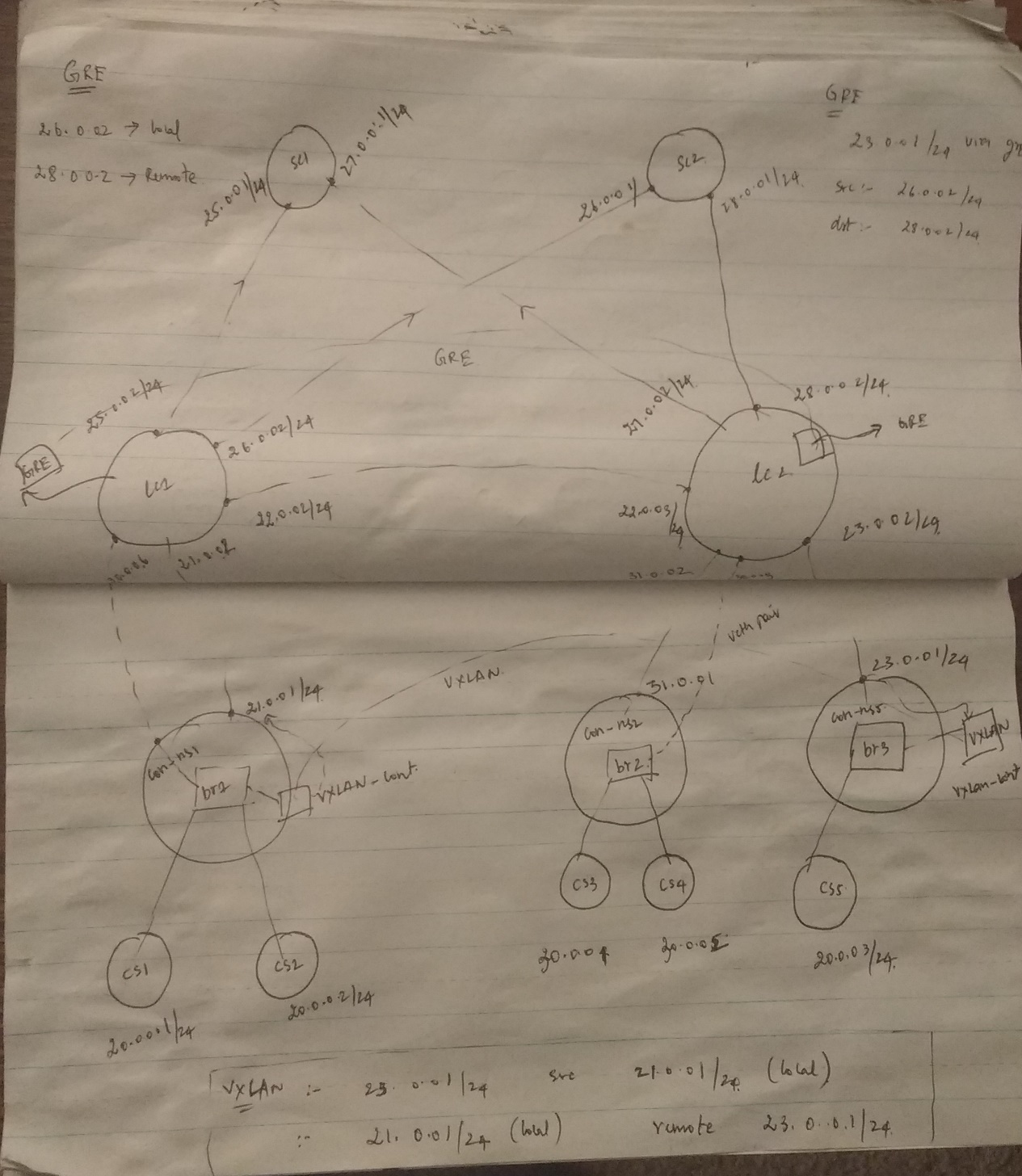
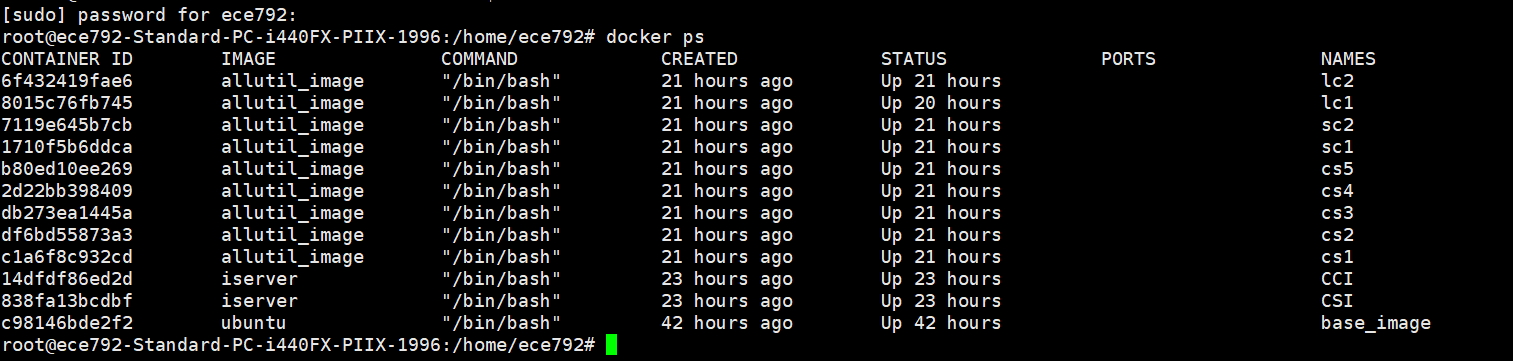
1. Bridge 1 and Bridge 3 are in same network. Traffic between the containers connected to bridge 1 and bridge 3 should use L2 Overlay

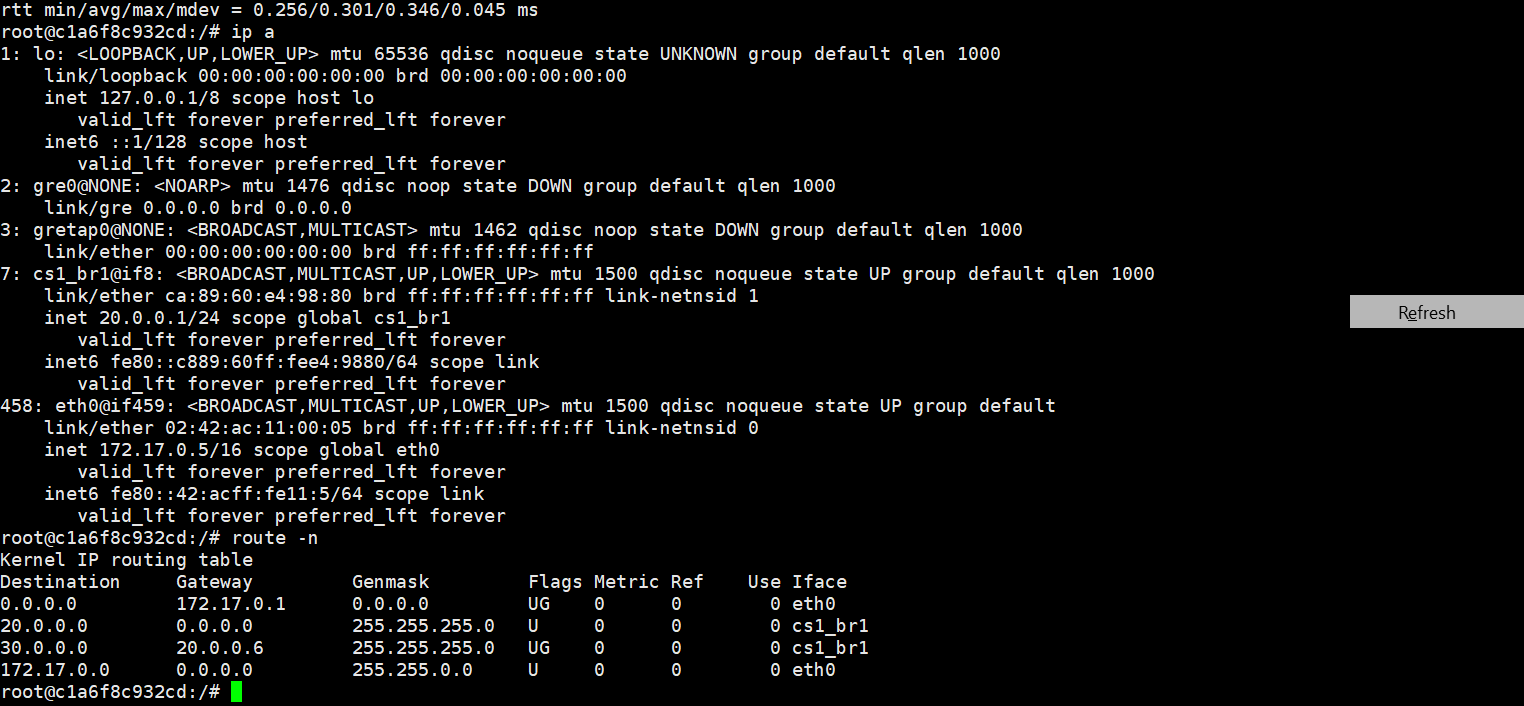
**Here is the topology and its connections:**

****

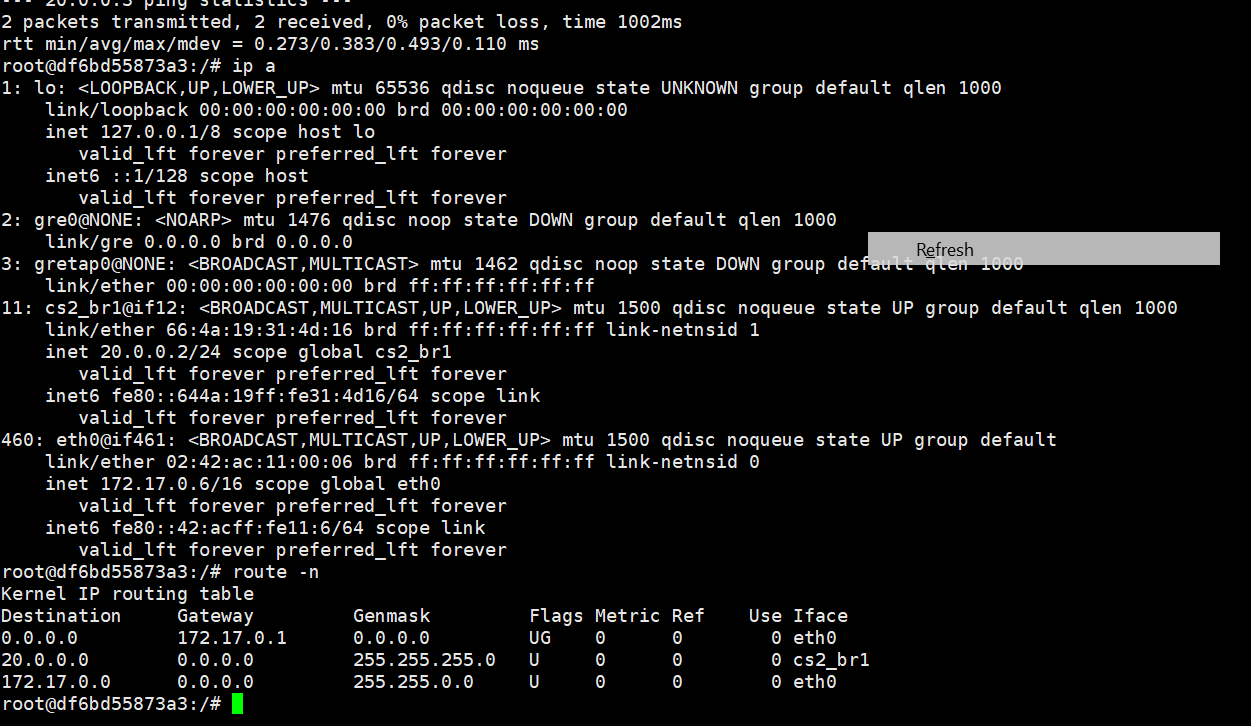
Here is the list of containers in the hypervisor 

