EE 555
Broadband Network Architectures
Project

Wenhan Tang 1210637460
Chenguang He 8977866134

Abstract

In this project, we will use python try to implement a router. This project contains two sections. The first section is create learning switch, we will replace a hub with switch by using python to implement that switch, then we need to create a network with one router and three host and implement a router to make this network can transmit message between each host. In the second section, we need to implement a network with two routers.

Process

Firstly we should read the tutorial in https://github.com/mininet/openflow-tutorial/wiki. After finishing Installing Required Software, Set up Virtual Machine, VirtualBox Specific Instructions, Learning Development Tools, we started our procedures at step 5: Create a Learning Switch

Part 1: Routing Experience
Firstly, on an SSH terminal, we used:

$ sudo killall controller

To kill any running controller program. Then we need to open mininet terminal, run:

$ sudo mn -c
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote

To initiate the topology. On a separate SSH terminal, run

*$./pox.py log.level --DEBUG misc.of_tutoria*l

To enable the controller. In the mininet terminal,

mininet>xterm h1 h2 h3

Is used to open three hosts windows. In h2 label, run:

 # tcpdump -XX -n -i h2-eth0

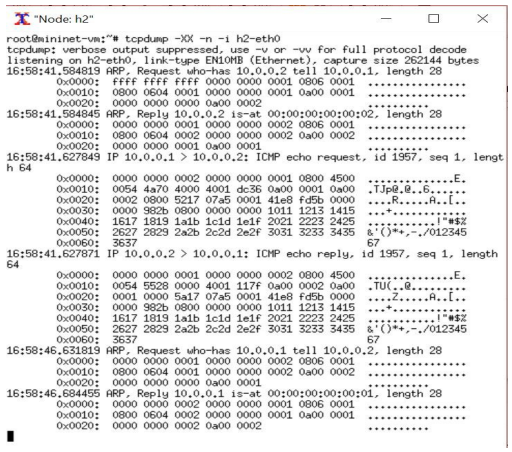
In h3 label, run:

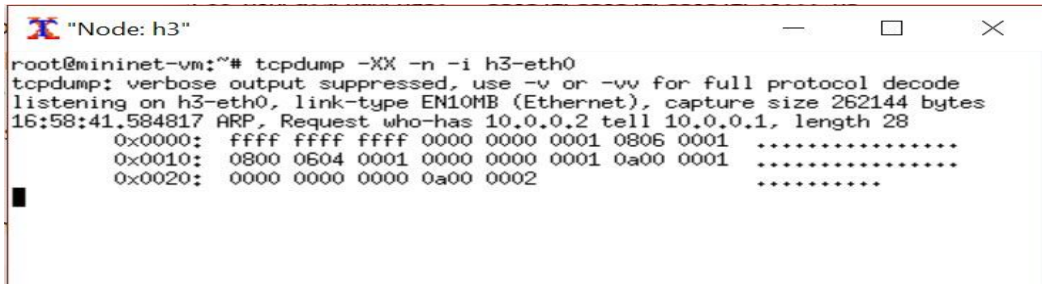
 # tcpdump -XX -n -i h3-eth0

Finally, in h1 label, run:

 # ping -c1 10.0.0.2

We get

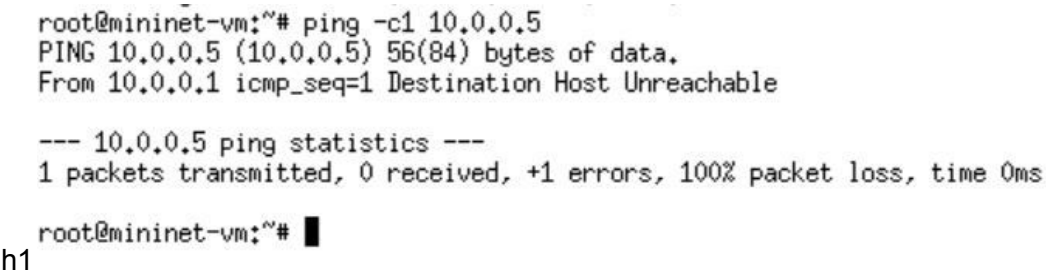




Which shows ping transmission. Then

ping -c1 10.0.0.5

We get



h1

```
17:00:19.940099 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:00:20.921214 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:00:21.905361 ARP, Request who-has 10.0.0.5 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 0005                 ..........
```

h2

```
16:58:41.584817 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                 ..........
17:00:19.940097 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:00:20.921210 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:00:21.905357 ARP, Request who-has 10.0.0.5 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 0005                 ..........
```

h3

When go to mininet terminal and enter

pingall

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

When enter

iperf

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['6.08 Mbits/sec', '7.33 Mbits/sec']
```

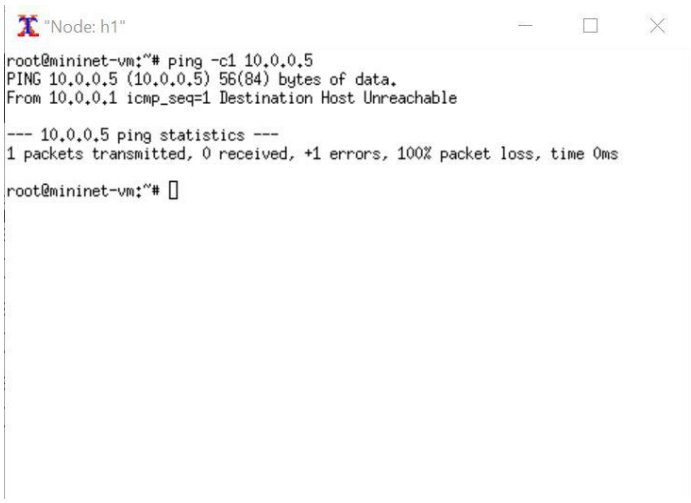
It is significantly faster than reference controller.

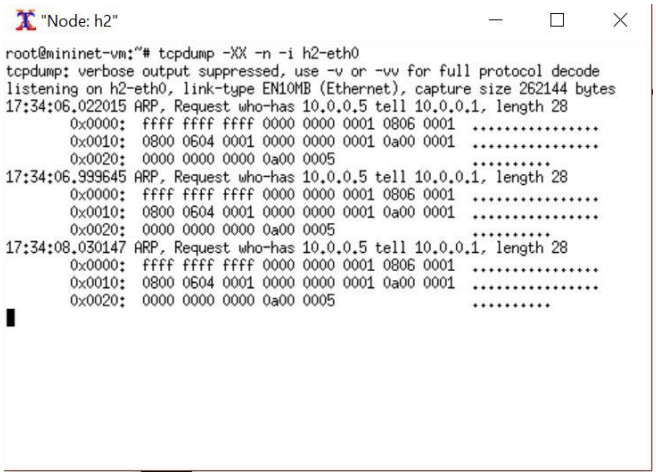
Now we tried to modify of_tutorial.py and get the following result:
In one terminal, we entered

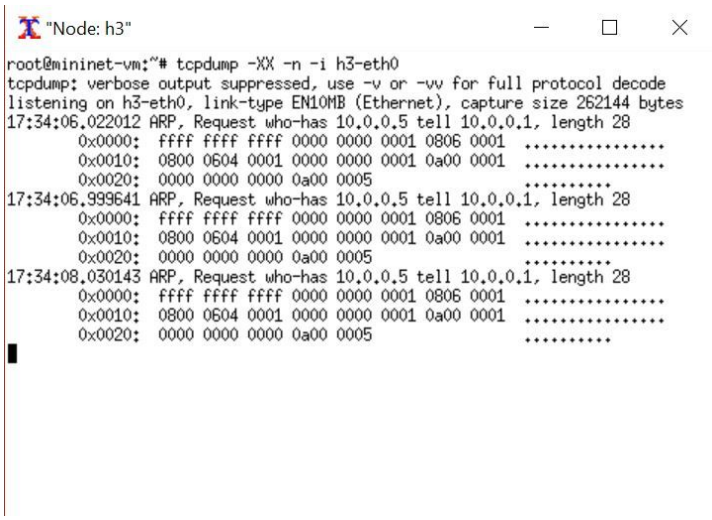
$./pox.py log.level --DEBUG misc.of_tutorial

In mininet terminal, we entered
$ sudo mn -c
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
mininet> xterm h1 h2 h3

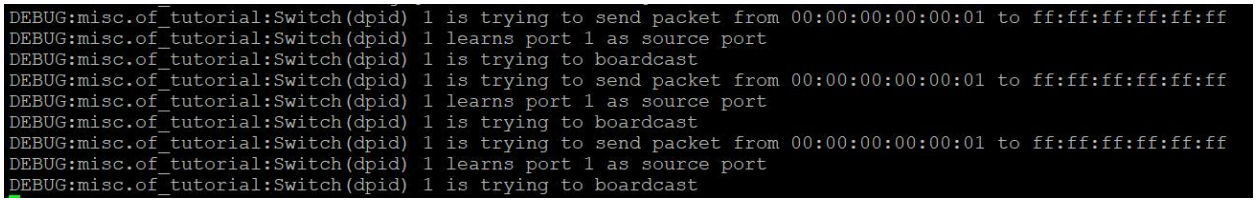
First when we ping 10.0.0.5:







In the SSH terminal we have:



Then we ping 10.0.0.2:

"Node: h2"

