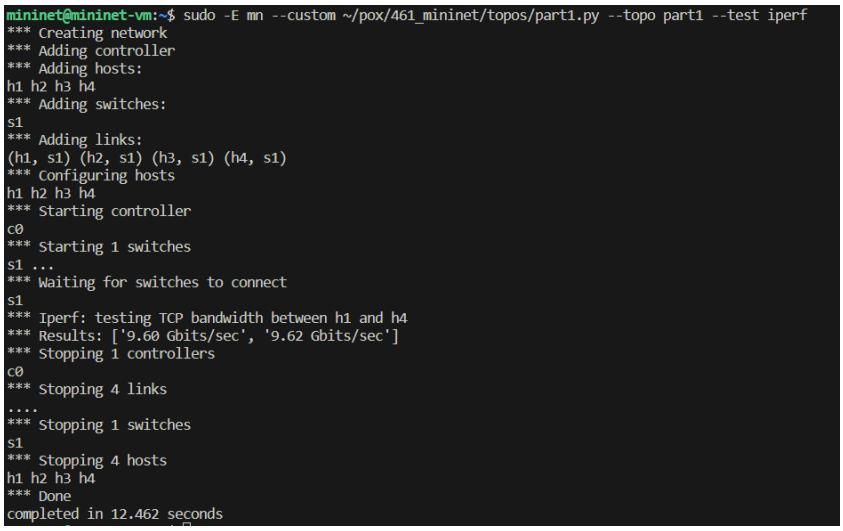
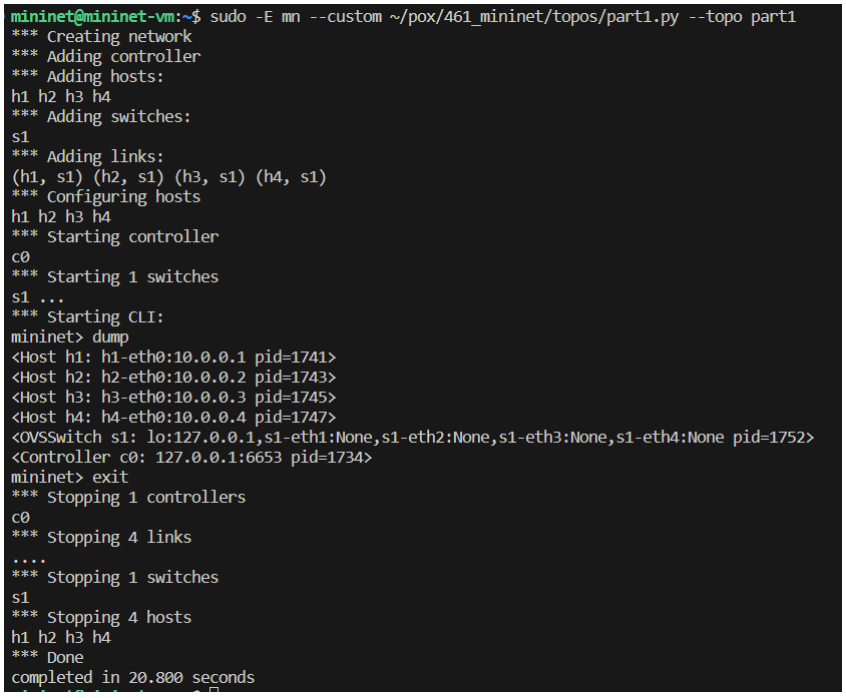
1. Result

1.1. Part 1

1.1.1. iperf



1.1.2. dump



1.1.3. pingall

```
mininet@mininet-vm:~$ sudo -E mn --custom ~/pox/461_mininet/topos/part1.py --topo part1 --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 4 links
....
*** Stopping 1 switches
s1
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 5.923 seconds
```

