Final Project Report

EE555

Xiaowen Tan, Yiming Shi

4/28/2020

Scenario 1:

Description: In this part, we first visit github.com/Mininet/OpenFlow-tutorial to read the wiki about how to install Mininet VM and try some test case which built-in Mininet VM. Then we find of_tutorial.py. This is very similar to what we should do on controller_1. So, we slightly modify it. The basic idea of a switch is to match the destination mac address and send the frame. And the topology is also very simple, we just need to fill in hosts' IP addresses, gateway, subnet mask, and add a link between host and switch. Then we are done.

Command:

Step 1: Start Mininet VM

Step 2: Open two terminals on your local computer (we use Putty as terminal)

Step 3: In first terminal we copy Controller_1.py to /home/mininet/pox/pox/misc and then back to /home/mininet/pox , under this folder we can see pox.py

Step 4: In the second terminal we copy topology_1 to /home/mininet/mininet/custom

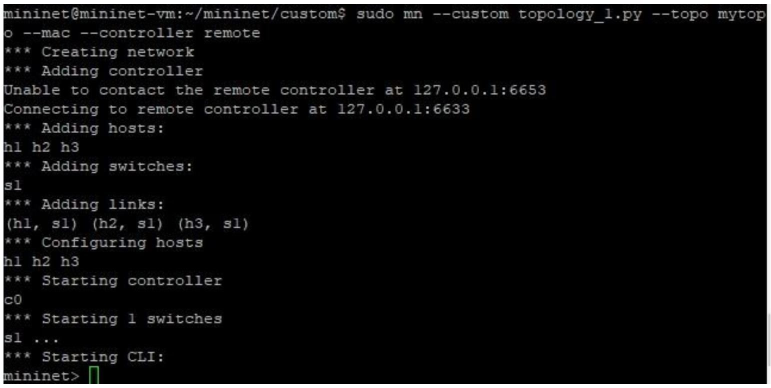
Step 5: We ran command sudo mn -c to clear all controller

Step 6: In the first terminal, we start controller_1 with command line :

./pox.py log.level --DEBUG misc.controller_1

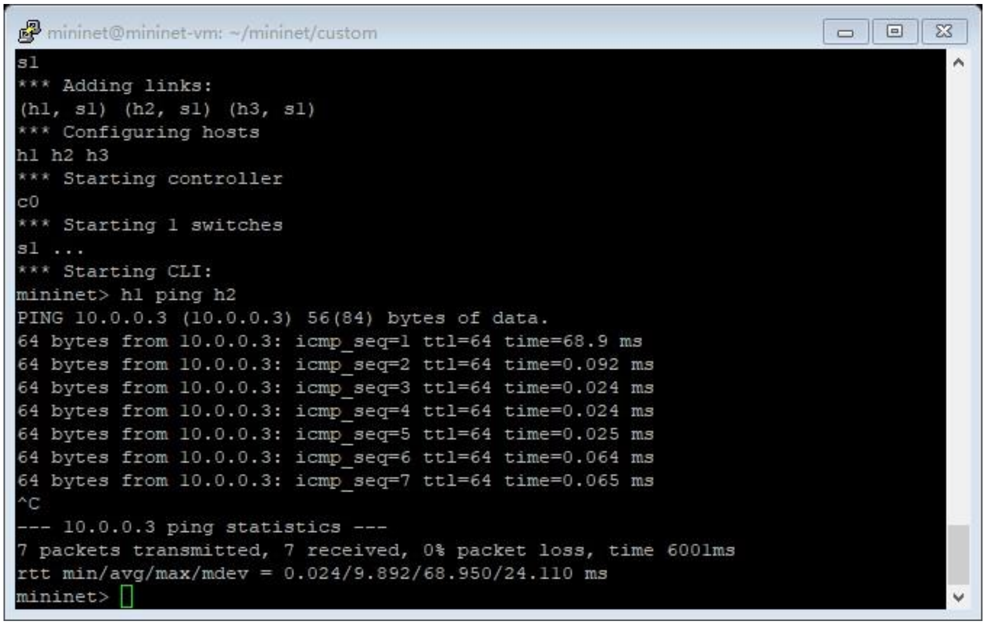
Step 7:In the second terminal, we load the topology_1 with command line: sudo

mn --custom topology_1.py --topo mytopo --mac --controller remote



```
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom topology_1.py --topo mytop
o --mac --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
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> 
```

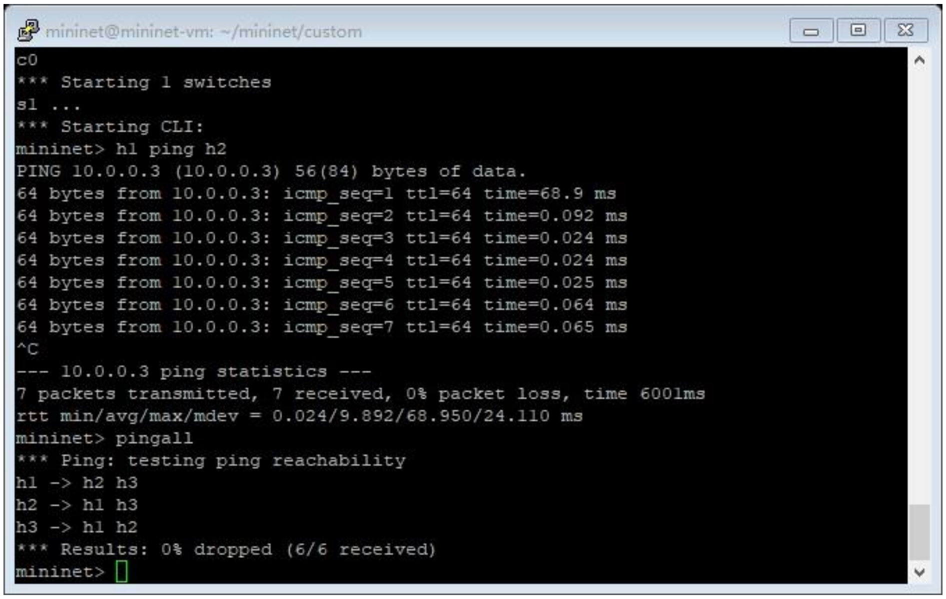
Result: h1 ping h2



```
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping h2
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=68.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.092 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.024 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.024 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.025 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.064 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.065 ms
^C
--- 10.0.0.3 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6001ms
rtt min/avg/max/mdev = 0.024/9.892/68.950/24.110 ms
mininet> 
```

Connection between h1 and h2 is established, and switch works.

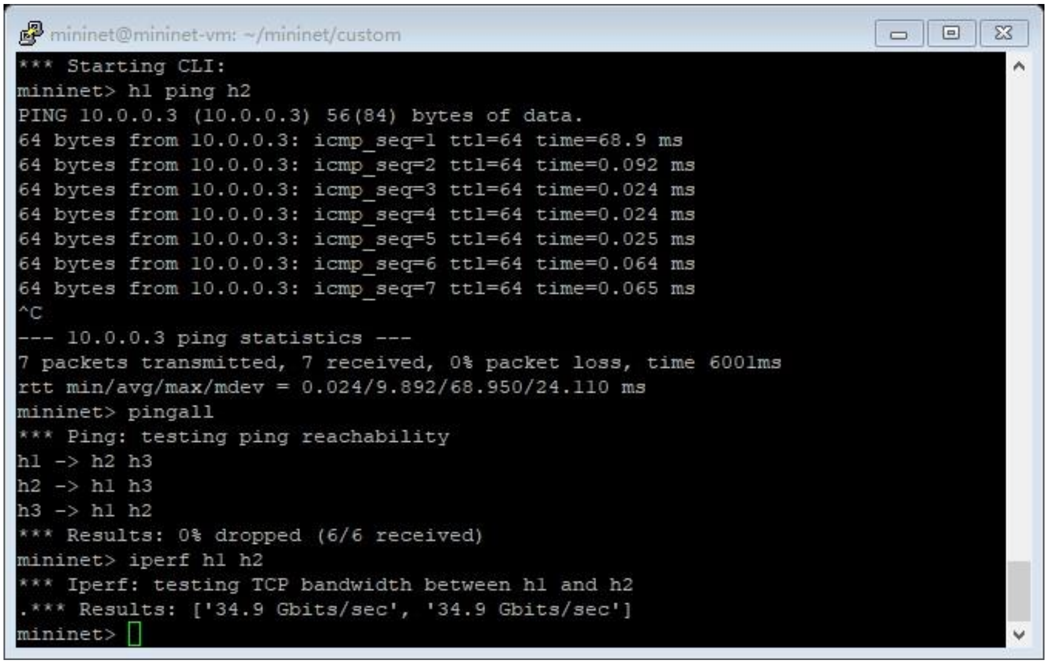
Pingall



```
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping h2
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=68.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.092 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.024 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.024 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.025 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.064 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.065 ms
^C
--- 10.0.0.3 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6001ms
rtt min/avg/max/mdev = 0.024/9.892/68.950/24.110 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Topology is right. And switch works.

Iperf h1 h2



