IT.3503 - Architecture Virtualisée

TP 2: An Introduction to Software Defined Networking (SDN)

GUO Xiaofan

YIN Chenghao

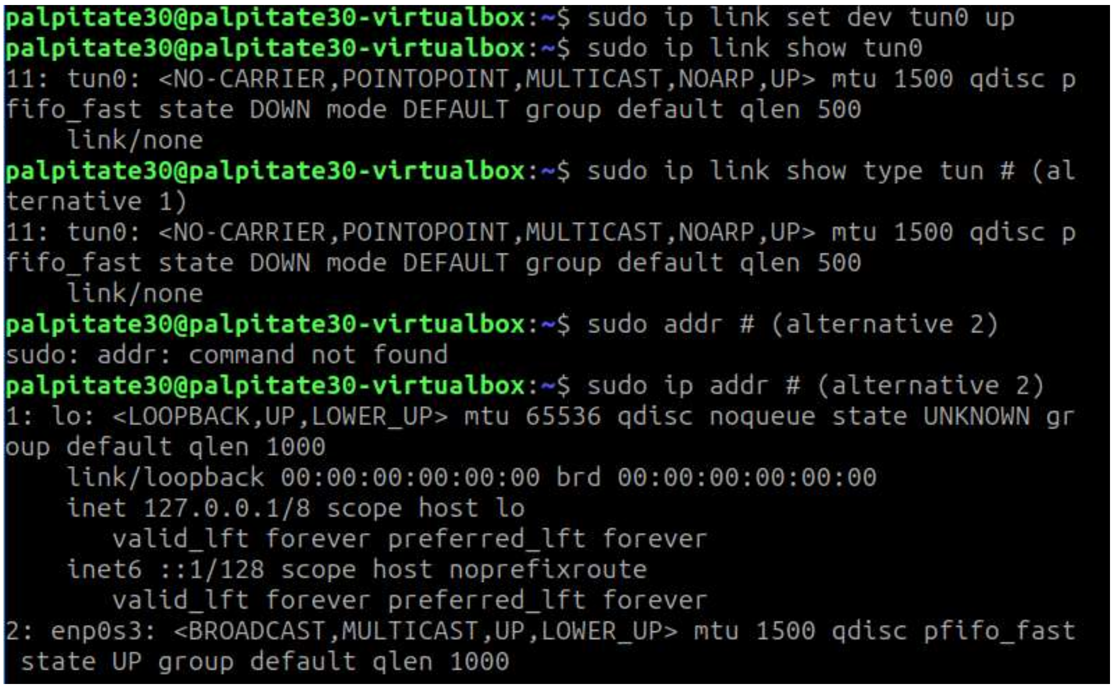
1 Linux Virtual Networking Devices

1.1 TUN/TAP virtual devices

1. What TUN and TAP devices can be used for ?
   - TUN devices are used to create point-to-point network tunnels, working at the network layer (layer 3).
   - TAP devices are used to create Ethernet tunnels, working at the data link layer (layer 2).

2. What is the difference between tap and tun devices ?
   - TUN devices simulate a network layer device and handle IP packets.
   - TAP devices simulate a link layer device and handle Ethernet frames.

3. Does tun0 have a MAC (Media Access Control) address ? Explain why ?

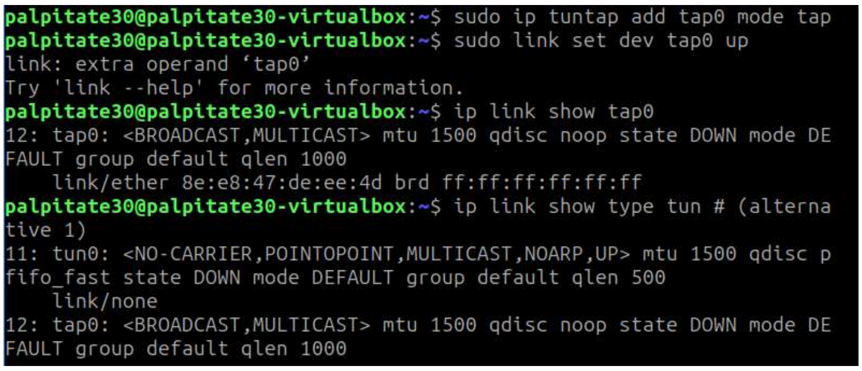


```
palpitate30@palpitate30-virtualbox:~$ sudo ip link set dev tun0 up
palpitate30@palpitate30-virtualbox:~$ sudo ip link show tun0
11: tun0: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc p
fifo_fast state DOWN mode DEFAULT group default qlen 500
    link/none
palpitate30@palpitate30-virtualbox:~$ sudo ip link show type tun # (al
ternative 1)
11: tun0: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc p
fifo_fast state DOWN mode DEFAULT group default qlen 500
    link/none
palpitate30@palpitate30-virtualbox:~$ sudo addr # (alternative 2)
sudo: addr: command not found
palpitate30@palpitate30-virtualbox:~$ sudo ip addr # (alternative 2)
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gr
oup 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 noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP group default qlen 1000
```

