HOMEWORK 2

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

[1]

What is the CPU/Memory requirement for 500 Mbps, 1000 Mbps, and 5000 Mbps throughput for CSR routers.

Based on the needed features the requirements are mentioned with AX meaning for all features to be present.

Throughput	IP Base	Security	AppX	AX(All Features)
500 Mbps	1 vCPU/4 GB	1 vCPU/4 GB	1 vCPU/4 GB	1 vCPU/4 GB
1000 Mbps	1 vCPU/4 GB	1 vCPU/4 GB	1 vCPU/4 GB	2 vCPU/4 GB
5000 Mbps	1 vCPU/4 GB	2 vCPU/4 GB	8 vCPU/4 GB	8 vCPU/4 GB

List three features of CSR for each of the following:

a) Networking:

- Routing: BGP, OSPF, EIGRP, Policy Based Routing, IPv6, Multicast,LISP, GRE and Connectionless Network Services
- NFV: vBNG, vISG, and vRR
- Application visibility, performance monitoring, and control: QoS and AVC
- Addressing: DHCP, Domain Name System (DNS), NAT, 802.1Q VLAN, Ethernet Virtual Connection (EVC), and VXLAN

b) Security:

- VPN: IPsec VPN, DMVPN, Easy VPN, FlexVPN, and GetVPN
- Firewall: ZBFW
- Access control: ACL, AAA, RADIUS, and TACACS+

c) Management interface:

- Virtual-machine creation and deployment: VMware vCenter and VMware vCloud Director
- Provisioning and management: Cisco IOS XE CLI, Secure Shell (SSH) Protocol, Telnet, Cisco Prime Infrastructure, Cisco Prime Network Services Controller, and OpenStack Configdrive
- Monitoring and troubleshooting: Simple Network Management Protocol (SNMP), Syslog, NetFlow, IP SLA, and Embedded Event Manager (EEM)

What would be per year cost to use two 1000 Mbps CSR in Amazon cloud ?

For 1000 Mbps CSR in Amazon Cloud it is recommended to use c4.xlarge instances which costs $4550 for each one so as we need 2 of them.

Total per year cost would be: **$9100**

Below Screenshots show the estimation given by AWS.

Product Overview

The AX Technology Package for Maximum Performance version of Cisco's Cloud Services Router (CSR1000V) delivers the maximum performance available in AWS cloud for virtual networking services. Deliver high-speed secure VPN services with High Availability, strong Firewall protection, Application Visibility & Control, and more... This AMI runs Cisco IOS XE technology features (ASR1000 and ISR4000 series) and uses AWS instances with direct I/O path for higher & more consistent performance, as well as 2x performance with IMIX packets. The CSR with full Cisco IOS-XE support enables customers to deploy the same enterprise-class networking services that they are so used to in their on-prem networks inside AWS. This AMI enables enterprise-class Routing, VPN, High-Availability, Firewall, IP SLA, VPC Interconnection, Application Visibility & Control, Performance Monitoring, Optimization. It includes the following functionality: (1) CSR Base Tech Package: BGP, OSPF, EIGRP, RIP, ISIS, IPv6, GRE, VRF-LITE, NTP, QoS, 802.1Q VLAN, EVC, NAT, DHCP, DNS, ACL, AAA, RADIUS, TACACS+, IOS-XE CLI, SSH, Flexible NetFlow, SNMP, EEM, and NETCONF. (2) CSR Security Tech Packag: Zone Based Firewall, IPSec, DMVPN, GETVPN, EZVPN, FLEXVPN, SSL VPN, and VTI-VPN. (3) CSR AppX Tech Package: BFD, MPLS, VXLAN, WCCPv2, AppXNAV, NBAR2, AVC, IP SLA, PTA, LNS, ISG, and LISP. The familiar IOS XE CLI and RESTful API ensures easy deployment, monitoring, troubleshooting, and service orchestration.

Highlights

- Enterprise-class VPN in AWS that's faster, cheaper, and more scalable than other VPN solutions. Manage both sides of your VPN for greater security. Familiar IOS-XE based VPN supports the same commands, tools, and logs as Cisco ISR and ASR platforms.
- More secure, reliable, and cost effective than native VPN. Feature-rich: IPSec, DMVPN, FlexVPN, GETVPN, EZVPN, SSL VPN, Zone-Based Firewall, and more...
- IPSec performance Guide: t2.medium for up to 250 Mbps. c4.large for up to 500 Mbps. c4.xlarge for 1 Gbps. c3.2xlarge for up to 1Gbps, c4.2xlarge for up to 2.5 Gbps and c4.4xlarge for up to 4.5Gbps

Estimating your costs

Choose your region and fulfillment option to see the pricing details. Then, modify the estimated price by choosing different instance types.

Region

US East (N. Virginia)

Fulfillment Option

64-bit Amazon Machine Image (AMI)

Software Pricing Details

Cisco Cloud Services Router (CSR) 1000V - AX Pkg. Max Performance **$4,550 /yr**

running on c4.xlarge

Infrastructure Pricing Details

Estimated Infrastructure Cost $0.199 EC2/hr

Free Trial Try one instance of this product for 30 days. There will be no hourly software charges for that instance, but AWS infrastructure charges still apply. Free Trials will automatically convert to a paid hourly subscription upon expiration.

The table shows current software and infrastructure pricing for services hosted in **US East (N. Virginia)**. Additional taxes or fees may apply.

Cisco Cloud Services Router (CSR) 1000V - AX Pkg. Max Performance

Switch to annual pricing for savings up to **59%**

Hourly Annual

EC2 Instance type	Software/yr	EC2/hr	Percent Savings (%)
t2.medium	$2,233	$0.046	59%
c4.large ★Vendor Recommended	$3,723	$0.1	59%
c4.xlarge	$4,550	$0.199	59%
c4.2xlarge	$6,363	$0.398	59%
c4.4xlarge	$8,952	$0.796	59%
c4.8xlarge	$8,952	$1.591	59%

[2] Creating VMs:

We have created the VM with the command:
sudo virt-install -n mreddy2_VM1 -r 2048 --vcpu=4 --cpu host --disk path=/var/lib/libvirt/images/mreddy2_VM1.img,size=10 --network network=default -c /home/ece792/CentOS-7-x86_64-Minimal-1708.iso.1 -v

And then configured the VM with the user and password and the IP addresses are assigned with dhclient.

