HOMEWORK 3

BY: Srinivas Parasa (sparasa)
Muppidi Harshavardhan Reddy (mreddy2)

[3]L2 Mode:

[3.1] Design 1:

[3.1.A]

What are the disadvantages for tenants? Is a tenant's traffic isolated from other tenants?

Scenario 1: Assumptions: The 2 tenants have different subnets,
as in VM1 in both hypervisors belong to 192.168.10.0/24 network
And VM2 in both hypervisors belong to 192.168.20.0/24 network
So we have both Tenants in different subnets.
So we have a L2 network here with both tenants in different subnets. Therefore traffic from one tenant would not be able to reach the other tenant and we would have isolation.

Other Scenario 2 : In another scenario if we have same subnets for both the tenants say 192.168.10.0/24 we will not have isolation since a VM from tenant A would be able to reach a VM in tenant B since it's a L2 network and the replies for ARP broadcast would be received and they can ping.

Disadvantages for tenants:

So incase of a scenario 2 the tenants would not have isolation which is definitely not a desired property for any tenant. They would anyday want isolation of their own networks. So basically since both tenants are connected to same bridges in the hypervisors, we cannot have same subnets for both the tenants and in case where both tenants would want to use a same subnet that wouldn't be possible with this design.

Successful Ping during Scenario 1 from VM1(TenantA-Hypervisor1) to VM1(TenantA-Hypervisor2)

```
[root@localhost ~]# ip addr show dev eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:aa:e4:6d:9a brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.2/24 scope global eth1
       valid_lft forever preferred_lft forever
[root@localhost ~]# ping 192.168.10.3
PING 192.168.10.3 (192.168.10.3) 56(84) bytes of data.
64 bytes from 192.168.10.3: icmp_seq=1 ttl=64 time=3.19 ms
64 bytes from 192.168.10.3: icmp_seq=2 ttl=64 time=3.15 ms
64 bytes from 192.168.10.3: icmp_seq=3 ttl=64 time=1.95 ms
64 bytes from 192.168.10.3: icmp_seq=4 ttl=64 time=1.58 ms
64 bytes from 192.168.10.3: icmp_seq=5 ttl=64 time=1.43 ms
^C
--- 192.168.10.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 1.434/2.264/3.192/0.762 ms
[root@localhost ~]#
```

Packets capture received at HypervisorB VM1 interface:

```
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
 Interface  Type       Source     Model       MAC
 -------------------------------------------------------
 vnet5      network    default1   rtl8139     52:54:00:6f:6e:ca
 vnet6      bridge     sw2_network virtio      52:54:ee:6f:6e:ca

[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet6 icmp
 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
 listening on vnet6, link-type EN10MB (Ethernet), capture size 262144 bytes
 02:07:08.588727 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9288, seq 1, length 64
 02:07:08.589853 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9288, seq 1, length 64
 02:07:09.590129 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9288, seq 2, length 64
 02:07:09.592115 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9288, seq 2, length 64
 02:07:10.593867 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9288, seq 3, length 64
 02:07:10.594177 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9288, seq 3, length 64
 02:07:11.594996 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9288, seq 4, length 64
 02:07:11.595549 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9288, seq 4, length 64
 02:07:12.597042 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9288, seq 5, length 64
 02:07:12.597543 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9288, seq 5, length 64
 ^C
 10 packets captured
 10 packets received by filter
 0 packets dropped by kernel
 ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
```

[3.1.B]

What, if anything, breaks if two tenants in the same hypervisor host use the same IP address?

So the tenants having same IP addresses in a hypervisor would mean that they are in the same subnet as well. So a ping to same IP(VM1 in Tenant A to VM2 in Tenant B, both in same hypervisor) though it is successful it is in fact pinging itself. While now 2 scenarios arise

Scenario 1: The other Hypervisor B has Tenants in the Same subnet as the Hypervisor A

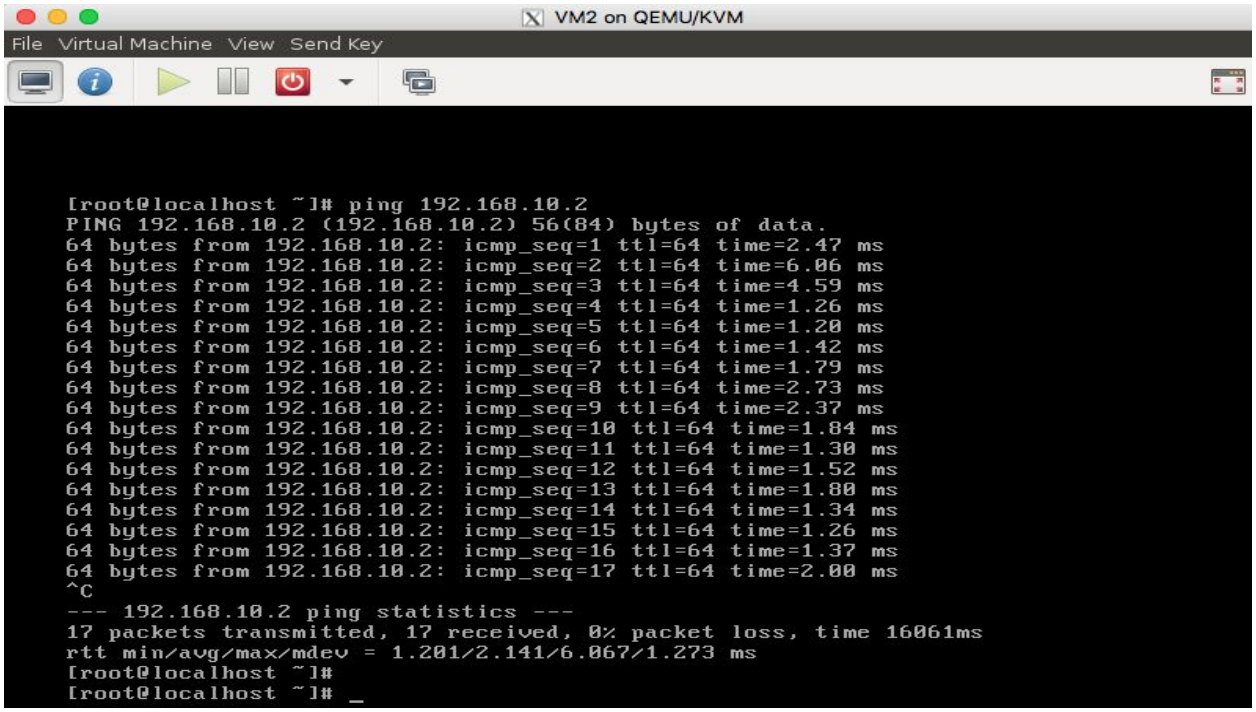
Since being L2 bridges when the arp request reaches the the bridge at the other Hypervisor B now we cannot assure which VM will it actually get connected to. So in our test we have tried pinging VM2(Tenant B in Hypervisor A) from VM2(Tenant B in Hypervisor B) and we have observed that the ping is successful but to VM1(Tenant A in Hypervisor A) which is a VM of a different tenant.

This is not desirable and potentially breaks the isolation and lets Tenant B connect to Tenant A. So we cannot assure which VM will it connect to in the other hypervisor because it depends on whose arp reply is received first.

Scenario 2: The other Hypervisor B has Tenants in the different subnet from the Hypervisor A

Actually the same issue as above arises again at both the hypervisors we do not know which VM will it get connected to whether to the same tenant VM or a different VM.

Ping VM2(Tenant B in Hypervisor A) from VM2(Tenant B in Hypervisor B):



Packets capture at VM1(Tenant A Hypervisor A)

```
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
 Interface  Type       Source     Model       MAC
 -------------------------------------------------------
 vnet6      network    default    virtio      52:54:00:e4:6d:9a
 vnet7      bridge     sw2_network virtio      52:54:aa:e4:6d:9a

[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet7 icmp
[[sudo] password for ece792:
 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
 listening on vnet7, link-type EN10MB (Ethernet), capture size 262144 bytes
 02:20:52.786173 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 1, length 64
 02:20:52.787032 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 1, length 64
 02:20:53.788110 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 2, length 64
 02:20:53.788532 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 2, length 64
 02:20:54.793388 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 3, length 64
 02:20:54.793787 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 3, length 64
 02:20:55.792927 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 4, length 64
 02:20:55.793336 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 4, length 64
 02:20:56.794473 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 5, length 64
 02:20:56.794727 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 5, length 64
 02:20:57.796076 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 6, length 64
 02:20:57.796499 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 6, length 64
 02:20:58.798048 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 47109, seq 7, length 64
 02:20:58.798455 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 47109, seq 7, length 64
```

[3.1.C]

What, if anything, breaks if two tenants in a different hypervisor host use the same IP address?

Assumption:

AT Hypervisor A: VM1 → 192.168.10.2 Tenant A

AT Hypervisor A: VM2 → 192.168.20.2 Tenant B

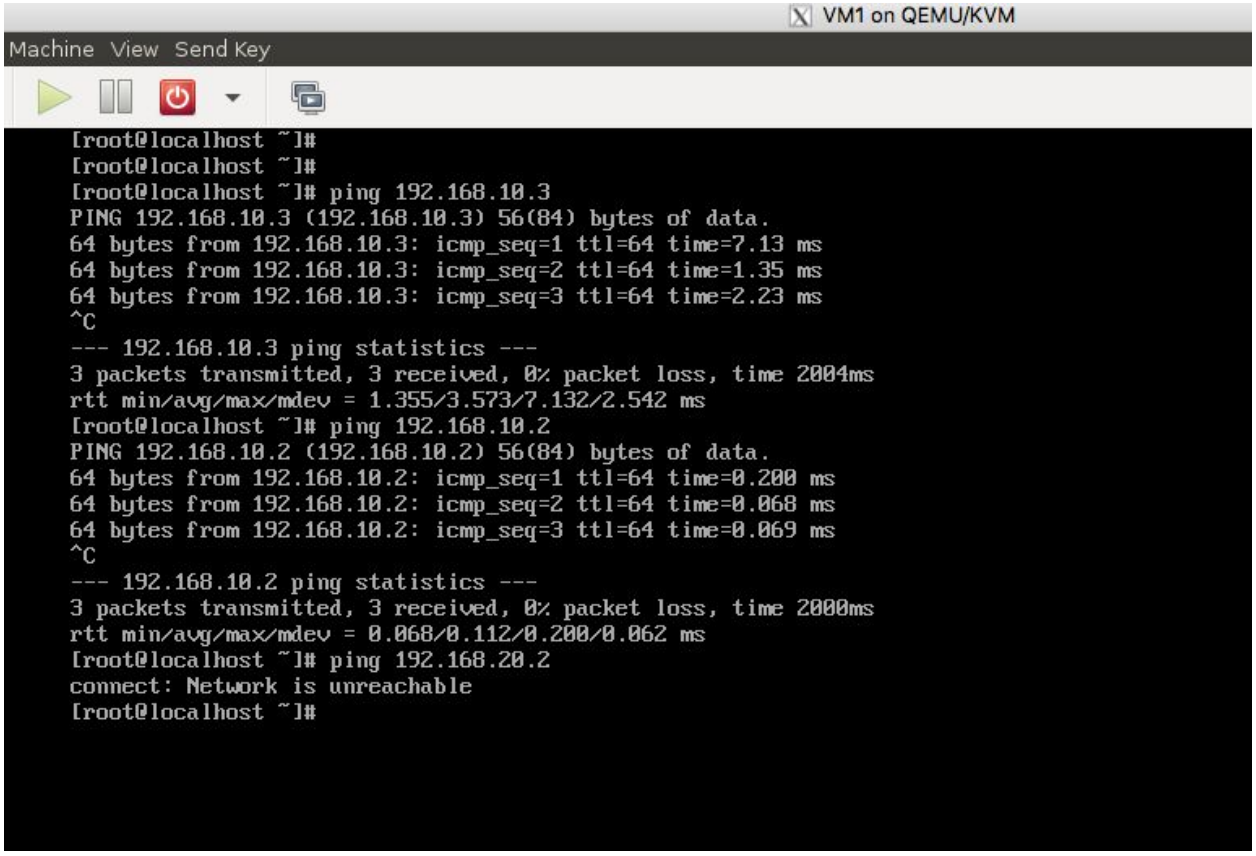
AT Hypervisor B: VM1 → 192.168.10.3 Tenant A

AT Hypervisor B: VM2 → 192.168.10.2 Tenant B (Same IP)

So here we have the Tenant A in subnet: 192.168.10.0/24 subnet and
Tenant B has subnet 192.168.20.0/24 and as we have same IP it also has 192.168.10.0/24.

In this scenario the VM1 at Tenant A in Hypervisor A would be able to reach the VM1 in Hypervisor B and a ping with IP of VM2(TB,HB) would be nothing but actually pinging itself(since same IP) which would be successful but it is actually not reaching the VM2(TB,HB). While a ping between VM's of Tenant A would not have any issue.
And also in our scenario a ping from VM2(HB) to VM1(HB) would also be successful since we have the same subnet and it's a L2 datapath but this is not desired as the tenants should have isolation but having a different subnet would resolve that issue though.
So with this configuration, it depends on what subnets the other VM's are configured which would decide whether we would really have a problem or not. But explanation above is given with respect to our assumption mentioned above.

Pinging Everyone else from VM1(HA):



Packets Capture at VM1(HB) Interface:

```
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
 Interface  Type       Source     Model       MAC
 -------------------------------------------------------
 vnet5      network    default1   rtl8139     52:54:00:6f:6e:ca
 vnet6      bridge     sw2_network virtio      52:54:ee:6f:6e:ca

[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet6 icmp
[[sudo] password for ece792:
 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
 listening on vnet6, link-type EN10MB (Ethernet), capture size 262144 bytes
 05:05:47.666804 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9410, seq 1, length 64
 05:05:47.667536 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9410, seq 1, length 64
 05:05:48.667931 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9410, seq 2, length 64
 05:05:48.668221 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9410, seq 2, length 64
 05:05:49.670494 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9410, seq 3, length 64
 05:05:49.671066 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9410, seq 3, length 64
 ^C
 6 packets captured
 6 packets received by filter
 0 packets dropped by kernel
 ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
```

[3.1.D]

What, if anything, breaks if two tenants in the same hypervisor host use the same MAC address?

Assumed Scenario:

AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B (Same MAC)
AT Hypervisor B: VM1 → 192.168.10.3 Tenant A
AT Hypervisor B: VM2 → 192.168.20.3 Tenant B

In this scenario, when we first ping from VM1(A) to VM1(B) it would be successful without packet loss. Then since they are in different subnet ping to VM2(A) from VM1(A) wouldn't work. Now after this we ping VM2(A) from VM2(B) and it was eventually successful but the first few icmp request packets reached VM1(A) since the arp table has the same mac associated with the VM1(A). Missing of few packets would be the only thing which might be affected else. Pings between the VM's for the respective tenants wouldn't be a problem.

2nd Ping mentioned above:(ICMP Seq to be referred):

```
[root@localhost ~]#
[root@localhost ~]# ping 192.168.20.2
PING 192.168.20.2 (192.168.20.2) 56(84) bytes of data.
64 bytes from 192.168.20.2: icmp_seq=10 ttl=64 time=2.17 ms
64 bytes from 192.168.20.2: icmp_seq=11 ttl=64 time=3.59 ms
64 bytes from 192.168.20.2: icmp_seq=12 ttl=64 time=2.05 ms
64 bytes from 192.168.20.2: icmp_seq=13 ttl=64 time=1.46 ms
64 bytes from 192.168.20.2: icmp_seq=14 ttl=64 time=1.57 ms
64 bytes from 192.168.20.2: icmp_seq=15 ttl=64 time=1.14 ms
64 bytes from 192.168.20.2: icmp_seq=16 ttl=64 time=1.16 ms
64 bytes from 192.168.20.2: icmp_seq=17 ttl=64 time=1.34 ms
64 bytes from 192.168.20.2: icmp_seq=18 ttl=64 time=2.04 ms
64 bytes from 192.168.20.2: icmp_seq=19 ttl=64 time=1.25 ms
^C
--- 192.168.20.2 ping statistics ---
19 packets transmitted, 10 received, 47% packet loss, time 18357ms
rtt min/avg/max/mdev = 1.144/1.780/3.590/0.706 ms
[root@localhost ~]# _
```

Packets Capture at VM1(A) and VM2(A) interfaces:

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
 Interface  Type       Source     Model       MAC
 -------------------------------------------------------
 vnet6      network    default    virtio      52:54:00:e4:6d:9a
 vnet7      bridge     sw2_network virtio      52:54:aa:e4:6d:9a

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ tcpcump -i vnet7 icmp
No command 'tcpcump' found, did you mean:
 Command 'tcpdump' from package 'tcpdump' (main)
tcpcump: command not found
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet7 icmp
[sudo] password for ece792:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vnet7, link-type EN10MB (Ethernet), capture size 262144 bytes
05:25:23.714613 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 1, length 64
05:25:23.715606 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 1, length 64
05:25:24.715959 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 2, length 64
05:25:24.716392 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 2, length 64
05:25:25.717901 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 3, length 64
05:25:25.718180 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 3, length 64
05:25:26.719223 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 4, length 64
05:25:26.719532 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 4, length 64
05:25:27.721204 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 5, length 64
05:25:27.721562 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 5, length 64
05:25:28.722802 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 6, length 64
05:25:28.723255 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 6, length 64
05:25:29.724070 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 7, length 64
05:25:29.724522 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 7, length 64
05:25:30.726709 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 8, length 64
05:25:30.727335 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 8, length 64
05:25:31.729697 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 9, length 64
05:25:31.730404 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 9, length 64
05:25:32.730669 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 10, length 64
05:25:32.731957 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 10, length 64
05:25:33.732520 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 11, length 64
05:25:33.733574 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 11, length 64
05:25:51.436646 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 1, length 64
05:25:52.435803 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 2, length 64
05:25:53.435580 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 3, length 64
05:25:54.436432 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 4, length 64
05:25:55.436339 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 5, length 64
05:25:56.436635 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 6, length 64
05:25:57.437193 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 7, length 64
05:25:58.436751 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 8, length 64
05:25:59.440990 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 9, length 64
```

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
 Interface  Type       Source     Model       MAC
 -------------------------------------------------------
 vnet6      network    default    virtio      52:54:00:e4:6d:9a
 vnet7      bridge     sw2_network virtio      52:54:aa:e4:6d:9a

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ tcpcump -i vnet7 icmp
No command 'tcpcump' found, did you mean:
 Command 'tcpdump' from package 'tcpdump' (main)
tcpcump: command not found
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet7 icmp
[sudo] password for ece792:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vnet7, link-type EN10MB (Ethernet), capture size 262144 bytes
05:25:23.714613 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 1, length 64
05:25:23.715606 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 1, length 64
05:25:24.715959 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 2, length 64
05:25:24.716392 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 2, length 64
05:25:25.717901 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 3, length 64
05:25:25.718180 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 3, length 64
05:25:26.719223 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 4, length 64
05:25:26.719532 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 4, length 64
05:25:27.721204 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 5, length 64
05:25:27.721562 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 5, length 64
05:25:28.722802 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 6, length 64
05:25:28.723255 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 6, length 64
05:25:29.724070 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 7, length 64
05:25:29.724522 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 7, length 64
05:25:30.726709 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 8, length 64
05:25:30.727335 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 8, length 64
05:25:31.729697 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 9, length 64
05:25:31.730404 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 9, length 64
05:25:32.730669 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 10, length 64
05:25:32.731957 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 10, length 64
05:25:33.732520 IP 192.168.10.3 > 192.168.10.2: ICMP echo request, id 49413, seq 11, length 64
05:25:33.733574 IP 192.168.10.2 > 192.168.10.3: ICMP echo reply, id 49413, seq 11, length 64
05:25:51.436646 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 1, length 64
05:25:52.435803 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 2, length 64
05:25:53.435580 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 3, length 64
05:25:54.436432 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 4, length 64
05:25:55.436339 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 5, length 64
05:25:56.436635 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 6, length 64
05:25:57.437193 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 7, length 64
05:25:58.436751 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 8, length 64
05:25:59.440990 IP 192.168.20.3 > 192.168.20.2: ICMP echo request, id 518, seq 9, length 64
```

[3.1.E]

What, if anything, breaks if two tenants in a different hypervisor host use the same MAC address?

Assumed Scenario:

AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same MAC)

AT Hypervisor A: VM2 → 192.168.20.2 Tenant B

AT Hypervisor B: VM1 → 192.168.10.3 Tenant A

AT Hypervisor B: VM2 → 192.168.20.3 Tenant B (Same MAC)

In scenario similar to above due to arp resolution some packets goes missing initially when we create a experiment where the series of Pings happen as below:

VM1(A) -> VM1(B)

VM2(B) -> VM2(A)

During second ping there would be a few packets missing because again the arp has to be resolved since a when a icmp connection is not established reinitiating a Arp request and resolving consumes a little time during which the packets go missing. Else the pings would be successful between the VM's within a tenant with respect to our assumed configuration mentioned above.

Packet Captures at VM1(A) and VM2(A) interfaces during the pings:

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet5      network    default1   rtl8139     52:54:00:6f:6e:ca
vnet6      bridge     sw2_network virtio      52:54:ee:6f:6e:ca

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet6 icmp
[sudo] password for ece792:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vnet6, link-type EN10MB (Ethernet), capture size 262144 bytes
06:21:57.938418 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9459, seq 1, length 64
06:21:57.938835 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9459, seq 1, length 64
06:21:58.937869 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9459, seq 2, length 64
06:21:58.938189 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9459, seq 2, length 64
06:21:59.940031 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9459, seq 3, length 64
06:21:59.940564 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9459, seq 3, length 64
06:22:00.941471 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9459, seq 4, length 64
06:22:00.942186 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9459, seq 4, length 64
06:22:01.943293 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9459, seq 5, length 64
06:22:01.943729 IP 192.168.10.3 > 192.168.10.2: ICMP echo reply, id 9459, seq 5, length 64
```

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM2
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet7      network    default1   rtl8139     52:54:00:d5:42:2d
vnet8      bridge     sw2_network virtio      52:54:de:d5:42:2d

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo tcpdump -i vnet8 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vnet8, link-type EN10MB (Ethernet), capture size 262144 bytes
06:21:57.938439 IP 192.168.10.2 > 192.168.10.3: ICMP echo request, id 9459, seq 1, length 64
06:22:24.486087 IP 192.168.20.2 > 192.168.20.3: ICMP echo request, id 9443, seq 10, length 64
06:22:24.486934 IP 192.168.20.3 > 192.168.20.2: ICMP echo reply, id 9443, seq 10, length 64
06:22:25.488009 IP 192.168.20.2 > 192.168.20.3: ICMP echo request, id 9443, seq 11, length 64
06:22:25.489096 IP 192.168.20.3 > 192.168.20.2: ICMP echo reply, id 9443, seq 11, length 64
06:22:26.489836 IP 192.168.20.2 > 192.168.20.3: ICMP echo request, id 9443, seq 12, length 64
06:22:26.490333 IP 192.168.20.3 > 192.168.20.2: ICMP echo reply, id 9443, seq 12, length 64
06:22:27.491753 IP 192.168.20.2 > 192.168.20.3: ICMP echo request, id 9443, seq 13, length 64
06:22:27.492528 IP 192.168.20.3 > 192.168.20.2: ICMP echo reply, id 9443, seq 13, length 64
```

[3.1.F]

What about a VLAN based solution? Will it work to provide isolation? What are the limitations of this solution? No need to perform experiments for this question.

Yes a VLAN solution would provide the isolation needed for this situation since we can restrict the broadcast domain and configure it so that the tenants would not able to reach the other tenants.. But when we see the downside of it , the admin would have to configure all the VLAN interfaces which would be a limitation and depends on the scale of VM's each tenant has.

[3.2] Design 2:

Assumption: In our setup we have attached both VM2's (in both hypervisors) to Ovs bridge and the VM1's with the linux bridge and only the OVS bridge is connected to ens4 in both hypervisors.

