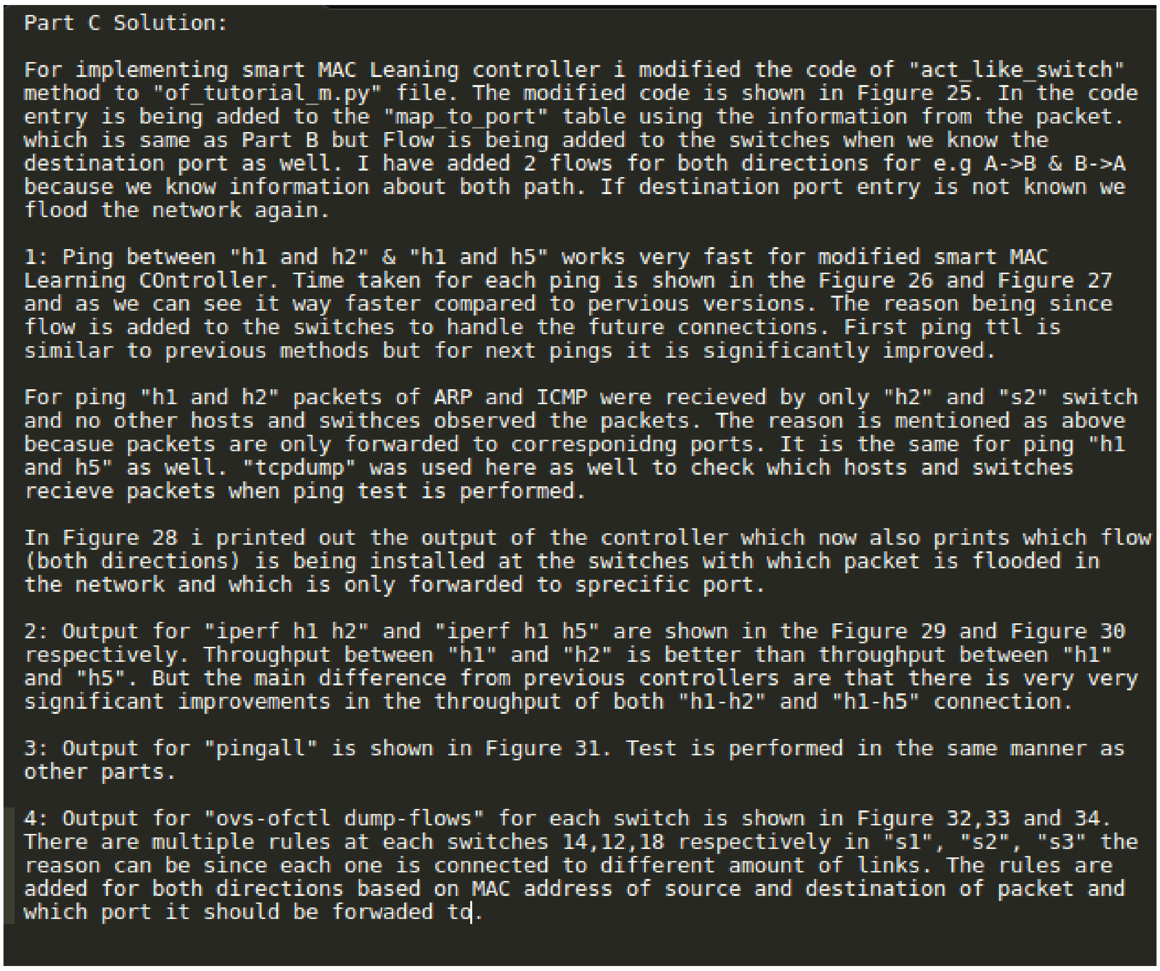
```
mininet>
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['3.45 Mbits/sec', '4.19 Mbits/sec']
mininet>
mininet>
mininet>
```

Figure 22: Throughput of h1 and h2 by running "iperf h1 h2" command.



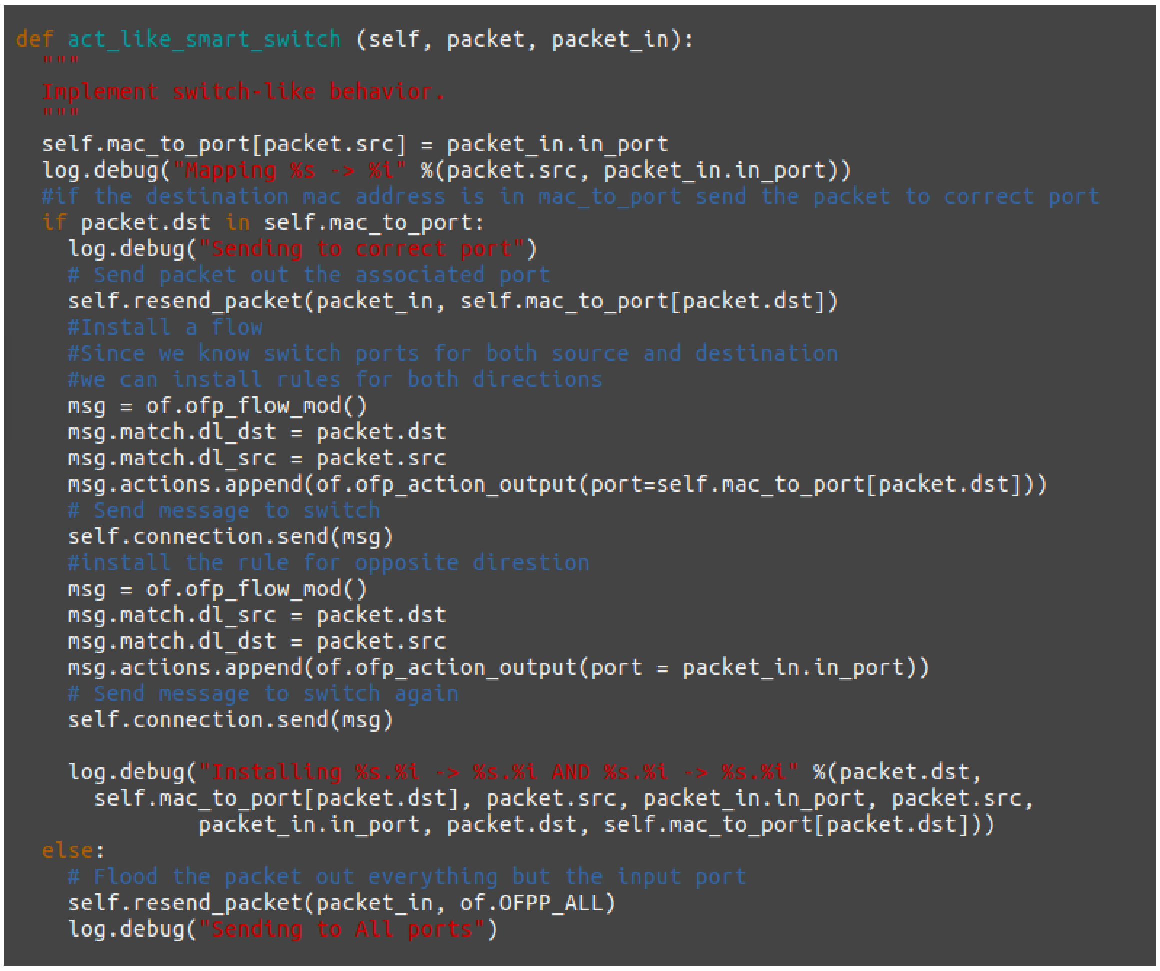
```
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo-3sw-5host.py --controller remote --topo mytopo --test pingall
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s1) (s1, s2) (s2, s3) (s3, h3) (s3, h4) (s3, h5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Waiting for switches to connect
s1 s2 s3
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
completed in 5.883 seconds
```

Figure 23: Output of "pingall" command test for MAC learning controller.