[2.1]

The IP Address and MAC at VM's NIC is as below:
VM IP Address: 192.168.122.106
VM Mac Address: 52:54:00:87:3f:9b

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domifaddr mreddy2_VM1
 Name       MAC address          Protocol     Address
-------------------------------------------------------------------------------
 vnet0      52:54:00:87:3f:9b    ipv4         192.168.122.106/24

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

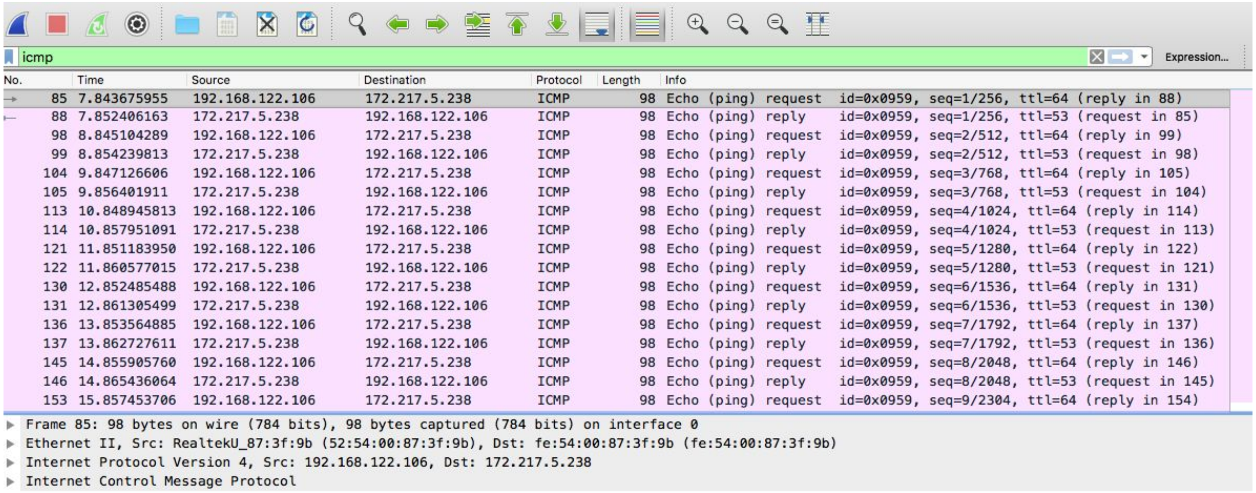
The IP Address and MAC at the Hypervisor's NIC is as below:
IP Address: 192.168.124.5
MAC Address: 52:54:00:06:55:66

```
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:06:55:66 brd ff:ff:ff:ff:ff:ff
    inet 192.168.124.5/24 brd 192.168.124.255 scope global dynamic ens3
       valid_lft 2828sec preferred_lft 2828sec
    inet6 fe80::faaa:649d:f116:3492/64 scope link
       valid_lft forever preferred_lft forever
```

[2.2]

Pinging Google.com from the VM and capturing packets at the output interface of the VM as well as at the output interface of the hypervisor.

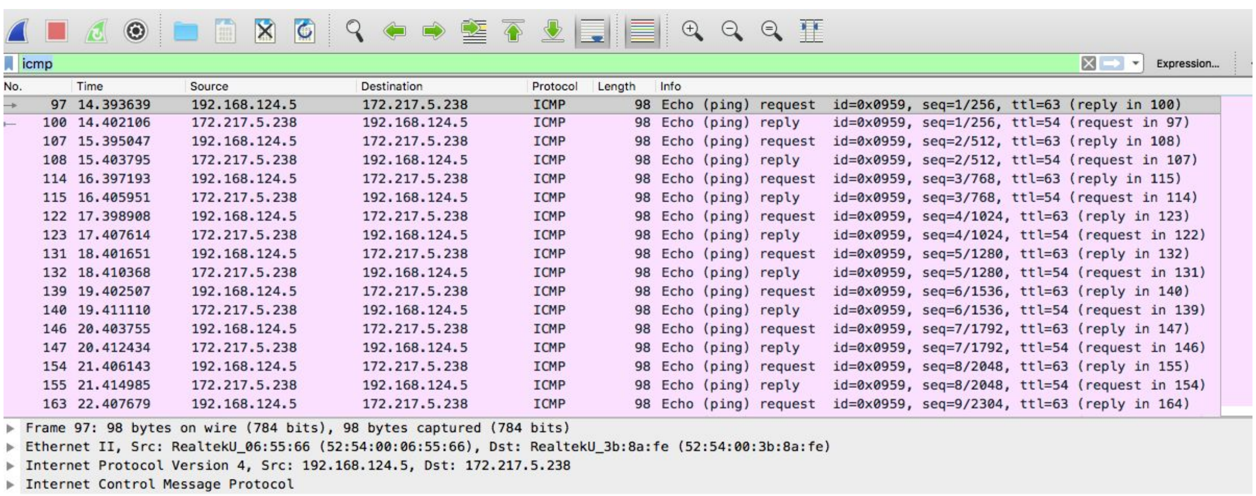
At the output Interface of VM:
srcIP: 192.168.122.106
DestIP: 172.217.5.238
srcMAC: 52:54:00:87:3f:9b
DestMAC: fe:54:00:87:3f:9b



No.	Time	Source	Destination	Protocol	Length	Info
85	7.843675955	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=1/256, ttl=64 (reply in 88)
88	7.852406163	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=1/256, ttl=53 (request in 85)
98	8.845104289	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=2/512, ttl=64 (reply in 99)
99	8.854239813	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=2/512, ttl=53 (request in 98)
104	9.847126606	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=3/768, ttl=64 (reply in 105)
105	9.856401911	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=3/768, ttl=53 (request in 104)
113	10.848945813	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=4/1024, ttl=64 (reply in 114)
114	10.857951091	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=4/1024, ttl=53 (request in 113)
121	11.851183950	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=5/1280, ttl=64 (reply in 122)
122	11.860577015	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=5/1280, ttl=53 (request in 121)
130	12.852485488	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=6/1536, ttl=64 (reply in 131)
131	12.861305499	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=6/1536, ttl=53 (request in 130)
136	13.853564885	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=7/1792, ttl=64 (reply in 137)
137	13.862727611	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=7/1792, ttl=53 (request in 136)
145	14.855905760	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=8/2048, ttl=64 (reply in 146)
146	14.865436064	172.217.5.238	192.168.122.106	ICMP	98	Echo (ping) reply id=0x0959, seq=8/2048, ttl=53 (request in 145)
153	15.857453706	192.168.122.106	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=9/2304, ttl=64 (reply in 154)

Frame 85: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: RealtekU_87:3f:9b (52:54:00:87:3f:9b), Dst: fe:54:00:87:3f:9b (fe:54:00:87:3f:9b)
Internet Protocol Version 4, Src: 192.168.122.106, Dst: 172.217.5.238
Internet Control Message Protocol

