**CS 552**

**Introduction to Cloud Computing**

**Mini Project – 3**

**Name – Radhika Dotihal**

**B-Number – B00870831**

**2.5 Task II Study of “of\_tutorial” Controller**

Q.1 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?

->

\_\_init\_\_(self,connection)

\_handle\_Packet(self,event)

act\_like\_hub(self,packet,packet in)

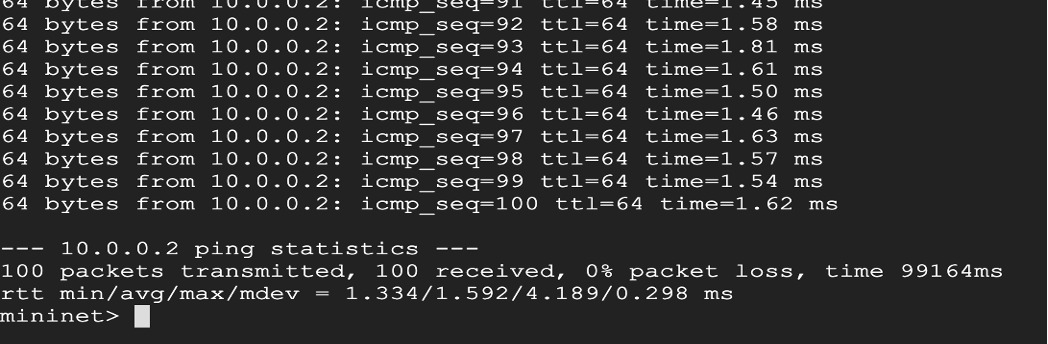
resend\_packet(self,packet in, out port)

Here, flow starts from the \_\_init\_\_ function goes to \_handle\_packet and act\_like\_hub and lastly to resend\_packet as \_handle\_packet and act\_like\_hub will call resend\_packet.

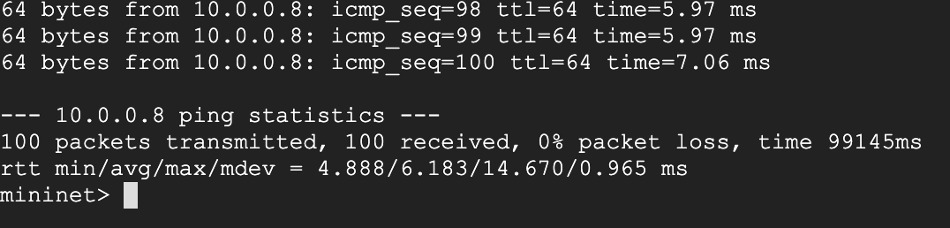
Q.2 Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2). How long does it take (on average) to ping for each case? What is the difference, and why?

->

h1 ping -c100 h2 is 1.592 ms



h1 ping -c100 h8 is 6.183 ms

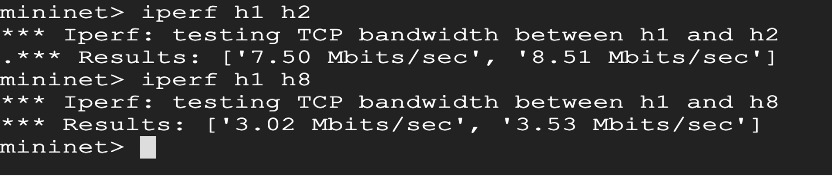


h8 is far away from h2 as compared from h1. So, in order to travel to h8, multiple jumps and hops are taken. So h8 takes more time.

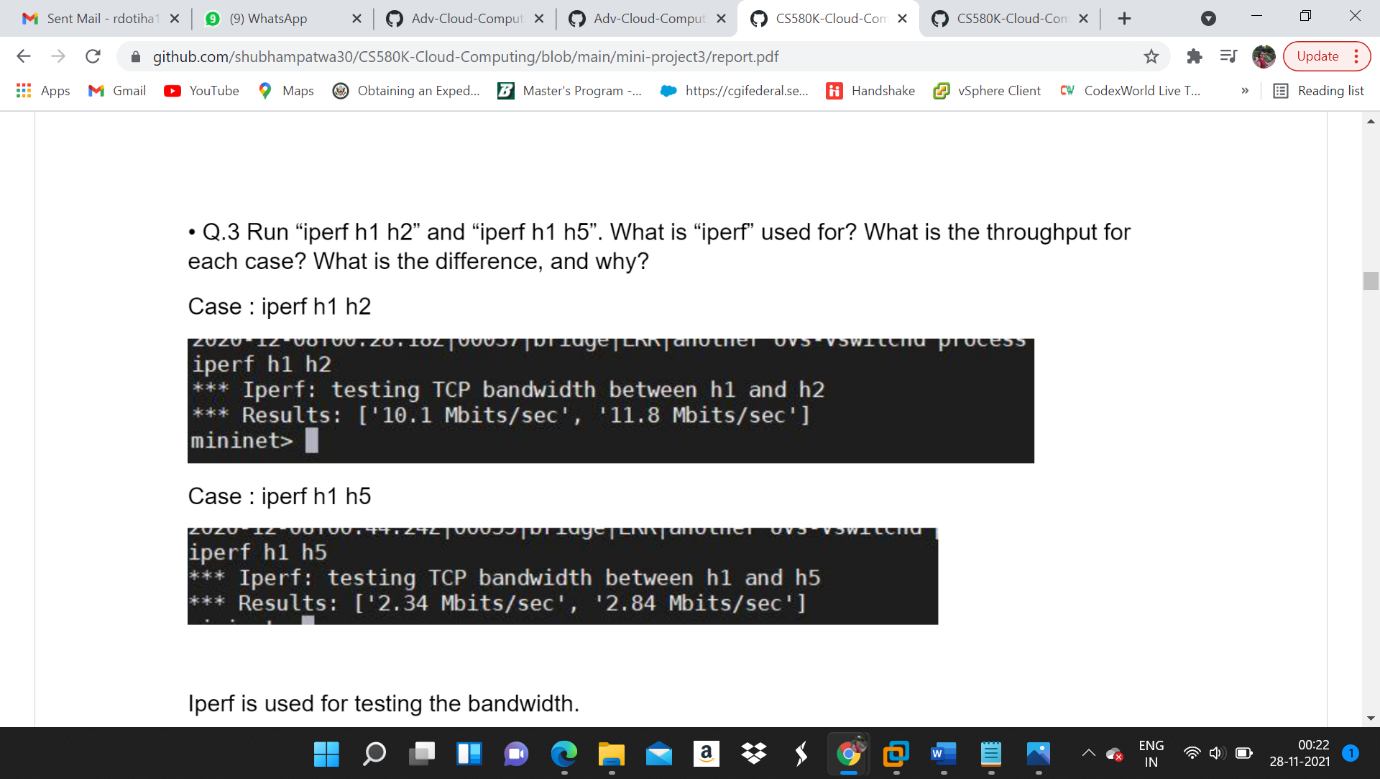
Q.3 Run “iperf h1 h2” and “iperf h1 h8”. What is “iperf” used for? What is the throughput for each case? What is the difference, and why?

->

iperf h1 h2 is 3.02 Mbits/sec, 3.53 Mbits/sec



iperf h1 h8 is 2.34 Mbits/sec, 2.84 Mbits/sec



Testing of bandwidth is done by iperf. Iperf is a cross-platform tool which is widely used for the network performance measurement and tuning.