1.2. Part 2

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.part2controller info.packet_dump samples.prett
y_log log.level --DEBUG
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
INFO:info.packet_dump:Packet dumper running
[core                  ] POX 0.7.0 (gar) going up...
[core                  ] Running on CPython (3.8.5/Jul 28 2020 12:59:40)
[core                  ] Platform is Linux-5.4.0-42-generic-x86_64-with-glibc2.29
[version               ] Support for Python 3 is experimental.
[core                  ] POX 0.7.0 (gar) is up.
[openflow.of_01        ] Listening on 0.0.0.0:6633
[openflow.of_01        ] [00-00-00-00-00-01 1] connected
[forwarding.part2controller] Controlling [00-00-00-00-00-01 1]
[openflow.of_01        ] [00-00-00-00-00-01 1] closed
[openflow.of_01        ] [00-00-00-00-00-01 3] connected
[forwarding.part2controller] Controlling [00-00-00-00-00-01 3]
[openflow.of_01        ] 1 connection aborted
```

1.2.1. pingall (and iperf)

```
mininet@mininet-vm:~$ sudo -E mn --custom ~/pox/461_mininet/topos/part2.py --topo part2 --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
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X h4
h2 -> X h3 X
h3 -> X h2 X
h4 -> h1 X X
*** Results: 66% dropped (4/12 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
^C
Interrupt
stopping h1
stopping h4
```

1.2.2. dpctl dump-flows

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=27.323s, table=0, n_packets=8, n_bytes=784, icmp actions=FLOOD
 cookie=0x0, duration=27.323s, table=0, n_packets=8, n_bytes=336, arp actions=FLOOD
 cookie=0x0, duration=27.323s, table=0, n_packets=4, n_bytes=296, ip actions=drop
mininet>
```

1.3. Part3

```
mininet@mininet-vm:
~$ sudo ~/pox/pox.py forwarding.part3controller info.packet_dump samples.pretty_log log.lev
el --DEBUG
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
INFO:info.packet_dump:Packet dumper running
[core                  ] POX 0.7.0 (gar) going up...
[core                  ] Running on CPython (3.8.5/Jul 28 2020 12:59:40)
[core                  ] Platform is Linux-5.4.0-42-generic-x86_64-with-glibc2.29
[version               ] Support for Python 3 is experimental.
[core                  ] POX 0.7.0 (gar) is up.
[openflow.of_01        ] Listening on 0.0.0.0:6633
[openflow.of_01        ] [00-00-00-00-00-15 2] connected
[forwarding.part3controller] Controlling [00-00-00-00-00-15 2]
21
[openflow.of_01        ] [00-00-00-00-00-01 3] connected
[forwarding.part3controller] Controlling [00-00-00-00-00-01 3]
1
[openflow.of_01        ] [00-00-00-00-00-03 4] connected
[forwarding.part3controller] Controlling [00-00-00-00-00-03 4]
3
[openflow.of_01        ] [00-00-00-00-00-02 5] connected
[forwarding.part3controller] Controlling [00-00-00-00-00-02 5]
2
[openflow.of_01        ] [00-00-00-00-00-1f 6] connected
[forwarding.part3controller] Controlling [00-00-00-00-00-1f 6]
31
[openflow.of_01        ] 1 connection aborted
```

1.3.1. pingall

```
mininet@mininet-vm:~$ sudo -E mn --custom ~/pox/461_mininet/topos/part3.py --topo part3 --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:
h10 h20 h30 hnotrust1 serv1
*** Adding switches:
cores21 dcs31 s1 s2 s3
*** Adding links:
(cores21, dcs31) (h10, s1) (h20, s2) (h30, s3) (hnotrust1, cores21) (s1, cores21) (s2, cores21) (s3, cores21) (ser
v1, dcs31)
*** Configuring hosts
h10 h20 h30 hnotrust1 serv1
*** Starting controller
c0
*** Starting 5 switches
cores21 dcs31 s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h10 -> h20 h30 X serv1
h20 -> h10 h30 X serv1
h30 -> h10 h20 X serv1
hnotrust1 -> X X X X
serv1 -> h10 h20 h30 X
*** Results: 40% dropped (12/20 received)
```