At the output Interface of Hypervisor:
srcIP: 192.168.124.5
DestIP: 172.217.5.238
srcMAC: 52:54:00:06:55:66
DestMAC:52:54:00:3b:8a:fe



No.	Time	Source	Destination	Protocol	Length	Info
97	14.393639	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=1/256, ttl=63 (reply in 100)
100	14.402106	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=1/256, ttl=54 (request in 97)
107	15.395047	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=2/512, ttl=63 (reply in 108)
108	15.403795	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=2/512, ttl=54 (request in 107)
114	16.397193	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=3/768, ttl=63 (reply in 115)
115	16.405951	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=3/768, ttl=54 (request in 114)
122	17.398908	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=4/1024, ttl=63 (reply in 123)
123	17.407614	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=4/1024, ttl=54 (request in 122)
131	18.401651	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=5/1280, ttl=63 (reply in 132)
132	18.410368	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=5/1280, ttl=54 (request in 131)
139	19.402507	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=6/1536, ttl=63 (reply in 140)
140	19.411110	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=6/1536, ttl=54 (request in 139)
146	20.403755	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=7/1792, ttl=63 (reply in 147)
147	20.412434	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=7/1792, ttl=54 (request in 146)
154	21.406143	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=8/2048, ttl=63 (reply in 155)
155	21.414985	172.217.5.238	192.168.124.5	ICMP	98	Echo (ping) reply id=0x0959, seq=8/2048, ttl=54 (request in 154)
163	22.407679	192.168.124.5	172.217.5.238	ICMP	98	Echo (ping) request id=0x0959, seq=9/2304, ttl=63 (reply in 164)

Frame 97: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: RealtekU_06:55:66 (52:54:00:06:55:66), Dst: RealtekU_3b:8a:fe (52:54:00:3b:8a:fe)
Internet Protocol Version 4, Src: 192.168.124.5, Dst: 172.217.5.238
Internet Control Message Protocol

We observe that the Destination IP Address stays the same while Source IP changes at VM and Hypervisor since when the packet is leaving the Hypervisor it changes the srcIP since the VM's IP address is not public IP. For the reply packet to reach back to the VM NAT operation is done at the bridge. This is the reason why we see different tuples. While in case of Destination MAC at VM is the MAC address of the Default Virtual Bridge and the Destination MAC at the Hypervisor is the next Hop after leaving the Hypervisor. This is the reason why we see different tuples.

[3]

[3.1] Use libvirt-CLI methods to add a network (name it as < your − unity − id >NETWORK2) in bridge mode.

At the file location etc/libvirt/qemu/networks we perform the following command which creates a new network xml file from the default file.

sudo virsh net-dumpxml default > mreddy2_NETWORK2.xml

Then we modify the file as below giving the network name and bridge name:

```
<network>
  <name>mreddy2_NETWORK2</name>
  <uuid>53a88238-85e9-4864-8175-d9972899b0cc</uuid>
  <forward mode='bridge'/>
  <bridge name='sw1'/>
</network>
```

Then we define and start the network and also create the bridge:

virsh net-define mreddy2_NETWORK2.xml
virsh net-start mreddy2_NETWORK2.xml
brctl addbr sw1

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh net-list
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes
 mreddy2_NETWORK2     active     no            yes
 sparasa_NETWORK2     active     yes           yes
 sparasa_NETWORK3     active     yes           yes

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

[3.2] Use libvirt-CLI methods to add an interface to your VM to connect to < your−unity−id >NETWORK2

Now we add an interface at our connected to the network we created, so we will basically have a connection between the VM's Interface and the Bridge.

virsh attach-interface --domain mreddy2_VM1 --type network --source mreddy2_NETWORK2 --model virtio --mac 52:54:00:4b:73:5f --config --live

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:/etc/libvirt/qemu$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:/etc/libvirt/qemu$ virsh domifaddr mreddy2_VM1
 Name       MAC address          Protocol     Address
-------------------------------------------------------------------------------
 vnet1      52:54:00:87:3f:9b    ipv4         192.168.122.106/24

ece792@ece792-Standard-PC-i440FX-PIIX-1996:/etc/libvirt/qemu$ virsh attach-interface --domain mreddy2_VM1 --type network --source mreddy2_NETWORK2 --model virtio --mac 52:54:00:4b:7
3:5f --config --live
Interface attached successfully

ece792@ece792-Standard-PC-i440FX-PIIX-1996:/etc/libvirt/qemu$
```

[3.3] Use libvirt-CLI methods to clone your VM (name it as < your − unity − id >lab2VM2.
Now we clone the VM which creates the new VM with the name specified in the command. Before cloning the VM is suspended for cloning.
virt-clone --original mreddy2_VM1 --name sparasa_lab2VM2 --auto-clone

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:/etc/libvirt/qemu$ virsh list
 Id    Name                           State
----------------------------------------------------
 9     sparasa_lab2VM2                running
 11    sparasa_vm                     running
 12    sparasa_vm2                    running
 13    Q4-VM1                         running
 15    Q4-VM2                         running
 16    mreddy2_VM1                    running

ece792@ece792-Standard-PC-i440FX-PIIX-1996:/etc/libvirt/qemu$
```

[3.4] List MAC and IP addresses of all interfaces of each VM.

The IP Address and MAC at VM1 is as below:
VM1 IP Address: 192.168.122.106
VM1 Mac Address: 52:54:00:87:3f:9b

```
[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 52:54:00:87:3f:9b brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.106/24 brd 192.168.122.255 scope global dynamic eth0
       valid_lft 3447sec preferred_lft 3447sec
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:4b:73:5f brd ff:ff:ff:ff:ff:ff
[root@localhost ~]#
```