```
*** Starting CLI:
mininet> h1 ping h2
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=68.9 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.092 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.024 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.024 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.025 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.064 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.065 ms
^C
--- 10.0.0.3 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6001ms
rtt min/avg/max/mdev = 0.024/9.892/68.950/24.110 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['34.9 Gbits/sec', '34.9 Gbits/sec']
mininet>
```

Iperf is a command to test the tcp connection speed between two host.

Scenario 2:

Description: We replace the switch with a router from scenario 1. Router is different from switch because it will deal with IP packet and ARP packets and switches are only responsible for distributing frames

In this part, router logical is referenced from https://github.com/suzhou1898/Switch-Router-withOpenFlow-.

For IP packet, router would firstly check whether the destination IP belongs to the network. if it does not, router would send ICMP_unreach packet to source host. When it comes to ICMP packet, the router would check whether the destination IP is an interface of the router, router would reply ICMP_REPLY to the source host. Otherwise, router would forward packet with known destination mac address through its port to the destination host or cache the packet and send ARP request to get the mac address of destination host. When it gets corresponding ARP reply, it would clear cached packets with the mac in ARP reply.

When it comes to normal IP packet, router would send packet with known destination mac address through its port to the destination host or cache the packet and send ARP request to get the mac address of destination host. When it gets corresponding ARP reply, it would clear cached packets with the mac in ARP reply

For ARP packet, when router receives an ARP request, router would check whether the packet is sent to it. If the packet is sent to the router, router would reply its corresponding interface mac address with ARP reply, and record source host mac address. When router receives an ARP reply, it would record source host mac address and clear its cached packet with mac address in the ARP reply.

Related function description in Controller_2:

def _handle_PacketIn (self, event): This determine what should do when a packet comes in.

def handle_arp(self, packet, packet_info, port_in): start arp process or deal with arp request/reply/pending

def send_frame(self, packet, frame_type, src, dst, outport): given destination mac address, frame_type and port number to send the frame with source information

def arp_packet(self, opcode, dst_mac, dst_ip, outport): arp packets with destination ip and mac address output port number and opcode

def icmp_unreach_packet(self, ip_packet, inport): When ICMP packet is unreached, send unreach packet def handle_IP(self, packet, IP_packet, port_in): handle regular IP packets def

handle_ICMP(self,IP_packet, port_in, packet): handle ICMP packets

Command:

Step 1: Start Mininet VM

Step 2: Open two terminals on your local computer (we use Putty as terminal)

Step 3: In first terminal we copy Controller_2.py to /home/mininet/pox/pox/misc and then back to /home/mininet/pox , under this folder we can see pox.py

Step 4: In the second terminal we copy topology_2 to /home/mininet/mininet/custom

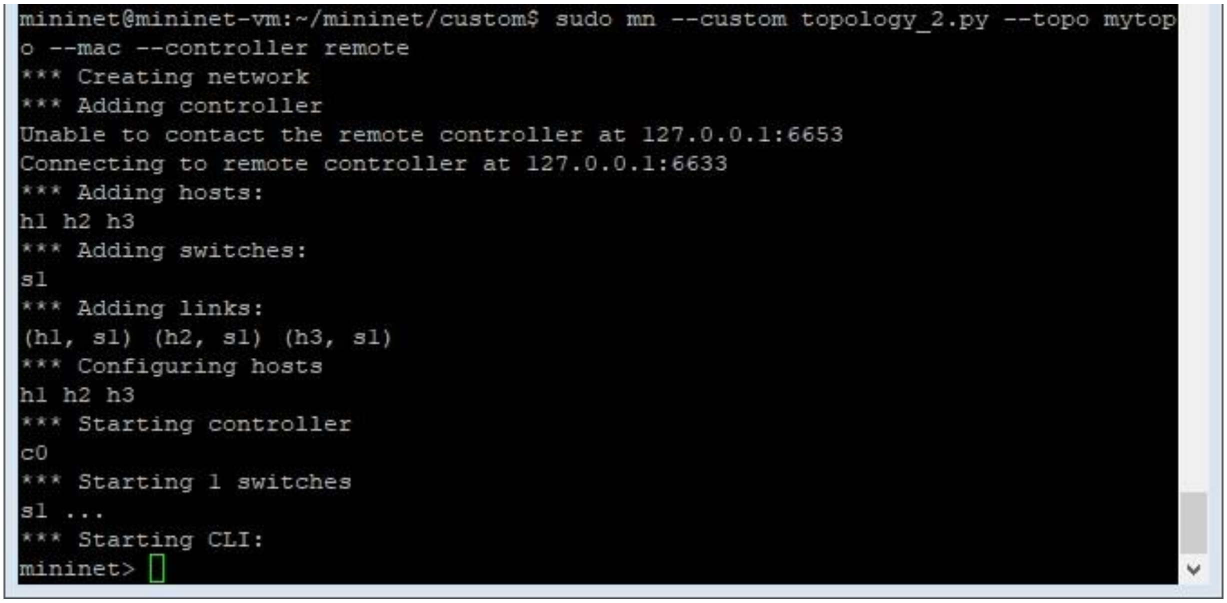
Step 5: We ran command sudo mn -c to clear all controller

Step 6: In the first terminal, we start controller_2 with command line :

./pox.py log.level --DEBUG misc.controller_2

Step 7:In the second terminal, we load the topology_2 with command line: sudo

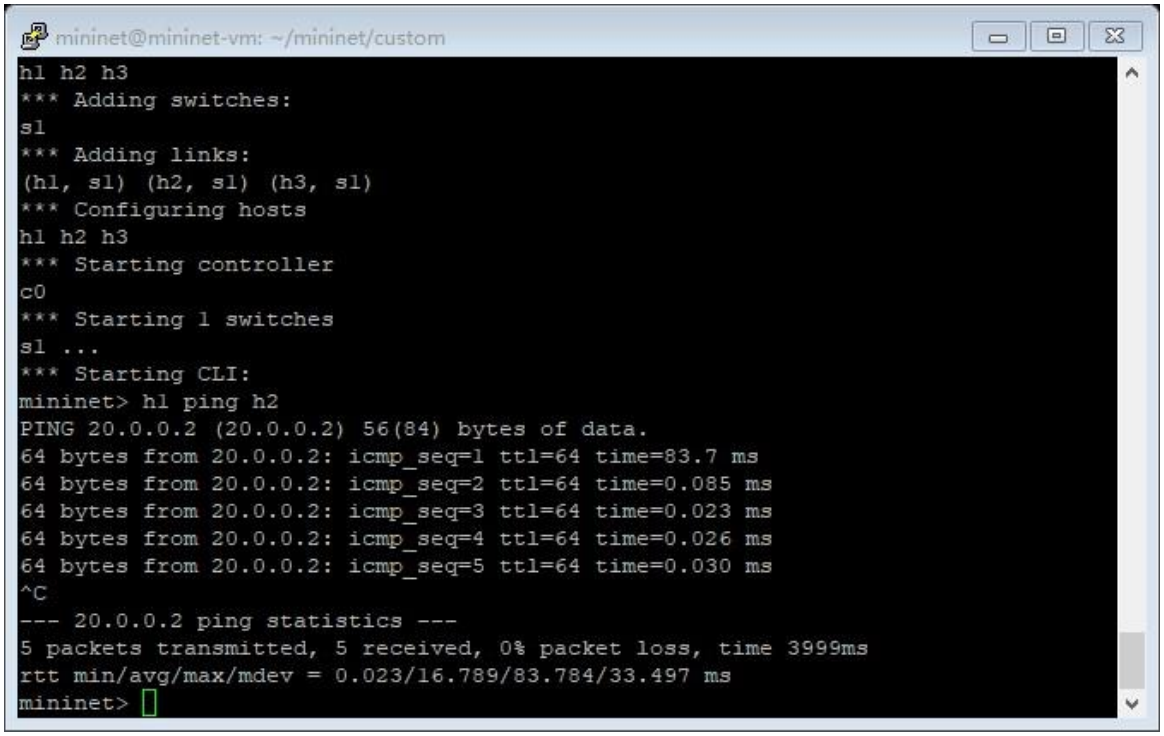
mn --custom topology_2.py --topo mytopo --mac --controller remote