[3.2.A]

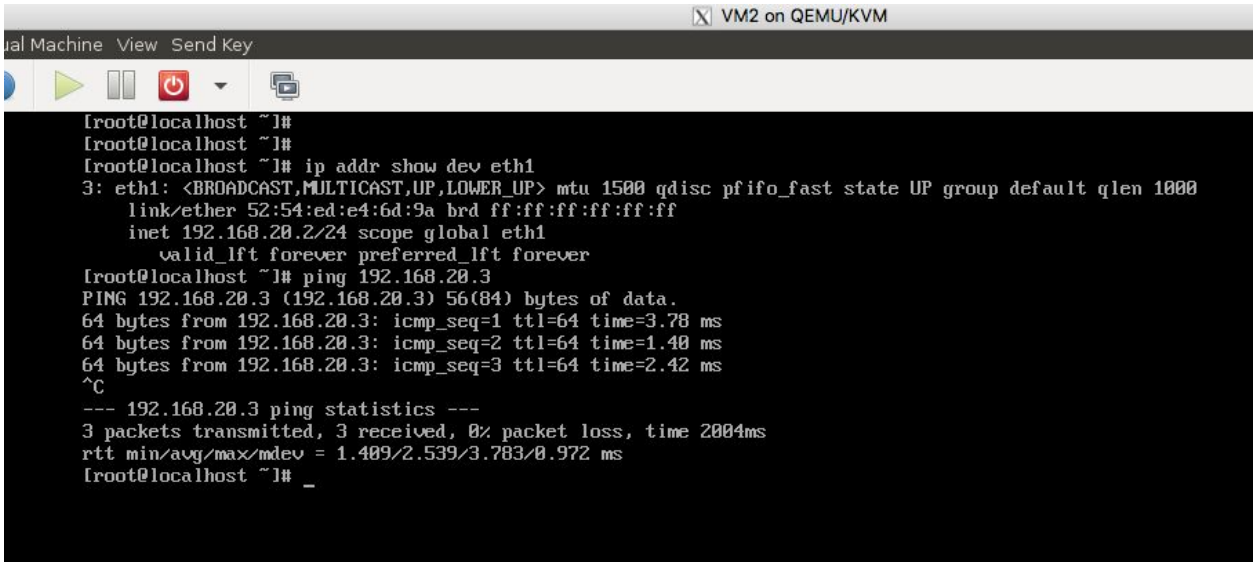
What are the disadvantages for the provider? Which resource in the hypervisor hosts will be a bottleneck?

Everytime we have a new tenant provider would have to create a new bridge and configure it which would be additional overhead which would also need a different subnet to be given by the provider. In case of multiple tenants we would have multiple bridges and we would need multiple ports at the hypervisor which would be a bottleneck issue.

[3.2.B]

What, if anything, breaks if two tenants in the same hypervisor host use the same IP address?

So in our case we do not have connectivity from VM1 in both hypervisors so basically Tenant 1 has no connections to anywhere and they can't ping anyone at all. Now keeping that in mind VM2 as in the tenant 2 since they have their own isolated connection datapath. Tenant 2 can connect between their VM's without any hassle. Since there is no other connections which can affect them. Below screenshot shows us the successful between tenant 2 VM's



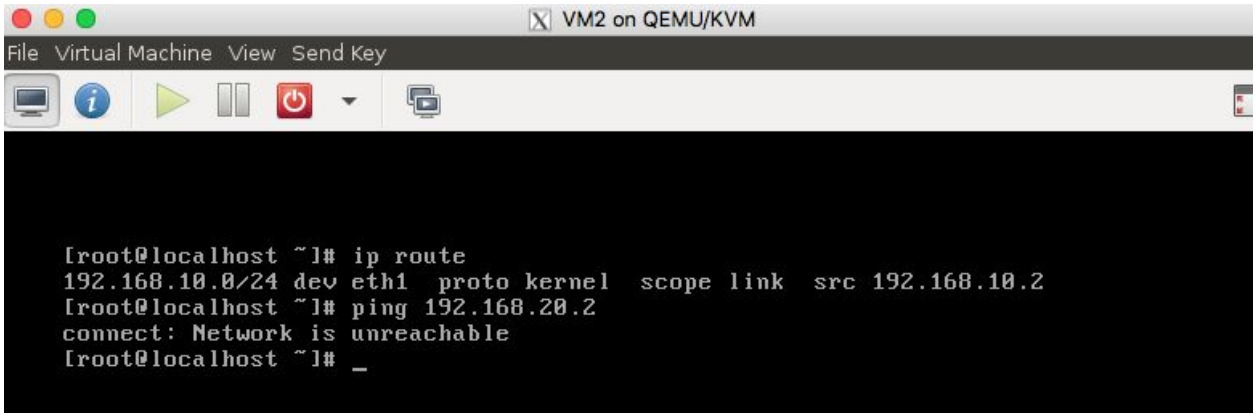
[3.2.C]

What, if anything, breaks if two tenants in a different hypervisor host use the same IP address?

Assumed Scenario:

AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same IP)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B
AT Hypervisor B: VM1 → 192.168.10.3 Tenant A
AT Hypervisor B: VM2 → 192.168.10.2 Tenant B (Same IP)

So in this now what happens is we have 2 VM's of Tenant B in 2 different subnets so a ping would not be successful since the the VM2 will not have a route to a different subnet other than itself until we explicitly give a default route. While Tenant 1 since has no connections at all wouldn't matter in this case as well.



[3.2.D]

What, if anything, breaks if two tenants in the same hypervisor host use the same MAC address?

Assumed Scenario:

AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B (Same MAC)
AT Hypervisor B: VM1 → 192.168.10.3 Tenant A
AT Hypervisor B: VM2 → 192.168.20.3 Tenant B

As we do not have any connection with the VM1 of in Hypervisor A so having same MAC at the VM2 of tenant B in Hypervisor wouldn't create any problem. There would be pings working between the VM's of tenant B without any issue.

[3.2.E]

What, if anything, breaks if two tenants in a different hypervisor host use the same MAC address?

Assumed Scenario:
AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B
AT Hypervisor B: VM1 → 192.168.10.3 Tenant A
AT Hypervisor B: VM2 → 192.168.20.3 Tenant B (Same MAC)

Same as the earlier case, even here since we have no connections between the other tenant and tenant B is isolated from tenant A. We will not have issues having same MAC. So again we observed that pings between VM2's are working without any issue.

[3.2.F]
Do we need VLANs in the hypervisor bridge or do VLANs in Physical L2 network suffice? No need to perform experiments for this question.
In this scenario if we have a VLan in the physical L2 network it would suffice which could provide isolation since we can differentiate and restrict the broadcast domains within the tenant at both hypervisors. Assuming unlike having only ens4 we also have another interface say ens5 at both hypervisors this would be possible.

[3.3] Design 1 vs. Design 2:

[3.3A] Admin Hat: List trade-offs with Design 1 and Design 2.
Design 2 when compared to Design 1 is lacking in proper connectivity since a tenant has completely no connection at all. But the benefit in design 2 would be that they are completely isolated and they wouldn't be any wrong paths for the packets which is not desired to happen. While In design 2 in cases where we have same IP or same MAC, it affects the connectivity but at least they have connectivity when compared with Design 1. So with a few restrictions over configurations in design 1. It could be used by both the tenants to communicate while design 2 completely isolates a particular tenant to have any sort of connectivity. In a Admin perspective in design 1 using VLans would provide isolation for the tenants but there would be overhead of additional configurations. As well admin has to manage additional bridges if we have more tenants in Design2 .

[3.3B] Provider hat (hypervisor host's configuration point of view): List trade-offs with Design 1 and Design 2.
In case of Design 2 we need to create 2 different networks for both the Tenants and additional bridge creation is present. While design 1 reduces the extra effort to create 2 different networks. While in Design 2, though the tenant 1 has no connectivity out of hypervisor right now but if that's needed then additional interfaces come in place and adding them to the respective bridges would be an additional task.

[4]L3 Mode:

[4.1] Design 1:

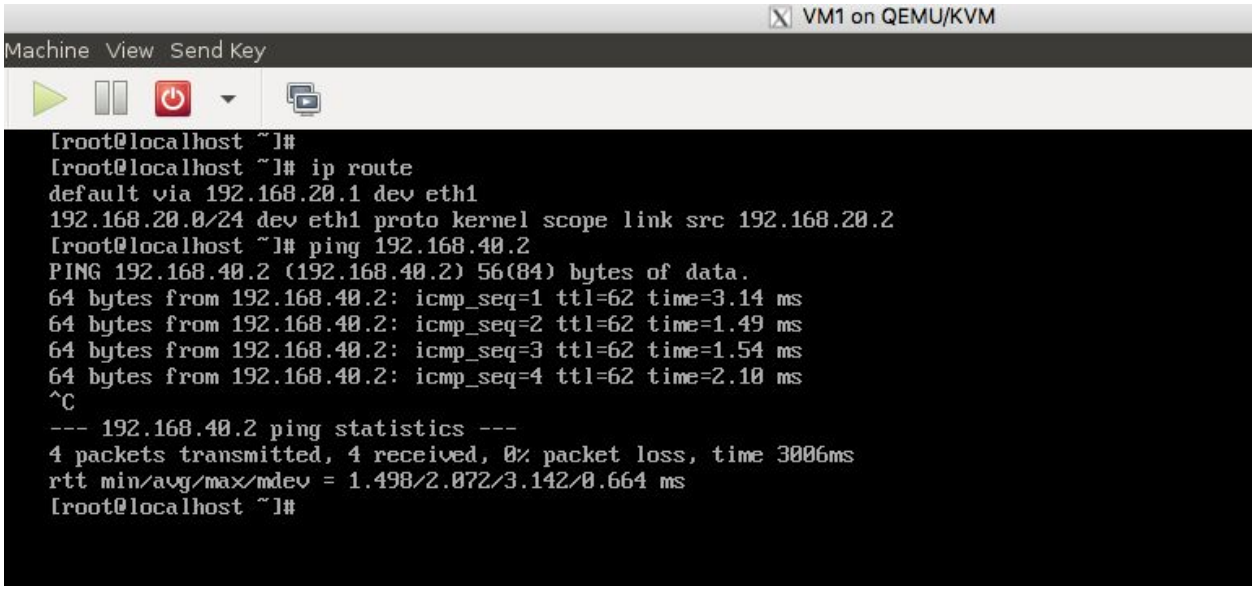
[4.1.A] What are the disadvantages for tenants? Is a tenant's traffic isolated from other tenants?

Main disadvantage being that there is no isolation between the tenants since they are connected to same bridges which are in routed mode so we will be having same subnet within a hypervisor. So with the respective to tenants their networks would be reachable by others who are not a part of tenant which is a disadvantage for the tenants.

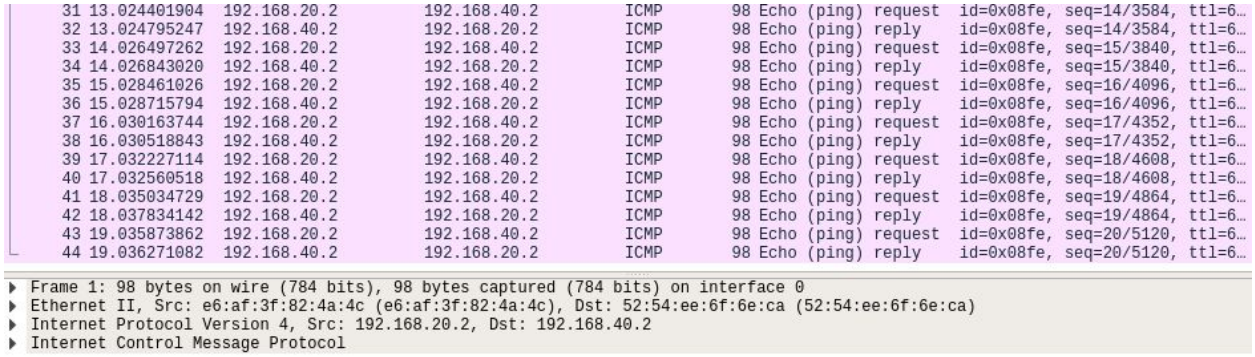
[4.1.B] What, if anything, breaks if two tenants in the same hypervisor host use the same IP address?

Assumed Scenario: (Inside a Hypervisor both have same IP's)
AT Hypervisor A: VM1 → 192.168.20.2 Tenant A
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B
AT Hypervisor B: VM1 → 192.168.40.2 Tenant A
AT Hypervisor B: VM2 → 192.168.40.2 Tenant B
Note: For all the scenarios we have configured all the required default routes at VM's and hypervisor and have assigned IP at all needed interfaces.
So in this scenario when we try to ping From VM2 in tenant A to VM2 in tenant B we would see the ping to be successful but since both the VM's in the other hypervisor have same IP addresses we would not be sure as to which VM is it actually getting connected to while VM2 is of Tenant 2 would want to connect to it's own VM in the other hypervisor. So this fails in this scenario.

Ping from VM1(HA) to desired VM1(HB):



Packet Capture at vnet8(which is a interface connected to VM2(HB):



31	13.024401904	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=14/3584, ttl=6…
32	13.024795247	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=14/3584, ttl=6…
33	14.026497262	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=15/3840, ttl=6…
34	14.026843020	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=15/3840, ttl=6…
35	15.028461026	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=16/4096, ttl=6…
36	15.028715794	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=16/4096, ttl=6…
37	16.030163744	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=17/4352, ttl=6…
38	16.030518843	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=17/4352, ttl=6…
39	17.032227114	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=18/4608, ttl=6…
40	17.032560518	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=18/4608, ttl=6…
41	18.035034729	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=19/4864, ttl=6…
42	18.037834142	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=19/4864, ttl=6…
43	19.035873862	192.168.20.2	192.168.40.2	ICMP	98 Echo (ping) request id=0x08fe, seq=20/5120, ttl=6…
44	19.036271082	192.168.40.2	192.168.20.2	ICMP	98 Echo (ping) reply id=0x08fe, seq=20/5120, ttl=6…

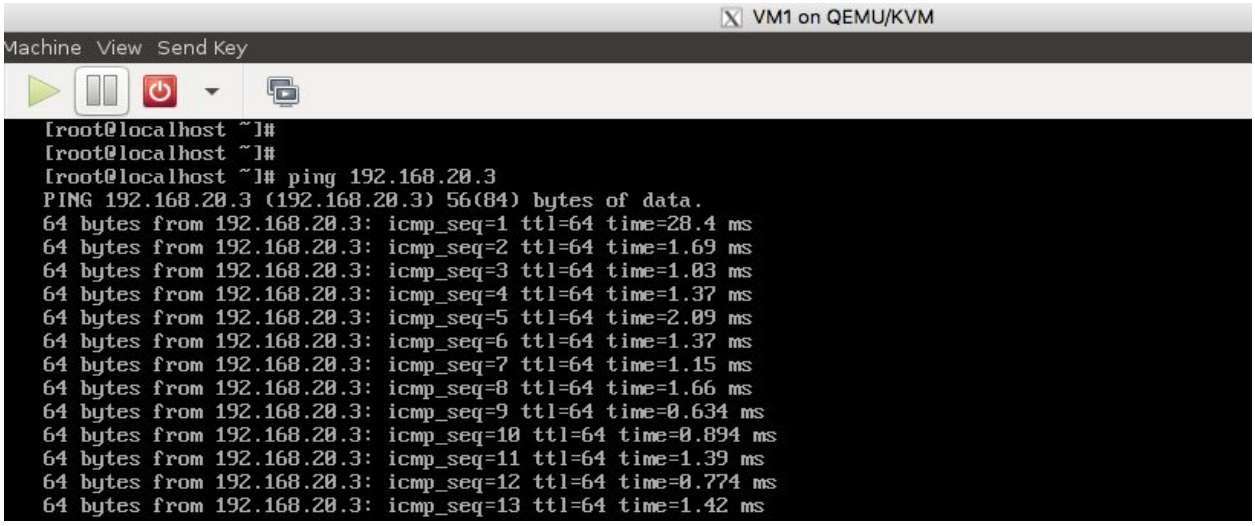
```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸ Ethernet II, Src: e6:af:3f:82:4a:4c (e6:af:3f:82:4a:4c), Dst: 52:54:ee:6f:6e:ca (52:54:ee:6f:6e:ca)
▸ Internet Protocol Version 4, Src: 192.168.20.2, Dst: 192.168.40.2
▸ Internet Control Message Protocol
```

[4.1.C] What, if anything, breaks if two tenants in a different hypervisor host use the same IP address?

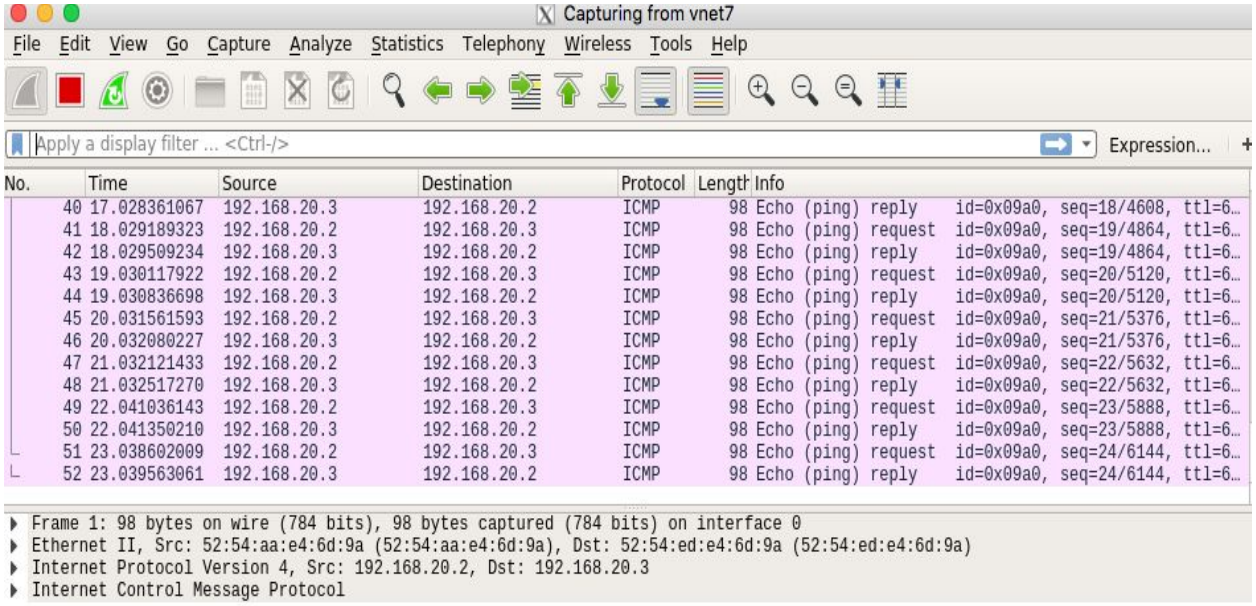
AT Hypervisor A: VM1 → 192.168.20.2 Tenant A
AT Hypervisor A: VM2 → 192.168.20.3 Tenant B
AT Hypervisor B: VM1 → 192.168.20.3 Tenant A
AT Hypervisor B: VM2 → 192.168.20.2 Tenant B

Now in this scenario again we have configured all the bridges and VM's with the above configuration also by changing the default routes as well.
When we try to ping from VM1(HA) to desiring VM1(HB), we would see a successful but then the other VM2 in our own Hypervisor which belongs to tenant 2 also has the same IP address. So the packet is actually reaching there instead of the one desired attached is below the screenshot which shows that the packets are reaching the interface at the VM2 in the same Hypervisor. This behaviour is not desired as we shouldn't let 2 different tenants connect each other.

Pinging from VM1(HA) to desiring VM1(HB):



Packet Capture at VM2(HA) interface:

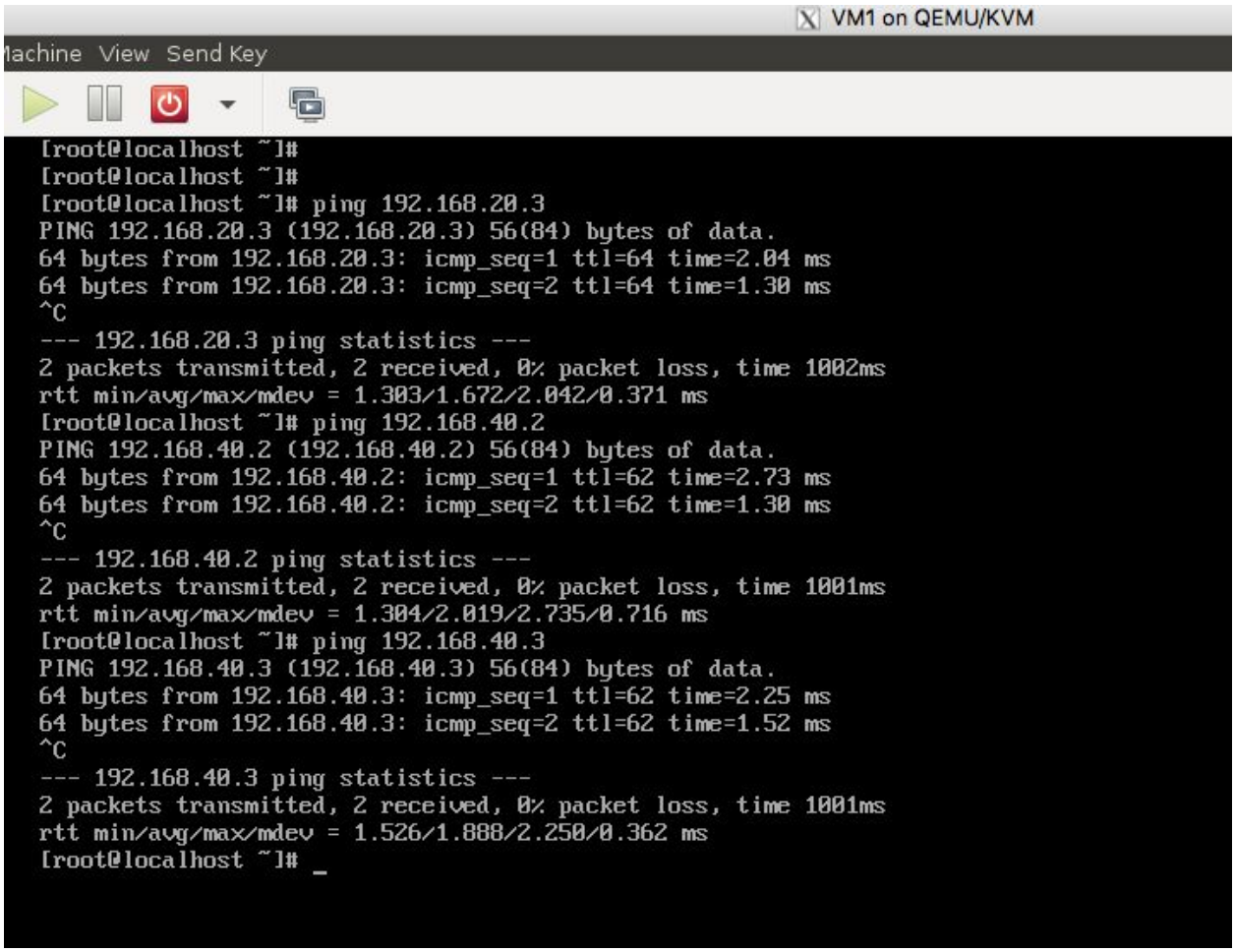


[4.1.D] What, if anything, breaks if two tenants in the same hypervisor host use the same MAC address?