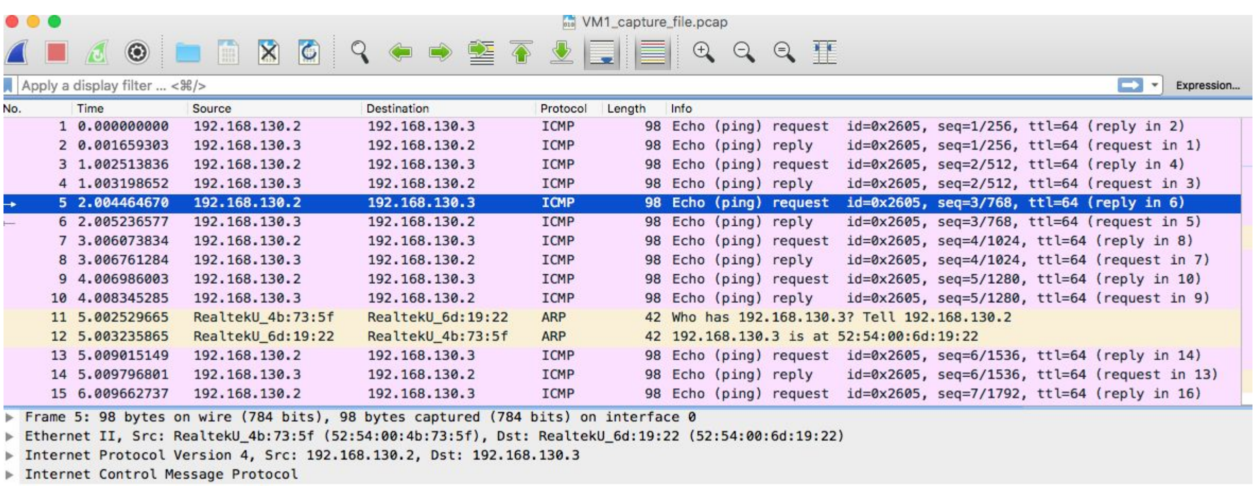
The IP Address and MAC at VM2(New VM created by cloning) is as below:
VM2 IP Address: 192.168.122.223
VM2 Mac Address: 52:54:00:f9:ae:55

```
[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:f9:ae:55 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.223/24 brd 192.168.122.255 scope global dynamic eth0
       valid_lft 3583sec preferred_lft 3583sec
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:6d:19:22 brd ff:ff:ff:ff:ff:ff
[root@localhost ~]# _
```

[3.5] Ping one VM from the other using IP from the < your − unity − id >NETWORK2 subnet. Use wireshark on the VMs and List down 4 tuples (srcIP, Dest IP, srcMAC and dest MAC) of packet going out at first VM and received at second VM. Are the tuple fields same or different for the packet? if different, explain why.

Now we need to assign IP address at both the VM's at the interface which is connected to the network2 we created since we need to ping using IP from the Network2 Subnet and also by default the bridge is down so we should bring the bridge up by **ip link set sw1 up** command. Then we ping the other VM with the Netwrok2 subnet addresses.

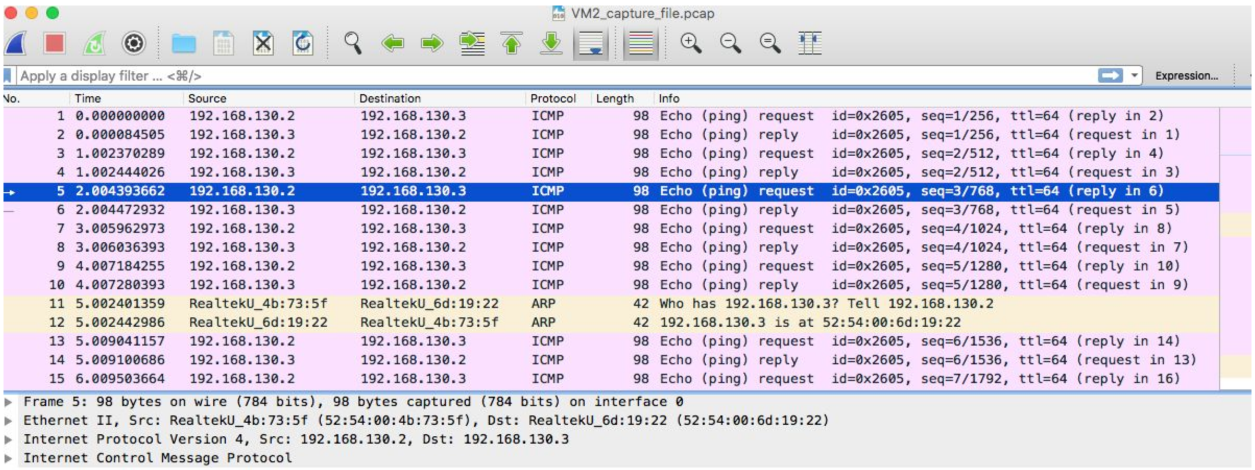
At the VM1 Interface:
srcIP: 192.168.130.2
DestIP: 192.168.130.3
srcMAC: 52:54:00:4b:73:5f
DestMAC: 52:54:00:6d:19:22



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=1/256, ttl=64 (reply in 2)
2	0.001659303	192.168.130.3	192.168.130.2	ICMP	98	Echo (ping) reply id=0x2605, seq=1/256, ttl=64 (request in 1)
3	1.002513836	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=2/512, ttl=64 (reply in 4)
4	1.003198652	192.168.130.3	192.168.130.2	ICMP	98	Echo (ping) reply id=0x2605, seq=2/512, ttl=64 (request in 3)
5	2.004464670	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=3/768, ttl=64 (reply in 6)
6	2.005236577	192.168.130.3	192.168.130.2	ICMP	98	Echo (ping) reply id=0x2605, seq=3/768, ttl=64 (request in 5)
7	3.006073834	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=4/1024, ttl=64 (reply in 8)
8	3.006761284	192.168.130.3	192.168.130.2	ICMP	98	Echo (ping) reply id=0x2605, seq=4/1024, ttl=64 (request in 7)
9	4.006986003	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=5/1280, ttl=64 (reply in 10)
10	4.008345285	192.168.130.3	192.168.130.2	ICMP	98	Echo (ping) reply id=0x2605, seq=5/1280, ttl=64 (request in 9)
11	5.002529665	RealtekU_4b:73:5f	RealtekU_6d:19:22	ARP	42	Who has 192.168.130.3? Tell 192.168.130.2
12	5.003235865	RealtekU_6d:19:22	RealtekU_4b:73:5f	ARP	42	192.168.130.3 is at 52:54:00:6d:19:22
13	5.009015149	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=6/1536, ttl=64 (reply in 14)
14	5.009796801	192.168.130.3	192.168.130.2	ICMP	98	Echo (ping) reply id=0x2605, seq=6/1536, ttl=64 (request in 13)
15	6.009662737	192.168.130.2	192.168.130.3	ICMP	98	Echo (ping) request id=0x2605, seq=7/1792, ttl=64 (reply in 16)

```
▶ Frame 5: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▶ Ethernet II, Src: RealtekU_4b:73:5f (52:54:00:4b:73:5f), Dst: RealtekU_6d:19:22 (52:54:00:6d:19:22)
▶ Internet Protocol Version 4, Src: 192.168.130.2, Dst: 192.168.130.3
▶ Internet Control Message Protocol
```

At the VM2 Interface:
srcIP: 192.168.130.2
DestIP: 192.168.130.3
srcMAC: 52:54:00:4b:73:5f
DestMAC: 52:54:00:6d:19:22



As we can see the tuples at both the VM's are same because since it is a pure L2 Bridge it doesn't modify the source and destination MAC addresses. That is the reason why we do not see different tuples at both points.

