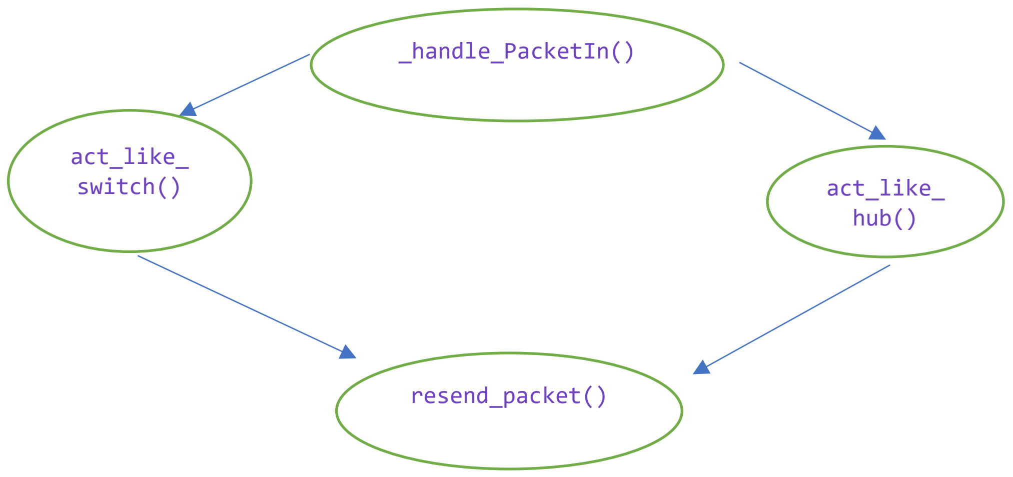
Part 2

Q1: Draw the function call graph of this controller. For example, once a packet comes to the controller, which function is the first to be called, which one is the second, and so forth?

Ans1:The function call graph is as follows



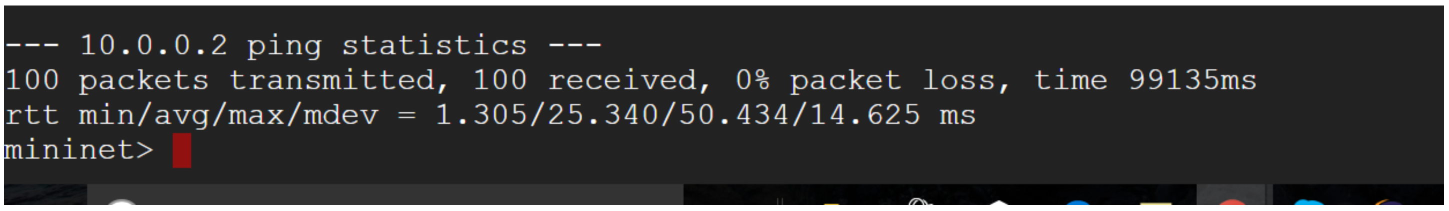
When the packet arrives, it first goes through _handle_PacketIn()

Then according to switch behaviors, call is made to either act_like_hub() or act_like_switch()

Final call is made to resend_packet().

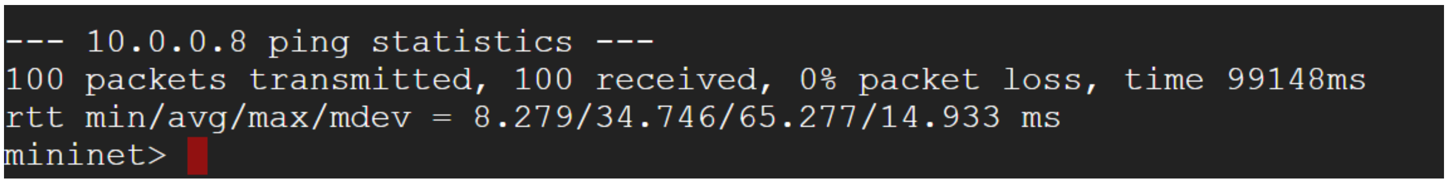
Ans2:

H1 ping h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99135ms
rtt min/avg/max/mdev = 1.305/25.340/50.434/14.625 ms
mininet>
```

H1 ping h8