```
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
17:36:26.812462 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:27.790954 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:28.822215 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:57.465598 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                 ..........
17:36:57.465619 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                 ..........
17:36:57.507635 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 3072, seq 1, lengt
h 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 6d8d 4000 4001 b919 0a00 0001 0a00  .Tm.@.@.........
        0x0020:  0002 0800 5c14 0c00 0001 39f1 fd5b 0000  ....\.....9..[..
        0x0030:  0000 93ca 0600 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
17:36:57.507657 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 3072, seq 1, length
64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 91ec 0000 4001 d4ba 0a00 0002 0a00  .T....@.........
        0x0020:  0001 0000 6414 0c00 0001 39f1 fd5b 0000  ....d.....9..[..
        0x0030:  0000 93ca 0600 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
17:37:02.520026 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                 ..........
17:37:02.545824 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                 ..........
```

"Node: h3"

```
root@mininet-vm:~# tcpdump -XX -n -i h3-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h3-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:36:26.812460 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:27.790950 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:28.822212 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:57.465596 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                 ..........
```

In the SSH terminal we have:

```
DEBUG:misc.of_tutorial:Switch(dpid) 1 is trying to send packet from 00:00:00:00:00:02 to 00:00:00:00:00:01
DEBUG:misc.of_tutorial:Switch(dpid) 1 learns port 2 as source port
DEBUG:misc.of_tutorial:Switch(dpid) 1 is sending packet to port 1
DEBUG:misc.of_tutorial:Switch(dpid) 1 is adding flow to the table, dst ip 00:00:00:00:00:01 & port 1
DEBUG:misc.of_tutorial:Switch(dpid) 1 is trying to send packet from 00:00:00:00:00:01 to 00:00:00:00:00:02
DEBUG:misc.of_tutorial:Switch(dpid) 1 learns port 1 as source port
DEBUG:misc.of_tutorial:Switch(dpid) 1 is sending packet to port 2
DEBUG:misc.of_tutorial:Switch(dpid) 1 is adding flow to the table, dst ip 00:00:00:00:00:02 & port 2
DEBUG:misc.of_tutorial:Switch(dpid) 1 is trying to send packet from 00:00:00:00:00:02 to 00:00:00:00:00:01
DEBUG:misc.of_tutorial:Switch(dpid) 1 learns port 2 as source port
DEBUG:misc.of_tutorial:Switch(dpid) 1 is sending packet to port 1
DEBUG:misc.of_tutorial:Switch(dpid) 1 is trying to send packet from 00:00:00:00:00:02 to 00:00:00:00:00:01
DEBUG:misc.of_tutorial:Switch(dpid) 1 learns port 2 as source port
DEBUG:misc.of_tutorial:Switch(dpid) 1 is sending packet to port 1
DEBUG:misc.of_tutorial:Switch(dpid) 1 is trying to send packet from 00:00:00:00:00:01 to 00:00:00:00:00:02
DEBUG:misc.of_tutorial:Switch(dpid) 1 learns port 1 as source port
DEBUG:misc.of_tutorial:Switch(dpid) 1 is sending packet to port 2
```

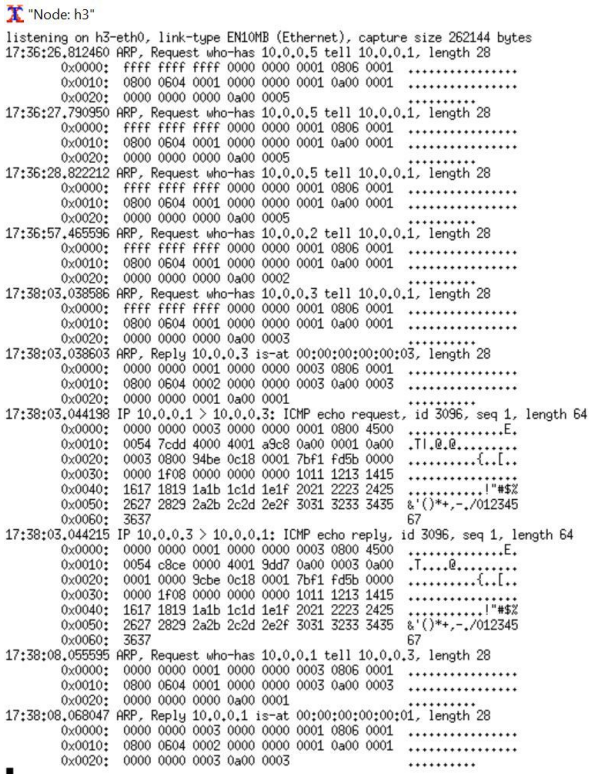
Lastly we ping 10.0.0.3:

"Node: h1"

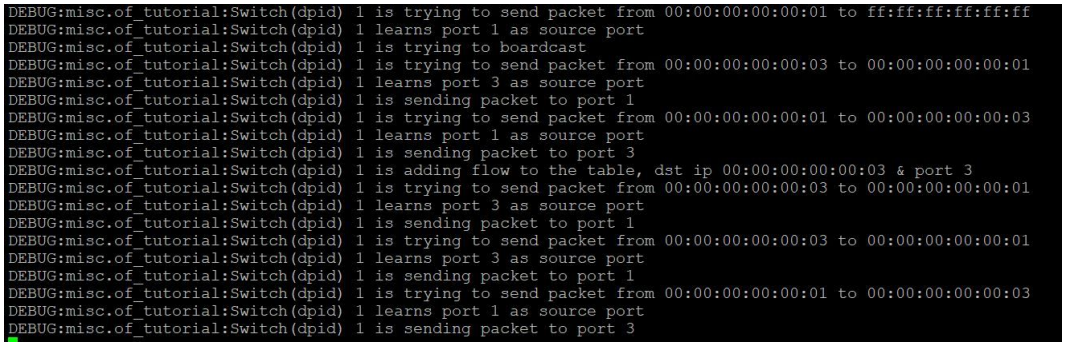
```
root@mininet-vm:~# ping -c1 10.0.0.5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.5 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@mininet-vm:~# ping -c1 10.0.0.2
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=95.0 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 95.053/95.053/95.053/0.000 ms
root@mininet-vm:~# ping -c1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=81.5 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 81.510/81.510/81.510/0.000 ms
root@mininet-vm:~#
```

"Node: h2"

