

Task 1:

Q 1:

Nodes:

```
mininet> nodes
```

available nodes are:

```
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
```

Net:

```
mininet> net
```

```
h1 h1-eth0:s3-eth1
```

```
h2 h2-eth0:s3-eth2
```

```
h3 h3-eth0:s4-eth1
```

```
h4 h4-eth0:s4-eth2
```

```
h5 h5-eth0:s6-eth1
```

```
h6 h6-eth0:s6-eth2
```

```
h7 h7-eth0:s7-eth1
```

```
h8 h8-eth0:s7-eth2
```

```
s1 lo: s1-eth1:s2-eth3 s1-eth2:s5-eth3
```

```
s2 lo: s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
```

```
s3 lo: s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
```

```
s4 lo: s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
```

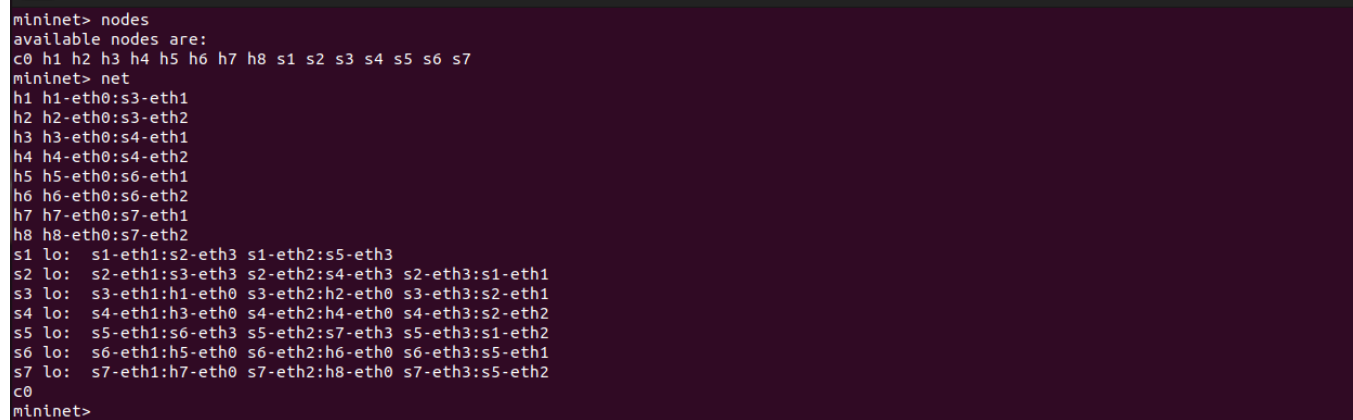
```
s5 lo: s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
```

```
s6 lo: s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
```

```
s7 lo: s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
```

```
C0
```

Screenshot –



```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo: s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo: s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo: s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
s4 lo: s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
s5 lo: s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
s6 lo: s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
s7 lo: s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
c0
mininet>
```

Q 2:

```
mininet> h7 ifconfig
```

Screenshot –

```
mininet> h7 ifconfig
h7-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.7 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::ccb3:9cff:feee:c74f prefixlen 64 scopeid 0x20<link>
    ether ce:b3:9c:ee:c7:4f txqueuelen 1000 (Ethernet)
    RX packets 69 bytes 5298 (5.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11 bytes 866 (866.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet>
```

Task 2:

Q 1:

We begin by turning on the POX listener, which then turns on the start switch. The `_handle PacketIn()` method is called by the start switch to handle the packet in messages from the switch.

The `act like hub()` function is then called by the `_handle PacketIn()` function. The `act like hub()` function generates a behavior that sends packets to all ports except the input port, simulating a hub environment. The `resend packet()` method is then called. The `resend packet()` function adds a packet to the message data and performs the action on it. The switch is then instructed to resend the packet to a specified port by this message.

The graph is shown as below:

start switch : `_handle_PacketIn()` -> `act_like_hub()` -> `resend_packet()` -> `send(msg)`

Question 2:

Case 1 –

`h1 ping -c100 h2 –`

Average ping=2.906ms

Output screenshot -

```
mininet> h1 ping -c100 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=1.90 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.73 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.85 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.17 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2.37 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=3.17 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=3.06 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=2.94 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=3.17 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=3.14 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=2.72 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=3.02 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=2.82 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=3.31 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=2.59 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=3.22 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=3.14 ms
^C
--- 10.0.0.2 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16025ms
rtt min/avg/max/mdev = 1.908/2.906/3.315/0.352 ms
mininet> █
```

Case 2 - `h1 ping -c100 h8`

Average ping=8.817ms

```

mininet> h1 ping -c100 h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=5.34 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=7.32 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=7.18 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=8.48 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=8.89 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=9.98 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=9.77 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=9.03 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=10.4 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=7.31 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=8.76 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=8.93 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=9.46 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=8.54 ms
64 bytes from 10.0.0.8: icmp_seq=15 ttl=64 time=9.10 ms
64 bytes from 10.0.0.8: icmp_seq=16 ttl=64 time=11.1 ms
64 bytes from 10.0.0.8: icmp_seq=17 ttl=64 time=10.1 ms
^C
--- 10.0.0.8 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16024ms
rtt min/avg/max/mdev = 5.344/8.817/11.118/1.376 ms
mininet>

```

2b) Minimum and Maximum ping –

- a. Minimum ping=1.908ms, Maximum ping=3.315ms
- b. Minimum ping=5.344ms, Maximum ping=11.118ms

2c) Ping time is higher for h1 to h8 compared with h1 to h2 - h1 has to traverse through s3,s2,s1,s5 and s7 to reach h8 while there is only one switch between h1 and h2. Therefore, ping time is higher in the second case.