There is a difference between values as h8 is located far away from h2 and have to travel more than h2 so bandwidth is reduced.

Q.4 Which of the switches observe traffic? Please describe your way for observing such traffic on switches (hint: adding some "print" functions in the “of\_tutorial” controller).

->

All the switches from s1 to s7 face the issue of traffic. Traffic can be observed by adding print functions in between. By running the mininet in one terminal and then pinging the same on the another terminal method, can be used to observe the traffic.

**3 Task III : MAC learning Controller**

Q.1 Please describe how the above code works, such as how the "MAC to Port" map is established. You could use a ‘ping’ example to describe the establishment process (e.g., h1 ping h2).

->

Switch is used to forward packet to the controller

The tasks performed by the controller as following –

If ( Host sends the first packet)

MAC address and source port is saved and learnt.

Further, port number gets attached to MAC address and packet is sent using the same port.

Else (not the first time)

If(port associated with MAC is known)

Packet is sent through that port

Else (Port is not known)

Packet is sent to all ports except the input port.

Q.2 (Please disable your output functions, i.e., print, before doing this experiment) Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2). How long did it take (on average) to ping for each case? Any difference from Task II (the hub case)?

->

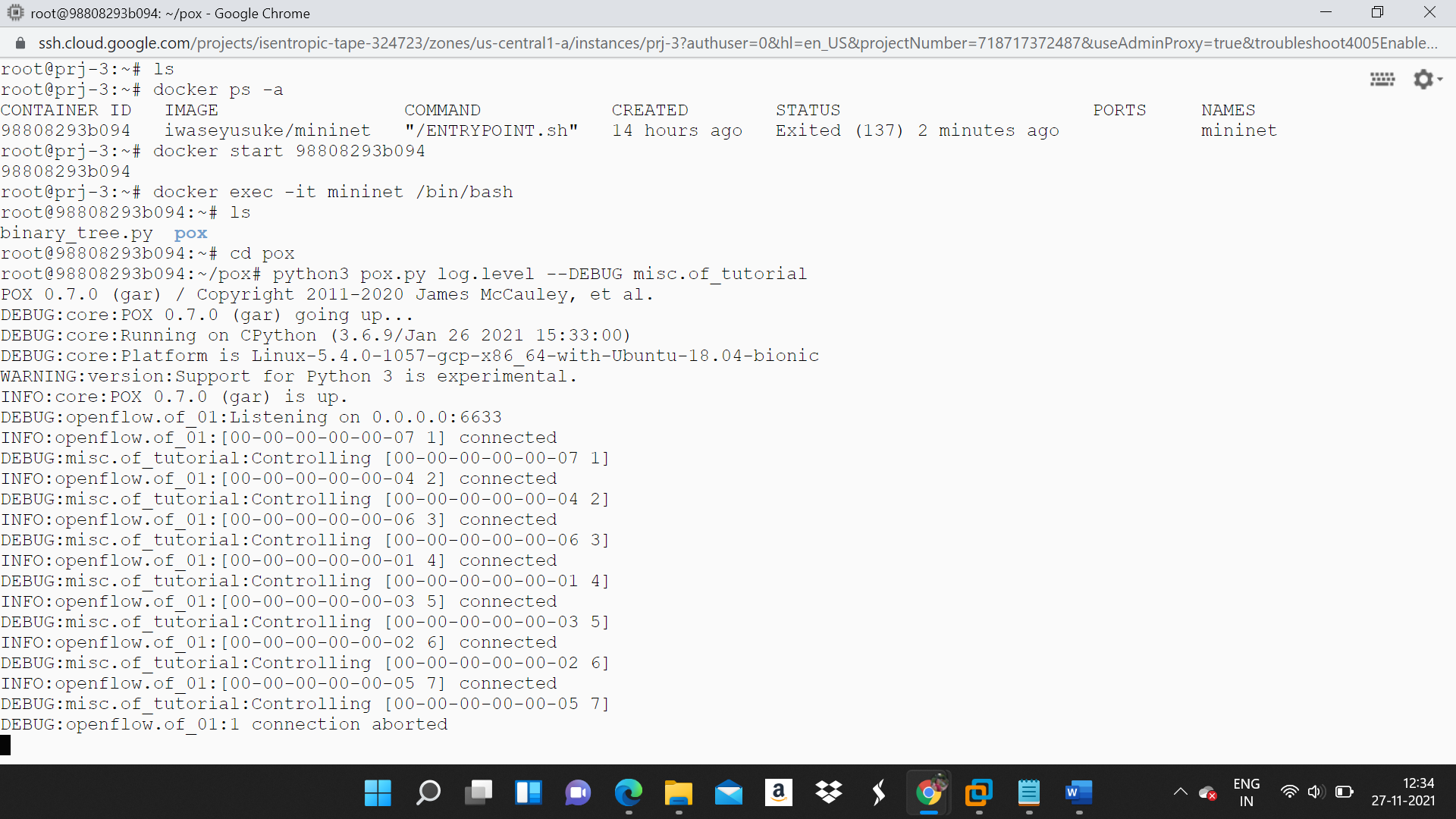
h1 ping -c100 h2

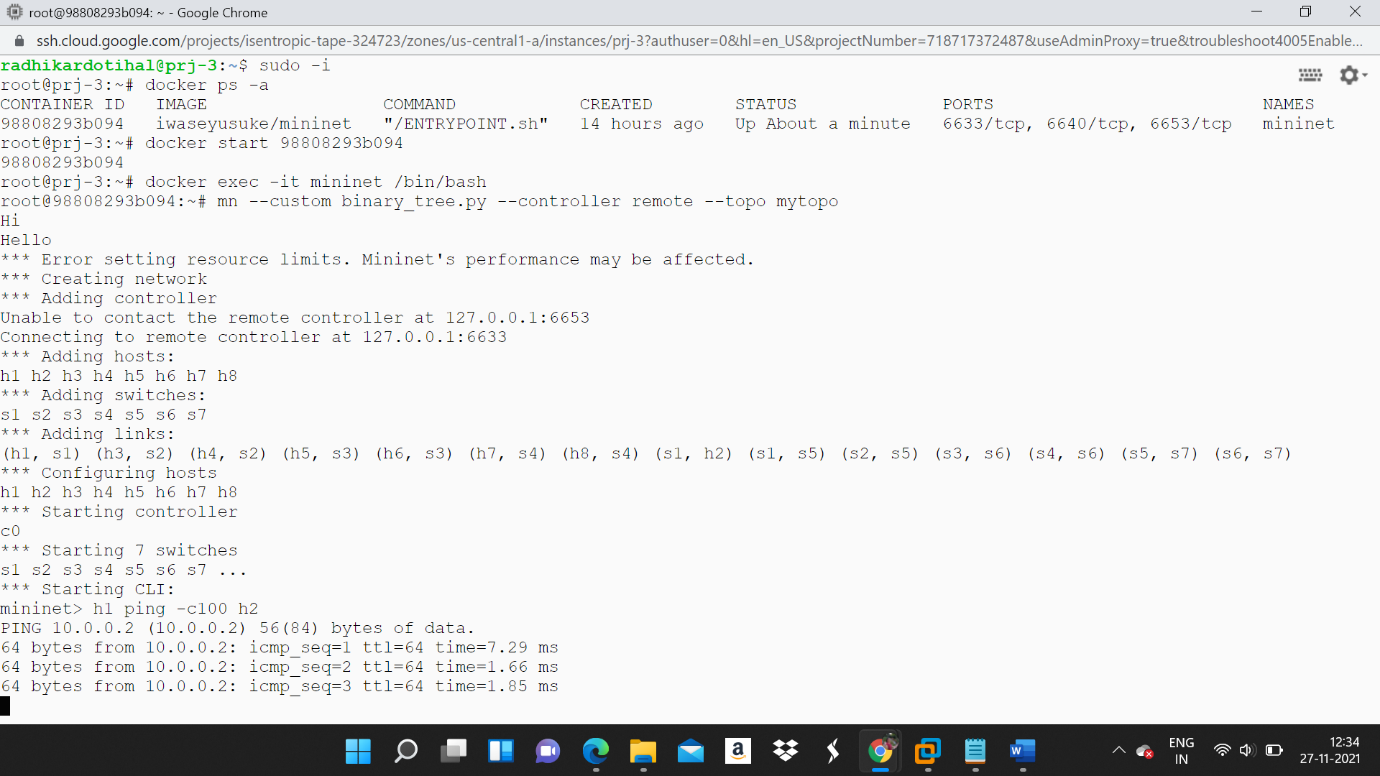
Average Time is = 1.962

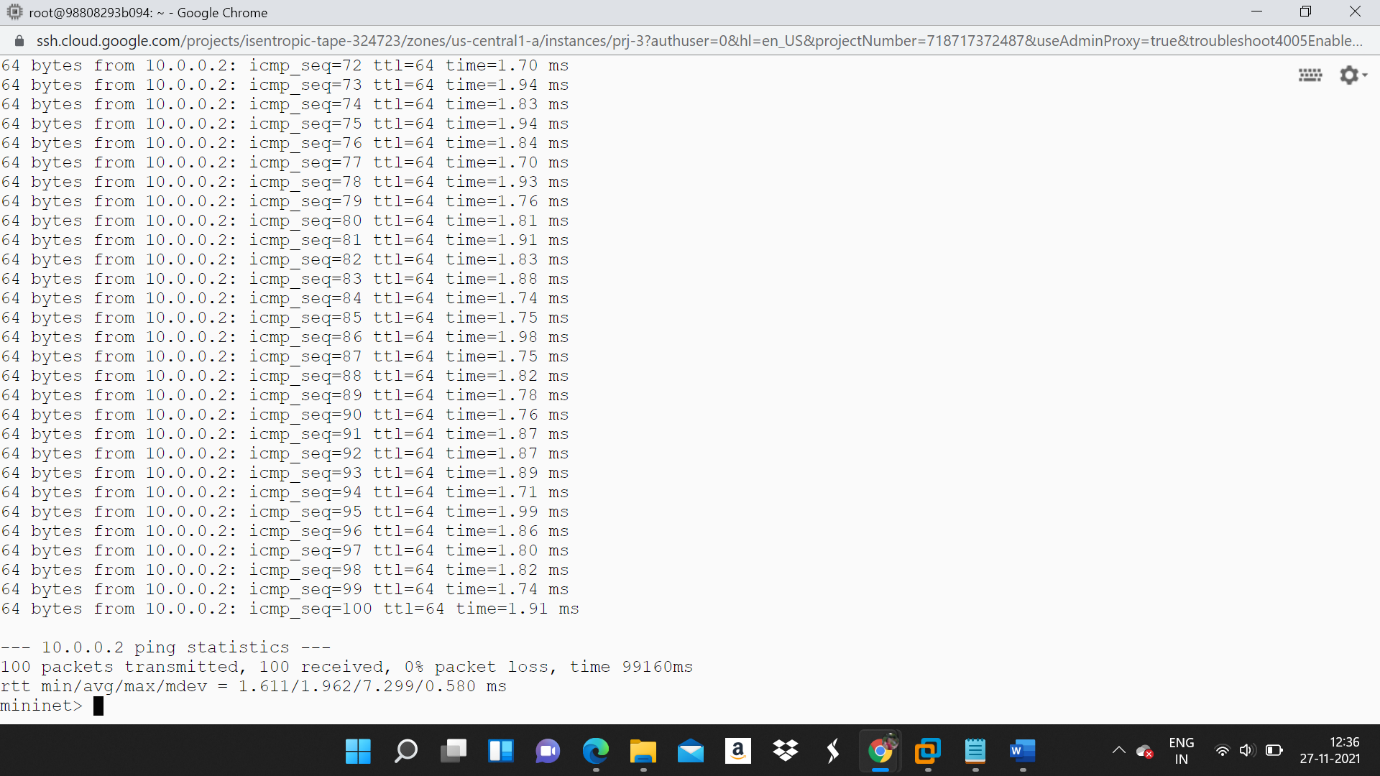
H1 ping -c100 h8

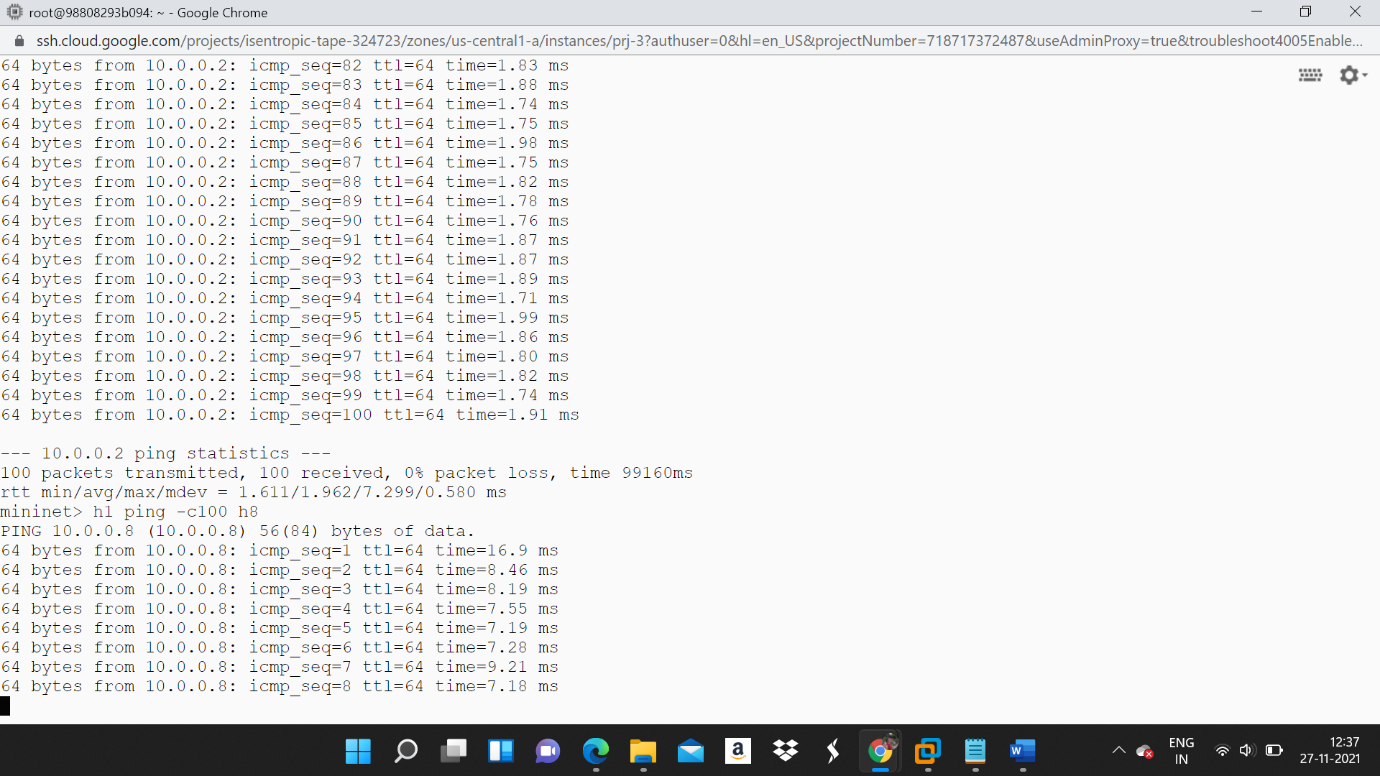
Average Time is = 7.299

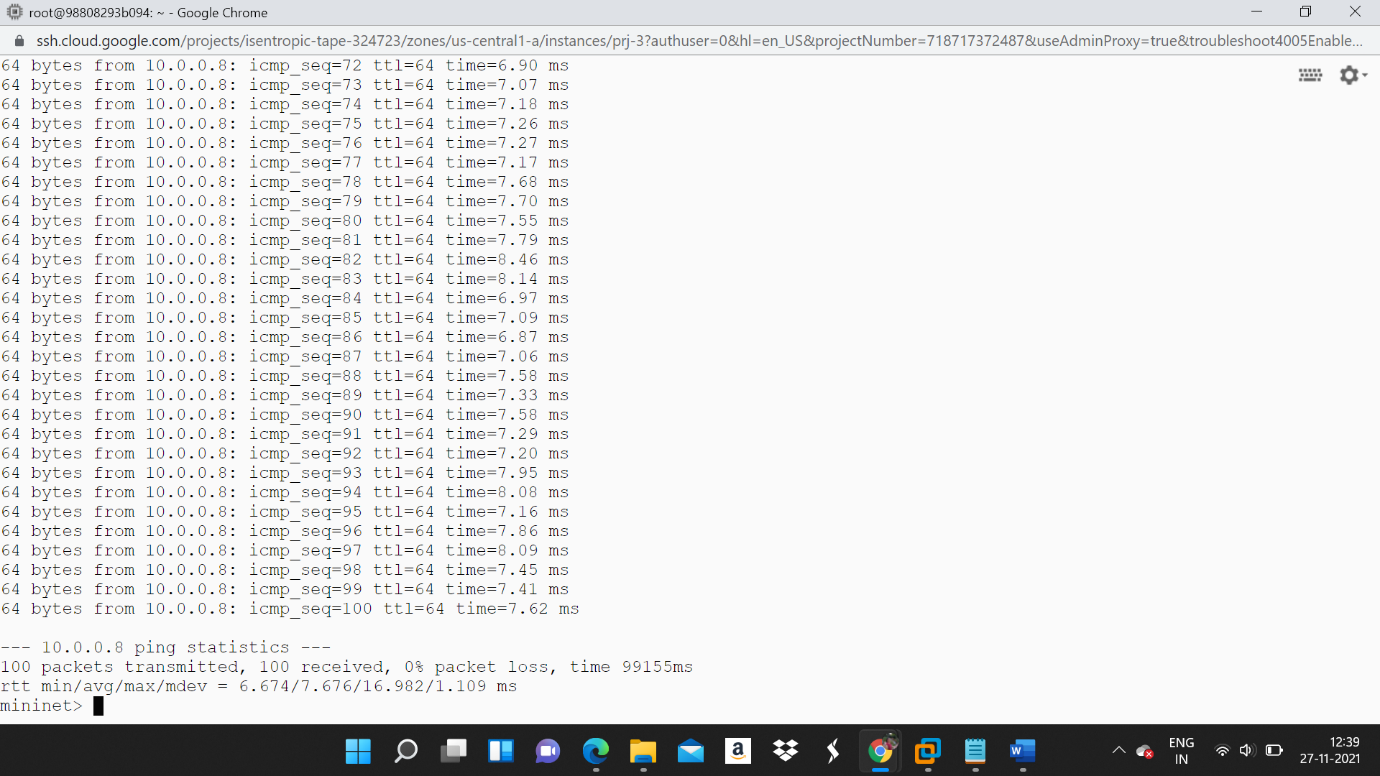
Average time is increased as compared to the task 2