```
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom topology_2.py --topo mytop
o --mac --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
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Results: h1

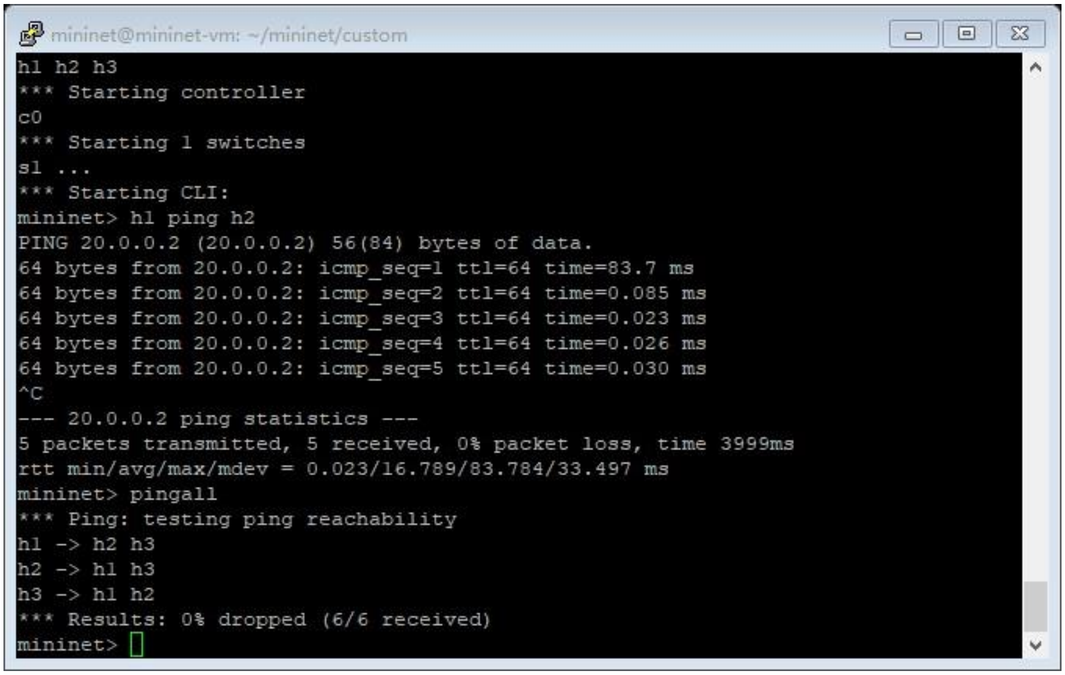
ping h2



```
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping h2
PING 20.0.0.2 (20.0.0.2) 56(84) bytes of data.
64 bytes from 20.0.0.2: icmp_seq=1 ttl=64 time=83.7 ms
64 bytes from 20.0.0.2: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 20.0.0.2: icmp_seq=3 ttl=64 time=0.023 ms
64 bytes from 20.0.0.2: icmp_seq=4 ttl=64 time=0.026 ms
64 bytes from 20.0.0.2: icmp_seq=5 ttl=64 time=0.030 ms
^C
--- 20.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.023/16.789/83.784/33.497 ms
mininet> 
```

Connection between h1 h2 and router is work

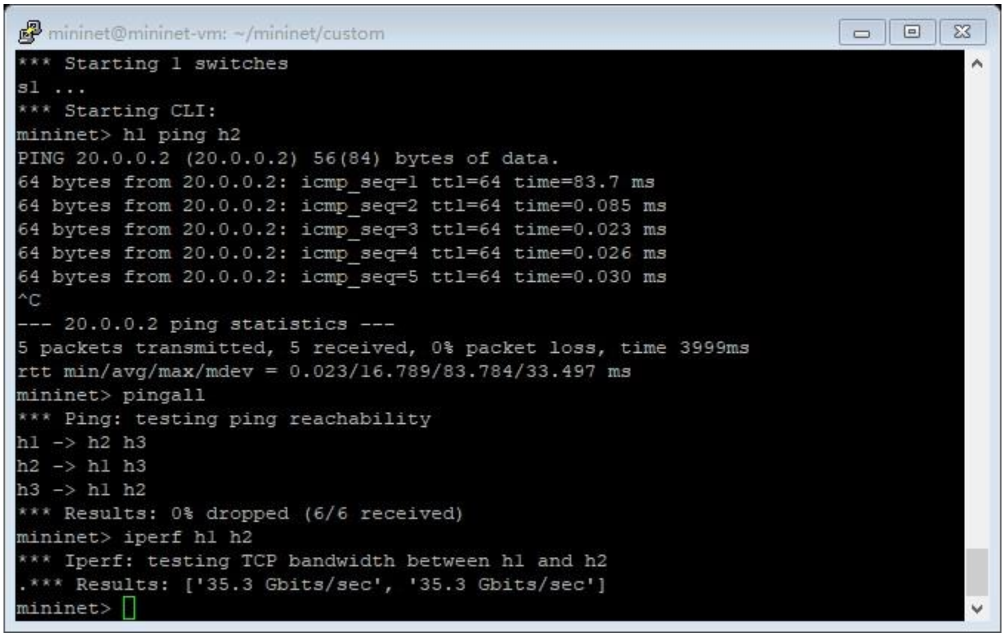
Pingall



```
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping h2
PING 20.0.0.2 (20.0.0.2) 56(84) bytes of data.
64 bytes from 20.0.0.2: icmp_seq=1 ttl=64 time=83.7 ms
64 bytes from 20.0.0.2: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 20.0.0.2: icmp_seq=3 ttl=64 time=0.023 ms
64 bytes from 20.0.0.2: icmp_seq=4 ttl=64 time=0.026 ms
64 bytes from 20.0.0.2: icmp_seq=5 ttl=64 time=0.030 ms
^C
--- 20.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.023/16.789/83.784/33.497 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> 
```

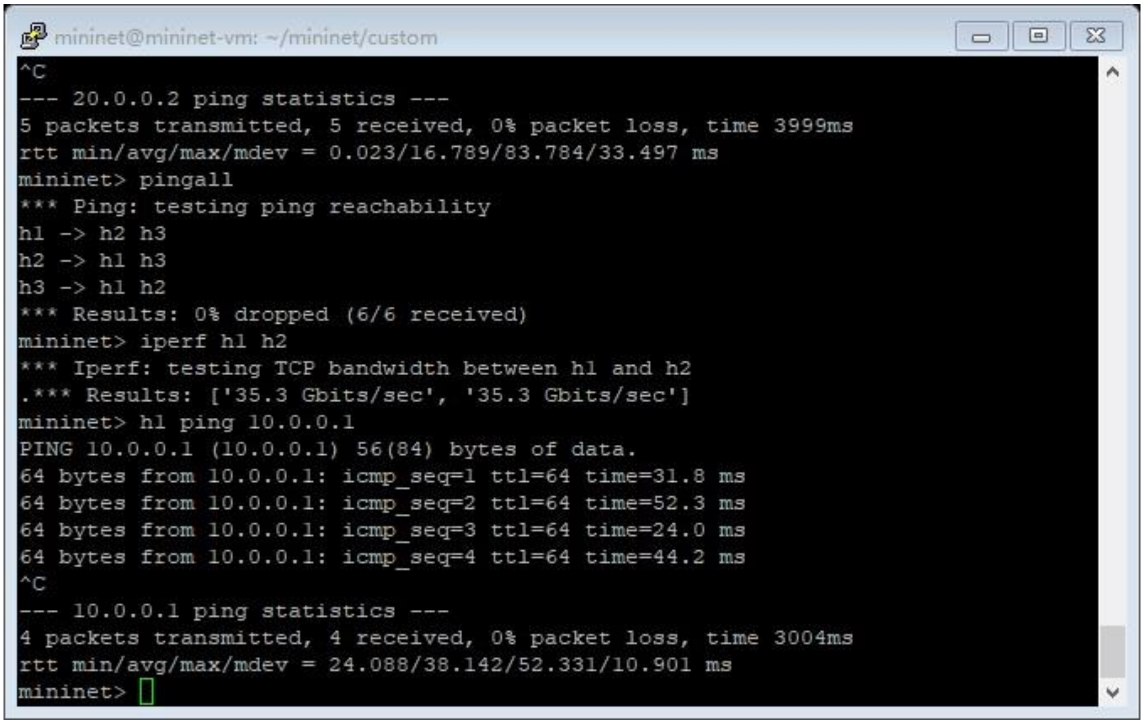
All connections work.