Here is the snapshots of cs1 & cs2 & cs5 containers configs & routes:

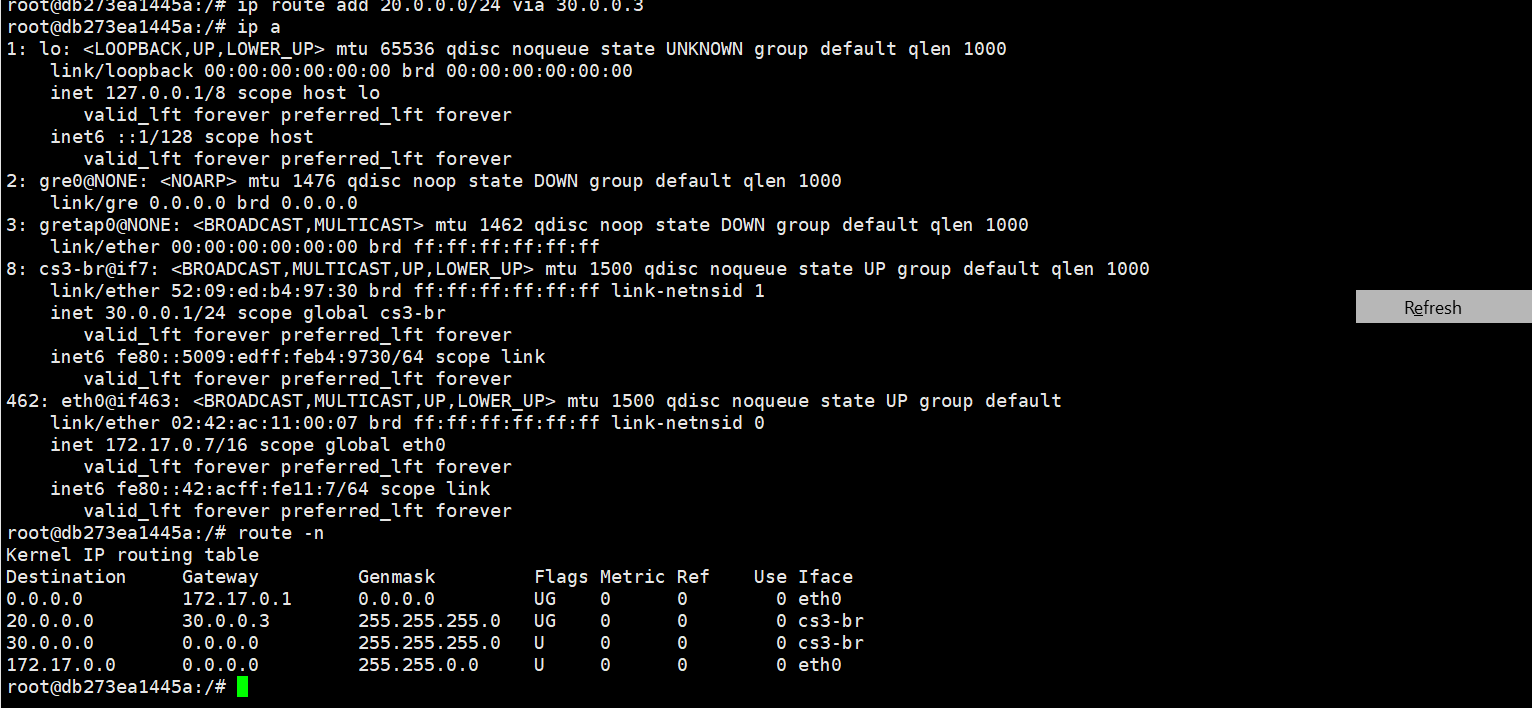
**CS1-config & Forwarding table :**

****

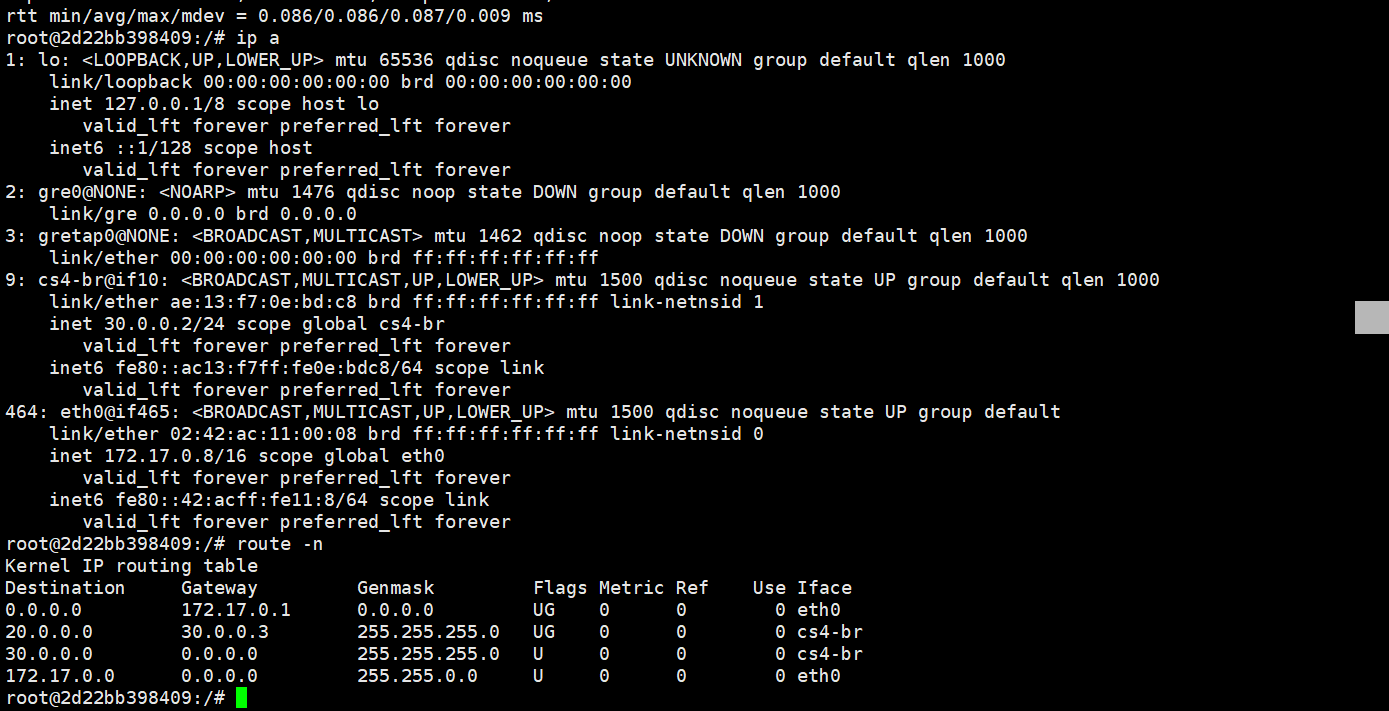
**CS2 config & forwarding table :**



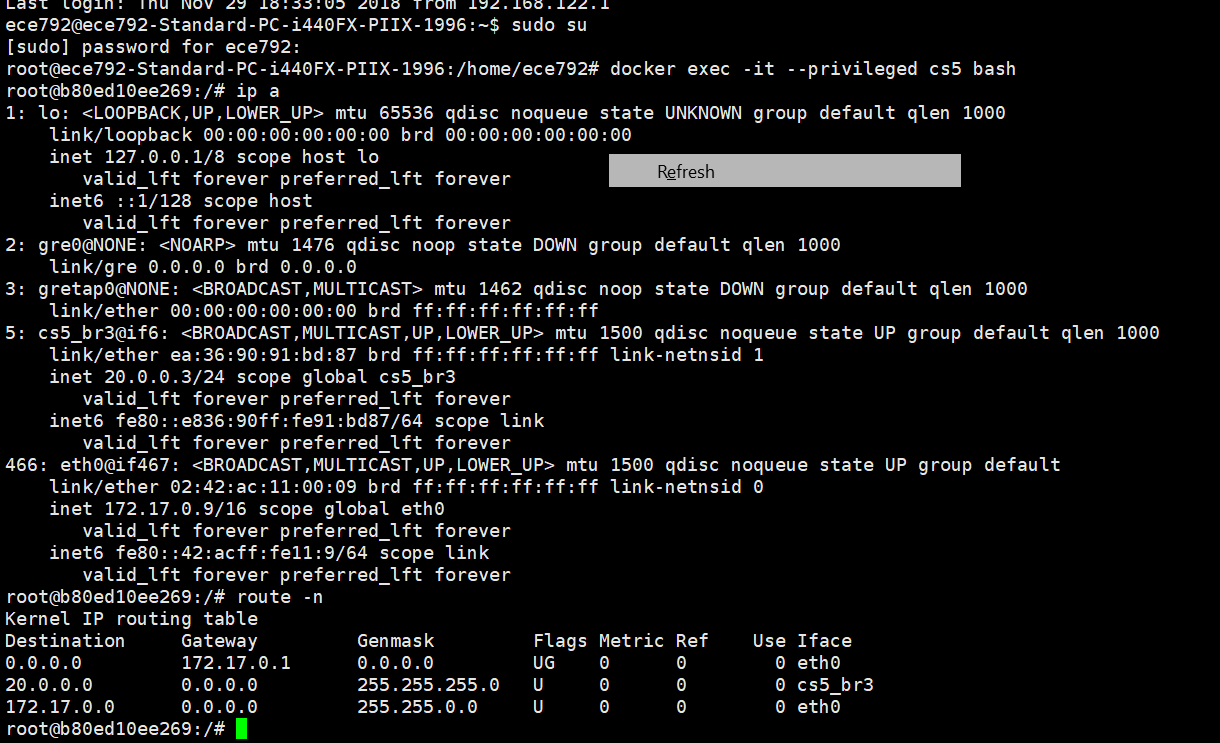
CS3 config & forwarding table :



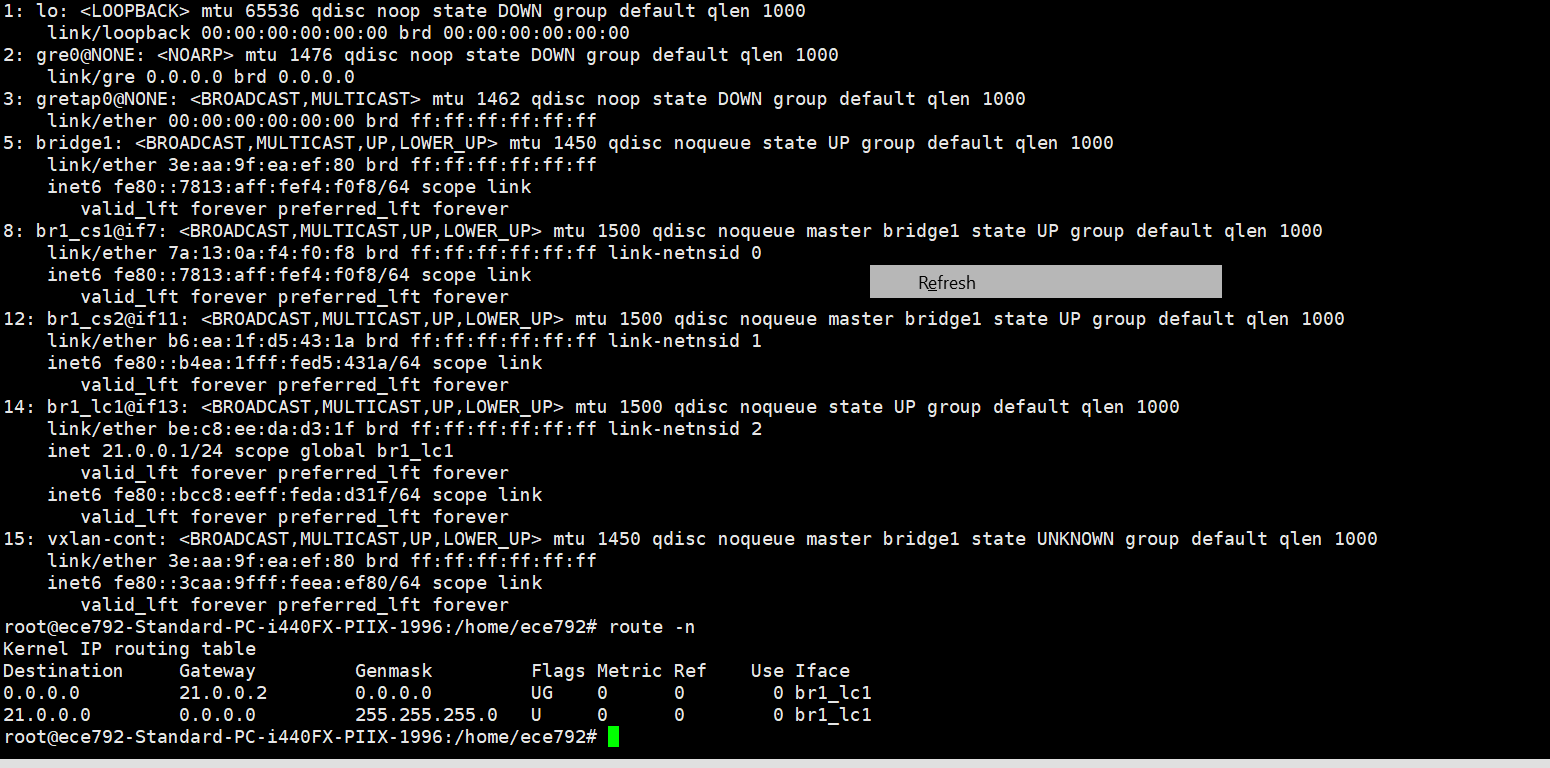
CS4 config & forwarding table:



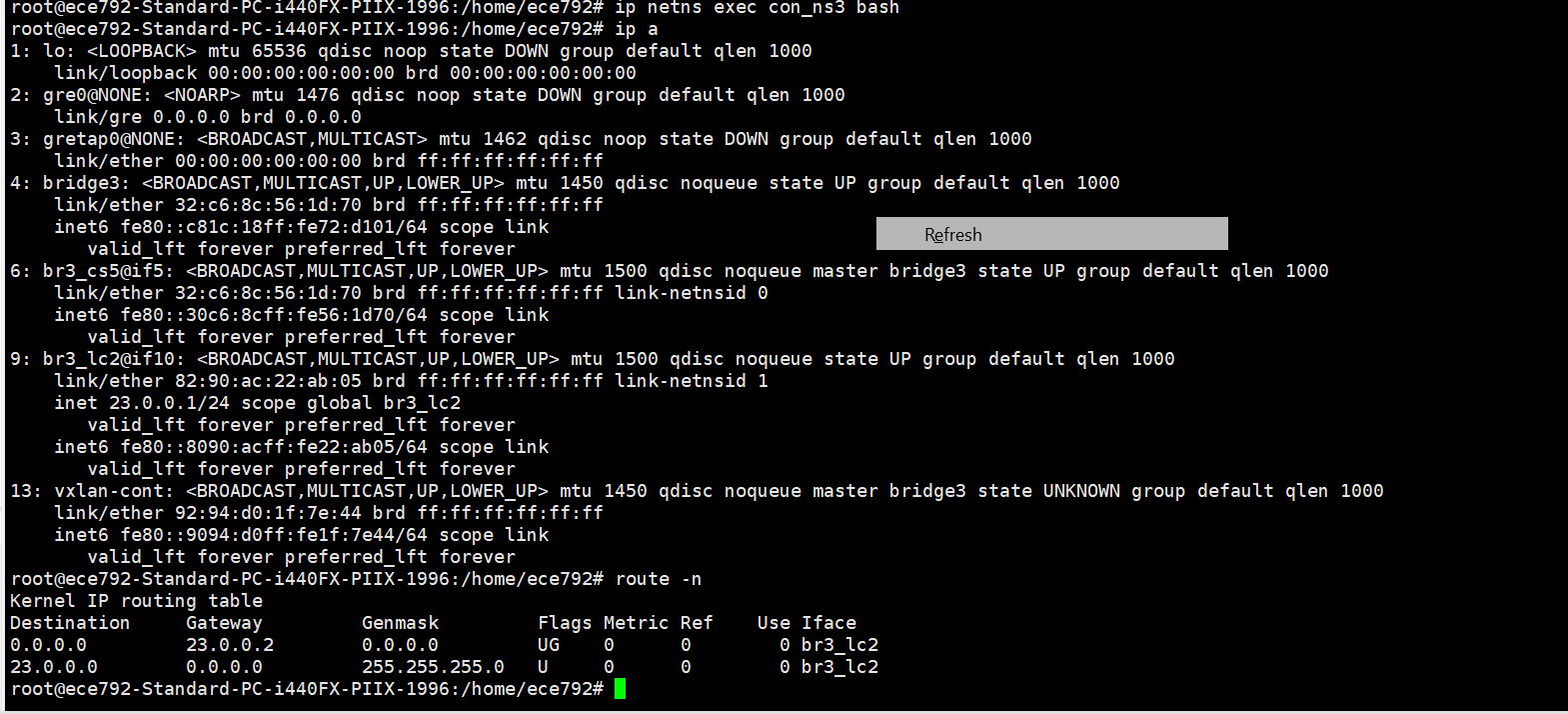
CS5 config & forwarding table :



**Here is the config of con\_ns1 namespace which has the bridge (bridge1) connected to cs1 and cs2**

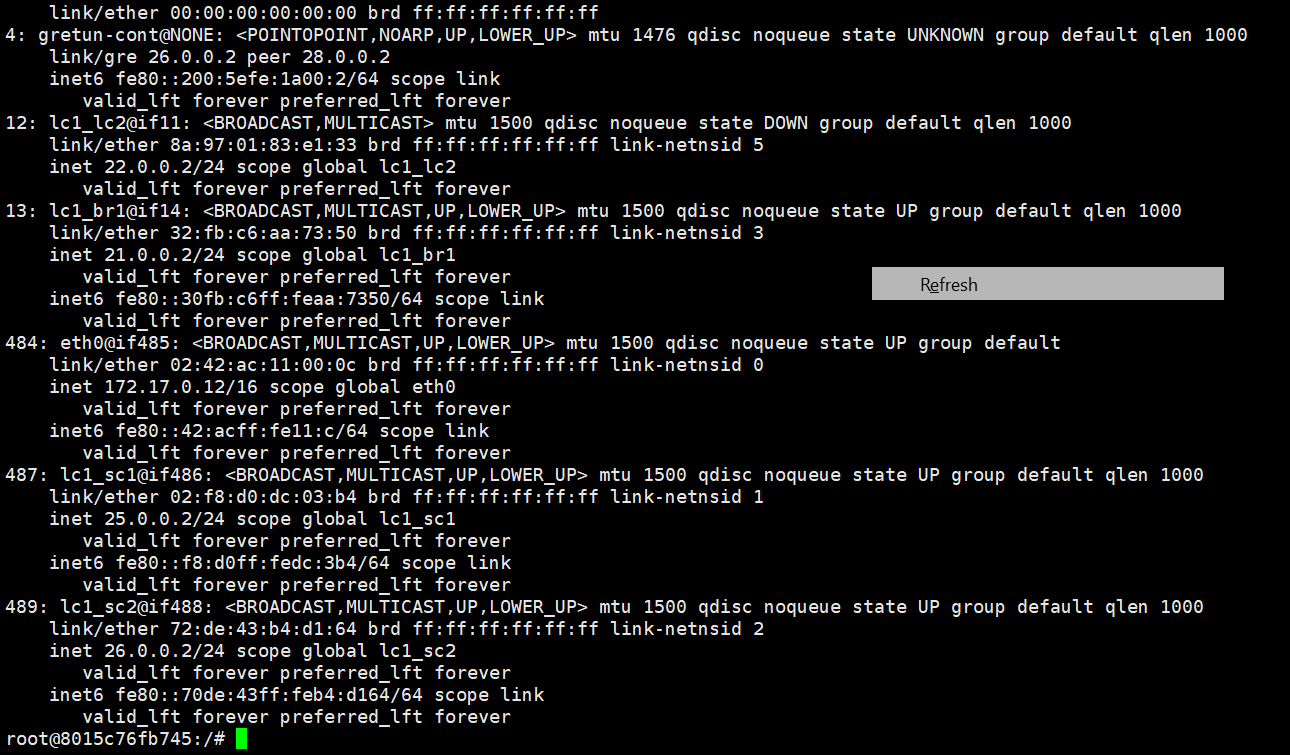


**Here is the config of con\_ns3 which has the bridge3(br3) connected to cs5 container:**

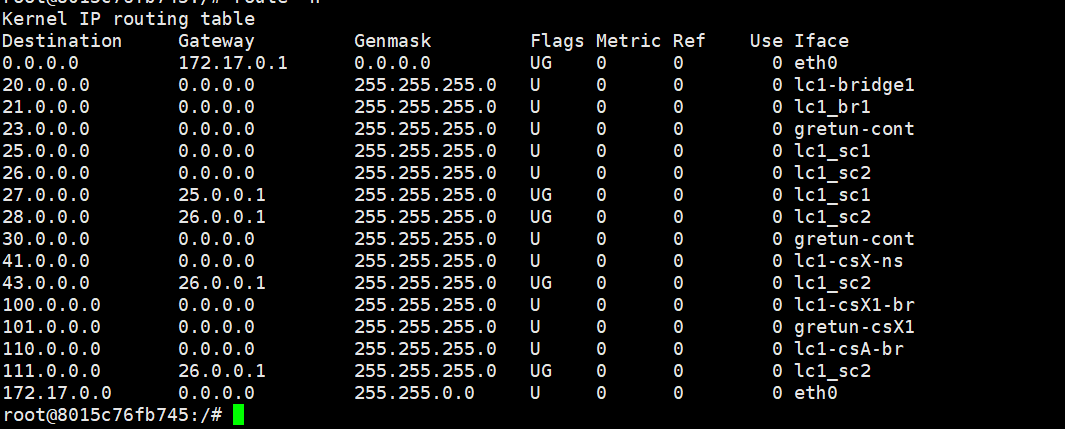
****

Here are the configs of lc1 & lc2 containers :