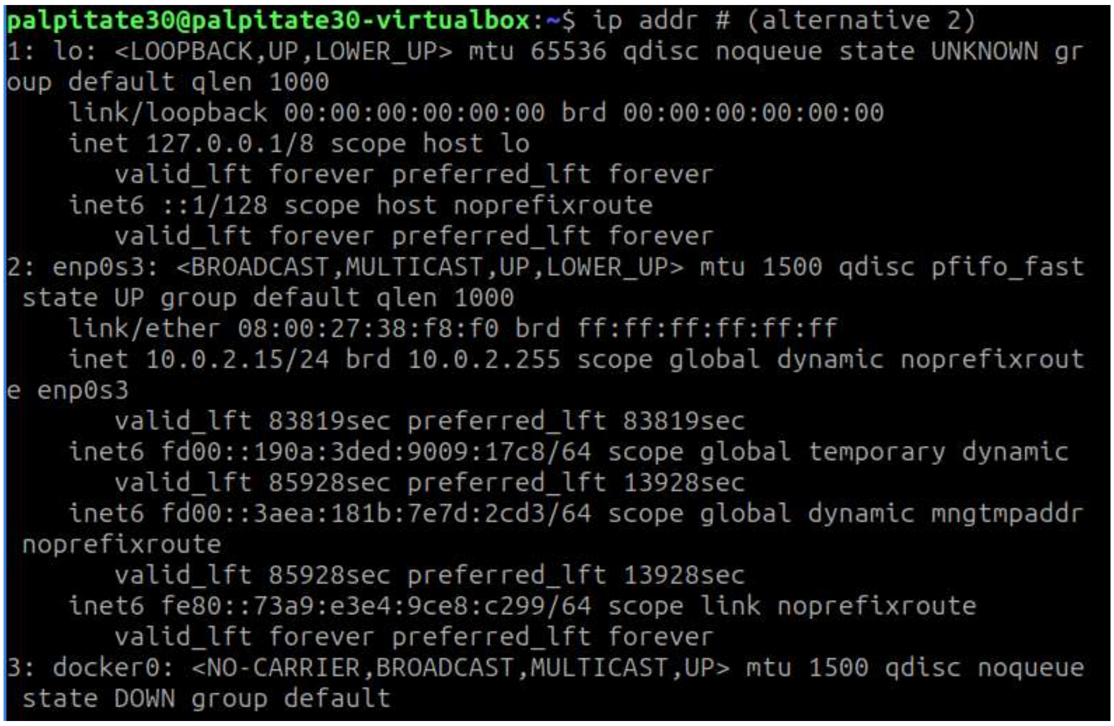
   - The tun0 device does not have a MAC address.
   - Because it operates at Layer 3 (Network Layer) and handles IP packets directly, bypassing the Layer 2 (Data Link Layer) where MAC addresses are used.

- MAC addresses are only necessary for devices that emulate Layer 2 functionality, such as TAP devices.

4. Does tap0 have a MAC (Media Access Control) address ? Explain why ?



```
palpitate30@palpitate30-virtualbox:~$ sudo ip tuntap add tap0 mode tap
palpitate30@palpitate30-virtualbox:~$ sudo link set dev tap0 up
link: extra operand 'tap0'
Try 'link --help' for more information.
palpitate30@palpitate30-virtualbox:~$ ip link show tap0
12: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DE
FAULT group default qlen 1000
    link/ether 8e:e8:47:de:ee:4d brd ff:ff:ff:ff:ff:ff
palpitate30@palpitate30-virtualbox:~$ ip link show type tun # (alterna
tive 1)
11: tun0: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc p
fifo_fast state DOWN mode DEFAULT group default qlen 500
    link/none
12: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DE
FAULT group default qlen 1000
```



```
palpitate30@palpitate30-virtualbox:~$ ip addr # (alternative 2)
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gr
oup 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 noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP group default qlen 1000
    link/ether 08:00:27:38:f8:f0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixrout
e enp0s3
       valid_lft 83819sec preferred_lft 83819sec
    inet6 fd00::190a:3ded:9009:17c8/64 scope global temporary dynamic
       valid_lft 85928sec preferred_lft 13928sec
    inet6 fd00::3aea:181b:7e7d:2cd3/64 scope global dynamic mngtmpaddr
 noprefixroute
       valid_lft 85928sec preferred_lft 13928sec
    inet6 fe80::73a9:e3e4:9ce8:c299/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
 state DOWN group default
```

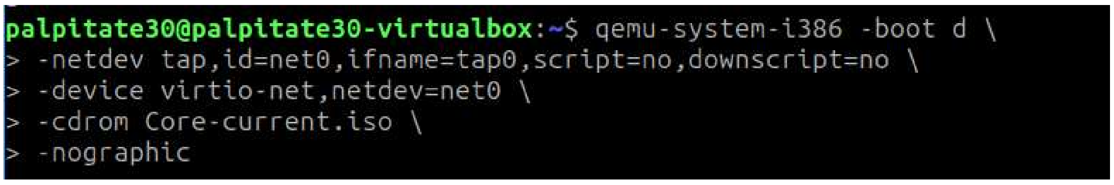
- Yes, tap0 has a MAC (Media Access Control) address, as shown in the output (link/ether 8e:e8:47:de:ee:4d).
- Because tap0 operates at the data link layer (Layer 2) of the OSI model, where MAC addresses are required for communication between devices.
- TAP devices simulate Ethernet interfaces and therefore must include a MAC address to enable Ethernet frame handling.