Iperf h1 h2



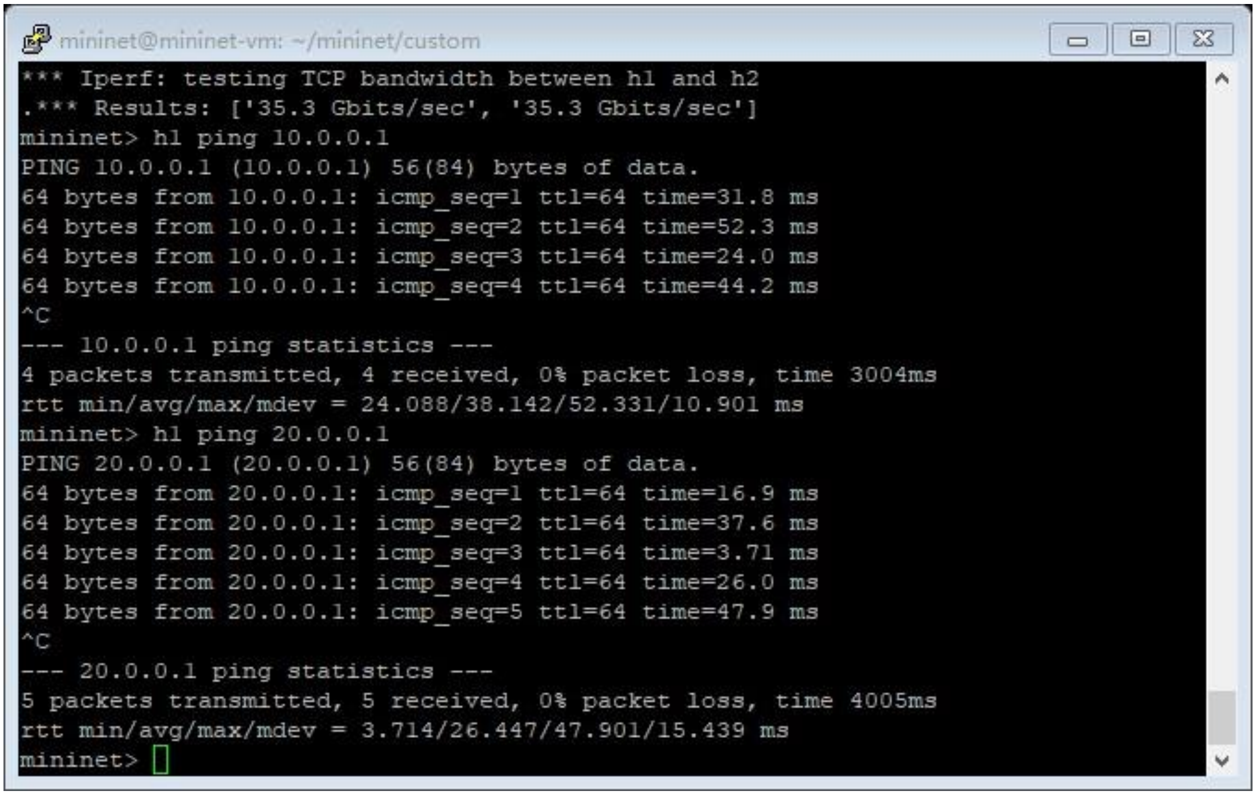
```
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping h2
PING 20.0.0.2 (20.0.0.2) 56(84) bytes of data.
64 bytes from 20.0.0.2: icmp_seq=1 ttl=64 time=83.7 ms
64 bytes from 20.0.0.2: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 20.0.0.2: icmp_seq=3 ttl=64 time=0.023 ms
64 bytes from 20.0.0.2: icmp_seq=4 ttl=64 time=0.026 ms
64 bytes from 20.0.0.2: icmp_seq=5 ttl=64 time=0.030 ms
^C
--- 20.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.023/16.789/83.784/33.497 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['35.3 Gbits/sec', '35.3 Gbits/sec']
mininet> 
```

Test tcp connection speed between h1 and h2 h1 ping 10.0.0.1



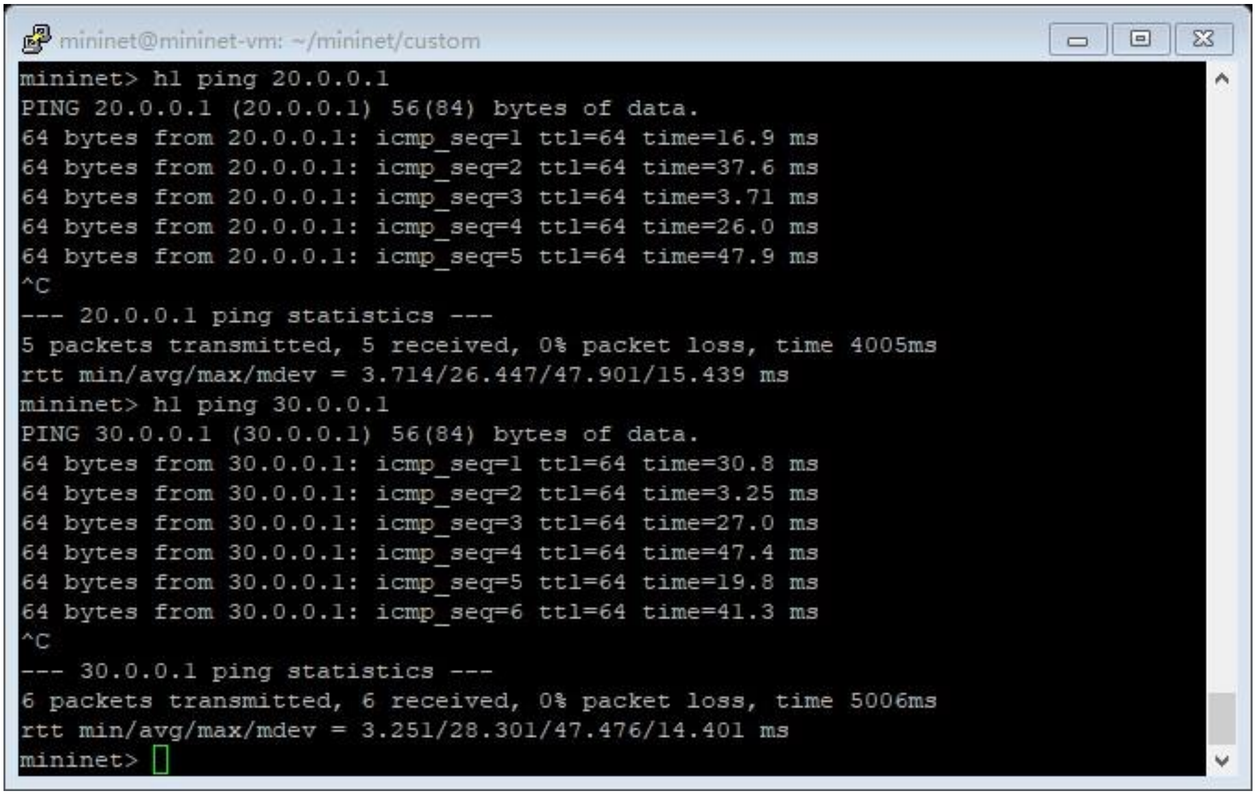
```
^C
--- 20.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.023/16.789/83.784/33.497 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['35.3 Gbits/sec', '35.3 Gbits/sec']
mininet> h1 ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=31.8 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=52.3 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=24.0 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=44.2 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 24.088/38.142/52.331/10.901 ms
mininet> 
```