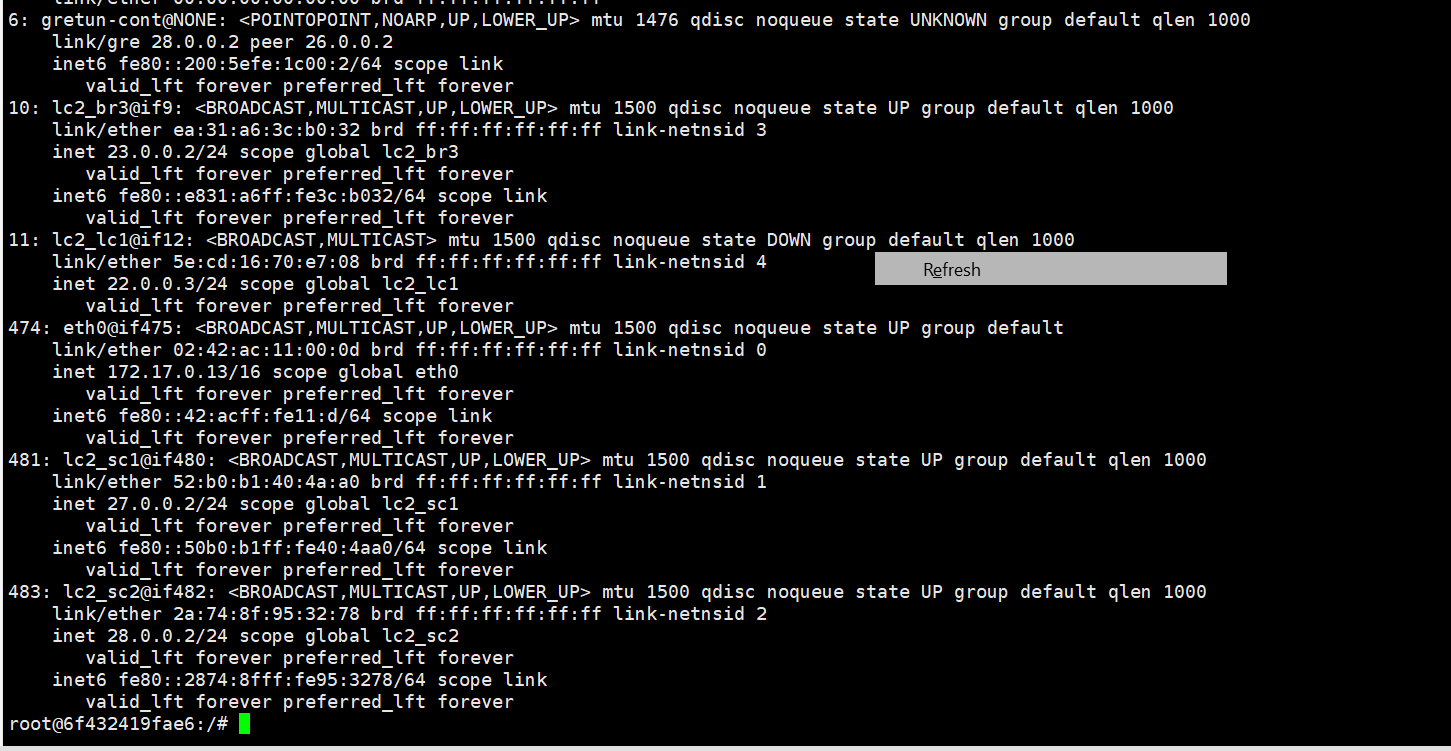
**Lc1 config (leaf container1)**

****

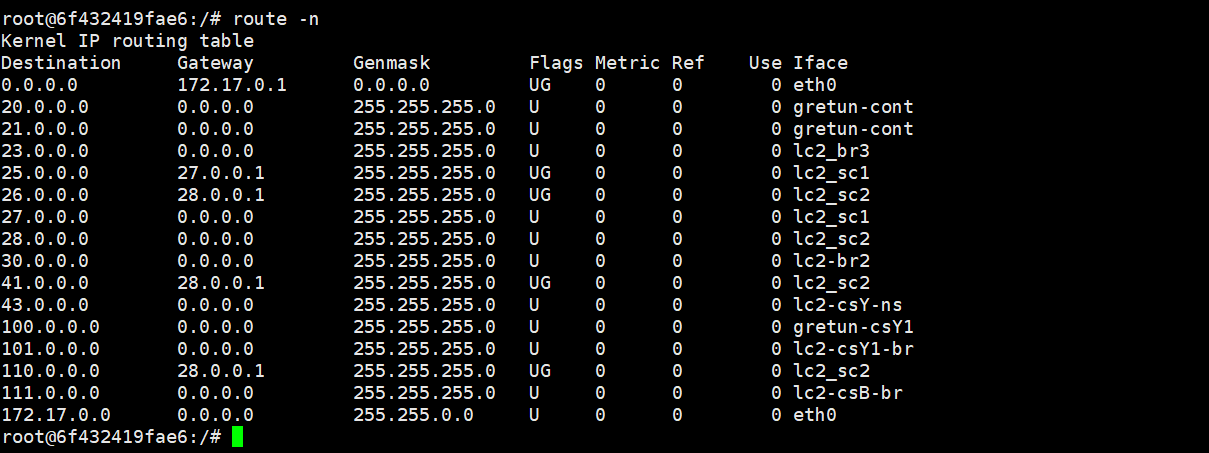
**Forwarding table of lc1:**

****

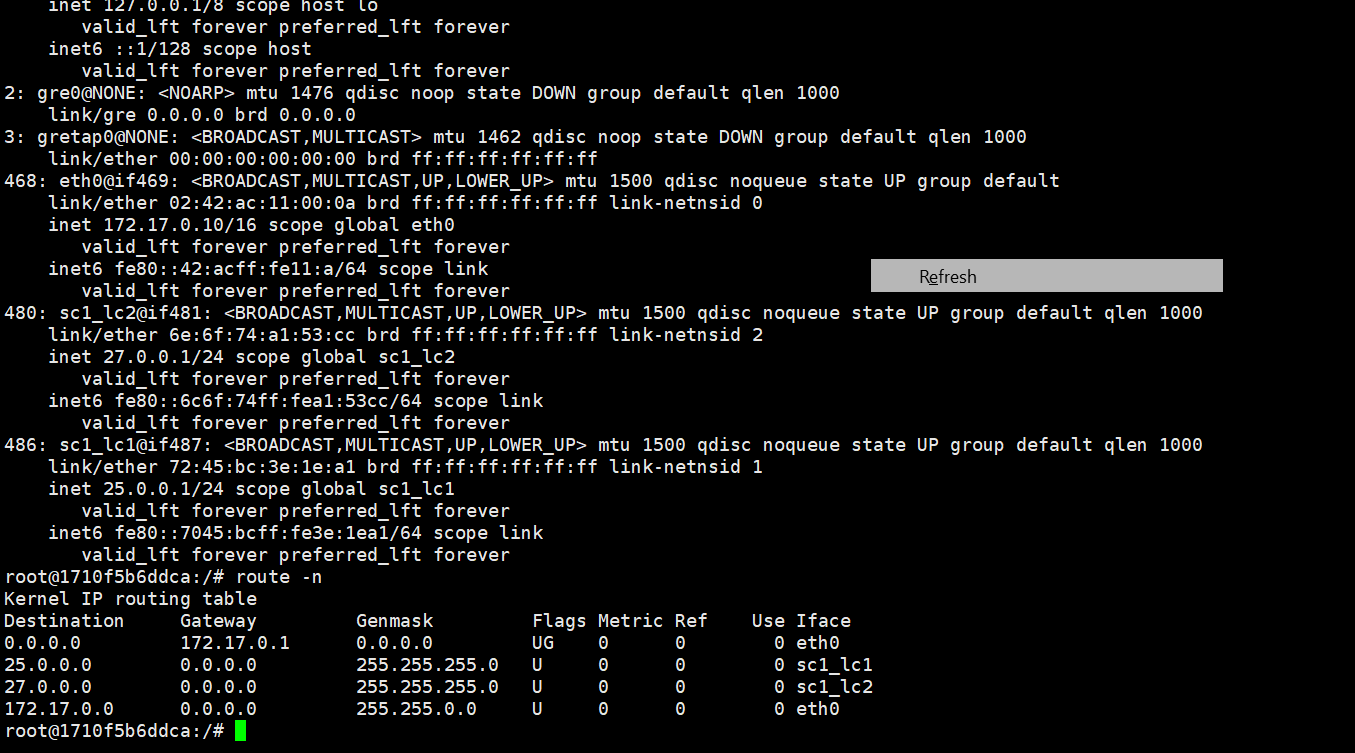
**LC2 config (lc2) config :**

****

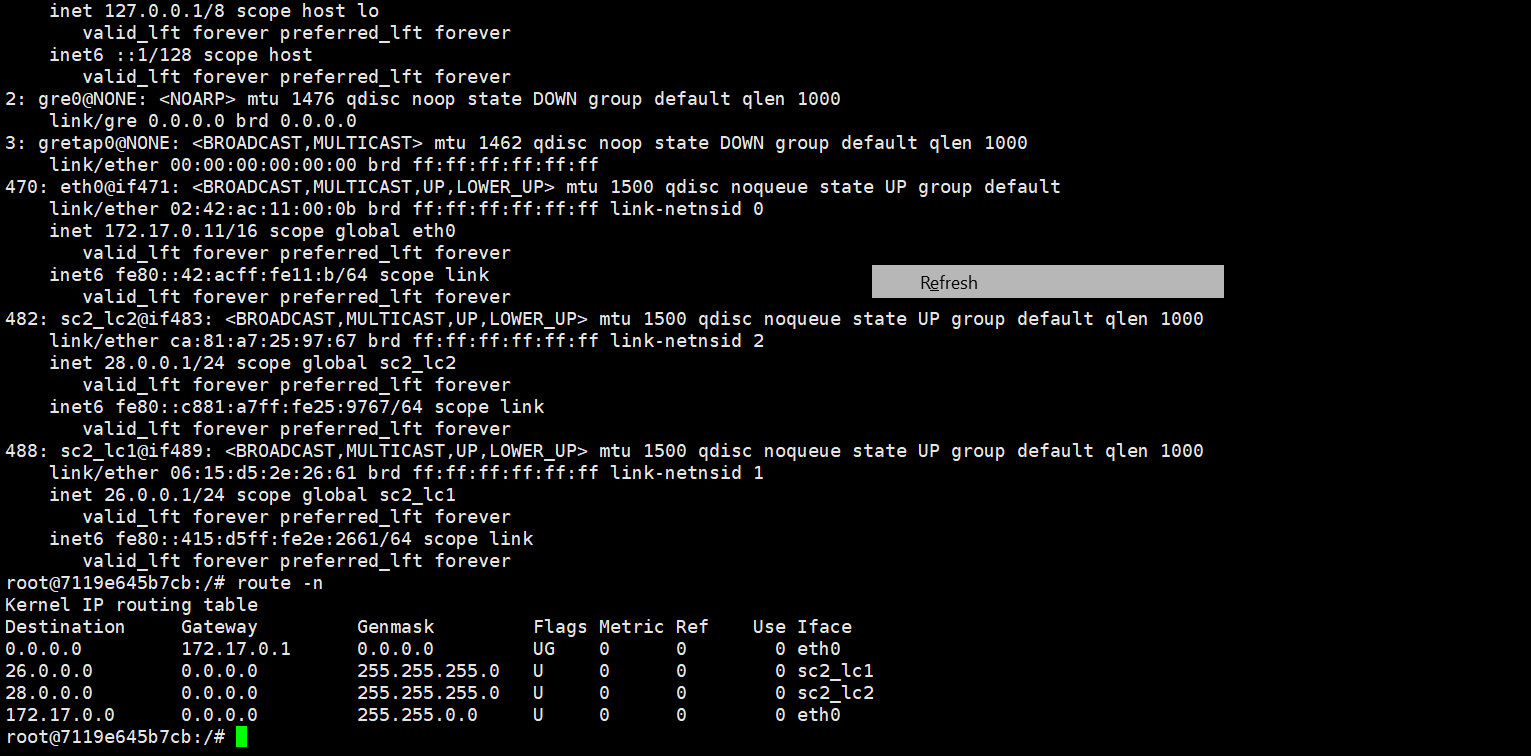
**Forwarding table of lc2:**

****

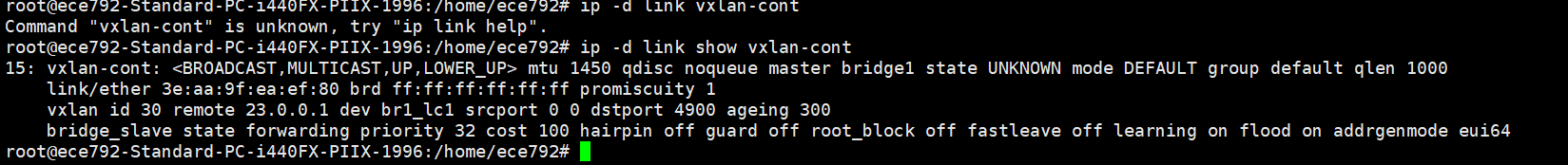
**sc1 config & forward table:**

****

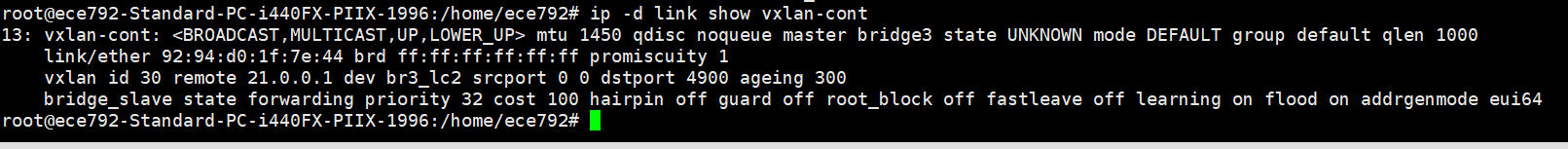
**Sc2 config & forwarding table:**

****

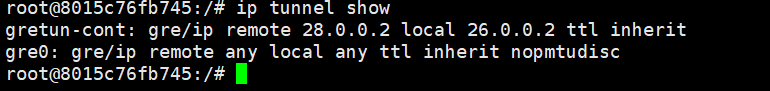
**Vxlan configuration in con\_ns1 namespace:**