```
        0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0000 0a00 0005                 ..........
17:36:27.790954 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:28.822215 ARP, Request who-has 10.0.0.5 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 0005                 ..........
17:36:57.465598 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                 ..........
17:36:57.465619 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                 ..........
17:36:57.507635 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 3072, seq 1, lengt
h 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 6d8d 4000 4001 b919 0a00 0001 0a00  .Tm.@.@.........
        0x0020:  0002 0800 5c14 0c00 0001 39f1 fd5b 0000  ....\.....9..[..
        0x0030:  0000 93ca 0600 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ..........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
17:36:57.507657 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 3072, seq 1, length
64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 91ec 0000 4001 d4ba 0a00 0002 0a00  .T....@.........
        0x0020:  0001 0000 6414 0c00 0001 39f1 fd5b 0000  ....d.....9..[..
        0x0030:  0000 93ca 0600 0000 0000 1011 1213 1415  ................
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ..........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
17:37:02.520026 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                 ..........
17:37:02.545824 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                 ..........
17:38:03.038588 ARP, Request who-has 10.0.0.3 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 0003                 ..........
```



In the SSH terminal we have:



We used the above procedures to test our of_tutorial.py. If h1 is sending packets to hosts that not in the network, h2 and h3 will get broadcast arp request for three times. If h1 is sending packets to h2 for the first time, h2 will arp reply and begin to communicate with h1, but h3 will only receive an arp request and drop it. If h1 is sending packets to h3 for the first time, h3 will respond and h2 will drop the packet. When the switch learnt above, next time when h1 sends packets to h2, it will only forward to h2 so that h3 will not hear it.

6. Router Exercise

In this part, we first create a topology modified from topo-2sw-2host.py. Our file is called part1_topo.py. We changed the topology to 3 hosts and 1 switch with configured subnet, IP, gateway, netmask. Then we wrote router.py(used

https://github.com/zhan849/ee555/blob/master/part1/router.py as reference), which is the router file needed for this part. To test it, open two terminals. In the first terminal,

$ cd pox/
$ ./pox.py log.level --DEBUG misc.router misc.full_payload

In the second terminal:

$ sudo mn -c
$ sudo mn --custom part1_topo.py --topo Part1_Topo --mac --controller=remote,ip=127.0.0.1
mininet>pingall

The result is as follow:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['30.9 Gbits/sec', '31.0 Gbits/sec']
```

This proves that our topology works for router.py

When we implement the Router, we run the following command

./pox.py log.level --DEBUG misc.simple_router misc.full_payload

We can see the following message in pox terminal

```
mininet@mininet-vm:~/pox$ ./pox.py log.level --DEBUG misc.router misc.full_payload
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
DEBUG:misc.router:router registered
INFO:misc.full_payload:Requesting full packet payloads
DEBUG:core:POX 0.1.0 (betta) going up...
DEBUG:core:Running on CPython (2.7.6/Oct 26 2016 20:32:47)
DEBUG:core:Platform is Linux-4.2.0-27-generic-i686-with-Ubuntu-14.04-trusty
INFO:core:POX 0.1.0 (betta) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
```

This means the router work properly

Then in another SSH terminal, we input the following command:
$ sudo mn -c

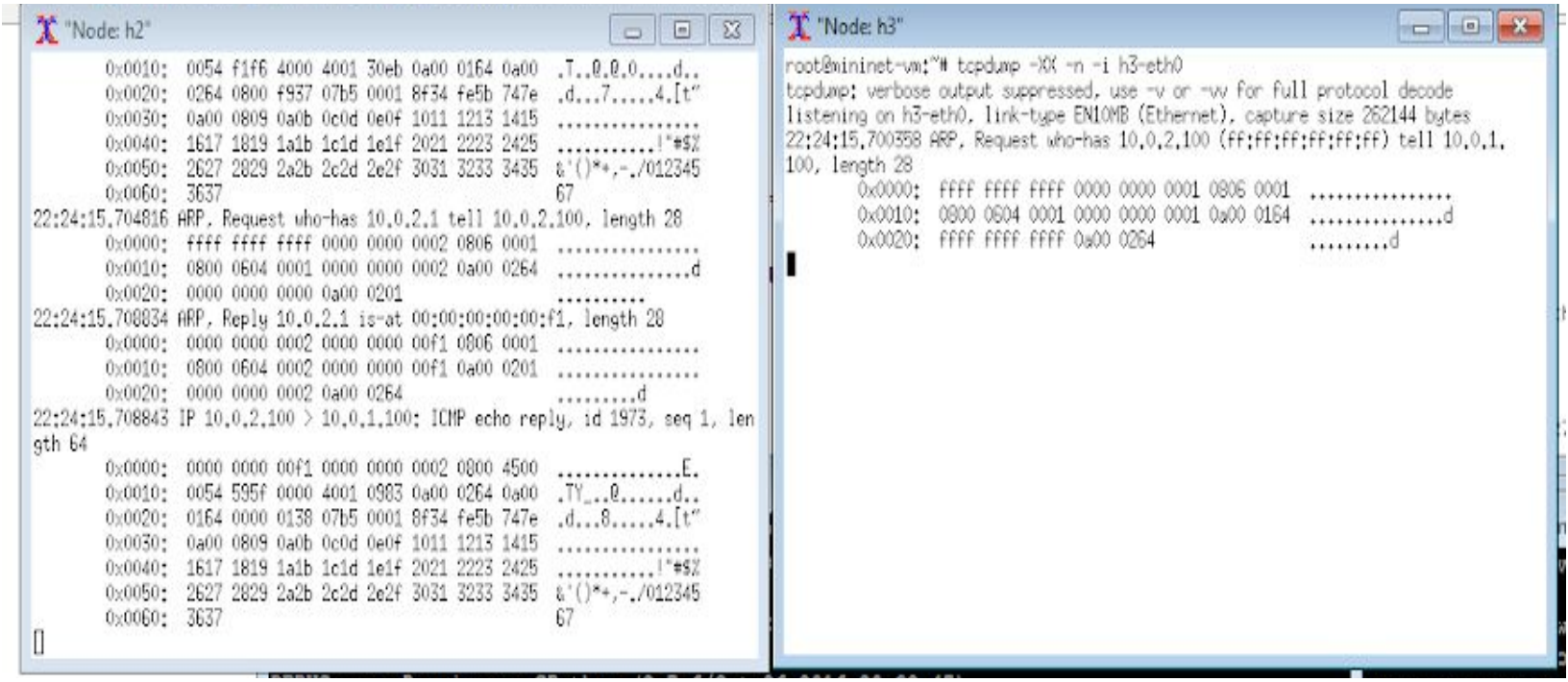
$ sudo mn --custom part1_topo.py --topo Part1_Topo --mac --controller=remote,ip=127.0.0.1
mininet>pingall

```
mininet@mininet-vm:~$ sudo mn --custom part1_topo.py --topo Part1_Topo --mac --controller=remo
*** 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> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet>
```