```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99148ms
rtt min/avg/max/mdev = 8.279/34.746/65.277/14.933 ms
mininet>
```

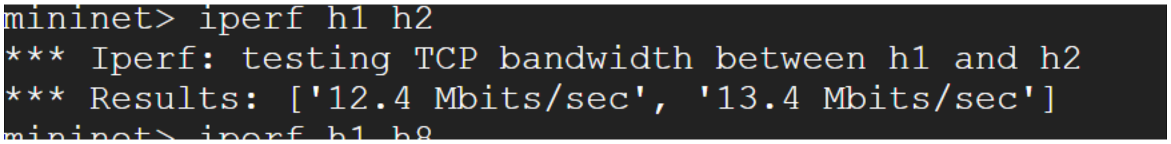
We can see from data above that h1 to h8 takes more time then h1 to h2.

When we ping h1 to h2 then the packet has to go through only one switch s2. H2 is neat to h1 but when we ping h1 to h8 then the packet has to undergo through switch s1 to s5. Then from s5 to s7. Then from s7 to s6 .Then from s6 to s4 and from switch s4 to host h8.So to cover all this switches it normally takes more time

Ans3: **Iperf** is a widely-used tool for network performance measurement and tuning. It is significant as a cross-platform tool that can produce standardized performance measurements for any network. Iperf has client and server functionality, and can create data streams to measure the throughput between the two ends in one or both directions. Typical Iperf output contains a time-stamped report of the amount of data transferred and the throughput measured.The data streams can be either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP)

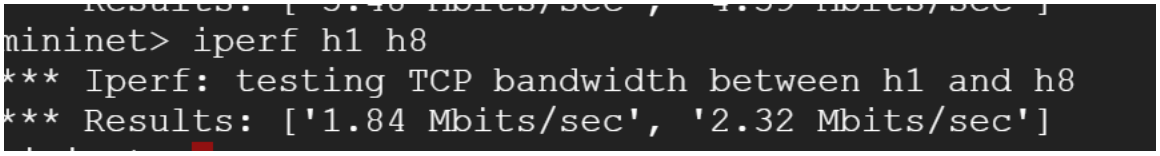
Following is throughput for each case.

Iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['12.4 Mbits/sec', '13.4 Mbits/sec']
```

Iperf h1 h8



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['1.84 Mbits/sec', '2.32 Mbits/sec']
```

Throughput is clearly more for h1 -h2 case then h1-h8. This is because between h1 and h2 the number of switches is pretty less that is just one switch s2. However between h1 and h8 there are total 5 switches listed as s1,s5,s7,s6 and s4.

Q4: Which of the switches observe traffic? Please describe your way for observing such traffic on switches (e.g., adding some functions in the "of_tutorial" controller).

Ans4. The switches contact the controller whenever a packet arrives in the switch . Thus all switches will almost have same traffic.

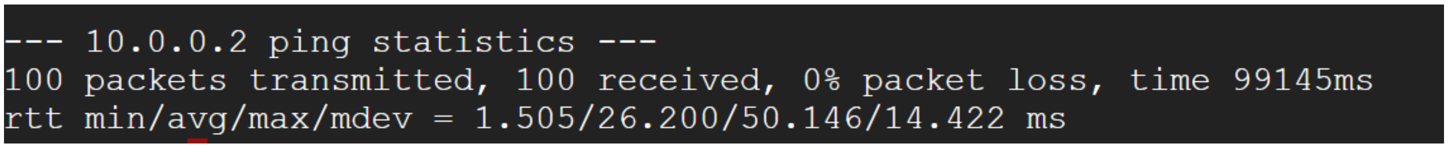
We can used (event.dpid) . In _handle_packetIn , check if dpid is present, if true then incrementing value and if not present setting the dpid =1.

TASK3:

Ans1: The active switch is configured according to the packet which arrives in network.Flow table of connection is checked against packet source that provides maping of host's address. If the key is not present, then port number entry is made with packet source. Destination information is checked for packet. If packet destination is known then packet is sent to the destined port .

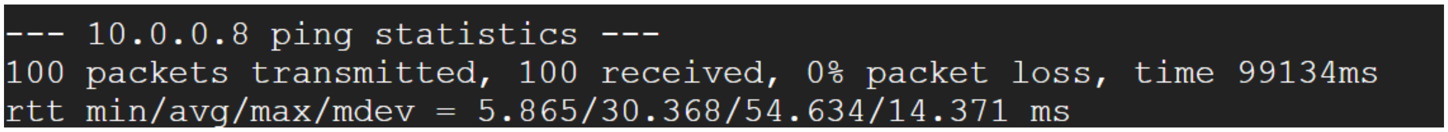
Ans2:

h1 ping h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99145ms
rtt min/avg/max/mdev = 1.505/26.200/50.146/14.422 ms
```