5. What is the status of tap0 interface ? Explain why ?

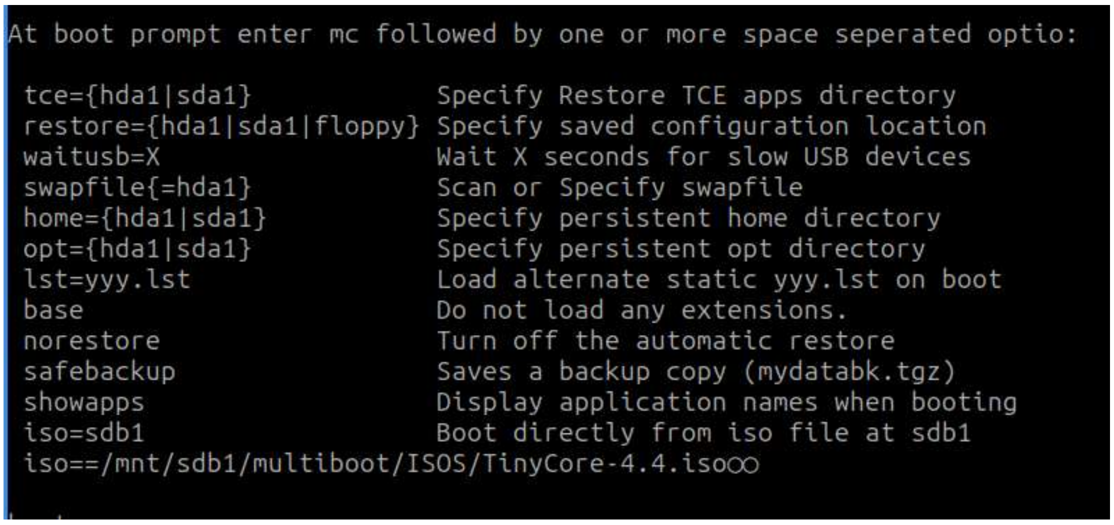
- The status of the tap0 interface is shown as DOWN. This indicates that the interface has been created but is not yet active or "up."

- Because the interface must be explicitly brought up using the command ip link set dev tap0 up. Without this command, the interface remains in the default "DOWN" state, meaning it cannot transmit or receive packets.