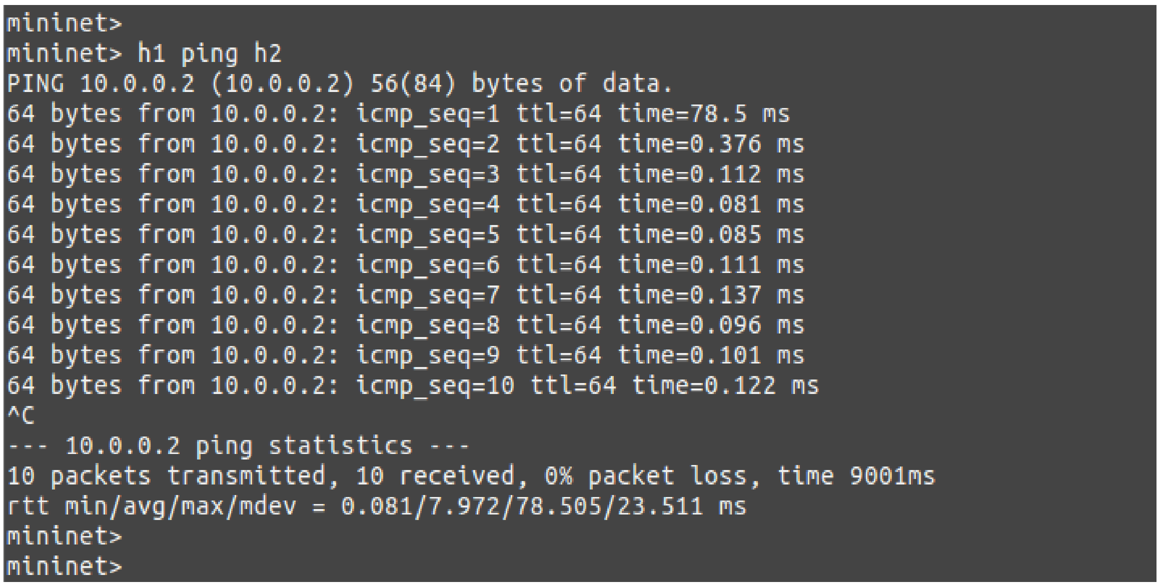
```
Part C Solution:

For implementing smart MAC Leaning controller i modified the code of "act_like_switch"
method to "of_tutorial_m.py" file. The modified code is shown in Figure 25. In the code
entry is being added to the "map_to_port" table using the information from the packet.
which is same as Part B but Flow is being added to the switches when we know the
destination port as well. I have added 2 flows for both directions for e.g A->B & B->A
because we know information about both path. If destination port entry is not known we
flood the network again.

1: Ping between "h1 and h2" & "h1 and h5" works very fast for modified smart MAC
Learning COntroller. Time taken for each ping is shown in the Figure 26 and Figure 27
and as we can see it way faster compared to pervious versions. The reason being since
flow is added to the switches to handle the future connections. First ping ttl is
similar to previous methods but for next pings it is significantly improved.

For ping "h1 and h2" packets of ARP and ICMP were recieved by only "h2" and "s2" switch
and no other hosts and swithces observed the packets. The reason is mentioned as above
becasue packets are only forwarded to corresponidng ports. It is the same for ping "h1
and h5" as well. "tcpdump" was used here as well to check which hosts and switches
recieve packets when ping test is performed.

In Figure 28 i printed out the output of the controller which now also prints which flow
(both directions) is being installed at the switches with which packet is flooded in
the network and which is only forwarded to sprecific port.

2: Output for "iperf h1 h2" and "iperf h1 h5" are shown in the Figure 29 and Figure 30
respectively. Throughput between "h1" and "h2" is better than throughput between "h1"
and "h5". But the main difference from previous controllers are that there is very very
significant improvements in the throughput of both "h1-h2" and "h1-h5" connection.

3: Output for "pingall" is shown in Figure 31. Test is performed in the same manner as
other parts.

4: Output for "ovs-ofctl dump-flows" for each switch is shown in Figure 32,33 and 34.
There are multiple rules at each switches 14,12,18 respectively in "s1", "s2", "s3" the
reason can be since each one is connected to different amount of links. The rules are
added for both directions based on MAC address of source and destination of packet and
which port it should be forwaded to.
```

Figure 24: Solution C.



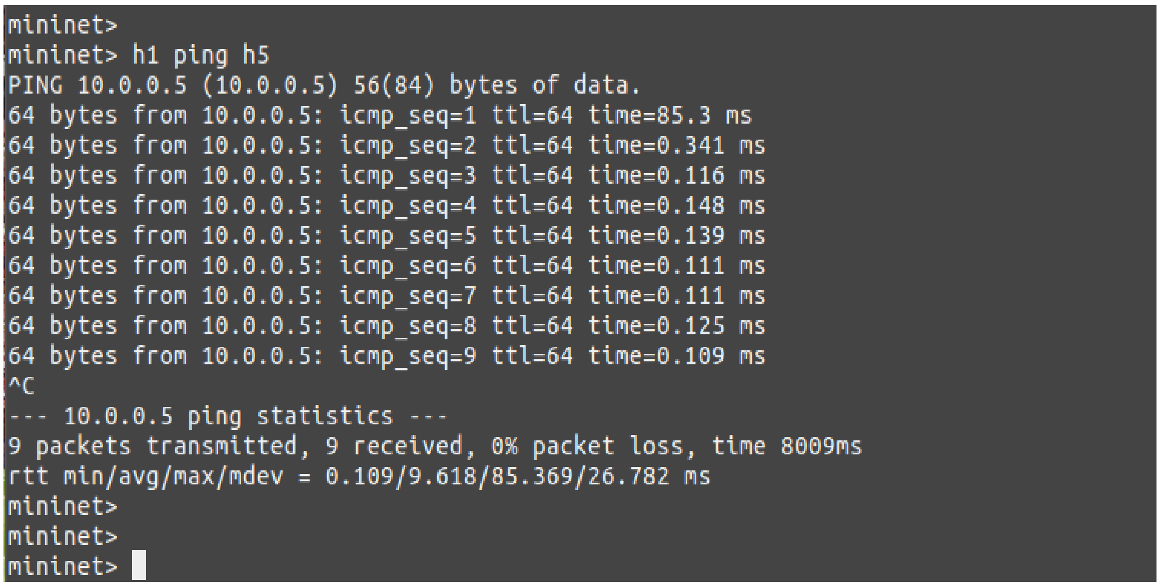
```python
def act_like_smart_switch (self, packet, packet_in):
  """
  Implement switch-like behavior.
  """
  self.mac_to_port[packet.src] = packet_in.in_port
  log.debug("Mapping %s -> %i" %(packet.src, packet_in.in_port))
  #if the destination mac address is in mac_to_port send the packet to correct port
  if packet.dst in self.mac_to_port:
    log.debug("Sending to correct port")
    # Send packet out the associated port
    self.resend_packet(packet_in, self.mac_to_port[packet.dst])
    #Install a flow
    #Since we know switch ports for both source and destination
    #we can install rules for both directions
    msg = of.ofp_flow_mod()
    msg.match.dl_dst = packet.dst
    msg.match.dl_src = packet.src
    msg.actions.append(of.ofp_action_output(port=self.mac_to_port[packet.dst]))
    # Send message to switch
    self.connection.send(msg)
    #install the rule for opposite direstion
    msg = of.ofp_flow_mod()
    msg.match.dl_src = packet.dst
    msg.match.dl_dst = packet.src
    msg.actions.append(of.ofp_action_output(port = packet_in.in_port))
    # Send message to switch again
    self.connection.send(msg)

    log.debug("Installing %s.%i -> %s.%i AND %s.%i -> %s.%i" %(packet.dst,
      self.mac_to_port[packet.dst], packet.src, packet_in.in_port, packet.src,
              packet_in.in_port, packet.dst, self.mac_to_port[packet.dst]))
  else:
    # Flood the packet out everything but the input port
    self.resend_packet(packet_in, of.OFPP_ALL)
    log.debug("Sending to All ports")
```

Figure 25: Code for Smart MAC learning. MAC address is mapped to the port and packet is only flooded only if port is not found in map table.



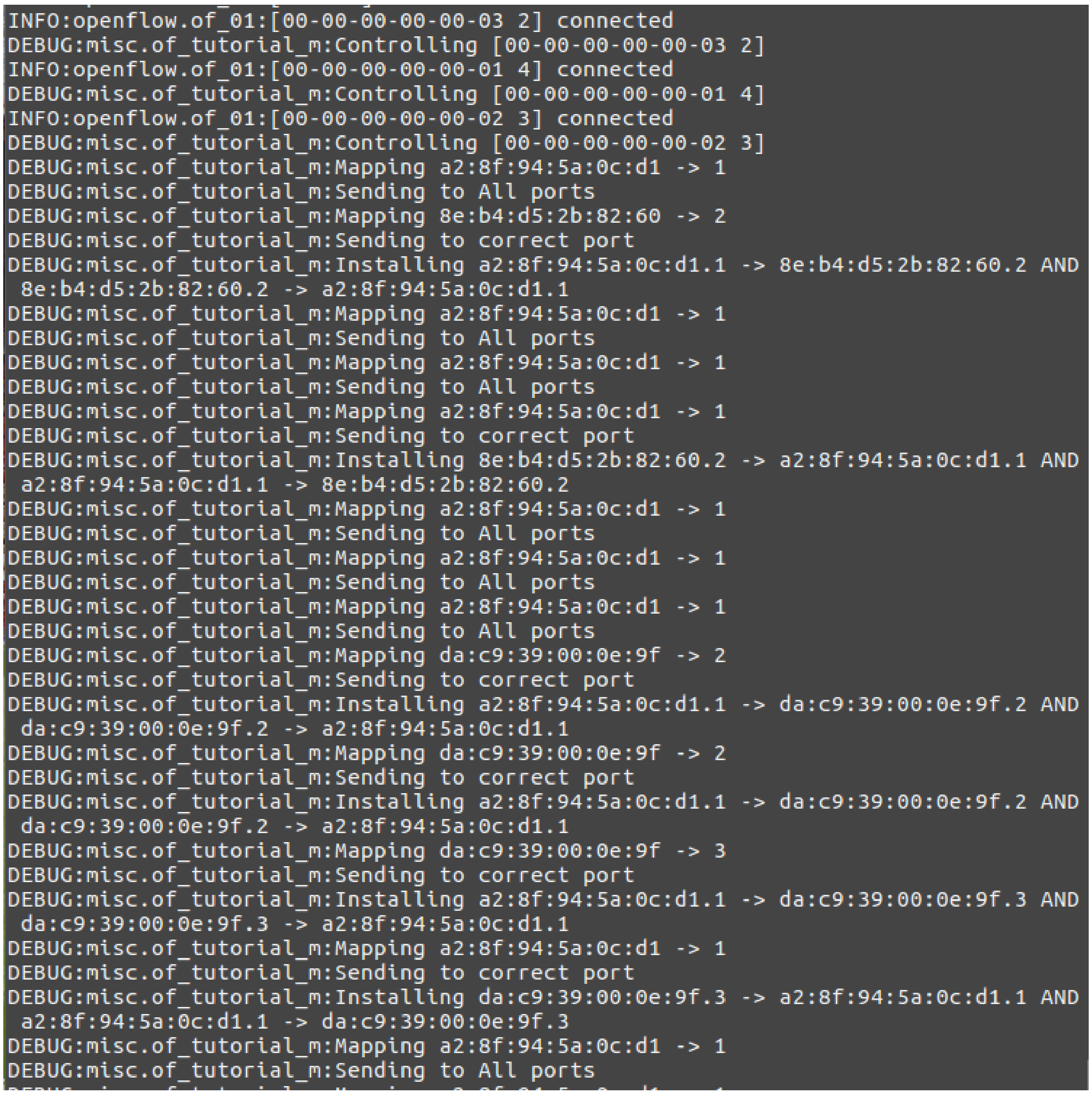
```
mininet>
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=78.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.376 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.112 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.081 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.085 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.111 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.137 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.101 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.122 ms
^C
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.081/7.972/78.505/23.511 ms
mininet>
mininet>
```

Figure 26: Output for “h1 ping h2” command for Smart MAC learning controller.