h1 ping h8

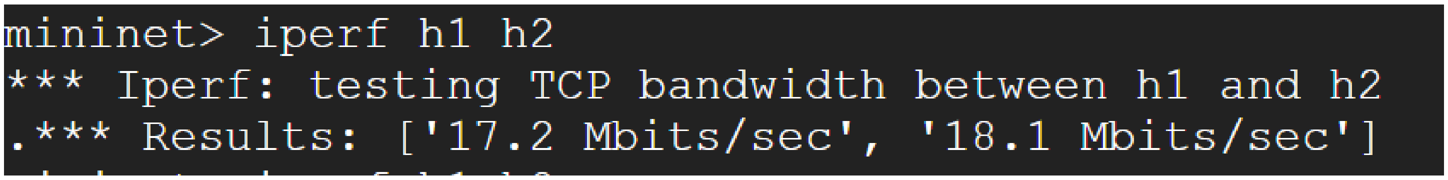


```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99134ms
rtt min/avg/max/mdev = 5.865/30.368/54.634/14.371 ms
```

The entry is established from previous transmissions so the time taken reduces.

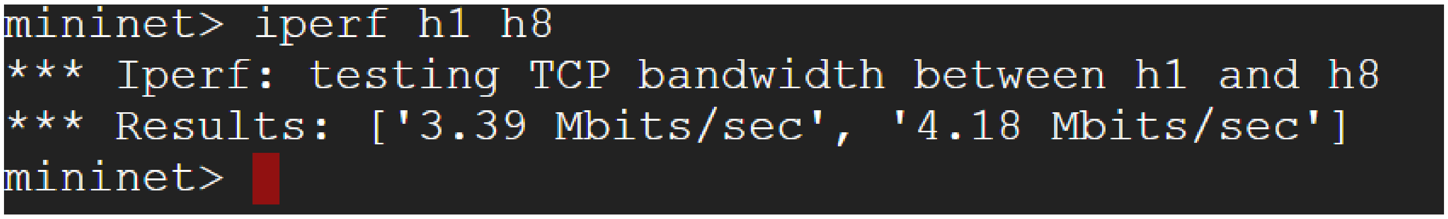
Ans3:

iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['17.2 Mbits/sec', '18.1 Mbits/sec']
```

Iperf h1 h8



```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['3.39 Mbits/sec', '4.18 Mbits/sec']
mininet>
```

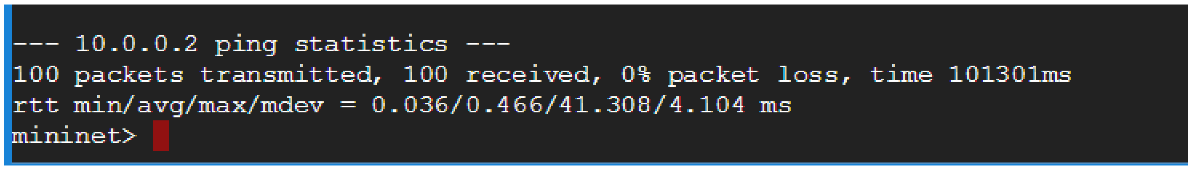
There is improvement in the bandwidth from the earlier scenario because every time the packets are not flooded to each port and if an entry exist in the mac to port table

PART 4

Q.1 Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2). How long does it take (on average) to ping for each case? Any difference from Task III (the MAC case without inserting flow rules)?

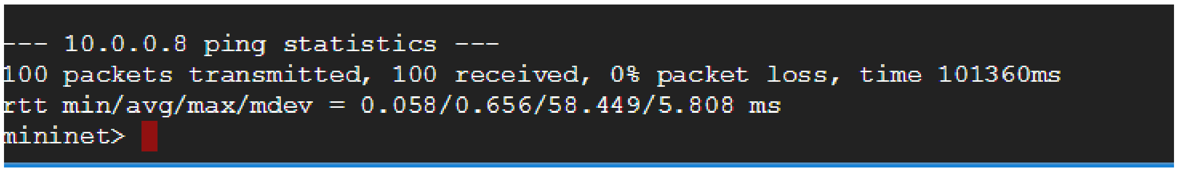
Ans1:

H1 ping h2



