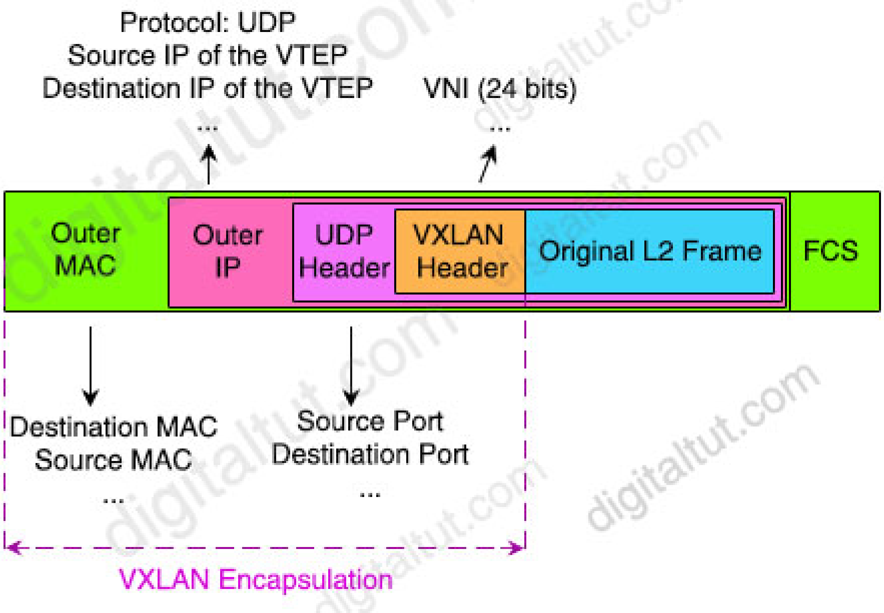
Overlay Network and VXLAN

Tang Peng: 517020910038

An overlay network is a telecommunications network that is built on the top of another network and is supported by its infrastructure. An overlay network decouples network services from the underlying infrastructure by encapsulating one packet inside of another packet. After the encapsulated packet has been forwarded to the endpoint, it is de-encapsulated.

VXLAN is an overlay technology and uses tunneling to stretch Layer 2 connection over an underlaying Layer 3 network. Specially, it encapsulates packets by adding a VXLAN header to the original Layer 2 frame and then placing this header in a UDP-IP packet. The following figure shows the encapsulated packet. Normally, VXLAN adds 50 bytes of additional header information to the original Ethernet frame, including 14 bytes outer MAC header, 20 bytes outer IP header, 8 bytes outer UDP header, and 8 bytes VXLAN header.



The encapsulation and decapsulation of packets is done by the VXLAN tunnel endpoint(VTEP).

1 Establish the Connection Between Two Machines

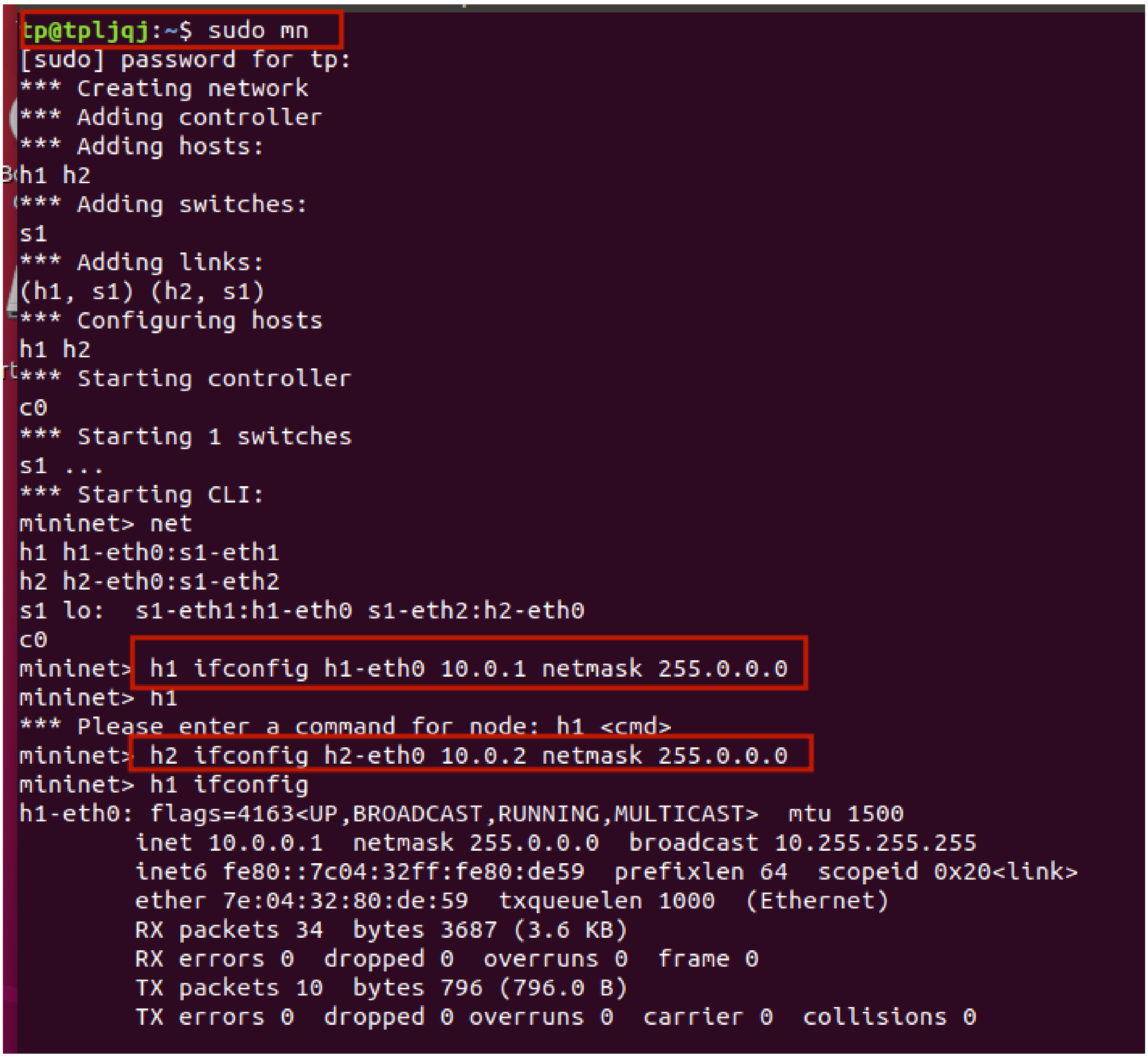
According to the instructions, we can set build the connection two VMs by using the VXLAN.

- On VM1, we run the following commands.

```
1  tp@tpljqj:~$ sudo mn
2  mininet> h1 ifconfig h1-eth0 10.0.0.1 netmask 255.0.0.0
3  mininet> h2 ifconfig h2-eth0 10.0.0.2 netmask 255.0.0.0
4  tp@tpljqj:~$ sudo ifconfig s1 10.0.0.101/8 up
```

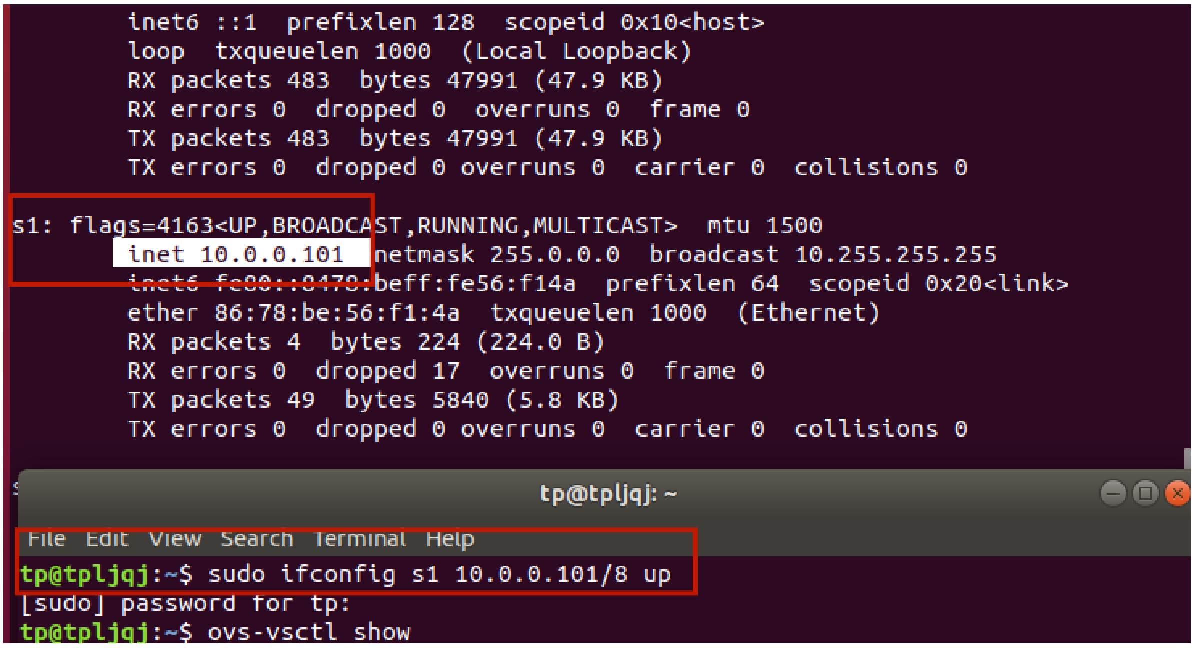
The line 1 is used to start mininet.