h1 ping 20.0.0.1



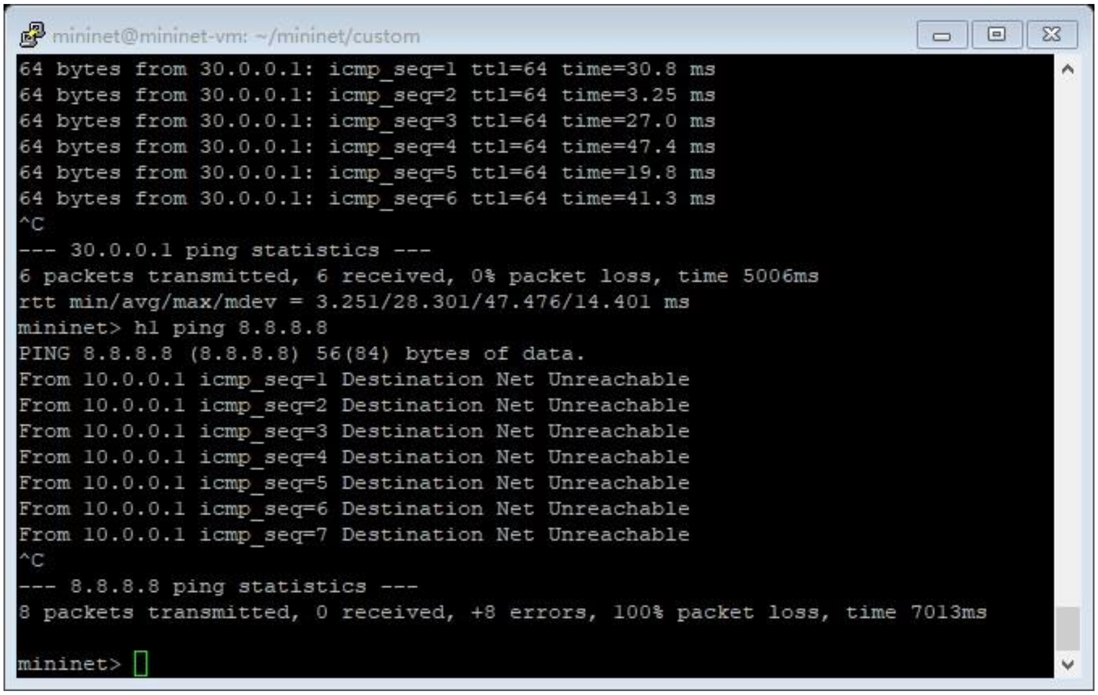
```
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['35.3 Gbits/sec', '35.3 Gbits/sec']
mininet> h1 ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=31.8 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=52.3 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=24.0 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=44.2 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 24.088/38.142/52.331/10.901 ms
mininet> h1 ping 20.0.0.1
PING 20.0.0.1 (20.0.0.1) 56(84) bytes of data.
64 bytes from 20.0.0.1: icmp_seq=1 ttl=64 time=16.9 ms
64 bytes from 20.0.0.1: icmp_seq=2 ttl=64 time=37.6 ms
64 bytes from 20.0.0.1: icmp_seq=3 ttl=64 time=3.71 ms
64 bytes from 20.0.0.1: icmp_seq=4 ttl=64 time=26.0 ms
64 bytes from 20.0.0.1: icmp_seq=5 ttl=64 time=47.9 ms
^C
--- 20.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 3.714/26.447/47.901/15.439 ms
mininet> 
```

h1 ping 30.0.0.1



```
mininet> h1 ping 20.0.0.1
PING 20.0.0.1 (20.0.0.1) 56(84) bytes of data.
64 bytes from 20.0.0.1: icmp_seq=1 ttl=64 time=16.9 ms
64 bytes from 20.0.0.1: icmp_seq=2 ttl=64 time=37.6 ms
64 bytes from 20.0.0.1: icmp_seq=3 ttl=64 time=3.71 ms
64 bytes from 20.0.0.1: icmp_seq=4 ttl=64 time=26.0 ms
64 bytes from 20.0.0.1: icmp_seq=5 ttl=64 time=47.9 ms
^C
--- 20.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 3.714/26.447/47.901/15.439 ms
mininet> h1 ping 30.0.0.1
PING 30.0.0.1 (30.0.0.1) 56(84) bytes of data.
64 bytes from 30.0.0.1: icmp_seq=1 ttl=64 time=30.8 ms
64 bytes from 30.0.0.1: icmp_seq=2 ttl=64 time=3.25 ms
64 bytes from 30.0.0.1: icmp_seq=3 ttl=64 time=27.0 ms
64 bytes from 30.0.0.1: icmp_seq=4 ttl=64 time=47.4 ms
64 bytes from 30.0.0.1: icmp_seq=5 ttl=64 time=19.8 ms
64 bytes from 30.0.0.1: icmp_seq=6 ttl=64 time=41.3 ms
^C
--- 30.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 3.251/28.301/47.476/14.401 ms
mininet> 
```

h1 ping 8.8.8.8



```
64 bytes from 30.0.0.1: icmp_seq=1 ttl=64 time=30.8 ms
64 bytes from 30.0.0.1: icmp_seq=2 ttl=64 time=3.25 ms
64 bytes from 30.0.0.1: icmp_seq=3 ttl=64 time=27.0 ms
64 bytes from 30.0.0.1: icmp_seq=4 ttl=64 time=47.4 ms
64 bytes from 30.0.0.1: icmp_seq=5 ttl=64 time=19.8 ms
64 bytes from 30.0.0.1: icmp_seq=6 ttl=64 time=41.3 ms
^C
--- 30.0.0.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5006ms
rtt min/avg/max/mdev = 3.251/28.301/47.476/14.401 ms
mininet> h1 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Net Unreachable
From 10.0.0.1 icmp_seq=2 Destination Net Unreachable
From 10.0.0.1 icmp_seq=3 Destination Net Unreachable
From 10.0.0.1 icmp_seq=4 Destination Net Unreachable
From 10.0.0.1 icmp_seq=5 Destination Net Unreachable
From 10.0.0.1 icmp_seq=6 Destination Net Unreachable
From 10.0.0.1 icmp_seq=7 Destination Net Unreachable
^C
--- 8.8.8.8 ping statistics ---
8 packets transmitted, 0 received, +8 errors, 100% packet loss, time 7013ms

mininet> 
```

There is no 8.8.8.8 inside this network. So, destination net is unreachable.

Scenario 3:

Description: There are a total of 3 routers, 5 switches 15 hosts. Each switch connected with three hosts. This part is a combination of routers and switches.

In this system, controller distinguish routers and switch with other routers and switches using dpid. router's dpid is 1,2,3 and switch's dpid is 4,5,6,7,8. So if dpid is larger than 3 we will use switch otherwise we use the router.

```
if dpid > 3:
    self.act_like_switch(packet, packet_in, dpid)
```

Switch is similar to part 1. It only cares about the mac address. It will forward the frame to the correct destination.

In this part, there are some hosts that one router cannot reach on its own. Thus, if a router needs to send an IP packet with assistance from another router, it would send the frame to the next hop router. Other part of router logical is same in Scenario 2.

Thus, code in this part mainly combines the code in Scenario 1 and Scenario 2

Related functions:

Almost the same as scenario 2

Def init_device(self) will contain all the information of all device in this system (ip address, interface, port number, mac address…) def act_like_switch (self, packet, packet_in, dpid): when dpid > 3 use this function as a switch def handle_arp(self, packet, packet_info, port_in): start arp process or deal with arp request/reply/pending

def send_frame(self, packet, frame_type, src, dst, outport): given destination mac address, frame_type and port number to send the frame with source information

def arp_packet(self, opcode, dst_mac, dst_ip, outport): arp packets with destination ip and mac address output port number and opcode

def icmp_unreach_packet(self, ip_packet, inport): When ICMP packet is unreached, send unreach packet

def handle_IP(self, packet, IP_packet, port_in): handle regular IP packets def

handle_ICMP(self,IP_packet, port_in, packet): handle ICMP packets

Router.py: class for all information needed by a router (port, ip, mac, what is the next hop router)

Switch.py: class for all information needed by a switch(mac)

Command:

Step 1: Start Mininet VM

Step 2: Open two terminals on your local computer (we use Putty as terminal)

Step 3: In first terminal we copy Controller_3.py, router.py, switch.py to /home/mininet/pox/pox/misc and then back to /home/mininet/pox , under this folder we can see pox.py