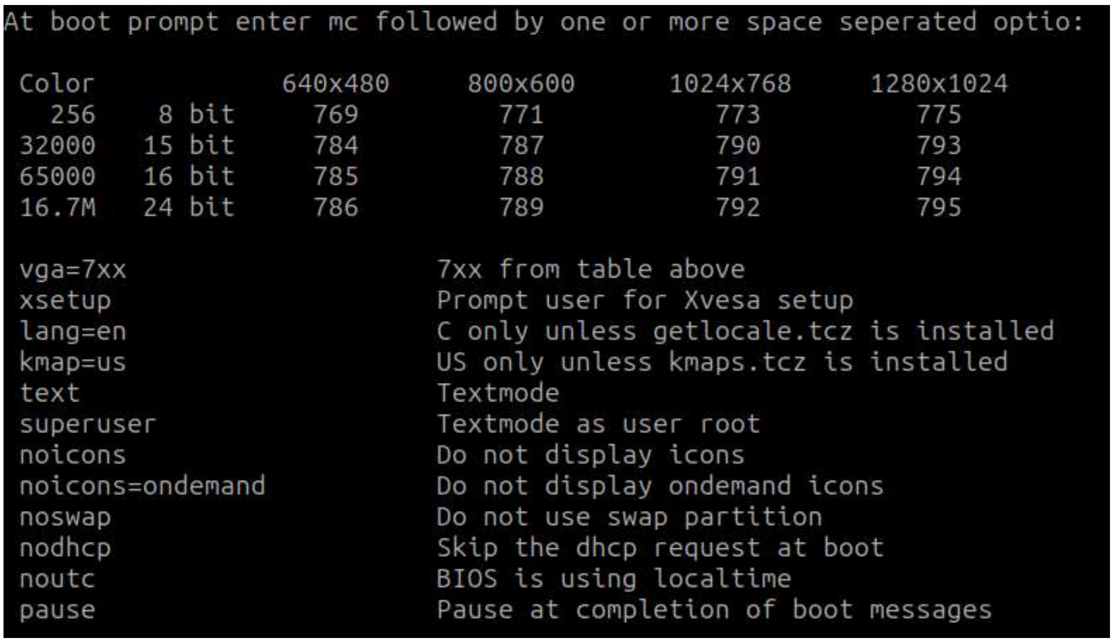
1.2 Optional section



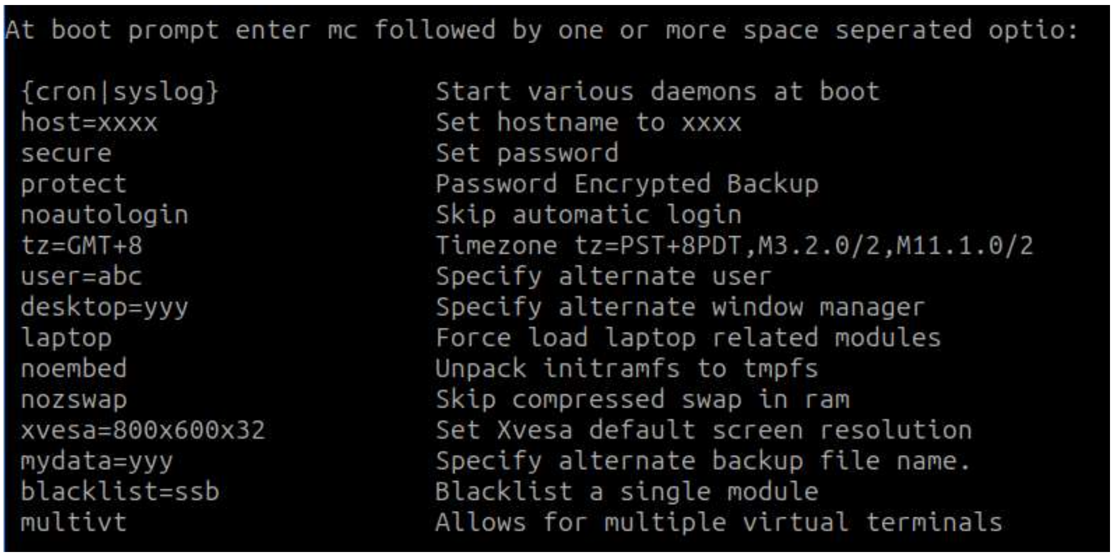
```
palpitate30@palpitate30-virtualbox:~$ qemu-system-i386 -boot d \
> -netdev tap,id=net0,ifname=tap0,script=no,downscript=no \
> -device virtio-net,netdev=net0 \
> -cdrom Core-current.iso \
> -nographic
```



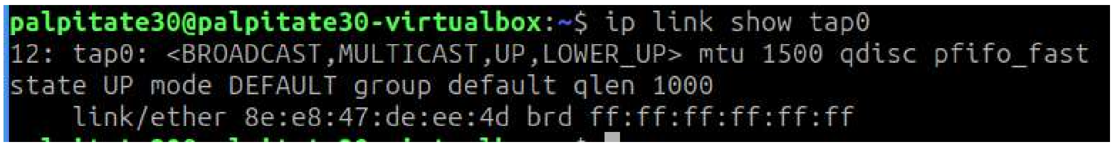
```
At boot prompt enter mc followed by one or more space seperated optio:

 tce={hda1|sda1}             Specify Restore TCE apps directory
 restore={hda1|sda1|floppy} Specify saved configuration location
 waitusb=X                   Wait X seconds for slow USB devices
 swapfile{=hda1}             Scan or Specify swapfile
 home={hda1|sda1}            Specify persistent home directory
 opt={hda1|sda1}             Specify persistent opt directory
 lst=yyy.lst                 Load alternate static yyy.lst on boot
 base                        Do not load any extensions.
 norestore                   Turn off the automatic restore
 safebackup                  Saves a backup copy (mydatabk.tgz)
 showapps                    Display application names when booting
 iso=sdb1                    Boot directly from iso file at sdb1
 iso==/mnt/sdb1/multiboot/ISOS/TinyCore-4.4.iso∞
```



```
At boot prompt enter mc followed by one or more space seperated optio:

 Color              640x480     800x600      1024x768      1280x1024
   256     8 bit      769         771           773           775
 32000    15 bit      784         787           790           793
 65000    16 bit      785         788           791           794
 16.7M    24 bit      786         789           792           795

 vga=7xx                     7xx from table above
 xsetup                      Prompt user for Xvesa setup
 lang=en                     C only unless getlocale.tcz is installed
 kmap=us                     US only unless kmaps.tcz is installed
 text                        Textmode
 superuser                   Textmode as user root
 noicons                     Do not display icons
 noicons=ondemand            Do not display ondemand icons
 noswap                      Do not use swap partition
 nodhcp                      Skip the dhcp request at boot
 noutc                       BIOS is using localtime
 pause                       Pause at completion of boot messages
```



```
At boot prompt enter mc followed by one or more space seperated optio:

 {cron|syslog}              Start various daemons at boot
 host=xxxx                  Set hostname to xxxx
 secure                     Set password
 protect                    Password Encrypted Backup
 noautologin                Skip automatic login
 tz=GMT+8                   Timezone tz=PST+8PDT,M3.2.0/2,M11.1.0/2
 user=abc                   Specify alternate user
 desktop=yyy                Specify alternate window manager
 laptop                     Force load laptop related modules
 noembed                    Unpack initramfs to tmpfs
 nozswap                    Skip compressed swap in ram
 xvesa=800x600x32           Set Xvesa default screen resolution
 mydata=yyy                 Specify alternate backup file name.
 blacklist=ssb              Blacklist a single module
 multivt                    Allows for multiple virtual terminals
```



```
palpitate30@palpitate30-virtualbox:~$ ip link show tap0
12: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP mode DEFAULT group default qlen 1000
    link/ether 8e:e8:47:de:ee:4d brd ff:ff:ff:ff:ff:ff
```