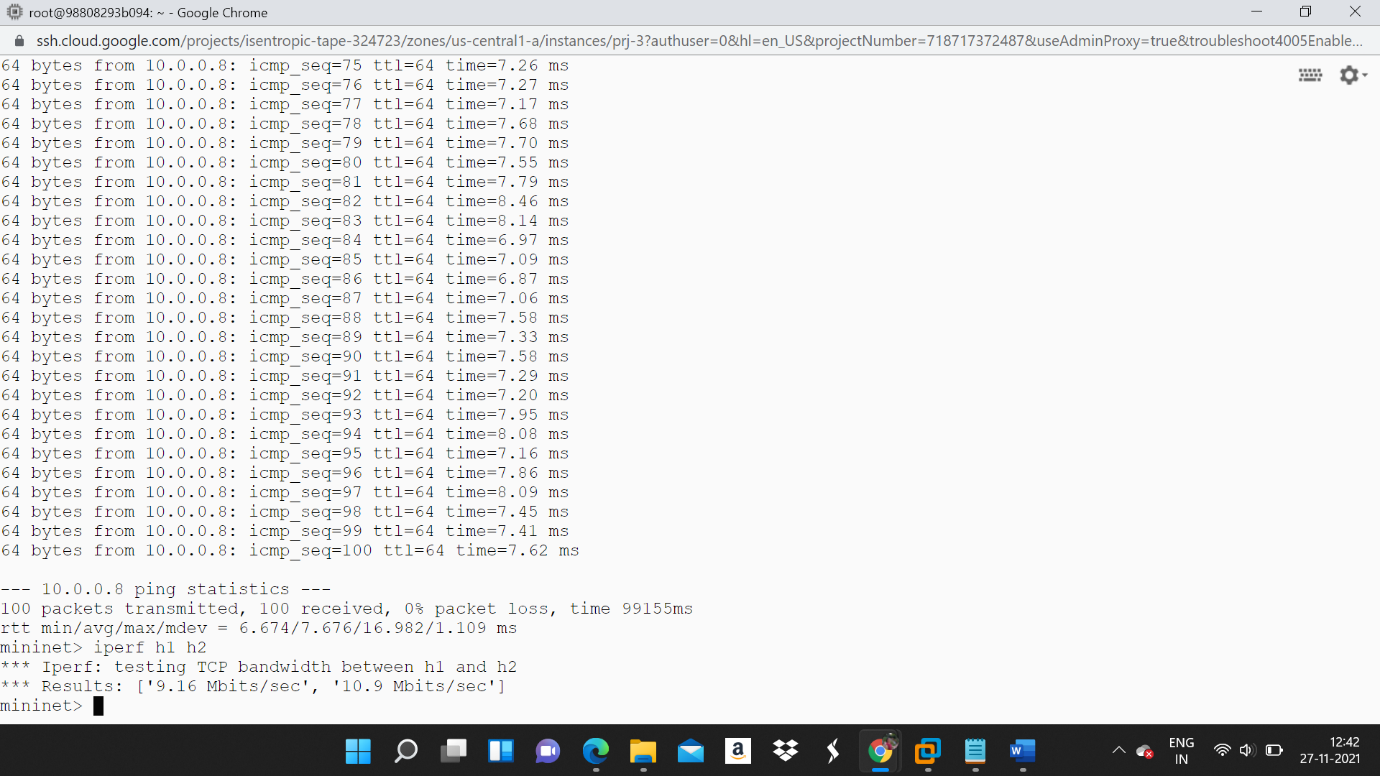




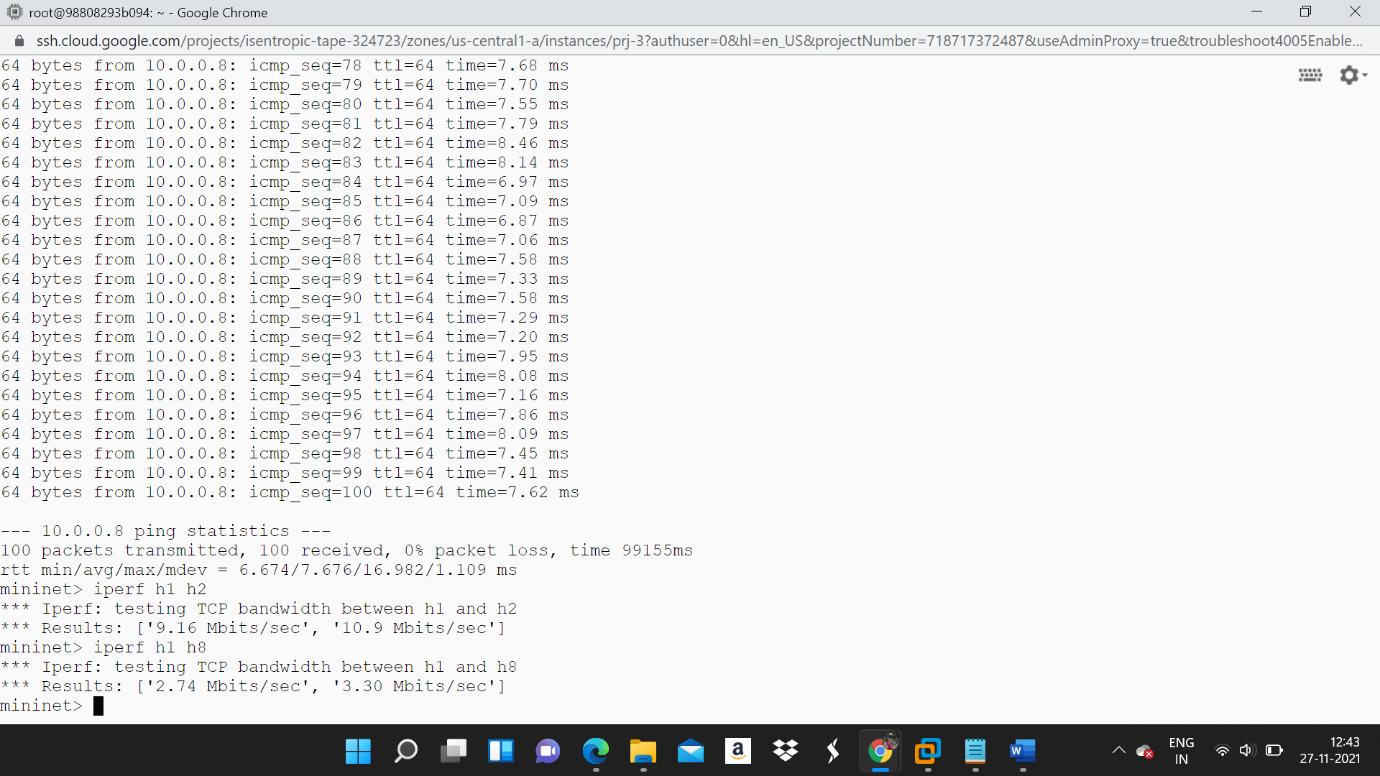
Q.3 Run “iperf h1 h2” and “iperf h1 h8”. What is the throughput for each case? What is the difference from Task II?

->

iperf h1 h2



iperf h1 h8



Throughput has increased in both the case as compared with the task 2

**4 Task IV: MAC learning controller with OpenFlow rules**

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)?