Q 3:

3a) Iperf is an open source program that assists administrators in determining bandwidth for network performance and line quality. Two hosts running iperf are limiting the network link. It is used to determine the amount of data transferred between any two nodes on a network line.

3b)

- a. iperf h1 h2 –
Results for throughput – Results['19.3 Mbits/sec', '22.1 Mbits/sec']
- b. iperf h1 h8 –
Results for throughput – Results['3.16 Mbits/sec', '3.67 Mbits/sec']

Output Screenshot -

```

mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['19.3 Mbits/sec', '22.1 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['3.16 Mbits/sec', '3.67 Mbits/sec']
mininet>

```

3c) Because of network congestion and latency, throughput is higher between h1 and h2 than between h1 and h8 (same as ping time being slower). Because the number of hops between h1 and h2 is lower, more data can be sent in less time. Because the number of hops between h1 and h8 is higher, less data may be transferred in a given amount of time.

Q 4:

We may inspect the information that lets us observe the traffic by adding `log.info("Switch observing traffic: percent s" percent (self.connection) to the line number 107 "of tutorial" controller.` As a result of this, we can deduce that all switches monitor traffic, particularly when they are overloaded with packets. The event listener function `_handle PacketIn` is invoked whenever a packet is received.

Task 3:

1. Answer -

```

root@fe486415b47c:~/pox# ./pox.py log.level --DEBUG misc.of_tutorial
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
DEBUG:core:POX 0.7.0 (gar) going up...
DEBUG:core:Running on CPython (3.6.9/Mar 15 2022 18:35:58)
DEBUG:core:Platform is Linux-5.4.0-110-generic-x86_64-with-Ubuntu-18.04-bionic
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-07 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-07 2]
INFO:openflow.of_01:[00-00-00-00-00-06 3] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-06 3]
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:[00-00-00-00-00-04 5] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-04 5]
INFO:openflow.of_01:[00-00-00-00-00-03 6] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-03 6]
INFO:openflow.of_01:[00-00-00-00-00-02 7] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-02 7]
INFO:openflow.of_01:[00-00-00-00-00-05 8] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-05 8]
Learning that fe:c7:33:e2:a7:63 is attached at port 3
33:33:00:00:00:16 not known, resend to everybody
Learning that fe:c7:33:e2:a7:63 is attached at port 3
33:33:00:00:00:16 not known, resend to everybody

```

[illegible]

33:33:ff:3e:0b:62 not known, resend to everybody
33:33:ff:3e:0b:62 not known, resend to everybody
33:33:ff:3e:0b:62 not known, resend to everybody
33:33:ff:3e:0b:62 not known, resend to everybody
33:33:ff:3e:0b:62 not known, resend to everybody
Learning that 0a:fa:38:0c:09:dc is attached at port 2
33:33:00:00:00:16 not known, resend to everybody
Learning that 0a:fa:38:0c:09:dc is attached at port 3
33:33:00:00:00:16 not known, resend to everybody
Learning that 0a:fa:38:0c:09:dc is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that 0a:fa:38:0c:09:dc is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that 0a:fa:38:0c:09:dc is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that fa:f1:d1:80:0c:b9 is attached at port 3
33:33:ff:80:0c:b9 not known, resend to everybody
Learning that fa:f1:d1:80:0c:b9 is attached at port 3
33:33:ff:80:0c:b9 not known, resend to everybody
Learning that fa:f1:d1:80:0c:b9 is attached at port 1
33:33:ff:80:0c:b9 not known, resend to everybody
Learning that fa:f1:d1:80:0c:b9 is attached at port 1
33:33:ff:80:0c:b9 not known, resend to everybody
Learning that fa:f1:d1:80:0c:b9 is attached at port 1
33:33:ff:80:0c:b9 not known, resend to everybody
Learning that fa:f1:d1:80:0c:b9 is attached at port 1
33:33:ff:80:0c:b9 not known, resend to everybody
Learning that a6:e8:e9:dd:39:24 is attached at port 2
33:33:00:00:00:16 not known, resend to everybody
Learning that a6:e8:e9:dd:39:24 is attached at port 2
33:33:00:00:00:16 not known, resend to everybody
Learning that a6:e8:e9:dd:39:24 is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that a6:e8:e9:dd:39:24 is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that a6:e8:e9:dd:39:24 is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that a6:e8:e9:dd:39:24 is attached at port 1
33:33:00:00:00:16 not known, resend to everybody