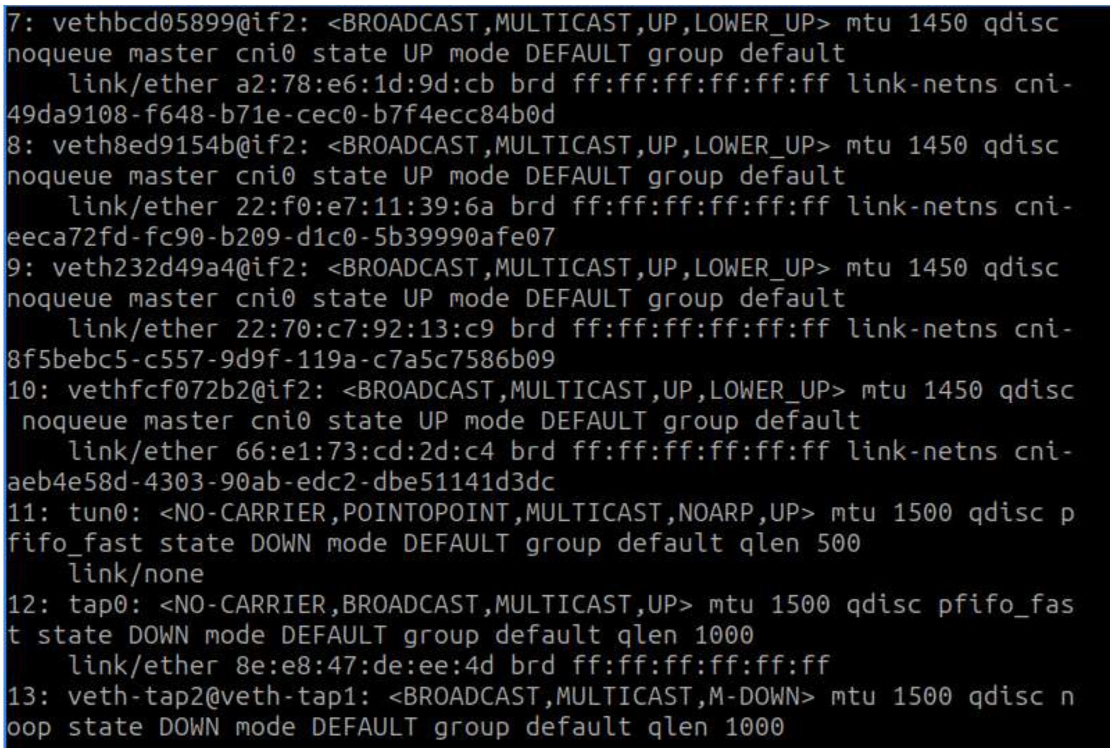
1. What do you notice ?

- After starting the VM using qemu-system-i386 and connecting the virtual network interface to tap0, the tap0 interface state changes, its status will show as UP when do verify using ip link show tap0.
- This happens because the QEMU process activates the tap0 interface to handle virtual network traffic. Additionally, if network traffic occurs within the VM, data may be transmitted through tap0.
- In the last picture, the tap0 status is already displayed as UP.