The line 2 and line 3 are used to set IPs for both hosts.



```
tp@tpljqj:~$ sudo mn
[sudo] password for tp:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> h1 ifconfig h1-eth0 10.0.1 netmask 255.0.0.0
mininet> h1
*** Please enter a command for node: h1 <cmd>
mininet> h2 ifconfig h2-eth0 10.0.2 netmask 255.0.0.0
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::7c04:32ff:fe80:de59  prefixlen 64  scopeid 0x20<link>
        ether 7e:04:32:80:de:59  txqueuelen 1000  (Ethernet)
        RX packets 34  bytes 3687 (3.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 796 (796.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The line 4 is used to set IP for the switch.



```
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 483  bytes 47991 (47.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 483  bytes 47991 (47.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.101  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::8478:beff:fe56:f14a  prefixlen 64  scopeid 0x20<link>
        ether 86:78:be:56:f1:4a  txqueuelen 1000  (Ethernet)
        RX packets 4  bytes 224 (224.0 B)
        RX errors 0  dropped 17  overruns 0  frame 0
        TX packets 49  bytes 5840 (5.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

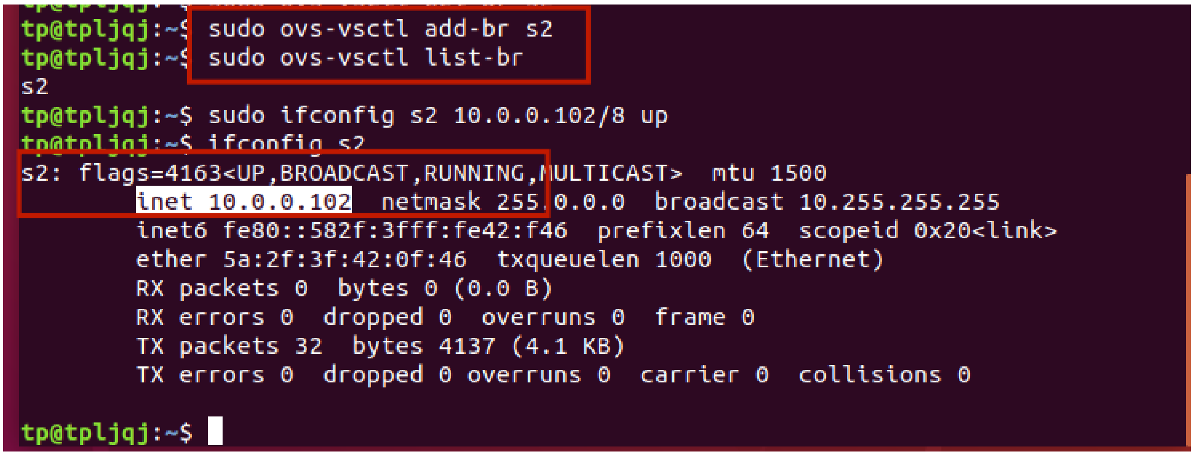
```
tp@tpljqj:~$ sudo ifconfig s1 10.0.0.101/8 up
[sudo] password for tp:
tp@tpljqj:~$ ovs-vsctl show
```

- On VM2, we run the following commands.

```
1  tp@tpljqj:~$ sudo ovs-vsctl add-br s2
2  tp@tpljqj:~$ sudo ifconfig s2 10.0.0.102/8 up
```

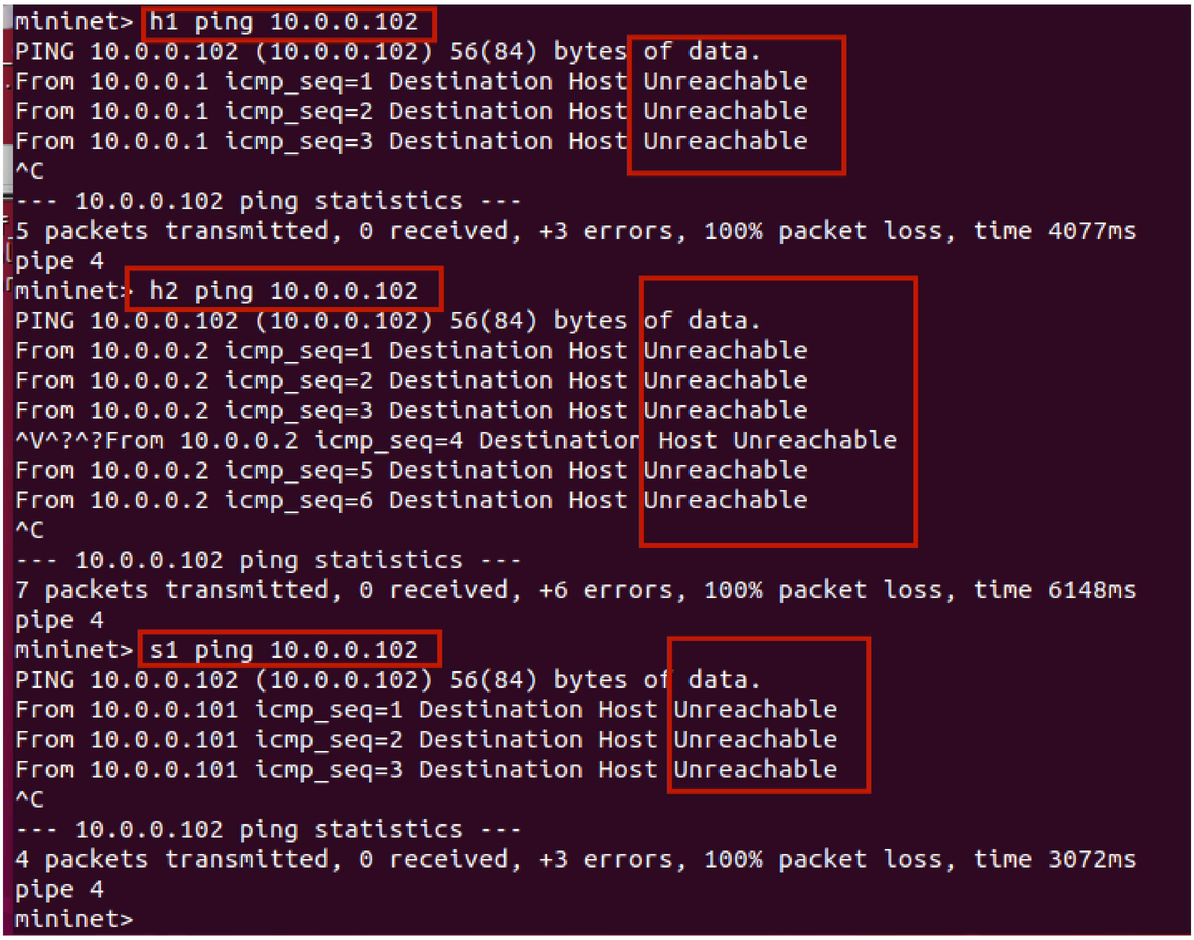
Line 1 is used to create a new bridge called s2.