****

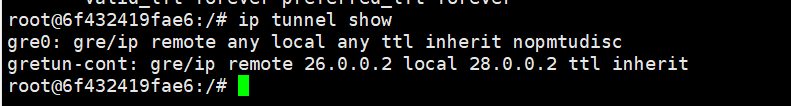
**Vxlan configuration in con\_ns3 namespace :**

****

**Gre tunnel configuration in lc1:**

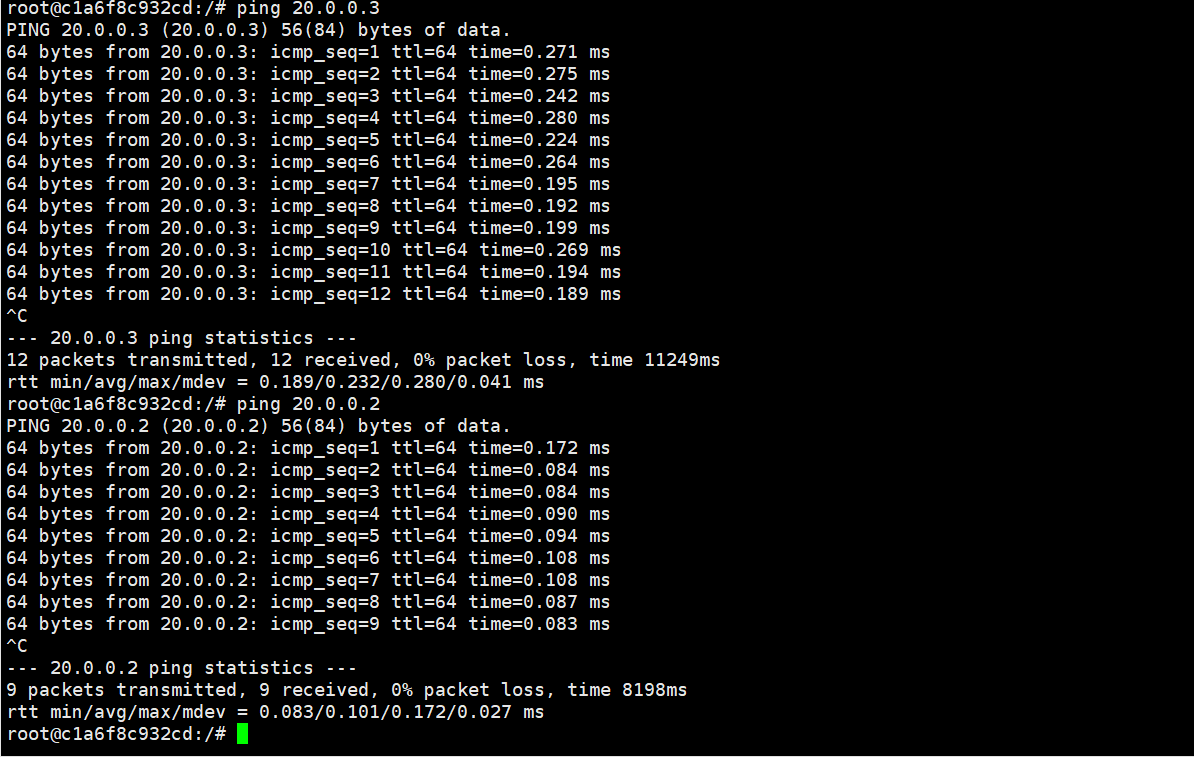
****

**Gre tunnel configuration in lc2:**

****

**Packet capture at cs1 (20.0.0.1) connected to lc1 via bridge1 to cs5 (20.0.0.3) which is connected to bridge3 to lc2:**

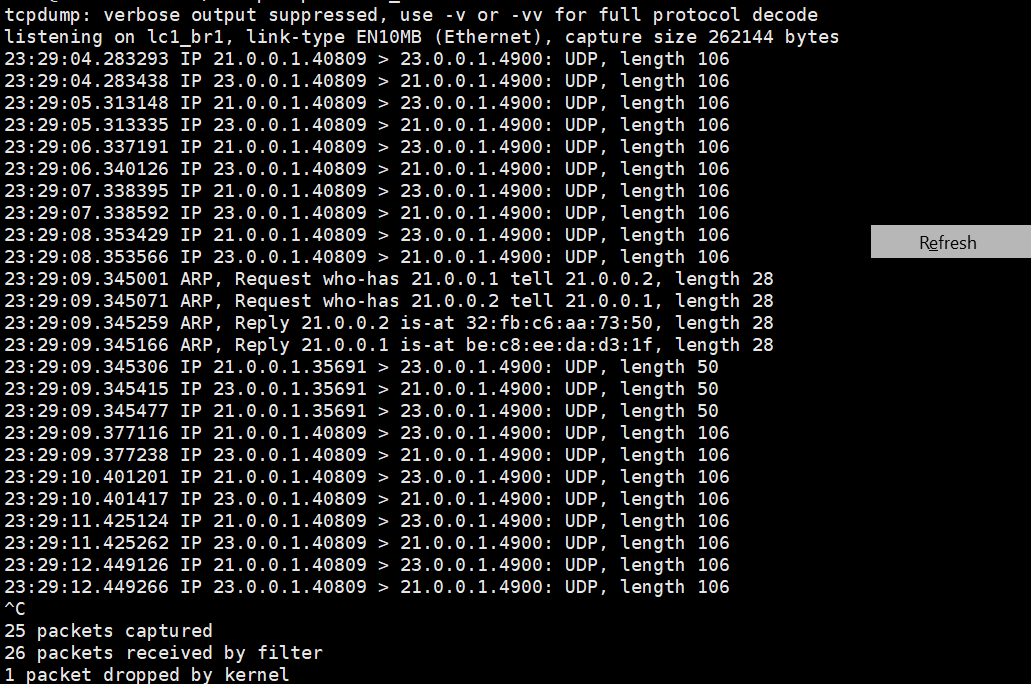
**Packet capture at cs1:**



Packet capture at **lc1-br1**(after vxlan encapsulation at lc1 entry interface):

VXLAN tunnel encapsulation at con\_ns1 namespace:

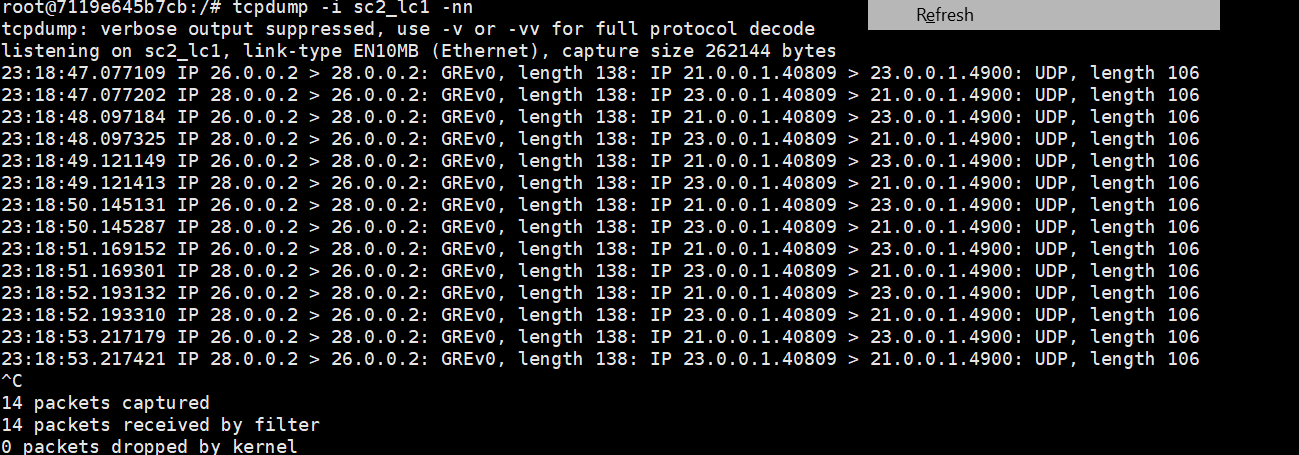
local: 21.0.0.1 remote :23.0.0.1



Packet capture at sc2\_lc1 interface (at sc2 ----lc1 ) link after the GRE encapsulation at GRE tunnel in lc1

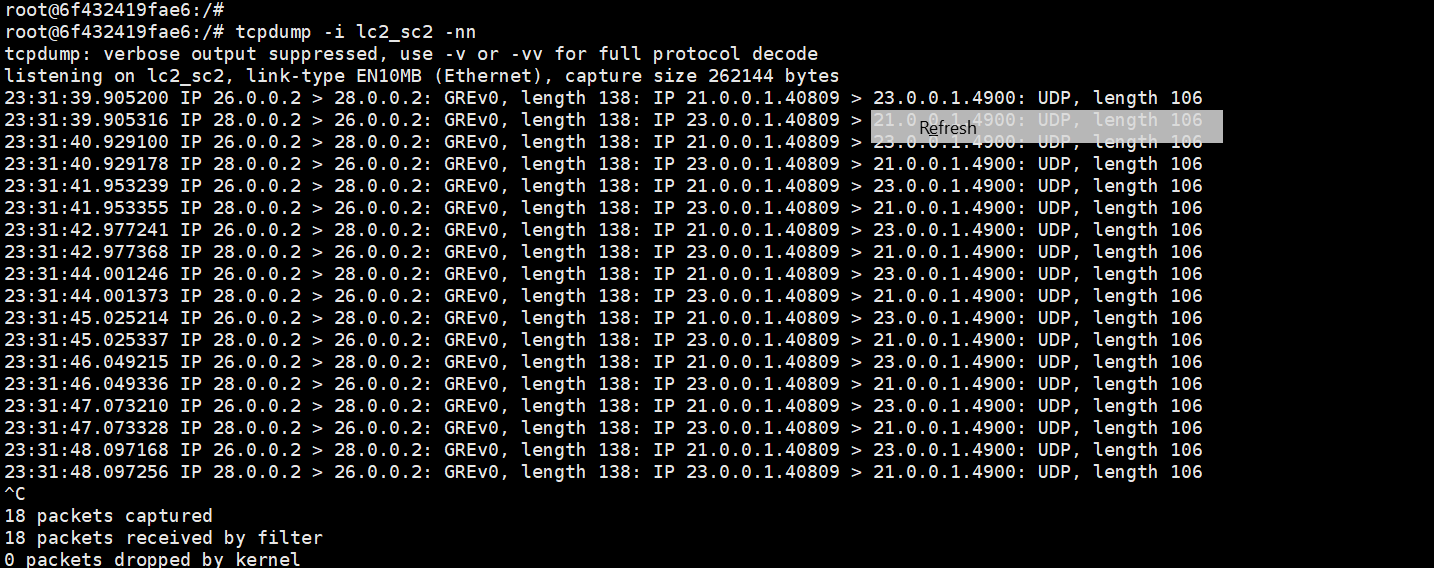
GRE tunnel at lc1:

Local ip :26.0.0.2 remote ip: 28.0.0.2



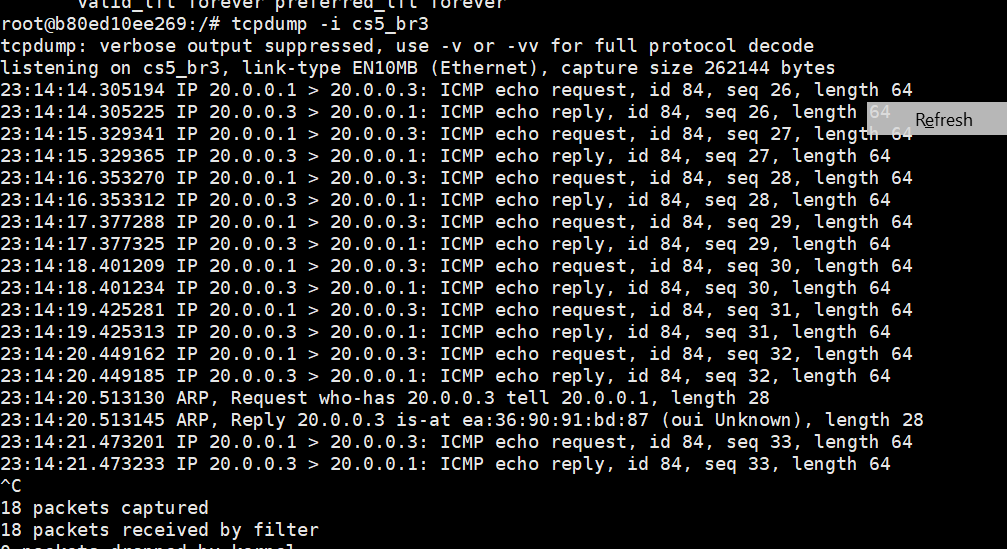
Packet capture at lc2\_sc1 (interface between lc2 and con\_ns3 namespace )

Have only the vxlan encapsulation.