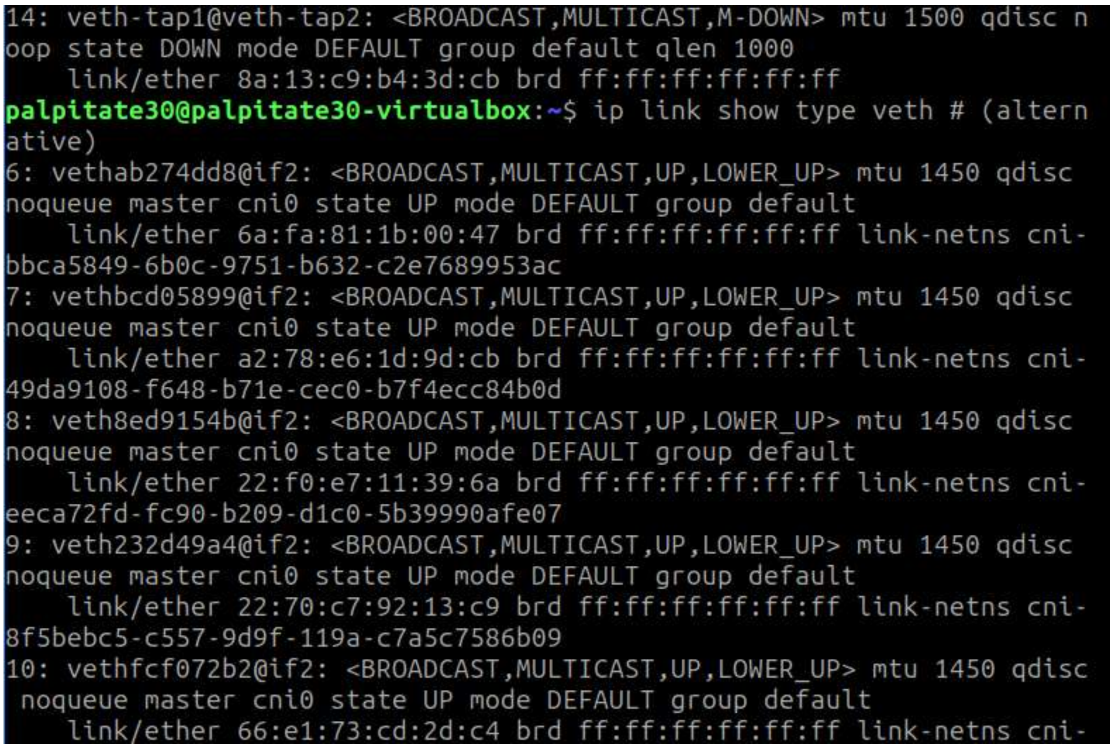
1.3 VETH virtual devices

1. What veth devices can be used for ?

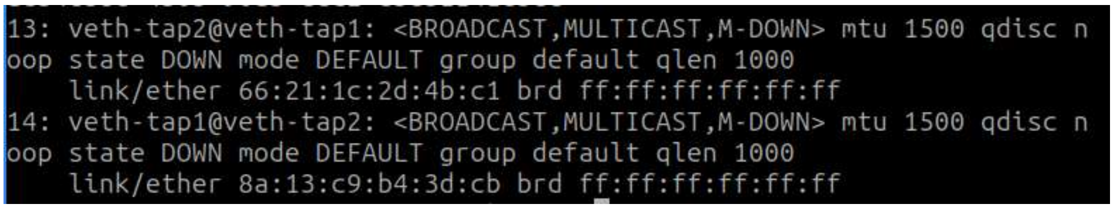
- Veth devices are used to create a pair of connected virtual Ethernet interfaces. They are commonly used for interconnecting network namespaces, containers, or virtual machines within a host.
- Each end of the veth pair behaves like a physical Ethernet interface, enabling data transmission between the connected namespaces or network entities.



```
7: vethbcd05899@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether a2:78:e6:1d:9d:cb brd ff:ff:ff:ff:ff:ff link-netns cni-
49da9108-f648-b71e-cec0-b7f4ecc84b0d
8: veth8ed9154b@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 22:f0:e7:11:39:6a brd ff:ff:ff:ff:ff:ff link-netns cni-
eeca72fd-fc90-b209-d1c0-5b39990afe07
9: veth232d49a4@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 22:70:c7:92:13:c9 brd ff:ff:ff:ff:ff:ff link-netns cni-
8f5bebc5-c557-9d9f-119a-c7a5c7586b09
10: vethfcf072b2@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
 noqueue master cni0 state UP mode DEFAULT group default
    link/ether 66:e1:73:cd:2d:c4 brd ff:ff:ff:ff:ff:ff link-netns cni-
aeb4e58d-4303-90ab-edc2-dbe51141d3dc
11: tun0: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc p
fifo_fast state DOWN mode DEFAULT group default qlen 500
    link/none
12: tap0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fas
t state DOWN mode DEFAULT group default qlen 1000
    link/ether 8e:e8:47:de:ee:4d brd ff:ff:ff:ff:ff:ff
13: veth-tap2@veth-tap1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc n
oop state DOWN mode DEFAULT group default qlen 1000
```



```
14: veth-tap1@veth-tap2: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc n
oop state DOWN mode DEFAULT group default qlen 1000
    link/ether 8a:13:c9:b4:3d:cb brd ff:ff:ff:ff:ff:ff
palpitate30@palpitate30-virtualbox:~$ ip link show type veth # (altern
ative)
6: vethab274dd8@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 6a:fa:81:1b:00:47 brd ff:ff:ff:ff:ff:ff link-netns cni-
bbca5849-6b0c-9751-b632-c2e7689953ac
7: vethbcd05899@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether a2:78:e6:1d:9d:cb brd ff:ff:ff:ff:ff:ff link-netns cni-
49da9108-f648-b71e-cec0-b7f4ecc84b0d
8: veth8ed9154b@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 22:f0:e7:11:39:6a brd ff:ff:ff:ff:ff:ff link-netns cni-
eeca72fd-fc90-b209-d1c0-5b39990afe07
9: veth232d49a4@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 22:70:c7:92:13:c9 brd ff:ff:ff:ff:ff:ff link-netns cni-
8f5bebc5-c557-9d9f-119a-c7a5c7586b09
10: vethfcf072b2@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
 noqueue master cni0 state UP mode DEFAULT group default
    link/ether 66:e1:73:cd:2d:c4 brd ff:ff:ff:ff:ff:ff link-netns cni-
```



```
13: veth-tap2@veth-tap1: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc n
oop state DOWN mode DEFAULT group default qlen 1000
    link/ether 66:21:1c:2d:4b:c1 brd ff:ff:ff:ff:ff:ff
14: veth-tap1@veth-tap2: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc n
oop state DOWN mode DEFAULT group default qlen 1000
    link/ether 8a:13:c9:b4:3d:cb brd ff:ff:ff:ff:ff:ff
```