AT Hypervisor A: VM1 → 192.168.20.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.3 Tenant B (Same MAC)
AT Hypervisor B: VM1 → 192.168.40.2 Tenant A
AT Hypervisor B: VM2 → 192.168.40.3 Tenant B

So in this setup firstly we will have pings successful for all of the VM's since everything is a L3 datapath and MAC wouldn't be an issue. But a same MAC would mean loss of packets at the desired VM we are pinging since there is already a entry in the MAC it would first send few icmp packets and once there is no reply re-initiates an arp request and then updates it. And again this occurs in a certain sequence of pings due to which the arp table changes. So that would be the only issue we would have but again Tenant's do no have isolation in general since they are using the same bridge and being routed mode. They can ping any VM's

Pings to everyone:



Ping to VM2(HA):

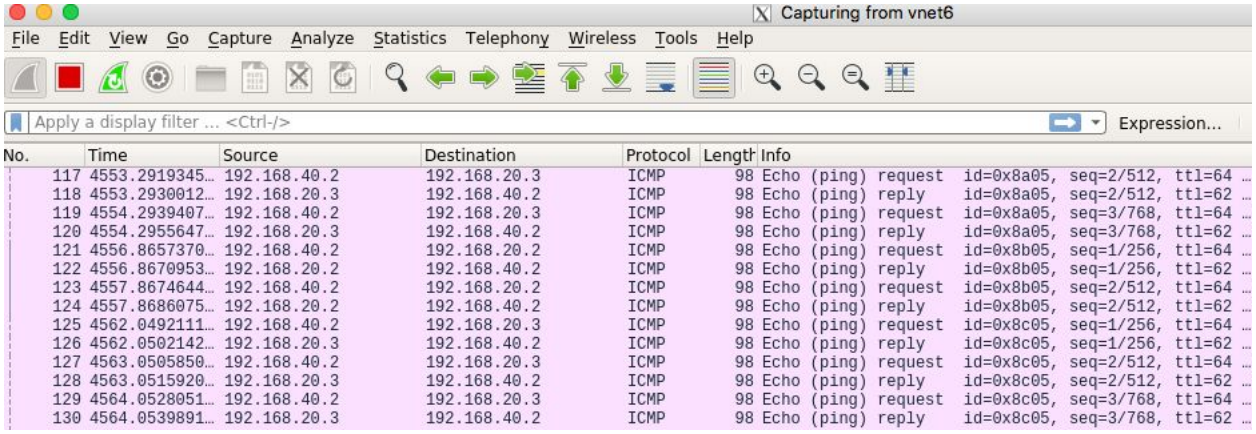
As we can see below the few icmp packets are missing due to the same MAC address for both the VM's in the same hypervisor.

```
    link/ether 52:54:00:e4:6d:9a brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:aa:e4:6d:9a brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.2/24 scope global eth1
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:ab:e8:6f:9a brd ff:ff:ff:ff:ff:ff
[root@localhost ~]# ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3) 56(84) bytes of data.
64 bytes from 192.168.20.3: icmp_seq=10 ttl=64 time=2.38 ms
64 bytes from 192.168.20.3: icmp_seq=11 ttl=64 time=0.768 ms
64 bytes from 192.168.20.3: icmp_seq=12 ttl=64 time=0.735 ms
64 bytes from 192.168.20.3: icmp_seq=13 ttl=64 time=1.02 ms
64 bytes from 192.168.20.3: icmp_seq=14 ttl=64 time=0.860 ms
64 bytes from 192.168.20.3: icmp_seq=15 ttl=64 time=0.879 ms
64 bytes from 192.168.20.3: icmp_seq=16 ttl=64 time=0.771 ms
^C
--- 192.168.20.3 ping statistics ---
16 packets transmitted, 7 received, 56% packet loss, time 15009ms
rtt min/avg/max/mdev = 0.735/1.060/2.388/0.550 ms
[root@localhost ~]#
```

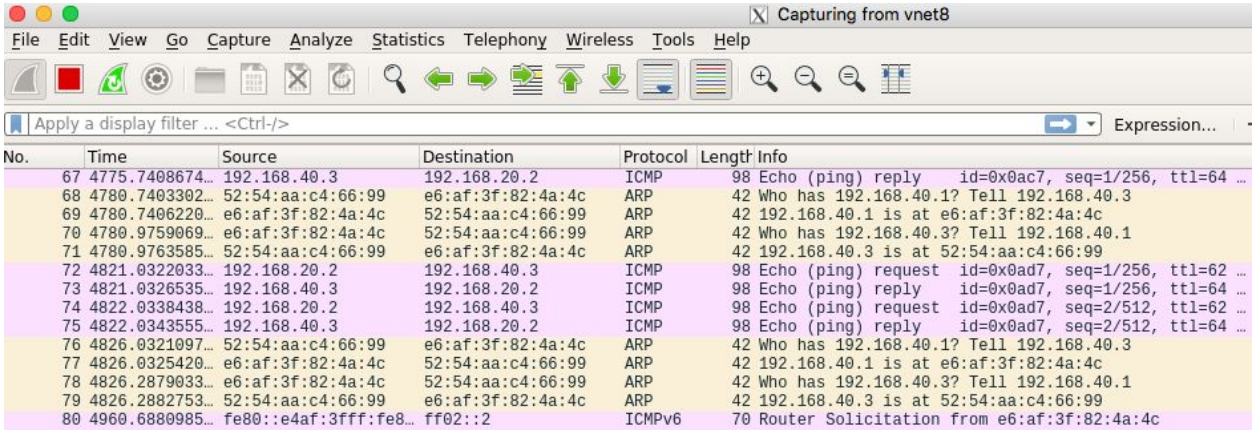
Supporting Interfaces:

```
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM1
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet5      network    default1   rtl8139     52:54:00:6f:6e:ca
vnet6      bridge     sw2_network virtio      52:54:ee:6f:6e:ca
vnet11     bridge     sw3_network virtio      52:54:ae:6b:6e:ca

[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM2
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet7      network    default1   rtl8139     52:54:00:d5:42:2d
vnet8      bridge     sw2_network virtio      52:54:de:d5:42:2d

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
```

Packet Capture at VM1(HB) Interface:



Packet Capture at VM2(HB) Interface:

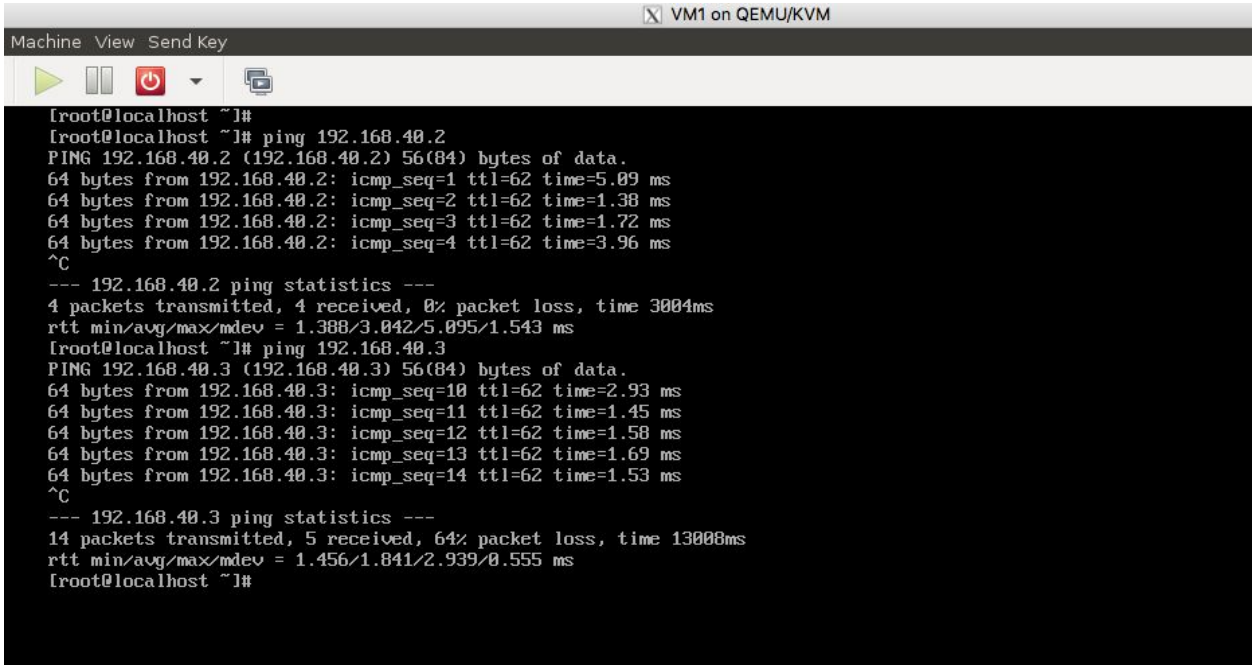


[4.1.E] What, if anything, breaks if two tenants in a different hypervisor host use the same MAC address?

AT Hypervisor A: VM1 → 192.168.20.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.3 Tenant B
AT Hypervisor B: VM1 → 192.168.40.2 Tenant A
AT Hypervisor B: VM2 → 192.168.40.3 Tenant B (Same MAC)

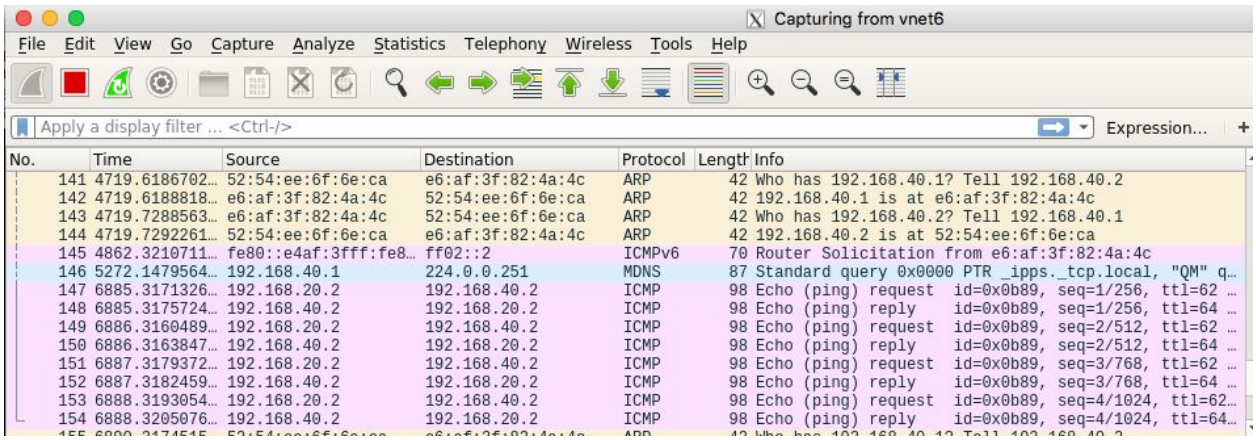
So in this scenario a similar case as above occurs but given that Tenant A tries to ping a VM at tenant B else if tenants ping to VM belonging to their own tenant then there wouldn't be an issue. Since having same bridge lets us connect between different tenants as well which is not actually desired and shouldn't happen.

So we have first pinged VM1(HB) and then VM2(HB):

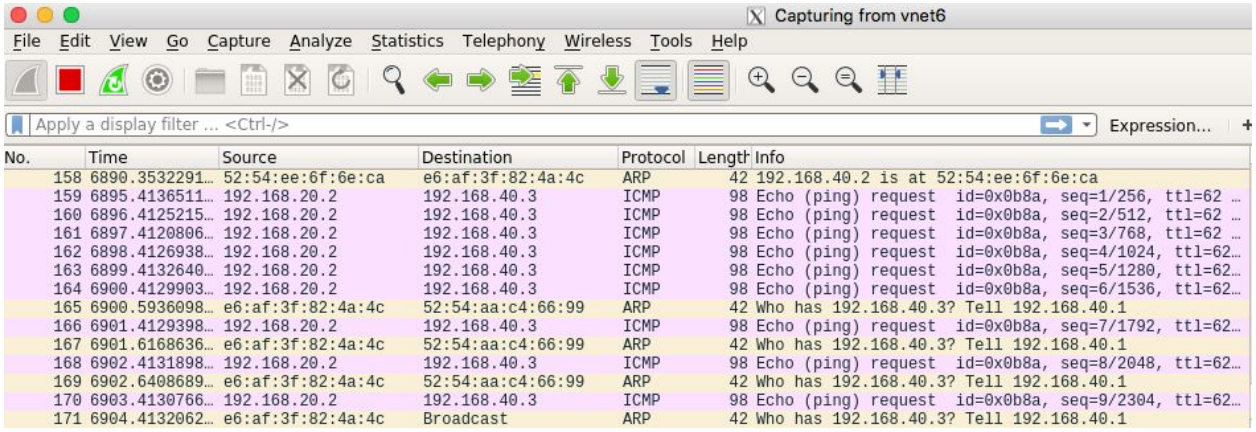


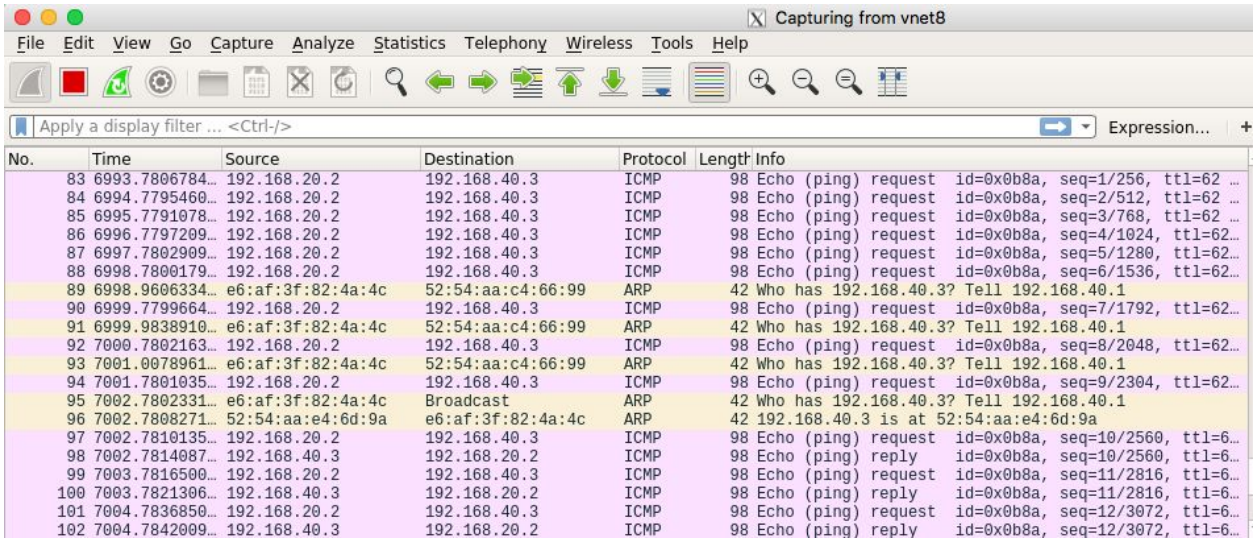
So as we can the ping to VM2(HB) a few packets went missing logically speaking it shouldn't be a problem since tenant's wouldn't want to connect others but given the design this takes place if they do try to ping.
We can also see the packet captures where those first 9 packets went to.

Ping to VM1(HB) from VM1(HA):



At VM1 and VM2 interface during the 2nd ping:





No.	Time	Source	Destination	Protocol	Length	Info
83	6993.7806784…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=1/256, ttl=62 …
84	6994.7795460…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=2/512, ttl=62 …
85	6995.7791078…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=3/768, ttl=62 …
86	6996.7797209…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=4/1024, ttl=62…
87	6997.7802909…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=5/1280, ttl=62…
88	6998.7800179…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=6/1536, ttl=62…
89	6998.9606334…	e6:af:3f:82:4a:4c	52:54:aa:c4:66:99	ARP	42	Who has 192.168.40.3? Tell 192.168.40.1
90	6999.7799664…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=7/1792, ttl=62…
91	6999.9838910…	e6:af:3f:82:4a:4c	52:54:aa:c4:66:99	ARP	42	Who has 192.168.40.3? Tell 192.168.40.1
92	7000.7802163…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=8/2048, ttl=62…
93	7001.0078961…	e6:af:3f:82:4a:4c	52:54:aa:c4:66:99	ARP	42	Who has 192.168.40.3? Tell 192.168.40.1
94	7001.7801035…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=9/2304, ttl=62…
95	7002.7802331…	e6:af:3f:82:4a:4c	Broadcast	ARP	42	Who has 192.168.40.3? Tell 192.168.40.1
96	7002.7808271…	52:54:aa:e4:6d:9a	e6:af:3f:82:4a:4c	ARP	42	192.168.40.3 is at 52:54:aa:e4:6d:9a
97	7002.7810135…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=10/2560, ttl=6…
98	7002.7814087…	192.168.40.3	192.168.20.2	ICMP	98	Echo (ping) reply id=0x0b8a, seq=10/2560, ttl=6…
99	7003.7816500…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=11/2816, ttl=6…
100	7003.7821306…	192.168.40.3	192.168.20.2	ICMP	98	Echo (ping) reply id=0x0b8a, seq=11/2816, ttl=6…
101	7004.7836850…	192.168.20.2	192.168.40.3	ICMP	98	Echo (ping) request id=0x0b8a, seq=12/3072, ttl=6…
102	7004.7842009…	192.168.40.3	192.168.20.2	ICMP	98	Echo (ping) reply id=0x0b8a, seq=12/3072, ttl=6…

Here you can see the missing packets which went towards the VM1 interface though they didn't get a reply and then again an arp request resolves it.
So in general apart from that usually pings to between everyone wouldn't have any issue.

[4.1.F] What about a VLAN based solution for providing L3 connectivity to each VM? Will it work? What are the limitations of this solution?

In this design a Vlan based solution for providing L3 connectivity would work but not complete isolation. Since a network used within a Hypervisor cannot be used by another tenant (as in same IP cannot be used) in other hypervisor so within a hypervisor we would have isolation. This would be limitation of this solution as well.

[4.2] Design 2:

[4.2.A] What are the disadvantages for the provider? Which resource in the hypervisor hosts will be a bottleneck?

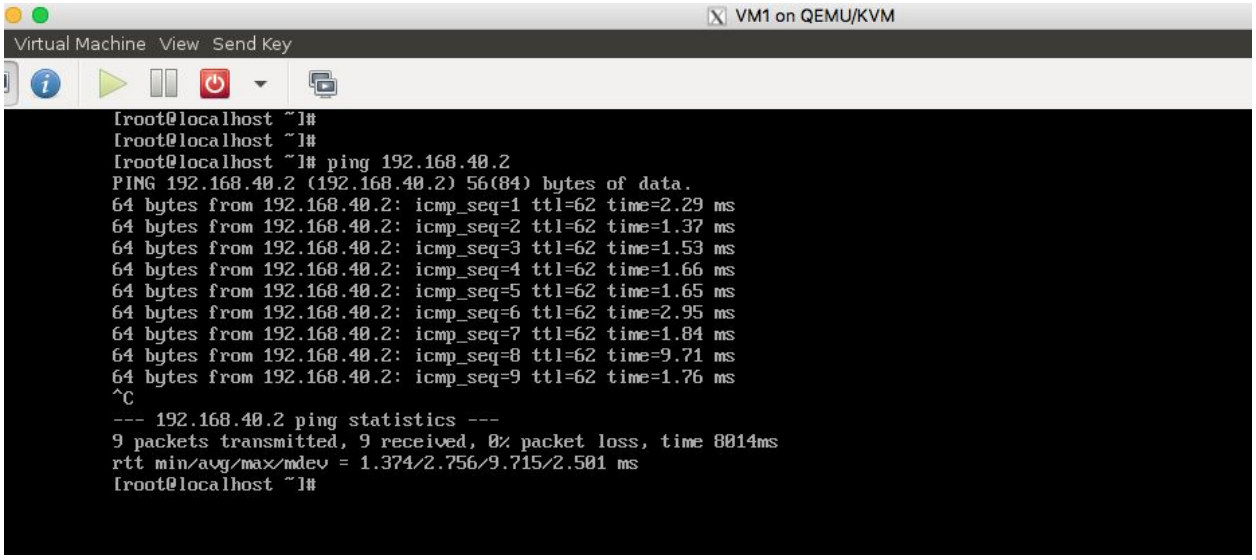
As the number of tenants increases the provider would have to increase the number of bridges as well which would be a disadvantage with respect to a provider. Also having different bridges would mean multiple routes have to be configured. Bottleneck would be that since we would have increase in number of bridges as the tenants increase and which will cause increase in number of ports as well. This leads to a bottleneck solution.

[4.2.B] What, if anything, breaks if two tenants in the same hypervisor host use the same IP address?

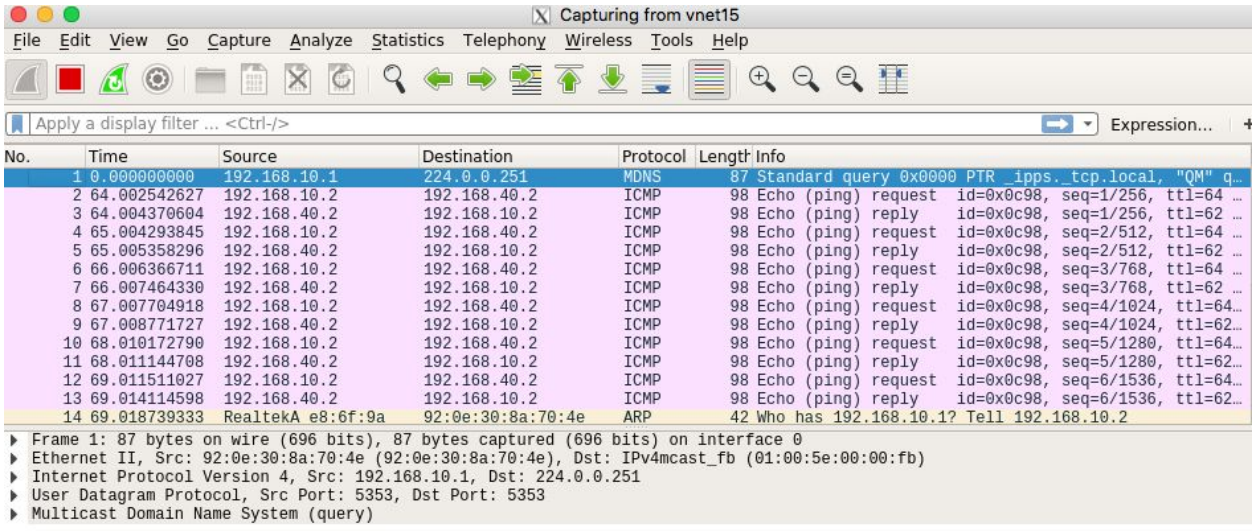
AT Hypervisor A: VM1 → 192.168.10.2 Tenant A
AT Hypervisor A: VM2 → 192.168.10.2 Tenant B
AT Hypervisor B: VM1 → 192.168.40.2 Tenant A
AT Hypervisor B: VM2 → 192.168.40.2 Tenant B

So in this configuration we would want to ping from VM1(HA) to VM1(HB) :
So when we have tried this we found out that the the ping packets are actually going to VM2(HB) which belong to different tenant this happens because of the same IP address in the same Hypervisor. So whichever route first is found it will get connected with that VM. This type of behaviour is definitely not desired. While trying to ping other VM in same hypervisor is nothing but a self ping and we wouldn't want to ping another tenant anyways.