```
--- 10.0.0.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101301ms
rtt min/avg/max/mdev = 0.036/0.466/41.308/4.104 ms
mininet>
```

H1 ping h8



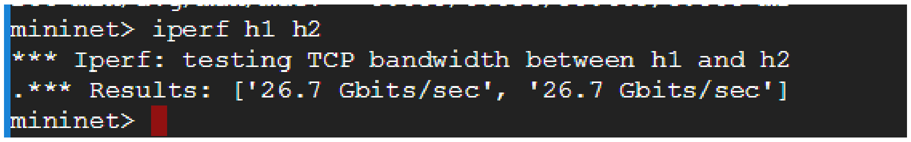
```
--- 10.0.0.8 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101360ms
rtt min/avg/max/mdev = 0.058/0.656/58.449/5.808 ms
mininet>
```

The flow entry decreases the time for packet transfer. The packet is forwarded to the destination instead of flooding.

Q.2 Run “iperf h1 h2” and “iperf h1 h8”. What is the throughput for each case? What is the difference from Task III?

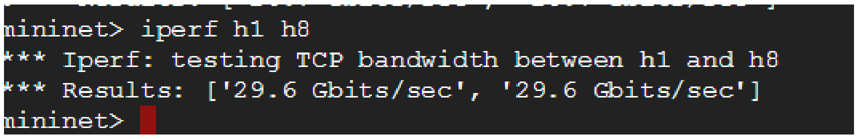
Ans2:

Iperf h1 h2



```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['26.7 Gbits/sec', '26.7 Gbits/sec']
mininet>
```

Iperf h1 h8



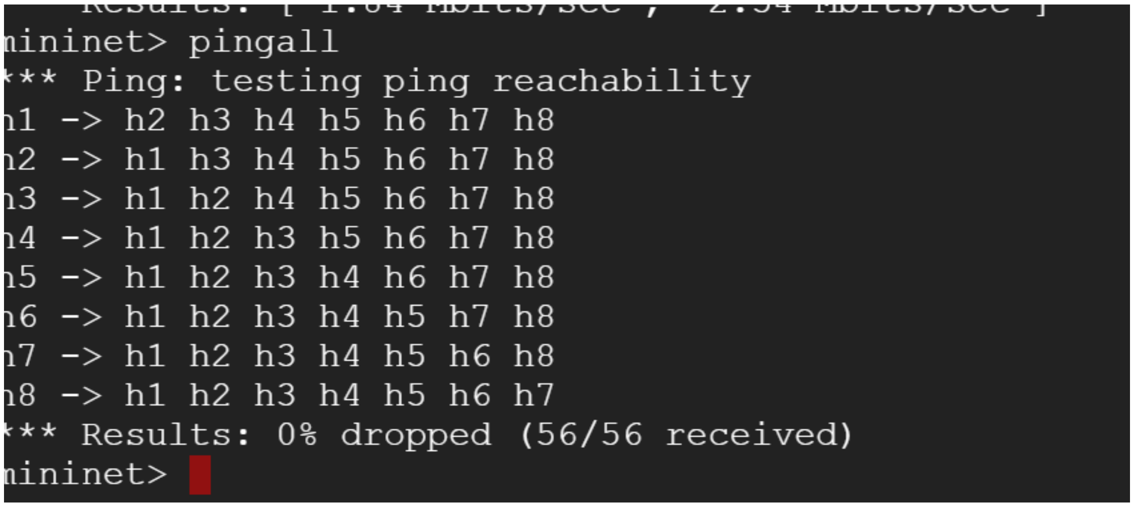
```
mininet> iperf h1 h8
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['29.6 Gbits/sec', '29.6 Gbits/sec']
mininet>
```

The flow entry decreases the time for packet transfer. The packet is forwarded to the destination instead of flooding.Thus the bandwidth is saved drastically as there is no packet flooding.

Q.3 Please explain the above results — why the results become better or worse?

Ans3: Insertion of flow entries in the flow table of the switches improves the performance. After the first packet is sent an entry is created by the controller in the flow table and rest of the packet are directly routed to the destination using these entries in the flow table.

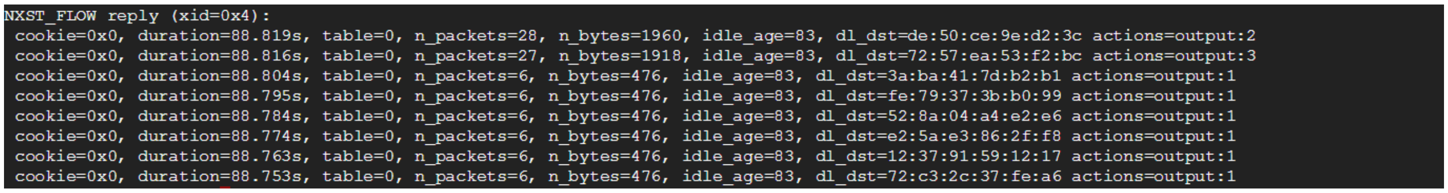
Q.4 Run pingall to verify connectivity and dump the output



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
```