Line 2 is used to set IP for s2.



```
tp@tpljqj:~$ sudo ovs-vsctl add-br s2
tp@tpljqj:~$ sudo ovs-vsctl list-br
s2
tp@tpljqj:~$ sudo ifconfig s2 10.0.0.102/8 up
tp@tpljqj:~$ ifconfig s2
s2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.102  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::582f:3fff:fe42:f46  prefixlen 64  scopeid 0x20<link>
        ether 5a:2f:3f:42:0f:46  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32  bytes 4137 (4.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tp@tpljqj:~$
```

- Now, we can not ping 10.0.0.102 from 10.0.0.101/10.0.0.1/10.0.0.2.



```
mininet> h1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.102 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4077ms
pipe 4
mininet> h2 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable
From 10.0.0.2 icmp_seq=2 Destination Host Unreachable
From 10.0.0.2 icmp_seq=3 Destination Host Unreachable
^V^?^?From 10.0.0.2 icmp_seq=4 Destination Host Unreachable
From 10.0.0.2 icmp_seq=5 Destination Host Unreachable
From 10.0.0.2 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.0.102 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6148ms
pipe 4
mininet> s1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
From 10.0.0.101 icmp_seq=1 Destination Host Unreachable
From 10.0.0.101 icmp_seq=2 Destination Host Unreachable
From 10.0.0.101 icmp_seq=3 Destination Host Unreachable
^C
--- 10.0.0.102 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3072ms
pipe 4
mininet>
```

- The we run the following commands on VM1.

```
1  tp@tpljqj:~$ sudo ovs-vsctl add-br br1
2  tp@tpljqj:~$ sudo ifconfig enp0s8 0 up
3  tp@tpljqj:~$ sudo ovs-vsctl add-port br1 enp0s8
4  tp@tpljqj:~$ sudo ifconfig br1 192.168.56.102/24 up
```

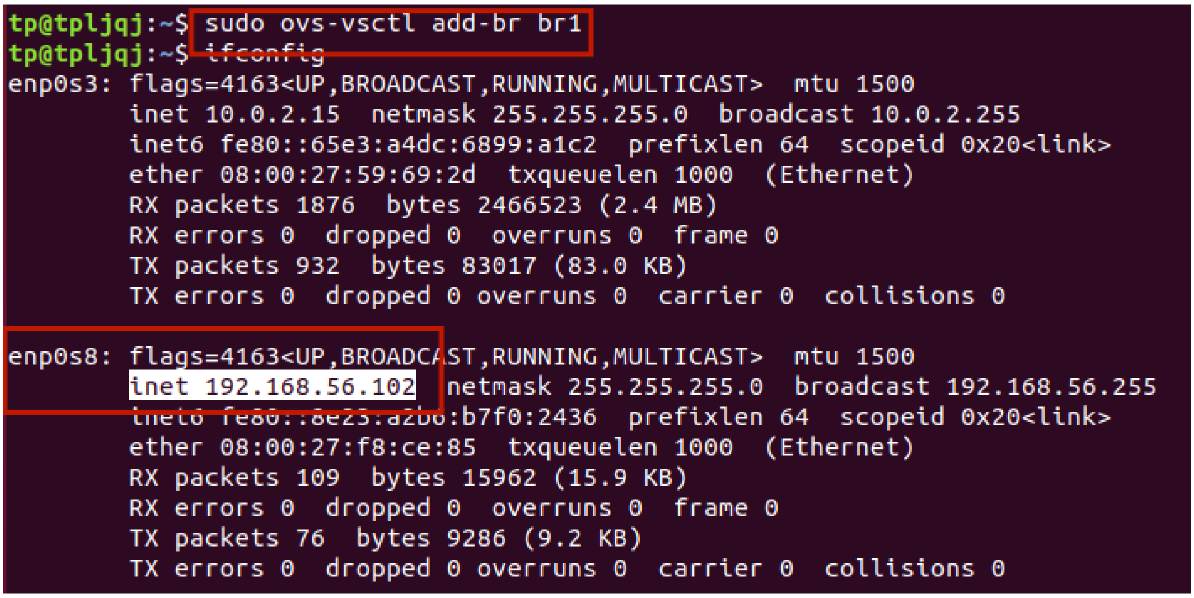
Line 1 is used to create a new bridge called br1.

Line 2 is used to remove the IP of the enp0s8.

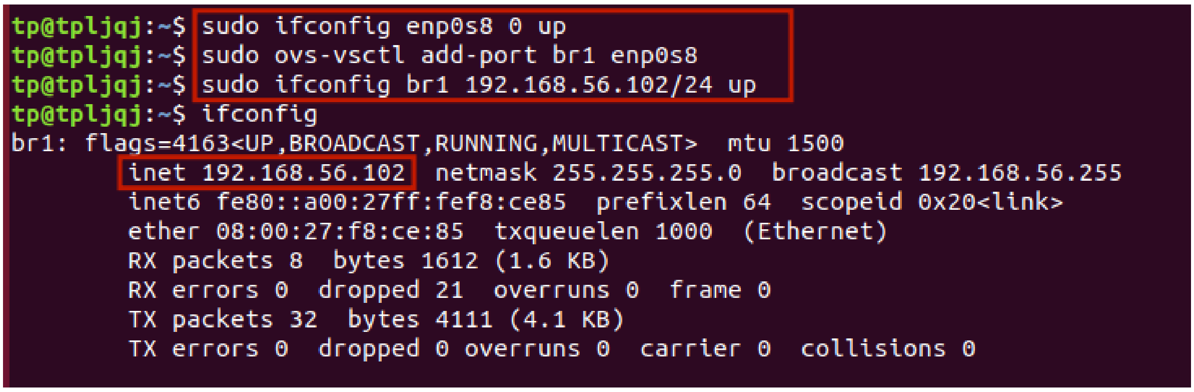
Line 3 is used to set the enp0s8 as the port of br1.

Line 4 is used to assign the IP of enp0s8 to br1.

The following figures shows the results of above commands.



```
tp@tpljqj:~$ sudo ovs-vsctl add-br br1
tp@tpljqj:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::65e3:a4dc:6899:a1c2  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:59:69:2d  txqueuelen 1000  (Ethernet)
        RX packets 1876  bytes 2466523 (2.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 932  bytes 83017 (83.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::8e23:a2b6:b7f0:2436  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:f8:ce:85  txqueuelen 1000  (Ethernet)
        RX packets 109  bytes 15962 (15.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 76  bytes 9286 (9.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

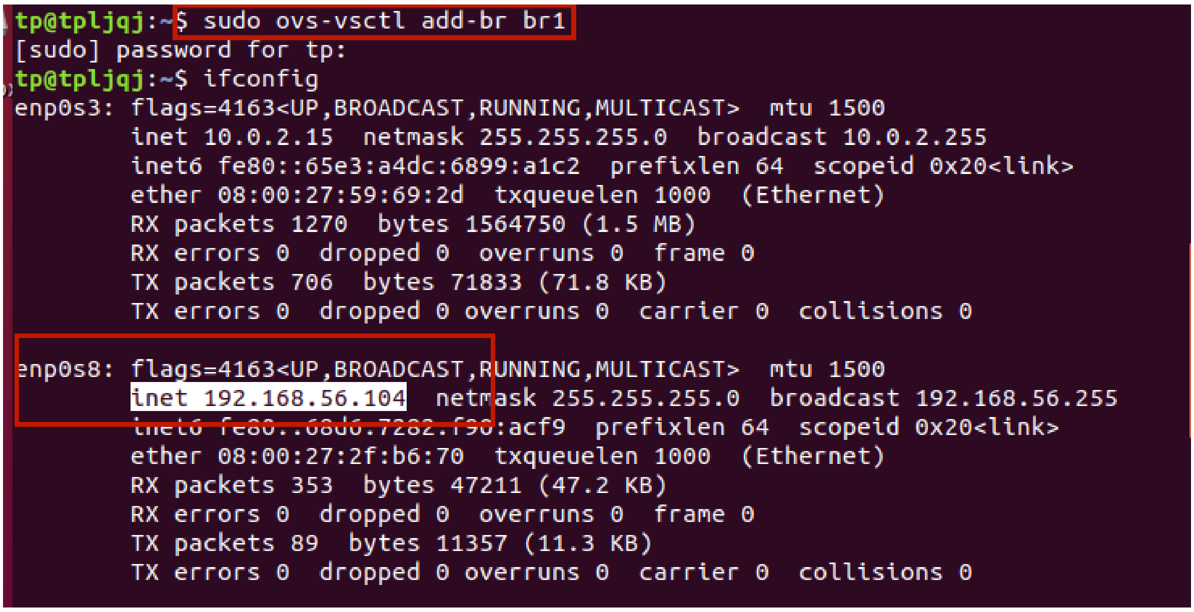


```
tp@tpljqj:~$ sudo ifconfig enp0s8 0 up
tp@tpljqj:~$ sudo ovs-vsctl add-port br1 enp0s8
tp@tpljqj:~$ sudo ifconfig br1 192.168.56.102/24 up
tp@tpljqj:~$ ifconfig
br1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fef8:ce85  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:f8:ce:85  txqueuelen 1000  (Ethernet)
        RX packets 8  bytes 1612 (1.6 KB)
        RX errors 0  dropped 21  overruns 0  frame 0
        TX packets 32  bytes 4111 (4.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