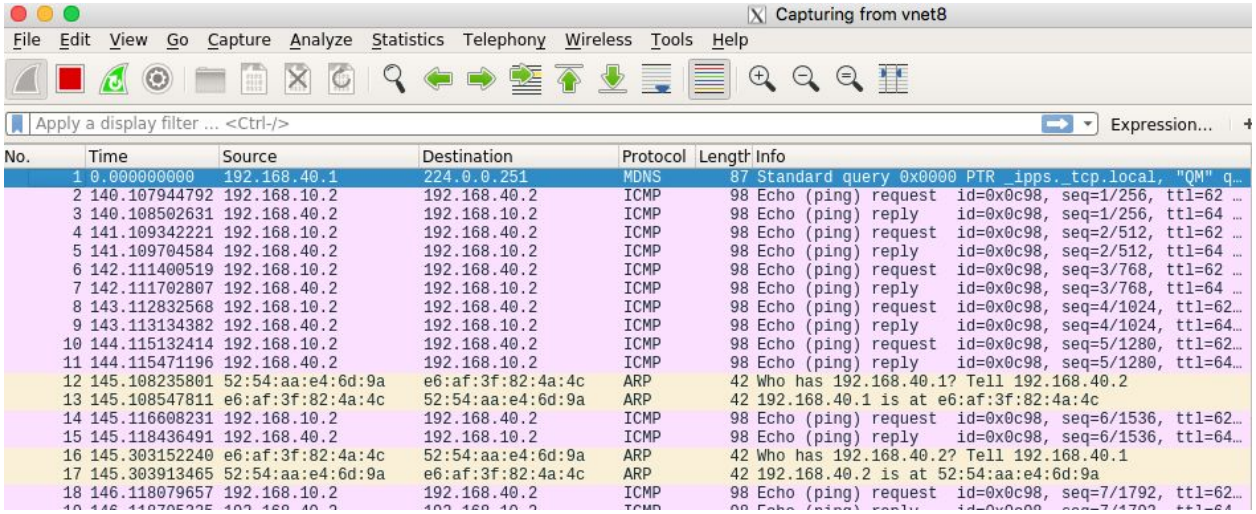
SSH from VM1(HA) desired for VM1(HB):



Packet Capture at VM1(HA) interface:



Packet Capture at VM2(HB) Interface:

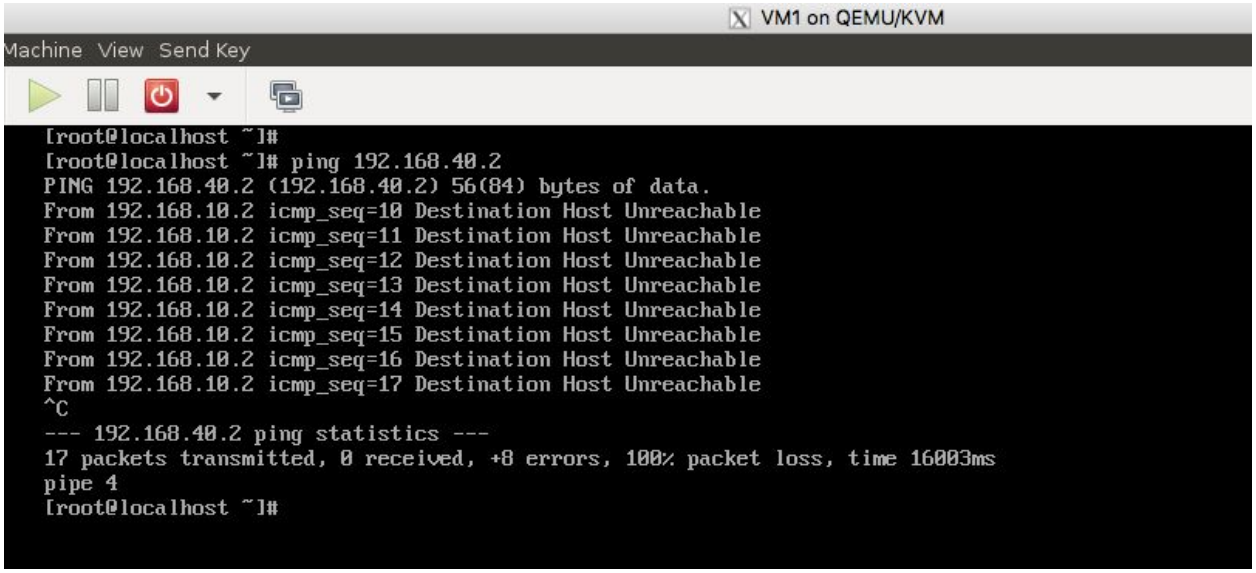


[4.2.C] What, if anything, breaks if two tenants in a different hypervisor host use the same IP address?

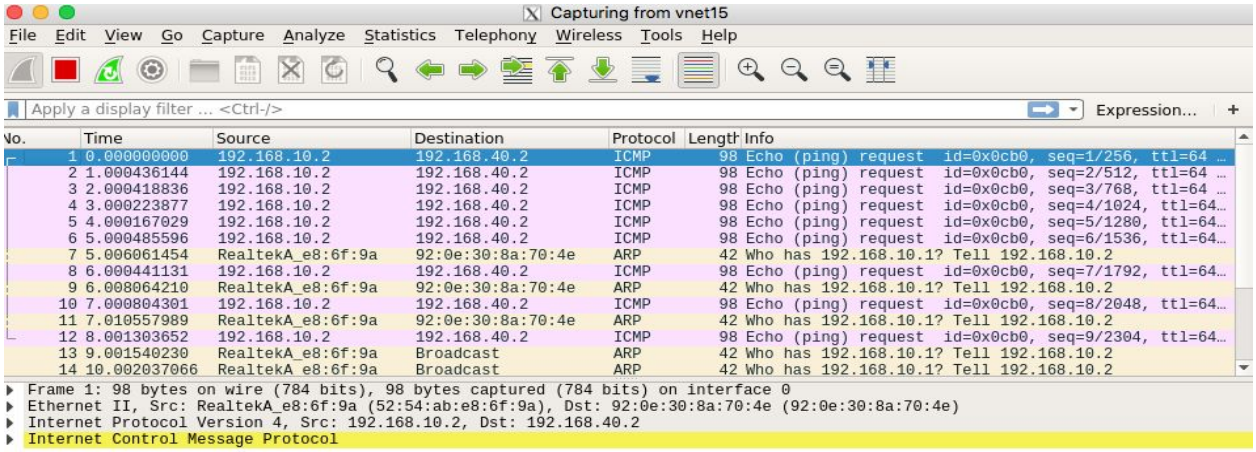
AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same IP)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B
AT Hypervisor B: VM1 → 192.168.40.2 Tenant A
AT Hypervisor B: VM2 → 192.168.10.2 Tenant B (Same IP)

So now here adding routes would be repetitive since inside hypervisor there would be routes to 10.0/24 subnet through the bridge interface but we would also need the route to be set so that VM2(HA) shoud be able to communicate VM2(HB) since as the scenario states the tenants would want their VM's to communicate. So we additionally add the rules again for 10.0/subnet but now the problem arises since the hypervisor wouldn't know where to send the packets to.

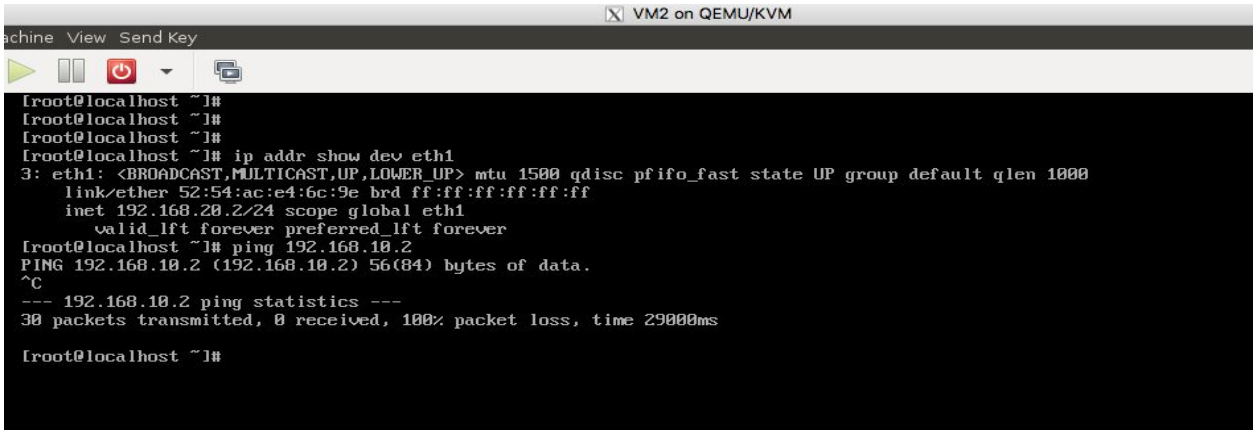
Ping from VM1(HA) to VM1(HB) which fails:



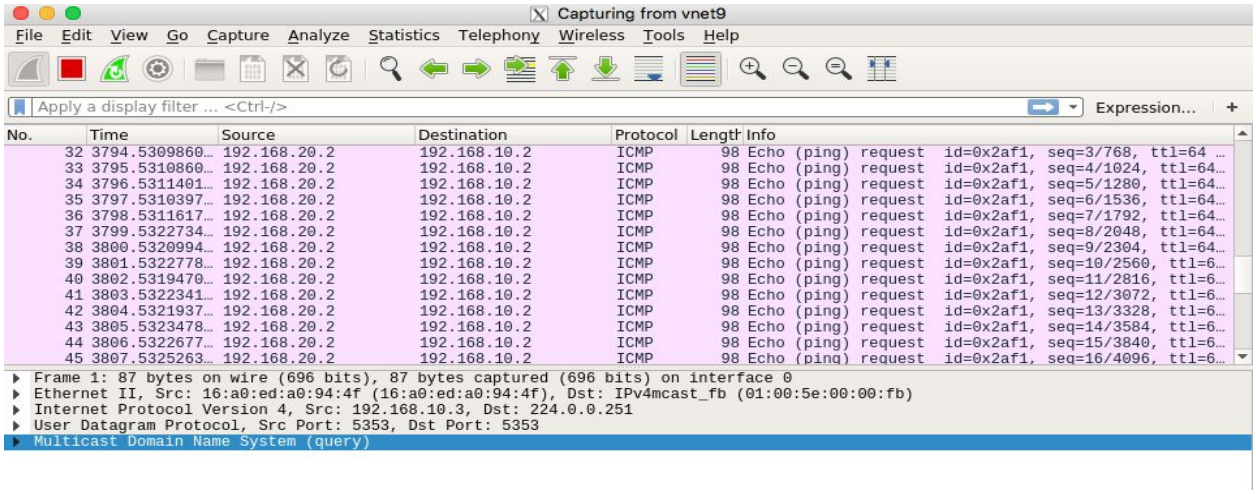
Packet Capture at the VM1(HA) interface:



Ping from VM2(HA) to VM2(HB):



Packet Capture:



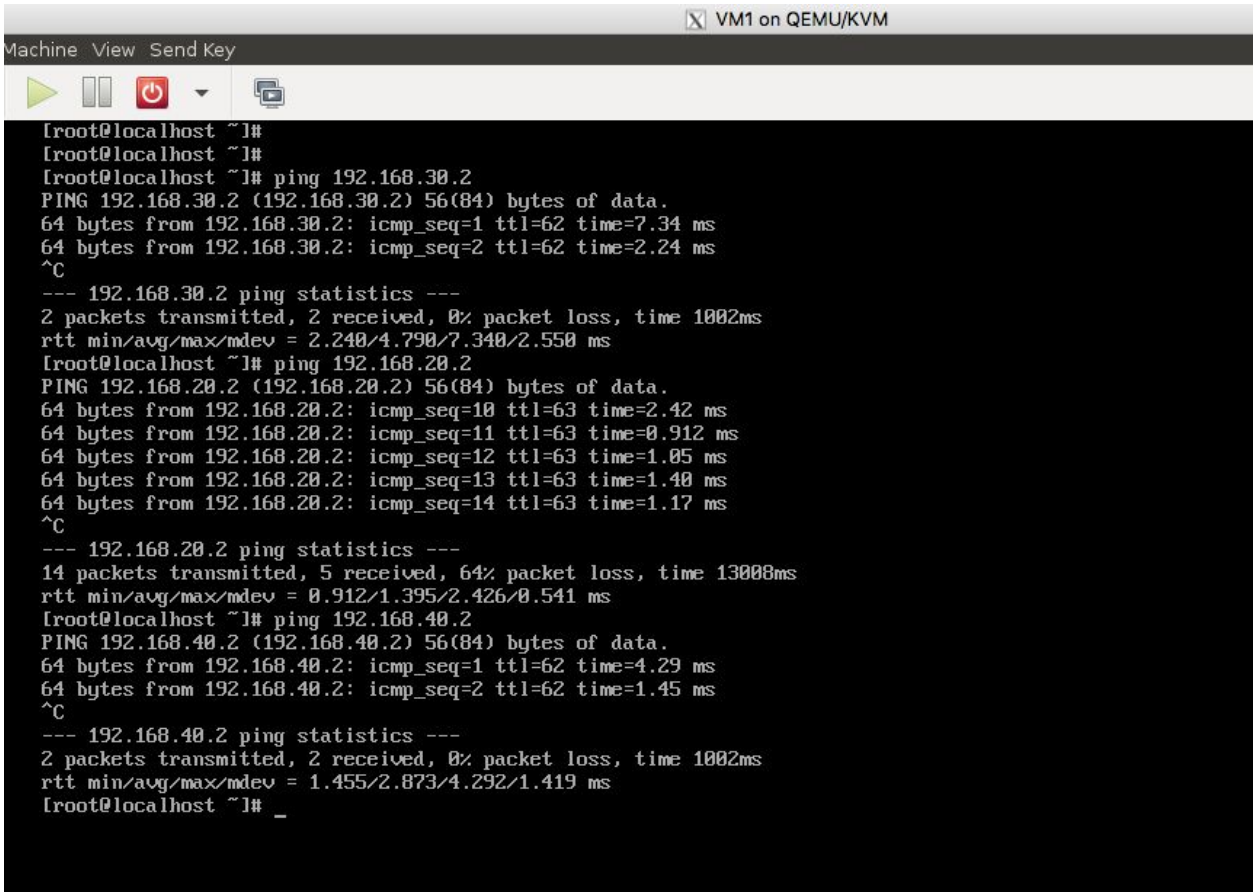
So this experiment in based on the assumption that since the tenants would want their Vm's to communicate over L3 network we added the routes but if we do not add those there might not have been problem but yet undesired VM's would ping and no connection would be present at all.

[4.2.D] What, if anything, breaks if two tenants in the same hypervisor host use the same MAC address?

AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B (Same MAC)
AT Hypervisor B: VM1 → 192.168.30.2 Tenant A
AT Hypervisor B: VM2 → 192.168.40.2 Tenant B

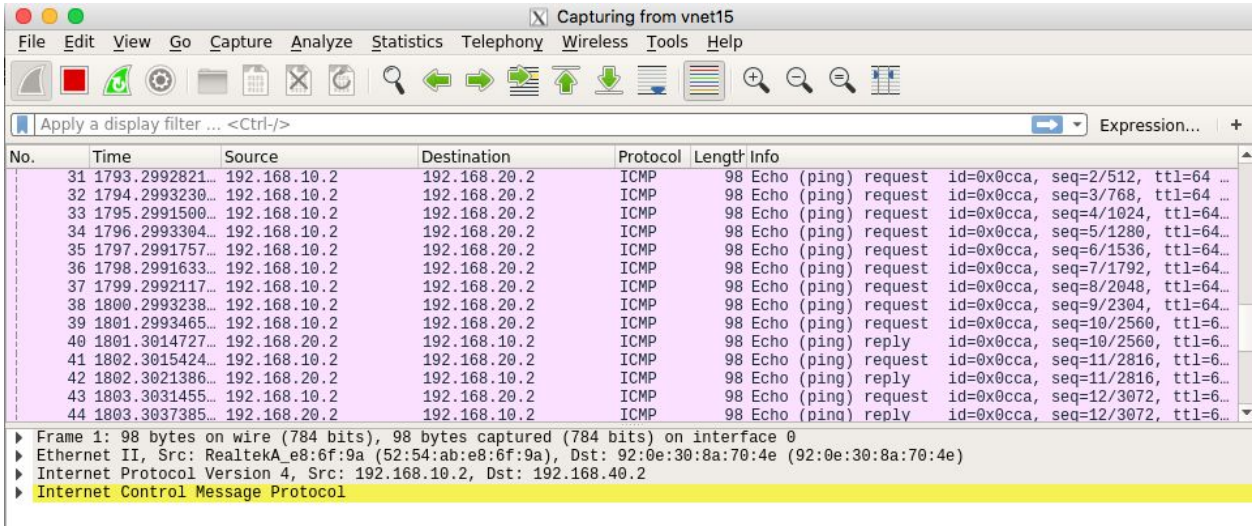
In this scenario if the tenants are reaching out to only their VM's then there wouldn't be any issue but if tenant A is trying to reach the VM with same MAC in the same hypervisor then they will be connection eventually but a few packets would be lost. But practically a tenant wouldn't want to reach some other tenant. So if we think in that way there wouldn't be a problem since our situation has different subnets as well. But if something that would be problem would be the ping from VM1(HA) to VM2(HB) which logically shouldn't be bothered about.

Ping from VM1(HA) to everyone else:

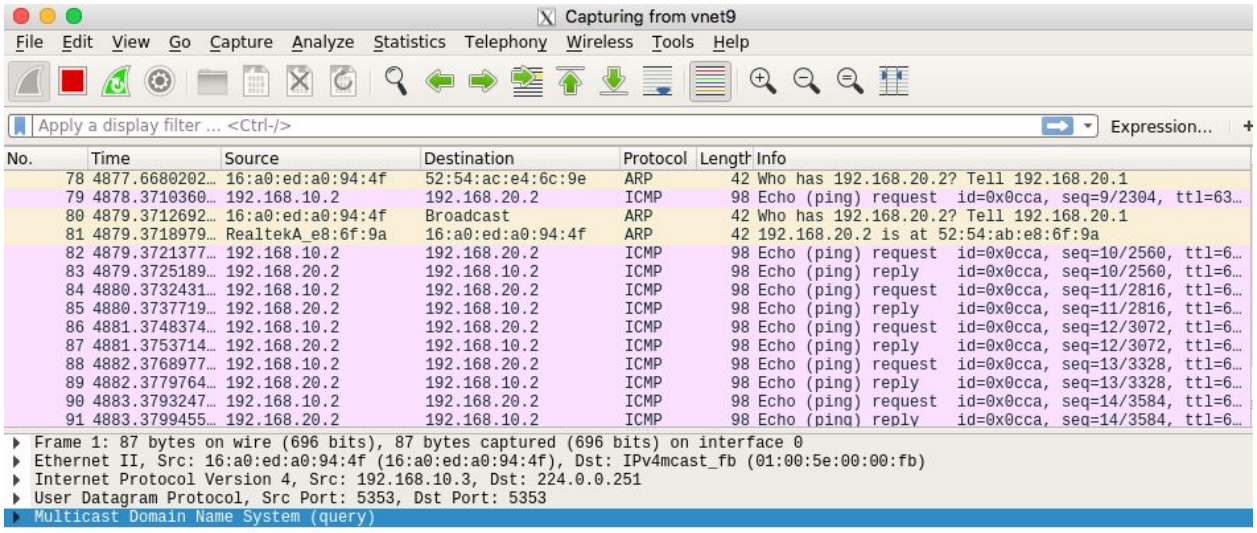


As you can see when we ping VM2(HA) a few packets were lost.

Packet Capture at VM1 Interface:



Packet Capture at VM2(HA) interface:

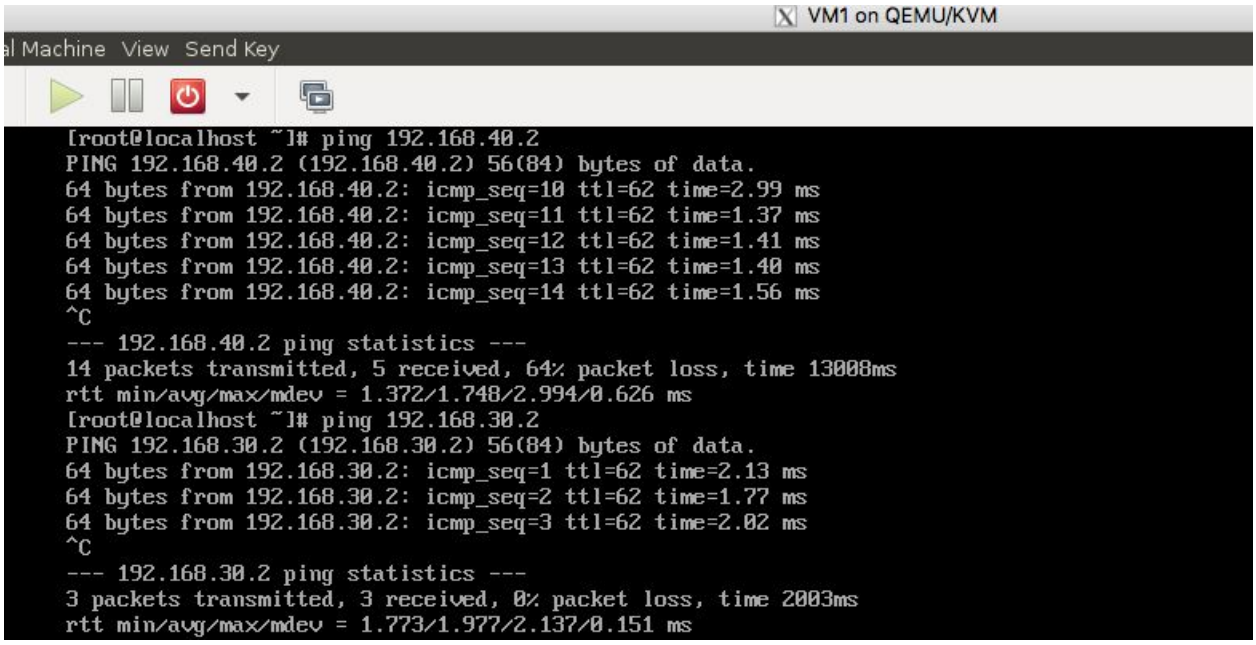


[4.2.E] What, if anything, breaks if two tenants in a different hypervisor host use the same MAC address?

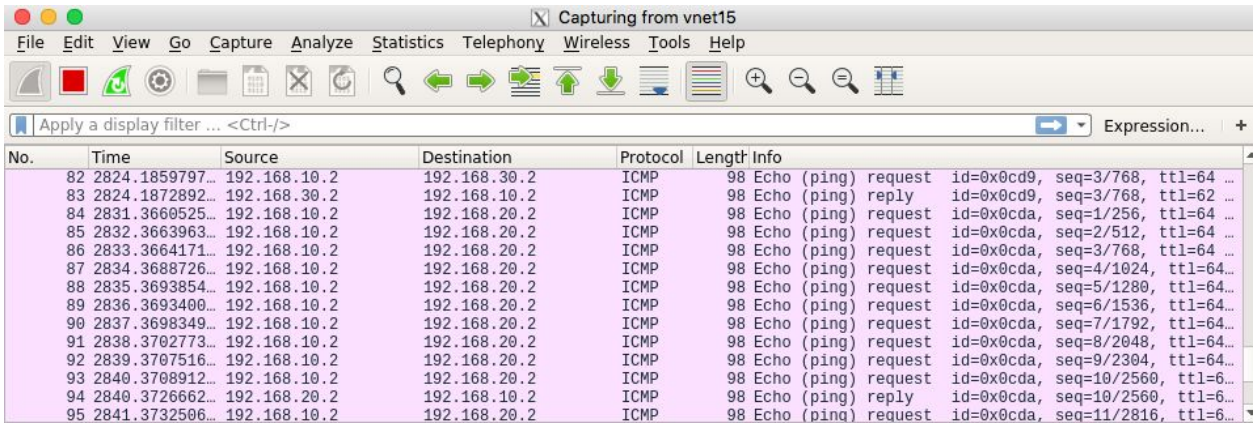
AT Hypervisor A: VM1 → 192.168.10.2 Tenant A (Same MAC)
AT Hypervisor A: VM2 → 192.168.20.2 Tenant B
AT Hypervisor B: VM1 → 192.168.30.2 Tenant A
AT Hypervisor B: VM2 → 192.168.40.2 Tenant B (Same MAC)

In this scenario again we have a similar situation where having the same MAC would lead to few missing packets which is not desired but again that is only when a tenant in trying to ping VM's of the other tenant which logically is not desired. But if that's not needed then the pings to anyone would be eventually successful.