->

h1 ping -c100 h2

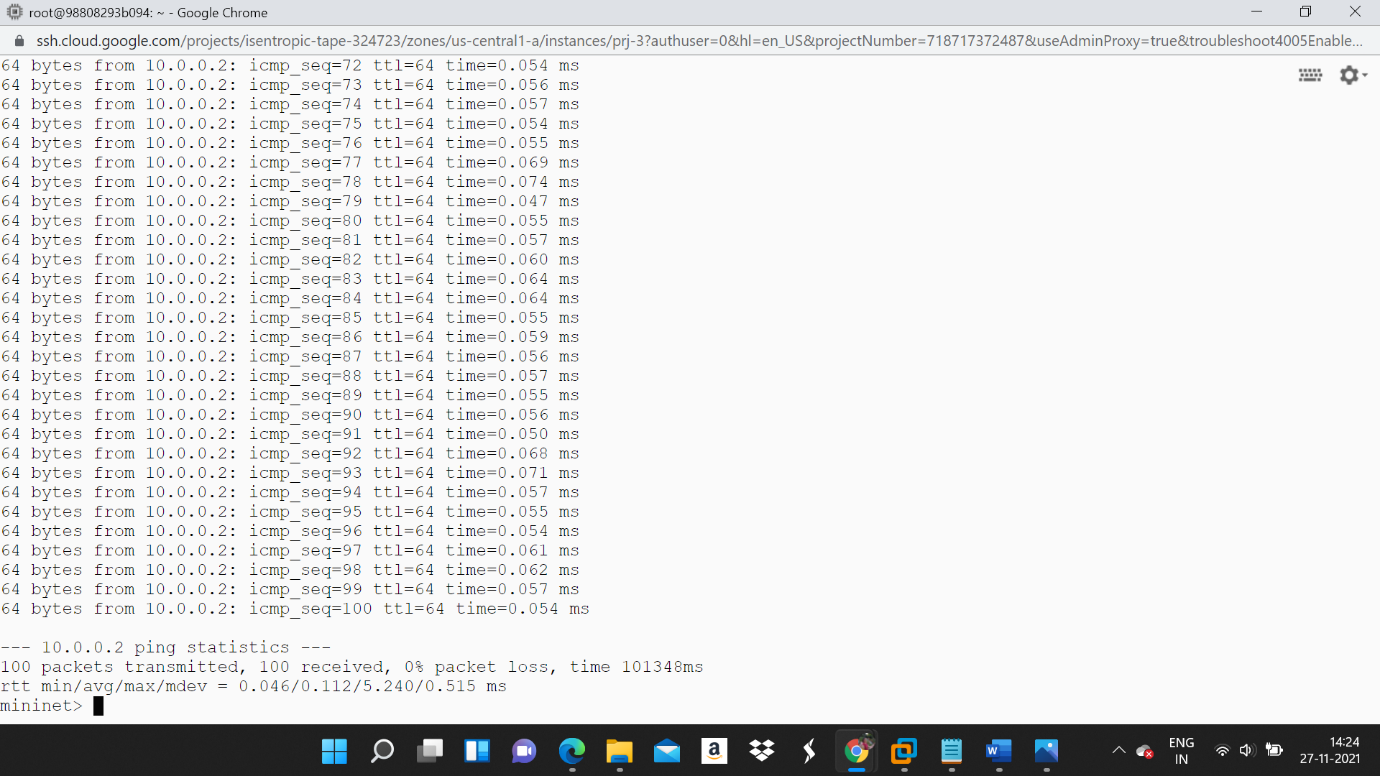
Average Time is 0.112

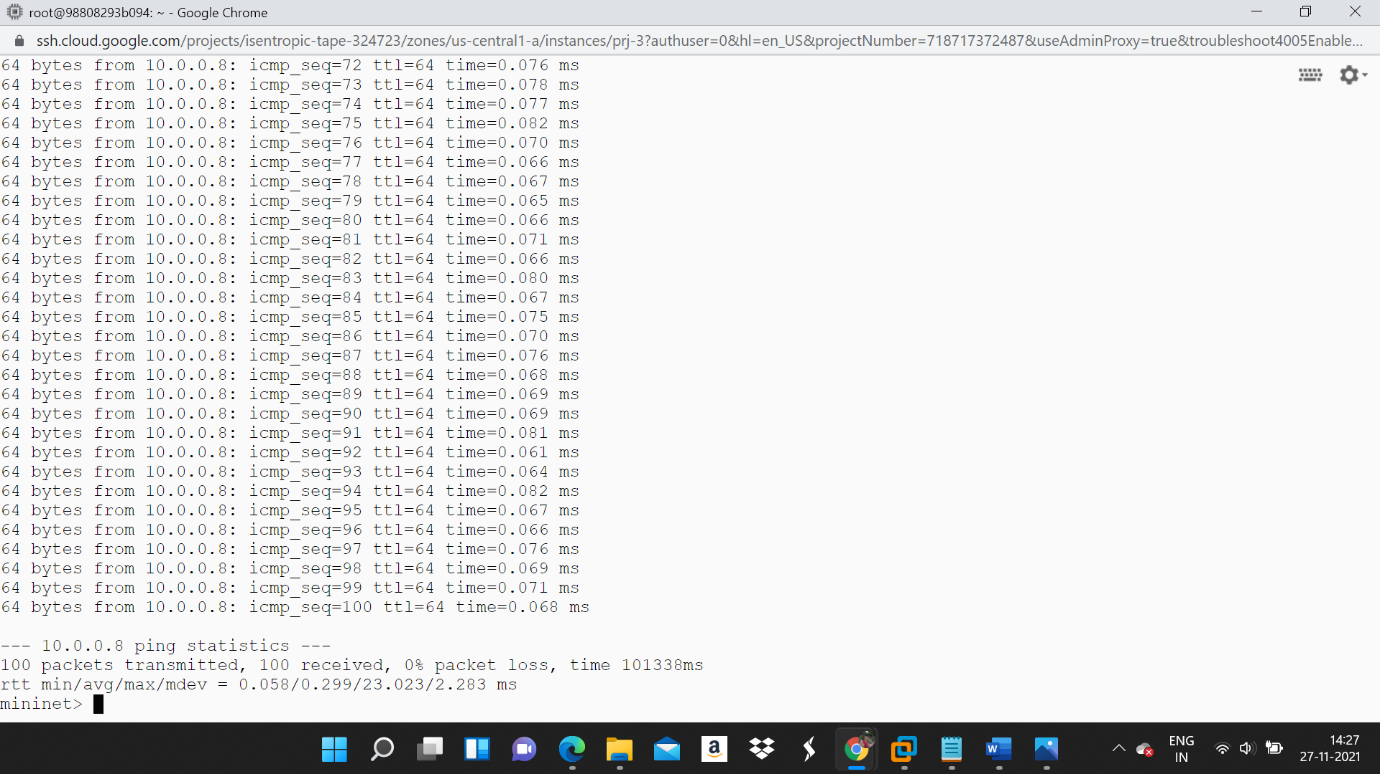
H1 ping -c100 h8

Average Time is 0.299

Difference between these two is 0.187

Thus the average time of task 4 is always better than the average time of task 3



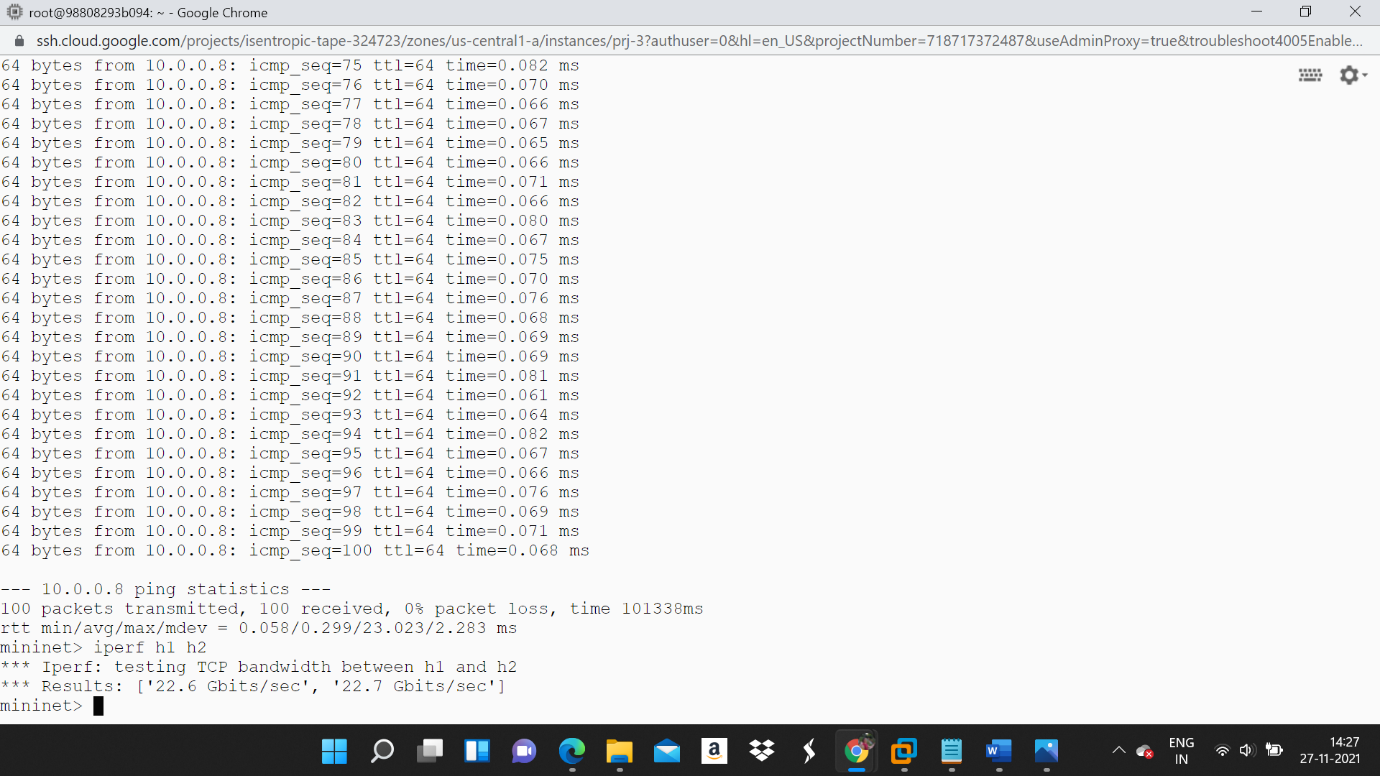


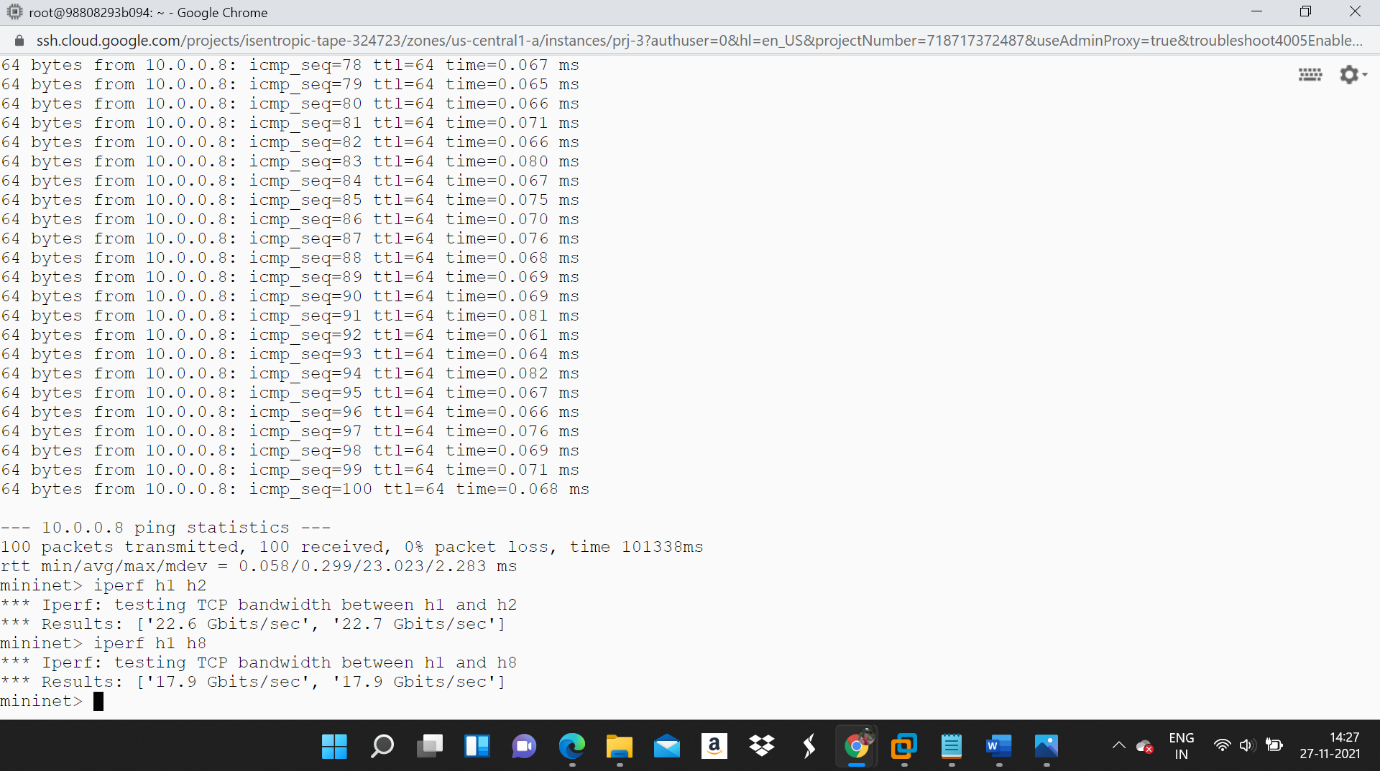
Q.2 Run “iperf h1 h2” and “iperf h1 h8”. What is the throughput for each case? What is the difference from Task III?

->

iperf h1 h2 is 22.6 Gbits/sec 22.7 Gbits/sec

iperf h1 h8 is 17.9 Gbits/sec 17.9 Gbits/sec





Throughput of task 4 is better than the throughput of task 3 as we can see the throughput of task 4 is in Gbites/sec whereas throughput of task 2 & 3 was in Mbits/sec.

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

->

The results of task 4 is better than the result of task 3.

Task 4 performance is better than task 3 and task 3 performance is better than task 2.

Throughput of task 3 is better than throughput of task 2 and throughput of task 2 is better than throughput of task 1.

Open Flow controller is better than Mac learning controller and Mac learning controller is better than SDN controller.

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