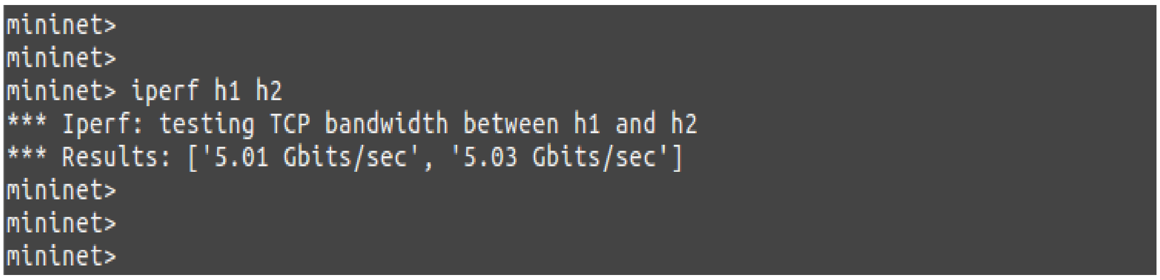
```
mininet>
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=85.3 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.341 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.116 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.148 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.139 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.111 ms
64 bytes from 10.0.0.5: icmp_seq=7 ttl=64 time=0.111 ms
64 bytes from 10.0.0.5: icmp_seq=8 ttl=64 time=0.125 ms
64 bytes from 10.0.0.5: icmp_seq=9 ttl=64 time=0.109 ms
^C
--- 10.0.0.5 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8009ms
rtt min/avg/max/mdev = 0.109/9.618/85.369/26.782 ms
mininet>
mininet>
mininet>
```

Figure 27: Output for “h1 ping h5” command for Smart MAC learning controller.



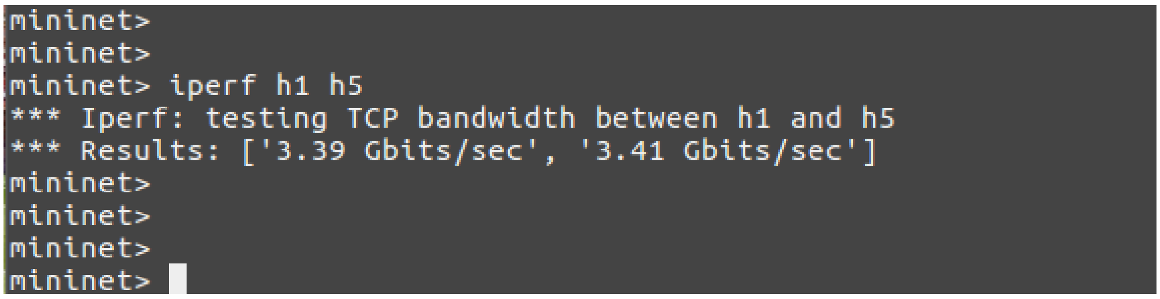
```
INFO:openflow.of_01:[00-00-00-00-00-03 2] connected
DEBUG:misc.of_tutorial_m:Controlling [00-00-00-00-00-03 2]
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
DEBUG:misc.of_tutorial_m:Controlling [00-00-00-00-00-01 4]
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
DEBUG:misc.of_tutorial_m:Controlling [00-00-00-00-00-02 3]
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping 8e:b4:d5:2b:82:60 -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Installing a2:8f:94:5a:0c:d1.1 -> 8e:b4:d5:2b:82:60.2 AND
 8e:b4:d5:2b:82:60.2 -> a2:8f:94:5a:0c:d1.1
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Installing 8e:b4:d5:2b:82:60.2 -> a2:8f:94:5a:0c:d1.1 AND
 a2:8f:94:5a:0c:d1.1 -> 8e:b4:d5:2b:82:60.2
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
DEBUG:misc.of_tutorial_m:Mapping da:c9:39:00:0e:9f -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Installing a2:8f:94:5a:0c:d1.1 -> da:c9:39:00:0e:9f.2 AND
 da:c9:39:00:0e:9f.2 -> a2:8f:94:5a:0c:d1.1
DEBUG:misc.of_tutorial_m:Mapping da:c9:39:00:0e:9f -> 2
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Installing a2:8f:94:5a:0c:d1.1 -> da:c9:39:00:0e:9f.2 AND
 da:c9:39:00:0e:9f.2 -> a2:8f:94:5a:0c:d1.1
DEBUG:misc.of_tutorial_m:Mapping da:c9:39:00:0e:9f -> 3
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Installing a2:8f:94:5a:0c:d1.1 -> da:c9:39:00:0e:9f.3 AND
 da:c9:39:00:0e:9f.3 -> a2:8f:94:5a:0c:d1.1
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to correct port
DEBUG:misc.of_tutorial_m:Installing da:c9:39:00:0e:9f.3 -> a2:8f:94:5a:0c:d1.1 AND
 a2:8f:94:5a:0c:d1.1 -> da:c9:39:00:0e:9f.3
DEBUG:misc.of_tutorial_m:Mapping a2:8f:94:5a:0c:d1 -> 1
DEBUG:misc.of_tutorial_m:Sending to All ports
```

Figure 28: Output of MAC learning controller which shows which packet is being forwarded to correct port and which one is flooded out.



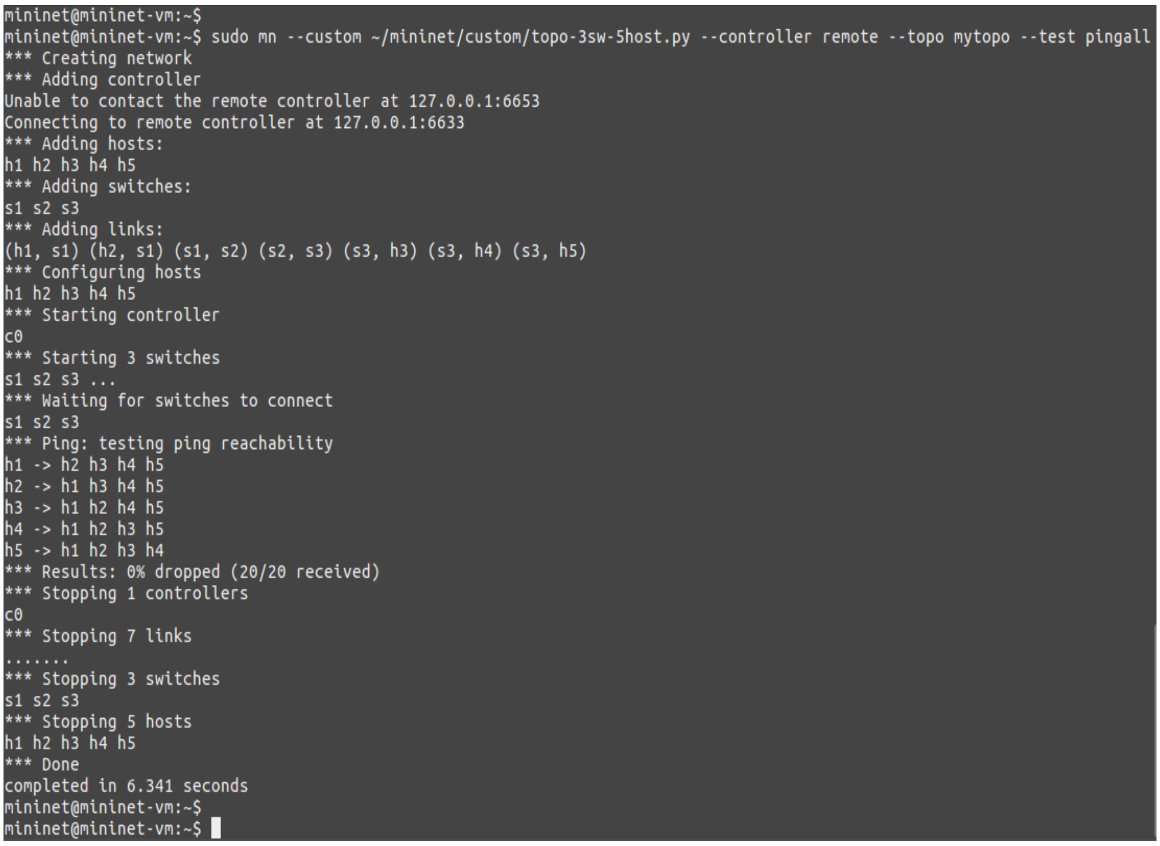
```
mininet>
mininet>
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['5.01 Gbits/sec', '5.03 Gbits/sec']
mininet>
mininet>
mininet>
```

Figure 29: Throughput of h1 and h2 by running “iperf h1 h2” command for Smart MAC learning controller.