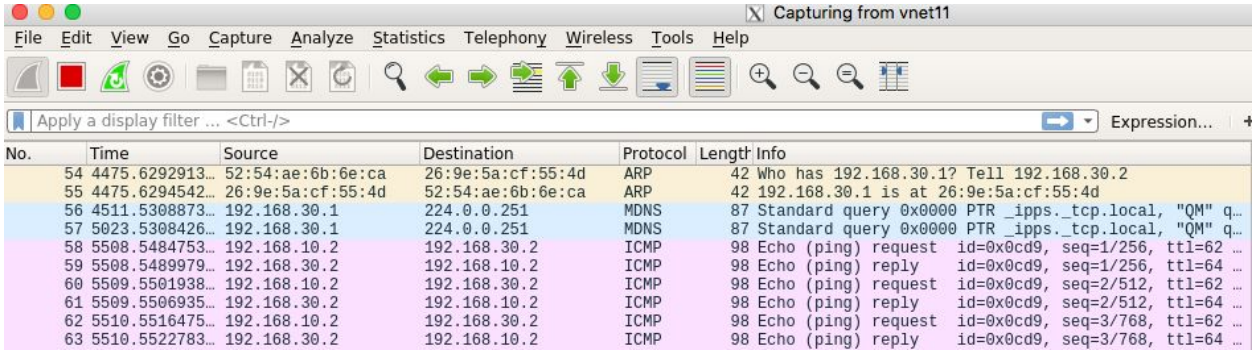
Ping from VM1(HA) to everyone else:



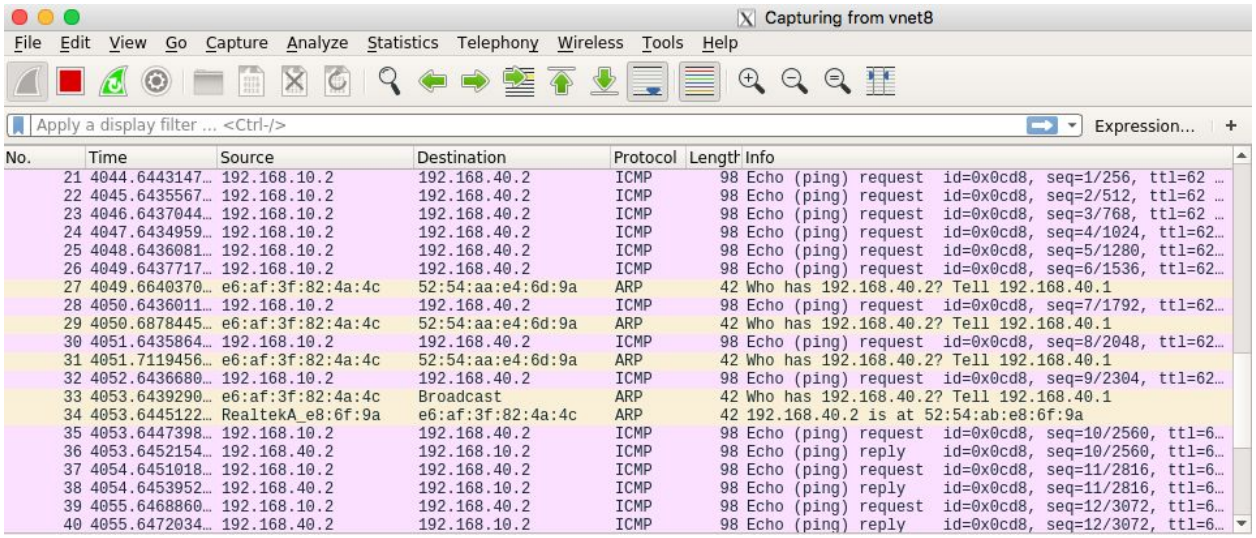
Packet Capture at VM1(HA) interface:



Packet Capture at VM1(HB) interface:



Packet Capture at VM2(HB) interface:



[4.2.F] Does a VLAN based solution in this design overcome any limitations of the VLAN based solution used in design 1?

No, the same limitation would still persist since the connectivity between the hypervisors is the same datapath.

[4.3.A] Design 1 vs. Design 2:

Admin Hat: List trade-offs with Design 1, Design 2, and Design 3.

While with respect to a Admin, in case of Design 1 you would have less number of bridges and routes to manage since you will have only 1 subnet for the bridge while in Design 2 there would be overhead of having extra interfaces and bridges which would again need you to configure the default routes accordingly. While incase of Design 3 there will be a lot overhead since there would be multiple bridges which would again need multiple default routes to be set at the VM's.

[4.3.B] Provider hat (hypervisor host's configuration point of view): List trade-offs with Design 1, Design 2, and Design 3.

Design 1 would let the provider network burden less but it cannot provide the isolation for the tenants though and the network might be affected when they have same MAC addresses.
While in Design 2 additionally the provider would have to configure routes at both the hypervisors which still wouldn't provide isolation since all the bridges are in routed mode.
In Design 3 the provider has to make sure there no same subnets are being used and also the number of routes to be added on both ends would keep increasing as the number of VM's increases.

[4.3C] What are the missing pieces to have complete network isolation (IP, MAC, L2/L3 FT) between tenants in the hypervisor network?

For complete network isolation between tenants in the hypervisor network we can use iptables and configure rules which would restrict traffic as per our requirements and also using namespaces we can provide isolation. With the combination of both we can also configure complex policies as per our need. With this we can provide complete network isolation.

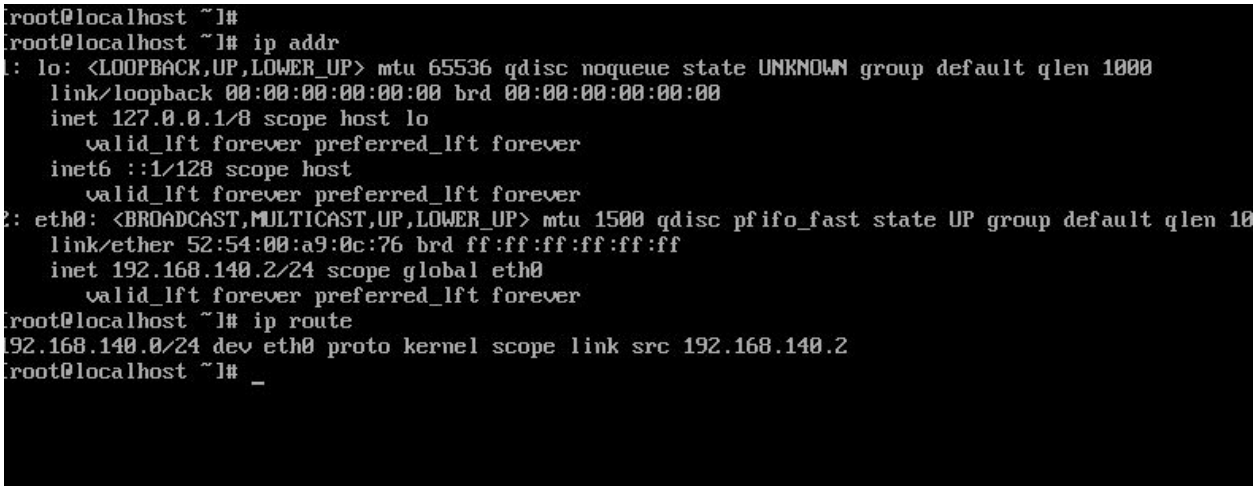
[5] IPTable, NAT/PAT:

[5.1] Topology Configurations:

Client: Public Network IP: 192.168.130.2
Router VM: Public Network IP: 192.168.130.1
Private Network IP: 192.168.140.1
Server1: Private Network IP: 192.168.140.2
Server2: Private Network IP: 192.168.140.3

Screenshots supporting the configurations along with forwarding table output of each VM:

At Server 1:



At Server 2:

```
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:27:a5:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.3/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# ip route
192.168.140.0/24 dev eth0 proto kernel scope link src 192.168.140.3
[root@localhost ~]#
```

At Client:

```
[root@localhost ~]#
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:55:d5:46 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:aa:39:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.130.2/24 scope global eth1
       valid_lft forever preferred_lft forever
[root@localhost ~]# ip route
192.168.130.0/24 dev eth1 proto kernel scope link src 192.168.130.2
[root@localhost ~]# _
```

At Router VM:

```
[root@localhost ~]#
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:47:0f:01 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:bb:39:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.130.1/24 scope global eth1
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:cc:39:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.1/24 scope global eth2
       valid_lft forever preferred_lft forever
[root@localhost ~]# ip route
192.168.130.0/24 dev eth1 proto kernel scope link src 192.168.130.1
192.168.140.0/24 dev eth2 proto kernel scope link src 192.168.140.1
[root@localhost ~]# _
```

[5.2] Configuring NAT on Router VM so that servers can ping the client

Now for the servers to ping the client we need to configure NAT at the Router VM which is done by the following command.

Iptables -t nat -A POSTROUTING -s 192.168.140.2 -p icmp -j SNAT --to-source 192.168.130.1
Iptables -t nat -A POSTROUTING -s 192.168.140.3 -p icmp -j SNAT --to-source 192.168.130.1

The commands above configures the NAT Operation on Router VM and we have the following rules as shown in the screenshot below:

```
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       icmp --  192.168.140.2        anywhere             to:192.168.130.1
SNAT       icmp --  192.168.140.3        anywhere             to:192.168.130.1

Chain OUTPUT_direct (0 references)
target     prot opt source               destination
target     prot opt source               destination

Chain POSTROUTING_ZONES (0 references)
target     prot opt source               destination

Chain POSTROUTING_ZONES_SOURCE (0 references)
target     prot opt source               destination

Chain POSTROUTING_direct (0 references)
target     prot opt source               destination

Chain POST_public (0 references)
```

Now we can see that the Pings from Server 1 and Server 2 are successful to Client on the Public Network.
We have created a default route to reach the Router_VM from Server1 and Server2

Successful Ping from Server 1 to Client

```
[root@localhost ~]#
[root@localhost ~]# ping 192.168.130.2
PING 192.168.130.2 (192.168.130.2) 56(84) bytes of data.
64 bytes from 192.168.130.2: icmp_seq=1 ttl=63 time=2.96 ms
64 bytes from 192.168.130.2: icmp_seq=2 ttl=63 time=1.12 ms
64 bytes from 192.168.130.2: icmp_seq=3 ttl=63 time=1.24 ms
64 bytes from 192.168.130.2: icmp_seq=4 ttl=63 time=1.14 ms
64 bytes from 192.168.130.2: icmp_seq=5 ttl=63 time=1.72 ms
64 bytes from 192.168.130.2: icmp_seq=6 ttl=63 time=2.03 ms
64 bytes from 192.168.130.2: icmp_seq=7 ttl=63 time=3.99 ms
64 bytes from 192.168.130.2: icmp_seq=8 ttl=63 time=1.11 ms
64 bytes from 192.168.130.2: icmp_seq=9 ttl=63 time=1.24 ms
64 bytes from 192.168.130.2: icmp_seq=10 ttl=63 time=1.67 ms
64 bytes from 192.168.130.2: icmp_seq=11 ttl=63 time=1.69 ms
64 bytes from 192.168.130.2: icmp_seq=12 ttl=63 time=1.90 ms
64 bytes from 192.168.130.2: icmp_seq=13 ttl=63 time=1.69 ms
64 bytes from 192.168.130.2: icmp_seq=14 ttl=63 time=1.50 ms
^C
--- 192.168.130.2 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13023ms
rtt min/avg/max/mdev = 1.112/1.789/3.991/0.772 ms
[root@localhost ~]#
```

Successful Ping from Server 2 to Client:

```
[root@localhost ~]# ping 192.168.130.2
PING 192.168.130.2 (192.168.130.2) 56(84) bytes of data.
64 bytes from 192.168.130.2: icmp_seq=1 ttl=63 time=2.69 ms
64 bytes from 192.168.130.2: icmp_seq=2 ttl=63 time=1.37 ms
64 bytes from 192.168.130.2: icmp_seq=3 ttl=63 time=5.40 ms
64 bytes from 192.168.130.2: icmp_seq=4 ttl=63 time=1.14 ms
64 bytes from 192.168.130.2: icmp_seq=5 ttl=63 time=1.60 ms
64 bytes from 192.168.130.2: icmp_seq=6 ttl=63 time=1.56 ms
64 bytes from 192.168.130.2: icmp_seq=7 ttl=63 time=1.49 ms
64 bytes from 192.168.130.2: icmp_seq=8 ttl=63 time=1.48 ms
64 bytes from 192.168.130.2: icmp_seq=9 ttl=63 time=1.55 ms
64 bytes from 192.168.130.2: icmp_seq=10 ttl=63 time=1.09 ms
64 bytes from 192.168.130.2: icmp_seq=11 ttl=63 time=2.25 ms
^C
--- 192.168.130.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10019ms
rtt min/avg/max/mdev = 1.092/1.969/5.406/1.175 ms
[root@localhost ~]#
```

Now we will see how the IP addresses are changed due to the NAT configuration with wireshark captures.

Capture during Server 1 pinging at Private Network Interface: (140.2 to 130.2)

2	68.923639386	192.168.140.2	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=1/256, ttl=64 …
3	68.924984033	192.168.130.2	192.168.140.2	ICMP	98	Echo (ping) reply id=0x4650, seq=1/256, ttl=63 …
4	69.925331736	192.168.140.2	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=2/512, ttl=64 …
5	69.926386611	192.168.130.2	192.168.140.2	ICMP	98	Echo (ping) reply id=0x4650, seq=2/512, ttl=63 …
6	70.927275831	192.168.140.2	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=3/768, ttl=64 …
7	70.932044252	192.168.130.2	192.168.140.2	ICMP	98	Echo (ping) reply id=0x4650, seq=3/768, ttl=63 …
8	71.929237110	192.168.140.2	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=4/1024, ttl=64…
9	71.930143092	192.168.130.2	192.168.140.2	ICMP	98	Echo (ping) reply id=0x4650, seq=4/1024, ttl=63…
10	72.930953185	192.168.140.2	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=5/1280, ttl=64…
11	72.932177455	192.168.130.2	192.168.140.2	ICMP	98	Echo (ping) reply id=0x4650, seq=5/1280, ttl=63…

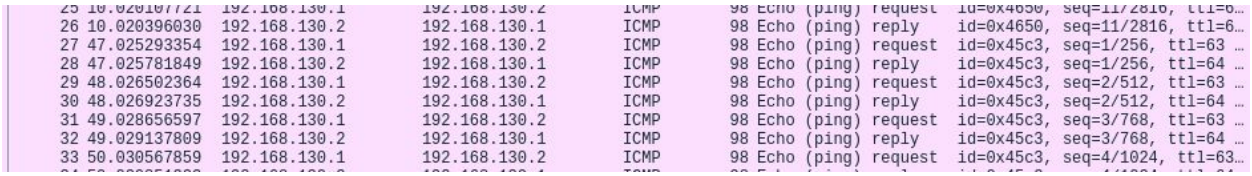
Capture during Server 2 pinging at Private Network Interface: (140.3 to 130.2)

41	120.959013488	192.168.130.2	192.168.140.3	ICMP	98	Echo (ping) reply id=0x45c3, seq=6/1536, ttl=63…
42	121.958882903	192.168.140.3	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=7/1792, ttl=64…
43	121.962489666	192.168.130.2	192.168.140.3	ICMP	98	Echo (ping) reply id=0x45c3, seq=7/1792, ttl=63…
44	122.960539550	192.168.140.3	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=8/2048, ttl=64…
45	122.961368275	192.168.130.2	192.168.140.3	ICMP	98	Echo (ping) reply id=0x45c3, seq=8/2048, ttl=63…
46	123.962043574	192.168.140.3	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=9/2304, ttl=64…
47	123.962919218	192.168.130.2	192.168.140.3	ICMP	98	Echo (ping) reply id=0x45c3, seq=9/2304, ttl=63…
48	124.964027252	192.168.140.3	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=10/2560, ttl=6…
49	124.965309568	192.168.130.2	192.168.140.3	ICMP	98	Echo (ping) reply id=0x45c3, seq=10/2560, ttl=6…
50	125.966219985	192.168.140.3	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=11/2816, ttl=6…

Capture during Server 1 pinging at Public Network Interface: (130.1 to 130.2)

1	0.000000000	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=1/256, ttl=63 …
2	0.000474001	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x4650, seq=1/256, ttl=64 …
3	1.001498040	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=2/512, ttl=63 …
4	1.001847915	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x4650, seq=2/512, ttl=64 …
5	2.003833689	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=3/768, ttl=63 …
6	2.004174079	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x4650, seq=3/768, ttl=64 …
7	3.005439870	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=4/1024, ttl=63…
8	3.005818569	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x4650, seq=4/1024, ttl=64…
9	4.007317891	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=5/1280, ttl=63…
10	4.007902801	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x4650, seq=5/1280, ttl=64…

Capture during Server 2 pinging at Public Network Interface: (130.1 to 130.2)



25	10.020107721	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x4650, seq=11/2816, ttl=6…
26	10.020396030	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x4650, seq=11/2816, ttl=6…
27	47.025293354	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=1/256, ttl=63 …
28	47.025781849	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x45c3, seq=1/256, ttl=64 …
29	48.026502364	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=2/512, ttl=63 …
30	48.026923735	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x45c3, seq=2/512, ttl=64 …
31	49.028656597	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=3/768, ttl=63 …
32	49.029137809	192.168.130.2	192.168.130.1	ICMP	98	Echo (ping) reply id=0x45c3, seq=3/768, ttl=64 …
33	50.030567859	192.168.130.1	192.168.130.2	ICMP	98	Echo (ping) request id=0x45c3, seq=4/1024, ttl=63…

As we can observe from the wireshark captures that the source IP is being changed as per the rule we gave to 192.168.130.1

[5.3] Now for Client to SSH to Server 1 and Server 2 with knowing only the IP of Router_VM

For this scenario basically, we need to configure iptables at Router VM to direct the ssh request to Server 1 and Server 2.
To differentiate we will assign ssh request with port 101 to Server 1
Ssh request with port 102 to Server 2

So at the Router VM we will configure Prerouting rule stating that if we have a connection request with the port number 101 with the IP of Router_VM, Since the client knows only that then we will establish the ssh connection with Server 1. For the the following rule is entered:

Iptables -t nat -A PREROUTING -s 192.168.130.2 -p tcp --dport 101 -j DNAT --to 192.168.140.2:22
Iptables -t nat -A PREROUTING -s 192.168.130.2 -p tcp --dport 102 -j DNAT --to 192.168.140.3:22

After having these rules Client was able to ssh to Server 1 with the knowledge of Router_VM IP address only.

```
[root@localhost ~]# ssh root@192.168.130.1 -p 101
The authenticity of host '[192.168.130.1]:101 ([192.168.130.1]:101)' can't be established.
ECDSA key fingerprint is SHA256:E9M2INn9ZGZE7w+h58UxMGy6i1PCoZdYY+H5bQ7FeMU.
ECDSA key fingerprint is MD5:3e:63:0e:44:a8:51:9c:8a:2b:41:8d:ce:1f:02:c8:f8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.130.1]:101' (ECDSA) to the list of known hosts.
root@192.168.130.1's password:
Last login: Thu Oct 25 14:57:27 2018
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:a9:0c:76 brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.2/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# _
```

Similarly, Client was able to ssh to Server 2 with a different port number but same IP

```
[[root@localhost ~]#
[[root@localhost ~]# ssh root@192.168.130.1 -p 102
[root@192.168.130.1's password:
 Last login: Thu Oct 25 23:52:08 2018 from 192.168.130.2
[[root@localhost ~]# ip addr
 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
     link/ether 52:54:00:27:a5:30 brd ff:ff:ff:ff:ff:ff
     inet 192.168.140.3/24 scope global eth0
        valid_lft forever preferred_lft forever
 [root@localhost ~]#
```

As you can see the following ip addr command after successful ssh shows the IP addresses of Server 1 and Server 2 respectively.

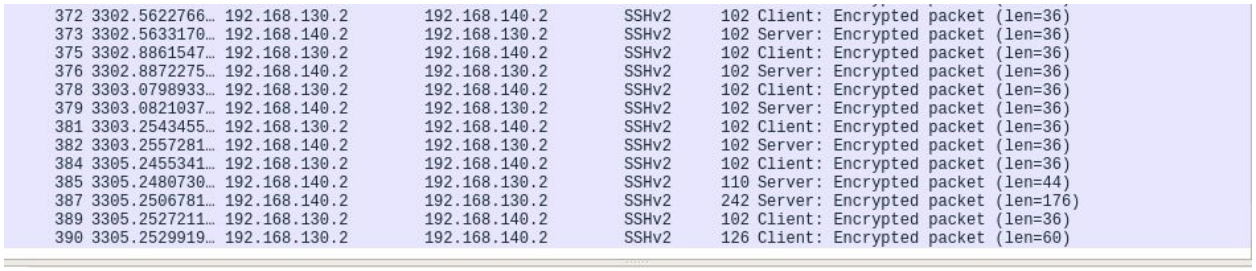
Now we have seen the wireshark captures at the Client Public network interface and as well as the server's private network interface.

While SSH to Server 1 with ssh root@192.168.130.2 -p 101 From Client

At Client:

25	8.384956624	192.168.130.2	192.168.130.1	TCP	150	35132 → 65315 [PSH, ACK] Seq=1686 Ack=1794 Win…
26	8.426321940	192.168.130.1	192.168.130.2	TCP	66	65315 → 35132 [ACK] Seq=1794 Ack=1770 Win=3200…
27	8.465080628	192.168.130.1	192.168.130.2	TCP	94	65315 → 35132 [PSH, ACK] Seq=1794 Ack=1770 Win…
28	8.466029695	192.168.130.2	192.168.130.1	TCP	66	35132 → 65315 [ACK] Seq=1770 Ack=1822 Win=3443…
29	8.466041200	192.168.130.2	192.168.130.1	TCP	178	35132 → 65315 [PSH, ACK] Seq=1770 Ack=1822 Win…
30	8.466753102	192.168.130.1	192.168.130.2	TCP	66	65315 → 35132 [ACK] Seq=1822 Ack=1882 Win=3200…
31	9.154135275	192.168.130.1	192.168.130.2	TCP	566	65315 → 35132 [PSH, ACK] Seq=1822 Ack=1882 Win…
32	9.193760861	192.168.130.2	192.168.130.1	TCP	66	35132 → 65315 [ACK] Seq=1882 Ack=2322 Win=3699…
33	9.195016441	192.168.130.1	192.168.130.2	TCP	110	65315 → 35132 [PSH, ACK] Seq=2322 Ack=1882 Win…
34	9.195302858	192.168.130.2	192.168.130.1	TCP	66	35132 → 65315 [ACK] Seq=1882 Ack=2366 Win=3699…
35	9.195939596	192.168.130.2	192.168.130.1	TCP	526	35132 → 65315 [PSH, ACK] Seq=1882 Ack=2366 Win…
36	9.197166758	192.168.130.1	192.168.130.2	TCP	66	65315 → 35132 [ACK] Seq=2366 Ack=2342 Win=3494…

At Server Side Private Interface:

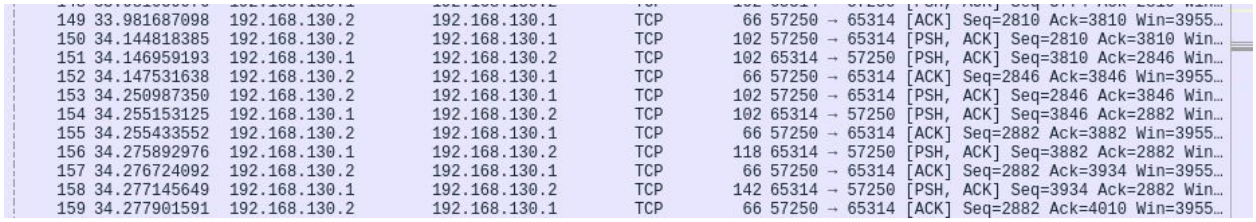


372	3302.5622766…	192.168.130.2	192.168.140.2	SSHv2	102 Client: Encrypted packet (len=36)
373	3302.5633170…	192.168.140.2	192.168.130.2	SSHv2	102 Server: Encrypted packet (len=36)
375	3302.8861547…	192.168.130.2	192.168.140.2	SSHv2	102 Client: Encrypted packet (len=36)
376	3302.8872275…	192.168.140.2	192.168.130.2	SSHv2	102 Server: Encrypted packet (len=36)
378	3303.0798933…	192.168.130.2	192.168.140.2	SSHv2	102 Client: Encrypted packet (len=36)
379	3303.0821037…	192.168.140.2	192.168.130.2	SSHv2	102 Server: Encrypted packet (len=36)
381	3303.2543455…	192.168.130.2	192.168.140.2	SSHv2	102 Client: Encrypted packet (len=36)
382	3303.2557281…	192.168.140.2	192.168.130.2	SSHv2	102 Server: Encrypted packet (len=36)
384	3305.2455341…	192.168.130.2	192.168.140.2	SSHv2	102 Client: Encrypted packet (len=36)
385	3305.2480730…	192.168.140.2	192.168.130.2	SSHv2	110 Server: Encrypted packet (len=44)
387	3305.2506781…	192.168.140.2	192.168.130.2	SSHv2	242 Server: Encrypted packet (len=176)
389	3305.2527211…	192.168.130.2	192.168.140.2	SSHv2	102 Client: Encrypted packet (len=36)
390	3305.2529919…	192.168.130.2	192.168.140.2	SSHv2	126 Client: Encrypted packet (len=60)

As you can see the SSH with destination IP to be that of Server 1's IP address could be seen here.