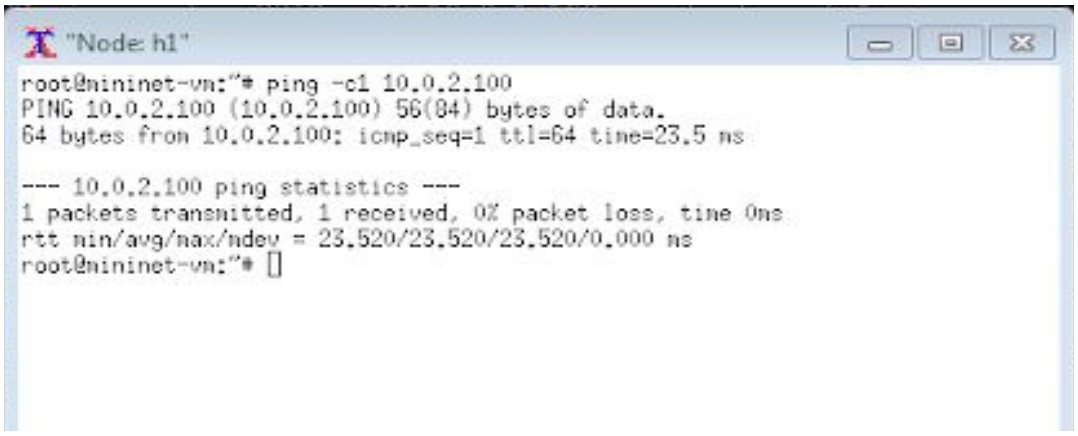
We can see this network has 3 hosts and 1 router
Next, firstly we test pingall

```
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

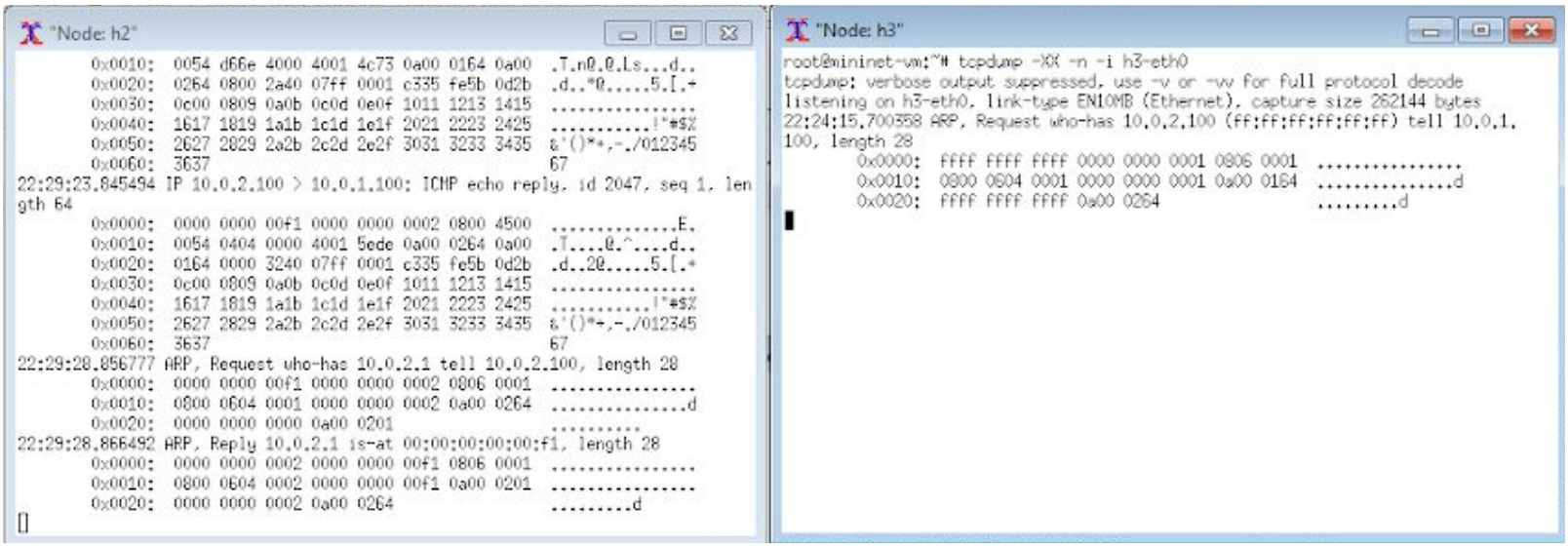
Then we test sending message from h1 to h2 , repeat previous part for h1,h2 and h3, change h2 ip address to 10.0.2.100, then we can see the following image



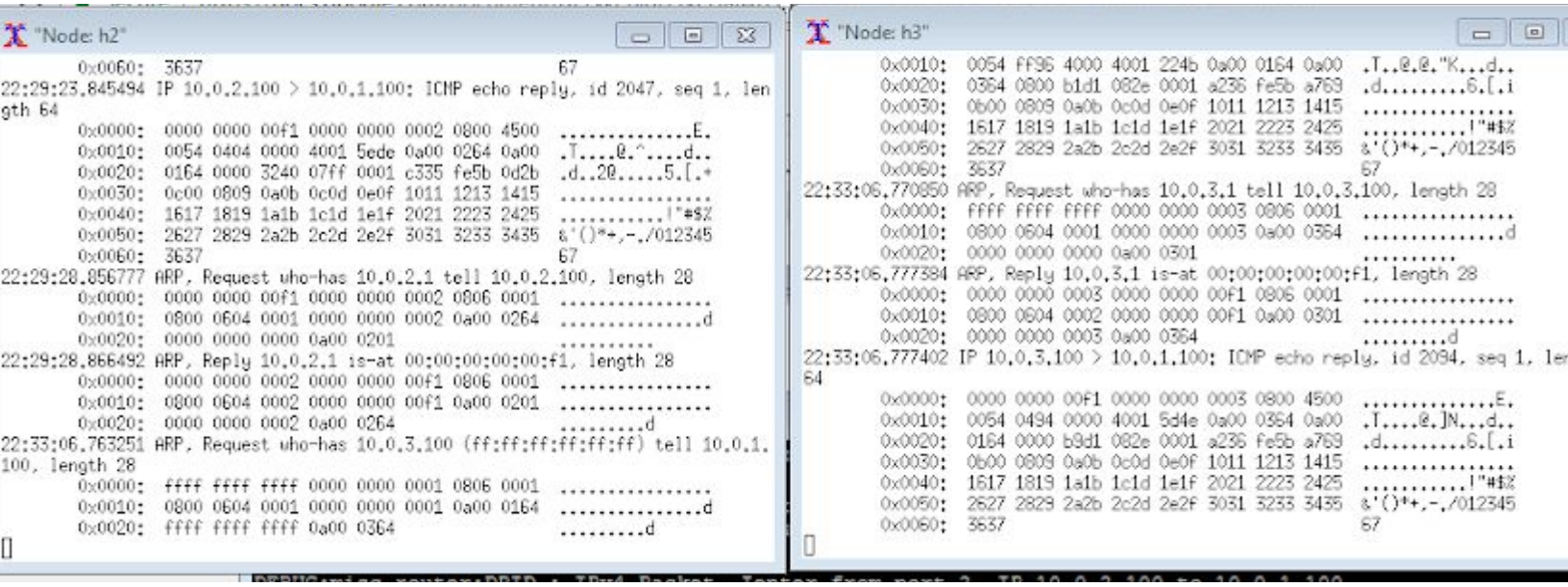
The h3 has received the ARP broadcast request and ignore it, the h2 receive the ARP broadcast request and reply to h1. The h1 shows successfully packet receive



Next, If we continue send message to h2, we can see the following miage:



We can see that h2 receive new packet and h3 doesn't receive anything. Because the router has update the routing table and ARP table for h1 and h2. The router can send packet to h1 na dh2 directly.
If we start send to h3, input h3 IP address 10.0.3.100 and we can see the following:



The h3 do what h2 did previously and h2 did what h3 did previously.

In the pox window, we can see detail process of transmission. For example, the following image show the steps of router if we input pingall (just last part of output)

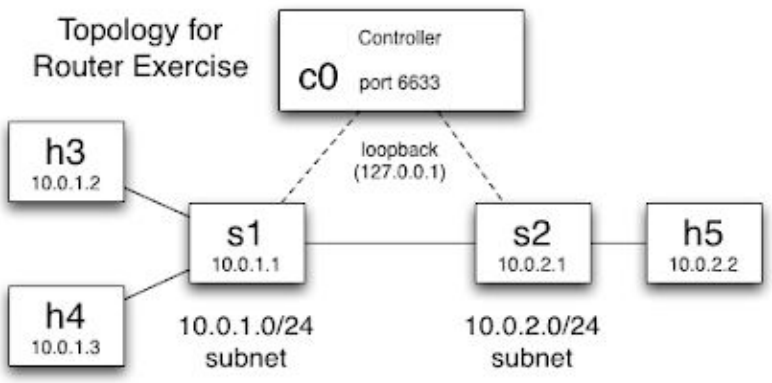
```
mininet@mininet-vm: ~/pox
DEBUG:misc.router:DPID : IP 10.0.1.100,  output port 1, ipve
DEBUG:misc.router:DPID  : Packet 10.0.1.100 to 10.0.2.100, add in ARP wait and send broadcast
DEBUG:misc.router:DPID : INPORT 1, sending ARP Request for 10.0.2.100 on behalf of 10.0.1.100
DEBUG:misc.router:DPID : Added IP 10.0.2.100 into routing Table to port 2, arp
DEBUG:misc.router:DPID : ARP packet, INPORT 2,  ARP  10.0.2.100 -> 10.0.1.100
DEBUG:misc.router:DPID : Added arpTable: ip = 10.0.2.100, mac = 00:00:00:00:00:02
DEBUG:misc.router:DPID : Pending packet in ARP wait line, ip 10.0.2.100
DEBUG:misc.router:DPID : Sending arp wait packet, destip: 10.0.2.100, destmac: 00:00:00:00:00:02, output port: 2
DEBUG:misc.router:DPID : IP 10.0.2.100 , output port 2, arp
DEBUG:misc.router:DPID : ARP packet, INPORT 2,  ARP  10.0.2.100 -> 10.0.2.1