[3.6] Send UDP traffic between the two VMs using iperf. What is the maximum UDP throughput achieved? Which is the bottleneck resource, CPU, memory or I/O? why? Provide logs of system commands to support your answer.

After installing VM's on both VM's we send UDP traffic from one VM acting as a client and another VM acts as a server receiving the traffic.
Before sending the iptables are flushed to allow UDP traffic at the VM's with
Iptables -F

Then at Client **iperf3 -c 192.168.122.106 -u**
At Server **iperf3 -s**
The following outputs are shown as below:

```
[root@localhost ~]#
[root@localhost ~]# iperf3 -c 192.168.122.106 -u
Connecting to host 192.168.122.106, port 5201
[  4] local 192.168.122.223 port 52866 connected to 192.168.122.106 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   116 KBytes   950 Kbits/sec  82
[  4]   1.00-2.00   sec   129 KBytes  1.05 Mbits/sec  91
[  4]   2.00-3.00   sec   127 KBytes  1.04 Mbits/sec  90
[  4]   3.00-4.00   sec   129 KBytes  1.05 Mbits/sec  91
[  4]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec  90
[  4]   5.00-6.00   sec   129 KBytes  1.05 Mbits/sec  91
[  4]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec  90
[  4]   7.00-8.00   sec   129 KBytes  1.05 Mbits/sec  91
[  4]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  90
[  4]   9.00-10.00  sec   129 KBytes  1.05 Mbits/sec  91
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.24 MBytes  1.04 Mbits/sec  0.059 ms  0/897 (0%)
[  4] Sent 897 datagrams

iperf Done.
[root@localhost ~]#
```

```
[root@localhost ~]#
[root@localhost ~]# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.122.223, port 59672
[  5] local 192.168.122.106 port 5201 connected to 192.168.122.223 port 52866
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   116 KBytes   950 Kbits/sec  0.081 ms  0/82 (0%)
[  5]   1.00-2.00   sec   129 KBytes  1.05 Mbits/sec  0.125 ms  0/91 (0%)
[  5]   2.00-3.00   sec   127 KBytes  1.04 Mbits/sec  0.068 ms  0/90 (0%)
[  5]   3.00-4.00   sec   129 KBytes  1.05 Mbits/sec  0.052 ms  0/91 (0%)
[  5]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec  0.136 ms  0/90 (0%)
[  5]   5.00-6.00   sec   129 KBytes  1.05 Mbits/sec  0.057 ms  0/91 (0%)
[  5]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec  0.037 ms  0/90 (0%)
[  5]   7.00-8.00   sec   129 KBytes  1.05 Mbits/sec  0.048 ms  0/91 (0%)
[  5]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  0.074 ms  0/90 (0%)
[  5]   9.00-10.00  sec   129 KBytes  1.05 Mbits/sec  0.059 ms  0/91 (0%)
[  5]  10.00-10.04  sec  0.00 Bytes  0.00 bits/sec  0.059 ms  0/0 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-10.04  sec  0.00 Bytes  0.00 bits/sec  0.059 ms  0/897 (0%)
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

As we can see the bandwidth to be 1.04Mbits/sec as we did not mention any size the default is taken to be 1Megabit. For further testing purposes we mentioned the size to be 500M, 1G, 4G
As the size increases we have seen at a point the bandwidth to converge around
Around 250 Mbits/sec as you can see below

```
iperf3: interrupt - the client has terminated
[root@localhost ~]# iperf3 -c 192.168.122.106 -u -b 4G
Connecting to host 192.168.122.106, port 5201
[  4] local 192.168.122.223 port 60201 connected to 192.168.122.106 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec  29.6 MBytes   248 Mbits/sec  21438
[  4]   1.00-2.00   sec  27.1 MBytes   227 Mbits/sec  19625
[  4]   2.00-3.00   sec  36.1 MBytes   303 Mbits/sec  26166
[  4]   3.00-4.00   sec  32.5 MBytes   273 Mbits/sec  23529
[  4]   4.00-5.00   sec  40.1 MBytes   336 Mbits/sec  29030
[  4]   5.00-6.00   sec  19.5 MBytes   164 Mbits/sec  14134
[  4]   6.00-7.00   sec  27.1 MBytes   228 Mbits/sec  19640
[  4]   7.00-8.00   sec  38.5 MBytes   323 Mbits/sec  27916
[  4]   8.00-9.00   sec  33.5 MBytes   281 Mbits/sec  24241
[  4]   9.00-10.00  sec  24.1 MBytes   202 Mbits/sec  17417
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec   308 MBytes   258 Mbits/sec  0.044 ms  22876/223097 (10%)
[  4] Sent 223097 datagrams

iperf Done.
[root@localhost ~]#
```

While we are sending UDP traffic, at the hypervisor(host) we ran the top command to see the usages and found out that the CPU is the bottleneck resource which exceeds 100% upto 119% which is show in the below screenshot as well. This happens since we are sending UDP traffic more than the capability due to which CPU is causing the bottleneck.