Step 4: In the second terminal we copy topology_3 to /home/mininet/mininet/custom

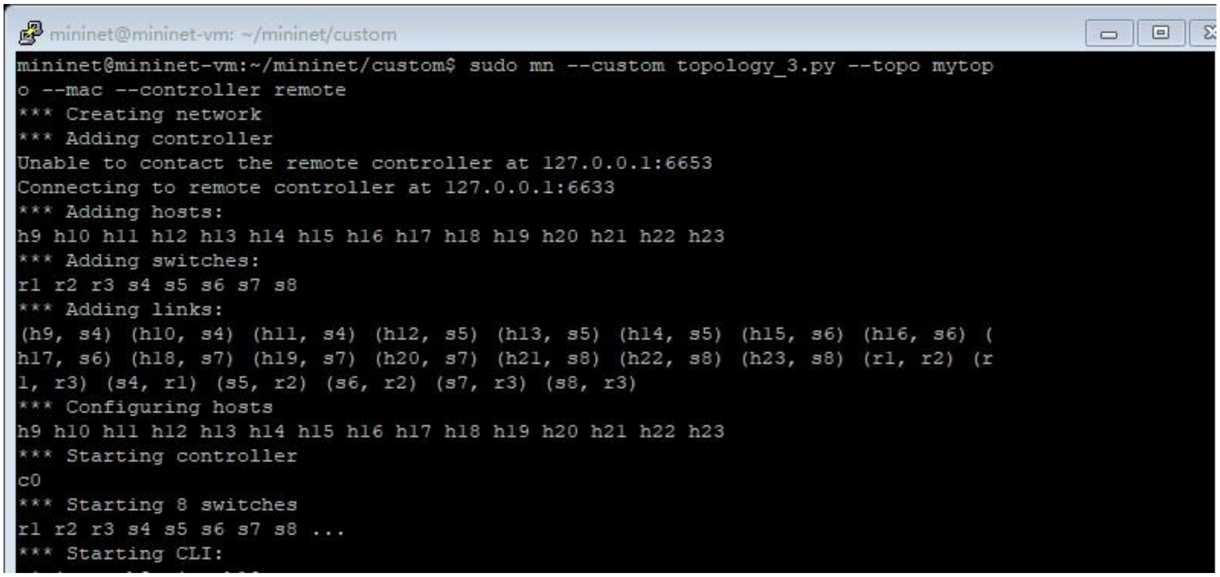
Step 5: We ran command sudo mn -c to clear all controller

Step 6: In the first terminal, we start controller_3 with command line :

./pox.py log.level --DEBUG misc.controller_3

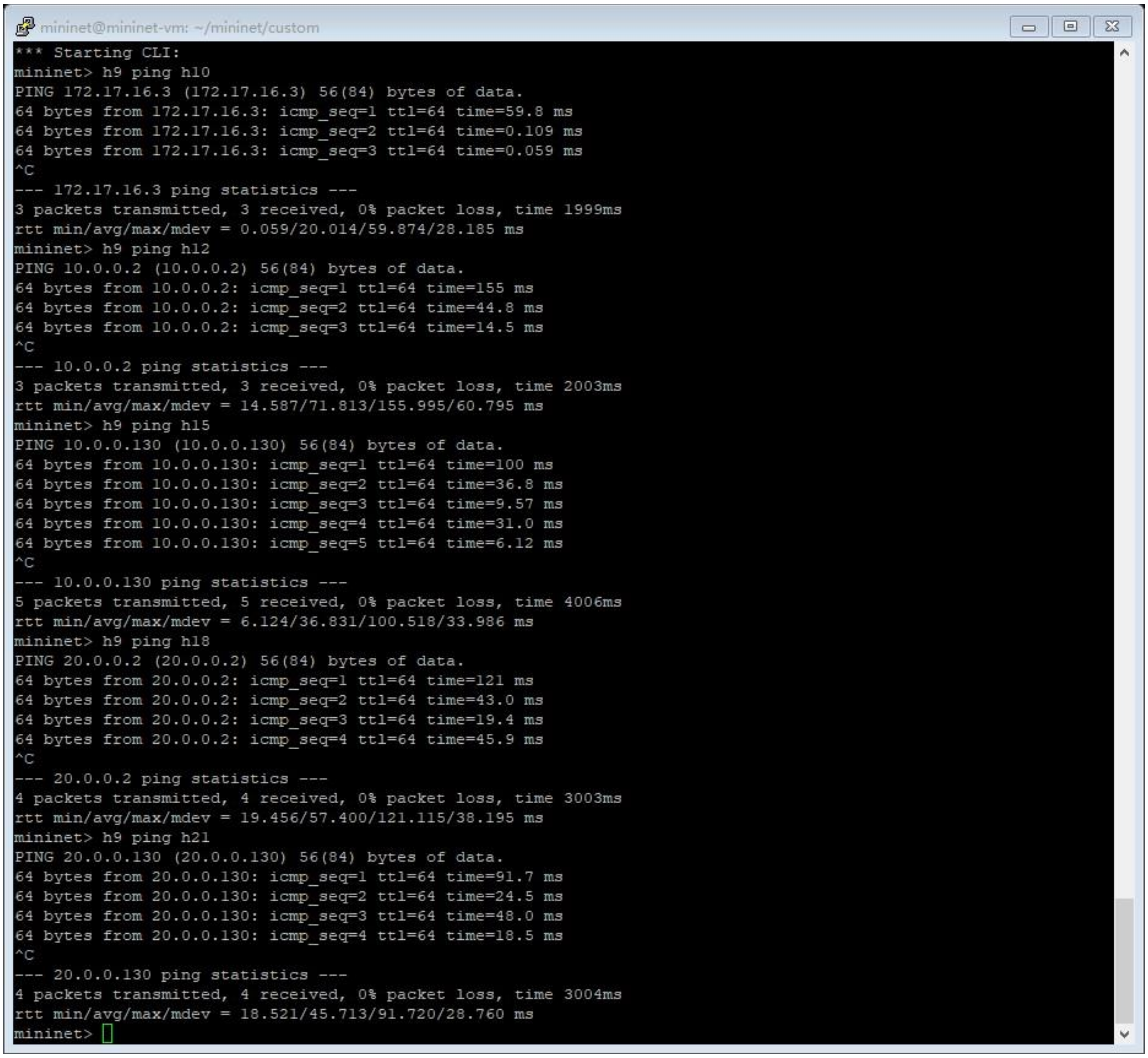
Step 7:In the second terminal, we load the topology_3 with command line: sudo

mn --custom topology_3.py --topo mytopo --mac --controller remote



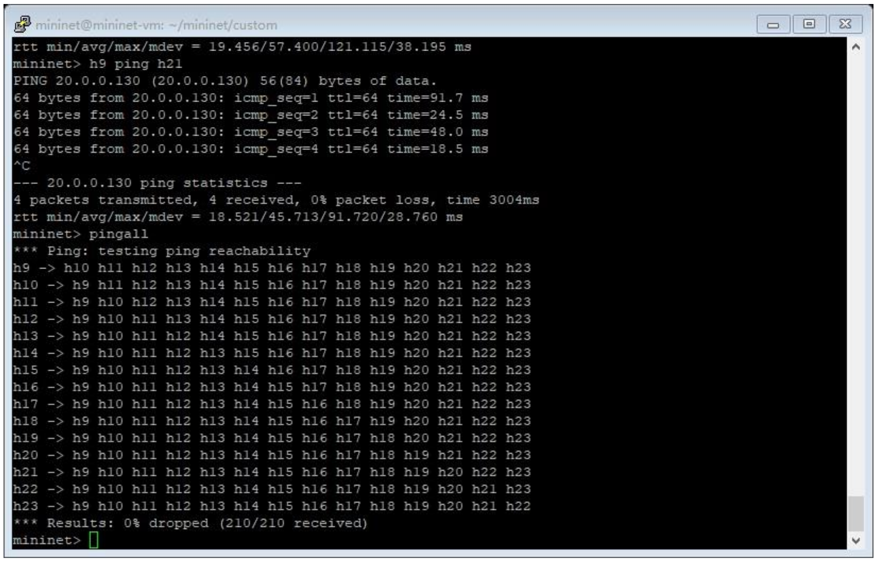
```
mininet@mininet-vm:~/mininet/custom$ sudo mn --custom topology_3.py --topo mytop
o --mac --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
*** Adding switches:
r1 r2 r3 s4 s5 s6 s7 s8
*** Adding links:
(h9, s4) (h10, s4) (h11, s4) (h12, s5) (h13, s5) (h14, s5) (h15, s6) (h16, s6) (
h17, s6) (h18, s7) (h19, s7) (h20, s7) (h21, s8) (h22, s8) (h23, s8) (r1, r2) (r
1, r3) (s4, r1) (s5, r2) (s6, r2) (s7, r3) (s8, r3)
*** Configuring hosts
h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
*** Starting controller
c0
*** Starting 8 switches
r1 r2 r3 s4 s5 s6 s7 s8 ...
*** Starting CLI:
```