->

Commands to perform pingall is –

mn –custom binary\_tree.py –topo mytopo –test pingall & > pingall.txt

cat pingall.txt

OUTPUT –

\*\*\* Error setting resource limits. Mininet's performance may be affected.

\*\*\* Creating network

\*\*\* Adding controller

\*\*\* Adding hosts:

h1 h2 h3 h4 h5 h6 h7 h8

\*\*\* Adding switches:

s1 s2 s3 s4 s5 s6 s7

\*\*\* Adding links:

(h1, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (s1, h2) (s1, s5) (s2, s5) (s3, s6) (s4, s6) (s5, s7) (s6, s7)

\*\*\* Configuring hosts

h1 h2 h3 h4 h5 h6 h7 h8

\*\*\* Starting controller

c0

\*\*\* Starting 7 switches

s1 s2 s3 s4 s5 s6 s7 ...

\*\*\* Waiting for switches to connect

s1 s2 s3 s4 s5 s6 s7

\*\*\* 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)

\*\*\* Stopping 1 controllers

c0

\*\*\* Stopping 14 links

..............

\*\*\* Stopping 7 switches

s1 s2 s3 s4 s5 s6 s7

\*\*\* Stopping 8 hosts

h1 h2 h3 h4 h5 h6 h7 h8

\*\*\* Done

completed in 14.791 seconds

Q.5 Dump the output of the flow rules using “ovs-ofctl dump-flows” (in your container, not mininet). How many rules are there for each OpenFlow switch, and why? What does each flow entry mean (select one flow entry and explain)?