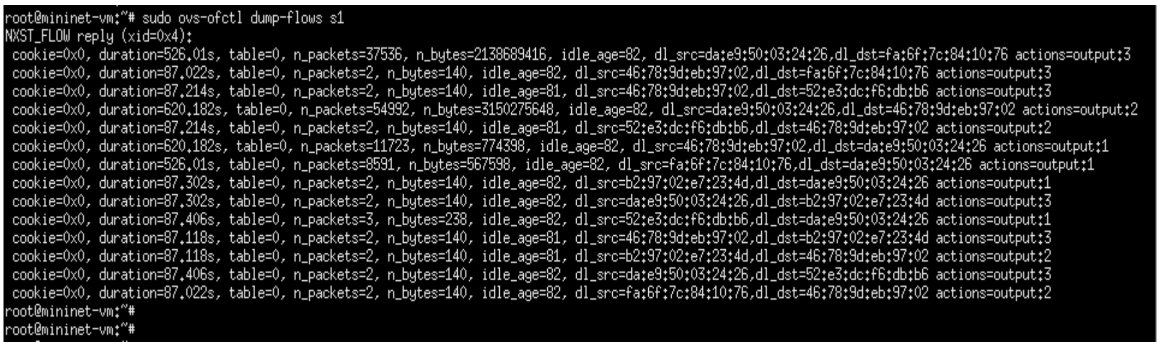
```
mininet>
mininet>
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['3.39 Gbits/sec', '3.41 Gbits/sec']
mininet>
mininet>
mininet>
mininet>
```

Figure 30: Throughput of h1 and h2 by running “iperf h1 h2” command for Smart MAC learning controller.



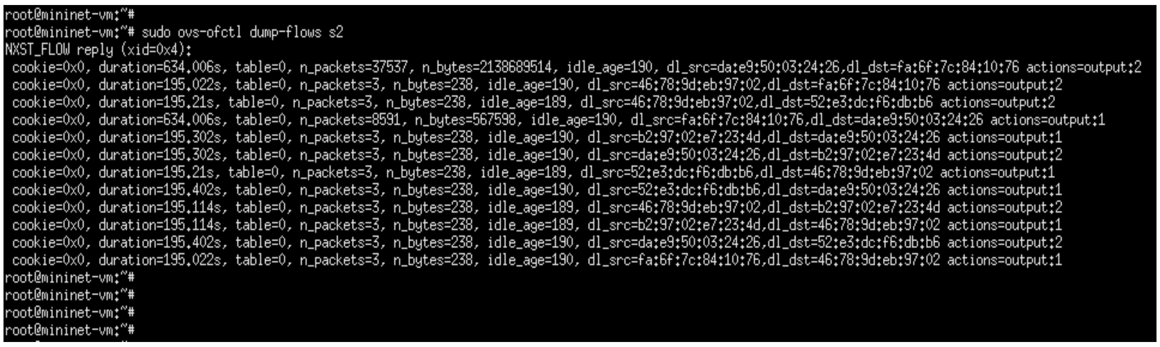
```
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo-3sw-5host.py --controller remote --topo mytopo --test pingall
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s1) (s1, s2) (s2, s3) (s3, h3) (s3, h4) (s3, h5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Waiting for switches to connect
s1 s2 s3
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
completed in 6.341 seconds
mininet@mininet-vm:~$
mininet@mininet-vm:~$
```

Figure 31: Output of “pingall” command test for Smart MAC learning controller.



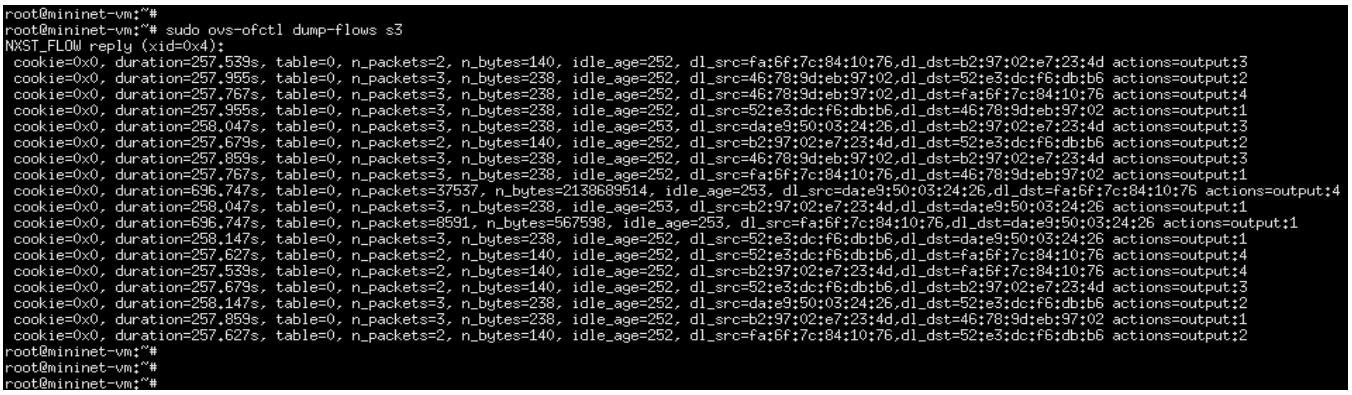
```
root@mininet-vm:~# sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=526.01s, table=0, n_packets=37536, n_bytes=2138689416, idle_age=82, dl_src=da:e9:50:03:24:26,dl_dst=fa:6f:7c:84:10:76 actions=output:3
 cookie=0x0, duration=87.022s, table=0, n_packets=2, n_bytes=140, idle_age=82, dl_src=46:78:9d:eb:97:02,dl_dst=fa:6f:7c:84:10:76 actions=output:3
 cookie=0x0, duration=87.214s, table=0, n_packets=2, n_bytes=140, idle_age=81, dl_src=46:78:9d:eb:97:02,dl_dst=52:e3:dc:f6:db:b6 actions=output:3
 cookie=0x0, duration=620.182s, table=0, n_packets=54992, n_bytes=3150275648, idle_age=82, dl_src=da:e9:50:03:24:26,dl_dst=46:78:9d:eb:97:02 actions=output:2
 cookie=0x0, duration=87.214s, table=0, n_packets=2, n_bytes=140, idle_age=81, dl_src=52:e3:dc:f6:db:b6,dl_dst=46:78:9d:eb:97:02 actions=output:2
 cookie=0x0, duration=620.182s, table=0, n_packets=11723, n_bytes=774398, idle_age=82, dl_src=46:78:9d:eb:97:02,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=526.01s, table=0, n_packets=8591, n_bytes=567598, idle_age=82, dl_src=fa:6f:7c:84:10:76,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=87.302s, table=0, n_packets=2, n_bytes=140, idle_age=82, dl_src=b2:97:02:e7:23:4d,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=87.302s, table=0, n_packets=2, n_bytes=140, idle_age=82, dl_src=da:e9:50:03:24:26,dl_dst=b2:97:02:e7:23:4d actions=output:3
 cookie=0x0, duration=87.406s, table=0, n_packets=3, n_bytes=238, idle_age=82, dl_src=52:e3:dc:f6:db:b6,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=87.118s, table=0, n_packets=2, n_bytes=140, idle_age=81, dl_src=46:78:9d:eb:97:02,dl_dst=b2:97:02:e7:23:4d actions=output:3
 cookie=0x0, duration=87.118s, table=0, n_packets=2, n_bytes=140, idle_age=81, dl_src=b2:97:02:e7:23:4d,dl_dst=46:78:9d:eb:97:02 actions=output:2
 cookie=0x0, duration=87.406s, table=0, n_packets=2, n_bytes=140, idle_age=82, dl_src=da:e9:50:03:24:26,dl_dst=52:e3:dc:f6:db:b6 actions=output:3
 cookie=0x0, duration=87.022s, table=0, n_packets=2, n_bytes=140, idle_age=82, dl_src=fa:6f:7c:84:10:76,dl_dst=46:78:9d:eb:97:02 actions=output:2
root@mininet-vm:~#
root@mininet-vm:~#
```

Figure 32: Output of "ovs-ofctl dump-flows" command at S1 switch for Smart MAC learning controller.