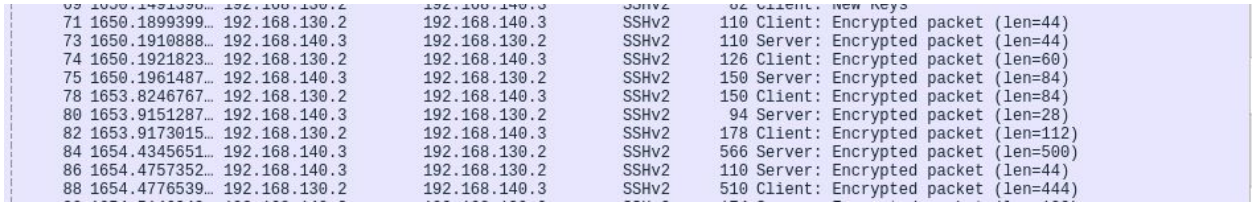
While SSH to Server 2 with ssh root@192.168.130.2 -p 102 From Client

At Client:



149	33.981687098	192.168.130.2	192.168.130.1	TCP	66 57250 → 65314 [ACK] Seq=2810 Ack=3810 Win=3955…
150	34.144818385	192.168.130.2	192.168.130.1	TCP	102 57250 → 65314 [PSH, ACK] Seq=2810 Ack=3810 Win…
151	34.146959193	192.168.130.1	192.168.130.2	TCP	102 65314 → 57250 [PSH, ACK] Seq=3810 Ack=2846 Win…
152	34.147531638	192.168.130.2	192.168.130.1	TCP	66 57250 → 65314 [ACK] Seq=2846 Ack=3846 Win=3955…
153	34.250987350	192.168.130.2	192.168.130.1	TCP	102 57250 → 65314 [PSH, ACK] Seq=2846 Ack=3846 Win…
154	34.255153125	192.168.130.1	192.168.130.2	TCP	102 65314 → 57250 [PSH, ACK] Seq=3846 Ack=2882 Win…
155	34.255433552	192.168.130.2	192.168.130.1	TCP	66 57250 → 65314 [ACK] Seq=2882 Ack=3882 Win=3955…
156	34.275892976	192.168.130.1	192.168.130.2	TCP	118 65314 → 57250 [PSH, ACK] Seq=3882 Ack=2882 Win…
157	34.276724092	192.168.130.2	192.168.130.1	TCP	66 57250 → 65314 [ACK] Seq=2882 Ack=3934 Win=3955…
158	34.277145649	192.168.130.1	192.168.130.2	TCP	142 65314 → 57250 [PSH, ACK] Seq=3934 Ack=2882 Win…
159	34.277901591	192.168.130.2	192.168.130.1	TCP	66 57250 → 65314 [ACK] Seq=2882 Ack=4010 Win=3955…

At Server Private Interface:



71	1650.1899399…	192.168.130.2	192.168.140.3	SSHv2	110 Client: Encrypted packet (len=44)
73	1650.1910888…	192.168.140.3	192.168.130.2	SSHv2	110 Server: Encrypted packet (len=44)
74	1650.1921823…	192.168.130.2	192.168.140.3	SSHv2	126 Client: Encrypted packet (len=60)
75	1650.1961487…	192.168.140.3	192.168.130.2	SSHv2	150 Server: Encrypted packet (len=84)
78	1653.8246767…	192.168.130.2	192.168.140.3	SSHv2	150 Client: Encrypted packet (len=84)
80	1653.9151287…	192.168.140.3	192.168.130.2	SSHv2	94 Server: Encrypted packet (len=28)
82	1653.9173015…	192.168.130.2	192.168.140.3	SSHv2	178 Client: Encrypted packet (len=112)
84	1654.4345651…	192.168.140.3	192.168.130.2	SSHv2	566 Server: Encrypted packet (len=500)
86	1654.4757352…	192.168.140.3	192.168.130.2	SSHv2	110 Server: Encrypted packet (len=44)
88	1654.4776539…	192.168.130.2	192.168.140.3	SSHv2	510 Client: Encrypted packet (len=444)

Here you can see the SSH is for the Server 2 with the destination address to be seen as 192.168.140.3.

[5.4] Now for Load Balancing the connections from Client to Servers.

In 5.3 we have specified destination ports and differentiated between the Servers to ssh, therefore, having 2 rules accommodated that. Now in case of load balancing having the same configuration as above wouldn't make send, instead we will not differentiate with port numbers instead let the load balancer decide on which Server to get connected with. Therefore we will flush the NAT IP Tables and again configure but this time without the destination port specification. With the NAT-based load balancing configured the if the Client tries to ssh with the

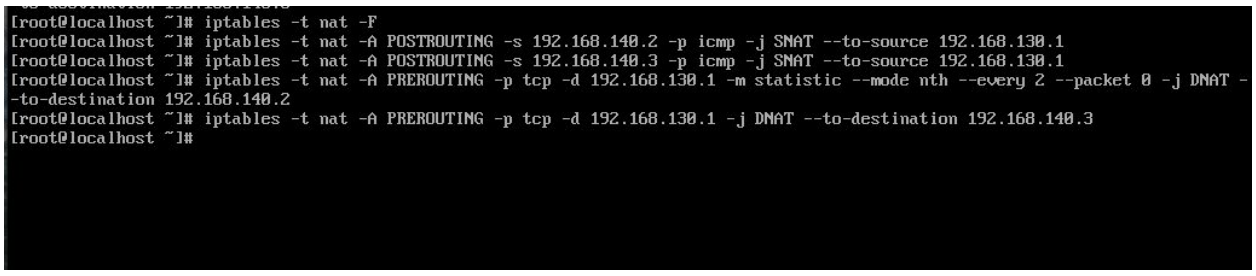
Router's VM it will always ssh into the same Server whichever comes 1st in the Chain.
So now we will configure the NAT-based Load Balancing.

[A] Define load precisely:
We will choose to do the load balancing in a Round-Robin Fashion where the rule will be evaluated every n packets starting at the packet with packet p.
For instance: For every 2 packets starting with packet 0 we will have Server 1
By default, it goes to Server 2 so we just mention the general rule

[B] Configuring the Balancing Knob at Router VM

Iptables -t nat -A PREROUTING -p tcp -d 192.168.130.1 -m statistic --mode nth --every 2 --packet 0 -j DNAT -to-destination 192.168.140.2

Iptables -t nat -A PREROUTING -p tcp -d 192.168.130.1 -j DNAT -to-destination 192.168.140.3




```
[root@localhost ~]# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       tcp  --  anywhere             localhost.localdomain  statistic mode nth every 2 to:192.168.140.2
DNAT       tcp  --  anywhere             localhost.localdomain  to:192.168.140.3

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       icmp --  192.168.140.2        anywhere             to:192.168.130.1
SNAT       icmp --  192.168.140.3        anywhere             to:192.168.130.1
```

[C] Verification:
For verification based on our logic, every other traffic should be alternatively switched between both the servers that means 1st time it should connect to Server 2 and next time to Server 1 and again Server 2 and so on.
Below Screenshot clearly shows that as well.

```
[root@localhost ~]# ssh root@192.168.130.1
The authenticity of host '192.168.130.1 (192.168.130.1)' can't be established.
ECDSA key fingerprint is SHA256:OcYKLe1tJxA/ZolfFnFWgU4oReET7ZvAVgx7Hp+uMd4.
ECDSA key fingerprint is MD5:7d:e6:f1:58:91:5c:50:ba:32:9e:39:48:19:22:3a:28.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.130.1' (ECDSA) to the list of known hosts.
root@192.168.130.1's password:
Permission denied, please try again.
root@192.168.130.1's password:
Last failed login: Fri Oct 26 01:37:23 EDT 2018 from 192.168.130.2 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Fri Oct 26 01:33:01 2018 from 192.168.130.2
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:27:a5:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.3/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# exit
logout
Connection to 192.168.130.1 closed.
[root@localhost ~]# ssh-keygen -R 192.168.130.1
# Host 192.168.130.1 found: line 5
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
[root@localhost ~]# ssh root@192.168.130.1
The authenticity of host '192.168.130.1 (192.168.130.1)' can't be established.
ECDSA key fingerprint is SHA256:E9M2INn9ZGZE7w+h58UxMGy6i1PCoZdYY+H5bQ7FeMU.
ECDSA key fingerprint is MD5:3e:63:0e:44:a8:51:9c:8a:2b:41:8d:ce:1f:02:c8:f8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.130.1' (ECDSA) to the list of known hosts.
root@192.168.130.1's password:
Last login: Fri Oct 26 01:33:21 2018 from 192.168.130.2
[root@localhost ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:a9:0c:76 brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.2/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# exit
logout
Connection to 192.168.130.1 closed.
[root@localhost ~]# ssh-keygen -R 192.168.130.1
# Host 192.168.130.1 found: line 5
/root/.ssh/known_hosts updated.
Original contents retained as /root/.ssh/known_hosts.old
[root@localhost ~]# ssh root@192.168.130.1
The authenticity of host '192.168.130.1 (192.168.130.1)' can't be established.
ECDSA key fingerprint is SHA256:OcYKLe1tJxA/ZolfFnFWgU4oReET7ZvAVgx7Hp+uMd4.
ECDSA key fingerprint is MD5:7d:e6:f1:58:91:5c:50:ba:32:9e:39:48:19:22:3a:28.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.130.1' (ECDSA) to the list of known hosts.
root@192.168.130.1's password:
Last login: Fri Oct 26 01:37:36 2018 from 192.168.130.2
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:27:a5:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.3/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]#
```

[6]Namespaces:

[6.1] L2 Isolation:

The following picture gives you the whole idea of the setup with veth pair names and IP addresses at all the interfaces.

After the setup, we have configured default routes at both the VM's to reach NS1.
Now for L2 Isolation even with same MAC addresses at both the VM's it should have a successful ping from T1-VM1 to T1-VM2 and vice versa.
Since both the VM's are in different subnets, even if we have the same MAC address for both VM's it wouldn't be a problem and we would see a successful ping as below:

Ping from T1-VM1(192.168.130.3) to T1-VM2(192.168.140.3)

```
[root@localhost ~]# ping 192.168.140.3
PING 192.168.140.3 (192.168.140.3) 56(84) bytes of data.
64 bytes from 192.168.140.3: icmp_seq=1 ttl=63 time=6.81 ms
64 bytes from 192.168.140.3: icmp_seq=2 ttl=63 time=0.646 ms
64 bytes from 192.168.140.3: icmp_seq=3 ttl=63 time=1.87 ms
64 bytes from 192.168.140.3: icmp_seq=4 ttl=63 time=0.901 ms
64 bytes from 192.168.140.3: icmp_seq=5 ttl=63 time=2.81 ms
64 bytes from 192.168.140.3: icmp_seq=6 ttl=63 time=1.94 ms
64 bytes from 192.168.140.3: icmp_seq=7 ttl=63 time=0.712 ms
64 bytes from 192.168.140.3: icmp_seq=8 ttl=63 time=1.05 ms
64 bytes from 192.168.140.3: icmp_seq=9 ttl=63 time=1.15 ms
64 bytes from 192.168.140.3: icmp_seq=10 ttl=63 time=1.82 ms
64 bytes from 192.168.140.3: icmp_seq=11 ttl=63 time=0.799 ms
64 bytes from 192.168.140.3: icmp_seq=12 ttl=63 time=0.895 ms
^C
--- 192.168.140.3 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11017ms
rtt min/avg/max/mdev = 0.646/1.786/6.817/1.642 ms
```

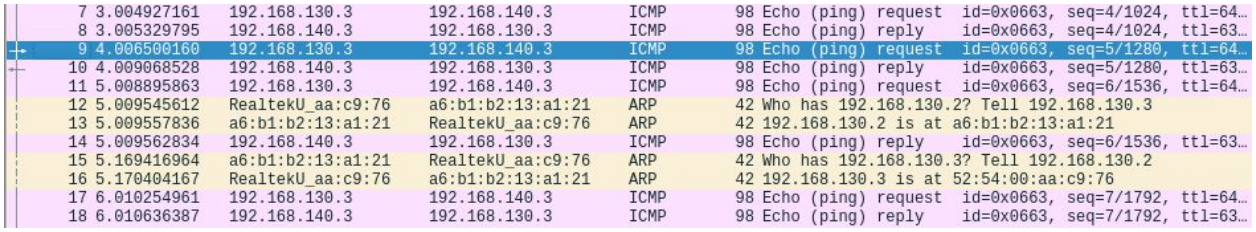
So as they are in different networks we attain isolation and we have also seen the arp resolution at the Namespace as below:

At NS1:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# ip route
 192.168.120.0/24 dev v12  proto kernel  scope link  src 192.168.120.3
 192.168.130.0/24 dev v21  proto kernel  scope link  src 192.168.130.2
 192.168.140.0/24 dev v31  proto kernel  scope link  src 192.168.140.2
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# arp -a
 ? (192.168.120.2) at 3e:8f:c1:8a:59:2a [ether] on v12
 ? (192.168.140.3) at 52:54:00:aa:c9:76 [ether] on v31
 ? (192.168.130.3) at 52:54:00:aa:c9:76 [ether] on v21
 root@ece792-Standard-PC-i440FX-PIIX-1996:~#
```

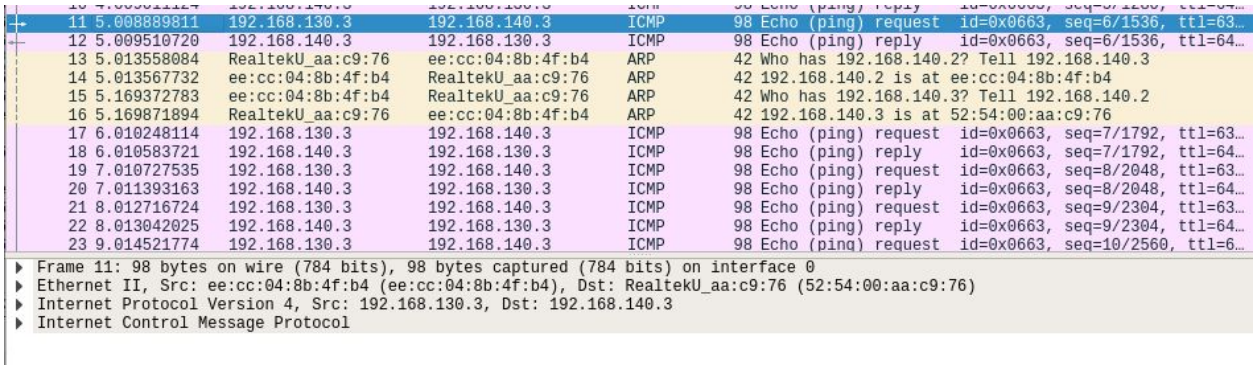
As we can see both the VM's have same MAC address but by being in different subnetworks at the arp table we still have the differentiation based on the interfaces. Therefore we have L2 Isolation. Further below screenshots shows us the wireshark captures at the interfaces of the subnets at the Namespace.

At the interface towards T1-VM1(V22):



7	3.004927161	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=4/1024, ttl=64…
8	3.005329795	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=4/1024, ttl=63…
9	4.006500160	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=5/1280, ttl=64…
10	4.009068528	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=5/1280, ttl=63…
11	5.008895863	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=6/1536, ttl=64…
12	5.009545612	RealtekU_aa:c9:76	a6:b1:b2:13:a1:21	ARP	42	Who has 192.168.130.2? Tell 192.168.130.3
13	5.009557836	a6:b1:b2:13:a1:21	RealtekU_aa:c9:76	ARP	42	192.168.130.2 is at a6:b1:b2:13:a1:21
14	5.009562834	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=6/1536, ttl=63…
15	5.169416964	a6:b1:b2:13:a1:21	RealtekU_aa:c9:76	ARP	42	Who has 192.168.130.3? Tell 192.168.130.2
16	5.170404167	RealtekU_aa:c9:76	a6:b1:b2:13:a1:21	ARP	42	192.168.130.3 is at 52:54:00:aa:c9:76
17	6.010254961	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=7/1792, ttl=64…
18	6.010636387	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=7/1792, ttl=63…

At the interface towards T1-VM2(V32):



11	5.008889811	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=6/1536, ttl=63…
12	5.009510720	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=6/1536, ttl=64…
13	5.013558084	RealtekU_aa:c9:76	ee:cc:04:8b:4f:b4	ARP	42	Who has 192.168.140.2? Tell 192.168.140.3
14	5.013567732	ee:cc:04:8b:4f:b4	RealtekU_aa:c9:76	ARP	42	192.168.140.2 is at ee:cc:04:8b:4f:b4
15	5.169372783	ee:cc:04:8b:4f:b4	RealtekU_aa:c9:76	ARP	42	Who has 192.168.140.3? Tell 192.168.140.2
16	5.169871894	RealtekU_aa:c9:76	ee:cc:04:8b:4f:b4	ARP	42	192.168.140.3 is at 52:54:00:aa:c9:76
17	6.010248114	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=7/1792, ttl=63…
18	6.010583721	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=7/1792, ttl=64…
19	7.010727535	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=8/2048, ttl=63…
20	7.011393163	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=8/2048, ttl=64…
21	8.012716724	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=9/2304, ttl=63…
22	8.013042025	192.168.140.3	192.168.130.3	ICMP	98	Echo (ping) reply id=0x0663, seq=9/2304, ttl=64…
23	9.014521774	192.168.130.3	192.168.140.3	ICMP	98	Echo (ping) request id=0x0663, seq=10/2560, ttl=6…

```
▸ Frame 11: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸ Ethernet II, Src: ee:cc:04:8b:4f:b4 (ee:cc:04:8b:4f:b4), Dst: RealtekU_aa:c9:76 (52:54:00:aa:c9:76)
▸ Internet Protocol Version 4, Src: 192.168.130.3, Dst: 192.168.140.3
▸ Internet Control Message Protocol
```

[6.2] L3 Isolation:
VMs of tenant 1 should not be able to ping to tenant 2 VMs whether they have same or different IP subnets.

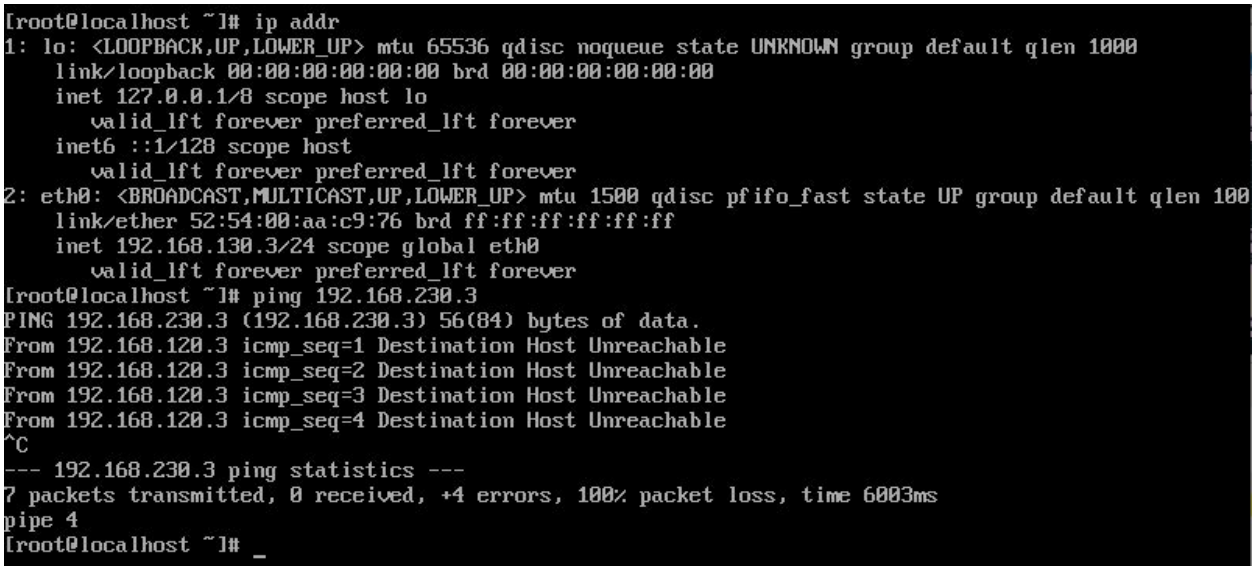
We have configured the default gateway for NS1 to be v12 (192.168.120.3) and enforced NAT rule by configuring NAT tables at NS1 with the command:
iptables -t nat -A POSTROUTING -o v12 -j MASQUERADE (NS1)

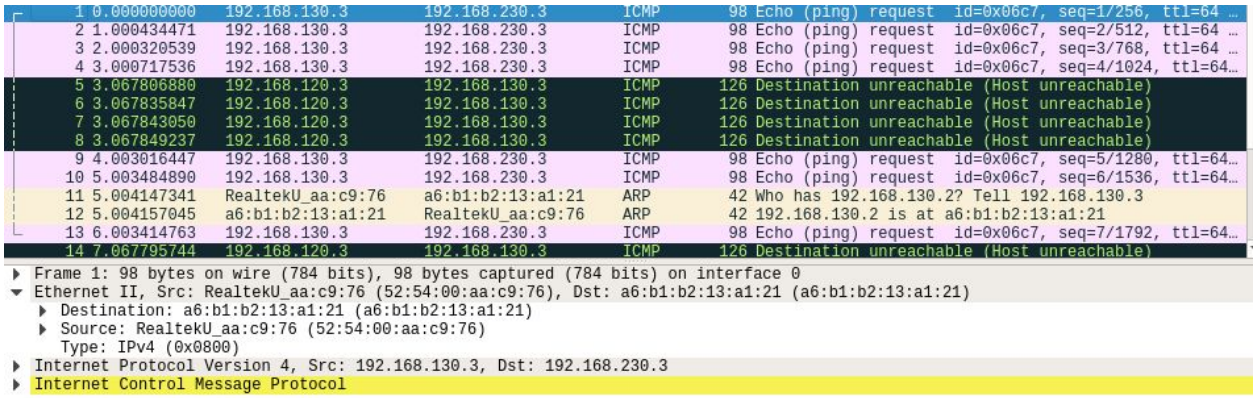
Similarly at NS2:
iptables -t nat -A POSTROUTING -o v912 -j MASQUERADE (NS2)

So now a ping to the Host or interface v11 from T1-VM1 or T1-VM2 is successful as we have configured NAT.

Now when we try to ping from T1-VM1(130.3) to T2-VM1(230.3) which are in different subnets the ping request fails as shown in below screenshot along with wireshark captures.

In case of different subnets- Ping from T1-VM1 to T2-VM1:





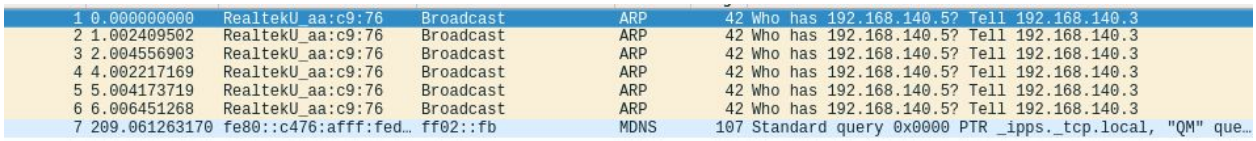
1	0.000000000	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=1/256, ttl=64 …
2	1.000434471	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=2/512, ttl=64 …
3	2.000320539	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=3/768, ttl=64 …
4	3.000717536	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=4/1024, ttl=64…
5	3.067806880	192.168.120.3	192.168.130.3	ICMP	126 Destination unreachable (Host unreachable)
6	3.067835847	192.168.120.3	192.168.130.3	ICMP	126 Destination unreachable (Host unreachable)
7	3.067843050	192.168.120.3	192.168.130.3	ICMP	126 Destination unreachable (Host unreachable)
8	3.067849237	192.168.120.3	192.168.130.3	ICMP	126 Destination unreachable (Host unreachable)
9	4.003016447	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=5/1280, ttl=64…
10	5.003484890	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=6/1536, ttl=64…
11	5.004147341	RealtekU_aa:c9:76	a6:b1:b2:13:a1:21	ARP	42 Who has 192.168.130.2? Tell 192.168.130.3
12	5.004157045	a6:b1:b2:13:a1:21	RealtekU_aa:c9:76	ARP	42 192.168.130.2 is at a6:b1:b2:13:a1:21
13	6.003414763	192.168.130.3	192.168.230.3	ICMP	98 Echo (ping) request id=0x06c7, seq=7/1792, ttl=64…
14	7.067795744	192.168.120.3	192.168.130.3	ICMP	126 Destination unreachable (Host unreachable)