```
...ot@localhost:~ — ssh -Y ece792@152.14.83.156 ... | ...X-PIIX-1996: ~ — ssh -Y ece792@152.14.83.156 | ...@localhost:~ — ssh -Y ece792@152.14.83.156 | ...@localhost:~ — ssh -Y ece792@152.14.83.156 ... | +
top - 02:15:54 up 28 days, 11:56,  7 users,  load average: 1.74, 0.65, 0.51
Tasks: 219 total,   1 running, 218 sleeping,   0 stopped,   0 zombie
%Cpu(s): 31.4 us, 33.8 sy,  0.0 ni, 34.5 id,  0.0 wa,  0.0 hi,  0.1 si,  0.2 st
KiB Mem : 24687732 total,  1212344 free,  8731180 used, 14744208 buff/cache
KiB Swap: 25162748 total, 25162748 free,        0 used. 15249668 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
31438 libvirt+  20   0 5152404 609864  26304 S 119.3  2.5 247:33.38 qemu-system-x86
11044 libvirt+  20   0 5128576 989956  25104 S 102.3  4.0 485:22.20 qemu-system-x86
11056 root      20   0       0      0      0 S  16.9  0.0   0:31.61 vhost-11044
31453 root      20   0       0      0      0 S  13.0  0.0   0:16.88 vhost-31438
28733 libvirt+  20   0 5173632 1.678g  25972 S   1.7  7.1 208:05.10 qemu-system-x86
  971 lightdm   20   0 1062620  85872  48180 S   1.3  0.3 198:22.92 unity-greeter
21188 libvirt+  20   0 5140428 1.751g  25808 S   1.3  7.4 235:29.94 qemu-system-x86
25913 libvirt+  20   0 5661908 2.159g  25856 S   1.0  9.2 183:03.86 qemu-system-x86
27997 libvirt+  20   0 5142576 580196  25664 S   1.0  2.4 122:26.16 qemu-system-x86
 1016 lightdm   20   0  826360 126452  28832 S   0.7  0.5 144:27.57 nm-applet
    7 root      20   0       0      0      0 S   0.3  0.0  11:46.40 rcu_sched
  900 root      20   0  364068  45220  28664 S   0.3  0.2  57:21.32 Xorg
  970 lightdm   20   0   43120   3712   3012 S   0.3  0.0  75:05.74 dbus-daemon
 1033 lightdm   20   0  403148  12864  11384 S   0.3  0.1 138:42.91 indicator-appli
29199 ece792    20   0   41928   3784   3128 R   0.3  0.0   0:00.54 top
    1 root      20   0  185244   5908   4016 S   0.0  0.0   0:48.49 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.39 kthreadd
    4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:07.71 ksoftirqd/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.17 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.10 migration/0
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S   0.0  0.0   0:07.22 watchdog/0
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:06.66 watchdog/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 migration/1
   16 root      20   0       0      0      0 S   0.0  0.0   1:05.75 ksoftirqd/1
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/2
   20 root      rt   0       0      0      0 S   0.0  0.0   0:07.47 watchdog/2
   21 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 migration/2
   22 root      20   0       0      0      0 S   0.0  0.0   0:01.44 ksoftirqd/2
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:0H
   25 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/3
   26 root      rt   0       0      0      0 S   0.0  0.0   0:06.56 watchdog/3
   27 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 migration/3
   28 root      20   0       0      0      0 S   0.0  0.0   0:01.21 ksoftirqd/3
   30 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/3:0H
   31 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   32 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   33 root      20   0       0      0      0 S   0.0  0.0   0:01.82 khungtaskd
   34 root      20   0       0      0      0 S   0.0  0.0   0:00.00 oom_reaper
   35 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   36 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kcompactd0
   37 root      25   5       0      0      0 S   0.0  0.0   0:00.00 ksmd
   38 root      39  19       0      0      0 S   0.0  0.0   0:15.49 khugepaged
```

[6] Distributed and Stand alone vSwitch:

Generally Stand alone vSwitch works within one ESX/ESXi host only while in case of Distributed vSwitches it allows different hosts to use the switch as long as they exist within the same host cluster. From this we understand that instead of making virtual networks more complicated with its additional options, the distributed vSwitch simplifies operations and helps catch configuration errors and increase network visibility which makes it easy to manage the network as well.

Use cases where the Distributed Switch is needed instead of Stand alone vSwitch would be:

- ❏ In case of Network rollback when a misconfiguration is found and for network health checking. Distributed Switch is needed which can be saved and restored in such scenarios.
- ❏ In use cases where you need to shape inbound traffic, Standalone won't support it and you would need a Distributed Switch, which can restrict network bandwidth available to a port and also configure to temporarily allow burst of traffic to flow through the port at higher speeds.
- ❏ One other case would be Port Mirroring, in cases where you need to log all the packets received at the switch to a networking device connected to another switch port, you would need a Distributed Switch.

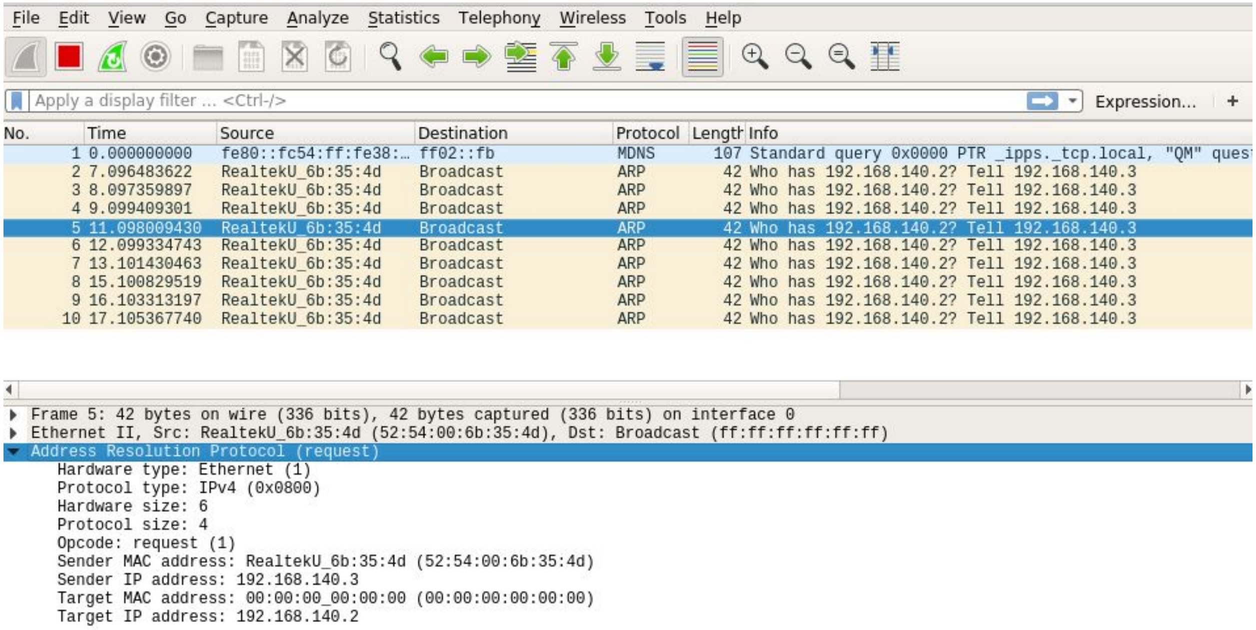
So in the industry say you have a scenario where a customer using Stand alone vSwitch has multiple VM's which means he has multiple switches as well handling these VM's and a case where a misconfiguration is found which was present in all the switches, so in this case he would have to manually configure all the switches again and has to know exactly what changes to make. While in such a scenario had the customer used a Distributed Switch it has the capability of storing the configurations and in the case where misconfigurations are found the customer could simply roll back to the the desired configurations which were in working desired configurations. In such a case Distributed Switch is needed.

[7] Set up a lab experiment to support your explanation to answer the following questions. What breaks if:

[7.1(a)] Two VMs connected to same bridge (bridge mode) have: Same MAC Address