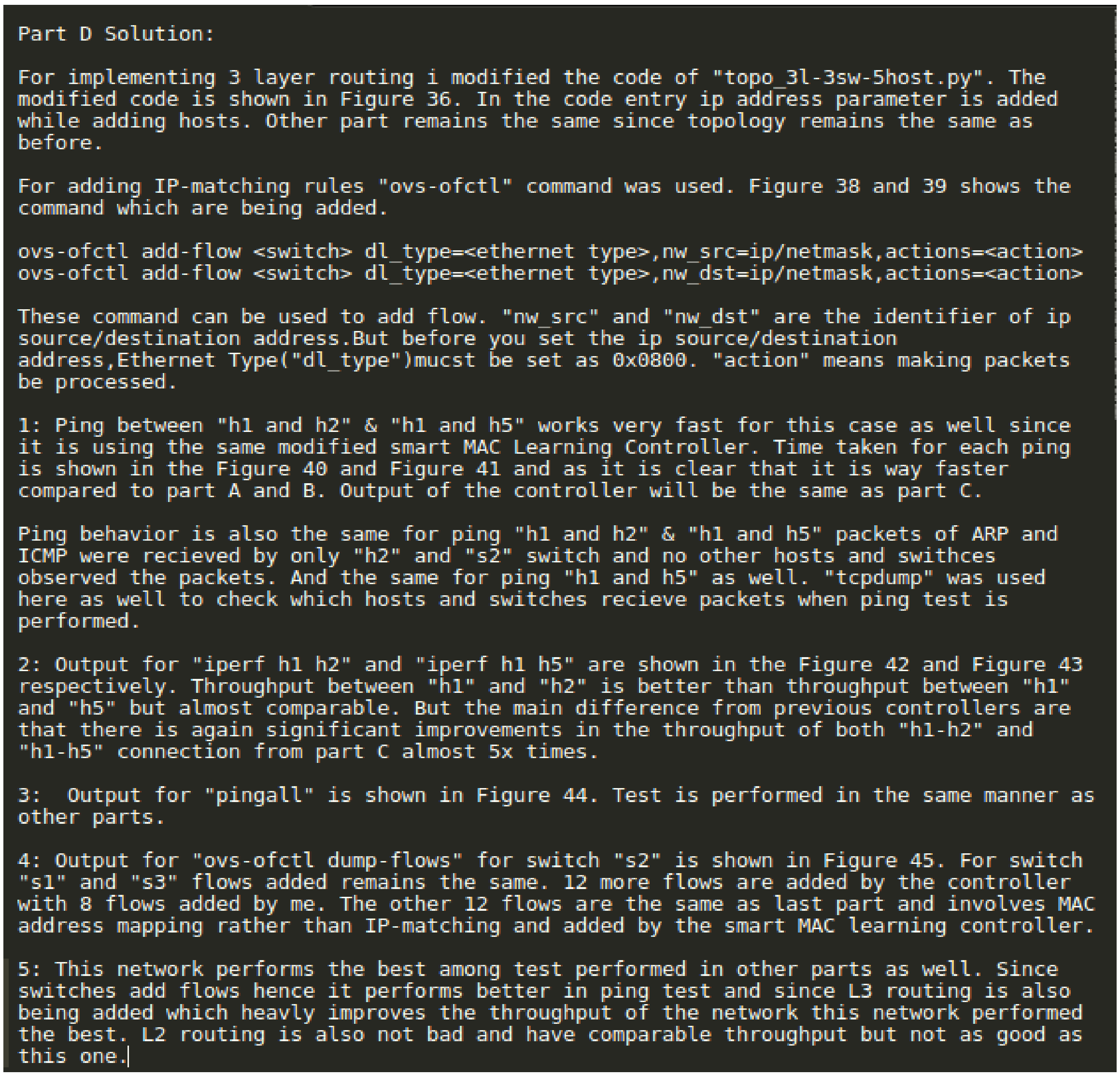
```
root@mininet-vm:~#
root@mininet-vm:~# sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=634.006s, table=0, n_packets=37537, n_bytes=2138689514, idle_age=190, dl_src=da:e9:50:03:24:26,dl_dst=fa:6f:7c:84:10:76 actions=output:2
 cookie=0x0, duration=195.022s, table=0, n_packets=3, n_bytes=238, idle_age=190, dl_src=46:78:9d:eb:97:02,dl_dst=fa:6f:7c:84:10:76 actions=output:2
 cookie=0x0, duration=195.21s, table=0, n_packets=3, n_bytes=238, idle_age=189, dl_src=46:78:9d:eb:97:02,dl_dst=52:e3:dc:f6:db:b6 actions=output:2
 cookie=0x0, duration=634.006s, table=0, n_packets=8591, n_bytes=567598, idle_age=190, dl_src=fa:6f:7c:84:10:76,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=195.302s, table=0, n_packets=3, n_bytes=238, idle_age=190, dl_src=b2:97:02:e7:23:4d,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=195.302s, table=0, n_packets=3, n_bytes=238, idle_age=190, dl_src=da:e9:50:03:24:26,dl_dst=b2:97:02:e7:23:4d actions=output:2
 cookie=0x0, duration=195.21s, table=0, n_packets=3, n_bytes=238, idle_age=189, dl_src=52:e3:dc:f6:db:b6,dl_dst=46:78:9d:eb:97:02 actions=output:1
 cookie=0x0, duration=195.402s, table=0, n_packets=3, n_bytes=238, idle_age=190, dl_src=52:e3:dc:f6:db:b6,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=195.114s, table=0, n_packets=3, n_bytes=238, idle_age=189, dl_src=46:78:9d:eb:97:02,dl_dst=b2:97:02:e7:23:4d actions=output:2
 cookie=0x0, duration=195.114s, table=0, n_packets=3, n_bytes=238, idle_age=189, dl_src=b2:97:02:e7:23:4d,dl_dst=46:78:9d:eb:97:02 actions=output:1
 cookie=0x0, duration=195.402s, table=0, n_packets=3, n_bytes=238, idle_age=190, dl_src=da:e9:50:03:24:26,dl_dst=52:e3:dc:f6:db:b6 actions=output:2
 cookie=0x0, duration=195.022s, table=0, n_packets=3, n_bytes=238, idle_age=190, dl_src=fa:6f:7c:84:10:76,dl_dst=46:78:9d:eb:97:02 actions=output:1
root@mininet-vm:~#
root@mininet-vm:~#
root@mininet-vm:~#
root@mininet-vm:~#
```

Figure 33: Output of "ovs-ofctl dump-flows" command at S2 switch for Smart MAC learning controller.



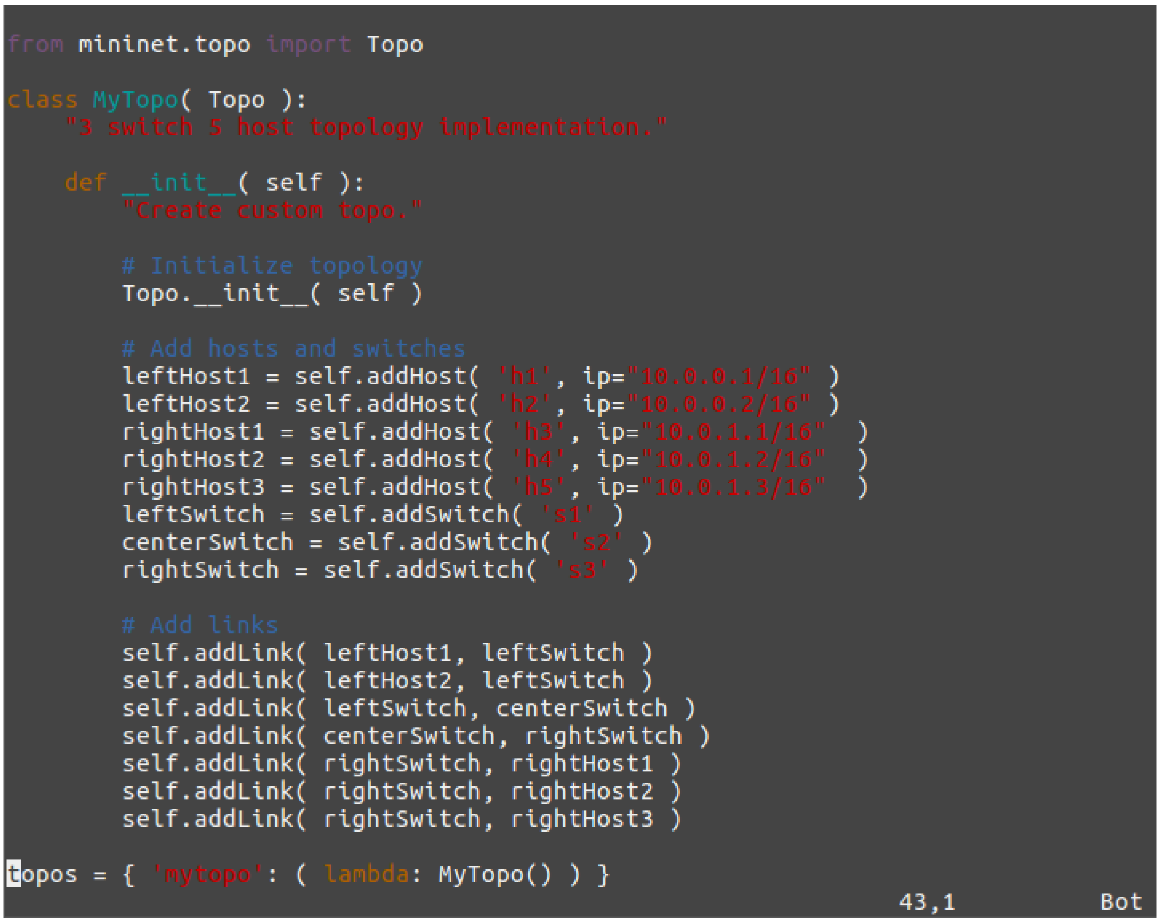
```
root@mininet-vm:~#
root@mininet-vm:~# sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=257.539s, table=0, n_packets=2, n_bytes=140, idle_age=252, dl_src=fa:6f:7c:84:10:76,dl_dst=b2:97:02:e7:23:4d actions=output:3
 cookie=0x0, duration=257.955s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=46:78:9d:eb:97:02,dl_dst=52:e3:dc:f6:db:b6 actions=output:2
 cookie=0x0, duration=257.767s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=46:78:9d:eb:97:02,dl_dst=fa:6f:7c:84:10:76 actions=output:4
 cookie=0x0, duration=257.955s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=52:e3:dc:f6:db:b6,dl_dst=46:78:9d:eb:97:02 actions=output:1
 cookie=0x0, duration=258.047s, table=0, n_packets=3, n_bytes=238, idle_age=253, dl_src=da:e9:50:03:24:26,dl_dst=b2:97:02:e7:23:4d actions=output:3
 cookie=0x0, duration=257.679s, table=0, n_packets=2, n_bytes=140, idle_age=252, dl_src=b2:97:02:e7:23:4d,dl_dst=52:e3:dc:f6:db:b6 actions=output:2
 cookie=0x0, duration=257.859s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=46:78:9d:eb:97:02,dl_dst=b2:97:02:e7:23:4d actions=output:3
 cookie=0x0, duration=257.767s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=fa:6f:7c:84:10:76,dl_dst=46:78:9d:eb:97:02 actions=output:1
 cookie=0x0, duration=696.747s, table=0, n_packets=37537, n_bytes=2138689514, idle_age=253, dl_src=da:e9:50:03:24:26,dl_dst=fa:6f:7c:84:10:76 actions=output:4
 cookie=0x0, duration=258.047s, table=0, n_packets=3, n_bytes=238, idle_age=253, dl_src=b2:97:02:e7:23:4d,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=696.747s, table=0, n_packets=8591, n_bytes=567598, idle_age=253, dl_src=fa:6f:7c:84:10:76,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=258.147s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=52:e3:dc:f6:db:b6,dl_dst=da:e9:50:03:24:26 actions=output:1
 cookie=0x0, duration=257.627s, table=0, n_packets=2, n_bytes=140, idle_age=252, dl_src=52:e3:dc:f6:db:b6,dl_dst=fa:6f:7c:84:10:76 actions=output:4
 cookie=0x0, duration=257.539s, table=0, n_packets=2, n_bytes=140, idle_age=252, dl_src=b2:97:02:e7:23:4d,dl_dst=fa:6f:7c:84:10:76 actions=output:4
 cookie=0x0, duration=257.679s, table=0, n_packets=2, n_bytes=140, idle_age=252, dl_src=52:e3:dc:f6:db:b6,dl_dst=b2:97:02:e7:23:4d actions=output:3
 cookie=0x0, duration=258.147s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=da:e9:50:03:24:26,dl_dst=52:e3:dc:f6:db:b6 actions=output:2
 cookie=0x0, duration=257.859s, table=0, n_packets=3, n_bytes=238, idle_age=252, dl_src=b2:97:02:e7:23:4d,dl_dst=46:78:9d:eb:97:02 actions=output:1
 cookie=0x0, duration=257.627s, table=0, n_packets=2, n_bytes=140, idle_age=252, dl_src=fa:6f:7c:84:10:76,dl_dst=52:e3:dc:f6:db:b6 actions=output:2
root@mininet-vm:~#
root@mininet-vm:~#
root@mininet-vm:~#
```