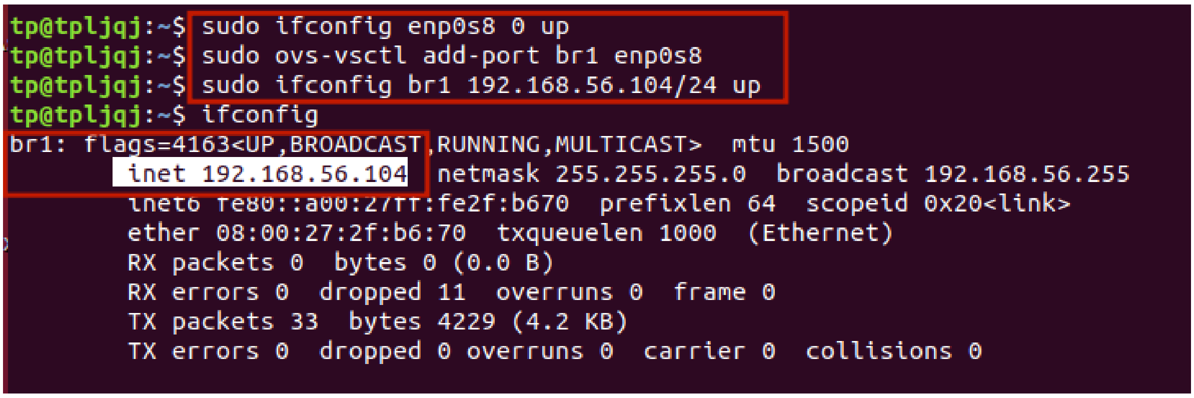
- The we run the following commands on VM2.

```
tp@tpljqj:~$ sudo ovs-vsctl add-br br1
tp@tpljqj:~$ sudo ifconfig enp0s8 0 up
tp@tpljqj:~$ sudo ovs-vsctl add-port br1 enp0s8
tp@tpljqj:~$ sudo ifconfig br1 192.168.56.104/24 up
```

These commands do the same things as these on VM1. The only difference is the IP of enp0s8.



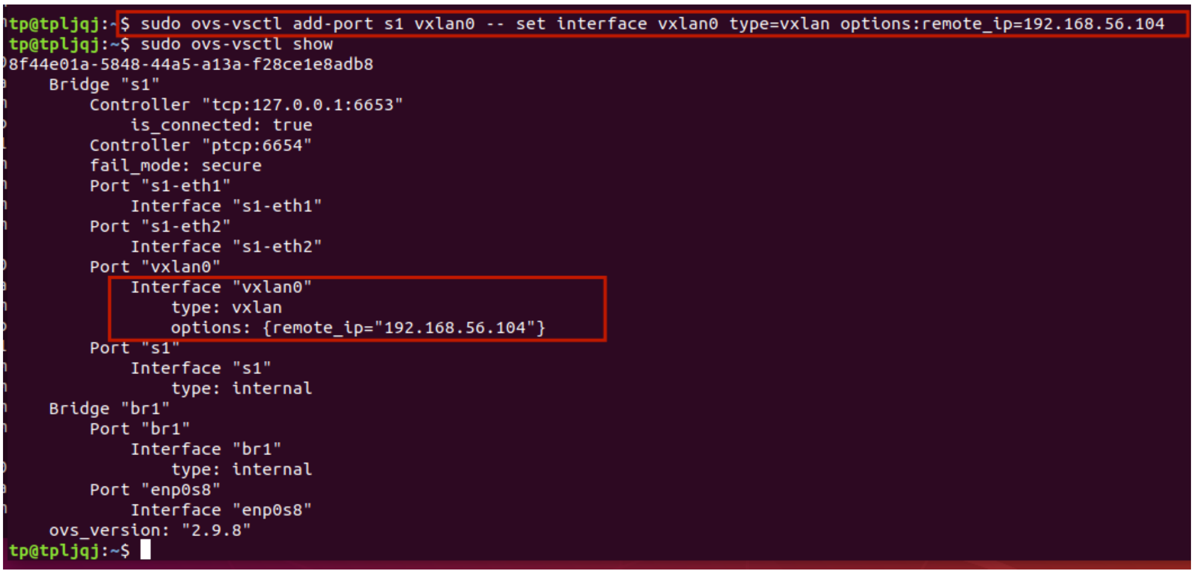
```
tp@tpljqj:~$ sudo ovs-vsctl add-br br1
[sudo] password for tp:
tp@tpljqj:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::65e3:a4dc:6899:a1c2  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:59:69:2d  txqueuelen 1000  (Ethernet)
        RX packets 1270  bytes 1564750 (1.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 706  bytes 71833 (71.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.104  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::68d6:7282:f90:acf9  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2f:b6:70  txqueuelen 1000  (Ethernet)
        RX packets 353  bytes 47211 (47.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 89  bytes 11357 (11.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```



```
tp@tpljqj:~$ sudo ifconfig enp0s8 0 up
tp@tpljqj:~$ sudo ovs-vsctl add-port br1 enp0s8
tp@tpljqj:~$ sudo ifconfig br1 192.168.56.104/24 up
tp@tpljqj:~$ ifconfig
br1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.104  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe2f:b670  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:2f:b6:70  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 11  overruns 0  frame 0
        TX packets 33  bytes 4229 (4.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- We run the following command on VM1 to implement the VXLAN.

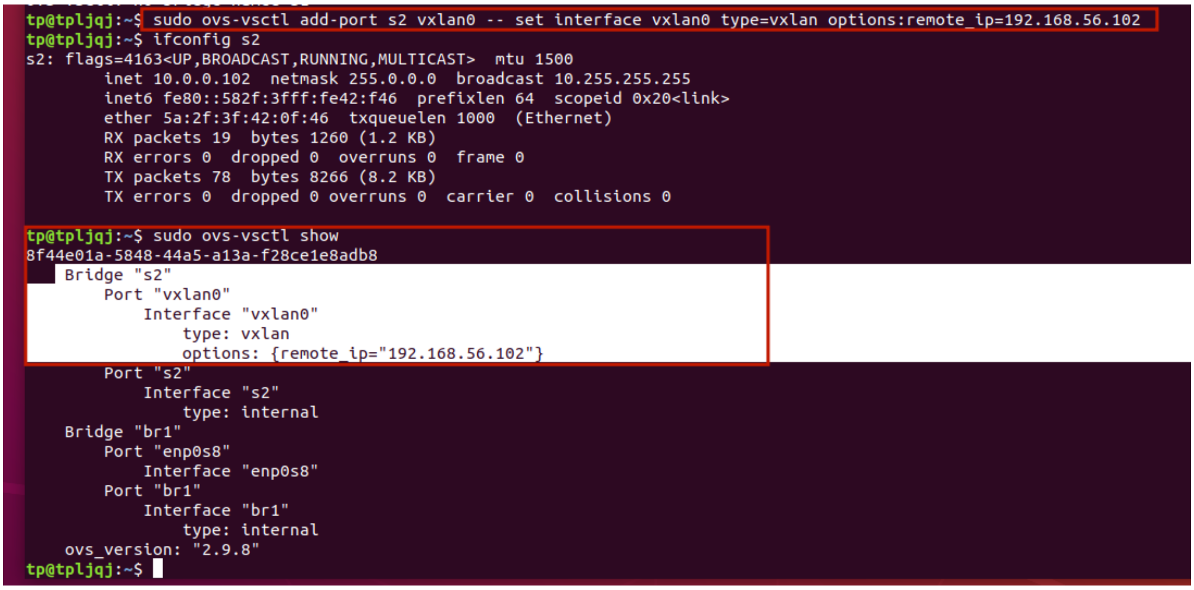
```
tp@tpljqj:~$ sudo ovs-vsctl add-port s1 vxlan0 -- set interface vxlan0
type=vxlan options:remote_ip=192.168.56.104
```



```
tp@tpljqj:~$ sudo ovs-vsctl add-port s1 vxlan0 -- set interface vxlan0 type=vxlan options:remote_ip=192.168.56.104
tp@tpljqj:~$ sudo ovs-vsctl show
8f44e01a-5848-44a5-a13a-f28ce1e8adb8
    Bridge "s1"
        Controller "tcp:127.0.0.1:6653"
            is_connected: true
        Controller "ptcp:6654"
        fail_mode: secure
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "vxlan0"
            Interface "vxlan0"
                type: vxlan
                options: {remote_ip="192.168.56.104"}
        Port "s1"
            Interface "s1"
                type: internal
    Bridge "br1"
        Port "br1"
            Interface "br1"
                type: internal
        Port "enp0s8"
            Interface "enp0s8"
    ovs_version: "2.9.8"
tp@tpljqj:~$
```