DEBUG:misc.router:DPID : INPORT 2, answer for arp from 10.0.2.100: MAC for 10.0.2.1 is 00:00:00:00:00:f1
DEBUG:misc.router:DPID : IPv4 Packet, Ienter from port 2, IP 10.0.2.100 to 10.0.1.100
DEBUG:misc.router:DPID : IP 10.0.2.100,  output port 2, ipve
DEBUG:misc.router:DPID : Packet 10.0.2.100 to 10.0.1.100 through port 1
DEBUG:misc.router:DPID : IPv4 Packet, Ienter from port 1, IP 10.0.1.100 to 10.0.3.100
DEBUG:misc.router:DPID : IP 10.0.1.100,  output port 1, ipve
DEBUG:misc.router:DPID  : Packet 10.0.1.100 to 10.0.3.100, add in ARP wait and send broadcast
DEBUG:misc.router:DPID : INPORT 1, sending ARP Request for 10.0.3.100 on behalf of 10.0.1.100
DEBUG:misc.router:DPID : Added IP 10.0.3.100 into routing Table to port 3, arp
DEBUG:misc.router:DPID : ARP packet, INPORT 3,  ARP  10.0.3.100 -> 10.0.1.100
DEBUG:misc.router:DPID : Added arpTable: ip = 10.0.3.100, mac = 00:00:00:00:00:03
DEBUG:misc.router:DPID : Pending packet in ARP wait line, ip 10.0.3.100
DEBUG:misc.router:DPID : Sending arp wait packet, destip: 10.0.3.100, destmac: 00:00:00:00:00:03, output port: 3
DEBUG:misc.router:DPID : IP 10.0.3.100 , output port 3, arp
DEBUG:misc.router:DPID : ARP packet, INPORT 3,  ARP  10.0.3.100 -> 10.0.3.1
DEBUG:misc.router:DPID : INPORT 3, answer for arp from 10.0.3.100: MAC for 10.0.3.1 is 00:00:00:00:00:f1
DEBUG:misc.router:DPID : IPv4 Packet, Ienter from port 1, IP 10.0.1.100 to 10.0.2.100
DEBUG:misc.router:DPID : IP 10.0.1.100,  output port 1, ipve
DEBUG:misc.router:DPID : Packet 10.0.1.100 to 10.0.2.100 through port 2
DEBUG:misc.router:DPID : IPv4 Packet, Ienter from port 2, IP 10.0.2.100 to 10.0.3.100
DEBUG:misc.router:DPID : IP 10.0.2.100,  output port 2, ipve
DEBUG:misc.router:DPID : Packet 10.0.2.100 to 10.0.3.100 through port 3
```

The output display detailed steps how the router deal with ARP request/replay, send ICMP message and deal with ARP waiting line

Part 2: Advanced Topology
In this part, we create the following topology

Topology