Figure 34: Output of "ovs-ofctl dump-flows" command at S3 switch for Smart MAC learning controller.



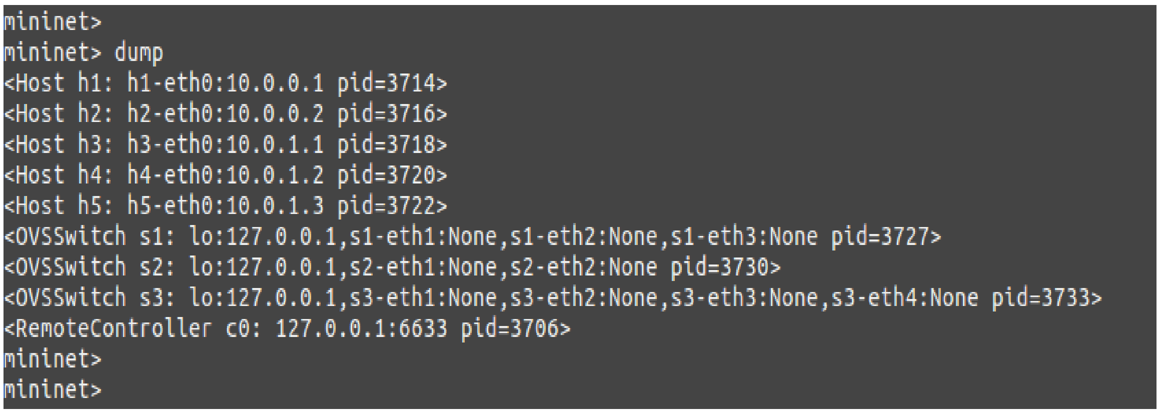
```
Part D Solution:

For implementing 3 layer routing i modified the code of "topo_3l-3sw-5host.py". The
modified code is shown in Figure 36. In the code entry ip address parameter is added
while adding hosts. Other part remains the same since topology remains the same as
before.

For adding IP-matching rules "ovs-ofctl" command was used. Figure 38 and 39 shows the
command which are being added.

ovs-ofctl add-flow <switch> dl_type=<ethernet type>,nw_src=ip/netmask,actions=<action>
ovs-ofctl add-flow <switch> dl_type=<ethernet type>,nw_dst=ip/netmask,actions=<action>

These command can be used to add flow. "nw_src" and "nw_dst" are the identifier of ip
source/destination address.But before you set the ip source/destination
address,Ethernet Type("dl_type")mucst be set as 0x0800. "action" means making packets
be processed.

1: Ping between "h1 and h2" & "h1 and h5" works very fast for this case as well since
it is using the same modified smart MAC Learning Controller. Time taken for each ping
is shown in the Figure 40 and Figure 41 and as it is clear that it is way faster
compared to part A and B. Output of the controller will be the same as part C.

Ping behavior is also the same for ping "h1 and h2" & "h1 and h5" packets of ARP and
ICMP were recieved by only "h2" and "s2" switch and no other hosts and swithces
observed the packets. And the same for ping "h1 and h5" as well. "tcpdump" was used
here as well to check which hosts and switches recieve packets when ping test is
performed.

2: Output for "iperf h1 h2" and "iperf h1 h5" are shown in the Figure 42 and Figure 43
respectively. Throughput between "h1" and "h2" is better than throughput between "h1"
and "h5" but almost comparable. But the main difference from previous controllers are
that there is again significant improvements in the throughput of both "h1-h2" and
"h1-h5" connection from part C almost 5x times.

3:  Output for "pingall" is shown in Figure 44. Test is performed in the same manner as
other parts.

4: Output for "ovs-ofctl dump-flows" for switch "s2" is shown in Figure 45. For switch
"s1" and "s3" flows added remains the same. 12 more flows are added by the controller
with 8 flows added by me. The other 12 flows are the same as last part and involves MAC
address mapping rather than IP-matching and added by the smart MAC learning controller.

5: This network performs the best among test performed in other parts as well. Since
switches add flows hence it performs better in ping test and since L3 routing is also
being added which heavly improves the throughput of the network this network performed
the best. L2 routing is also not bad and have comparable throughput but not as good as
this one.
```

Figure 35: Solution D.



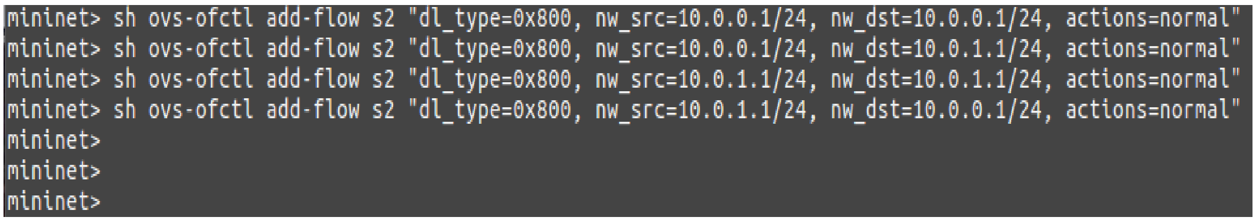
```
from mininet.topo import Topo

class MyTopo( Topo ):
    "3 switch 5 host topology implementation."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        leftHost1 = self.addHost( 'h1', ip="10.0.0.1/16" )
        leftHost2 = self.addHost( 'h2', ip="10.0.0.2/16" )
        rightHost1 = self.addHost( 'h3', ip="10.0.1.1/16"  )
        rightHost2 = self.addHost( 'h4', ip="10.0.1.2/16"  )
        rightHost3 = self.addHost( 'h5', ip="10.0.1.3/16"  )
        leftSwitch = self.addSwitch( 's1' )
        centerSwitch = self.addSwitch( 's2' )
        rightSwitch = self.addSwitch( 's3' )

        # Add links
        self.addLink( leftHost1, leftSwitch )
        self.addLink( leftHost2, leftSwitch )
        self.addLink( leftSwitch, centerSwitch )
        self.addLink( centerSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost1 )
        self.addLink( rightSwitch, rightHost2 )
        self.addLink( rightSwitch, rightHost3 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
                                                            43,1          Bot
```

Figure 36: Code for Smart MAC learning. MAC address is mapped to the port and packet is only flooded only if port is not found in map table.



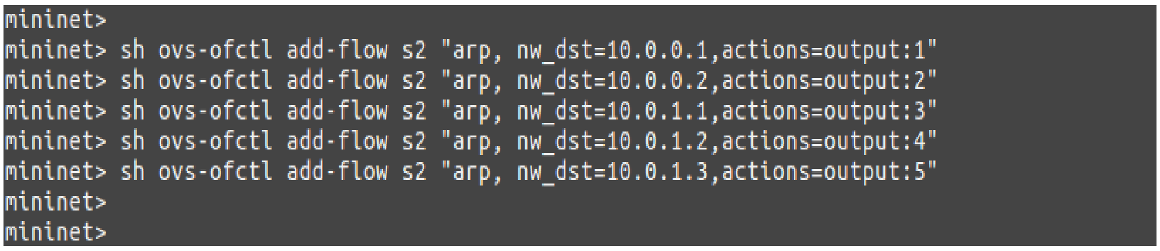
```
mininet>
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3714>
<Host h2: h2-eth0:10.0.0.2 pid=3716>
<Host h3: h3-eth0:10.0.1.1 pid=3718>
<Host h4: h4-eth0:10.0.1.2 pid=3720>
<Host h5: h5-eth0:10.0.1.3 pid=3722>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=3727>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=3730>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None,s3-eth4:None pid=3733>
<RemoteController c0: 127.0.0.1:6633 pid=3706>
mininet>
mininet>
```

Figure 37: Network topology created with specified IP/Subnets in the assignment.