Learning that a6:e8:e9:dd:39:24 is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
33:33:ff:0c:09:dc not known, resend to everybody
33:33:ff:0c:09:dc not known, resend to everybody
33:33:ff:0c:09:dc not known, resend to everybody
33:33:ff:0c:09:dc not known, resend to everybody
33:33:ff:0c:09:dc not known, resend to everybody
33:33:ff:0c:09:dc not known, resend to everybody
Learning that 1a:d1:21:83:fe:bb is attached at port 3
33:33:00:00:00:16 not known, resend to everybody
Learning that 1a:d1:21:83:fe:bb is attached at port 2
33:33:00:00:00:16 not known, resend to everybody
Learning that 1a:d1:21:83:fe:bb is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that 1a:d1:21:83:fe:bb is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that 1a:d1:21:83:fe:bb is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
Learning that 1a:d1:21:83:fe:bb is attached at port 1
33:33:00:00:00:16 not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 2
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 3
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 1
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 2
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 1
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 1
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:ff:e2:a7:63 not known, resend to everybody
Learning that fa:94:b9:5b:2f:4d is attached at port 1
33:33:ff:5b:2f:4d not known, resend to everybody
33:33:00:00:00:16 not known, resend to everybody
33:33:00:00:00:16 not known, resend to everybody

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Q 2: a)

a. h1 ping -c100 h2 -

Average time to ping- 2.648ms

Screenshot -

```

mininet> h1 ping -c100 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=1.72 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3.23 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=2.27 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.16 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=2.57 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=2.38 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=2.89 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=2.21 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=2.93 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=2.83 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=2.89 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=2.88 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=2.40 ms
^C
--- 10.0.0.2 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12018ms
rtt min/avg/max/mdev = 1.728/2.648/3.237/0.416 ms
mininet>

```

b. h1 ping -c100 h8 –

Average time to ping - 8.111ms

Screenshot –

```

mininet> h1 ping -c100 h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data.
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=4.33 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=64 time=11.1 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=64 time=2.69 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=64 time=8.28 ms
64 bytes from 10.0.0.8: icmp_seq=5 ttl=64 time=11.0 ms
64 bytes from 10.0.0.8: icmp_seq=6 ttl=64 time=8.48 ms
64 bytes from 10.0.0.8: icmp_seq=7 ttl=64 time=6.98 ms
64 bytes from 10.0.0.8: icmp_seq=8 ttl=64 time=9.90 ms
64 bytes from 10.0.0.8: icmp_seq=9 ttl=64 time=9.53 ms
64 bytes from 10.0.0.8: icmp_seq=10 ttl=64 time=7.02 ms
64 bytes from 10.0.0.8: icmp_seq=11 ttl=64 time=10.2 ms
64 bytes from 10.0.0.8: icmp_seq=12 ttl=64 time=6.65 ms
64 bytes from 10.0.0.8: icmp_seq=13 ttl=64 time=8.57 ms
64 bytes from 10.0.0.8: icmp_seq=14 ttl=64 time=8.66 ms
^C
--- 10.0.0.8 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13019ms
rtt min/avg/max/mdev = 2.698/8.111/11.166/2.335 ms
mininet>

```

2b) h1 ping -c100 h2 -

Minimum time=1.728ms

Maximum time=3

h1 ping -c100 h8

Minimum time=6.837

Maximum time=10.798

2c) Task 3 takes somewhat less time than task 2 for the value for h1 ping h2, while the difference is not substantial. In the instance of h1 and h8, the difference in ping time figures is large because it needs to go through a lot more changes. Because only the first few packets are flooded in job 3, it is clear that task 3 is considerably faster/ or has less ping time. The switches will only resend the packet to the corresponding port that is mapped to in the "mac to port" mapping after the destination MAC address is identified in the map. As a result, future pings are substantially faster due to the lack of network congestion.

Q 3:

3a) What is the throughput for each case?

a. iperf h1 h2 -

*** Iperf: testing TCP bandwidth between h1 and h2

*** Results: ['18.2 Mbits/sec', '21.2 Mbits/sec']

b. Iperf h1 h8 -

testing TCP bandwidth between h1 and h8

*** Results: ['2.54 Mbits/sec', '2.93 Mbits/sec']

Output -

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['18.2 Mbits/sec', '21.2 Mbits/sec']
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['2.54 Mbits/sec', '2.93 Mbits/sec']
mininet>
```

3b) What is the difference from Task 2 and why do you think there is a change if there is?

Answer:

In both cases, the throughput for task 3 is higher than that of task 2. This is because, unlike task 3, there will be less network congestion since when mac to port. Map has learned all the ports, there will be no flooding of packets and the switches will be less burdened. Given that the routes are more pre-computed and learned with changes in controller, we can see in h1 and h2, task 2 and 3 had significant improvement in throughputs. There is no significant improvement in the case of h1 and h8, due to the number of hops and packet dropping.