You'll need a slightly different topology, something like this:

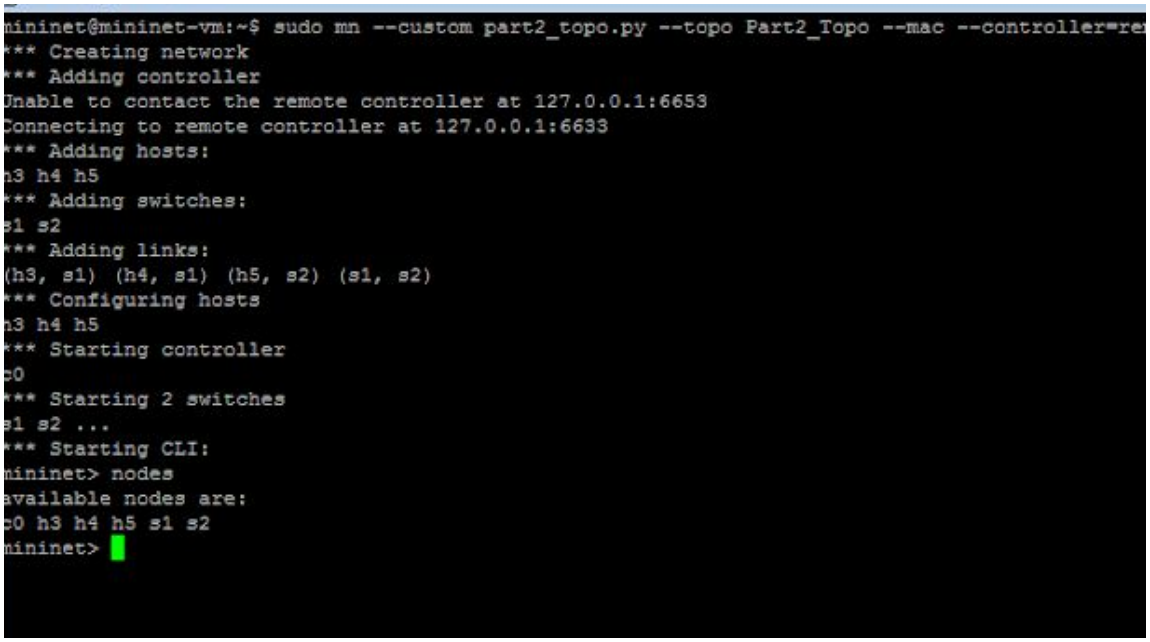


Then, we can change some codes of router.py we wrote in first part.
To change the router, we need to add following function:
(a)The network need to identify different switch. So dpid will be changed in this part.
(b)The routers should identify host in different subnetwork
(c)The controller get knowledge of routing table, arp table and arp wait line of different routers to send message in indicated direction
(4) Should deal with feedback message carefully to avoid infinite loop

To solve above concerns, once the controller need to process one router to send ARP request/reply and deal with ICMP message, the controller will check all routing table and determine what to do at current time. In addition, the router will change destination IP address to that of other router if it find the host is not in his subnetwork.

When we working on this part, we used
https://github.com/zhan849/ee555/blob/master/part1/router.py and
https://github.com/zhan849/ee555/blob/master/part2/router_part2.py as references.

Repeat the previous step, we can get topology as follow:



Firstly, test pingall that we can see:

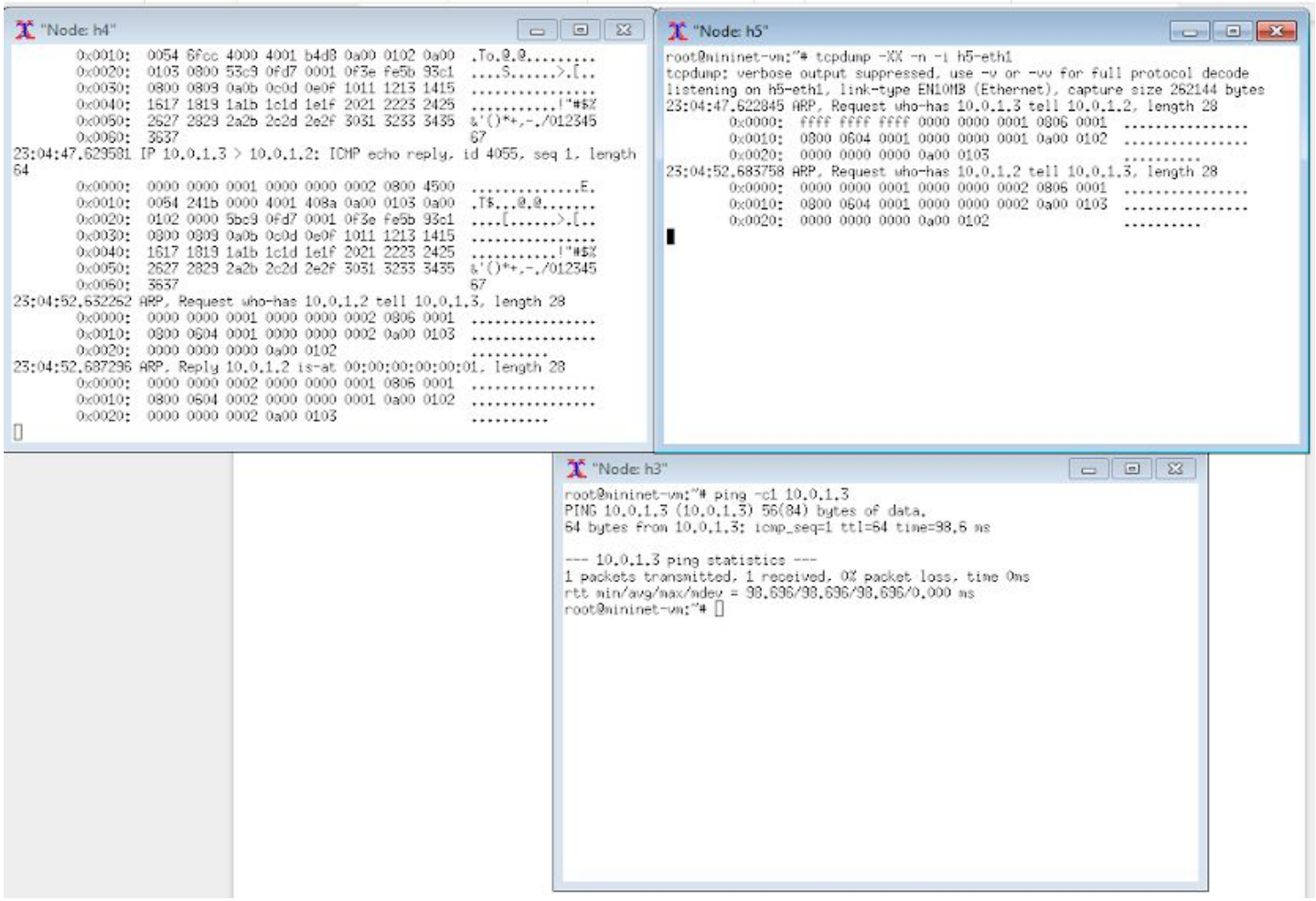
```
mininet@mininet-vm:~$ sudo mn --custom part2_topo.py --topo Part2_Topo --mac --controller=remote,
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h3 h4 h5
*** Adding switches:
s1 s2
*** Adding links:
(h3, s1) (h4, s1) (h5, s2) (s1, s2)
*** Configuring hosts
h3 h4 h5
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h3 h4 h5 s1 s2
mininet> pingall
*** Ping: testing ping reachability
h3 -> h4 h5
h4 -> h3 h5
h5 -> h3 h4
*** Results: 0% dropped (6/6 received)
mininet>
```

In pox terminal, we can see some detail of process:

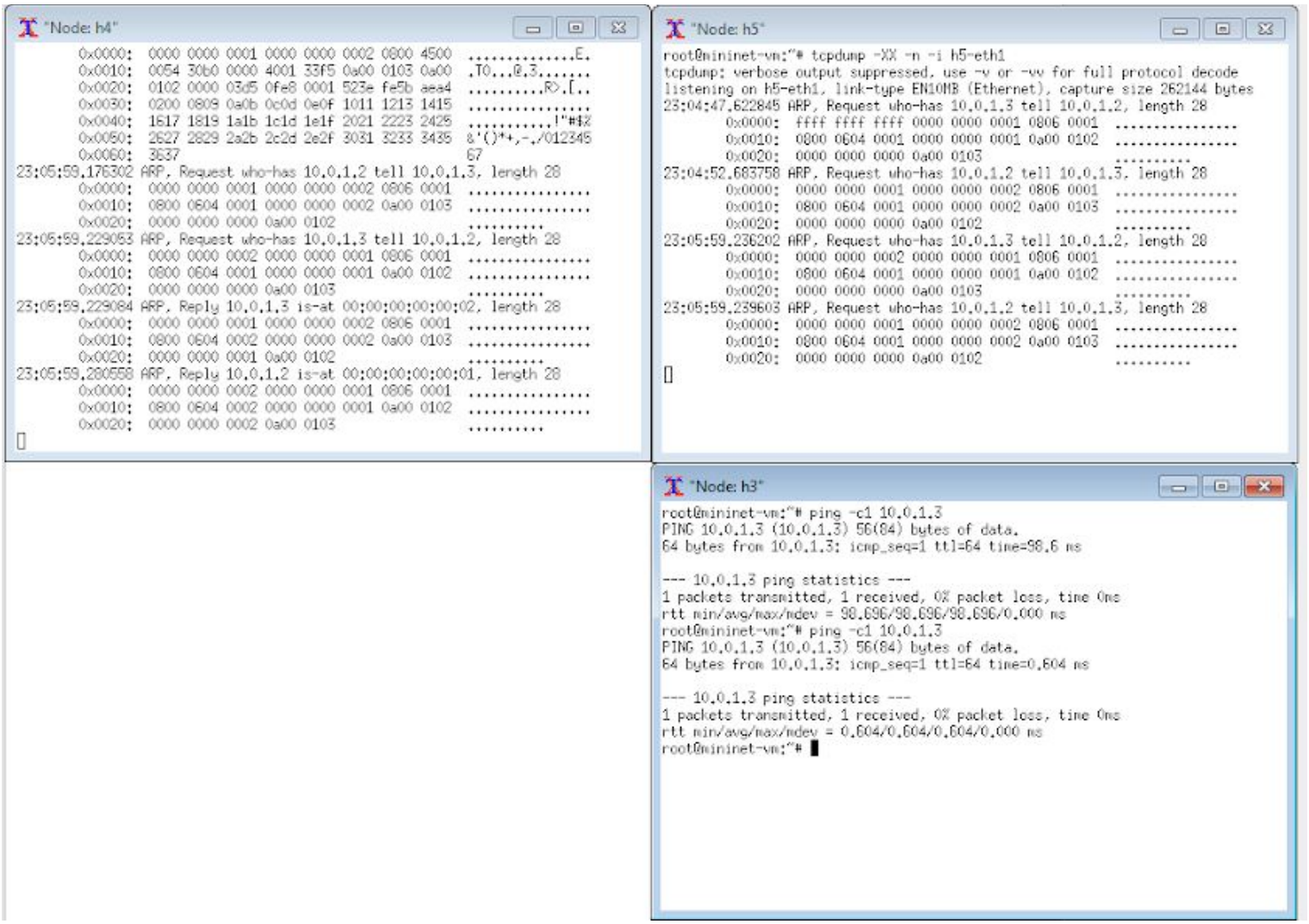
```
DEBUG:misc.router_part2:DPID 2, INPORT 1, sending ARP Request for 10.0.2.2 on behalf of 10.0.1.2
DEBUG:misc.router_part2:DPID 1: IP 10.0.1.2 is in routing Table, RE_LEARNED, output port 1