2. Can veth devices operate on the data link layer of the OSI model ? and why?

- Yes, veth devices operate on the data link layer (Layer 2) of the OSI model.

- Because veth devices simulate Ethernet interfaces, which work at the data link layer by transmitting and receiving Ethernet frames. They support MAC addresses, frame encapsulation, and other Layer 2 features, making them suitable for bridging and Layer 2 communication.

1.4 Bridge devices

1. Linux bridge is equivalent to a router, switch or both ?

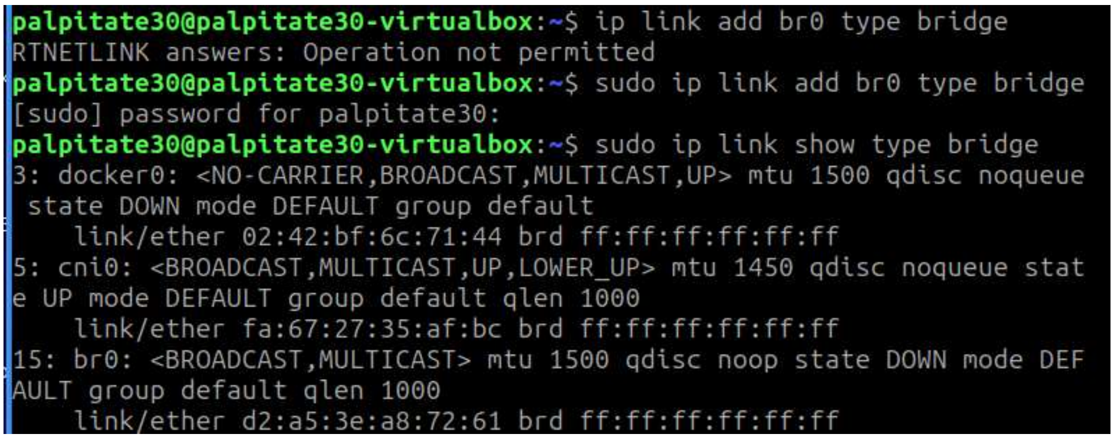
- A Linux bridge is equivalent to a switch, as it forwards Ethernet frames between connected devices based on MAC addresses.
- Bridges operate at the data link layer (Layer 2), routers and handle IP packets work at the network layer (Layer 3).

2. In which level of the OSI model a linux bridge operates ?

- A Linux bridge operates at the data link layer (Layer 2) of the OSI model.
- It forwards Ethernet frames based on MAC addresses and does not process IP packets or higher-layer protocols.

3. Can a bridge have an IP address ? If yes, what for ?

- Yes, a Linux bridge can have an IP address, but it is not required for the bridge's primary function of forwarding frames.
- The IP address is used for management purposes, such as configuring the bridge, monitoring traffic, or enabling administrative tasks.



```
palpitate30@palpitate30-virtualbox:~$ ip link add br0 type bridge
RTNETLINK answers: Operation not permitted
palpitate30@palpitate30-virtualbox:~$ sudo ip link add br0 type bridge
[sudo] password for palpitate30:
palpitate30@palpitate30-virtualbox:~$ sudo ip link show type bridge
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
 state DOWN mode DEFAULT group default
    link/ether 02:42:bf:6c:71:44 brd ff:ff:ff:ff:ff:ff
5: cni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether fa:67:27:35:af:bc brd ff:ff:ff:ff:ff:ff
15: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEF
AULT group default qlen 1000
    link/ether d2:a5:3e:a8:72:61 brd ff:ff:ff:ff:ff:ff
```