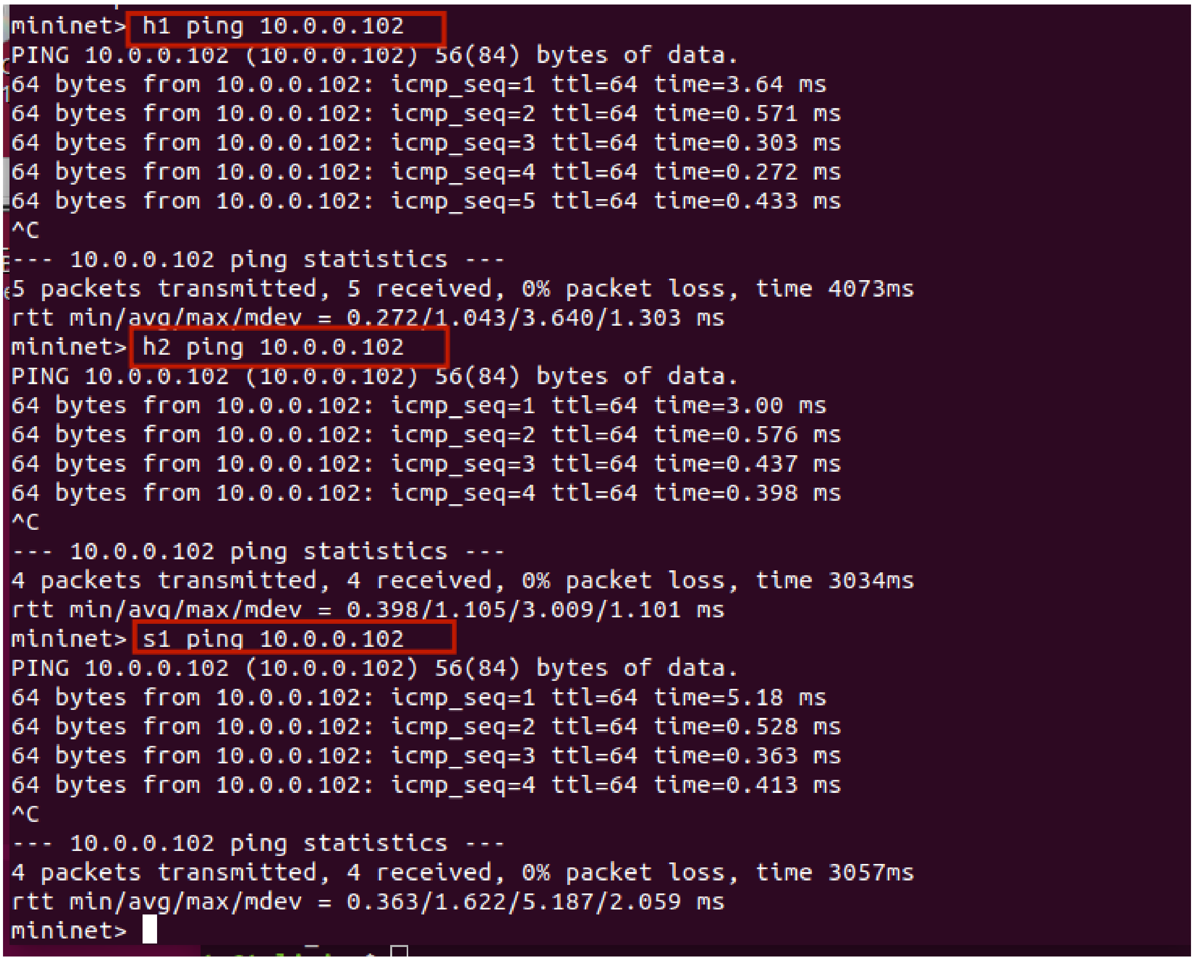
- We run the following command on VM2 to implement the VXLAN.

```
tp@tpljqj:~$ sudo ovs-vsctl add-port s2 vxlan0 -- set interface vxlan0
type=vxlan options:remote_ip=192.168.56.102
```



```
tp@tpljqj:~$ sudo ovs-vsctl add-port s2 vxlan0 -- set interface vxlan0 type=vxlan options:remote_ip=192.168.56.102
tp@tpljqj:~$ ifconfig s2
s2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.102  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::582f:3fff:fe42:f46  prefixlen 64  scopeid 0x20<link>
        ether 5a:2f:3f:42:0f:46  txqueuelen 1000  (Ethernet)
        RX packets 19  bytes 1260 (1.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 78  bytes 8266 (8.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tp@tpljqj:~$ sudo ovs-vsctl show
8f44e01a-5848-44a5-a13a-f28ce1e8adb8
    Bridge "s2"
        Port "vxlan0"
            Interface "vxlan0"
                type: vxlan
                options: {remote_ip="192.168.56.102"}
        Port "s2"
            Interface "s2"
                type: internal
    Bridge "br1"
        Port "enp0s8"
            Interface "enp0s8"
        Port "br1"
            Interface "br1"
                type: internal
    ovs_version: "2.9.8"
tp@tpljqj:~$
```

After doing above things, we can ping 10.0.0.102 from 10.0.0.101/10.0.0.1/10.0.0.2.

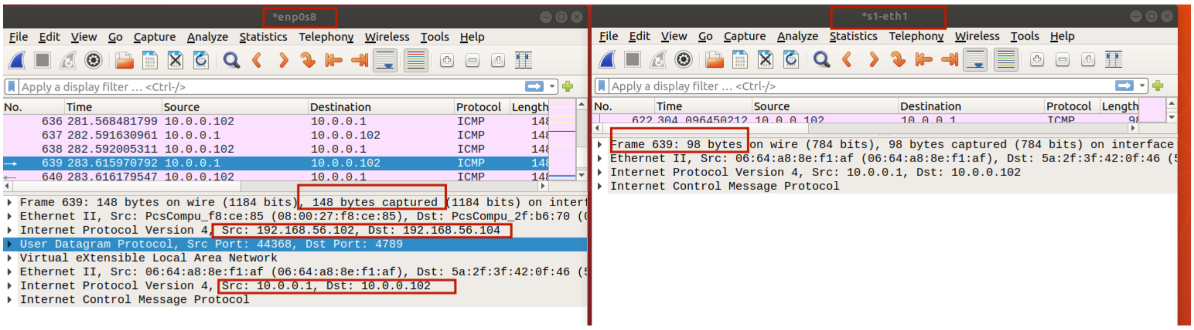


```
mininet> h1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=3.64 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.571 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.303 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.272 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=0.433 ms
^C
--- 10.0.0.102 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4073ms
rtt min/avg/max/mdev = 0.272/1.043/3.640/1.303 ms
mininet> h2 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=3.00 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.576 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.437 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.398 ms
^C
--- 10.0.0.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.398/1.105/3.009/1.101 ms
mininet> s1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=5.18 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.528 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.363 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.413 ms
^C
--- 10.0.0.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3057ms
rtt min/avg/max/mdev = 0.363/1.622/5.187/2.059 ms
mininet> 
```

2 Exercise 1

When pinging s2 from h1 and using Wireshark to monitor the interfaces s1 and enp0s8, we find

- Packets   transmitted through interfaces s1 will be transmitted through enp0s8. The following figure shows that the packets of interfaces s1 and enp0s8. From the packets, we can see that the packets in enp0s8 has 4 headers more than packets in s1 and the size of the packets in enp0s8 is 50 bytes bigger than this in s1, which means the packets in interfaces s1 is encapsulated by enp0s8.
- What's more, the IP addresses of the outer IP Layer are different from the interior IP Layer in enp0s8. The interior IP addresses is the addresses of h1 and s2, while the outer IP addresses is the address of two br1s in two VMs.