DEBUG:misc.router_part2:DPID 1: ARP packet, INPORT 1,  ARP request 10.0.1.2 => 10.0.2.2
DEBUG:misc.router_part2:DPID 2: Added IP 10.0.2.2 into routing Table, output port 2
DEBUG:misc.router_part2:DPID 2: ARP packet, INPORT 2,  ARP reply 10.0.2.2 => 10.0.1.2
DEBUG:misc.router_part2:DPID 2: added arpTable entry: ip = 10.0.2.2, mac = 00:00:00:00:00:03
DEBUG:misc.router_part2:DPID 2: processing pending arpWait packet for ip 10.0.2.2
DEBUG:misc.router_part2:DPID 2: sending arp wait packet, buffer id: 258, destip: 10.0.2.2, destmac: 00:00:00:00:00:0
3, output port: 2
DEBUG:misc.router_part2:DPID 1: Added IP 10.0.2.2 into routing Table, output port 1
DEBUG:misc.router_part2:DPID 1: ARP packet, INPORT 1,  ARP reply 10.0.2.2 => 10.0.1.2
DEBUG:misc.router_part2:DPID 1: added arpTable entry: ip = 10.0.2.2, mac = 00:00:00:00:00:03
DEBUG:misc.router_part2:DPID 2: IP 10.0.2.2 is in routing Table, RE_LEARNED, output port 2
DEBUG:misc.router_part2:DPID 2: ARP packet, INPORT 2,  ARP request 10.0.2.2 => 10.0.2.1
DEBUG:misc.router_part2:DPID 2, INPORT 2, answering for arp from 10.0.2.2: MAC for 10.0.2.1 is 00:00:00:00:00:f2
DEBUG:misc.router_part2:DPID 2: IPv4 Packet, INPORT 2, IP 10.0.2.2 => 10.0.1.2
DEBUG:misc.router_part2:DPID 2: IP 10.0.2.2 is in routing Table, RE_LEARNED, output port 2
DEBUG:misc.router_part2:DPID 2, packet 10.0.2.2 => 10.0.1.2, different subnet, sent to port 1
DEBUG:misc.router_part2:DPID 1: IP 10.0.1.3 is in routing Table, RE_LEARNED, output port 3
DEBUG:misc.router_part2:DPID 1: ARP packet, INPORT 3,  ARP request 10.0.1.3 => 10.0.1.1
DEBUG:misc.router_part2:DPID 1, INPORT 3, answering for arp from 10.0.1.3: MAC for 10.0.1.1 is 00:00:00:00:00:f1
DEBUG:misc.router_part2:DPID 2: IPv4 Packet, INPORT 1, IP 10.0.1.3 => 10.0.2.2
DEBUG:misc.router_part2:DPID 2: Added IP 10.0.1.3 into routing Table, output port 1
DEBUG:misc.router_part2:DPID 2, packet 10.0.1.3 => 10.0.2.2, same subnet, sent to port 2
DEBUG:misc.router_part2:DPID 2: IPv4 Packet, INPORT 2, IP 10.0.2.2 => 10.0.1.3