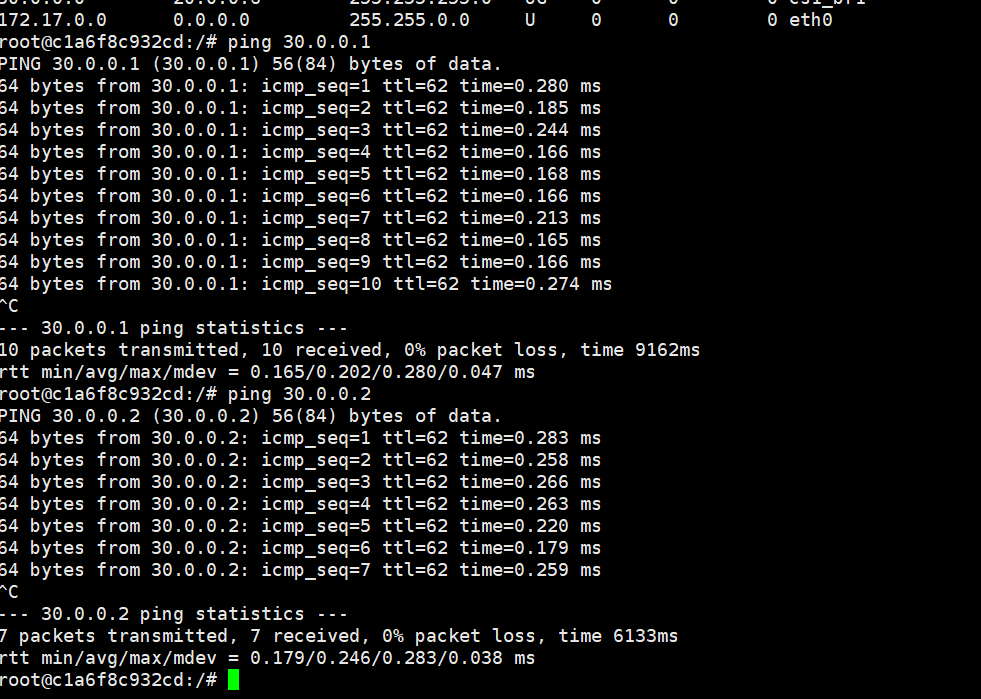
Packet capture at cs5:

Both VXLAN and GRE headers have been decapsulated at this point.

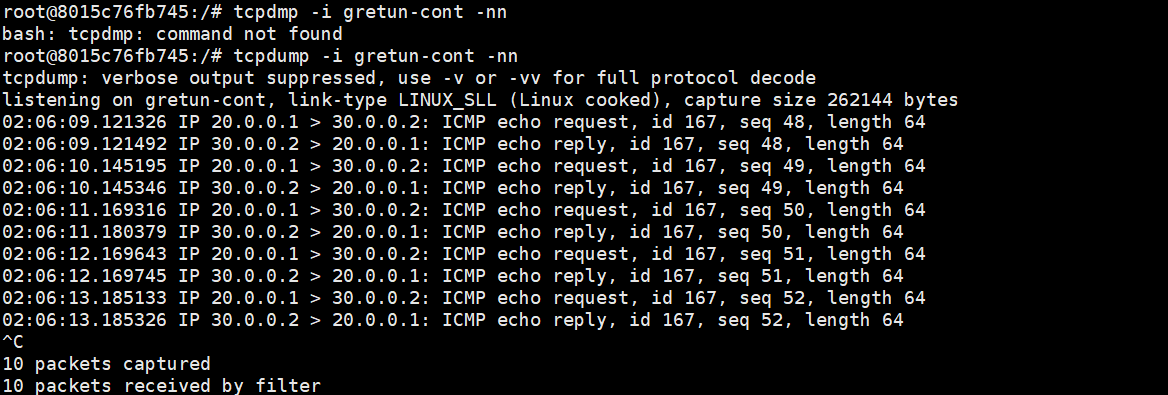


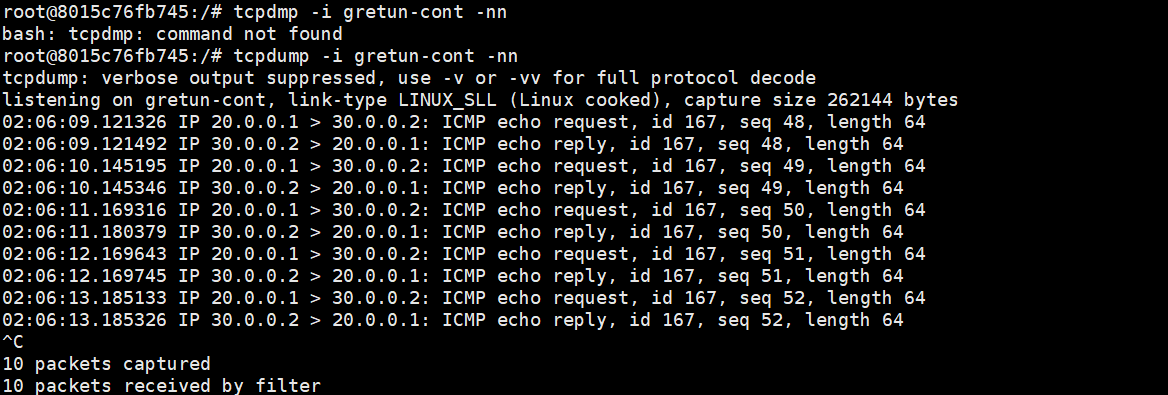
Pinging between the cs1 (20.0.0.1) cs3 (30.0.0.1) and cs4(30.0.0.2)

Capture at cs1 :

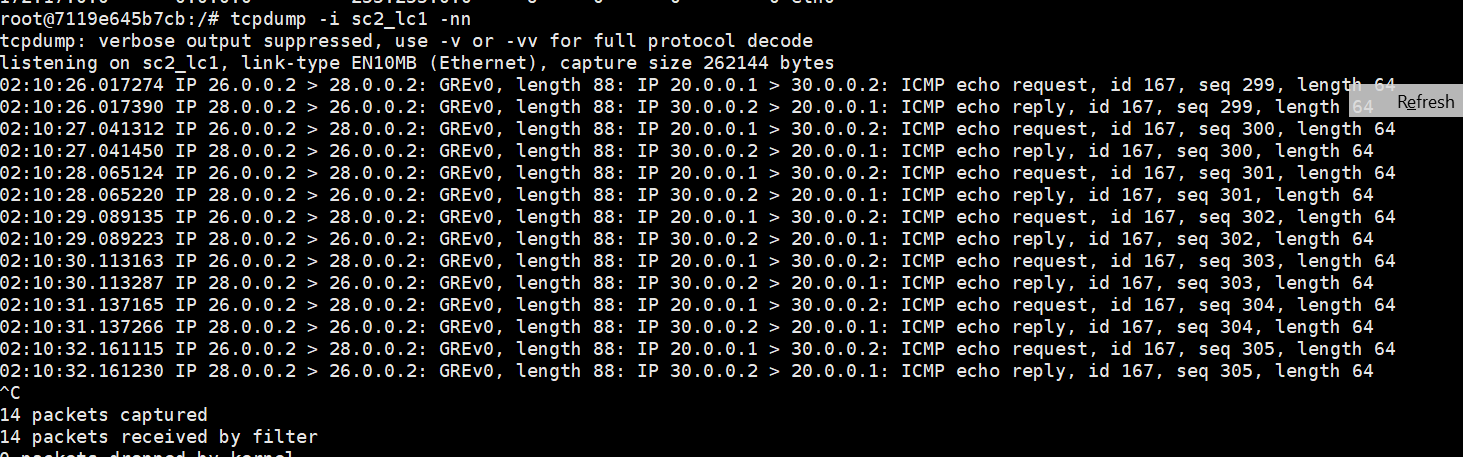


Capture at gretun-cont in lc1:

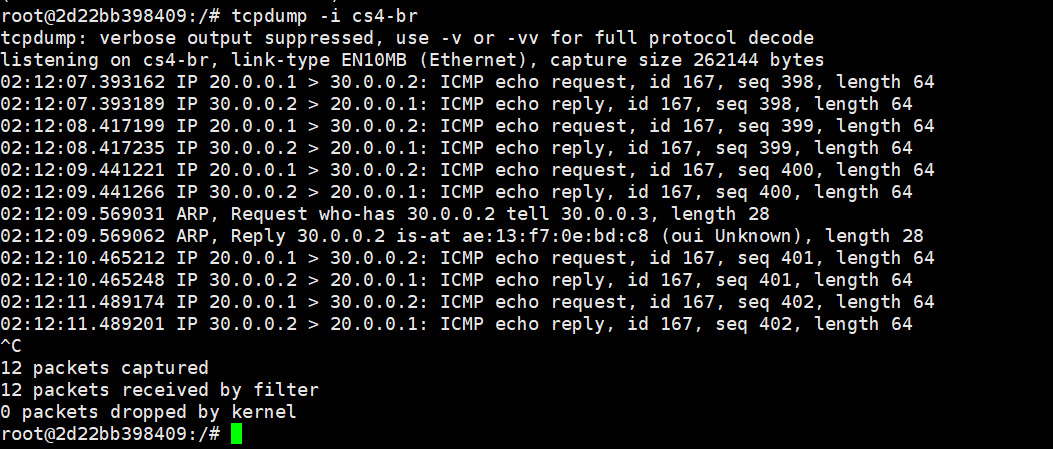


Capture at gretun in lc2:

Capture at sc2-lc1 node (connection between lc1-sc2):

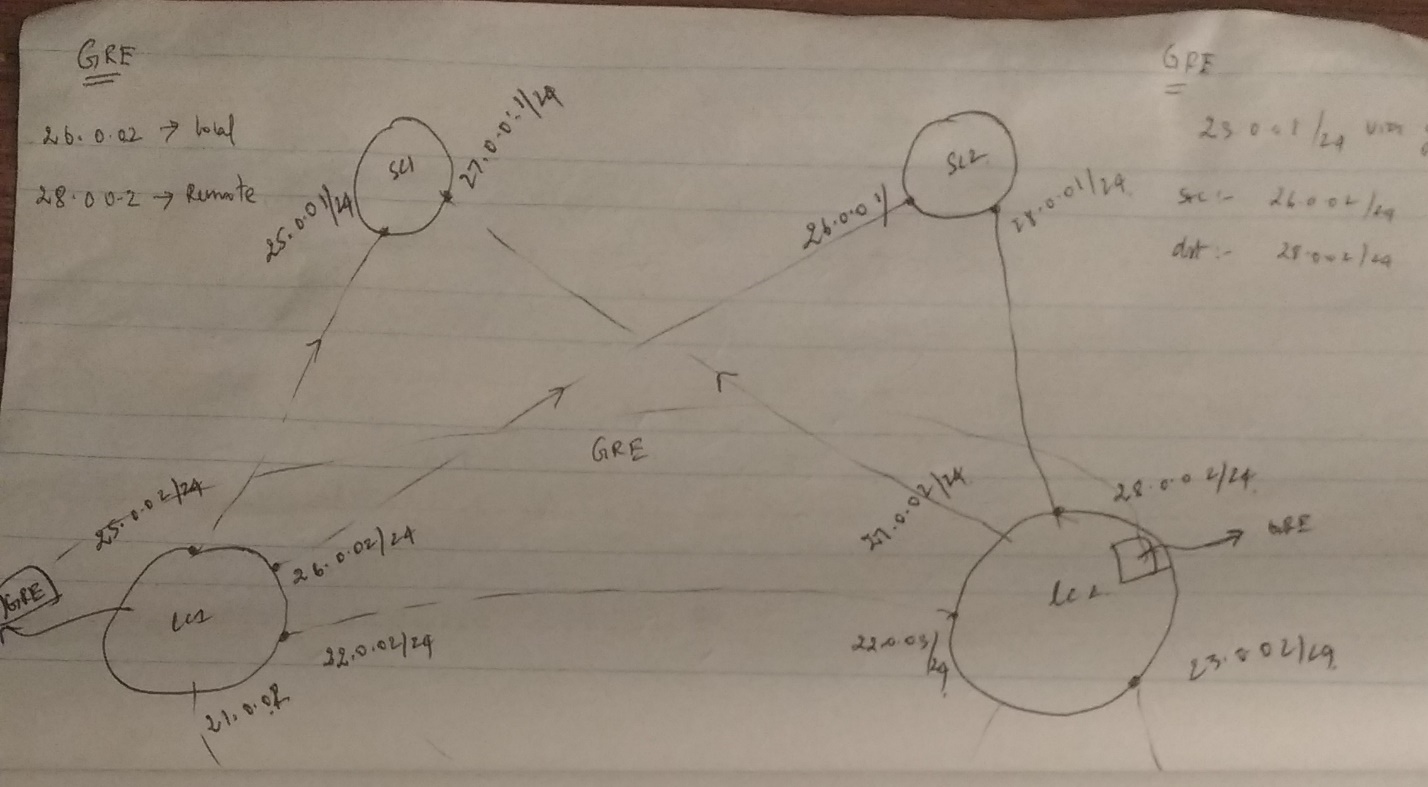


Capture at cs4 :