Results: h9 ping h10, h12, h15, h18, h21



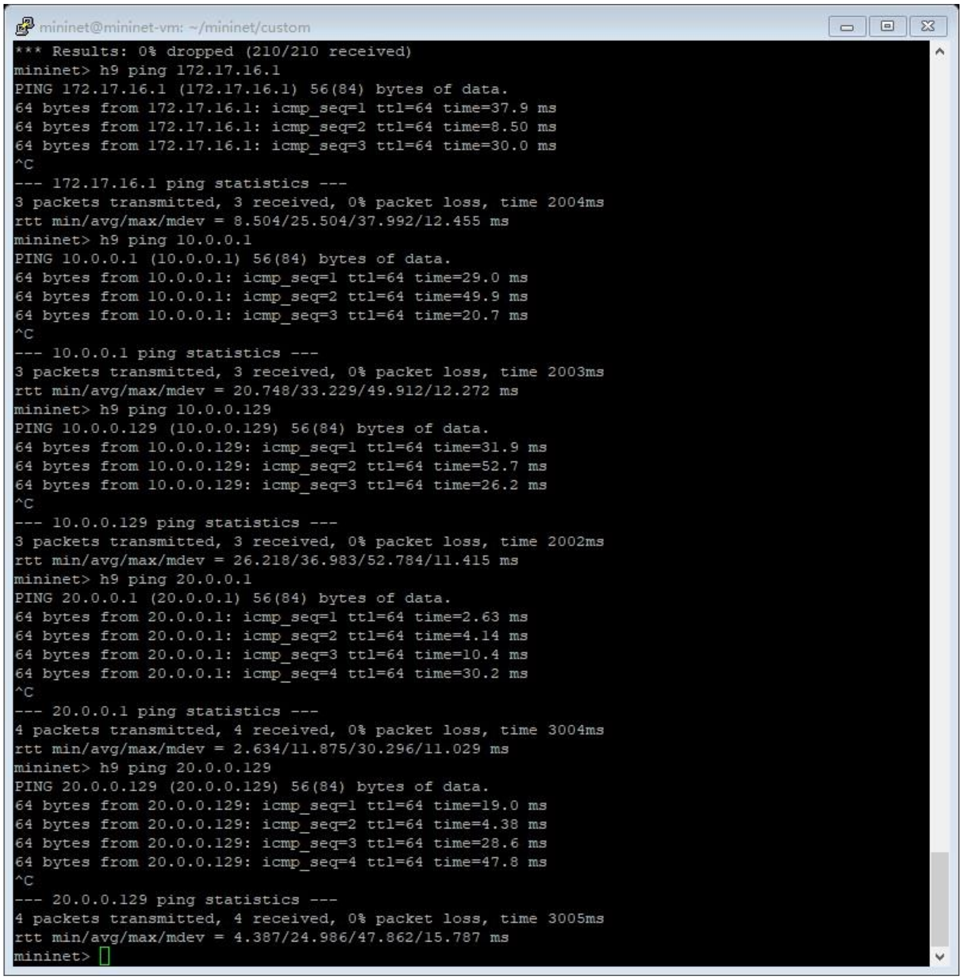
```
*** Starting CLI:
mininet> h9 ping h10
PING 172.17.16.3 (172.17.16.3) 56(84) bytes of data.
64 bytes from 172.17.16.3: icmp_seq=1 ttl=64 time=59.8 ms
64 bytes from 172.17.16.3: icmp_seq=2 ttl=64 time=0.109 ms
64 bytes from 172.17.16.3: icmp_seq=3 ttl=64 time=0.059 ms
^C
--- 172.17.16.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.059/20.014/59.874/28.185 ms
mininet> h9 ping h12
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=155 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=44.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=14.5 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 14.587/71.813/155.995/60.795 ms
mininet> h9 ping h15
PING 10.0.0.130 (10.0.0.130) 56(84) bytes of data.
64 bytes from 10.0.0.130: icmp_seq=1 ttl=64 time=100 ms
64 bytes from 10.0.0.130: icmp_seq=2 ttl=64 time=36.8 ms
64 bytes from 10.0.0.130: icmp_seq=3 ttl=64 time=9.57 ms
64 bytes from 10.0.0.130: icmp_seq=4 ttl=64 time=31.0 ms
64 bytes from 10.0.0.130: icmp_seq=5 ttl=64 time=6.12 ms
^C
--- 10.0.0.130 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 6.124/36.831/100.518/33.986 ms
mininet> h9 ping h18
PING 20.0.0.2 (20.0.0.2) 56(84) bytes of data.
64 bytes from 20.0.0.2: icmp_seq=1 ttl=64 time=121 ms
64 bytes from 20.0.0.2: icmp_seq=2 ttl=64 time=43.0 ms
64 bytes from 20.0.0.2: icmp_seq=3 ttl=64 time=19.4 ms
64 bytes from 20.0.0.2: icmp_seq=4 ttl=64 time=45.9 ms
^C
--- 20.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 19.456/57.400/121.115/38.195 ms
mininet> h9 ping h21
PING 20.0.0.130 (20.0.0.130) 56(84) bytes of data.
64 bytes from 20.0.0.130: icmp_seq=1 ttl=64 time=91.7 ms
64 bytes from 20.0.0.130: icmp_seq=2 ttl=64 time=24.5 ms
64 bytes from 20.0.0.130: icmp_seq=3 ttl=64 time=48.0 ms
64 bytes from 20.0.0.130: icmp_seq=4 ttl=64 time=18.5 ms
^C
--- 20.0.0.130 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 18.521/45.713/91.720/28.760 ms
mininet> 
```

Pingall



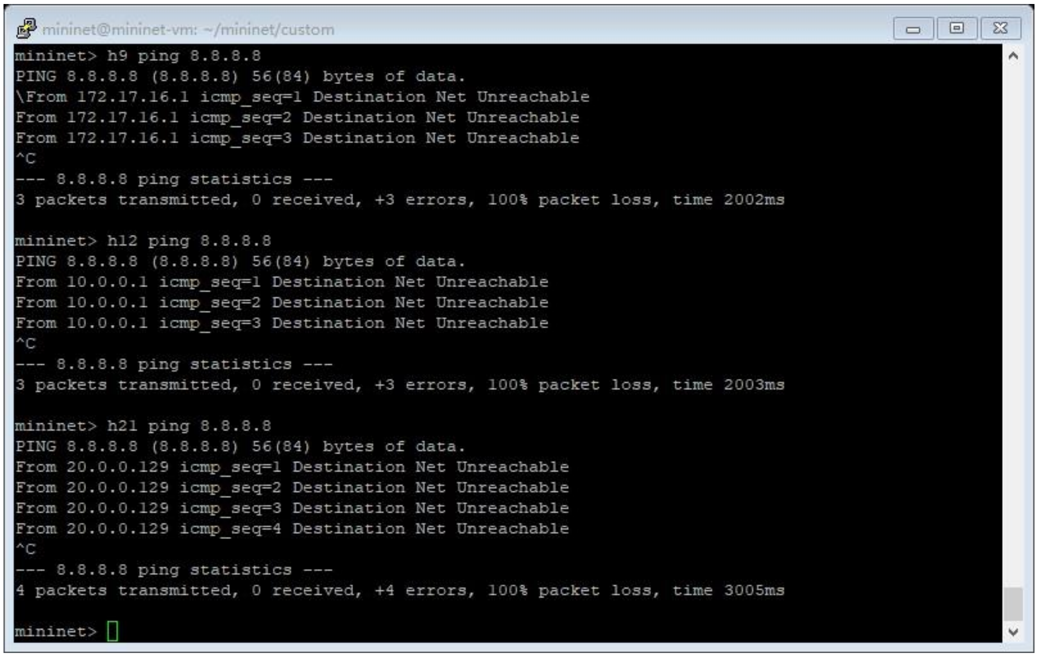
```
rtt min/avg/max/mdev = 19.456/57.400/121.115/38.195 ms
mininet> h9 ping h21
PING 20.0.0.130 (20.0.0.130) 56(84) bytes of data.
64 bytes from 20.0.0.130: icmp_seq=1 ttl=64 time=91.7 ms
64 bytes from 20.0.0.130: icmp_seq=2 ttl=64 time=24.5 ms
64 bytes from 20.0.0.130: icmp_seq=3 ttl=64 time=48.0 ms
64 bytes from 20.0.0.130: icmp_seq=4 ttl=64 time=18.5 ms
^C
--- 20.0.0.130 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 18.521/45.713/91.720/28.760 ms
mininet> pingall
*** Ping: testing ping reachability
h9 -> h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
h10 -> h9 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
h11 -> h9 h10 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
h12 -> h9 h10 h11 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
h13 -> h9 h10 h11 h12 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23
h14 -> h9 h10 h11 h12 h13 h15 h16 h17 h18 h19 h20 h21 h22 h23
h15 -> h9 h10 h11 h12 h13 h14 h16 h17 h18 h19 h20 h21 h22 h23
h16 -> h9 h10 h11 h12 h13 h14 h15 h17 h18 h19 h20 h21 h22 h23
h17 -> h9 h10 h11 h12 h13 h14 h15 h16 h18 h19 h20 h21 h22 h23
h18 -> h9 h10 h11 h12 h13 h14 h15 h16 h17 h19 h20 h21 h22 h23
h19 -> h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h20 h21 h22 h23
h20 -> h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h21 h22 h23
h21 -> h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h22 h23
h22 -> h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h23
h23 -> h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
*** Results: 0% dropped (210/210 received)
mininet> 
```