```
[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 1000
    link/ether 52:54:00:6b:35:4d brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.2/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]#
```

```
[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 1000
    link/ether 52:54:00:6b:35:4d brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.3/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# ping 192.168.140.2
PING 192.168.140.2 (192.168.140.2) 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
From 192.168.140.3 icmp_seq=9 Destination Host Unreachable
From 192.168.140.3 icmp_seq=10 Destination Host Unreachable
From 192.168.140.3 icmp_seq=11 Destination Host Unreachable
From 192.168.140.3 icmp_seq=12 Destination Host Unreachable
^C
--- 192.168.140.2 ping statistics ---
12 packets transmitted, 0 received, +12 errors, 100% packet loss, time 11004ms
pipe 4
[root@localhost ~]#
```



As we can see above in this scenario with same MAC and VM's connected to same Bridge(Bridge Mode) we have the ping request failing this is because when it sends out an ARP request the bridge receives the reply from VM2 with the MAC address which will be same as VM1 as well since we configured in that way. Therefore, the L2 bridge since it works on MAC addresses, it will again send back the destined arp reply to VM2 since the ARP tables get updated for the same MAC address to the port connected with VM2 and the reply never reaches VM1 due to which ping fails and we continuously see ARP requests at the bridge interface connected to VM1 to be coming from the wireshark screenshot above. Due to this the ping fails.

[7.1b] Same IP Addresses

```
[root@localhost ~]#
[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 1000
    link/ether 52:54:00:6b:35:4d brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.2/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]#
```

```
[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 1000
    link/ether 52:54:00:6a:35:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.2/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# ping 192.168.140.2
PING 192.168.140.2 (192.168.140.2) 56(84) bytes of data.
64 bytes from 192.168.140.2: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 192.168.140.2: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 192.168.140.2: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 192.168.140.2: icmp_seq=4 ttl=64 time=0.067 ms
64 bytes from 192.168.140.2: icmp_seq=5 ttl=64 time=0.082 ms
64 bytes from 192.168.140.2: icmp_seq=6 ttl=64 time=0.071 ms
64 bytes from 192.168.140.2: icmp_seq=7 ttl=64 time=0.069 ms
64 bytes from 192.168.140.2: icmp_seq=8 ttl=64 time=0.068 ms
64 bytes from 192.168.140.2: icmp_seq=9 ttl=64 time=0.080 ms
64 bytes from 192.168.140.2: icmp_seq=10 ttl=64 time=0.075 ms
64 bytes from 192.168.140.2: icmp_seq=11 ttl=64 time=0.070 ms
64 bytes from 192.168.140.2: icmp_seq=12 ttl=64 time=0.071 ms
64 bytes from 192.168.140.2: icmp_seq=13 ttl=64 time=0.069 ms
^C
--- 192.168.140.2 ping statistics ---
13 packets transmitted, 13 received, 0% packet loss, time 12000ms
rtt min/avg/max/mdev = 0.067/0.071/0.082/0.011 ms
[root@localhost ~]# _
```

Now in this scenario where we have 2 VM's with same IP address but different MAC address, since the ping is the same IP as ours the packet would not leave the VM's Interface and go the bridge since our route will have our own interface for a packet to our IP address and therefore we are just pinging ourselves due to which we have the ping successful and while seen on wireshark we have no captures at the Bridge's interfaces.

[7.2a]

The setup for this scenario is both VM's Connected to different Bridges in bridge mode with Same MAC address.

Bridges in the screenshot are "q7" and "q72" with interfaces vnet17 and vnet18.

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM71
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet17     bridge     q7_network virtio      52:54:00:dd:73:5f

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM72
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet18     bridge     q7_network2 virtio      52:54:00:dd:73:5f

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ brctl show
bridge name     bridge id               STP enabled     interfaces
nat_ansible             8000.fe5400b8645f       yes             vnet15
                                                        vnet16
q7              8000.fe5400dd735f       no              vnet17
q72             8000.fe5400dd735f       no              vnet18
sw1             8000.fe54004b735f       no              vnet10
                                                        vnet11
                                                        vnet12
                                                        vnet13
                                                        vnet14
                                                        vnet3
                                                        vnet9
sw2             8000.fe54009f186c       no              vnet5
                                                        vnet7
sw3             8000.fe5400262ea4       no              vnet0
                                                        vnet8
virbr0          8000.fe5400351f3c       yes             vnet1
                                                        vnet2
                                                        vnet4
                                                        vnet6

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

```
[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 1
    link/ether 52:54:00:dd:73:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.2/24 scope global eth0
       valid_lft forever preferred_lft forever
[root@localhost ~]# ping 192.168.140.3
PING 192.168.140.3 (192.168.140.3) 56(84) bytes of data.
From 192.168.140.2 icmp_seq=1 Destination Host Unreachable
From 192.168.140.2 icmp_seq=2 Destination Host Unreachable
From 192.168.140.2 icmp_seq=3 Destination Host Unreachable
From 192.168.140.2 icmp_seq=4 Destination Host Unreachable
^C
--- 192.168.140.3 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3000ms
pipe 4
[root@localhost ~]# _
```

```
root@localhost ~]#
root@localhost ~]# ip addr
: 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
: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen
    link/ether 52:54:00:dd:73:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.3/24 scope global eth0
       valid_lft forever preferred_lft forever
root@localhost ~]#
```

So here there is actually no connectivity between the VM's at all. They are just independent of themselves. Though we just tried pinging the other VM but obviously since there is no connectivity between the bridges and the VM with the others. The packet wouldn't find the destination. This setup probably doesn't do anything and just has a bridge attached to themselves and only ping to themselves would work.

[7.2b]

In this setup we will be having the same bridges as above instead we will have same IP addresses and not MAC addresses.

```
[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 10
    link/ether 52:54:00:fe:65:3d brd ff:ff:ff:ff:ff:ff
    inet 192.168.140.3/24 scope global eth0
       valid_lft forever preferred_lft forever
[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=64 time=0.075 ms
64 bytes from 192.168.140.3: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 192.168.140.3: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 192.168.140.3: icmp_seq=4 ttl=64 time=0.062 ms
^C
--- 192.168.140.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.062/0.068/0.075/0.004 ms
[root@localhost ~]#
```

```
[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
    link/ether 52:54:00:dd:73:5f 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 now that we have same IP addresses but different MAC's but it still the same case as above there is no connectivity at all so the ping which is successful is not actually reaching the other VM but it's pinging itself similar to the case we have seen in (7.1b) as there is no wireshark capture seen at the bridge interface. This setup basically doesn't help with no connections at all while this setup usually ensures network isolation.