(Part2):

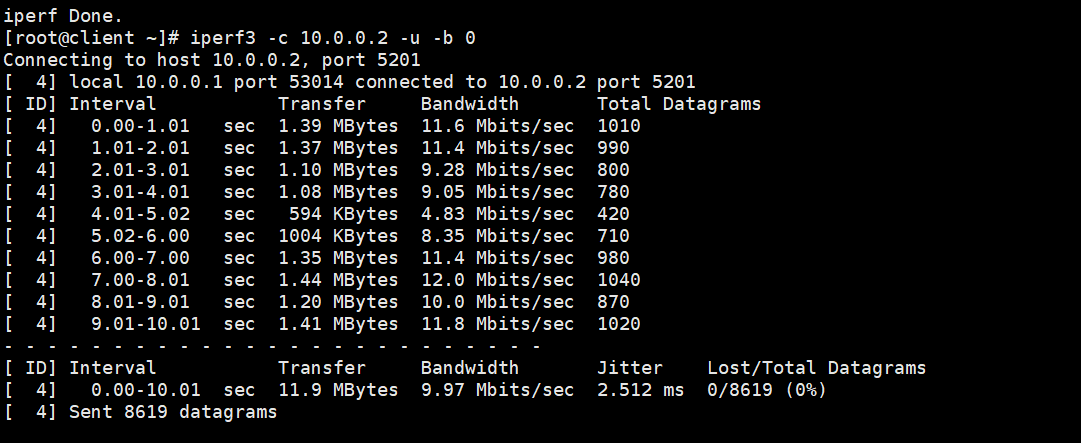
The lc1 and lc2 and sc1 and sc2 topology and configurations are below.



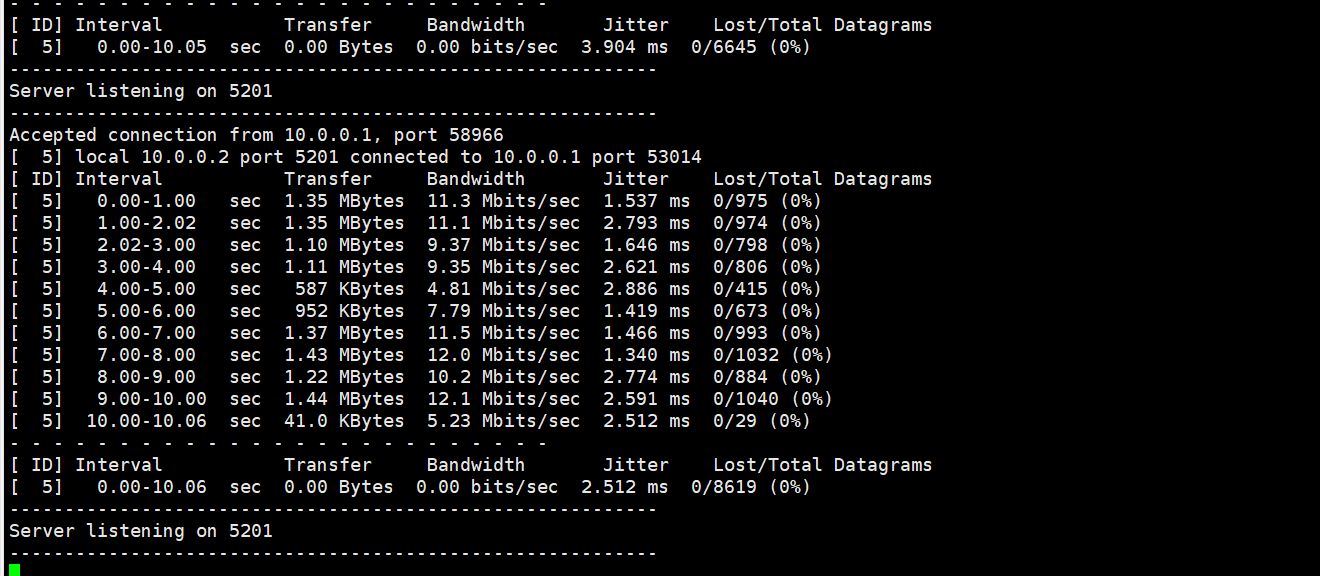
**Please refer to Readme file.**

**2.1 Iperf performance on the virtual machines:**

Iperf output on the client VM:

****

Iperf output on the server VM:



**Trace ouput for the the server vm ip\_rcv() function:**

Ran the following commands before taking the trace output :

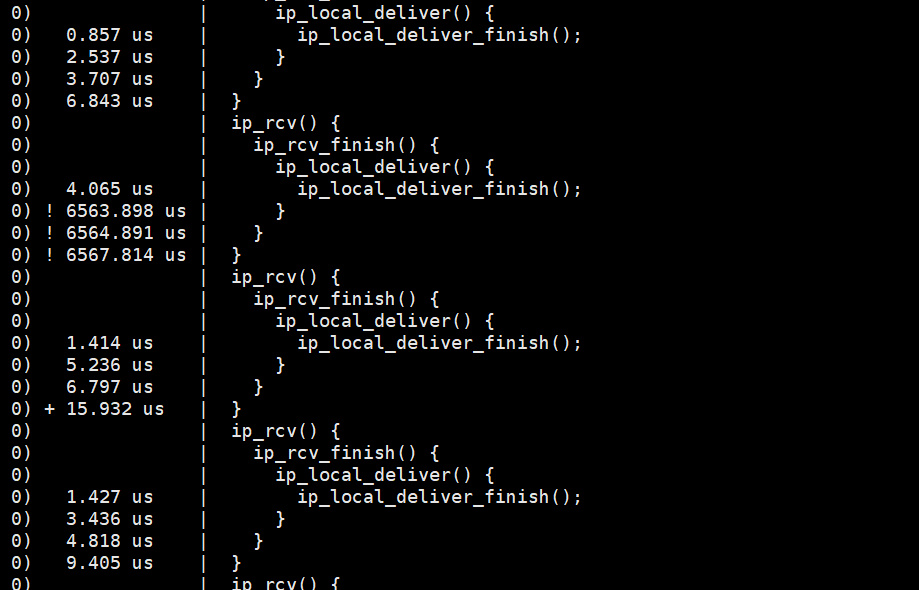
echo function\_graph > current\_tracer

echo ip-\* > set\_ftrace\_filter

echo <PID of IPERF3>  > set\_ftrace\_pid

echo > trace

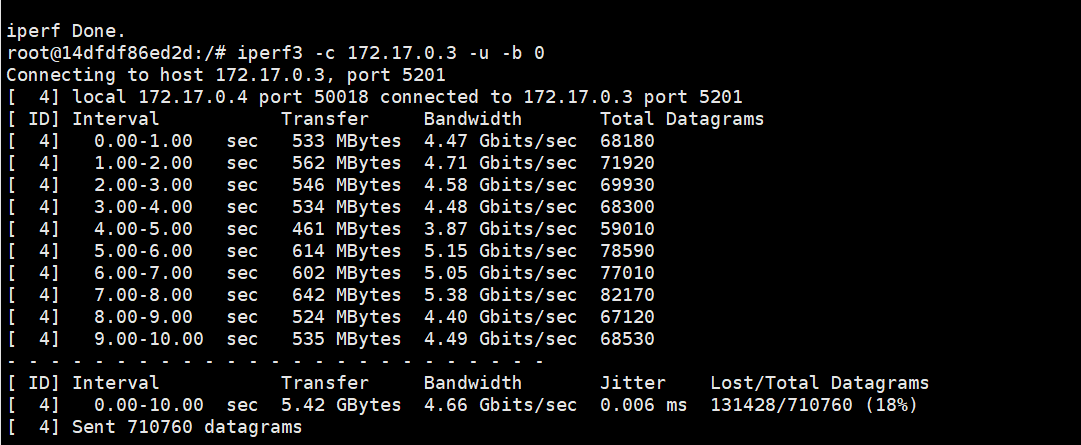
ps -ef | grep iperf3  ------ to find pid of iperf



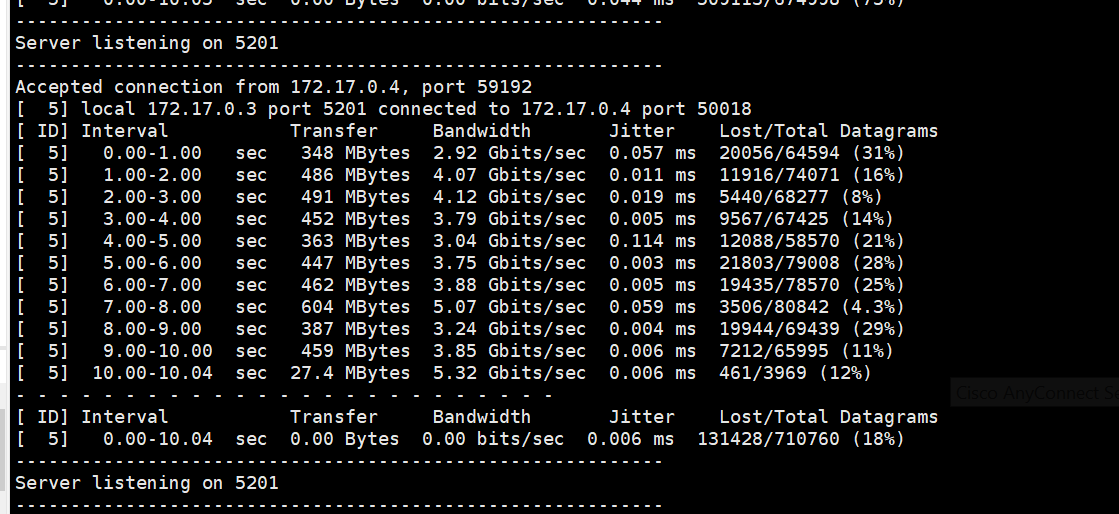
Here the ip\_rcv function is taking around 6000 us because of which the throughput is low when compared to the containers.