->

Below are the steps

mn –custom binary\_tree.py –controller remote –topo mytopo

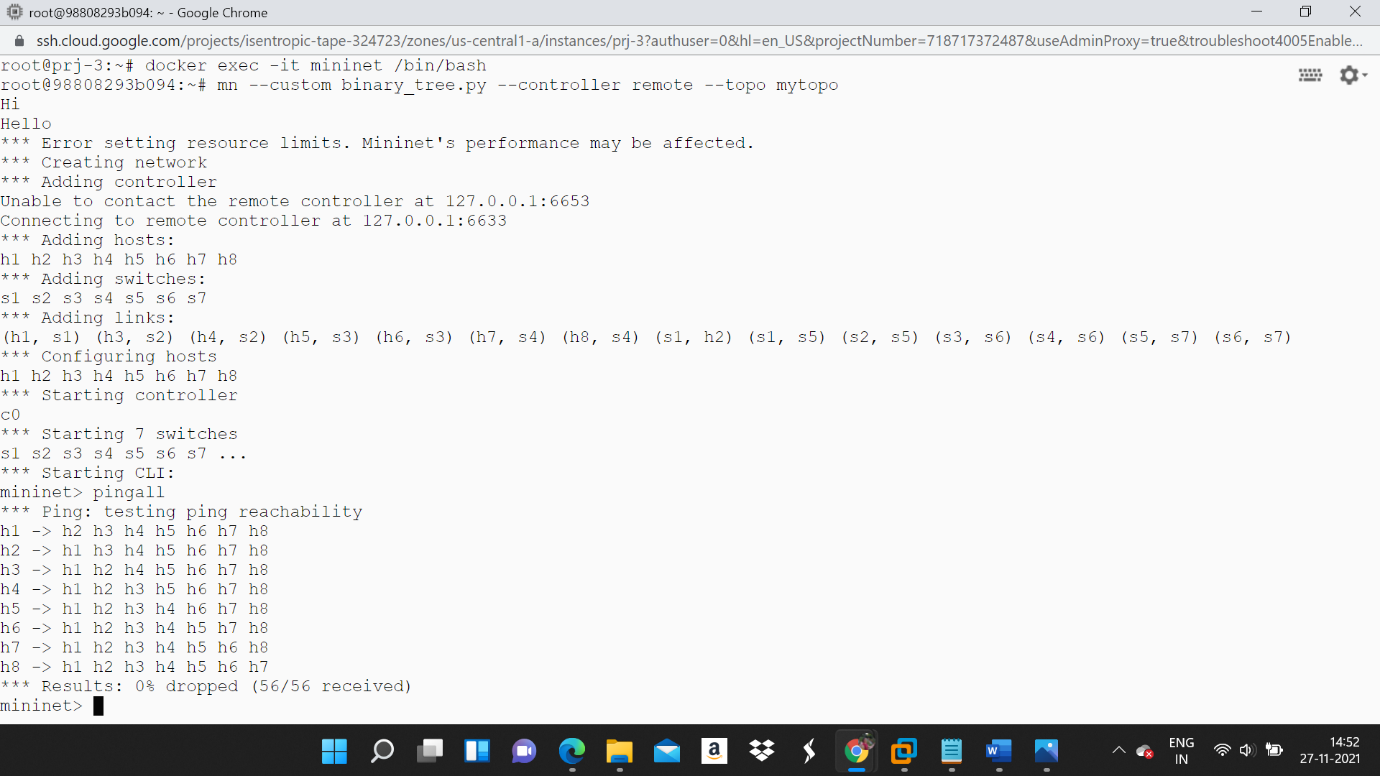
mininet> pingall

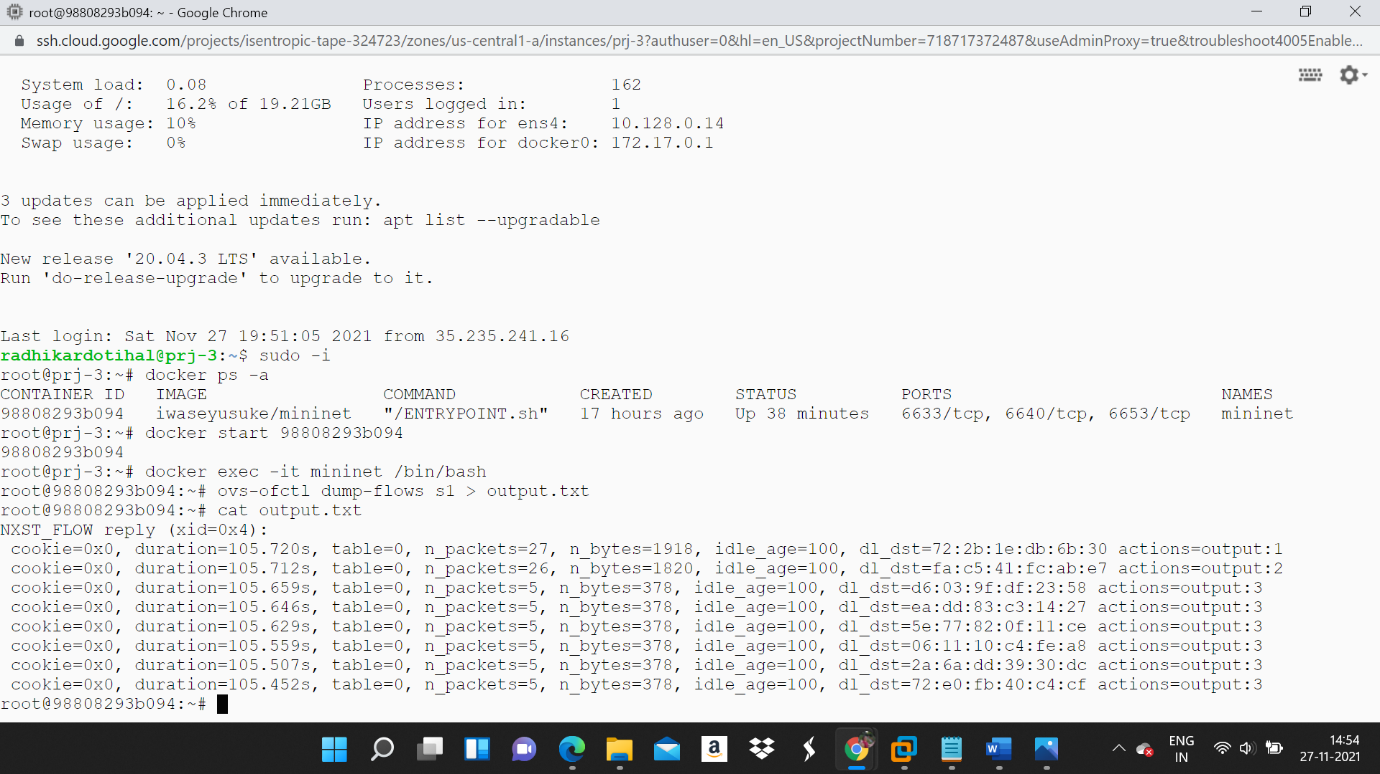
Open second terminal with an active controller

$ ovs-ofctl dump-flows s1 > flow.txt

$ cat flow.txt

OUTPUT –





As we can see in the output,

Cookie - stores temporary information.

N packets – total no. of packets which have matched the entry from the flow table

N Bytes – Total no. of bytes from packets that have matched the entry also

D1 dst – destination of MAC address.

**5 Task V: Layer-3 routing**

We need to install IP matching rules on switch s7. Other switches stay as MAC learning switches.

The main goal is to allow all the hosts from h1 to h7 ping h8 ad switch forward packets. This is done by IP matching flow rules and not MAC learning rules.

act\_like\_switch function handles all the switches thus it can be changed/updated a bit by adding if else condition.

The required functionality can be achieved using ovs-ofctl

Steps are as following –

Set the flow table rules to match the IP.

Set ethernet type