h9 ping 172.17.16.1, 10.0.0.1, 10.0.0.129, 20.0.0.1, 20.0.0.129



```
*** Results: 0% dropped (210/210 received)
mininet> h9 ping 172.17.16.1
PING 172.17.16.1 (172.17.16.1) 56(84) bytes of data.
64 bytes from 172.17.16.1: icmp_seq=1 ttl=64 time=37.9 ms
64 bytes from 172.17.16.1: icmp_seq=2 ttl=64 time=8.50 ms
64 bytes from 172.17.16.1: icmp_seq=3 ttl=64 time=30.0 ms
^C
--- 172.17.16.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 8.504/25.504/37.992/12.455 ms
mininet> h9 ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=29.0 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=49.9 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=20.7 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 20.748/33.229/49.912/12.272 ms
mininet> h9 ping 10.0.0.129
PING 10.0.0.129 (10.0.0.129) 56(84) bytes of data.
64 bytes from 10.0.0.129: icmp_seq=1 ttl=64 time=31.9 ms
64 bytes from 10.0.0.129: icmp_seq=2 ttl=64 time=52.7 ms
64 bytes from 10.0.0.129: icmp_seq=3 ttl=64 time=26.2 ms
^C
--- 10.0.0.129 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 26.218/36.983/52.784/11.415 ms
mininet> h9 ping 20.0.0.1
PING 20.0.0.1 (20.0.0.1) 56(84) bytes of data.
64 bytes from 20.0.0.1: icmp_seq=1 ttl=64 time=2.63 ms
64 bytes from 20.0.0.1: icmp_seq=2 ttl=64 time=4.14 ms
64 bytes from 20.0.0.1: icmp_seq=3 ttl=64 time=10.4 ms
64 bytes from 20.0.0.1: icmp_seq=4 ttl=64 time=30.2 ms
^C
--- 20.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 2.634/11.875/30.296/11.029 ms
mininet> h9 ping 20.0.0.129
PING 20.0.0.129 (20.0.0.129) 56(84) bytes of data.
64 bytes from 20.0.0.129: icmp_seq=1 ttl=64 time=19.0 ms
64 bytes from 20.0.0.129: icmp_seq=2 ttl=64 time=4.38 ms
64 bytes from 20.0.0.129: icmp_seq=3 ttl=64 time=28.6 ms
64 bytes from 20.0.0.129: icmp_seq=4 ttl=64 time=47.8 ms
^C
--- 20.0.0.129 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 4.387/24.986/47.862/15.787 ms
mininet> 
```

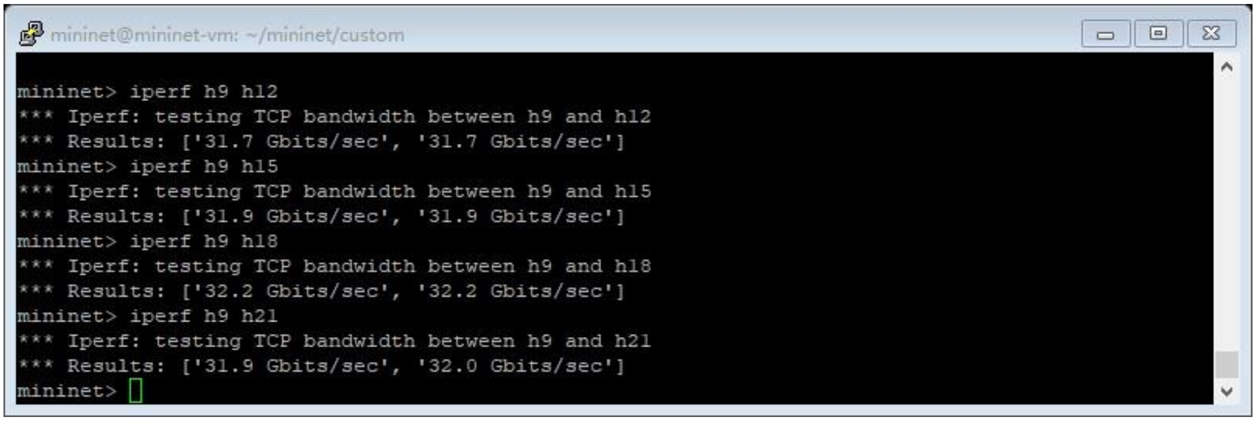
h9, h12, h21 ping 8.8.8.8



```
mininet> h9 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
\From 172.17.16.1 icmp_seq=1 Destination Net Unreachable
From 172.17.16.1 icmp_seq=2 Destination Net Unreachable
From 172.17.16.1 icmp_seq=3 Destination Net Unreachable
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2002ms

mininet> h12 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Net Unreachable
From 10.0.0.1 icmp_seq=2 Destination Net Unreachable
From 10.0.0.1 icmp_seq=3 Destination Net Unreachable
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2003ms

mininet> h21 ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 20.0.0.129 icmp_seq=1 Destination Net Unreachable
From 20.0.0.129 icmp_seq=2 Destination Net Unreachable
From 20.0.0.129 icmp_seq=3 Destination Net Unreachable
From 20.0.0.129 icmp_seq=4 Destination Net Unreachable
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3005ms

mininet> 
```

Because there is no 8.8.8.8 so all host should not reach 8.8.8.8

iperf h9 h12 iperf h9 h15 iperf h9 h18 iperf h9 h21



```
mininet> iperf h9 h12
*** Iperf: testing TCP bandwidth between h9 and h12
*** Results: ['31.7 Gbits/sec', '31.7 Gbits/sec']
mininet> iperf h9 h15
*** Iperf: testing TCP bandwidth between h9 and h15
*** Results: ['31.9 Gbits/sec', '31.9 Gbits/sec']
mininet> iperf h9 h18
*** Iperf: testing TCP bandwidth between h9 and h18
*** Results: ['32.2 Gbits/sec', '32.2 Gbits/sec']
mininet> iperf h9 h21
*** Iperf: testing TCP bandwidth between h9 and h21
*** Results: ['31.9 Gbits/sec', '32.0 Gbits/sec']
mininet> 
```

Speed of tcp connection between h9 and other hosts

Reference：

1.https://noxrepo.github.io/pox-doc/html/#forwarding-l3-learning
2.https://blog.csdn.net/qq_15437629/article/details/45950633
3.https://openflow.stanford.edu/display/ONL/POX+Wiki.html#POXWiki-Workingwithpackets%3Apox.lib.packet
4.https://github.com/suzhou1898/Switch-Router-with-OpenFlow-