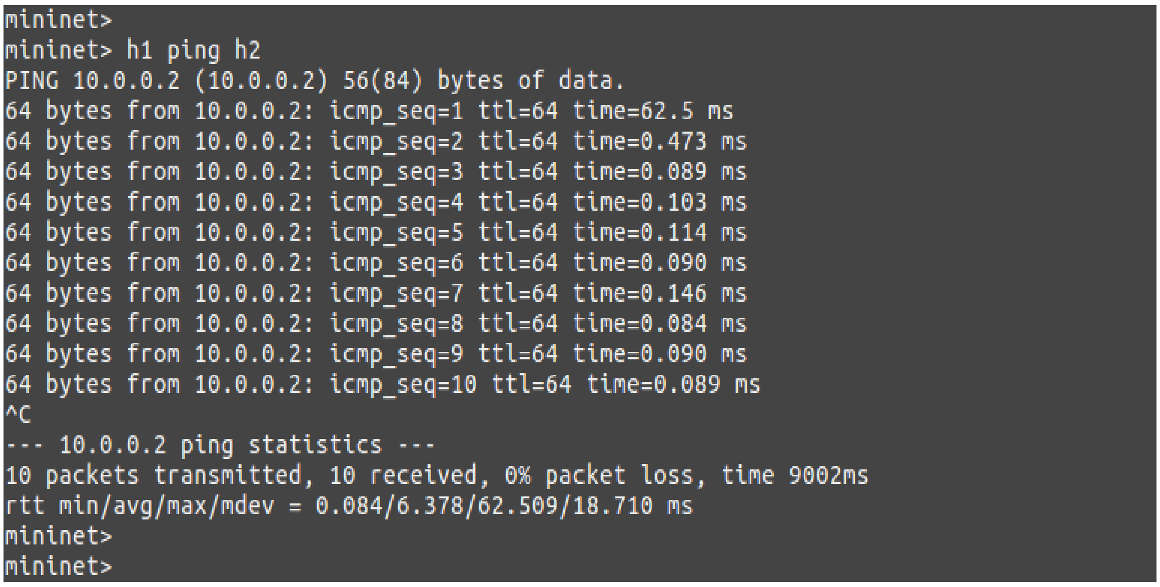
```
mininet> sh ovs-ofctl add-flow s2 "dl_type=0x800, nw_src=10.0.0.1/24, nw_dst=10.0.0.1/24, actions=normal"
mininet> sh ovs-ofctl add-flow s2 "dl_type=0x800, nw_src=10.0.0.1/24, nw_dst=10.0.1.1/24, actions=normal"
mininet> sh ovs-ofctl add-flow s2 "dl_type=0x800, nw_src=10.0.1.1/24, nw_dst=10.0.1.1/24, actions=normal"
mininet> sh ovs-ofctl add-flow s2 "dl_type=0x800, nw_src=10.0.1.1/24, nw_dst=10.0.0.1/24, actions=normal"
mininet>
mininet>
mininet>
```

Figure 38: IP matching Rule addition



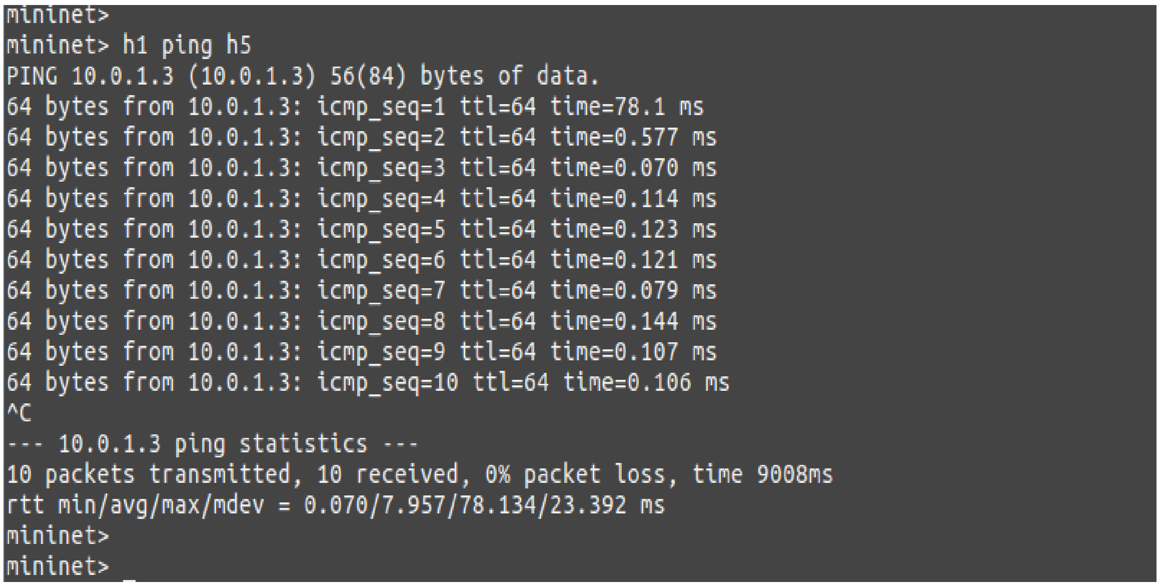
```
mininet>
mininet> sh ovs-ofctl add-flow s2 "arp, nw_dst=10.0.0.1,actions=output:1"
mininet> sh ovs-ofctl add-flow s2 "arp, nw_dst=10.0.0.2,actions=output:2"
mininet> sh ovs-ofctl add-flow s2 "arp, nw_dst=10.0.1.1,actions=output:3"
mininet> sh ovs-ofctl add-flow s2 "arp, nw_dst=10.0.1.2,actions=output:4"
mininet> sh ovs-ofctl add-flow s2 "arp, nw_dst=10.0.1.3,actions=output:5"
mininet>
mininet>
```

Figure 39: IP matching Rule addition for specific address.



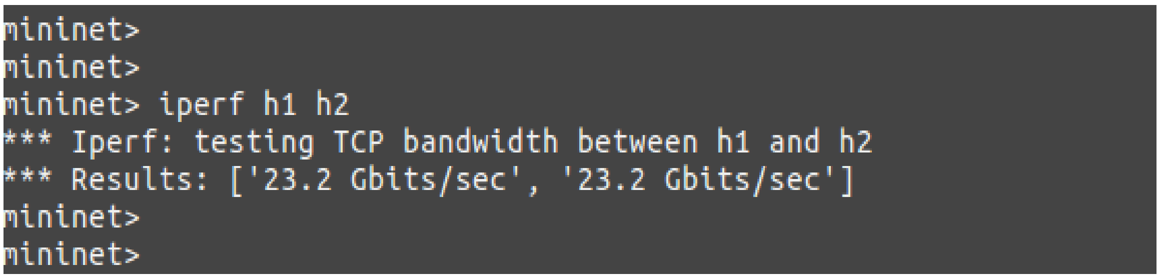
```
mininet>
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=62.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.473 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.103 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.146 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.089 ms
^C
--- 10.0.0.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9002ms
rtt min/avg/max/mdev = 0.084/6.378/62.509/18.710 ms
mininet>
mininet>
```

Figure 40: Output for "h1 ping h2" command for Smart MAC learning controller.



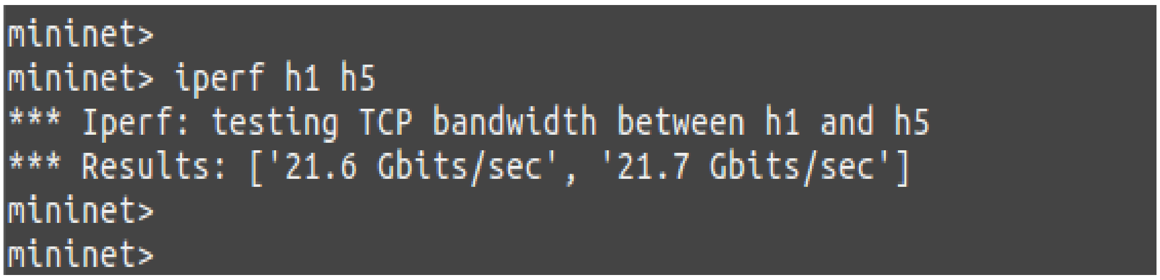
```
mininet>
mininet> h1 ping h5
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=78.1 ms
64 bytes from 10.0.1.3: icmp_seq=2 ttl=64 time=0.577 ms
64 bytes from 10.0.1.3: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 10.0.1.3: icmp_seq=4 ttl=64 time=0.114 ms
64 bytes from 10.0.1.3: icmp_seq=5 ttl=64 time=0.123 ms
64 bytes from 10.0.1.3: icmp_seq=6 ttl=64 time=0.121 ms
64 bytes from 10.0.1.3: icmp_seq=7 ttl=64 time=0.079 ms
64 bytes from 10.0.1.3: icmp_seq=8 ttl=64 time=0.144 ms
64 bytes from 10.0.1.3: icmp_seq=9 ttl=64 time=0.107 ms
64 bytes from 10.0.1.3: icmp_seq=10 ttl=64 time=0.106 ms
^C
--- 10.0.1.3 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9008ms
rtt min/avg/max/mdev = 0.070/7.957/78.134/23.392 ms
mininet>
mininet> _
```

Figure 41: Output for "h1 ping h5" command for Smart MAC learning controller.