1.3.2. iperf hnotrust1 h10 and iperf h10 serv1

```
mininet> iperf hnotrust1 h10
*** Iperf: testing TCP bandwidth between hnotrust1 and h10
.*** Results: ['22.3 Gbits/sec', '22.4 Gbits/sec']
mininet> iperf h10 serv1
*** Iperf: testing TCP bandwidth between h10 and serv1
*** Results: ['21.0 Gbits/sec', '20.9 Gbits/sec']
```

1.3.3. dpctl dump-flows

```
mininet> dpctl dump-flows
*** cores21 ------------------------------------------------------------------------
 cookie=0x0, duration=111.403s, table=0, n_packets=8, n_bytes=784, icmp,nw_src=172.16.10.100 actions=drop
 cookie=0x0, duration=111.403s, table=0, n_packets=0, n_bytes=0, ip,nw_src=172.16.10.100,nw_dst=10.0.4.10 actions=drop
 cookie=0x0, duration=111.403s, table=0, n_packets=619896, n_bytes=14027868632, ip,nw_dst=10.0.1.10 actions=output:"cores21-eth1"
 cookie=0x0, duration=111.403s, table=0, n_packets=6, n_bytes=588, ip,nw_dst=10.0.2.20 actions=output:"cores21-eth2"
 cookie=0x0, duration=111.403s, table=0, n_packets=6, n_bytes=588, ip,nw_dst=10.0.3.30 actions=output:"cores21-eth3"
 cookie=0x0, duration=111.403s, table=0, n_packets=299716, n_bytes=13110859608, ip,nw_dst=10.0.4.10 actions=output:"cores21-eth4"
 cookie=0x0, duration=111.403s, table=0, n_packets=320198, n_bytes=21133200, ip,nw_dst=172.16.10.100 actions=output:"cores21-eth5"
 cookie=0x0, duration=111.403s, table=0, n_packets=50, n_bytes=2100, actions=FLOOD
*** dcs31 --------------------------------------------------------------------------
 cookie=0x0, duration=111.395s, table=0, n_packets=599447, n_bytes=13130640894, actions=FLOOD
*** s1 -----------------------------------------------------------------------------
 cookie=0x0, duration=111.413s, table=0, n_packets=1239857, n_bytes=27159863246, actions=FLOOD
*** s2 -----------------------------------------------------------------------------
 cookie=0x0, duration=111.404s, table=0, n_packets=63, n_bytes=3374, actions=FLOOD
*** s3 -----------------------------------------------------------------------------
 cookie=0x0, duration=111.409s, table=0, n_packets=63, n_bytes=3374, actions=FLOOD
mininet>
```

1.4. Part 4

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py forwarding.part4controller info
.packet_dump samples.pretty_log log.level --DEBUG
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
INFO:info.packet_dump:Packet dumper running
[core                  ] POX 0.7.0 (gar) going up...
[core                  ] Running on CPython (3.8.5/Jul 28 2020 12:59:40
)
[core                  ] Platform is Linux-5.4.0-42-generic-x86_64-with
-glibc2.29
[version               ] Support for Python 3 is experimental.
[core                  ] POX 0.7.0 (gar) is up.
[openflow.of_01        ] Listening on 0.0.0.0:6633
[openflow.of_01        ] [00-00-00-00-00-15 2] connected
[forwarding.part4controller] Controlling [00-00-00-00-00-15 2]
21
[openflow.of_01        ] [00-00-00-00-00-01 3] connected
[forwarding.part4controller] Controlling [00-00-00-00-00-01 3]
1
[openflow.of_01        ] [00-00-00-00-00-03 4] connected
[forwarding.part4controller] Controlling [00-00-00-00-00-03 4]
3
[openflow.of_01        ] [00-00-00-00-00-02 5] connected
[forwarding.part4controller] Controlling [00-00-00-00-00-02 5]
2
[openflow.of_01        ] [00-00-00-00-00-1f 6] connected
[forwarding.part4controller] Controlling [00-00-00-00-00-1f 6]
31
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1}
[openflow.of_01        ] 1 connection aborted
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1}
```