2.2 Iperf throughput on the container side :

**Iperf calculation on client container:**



**Iperf calculation on server container:**

****

**Trace output for the the server container ip\_rcv() function:**

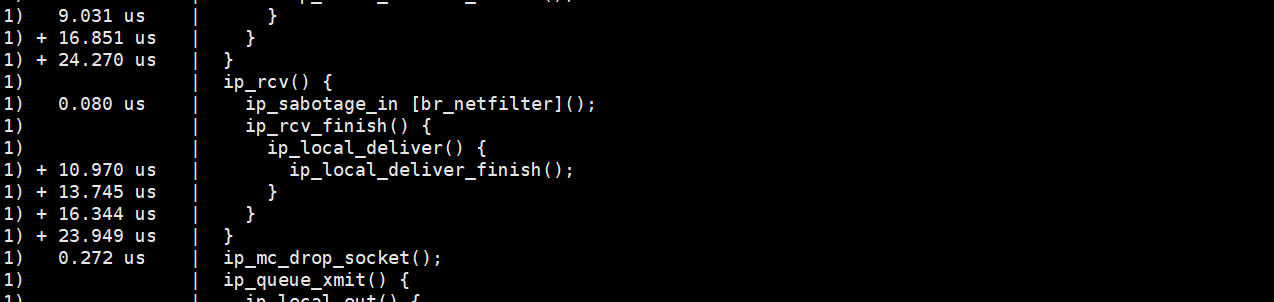
echo function\_graph > current\_tracer

echo ip-\* > set\_ftrace\_filter

echo <PID of IPERF3>  > set\_ftrace\_pid

echo > trace

ps -ef | grep iperf3  ------ to find pid of iperf

****

Here the ip\_rcv function is taking around 10 us because of which the throughput is higher which is around 4gbps.

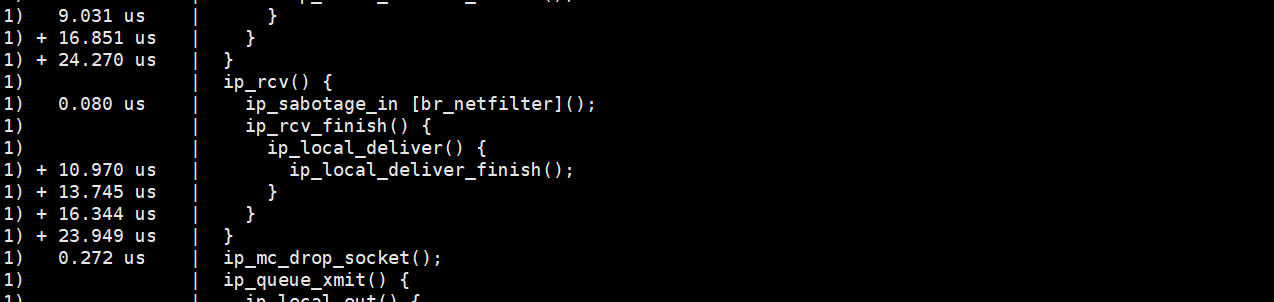
**2.3 comparision between the iperf performance in containers & Virtual machines :**

**From the trace output :**

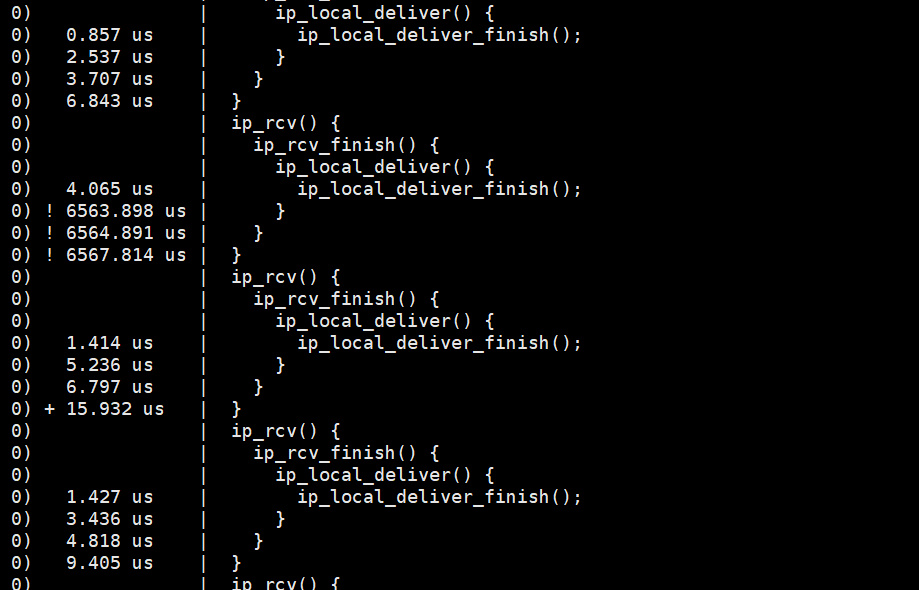
**Ip\_rcv() is taking around 6ms in the virtual machines where as ip\_rcv() function is taking in around 10ms**

Which is the reason the container iperf throughput is higher in containers when compared to iperf throughput in VM

**Trace output in containers:**

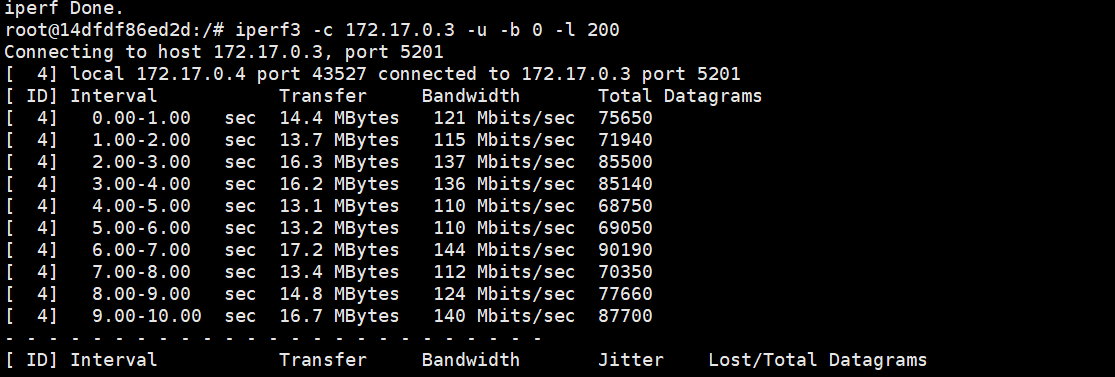


**Trace output in Virtual machines:**

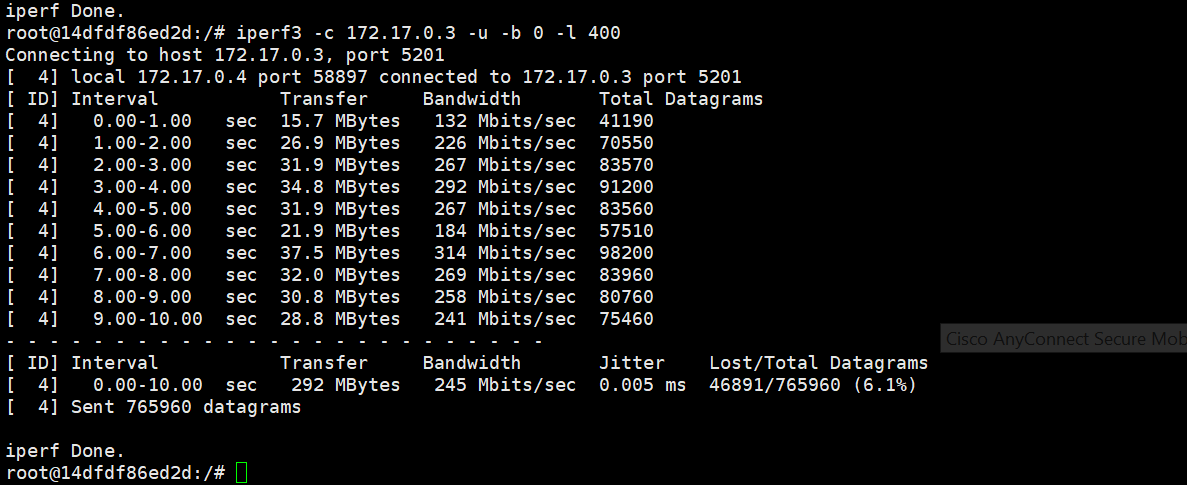
****

2.4 iperf performance with different packet sizes in container:

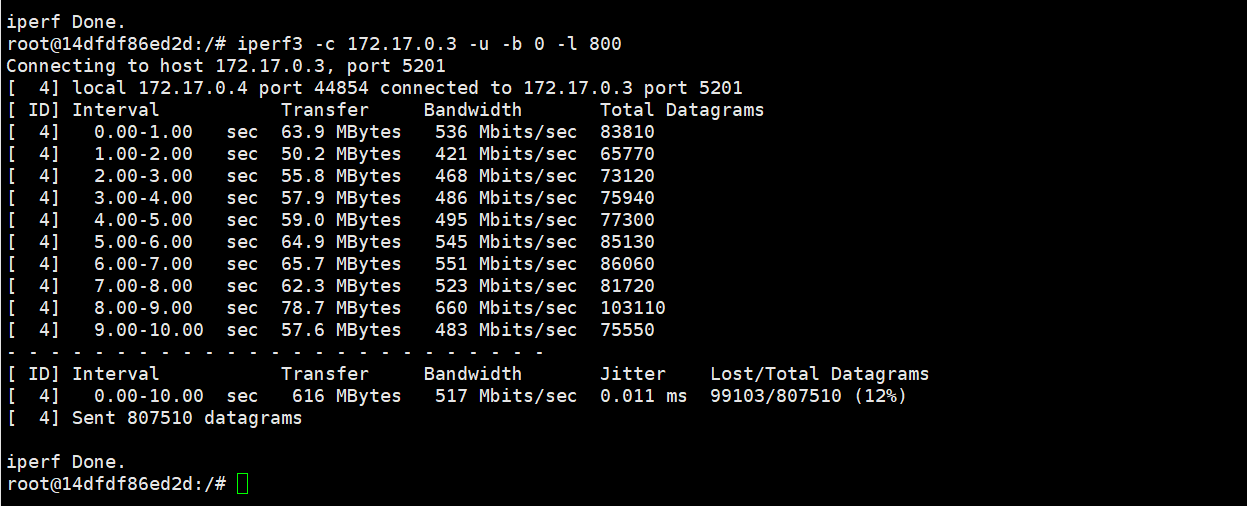
**Iperf performance with packet size of 200:**



**Iperf performance with packet size of 400:**



**Iperf performance with packet size of 800:**



With doubling the packet size the throughput is getting doubled

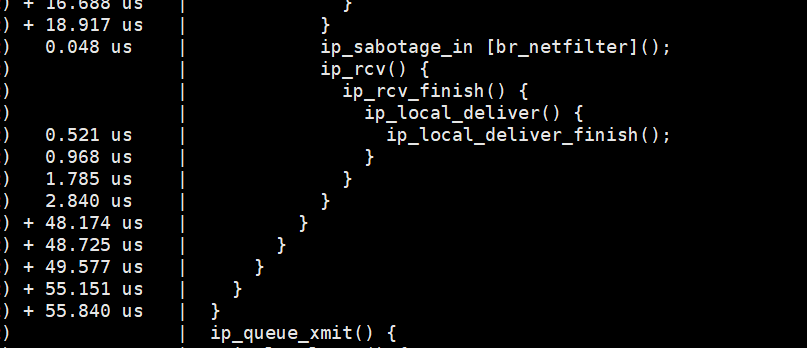
With increase in packet size the throughput is getting increased because

The number of cpu cycles taken to send packets is same irrespective of packet size

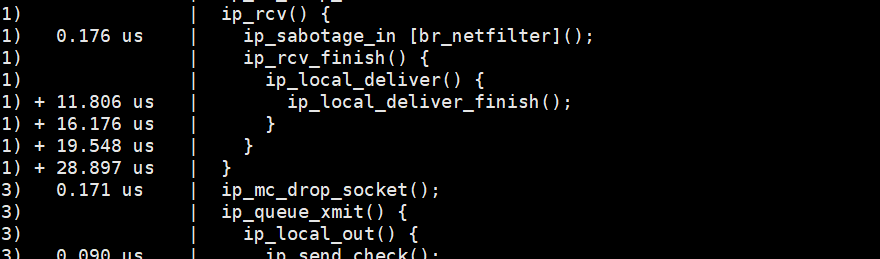
where as in smaller packet the number of cpu cycles taken to send the same data is higher when compared to higher sized packets.

Which is the reason the performance is better in higher sized packets.

**Ip\_rcv() function is taking time around 56us in case of packet size = 400:**

I

**Ip\_rcv() function takes time of around 28 us in case of packet size =800**



The time taken by the ip\_rcv() function becomes half in packet size of 800 when compared to the ip\_rcv() function time if packet size is 400