DEBUG:misc.router_part2:DPID 2: IP 10.0.2.2 is in routing Table, RE_LEARNED, output port 2
DEBUG:misc.router_part2:DPID 2, packet 10.0.2.2 => 10.0.1.3, different subnet, sent to port 1
DEBUG:misc.router_part2:DPID 2: IPv4 Packet, INPORT 2, IP 10.0.2.2 => 10.0.1.3
DEBUG:misc.router_part2:DPID 2: IP 10.0.2.2 is in routing Table, RE_LEARNED, output port 2
DEBUG:misc.router_part2:DPID 2, packet 10.0.2.2 => 10.0.1.3, different subnet, sent to port 1
DEBUG:misc.router_part2:DPID 1: IP 10.0.1.3 is in routing Table, RE_LEARNED, output port 3
DEBUG:misc.router_part2:DPID 1: ARP packet, INPORT 3,  ARP request 10.0.1.3 => 10.0.1.2
DEBUG:misc.router_part2:DPID 2: IP 10.0.1.3 is in routing Table, RE_LEARNED, output port 1
DEBUG:misc.router_part2:DPID 2: ARP packet, INPORT 1,  ARP request 10.0.1.3 => 10.0.1.2
DEBUG:misc.router_part2:DPID 2: added arpTable entry: ip = 10.0.1.3, mac = 00:00:00:00:00:02
DEBUG:misc.router_part2:DPID 1: IP 10.0.1.2 is in routing Table, RE_LEARNED, output port 2
DEBUG:misc.router_part2:DPID 1: ARP packet, INPORT 2,  ARP reply 10.0.1.2 => 10.0.1.3
```

Should notice two parts: firstly, in this case we have two routers, so the DPID has indicator 1 or 2 to indicate that whether is router 1 or router 2 is being executed. Second is that if the source and destination IP address at different network, source/destination IP address will be changed to router's IP address temporary.

Then we input xterm h3 h4 h5, IP address of h4 is 10.0.1.3, path is h4-eth1. IP address for h5 is 10.0.2.2 and path is h5-eth1. Firstly send packet from h3 to h4

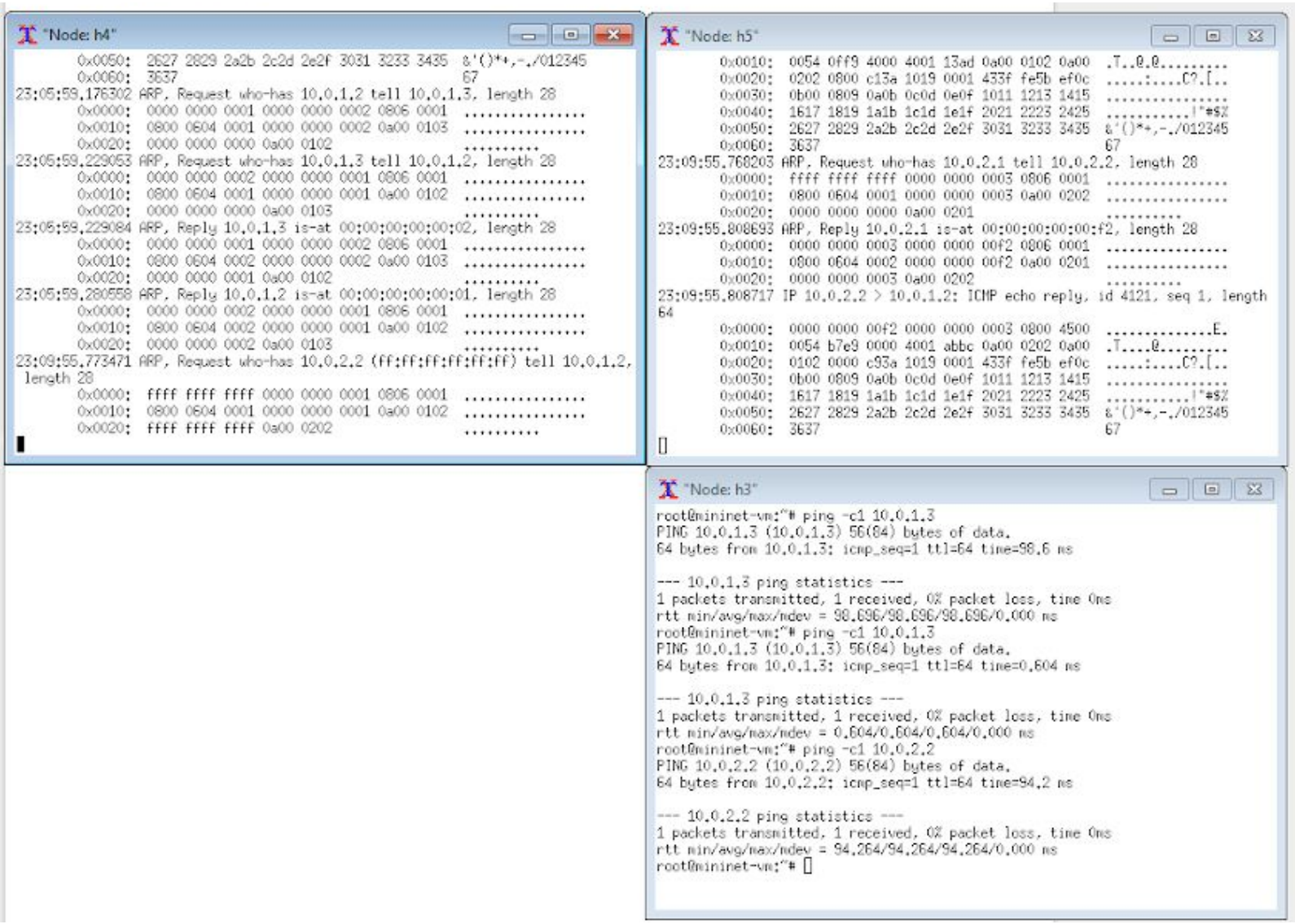


Then send to h4 again



Should notice that in part 2 the h5 has receive more APR request than h3 in part1.

Then send to h5



The h5 receive and response packet. It similar as what we seen in part 1.

For the bonus part, we create the topology and modified router we used in part 2. When pingall was inputted, even though we can see some packet can transmit successfully, but finally the network become infinite loop and had to be forced to terminated