```
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthAddr('00
:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAd
dr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthAddr('00
:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][icmp][echo][56 bytes]
```

```
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][arp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
ddr('00:00:00:00:00:04'): 4}
[dump:00-00-00-00-00-15 ] [ethernet][ipv4][tcp]
{EthAddr('00:00:00:00:00:01'): 1, EthAddr('00:00:00:00:00:02'): 2, EthAddr('00:00:00:00:00:03'): 3, EthAddr('00:00:00:00:00:05'): 5, EthA
```

1.4.1. pingall

```
mininet@mininet-vm:~$ sudo -E mn --custom ~/pox/461_mininet/topos/part4.py --topo part4 --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:
h10 h20 h30 hnotrust1 serv1
*** Adding switches:
cores21 dcs31 s1 s2 s3
*** Adding links:
(cores21, dcs31) (h10, s1) (h20, s2) (h30, s3) (hnotrust1, cores21) (s1, cores21) (s2, cores21) (s3, cores21) (serv1, dcs31)
*** Configuring hosts
h10 h20 h30 hnotrust1 serv1
*** Starting controller
c0
*** Starting 5 switches
cores21 dcs31 s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h10 -> X X X X
h20 -> X X X X
h30 -> X X X X
hnotrust1 -> X X X X
serv1 -> X X X X
*** Results: 100% dropped (0/20 received)
```

1.4.2. iperf hnotrust1 h10 and iperf h10 serv1

```
mininet> iperf hnotrust1 h10
*** Iperf: testing TCP bandwidth between hnotrust1 and h10
^C
Interrupt
stopping h10
stopping hnotrust1
mininet> iperf h10 serv1
*** Iperf: testing TCP bandwidth between h10 and serv1
^C
Interrupt
stopping h10
stopping serv1
mininet> dpctl dump-flows
```

1.4.3. dpctl dump-flows