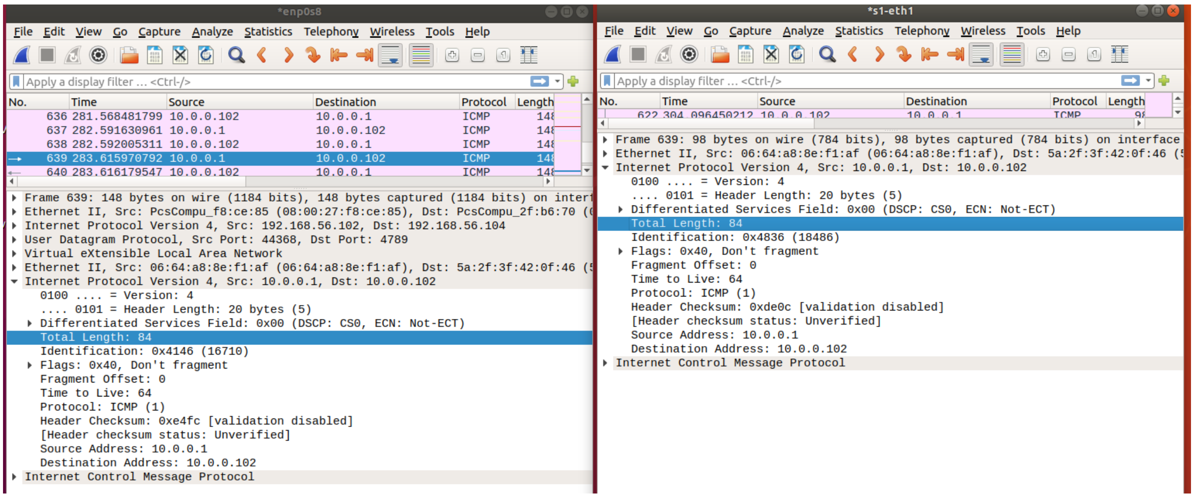

*enp0s8
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time            Source       Destination   Protocol  Length
  636 281.568481799 10.0.0.102    10.0.0.1      ICMP      148
  637 282.591630961 10.0.0.1      10.0.0.102    ICMP      148
  638 282.592005311 10.0.0.102    10.0.0.1      ICMP      148
  639 283.615970792 10.0.0.1      10.0.0.102    ICMP      148
  640 283.616179547 10.0.0.102    10.0.0.1      ICMP      148

Frame 639: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interf
Ethernet II, Src: PcsCompu_f8:ce:85 (08:00:27:f8:ce:85), Dst: PcsCompu_2f:b6:70 ((
Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.104
User Datagram Protocol, Src Port: 44368, Dst Port: 4789
Virtual eXtensible Local Area Network
Ethernet II, Src: 06:64:a8:8e:f1:af (06:64:a8:8e:f1:af), Dst: 5a:2f:3f:42:0f:46 (5
Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.102
Internet Control Message Protocol
```

*s1-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time            Source       Destination   Protocol  Length
  622 304.096450212 10.0.0.102    10.0.0.1      ICMP      98

Frame 639: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface
Ethernet II, Src: 06:64:a8:8e:f1:af (06:64:a8:8e:f1:af), Dst: 5a:2f:3f:42:0f:46 (5
Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.102
Internet Control Message Protocol
```

- And when checking the inter IP Layer in enp0s8 and the IP Layer in s1, we can see the size is same.



```
*enp0s8
No.  Time            Source       Destination  Protocol Length
 636 281.568481799 10.0.0.102    10.0.0.1     ICMP     148
 637 282.591630961 10.0.0.1      10.0.0.102   ICMP     148
 638 282.592005311 10.0.0.102    10.0.0.1     ICMP     148
 639 283.615970792 10.0.0.1      10.0.0.102   ICMP     148
 640 283.616179547 10.0.0.102    10.0.0.1     ICMP     148

▸ Frame 639: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interf
▸ Ethernet II, Src: PcsCompu_f8:ce:85 (08:00:27:f8:ce:85), Dst: PcsCompu_2f:b6:70 ((
▸ Internet Protocol Version 4, Src: 192.168.56.102, Dst: 192.168.56.104
▸ User Datagram Protocol, Src Port: 44368, Dst Port: 4789
▸ Virtual eXtensible Local Area Network
▸ Ethernet II, Src: 06:64:a8:8e:f1:af (06:64:a8:8e:f1:af), Dst: 5a:2f:3f:42:0f:46 (5
▾ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.102
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x4146 (16710)
  ▸ Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xe4fc [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.0.1
    Destination Address: 10.0.0.102
▸ Internet Control Message Protocol
```

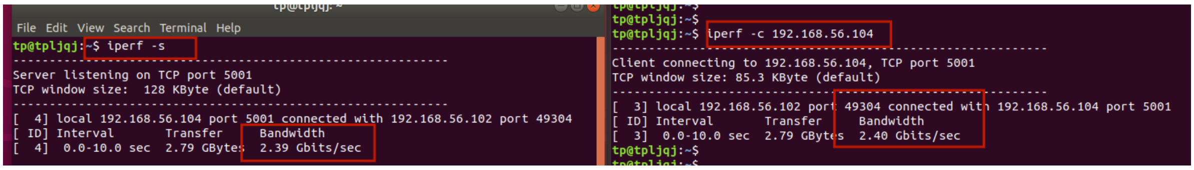
```
*s1-eth1
No.  Time            Source       Destination  Protocol Length
 622 304.096450212 10.0.0.102    10.0.0.1     ICMP     98

▸ Frame 639: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface
▸ Ethernet II, Src: 06:64:a8:8e:f1:af (06:64:a8:8e:f1:af), Dst: 5a:2f:3f:42:0f:46 (5
▾ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.102
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x4836 (18486)
  ▸ Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xde0c [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.0.1
    Destination Address: 10.0.0.102
▸ Internet Control Message Protocol
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

- In summary, when a packet is transmitted from h1 on VM1 to s2 on VM2, it will be forwarded by s1 at first, and then it will be encapsulated by enp0s8 on VM1 and then be transmitted to another enp0s8 on VM2. Then, the enp0s8 on VM2 will decapsulated the packet and forward it to s2.

3 Exercise 2

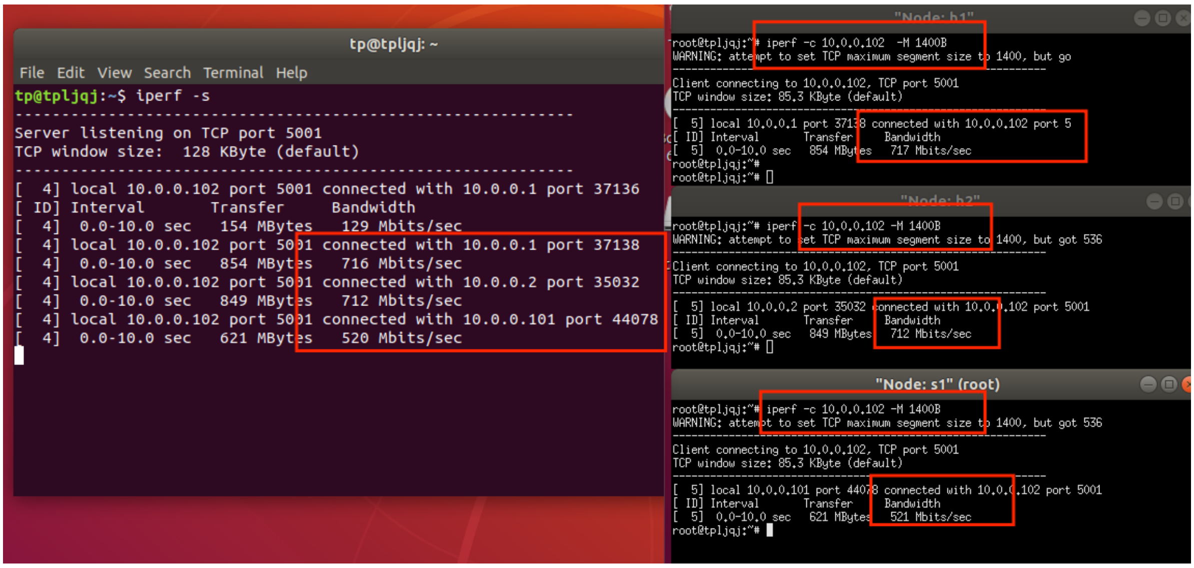
- When testing the bandwidth between 192.168.56.127 and 192.168.56.128 by using iperf, we get followings result.



```
tp@tpljqj:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.56.104 port 5001 connected with 192.168.56.102 port 49304
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  2.79 GBytes  2.39 Gbits/sec
```

```
tp@tpljqj:~$
tp@tpljqj:~$ iperf -c 192.168.56.104
------------------------------------------------------------
Client connecting to 192.168.56.104, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.56.102 port 49304 connected with 192.168.56.104 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  2.79 GBytes  2.40 Gbits/sec
tp@tpljqj:~$
```

- When testing the bandwidth between 10.0.0.1/10.0.0.2/10.0.0.101 and 10.0.0.102, we get following results. And because the size of encapsulated packets is 50 bytes bigger than original packet, we need to set a smaller MTU size for original packet. Otherwise, the encapsulated packets will be dropped because their size is bigger than the MTU of the network.