Q5:Dump the output of the flow rules using “ovs-ofctl dump-flows” (in your container, not mininet). How many rules are there for each Open Flow switch, and why? What does each flow entry mean (select one flow entry and explain)?

Pingall dump



```
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=88.819s, table=0, n_packets=28, n_bytes=1960, idle_age=83, dl_dst=de:50:ce:9e:d2:3c actions=output:2
 cookie=0x0, duration=88.816s, table=0, n_packets=27, n_bytes=1918, idle_age=83, dl_dst=72:57:ea:53:f2:bc actions=output:3
 cookie=0x0, duration=88.804s, table=0, n_packets=6, n_bytes=476, idle_age=83, dl_dst=3a:ba:41:7d:b2:b1 actions=output:1
 cookie=0x0, duration=88.795s, table=0, n_packets=6, n_bytes=476, idle_age=83, dl_dst=fe:79:37:3b:b0:99 actions=output:1
 cookie=0x0, duration=88.784s, table=0, n_packets=6, n_bytes=476, idle_age=83, dl_dst=52:8a:04:a4:e2:e6 actions=output:1
 cookie=0x0, duration=88.774s, table=0, n_packets=6, n_bytes=476, idle_age=83, dl_dst=e2:5a:e3:86:2f:f8 actions=output:1
 cookie=0x0, duration=88.763s, table=0, n_packets=6, n_bytes=476, idle_age=83, dl_dst=12:37:91:59:12:17 actions=output:1
 cookie=0x0, duration=88.753s, table=0, n_packets=6, n_bytes=476, idle_age=83, dl_dst=72:c3:2c:37:fe:a6 actions=output:1
```

Flow entries =8

Taking flow entry description

Duration:	duration of the flow.
n_packet:	The number of packets flowing through using the mapping from source to destination.
idle_age:	The The idle age is simply how long the flow has not matched any packets.
dl_dst:	The mac address of the destination as we are using level 2 switching.
n_byte:	The number of bytes flowing through using the mapping from source to destination
Table:	The open switch table used to store the flow entry.

.

TASK 5 –
LAYER 5 ROUTING

In mininet terminal

1) h1 ifconfig

It is 10.0.0.1

Now for h2 it will be 10.0.0.2

2) To manually setup necessary flow between host with port

It is done by 6633+n where n is host number

```
ovs-ofctl add-flow s7 in_port=6634,actions=output:6635
ovs-ofctl add-flow s7 in_port=6635,actions=output:6634
It will forward packets coming at port=6634 and port=6635 in switch s7
```

3) To validate flow table in switch s7

ovs-ofctl dump-flows s7