```
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: RealtekU_aa:c9:76 (52:54:00:aa:c9:76), Dst: a6:b1:b2:13:a1:21 (a6:b1:b2:13:a1:21)
  ▸ Destination: a6:b1:b2:13:a1:21 (a6:b1:b2:13:a1:21)
  ▸ Source: RealtekU_aa:c9:76 (52:54:00:aa:c9:76)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.130.3, Dst: 192.168.230.3
▸ Internet Control Message Protocol
```

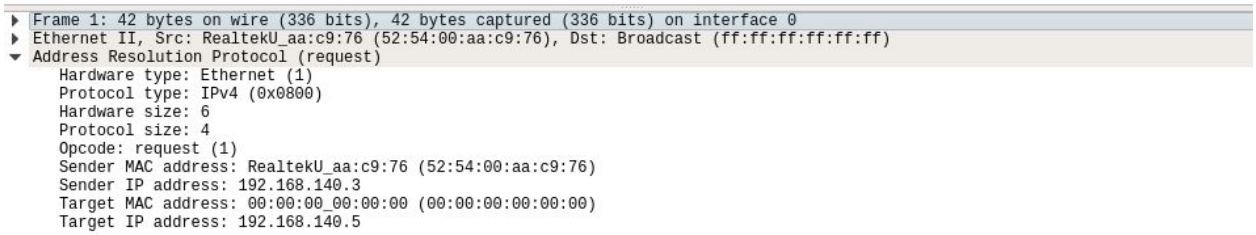
This fails as the hypervisor has no knowledge of the internal subnets of a different namespace which tells us there is isolation.

In case of same subnets- Ping from T1-VM2(140.3) to T2-VM2(140.5):

```
[root@localhost ~]# ping 192.168.140.5
PING 192.168.140.5 (192.168.140.5) 56(84) bytes of data.
From 192.168.140.3 icmp_seq=1 Destination Host Unreachable
From 192.168.140.3 icmp_seq=2 Destination Host Unreachable
From 192.168.140.3 icmp_seq=3 Destination Host Unreachable
From 192.168.140.3 icmp_seq=4 Destination Host Unreachable
From 192.168.140.3 icmp_seq=5 Destination Host Unreachable
From 192.168.140.3 icmp_seq=6 Destination Host Unreachable
From 192.168.140.3 icmp_seq=7 Destination Host Unreachable
From 192.168.140.3 icmp_seq=8 Destination Host Unreachable
^C
--- 192.168.140.5 ping statistics ---
8 packets transmitted, 0 received, +8 errors, 100% packet loss, time 7001ms
pipe 4
```



1	0.000000000	RealtekU_aa:c9:76	Broadcast	ARP	42 Who has 192.168.140.5? Tell 192.168.140.3
2	1.002409502	RealtekU_aa:c9:76	Broadcast	ARP	42 Who has 192.168.140.5? Tell 192.168.140.3
3	2.004556903	RealtekU_aa:c9:76	Broadcast	ARP	42 Who has 192.168.140.5? Tell 192.168.140.3
4	4.002217169	RealtekU_aa:c9:76	Broadcast	ARP	42 Who has 192.168.140.5? Tell 192.168.140.3
5	5.004173719	RealtekU_aa:c9:76	Broadcast	ARP	42 Who has 192.168.140.5? Tell 192.168.140.3
6	6.006451268	RealtekU_aa:c9:76	Broadcast	ARP	42 Who has 192.168.140.5? Tell 192.168.140.3
7	209.061263170	fe80::c476:afff:fed…	ff02::fb	MDNS	107 Standard query 0x0000 PTR _ipps._tcp.local, "QM" que…



```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: RealtekU_aa:c9:76 (52:54:00:aa:c9:76), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: RealtekU_aa:c9:76 (52:54:00:aa:c9:76)
    Sender IP address: 192.168.140.3
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.140.5
```

In case of same subnets in our tenant 1 itself the wireshark capture at the v32 which is the VM2 subnet interface connecting to NS2 we find only the Arp packets since the pinging IP is in the same network the arp requests are broadcasted while there is no reply.

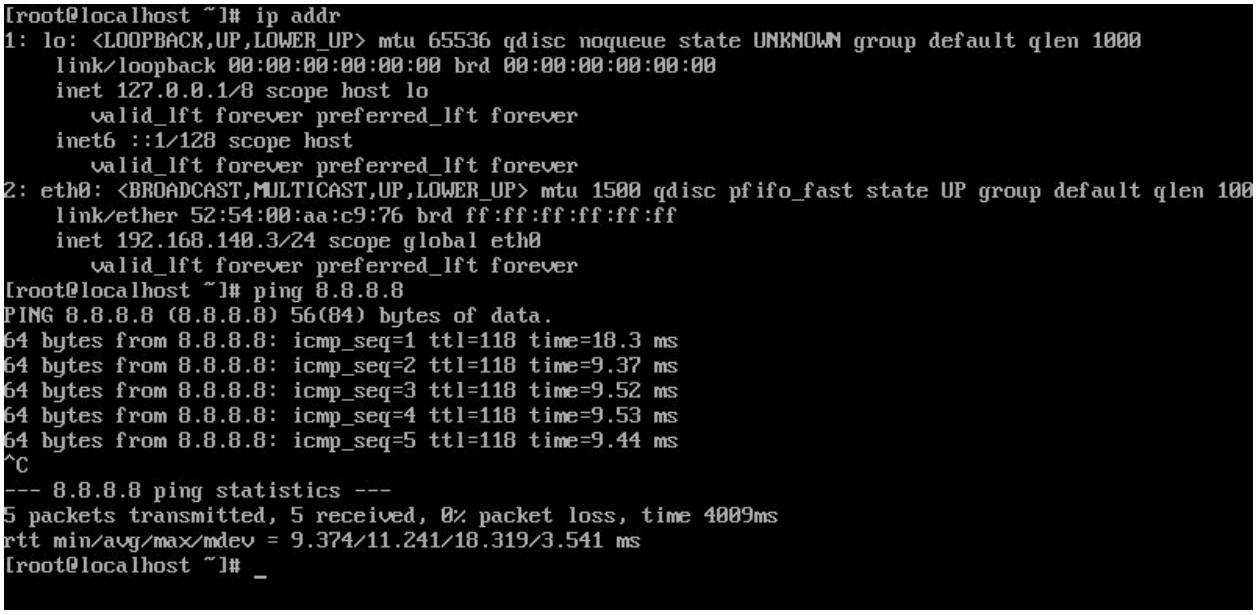
Therefore even with the same or different subnets, we have L3 Isolation between the tenants.

In another experiment, both tenants use one subnet that is common (e.g., 10.0.0.0/8) and one that is different. The hosts in the common subnet for tenant red and tenant blue should be able to ping the internet.

So in our setup, we have subnets of T1-VM2 and T2-VM2 with same subnet (192.168.140.0/24) Now these 2 VM's should be able to reach the internet. While so far we have the NAT rules inside the namespace, but we do not have the NAT rules inside the hypervisor which would be the endpoint while going out to reach the internet. So we would be adding NAT rules inside the hypervisor. Also inside the namespace, the default gateway is set to v11 interface which is the veth pair end inside the hypervisor.
Command at the Hypervisor for the NAT rule:
sudo iptables -t nat -A POSTROUTING -o ens3 -j MASQUERADE

Following screenshots below show us that the ping is successful:

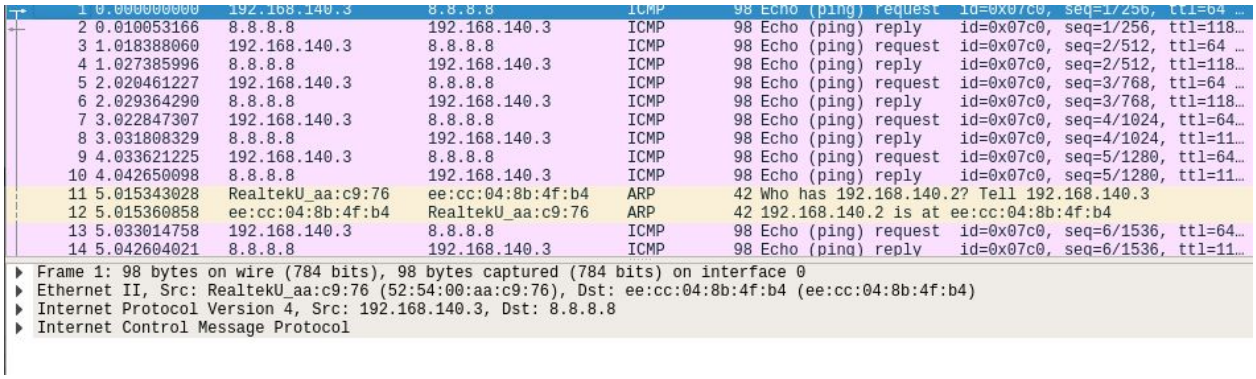
From T1-VM2(192.168.140.3):



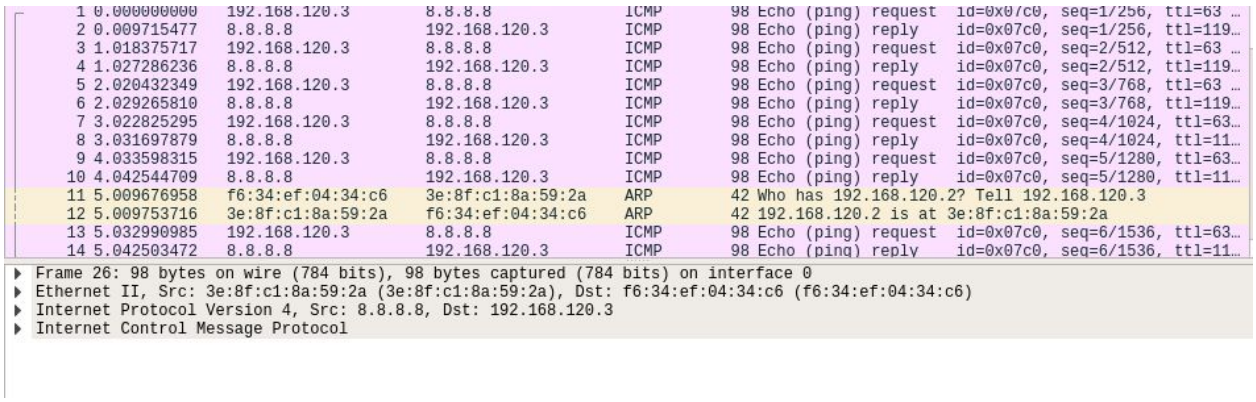
From T2-VM2: (192.168.140.5) :

```
[root@localhost ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00                                    en 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
    link/ether 52:54:00:dd:c9:76 brd ff:ff:ff:ff:ff:ff                                       default qlen 100
    inet 192.168.140.5/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=21.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=11.5 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 11.287/14.821/21.579/4.781 ms
[root@localhost ~]# _
```

Wireshark Captures at V32 veth pair interface connection T1-VM2 subnet to Namespace(NS1):



Wireshark Captures at V11 veth pair interface inside Hypervisor(Tenant1-Blue):



As we can see before entering Namespace the Source IP is 192.168.140.3 and while leaving Namespace due to our NAT rule it is changed to 192.168.120.3. Then at the Hypervisor, it is again changed to the IP addresses of the interface it is leaving through.

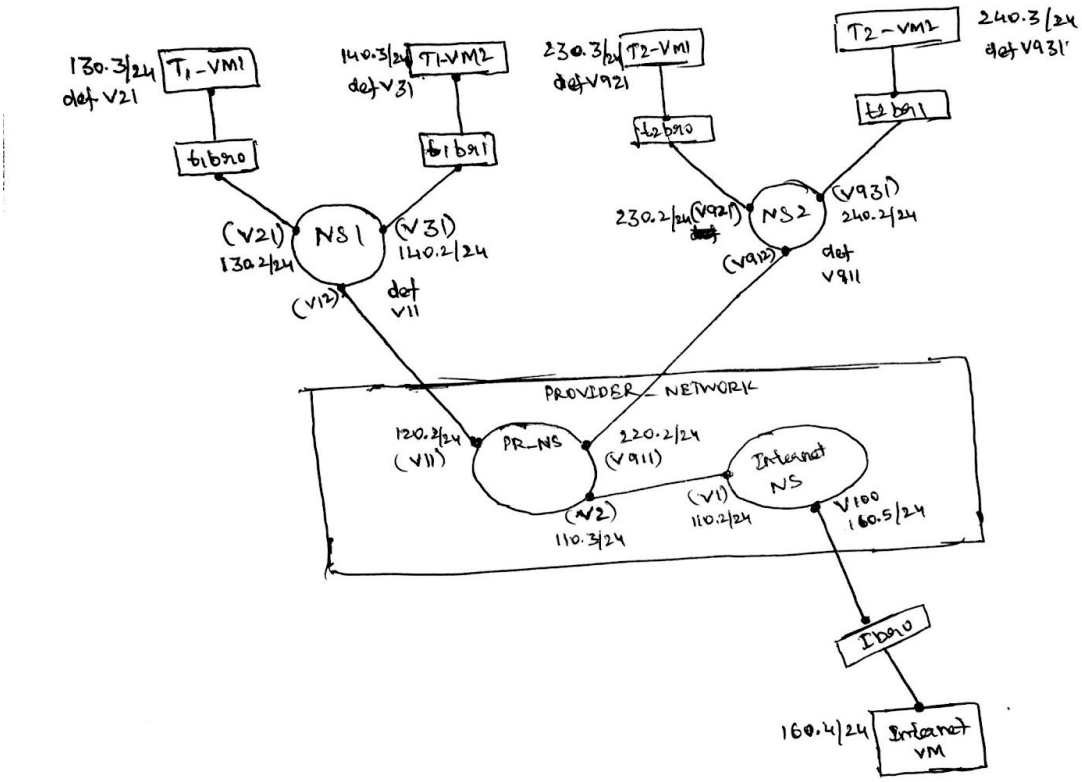
This again shows us that even if we have 2 same subnets for 2 different tenants inside our network pings to internet work due to the namespaces which provide us with the L3 Isolation due to which we are able to see the successful pings.

[6.3] Forwarding table and IP tables in Host hypervisor. In this experiment, make sure not to use the hypervisor host's default forwarding table/IP tables. Configure a new network namespaces (call it provider ns) in the hypervisor. Implement and verify the following policies in the provider network namespace.

[6.3.A] Internet policy. Allow ICMP traffic for both tenants. Allow SSH traffic for only the blue tenant.
[6.3.B] Local L3 policy. Allow red tenant and blue tenant to ssh each other's VM, provided the subnets are different.

Now with the provider network we create a namespace and implement the following policies. Please see the below picture for clear idea of the architecture and design with the IP addresses and interfaces as well.
So along with provider namespace we have also created a internet namespace which is again connected to a VM which for our testing will be nothing but internet. So any ping to the Internet_VM is nothing but pinging the internet.
Internet VM IP: 192.168.160.4

All the IP addresses in the below picture is of the format 192.168.XXX.X
And the def (interface) means from there the default gateway route is set to that particular interface.



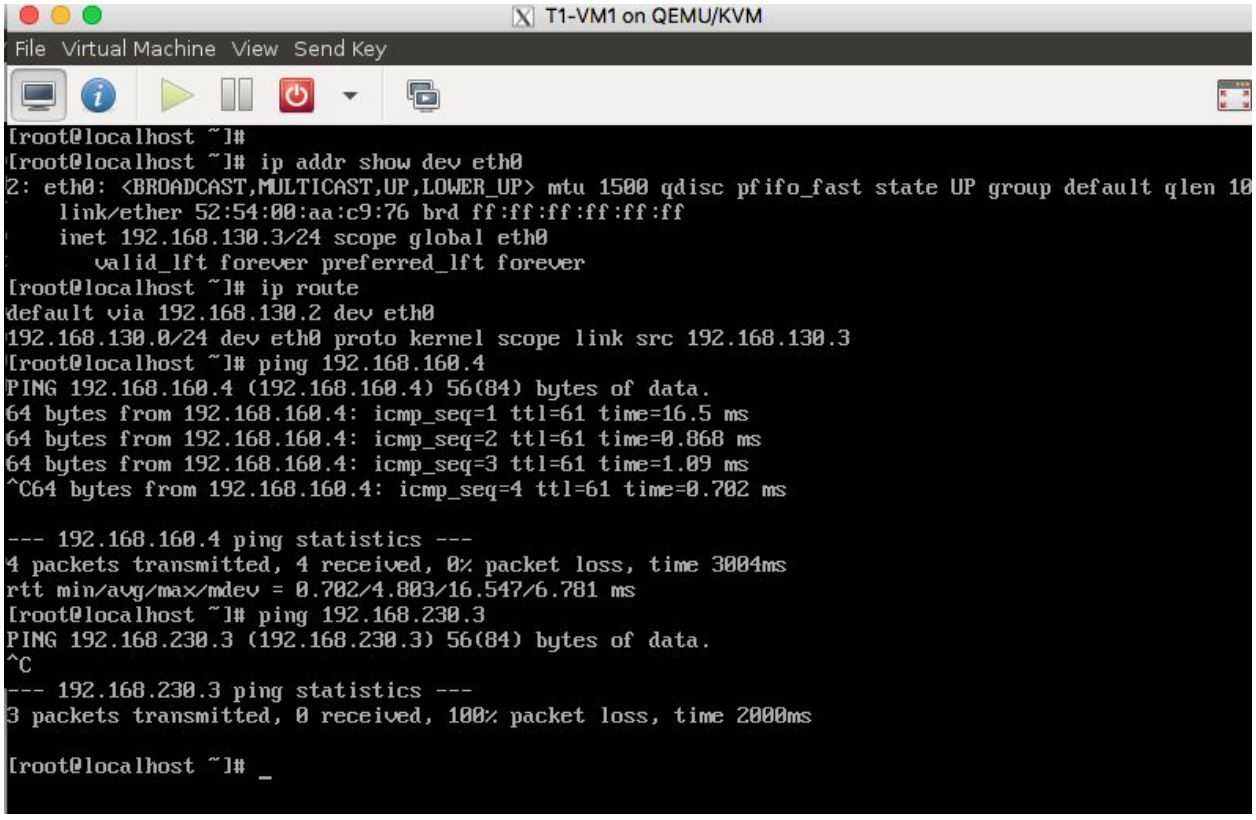
Now for initially to allow ICMP traffic for both tenants which would mean both the tenants should be able to ping the internet. First thing would be to perform NAT operation before the tenant leaves from it's Namespace.
So now NAT rules for icmp protocol at both the namespaces so that we should be getting the replies back and we have to the same thing at the provider namespace and also inside internet_namespace as the last end point from the provider network which will be from the internet namespace which is v100.

At NS1,NS2,provider_ns,internet_ns

iptables -t nat -A POSTROUTING -p icmp -o v12 -j MASQUERADE
iptables -t nat -A POSTROUTING -p icmp -o v912 -j MASQUERADE
iptables -t nat -A POSTROUTING -p icmp -o v2 -j MASQUERADE
iptables -t nat -A POSTROUTING -p icmp -o v100 -j MASQUERADE

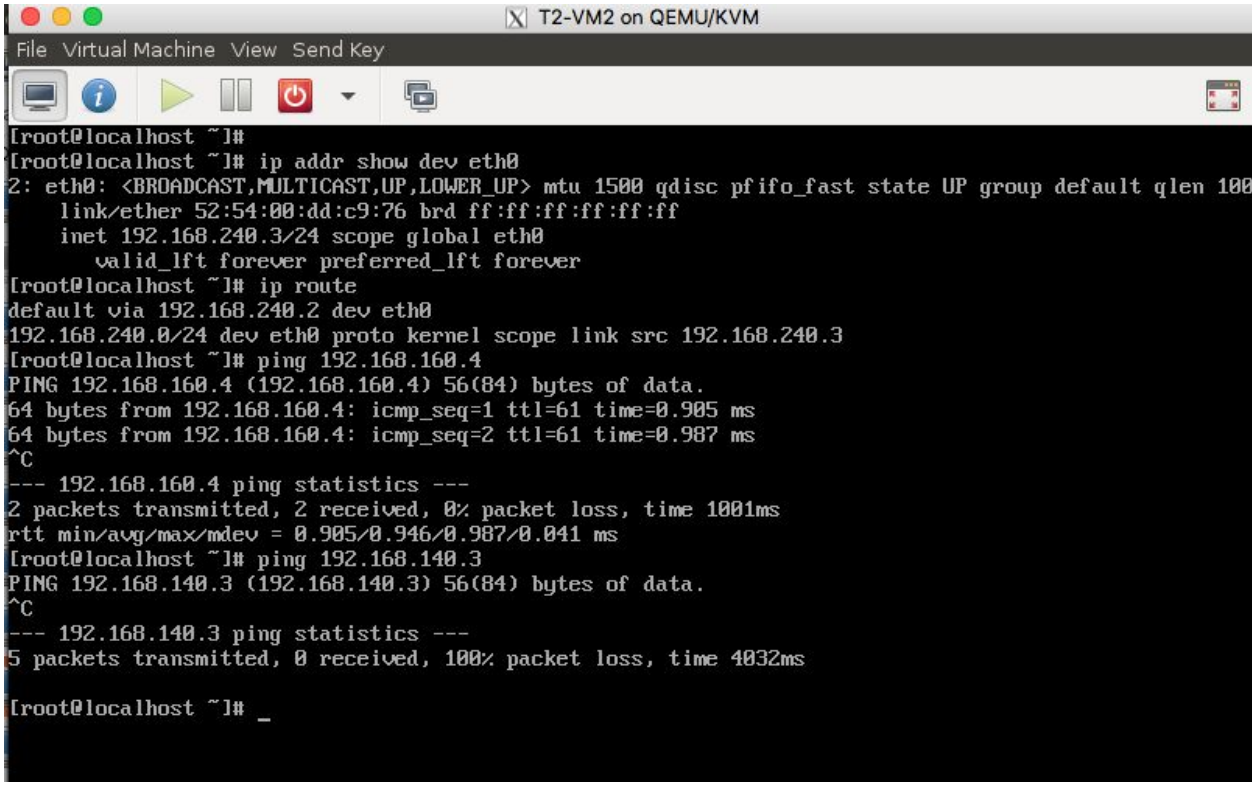
With these rules we can see a ping request from the tenants to internet is successful and tenants cannot ping each other as well maintaining the isolation needed.

Successful ping to internet but not to other tenant from T1-VM1:



You can also see the **TTL value to be 61** which indicates it has the 3 hops which would be NS1, provider_ns,internet_ns. You can also see the routes and IP addresses in these screenshots.

Similarly as above screenshot but from T2-VM2:



Packet Capture at NS1(v21):

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~#
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v21
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v21, link-type EN10MB (Ethernet), capture size 262144 bytes
^C22:26:14.104898 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4158, seq 1, length 64
22:26:14.109158 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4158, seq 1, length 64
22:26:15.105665 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4158, seq 2, length 64
22:26:15.107027 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4158, seq 2, length 64
22:26:16.107876 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4158, seq 3, length 64
22:26:16.108233 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4158, seq 3, length 64
22:26:17.108478 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4158, seq 4, length 64
22:27:01.429953 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4159, seq 1, length 64
22:27:01.431916 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4159, seq 1, length 64
22:27:02.434335 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4159, seq 2, length 64
22:27:02.434709 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4159, seq 2, length 64
22:27:03.437980 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4159, seq 3, length 64
22:27:03.438363 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4159, seq 3, length 64
22:27:04.440094 IP 192.168.130.3 > 192.168.160.4: ICMP echo request, id 4159, seq 4, length 64
22:27:04.440576 IP 192.168.160.4 > 192.168.130.3: ICMP echo reply, id 4159, seq 4, length 64
```

Packet Capture at provider_ns(v11):

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ sudo ip netns exec provider_ns bash
[sudo] password for ece792:
root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v11
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v11, link-type EN10MB (Ethernet), capture size 262144 bytes
^C22:30:05.072941 IP 192.168.120.3 > 192.168.160.4: ICMP echo request, id 4161, seq 1, length 64
22:30:05.073908 IP 192.168.160.4 > 192.168.120.3: ICMP echo reply, id 4161, seq 1, length 64
22:30:06.070522 IP 192.168.120.3 > 192.168.160.4: ICMP echo request, id 4161, seq 2, length 64
22:30:06.071434 IP 192.168.160.4 > 192.168.120.3: ICMP echo reply, id 4161, seq 2, length 64
22:30:07.072797 IP 192.168.120.3 > 192.168.160.4: ICMP echo request, id 4161, seq 3, length 64
22:30:07.073738 IP 192.168.160.4 > 192.168.120.3: ICMP echo reply, id 4161, seq 3, length 64
22:30:08.073647 IP 192.168.120.3 > 192.168.160.4: ICMP echo request, id 4161, seq 4, length 64
```

Packet Capture at internet_ns(v1):