```
tp@tpljqj:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.102 port 5001 connected with 10.0.0.1 port 37136
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   154 MBytes   129 Mbits/sec
[  4] local 10.0.0.102 port 5001 connected with 10.0.0.1 port 37138
[  4]  0.0-10.0 sec   854 MBytes   716 Mbits/sec
[  4] local 10.0.0.102 port 5001 connected with 10.0.0.2 port 35032
[  4]  0.0-10.0 sec   849 MBytes   712 Mbits/sec
[  4] local 10.0.0.102 port 5001 connected with 10.0.0.101 port 44078
[  4]  0.0-10.0 sec   621 MBytes   520 Mbits/sec
```

"Node: h1"

```
root@tpljqj:~# iperf -c 10.0.0.102  -M 1400B
WARNING: attempt to set TCP maximum segment size to 1400, but go
------------------------------------------------------------
Client connecting to 10.0.0.102, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 37138 connected with 10.0.0.102 port 5
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   854 MBytes   717 Mbits/sec
root@tpljqj:~#
root@tpljqj:~# 
```

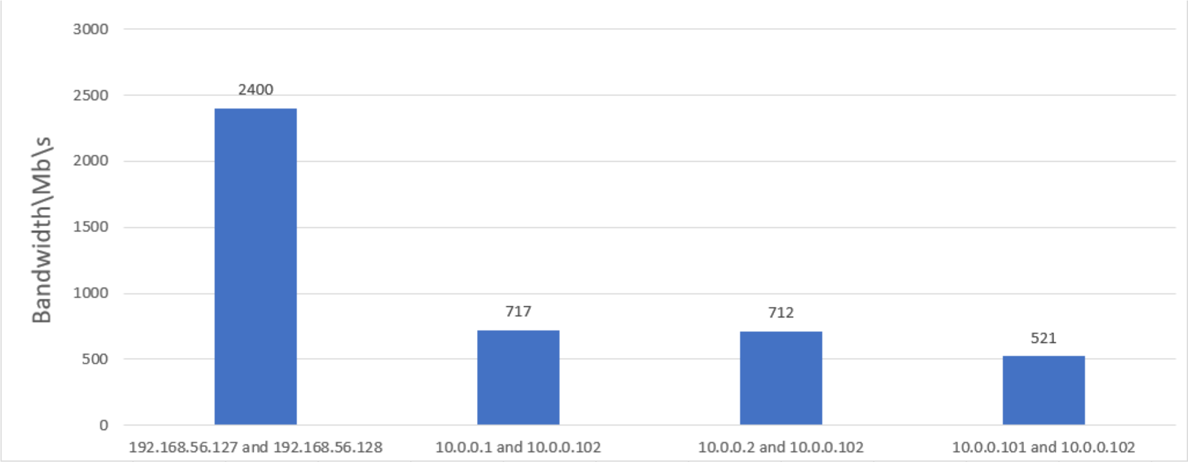
"Node: h2"

```
root@tpljqj:~# iperf -c 10.0.0.102 -M 1400B
WARNING: attempt to set TCP maximum segment size to 1400, but got 536
------------------------------------------------------------
Client connecting to 10.0.0.102, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.2 port 35032 connected with 10.0.0.102 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   849 MBytes   712 Mbits/sec
root@tpljqj:~# 
```

"Node: s1" (root)

```
root@tpljqj:~# iperf -c 10.0.0.102 -M 1400B
WARNING: attempt to set TCP maximum segment size to 1400, but got 536
------------------------------------------------------------
Client connecting to 10.0.0.102, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.101 port 44078 connected with 10.0.0.102 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   621 MBytes   521 Mbits/sec
root@tpljqj:~# 
```

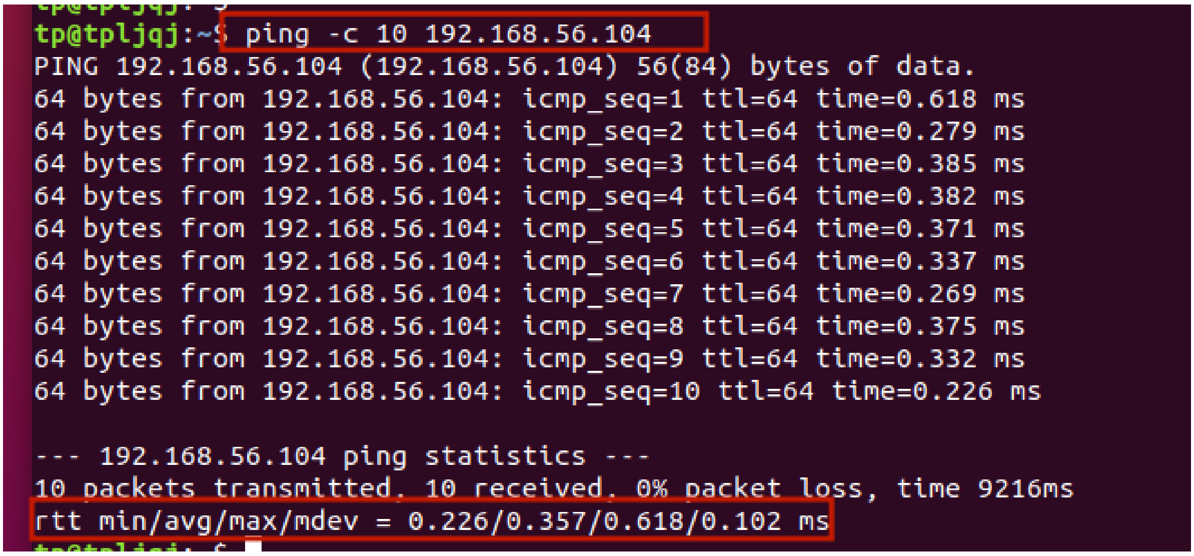
- From above results, we can get the following figure.



- The bandwidth between 10.0.0.1/10.0.0.2/10.0.0.101 and 10.0.0.102 is almost same because all of them use VXLAN, that is to say all packets will be encapsulated by enp0s8 in one VM and be decapsulated by enp0s8 in another VM.
- The bandwidth between 192.168.56.127 and 192.168.56.128 is much bigger than the three others. This is because packets transmitted between 192.168.56.127 and 192.168.56.128 don't use the XVLAN, which means this transmission doesn't spend time on encapsulating and decapsulating packets. What's more, because encapsulated packet has more headers than original packet and the total size of them is same because of the limitation of network MTU, the size of actual data in the original packet is bigger then the encapsulated packet. Therefore, the bandwidth between 192.168.56.127 and 192.168.56.128 should be bigger.

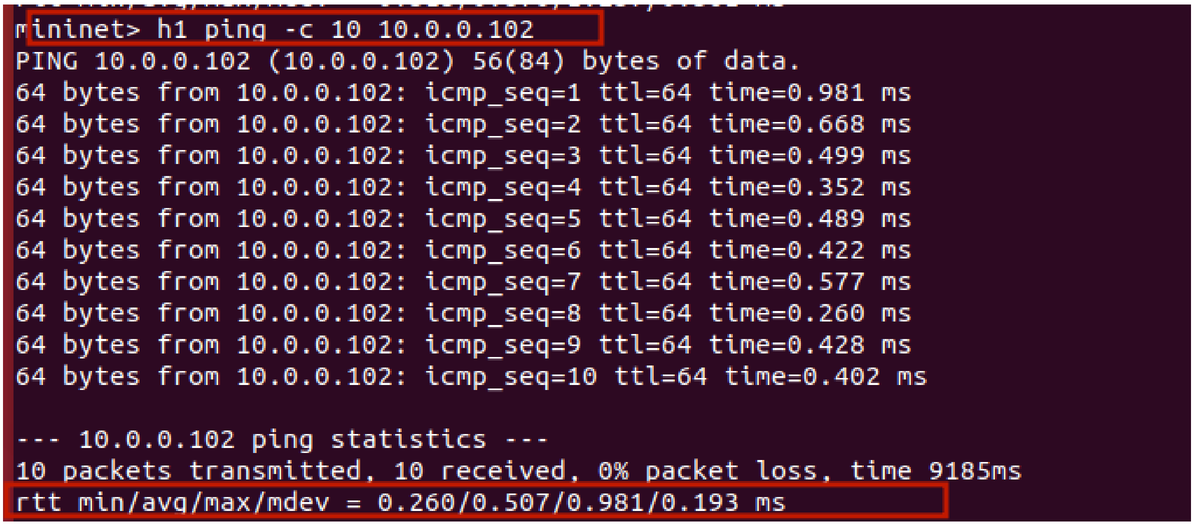
4 Exercise 3

- When testing the latency between 192.168.56.127 and 192.168.56.128 by using ping, we get followings result.

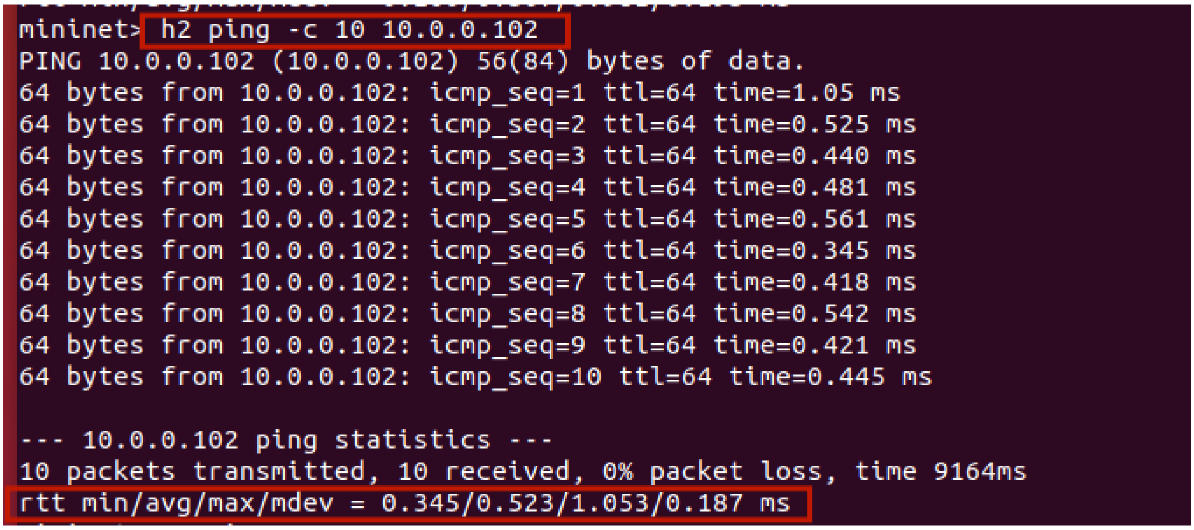