[7.3a]

In this scenario we have 2 bridges connected to 2 VM's but in routed mode which could be seen below with network type and IP addresses with respect to their networks.

```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM71
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet17     network    q7_network virtio      52:54:00:df:2b:ee

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domiflist VM72
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet18     network    q7_network2 virtio      52:54:00:ff:78:5e

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domifaddr VM71
 Name       MAC address          Protocol     Address
-------------------------------------------------------------------------------
 vnet17     52:54:00:df:2b:ee    ipv4         192.168.150.93/24

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ virsh domifaddr VM72
 Name       MAC address          Protocol     Address
-------------------------------------------------------------------------------
 vnet18     52:54:00:ff:78:5e    ipv4         192.168.160.201/24

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

Now we are configuring same MAC addresses and pinging the other VM as shown below:

```
[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 q
    link/ether 52:54:00:ff:78:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.160.201/24 brd 192.168.160.255 scope global dynamic eth0
       valid_lft 2370sec preferred_lft 2370sec
[root@localhost ~]# ping 192.168.150.93
PING 192.168.150.93 (192.168.150.93) 56(84) bytes of data.
64 bytes from 192.168.150.93: icmp_seq=1 ttl=63 time=1.43 ms
64 bytes from 192.168.150.93: icmp_seq=2 ttl=63 time=0.613 ms
64 bytes from 192.168.150.93: icmp_seq=3 ttl=63 time=3.60 ms
64 bytes from 192.168.150.93: icmp_seq=4 ttl=63 time=0.766 ms
64 bytes from 192.168.150.93: icmp_seq=5 ttl=63 time=13.5 ms
^C
--- 192.168.150.93 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.613/3.990/13.536/4.891 ms
[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:ff:78:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.150.93/24 brd 192.168.150.255 scope global dynamic eth0
       valid_lft 3444sec preferred_lft 3444sec
[root@localhost ~]#
```

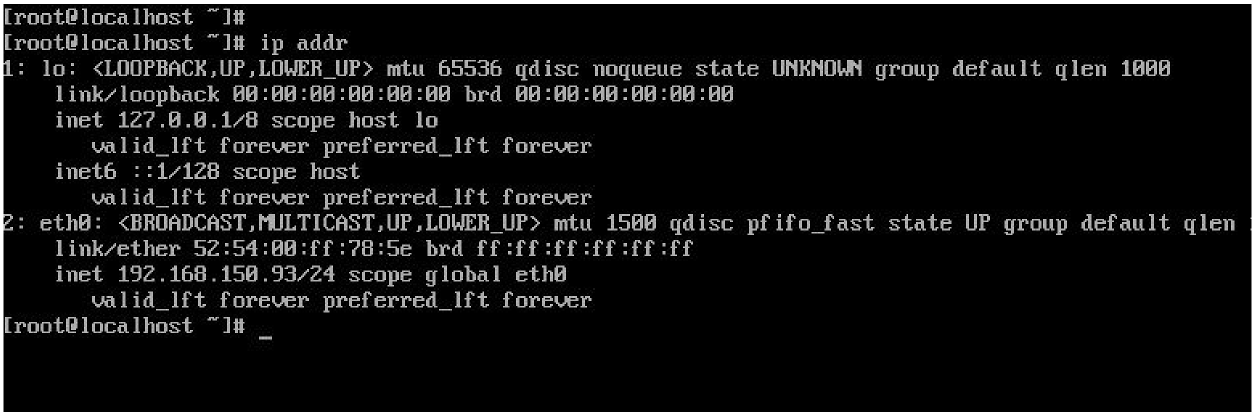
As we can see the Ping is successful because since the bridge is in routed mode and both of them are in different networks. It forwards packet first to the default gateway of the network which is the L3 interface at the bridge. Since the destination IP is in another network the routing table at the Hypervisor has the following entries(screenshot below) where we have routes defined to VM2 network through q72 which is connected to it. So any packets which wants to go the VM2 will be forwarded to q72 bridge and therefore the ping is successful which could be also supported by the fact that the TTL is seen to 63. In the below screenshot our networks are 192.168.150.0 and 192.168.160.0 with bridges q7 and q72.

```
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
[ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ ip route
default via 192.168.124.1 dev ens3  proto static  metric 100
default via 192.168.123.1 dev ens5  proto static  metric 101
169.254.0.0/16 dev ens3  scope link  metric 1000
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1
192.168.123.0/24 dev ens5  proto kernel  scope link  src 192.168.123.185  metric 100
192.168.124.0/24 dev ens3  proto kernel  scope link  src 192.168.124.5  metric 100
192.168.132.0/24 dev nat_ansible  proto kernel  scope link  src 192.168.132.1
192.168.150.0/24 dev q7  proto kernel  scope link  src 192.168.150.1
192.168.160.0/24 dev q72  proto kernel  scope link  src 192.168.160.1
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$
```

[7.3b]

Now in this set up same as above with the bridges but we will be configuring same IP addresses.

```
[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
    link/ether 52:54:00:df:2b:ee brd ff:ff:ff:ff:ff:ff
    inet 192.168.150.93/24 brd 192.168.150.255 scope global dynamic eth0
       valid_lft 2598sec preferred_lft 2598sec
[root@localhost ~]# ping 192.168.150.93
PING 192.168.150.93 (192.168.150.93) 56(84) bytes of data.
64 bytes from 192.168.150.93: icmp_seq=1 ttl=64 time=0.332 ms
64 bytes from 192.168.150.93: icmp_seq=2 ttl=64 time=0.065 ms
64 bytes from 192.168.150.93: icmp_seq=3 ttl=64 time=0.075 ms
64 bytes from 192.168.150.93: icmp_seq=4 ttl=64 time=0.099 ms
64 bytes from 192.168.150.93: icmp_seq=5 ttl=64 time=0.091 ms
64 bytes from 192.168.150.93: icmp_seq=6 ttl=64 time=0.065 ms
64 bytes from 192.168.150.93: icmp_seq=7 ttl=64 time=0.072 ms
^C
--- 192.168.150.93 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6000ms
rtt min/avg/max/mdev = 0.065/0.114/0.332/0.089 ms
[root@localhost ~]# _
```



As we can see above the VM's have same IP addresses but different MAC addresses but having same IP addresses and trying to ping the same IP is nothing but pinging ourselves. So though ping is successful it is not actually not leaving the VM's Interface which could be also supported by the fact that the TTL is 64 which was seen to be 63 in the previous case where the packed actually goes out of VM's interface. So when we tried capturing packets at the bridge interface we have seen no icmp packets reaching the bridge.