```
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v1 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C22:33:09.848394 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 1, length 64
22:33:09.849920 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 1, length 64
22:33:10.850409 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 2, length 64
22:33:10.851445 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 2, length 64
22:33:11.852496 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 3, length 64
22:33:11.852968 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 3, length 64
22:33:12.853992 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 4, length 64
22:33:49.918120 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 41, length 64
22:33:49.918781 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 41, length 64
22:33:50.920625 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 42, length 64
22:33:50.921080 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 42, length 64
22:33:51.922090 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 43, length 64
22:33:51.922495 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 43, length 64
22:33:52.922437 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 44, length 64
22:33:52.922896 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 44, length 64
22:33:53.924464 IP 192.168.110.3 > 192.168.160.4: ICMP echo request, id 4168, seq 45, length 64
22:33:53.925040 IP 192.168.160.4 > 192.168.110.3: ICMP echo reply, id 4168, seq 45, length 64
```

You can observer the source ip's being changed at each point due to our rules implemented.

Now combining the 6.3B and 6.3A parts for SSH we would explain below:
So right now we have added the NAT rules with respect to icmp traffic.
Now we also need to add NAT rules for SSH traffic as well at all the Namespaces.

At NS1,NS2,provider_ns,internet_ns
iptables -t nat -A POSTROUTING -o v12 -p tcp --dport 22 -j MASQUERADE
iptables -t nat -A POSTROUTING -o v912 -p tcp --dport 22 -j MASQUERADE
iptables -t nat -A POSTROUTING -o v2 -p tcp --dport 22 -j MASQUERADE
iptables -t nat -A POSTROUTING -o v100 -p tcp --dport 22 -j MASQUERADE

So now we will have SSH traffic going out for both the Tenants while we only want that for Blue Tenant. So we will be dropping SSH traffic coming from Red Tenant but not only the ones which are not destined to Tenant Blue, since out 2nd part of the question needs us to let the both tenants SSH each other.

At Provider_ns (Any ssh packet coming from subnet 220.0/24 through interface v911 and about to go to interface v2 which means towards internet drop it)
iptables -I FORWARD 1 -i v911 -o v2 -p tcp --dport 22 -s 192.168.220.0/24 -j DROP

So we have our network where only Blue Tenant would be able to SSH to internet.

Now for both SSH's to work since provider_ns only knows about the NS1 and NS2 subnets we need to add routes inside provider_ns with directing the packets to T1-VM1 subnet and T1-VM2 subnet through v11(120.2/24) and vice-versa.

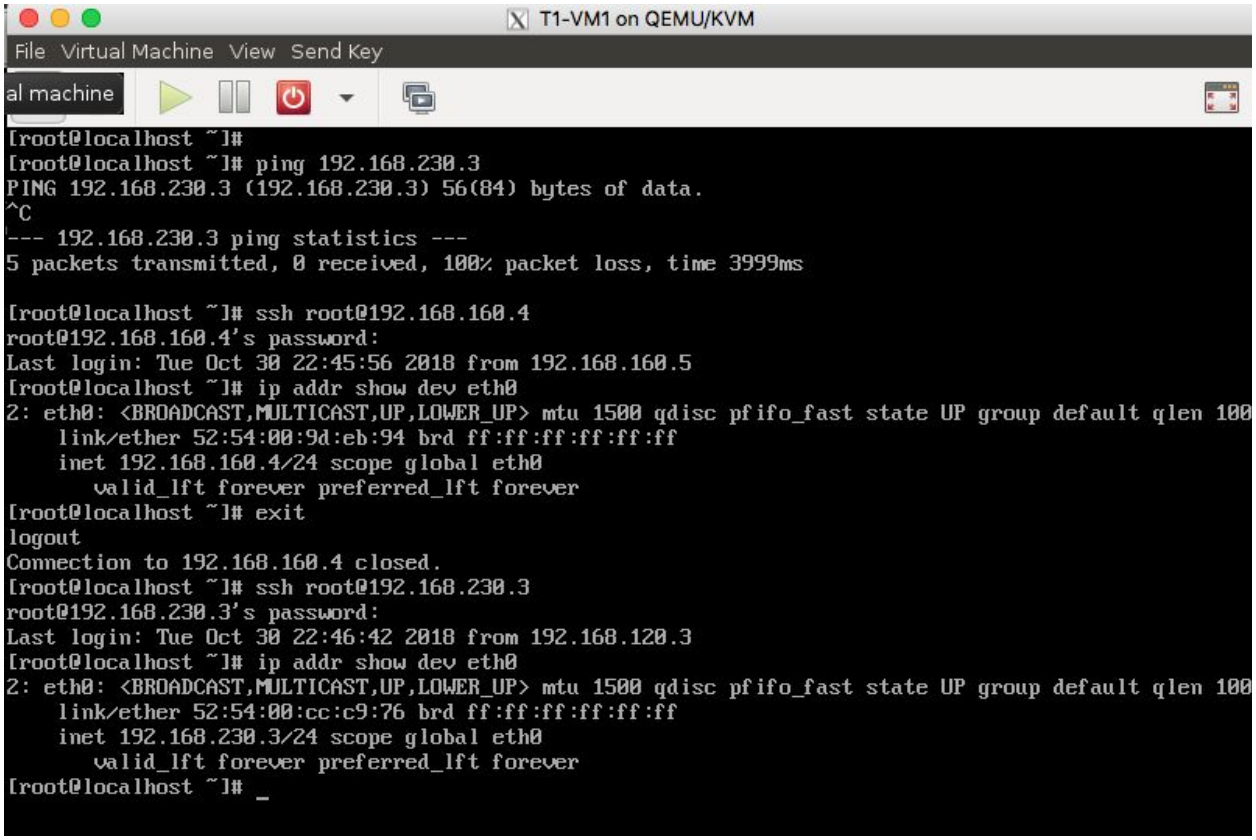
At provider_ns:
ip route add 192.168.230.0/24 via 192.168.220.3 dev v911
ip route add 192.168.240.0/24 via 192.168.220.3 dev v911
ip route add 192.168.130.0/24 via 192.168.120.3 dev v11
ip route add 192.168.140.0/24 via 192.168.120.3 dev v11

But doing so we now open the connections for icmp packets as well. So a ping for T2 to T1 would be successful which should not happen.
So we have to block the icmp traffic which is coming from T2 and wants to go to T1 should be dropped and vice versa.

At provider_ns:
//For T2 to not reach T1
iptables -I FORWARD 1 -i v911 -o v11 -p icmp -s 192.168.220.0/24 -j DROP
//For T1 to not reach T2
iptables -I FORWARD 1 -i v11 -o v911 -p icmp -s 192.168.120.0/24 -j DROP

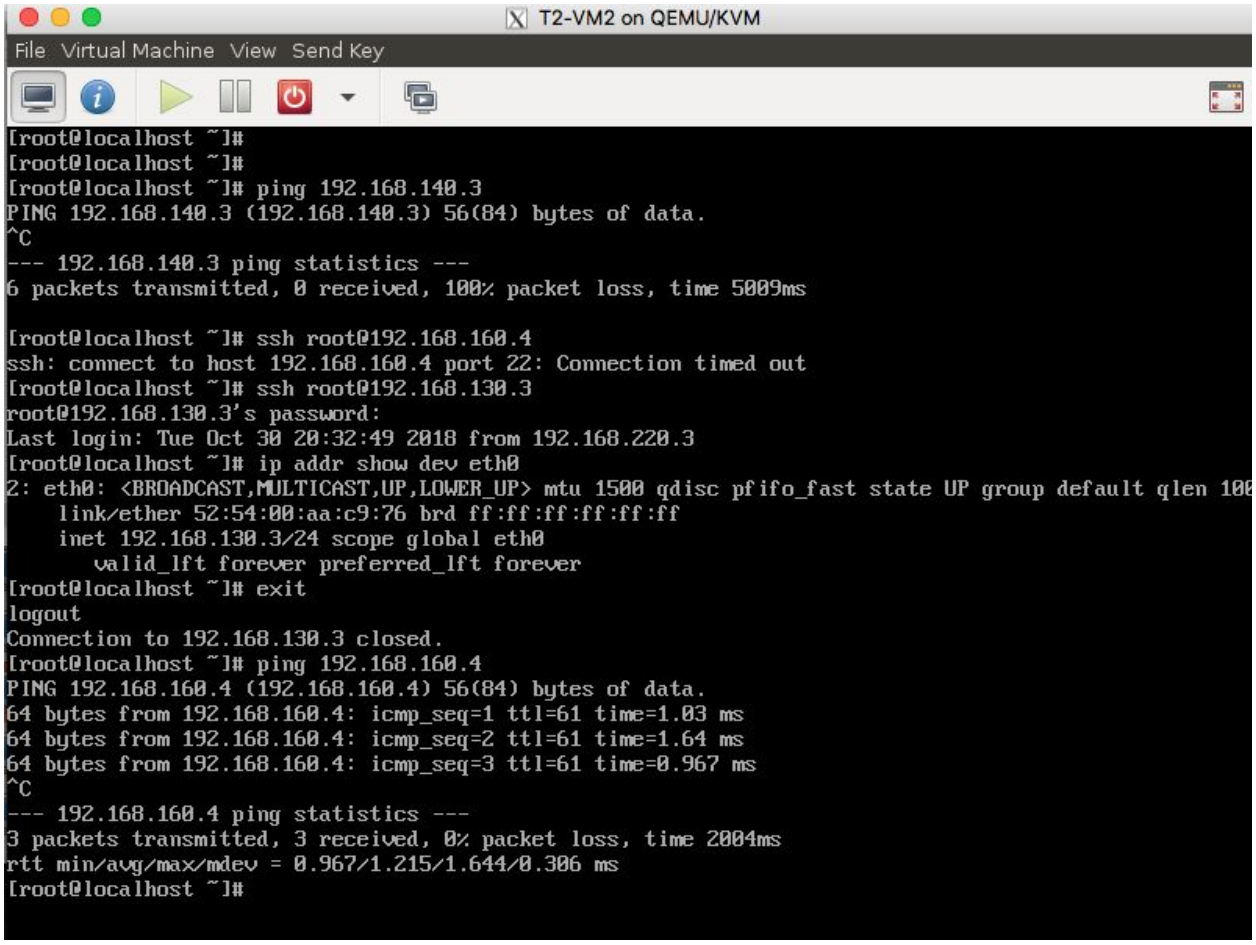
With these configuration we would have both the policies working in parallel and all the requirements would be satisfied also no one from internet can ping or ssh anyone.

SSH from T1-VM1 to internet and other tenant parallely:



As you can see ping won't be working to Tenant 2(Red) while SSH is possible and also parallely we can ssh to internet as well.

SSH from T2-VM2 to internet and other tenant parallely:



As you can see here from Tenant 2(Red) we are unable to SSH to internet while we are able to ssh to other tenant and at the same time it is not able to ping other tenant(blue) and can ping the internet. This is the desired policy combined.

Packet Capture when Red Tenant tries to SSH to tenant Blue at NS1:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~#
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v21
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v21, link-type EN10MB (Ethernet), capture size 262144 bytes
^C23:16:57.876008 IP 192.168.130.3.35778 > 192.168.230.3.ssh: Flags [P.], seq 3278354631:3278354675, ack 1437497620, win 309, options [nop,nop,TS val 348574407 ecr 330648684], le
h 44
23:16:57.877919 IP 192.168.230.3.ssh > 192.168.130.3.35778: Flags [P.], seq 1:61, ack 44, win 273, options [nop,nop,TS val 332046039 ecr 348574407], length 60
23:16:57.878814 IP 192.168.130.3.35778 > 192.168.230.3.ssh: Flags [.], ack 61, win 309, options [nop,nop,TS val 348574412 ecr 332046039], length 0
23:17:02.890545 ARP, Request who-has 192.168.130.2 tell 192.168.130.3, length 28
23:17:02.890567 ARP, Reply 192.168.130.2 is-at a6:b1:b2:13:a1:21 (oui Unknown), length 28
23:17:03.128804 ARP, Request who-has 192.168.130.3 tell 192.168.130.2, length 28
23:17:03.129267 ARP, Reply 192.168.130.3 is-at 52:54:00:aa:c9:76 (oui Unknown), length 28
23:19:32.364081 IP 192.168.130.3.35778 > 192.168.230.3.ssh: Flags [P.], seq 952:988, ack 1145, win 309, options [nop,nop,TS val 348728897 ecr 332074022], length 36
23:19:32.364541 IP 192.168.230.3.ssh > 192.168.130.3.35778: Flags [.], ack 988, win 273, options [nop,nop,TS val 332200525 ecr 348728897], length 0
23:19:32.365554 IP 192.168.230.3.ssh > 192.168.130.3.35778: Flags [P.], seq 1145:1181, ack 988, win 273, options [nop,nop,TS val 332200526 ecr 348728897], length 36
23:19:32.367234 IP 192.168.130.3.35778 > 192.168.230.3.ssh: Flags [.], ack 1181, win 309, options [nop,nop,TS val 348728900 ecr 332200526], length 0
23:19:32.369587 IP 192.168.230.3.ssh > 192.168.130.3.35778: Flags [P.], seq 1181:1241, ack 988, win 273, options [nop,nop,TS val 332200530 ecr 348728900], length 60
23:19:32.369882 IP 192.168.130.3.35778 > 192.168.230.3.ssh: Flags [.], ack 1241, win 309, options [nop,nop,TS val 348728903 ecr 332200530], length 0
```

You can see the source IP and Destination IP above which tells you where the ssh is intended and the seq and ack tells us that there is a successful connection as well.

Packet Capture when Red Tenant tries to SSH to tenant Blue at provider_ns:

```
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v11
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v11, link-type EN10MB (Ethernet), capture size 262144 bytes
^C23:20:27.023004 IP 192.168.120.3.35778 > 192.168.230.3.ssh: Flags [P.], seq 3278355619:3278355663, ack 1437498860, win 309, options [nop,nop,TS val 348783556 ecr 3
h 44
23:20:27.024509 IP 192.168.230.3.ssh > 192.168.120.3.35778: Flags [P.], seq 1:61, ack 44, win 273, options [nop,nop,TS val 332255185 ecr 348783556], length 60
23:20:27.026841 IP 192.168.120.3.35778 > 192.168.230.3.ssh: Flags [.], ack 61, win 309, options [nop,nop,TS val 348783559 ecr 332255185], length 0
23:20:29.217083 IP 192.168.120.3.35778 > 192.168.230.3.ssh: Flags [P.], seq 44:80, ack 61, win 309, options [nop,nop,TS val 348785750 ecr 332255185], length 36
23:20:29.286891 IP 192.168.230.3.ssh > 192.168.120.3.35778: Flags [.], ack 80, win 273, options [nop,nop,TS val 332257446 ecr 348785750], length 0
23:20:29.297490 IP 192.168.230.3.ssh > 192.168.120.3.35778: Flags [P.], seq 61:97, ack 80, win 273, options [nop,nop,TS val 332257458 ecr 348785750], length 36
23:20:29.301522 IP 192.168.120.3.35778 > 192.168.230.3.ssh: Flags [.], ack 97, win 309, options [nop,nop,TS val 348785833 ecr 332257458], length 0

7 packets captured
11 packets received by filter
4 packets dropped by kernel
```

Packet Capture when Blue Tenant tries to SSH to internet at NS1:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~#
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v21
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v21, link-type EN10MB (Ethernet), capture size 262144 bytes
^C23:30:54.033567 IP 192.168.130.3.48164 > 192.168.160.4.ssh: Flags [P.], seq 2426353908:2426353952, ack 3377694454, win 309, options [nop,nop,TS val 349410567 ecr 17585188],
 44
23:30:54.036041 IP 192.168.160.4.ssh > 192.168.130.3.48164: Flags [P.], seq 1:45, ack 44, win 273, options [nop,nop,TS val 17694834 ecr 349410567], length 44
23:30:54.037416 IP 192.168.130.3.48164 > 192.168.160.4.ssh: Flags [.], ack 45, win 309, options [nop,nop,TS val 349410570 ecr 17694834], length 0
23:30:54.518309 IP 192.168.130.3.48164 > 192.168.160.4.ssh: Flags [P.], seq 44:88, ack 45, win 309, options [nop,nop,TS val 349411051 ecr 17694834], length 44
23:30:54.521214 IP 192.168.160.4.ssh > 192.168.130.3.48164: Flags [P.], seq 45:89, ack 88, win 273, options [nop,nop,TS val 17695319 ecr 349411051], length 44
23:30:54.521641 IP 192.168.130.3.48164 > 192.168.160.4.ssh: Flags [.], ack 89, win 309, options [nop,nop,TS val 349411055 ecr 17695319], length 0
23:30:54.866789 IP 192.168.130.3.48164 > 192.168.160.4.ssh: Flags [P.], seq 88:132, ack 89, win 309, options [nop,nop,TS val 349411400 ecr 17695319], length 44
```

Packet Capture when Blue Tenant tries to SSH to internet at Provider_ns:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v11
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v11, link-type EN10MB (Ethernet), capture size 262144 bytes
^C23:39:54.980263 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [S], seq 1334491976, win 29200, options [mss 1460,sackOK,TS val 349951513 ecr 0,nop,wscale 7], length 0
23:39:54.982126 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [S.], seq 2774253402, ack 1334491977, win 28960, options [mss 1460,sackOK,TS val 18235780 ecr 349951513,nop,wscale
7], length 0
23:39:54.982974 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [.], ack 1, win 229, options [nop,nop,TS val 349951516 ecr 18235780], length 0
23:39:54.987295 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [P.], seq 1:22, ack 1, win 229, options [nop,nop,TS val 349951520 ecr 18235780], length 21
23:39:54.987732 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [.], ack 22, win 227, options [nop,nop,TS val 18235786 ecr 349951520], length 0
23:39:55.040117 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [P.], seq 1:22, ack 22, win 227, options [nop,nop,TS val 18235838 ecr 349951520], length 21
23:39:55.041940 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [.], ack 22, win 229, options [nop,nop,TS val 349951575 ecr 18235838], length 0
23:41:46.162093 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [P.], seq 3474:3518, ack 11606, win 529, options [nop,nop,TS val 350062695 ecr 18266465], length 44
23:41:46.190859 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [P.], seq 11606:11650, ack 3518, win 273, options [nop,nop,TS val 18346987 ecr 350062695], length 44
23:41:46.202308 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [P.], seq 11650:11694, ack 3518, win 273, options [nop,nop,TS val 18347000 ecr 350062695], length 44
23:41:46.204806 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [P.], seq 11694:11754, ack 3518, win 273, options [nop,nop,TS val 18347003 ecr 350062695], length 60
23:41:46.208106 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [.], ack 11650, win 529, options [nop,nop,TS val 350062741 ecr 18346987], length 0
23:41:46.208209 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [.], ack 11694, win 529, options [nop,nop,TS val 350062741 ecr 18347000], length 0
23:41:46.208502 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [.], ack 11754, win 529, options [nop,nop,TS val 350062741 ecr 18347003], length 0
23:41:49.641627 IP 192.168.120.3.48172 > 192.168.160.4.ssh: Flags [P.], seq 3518:3554, ack 11754, win 529, options [nop,nop,TS val 350066175 ecr 18347003], length 36
23:41:49.643756 IP 192.168.160.4.ssh > 192.168.120.3.48172: Flags [P.], seq 11754:11814, ack 3554, win 273, options [nop,nop,TS val 18350442 ecr 350066175], length 60
```

Packet Capture when Blue Tenant tries to SSH to internet at internet_ns:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# tcpdump -i v1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on v1, link-type EN10MB (Ethernet), capture size 262144 bytes
^C23:46:32.815794 IP 192.168.110.3.48174 > 192.168.160.4.ssh: Flags [S], seq 1913342630, win 29200, options [mss 1460,sackOK,TS val 350349349 ecr 0,nop,wscale 7], length 0
23:46:32.816489 IP 192.168.160.4.ssh > 192.168.110.3.48174: Flags [S.], seq 2199655694, ack 1913342631, win 28960, options [mss 1460,sackOK,TS val 18633614 ecr 350349349,nop,wscale
7], length 0
23:46:32.818067 IP 192.168.110.3.48174 > 192.168.160.4.ssh: Flags [.], ack 1, win 229, options [nop,nop,TS val 350349351 ecr 18633614], length 0
23:46:32.822683 IP 192.168.110.3.48174 > 192.168.160.4.ssh: Flags [P.], seq 1:22, ack 1, win 229, options [nop,nop,TS val 350349353 ecr 18633614], length 21
23:46:32.825926 IP 192.168.160.4.ssh > 192.168.110.3.48174: Flags [.], ack 22, win 227, options [nop,nop,TS val 18633624 ecr 350349353], length 0
23:46:32.874351 IP 192.168.160.4.ssh > 192.168.110.3.48174: Flags [P.], seq 1:22, ack 22, win 227, options [nop,nop,TS val 18633672 ecr 350349353], length 21
23:46:32.882711 IP 192.168.110.3.48174 > 192.168.160.4.ssh: Flags [.], ack 22, win 229, options [nop,nop,TS val 350349416 ecr 18633672], length 0
```

You can observe above all the screenshots packet capture show you the Source Ip changing at places where we applied the NAT rules while the destination staying as the Internet.

Now we will see all the Forwarding Tables
Note: All the above screenshots have been run with all the rules at the same time.

Table at NS1:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# ip route
default via 192.168.120.2 dev v12
192.168.120.0/24 dev v12  proto kernel  scope link  src 192.168.120.3
192.168.130.0/24 dev v21  proto kernel  scope link  src 192.168.130.2
192.168.140.0/24 dev v31  proto kernel  scope link  src 192.168.140.2
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  tcp  --  anywhere             anywhere             tcp dpt:ssh
MASQUERADE  icmp --  anywhere             anywhere
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
```

Tables at NS2:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~#
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# ip route
default via 192.168.220.2 dev v912
192.168.220.0/24 dev v912  proto kernel  scope link  src 192.168.220.3
192.168.230.0/24 dev v921  proto kernel  scope link  src 192.168.230.2
192.168.240.0/24 dev v931  proto kernel  scope link  src 192.168.240.2
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  icmp --  anywhere             anywhere
MASQUERADE  tcp  --  anywhere             anywhere             tcp dpt:ssh
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
```

Tables at Provider_ns:

```
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
root@ece792-Standard-PC-i440FX-PIIX-1996:~# ip route
default via 192.168.110.2 dev v2
192.168.110.0/24 dev v2  proto kernel  scope link  src 192.168.110.3
192.168.120.0/24 dev v11  proto kernel  scope link  src 192.168.120.2
192.168.130.0/24 via 192.168.120.3 dev v11
192.168.140.0/24 via 192.168.120.3 dev v11
192.168.220.0/24 dev v911  proto kernel  scope link  src 192.168.220.2
192.168.230.0/24 via 192.168.220.3 dev v911
192.168.240.0/24 via 192.168.220.3 dev v911
root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  tcp  --  anywhere             anywhere             tcp dpt:ssh
MASQUERADE  icmp --  anywhere             anywhere
root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A FORWARD -s 192.168.220.0/24 -i v911 -o v11 -p icmp -j DROP
-A FORWARD -s 192.168.120.0/24 -i v11 -o v911 -p icmp -j DROP
-A FORWARD -s 192.168.220.0/24 -i v911 -o v2 -p tcp -m tcp --dport 22 -j DROP
root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  192.168.220.0/24     anywhere
DROP       icmp --  192.168.120.0/24     anywhere
DROP       tcp  --  192.168.220.0/24     anywhere             tcp dpt:ssh

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
```

Tables at Internet_ns:

```
[root@ece792-Standard-PC-i440FX-PIIX-1996:~#
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# ip route
default via 192.168.160.4 dev v100
192.168.110.0/24 dev v1  proto kernel  scope link  src 192.168.110.2
192.168.160.0/24 dev v100  proto kernel  scope link  src 192.168.160.5
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  icmp --  anywhere             anywhere
MASQUERADE  tcp  --  192.168.110.0/24     anywhere
[root@ece792-Standard-PC-i440FX-PIIX-1996:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@ece792-Standard-PC-i440FX-PIIX-1996:~#
```