```
tp@tpljqj:~$
tp@tpljqj:~$ ping -c 10 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=0.618 ms
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=0.279 ms
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=0.385 ms
64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=0.382 ms
64 bytes from 192.168.56.104: icmp_seq=5 ttl=64 time=0.371 ms
64 bytes from 192.168.56.104: icmp_seq=6 ttl=64 time=0.337 ms
64 bytes from 192.168.56.104: icmp_seq=7 ttl=64 time=0.269 ms
64 bytes from 192.168.56.104: icmp_seq=8 ttl=64 time=0.375 ms
64 bytes from 192.168.56.104: icmp_seq=9 ttl=64 time=0.332 ms
64 bytes from 192.168.56.104: icmp_seq=10 ttl=64 time=0.226 ms

--- 192.168.56.104 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9216ms
rtt min/avg/max/mdev = 0.226/0.357/0.618/0.102 ms
tp@tpljqj:~$
```

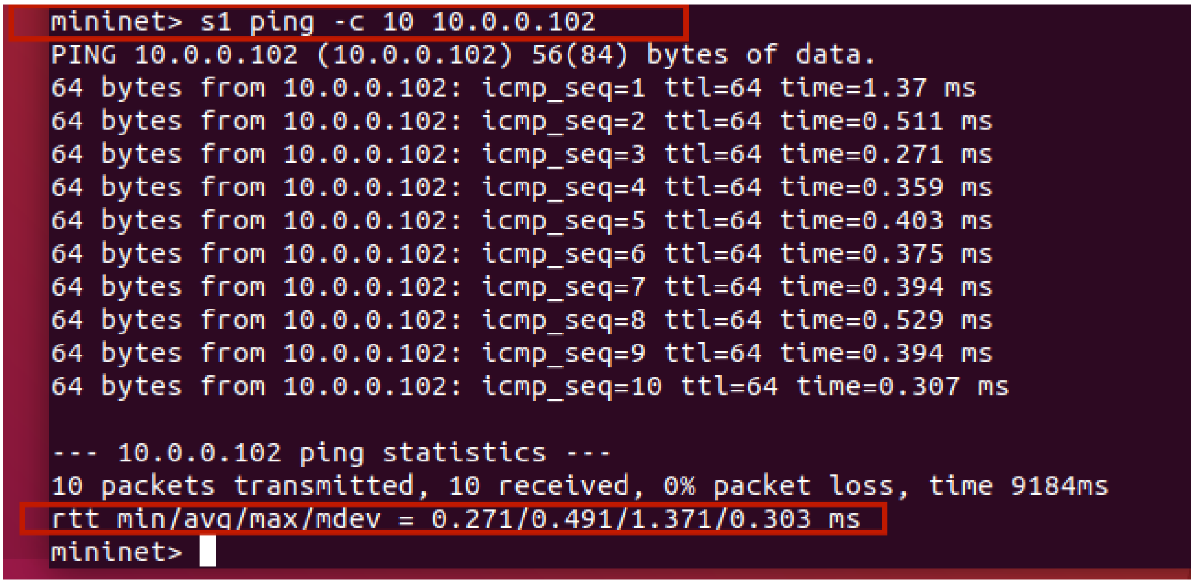
- When testing the latency between 10.0.0.1/10.0.0.2/10.0.0.101 and 10.0.0.102 by using ping, we get followings result.



```
mininet> h1 ping -c 10 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=0.981 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.668 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.499 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.352 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=0.489 ms
64 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=0.422 ms
64 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=0.577 ms
64 bytes from 10.0.0.102: icmp_seq=8 ttl=64 time=0.260 ms
64 bytes from 10.0.0.102: icmp_seq=9 ttl=64 time=0.428 ms
64 bytes from 10.0.0.102: icmp_seq=10 ttl=64 time=0.402 ms

--- 10.0.0.102 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9185ms
rtt min/avg/max/mdev = 0.260/0.507/0.981/0.193 ms
```

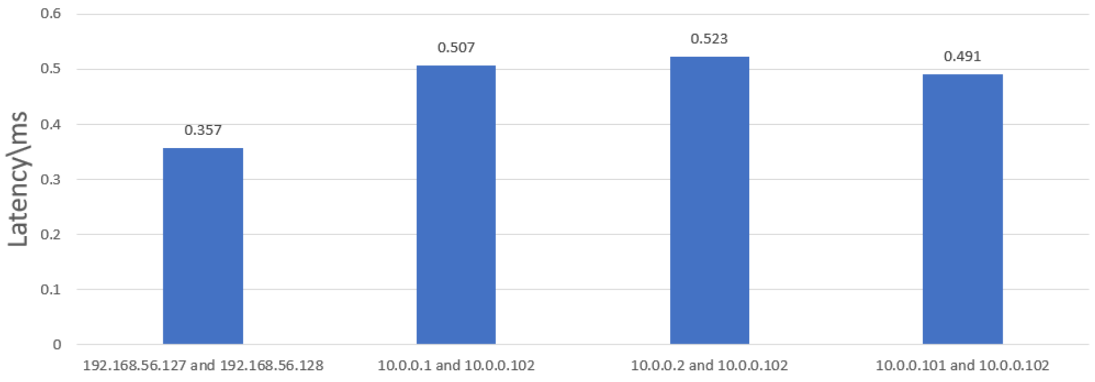


```
mininet> h2 ping -c 10 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=1.05 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.525 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.440 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.481 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=0.561 ms
64 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=0.345 ms
64 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=0.418 ms
64 bytes from 10.0.0.102: icmp_seq=8 ttl=64 time=0.542 ms
64 bytes from 10.0.0.102: icmp_seq=9 ttl=64 time=0.421 ms
64 bytes from 10.0.0.102: icmp_seq=10 ttl=64 time=0.445 ms

--- 10.0.0.102 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9164ms
rtt min/avg/max/mdev = 0.345/0.523/1.053/0.187 ms
```



```
mininet> s1 ping -c 10 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=1.37 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=0.511 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=0.271 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=0.359 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=0.403 ms
64 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=0.375 ms
64 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=0.394 ms
64 bytes from 10.0.0.102: icmp_seq=8 ttl=64 time=0.529 ms
64 bytes from 10.0.0.102: icmp_seq=9 ttl=64 time=0.394 ms
64 bytes from 10.0.0.102: icmp_seq=10 ttl=64 time=0.307 ms

--- 10.0.0.102 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9184ms
rtt min/avg/max/mdev = 0.271/0.491/1.371/0.303 ms
mininet> 
```

- From above results, we use the average RTT to compare their performance and get the following figure. From the figure, we can see that the latency between 192.168.56.127 and 192.168.56.128 is smaller than the three others. The reason for this is same as the first case in bandwidth, that is to say the time on encapsulating and decapsulating packets isn't required when pinging from 192.168.56.127 to 192.168.56.128.



5 Conclusion

In this lab, we learn what is VXLAN and how to use it to connect hosts in two VMs.