```
mininet> dpctl dump-flows
*** cores21 ------------------------------------------------------------------------
 cookie=0x0, duration=394.158s, table=0, n_packets=4, n_bytes=392, icmp,nw_src=172.16.10.100 actions=drop
 cookie=0x0, duration=394.158s, table=0, n_packets=0, n_bytes=0, ip,nw_src=172.16.10.100,nw_dst=10.0.4.10 actions=drop
*** dcs31 --------------------------------------------------------------------------
 cookie=0x0, duration=394.154s, table=0, n_packets=8, n_bytes=560, actions=FLOOD
*** s1 -----------------------------------------------------------------------------
 cookie=0x0, duration=394.164s, table=0, n_packets=18, n_bytes=1172, actions=FLOOD
*** s2 -----------------------------------------------------------------------------
 cookie=0x0, duration=394.164s, table=0, n_packets=6, n_bytes=476, actions=FLOOD
*** s3 -----------------------------------------------------------------------------
 cookie=0x0, duration=394.167s, table=0, n_packets=8, n_bytes=560, actions=FLOOD
```

2. Explanations

2.1. Part 1

- self.addSwitch('s1'): Create a switch named 's1'.
- self.addHost('h1'), self.addHost('h2'), self.addHost('h3'), self.addHost('h4'): Create 4 hosts 'h1', 'h2', 'h3', 'h4'.

- self.addLink(host1, switch1), self.addLink(host2, switch1), self.addLink(host3, switch1), self.addLink(host4, switch1): Create connections between hosts và switchs.

2.2. Part 2

- Initialize a flow table
- Create headers to match the packets by function of.ofp_match()
    - Layer 2: dl_type = 0x0800 (IPv4)
    - Layer 3: nw_proto = 1 (ICMP)
- Instruct switch the next reactions to each receiving packet having matching properties with “match”.
    - of.ofp_action_output(): Forward packets to a physical/virtual port.
        - of.OFPP_FLOOD: forward to all openflow ports except source port and ports which block flood (OFPPC_NO_FLOOD).
    - If there is no instruction, the packet will be dropped.
- Send OpenFlow messages to controller.

2.3. Part 3

- s1_setup, s2_setup, s3_setup:
    - These functions are applied into setting rules for switches (1, 2, 3), and every packet which are flooded (OFPP_FLOOD).
    - If there is no instruction, the packet will be dropped.
- cores21_setup:
    - DROP PACKETS:
        - Block ICMP message from hnotrust1 to anyone
            - Use of.ofp_flow_mod() with dl_type=0x0800 (IPv4) and nw_proto=1 (ICMP) and source address IPS["hnotrust"]. Switch will do nothing (because there is no instruction).
        - Block packets from hnotrust1 to serv1

- Use dl_type=0x800(IPv4), source address IPS["hnotrust"], destination address IPS["serv1"].

- If there is no instruction, the packet will be dropped.

- SPECIFY PORTS:

  - Packets from h10, h20, h30, serv1, hnotrust are respectively sent to switch (port 1, 2, 3, 4, 5).

- dcs31_setup:

  - This function is used to apply rules to switch dcs31. All packets are flooded (OFPP_FLOOD).
  - If there is no instruction, the packet will be dropped.

2.4. Part 4

- s1_setup, s2_setup, s3_setup and dcs31_setup are similar to Part 3's. Function cores21_setup is the same to Part 3's except SPECIFY PORT.

- _handle_PacketIn(self, event):

  - Handle ARP traffic:

    - Create two lookup tables self.mac_to_port = {} to determine MAC address for each IP address and self.ip_to_mac = {} to determine packet's outgoing port when attending to switch for each destination MAC address.

    - If packet A has a MAC address not existing in mac_to_port, outgoing port for each packet P having destination MAC address of packet A's MAC address will be packet A's attending port.

    - If packet A has a MAC address existing in mac_to_port, take the respective port to the MAC address and send the packet back to source host through that port.

  - Create ARP reply:

    - If IP address of the request packet has not existed in ip_to_mac, update respective MAC address to that IP address.

    - Create an ARP reply from MAC address of current connection to the

packet’s source address through its attending port.

- Update outgoing port for each MAC address:
    - With each MAC address inside mac_to_port, create a flow table having that MAC address as dl_src and determine sending action to the respective port.

Reference

[1] POX Wiki - Open Networking Lab - Confluence, 5 March 2015, https://intronetworks.cs.luc.edu/auxiliary_files/mininet/poxwiki.pdf.

[2] "demos-with-POX-and-mininet/l3_learning.py at master · chapter09/demos-with-POX-and-mininet." GitHub, https://github.com/chapter09/demos-with-POX-and-mininet/blob/master/l3_learning.py#L179.

[3] "EtherType." Wikipedia, https://en.wikipedia.org/wiki/EtherType.

[4] Haichen Shen. "Module to perform round-robin load balancing." University of Washington, https://courses.cs.washington.edu/courses/csep561/13au/projects/sol/proj2/pox/loadbalancer.py.

[5] Junaid Khalid. "L2 learning switch written directly against the OpenFlow library." University of Washington.

"mahdafr/20u_cs5391-t5: Software Defined Networking (SDN)." GitHub, https://github.com/mahdafr/20u_cs5391-t5.

[6] "mahdafr/20u_cs5391-t6: SDN: Level3 Routing with Learning." GitHub, https://github.com/mahdafr/20u_cs5391-t6.

[7] "OpenFlow flow matching." OpenFaucet v1.1 documentation, Github, 2 October 2022, https://rlenglet.github.io/openfaucet/match.html#openfaucet.ofmatch.Match.

[8] "OpenFlow Switch Specification." Open Networking Foundation, https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf.

[9] Russell, Sam. "ARP and ping in POX - Building a POX-based OpenFlow router." Sam Russell Central, 31 August 2012, https://www.samrussell.nz/2012/08/arp-and-ping-in-pox-building-pox-based.html.