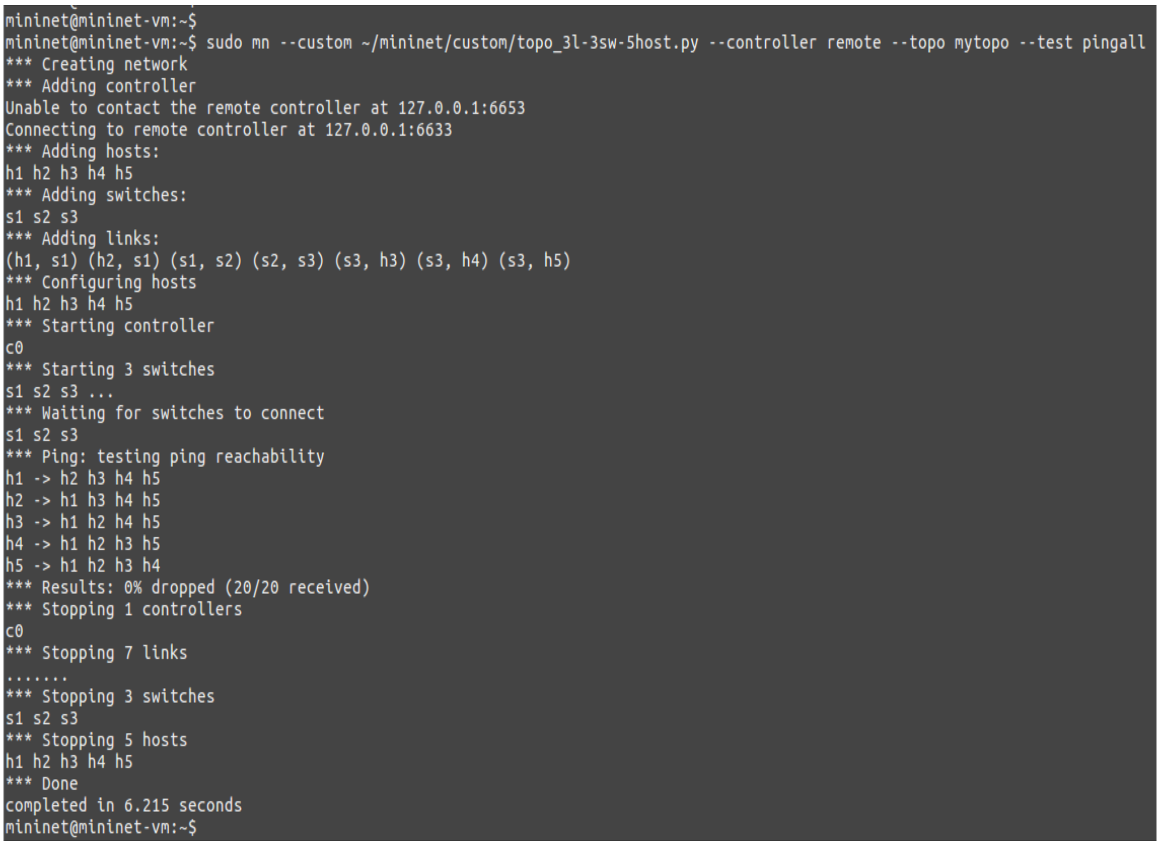
```
mininet>
mininet>
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['23.2 Gbits/sec', '23.2 Gbits/sec']
mininet>
mininet>
```

Figure 42: Throughput of h1 and h2 by running "iperf h1 h2" command for Smart MAC learning controller.



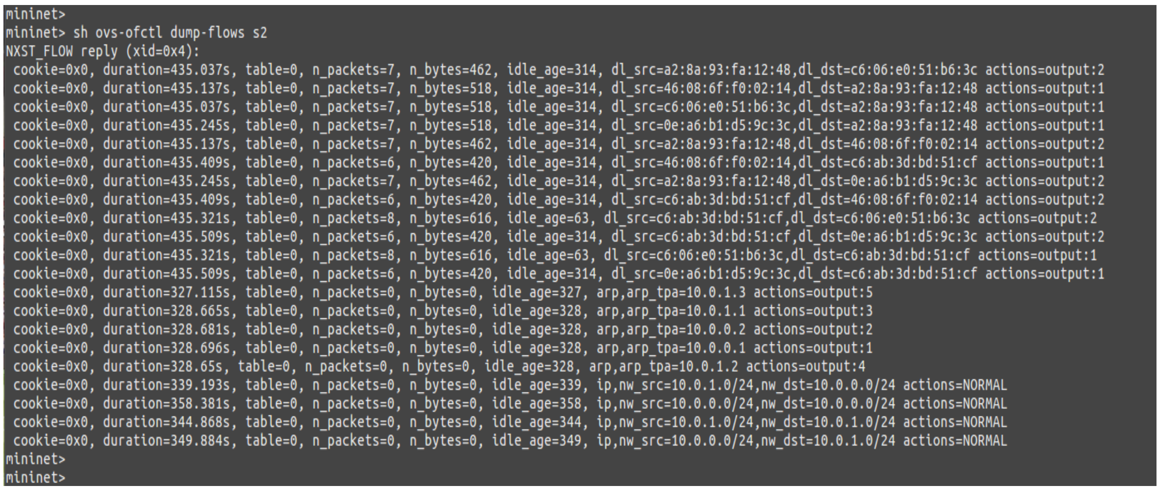
```
mininet>
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['21.6 Gbits/sec', '21.7 Gbits/sec']
mininet>
mininet>
```

Figure 43: Throughput of h1 and h2 by running "iperf h1 h2" command for Smart MAC learning controller.



```
mininet@mininet-vm:~$
mininet@mininet-vm:~$ sudo mn --custom ~/mininet/custom/topo_3l-3sw-5host.py --controller remote --topo mytopo --test pingall
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s1) (s1, s2) (s2, s3) (s3, h3) (s3, h4) (s3, h5)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Waiting for switches to connect
s1 s2 s3
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
completed in 6.215 seconds
mininet@mininet-vm:~$
```

Figure 44: Output of "pingall" command test for Smart MAC learning controller.



```
mininet>
mininet> sh ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=435.037s, table=0, n_packets=7, n_bytes=462, idle_age=314, dl_src=a2:8a:93:fa:12:48,dl_dst=c6:06:e0:51:b6:3c actions=output:2
 cookie=0x0, duration=435.137s, table=0, n_packets=7, n_bytes=518, idle_age=314, dl_src=46:08:6f:f0:02:14,dl_dst=a2:8a:93:fa:12:48 actions=output:1
 cookie=0x0, duration=435.037s, table=0, n_packets=7, n_bytes=518, idle_age=314, dl_src=c6:06:e0:51:b6:3c,dl_dst=a2:8a:93:fa:12:48 actions=output:1
 cookie=0x0, duration=435.245s, table=0, n_packets=7, n_bytes=518, idle_age=314, dl_src=0e:a6:b1:d5:9c:3c,dl_dst=a2:8a:93:fa:12:48 actions=output:1
 cookie=0x0, duration=435.137s, table=0, n_packets=7, n_bytes=462, idle_age=314, dl_src=a2:8a:93:fa:12:48,dl_dst=46:08:6f:f0:02:14 actions=output:2
 cookie=0x0, duration=435.409s, table=0, n_packets=6, n_bytes=420, idle_age=314, dl_src=46:08:6f:f0:02:14,dl_dst=c6:ab:3d:bd:51:cf actions=output:1
 cookie=0x0, duration=435.245s, table=0, n_packets=7, n_bytes=462, idle_age=314, dl_src=a2:8a:93:fa:12:48,dl_dst=0e:a6:b1:d5:9c:3c actions=output:2
 cookie=0x0, duration=435.409s, table=0, n_packets=6, n_bytes=420, idle_age=314, dl_src=c6:ab:3d:bd:51:cf,dl_dst=46:08:6f:f0:02:14 actions=output:2
 cookie=0x0, duration=435.321s, table=0, n_packets=8, n_bytes=616, idle_age=63, dl_src=c6:ab:3d:bd:51:cf,dl_dst=c6:06:e0:51:b6:3c actions=output:2
 cookie=0x0, duration=435.509s, table=0, n_packets=6, n_bytes=420, idle_age=314, dl_src=c6:ab:3d:bd:51:cf,dl_dst=0e:a6:b1:d5:9c:3c actions=output:2
 cookie=0x0, duration=435.321s, table=0, n_packets=8, n_bytes=616, idle_age=63, dl_src=c6:06:e0:51:b6:3c,dl_dst=c6:ab:3d:bd:51:cf actions=output:1
 cookie=0x0, duration=435.509s, table=0, n_packets=6, n_bytes=420, idle_age=314, dl_src=0e:a6:b1:d5:9c:3c,dl_dst=c6:ab:3d:bd:51:cf actions=output:1
 cookie=0x0, duration=327.115s, table=0, n_packets=0, n_bytes=0, idle_age=327, arp,arp_tpa=10.0.1.3 actions=output:5
 cookie=0x0, duration=328.665s, table=0, n_packets=0, n_bytes=0, idle_age=328, arp,arp_tpa=10.0.1.1 actions=output:3
 cookie=0x0, duration=328.681s, table=0, n_packets=0, n_bytes=0, idle_age=328, arp,arp_tpa=10.0.0.2 actions=output:2
 cookie=0x0, duration=328.696s, table=0, n_packets=0, n_bytes=0, idle_age=328, arp,arp_tpa=10.0.0.1 actions=output:1
 cookie=0x0, duration=328.65s, table=0, n_packets=0, n_bytes=0, idle_age=328, arp,arp_tpa=10.0.1.2 actions=output:4
 cookie=0x0, duration=339.193s, table=0, n_packets=0, n_bytes=0, idle_age=339, ip,nw_src=10.0.1.0/24,nw_dst=10.0.0.0/24 actions=NORMAL
 cookie=0x0, duration=358.381s, table=0, n_packets=0, n_bytes=0, idle_age=358, ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24 actions=NORMAL
 cookie=0x0, duration=344.868s, table=0, n_packets=0, n_bytes=0, idle_age=344, ip,nw_src=10.0.1.0/24,nw_dst=10.0.1.0/24 actions=NORMAL
 cookie=0x0, duration=349.884s, table=0, n_packets=0, n_bytes=0, idle_age=349, ip,nw_src=10.0.0.0/24,nw_dst=10.0.1.0/24 actions=NORMAL
mininet>
mininet>
```

Figure 45: Output of "ovs-ofctl dump-flows" command at S2 switch for Smart MAC learning controller.