1.5 Miscellaneous

1. How to get interfaces list in JSON format using ip ?

- Use the following command:
- *ip -j link show*

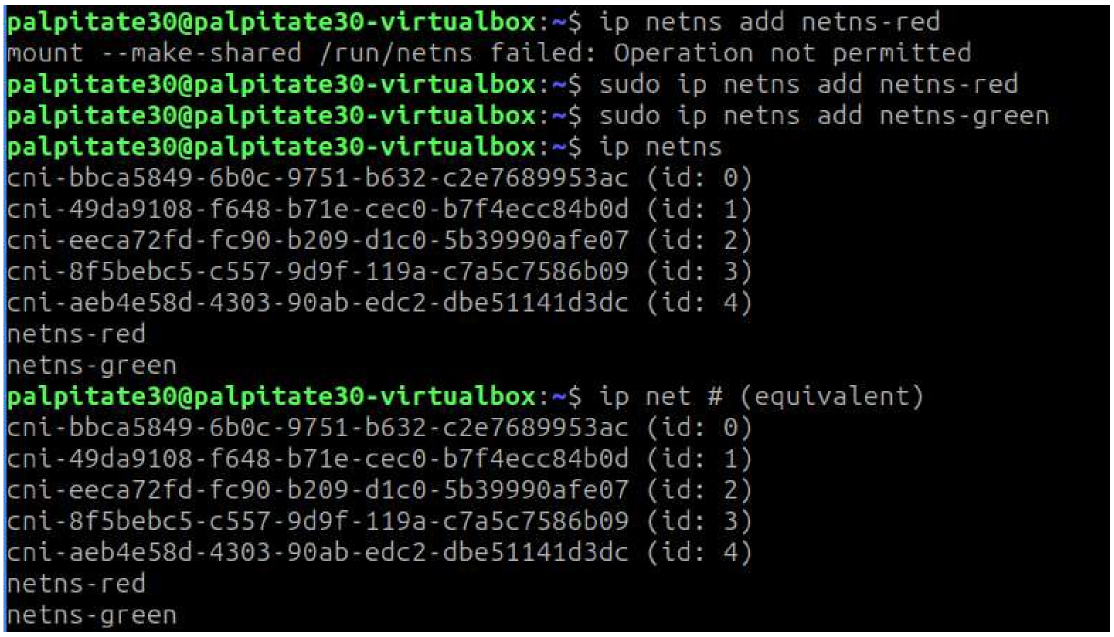
2. How to show more details about a given interface using ip ?

- Use the following command:
- *ip -details link show <interface-name>*
- Replace <interface-name> with the name of the interface (e.g., eth0).

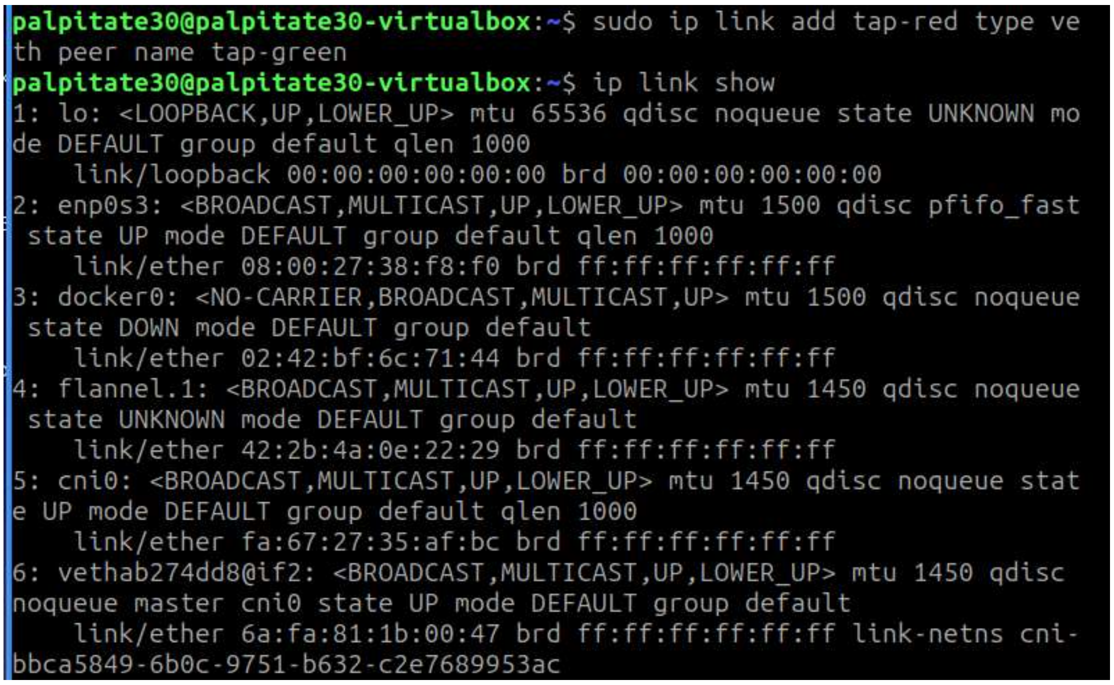
2 Virtual Network Infrastructure

2.1 veth pairs in practice

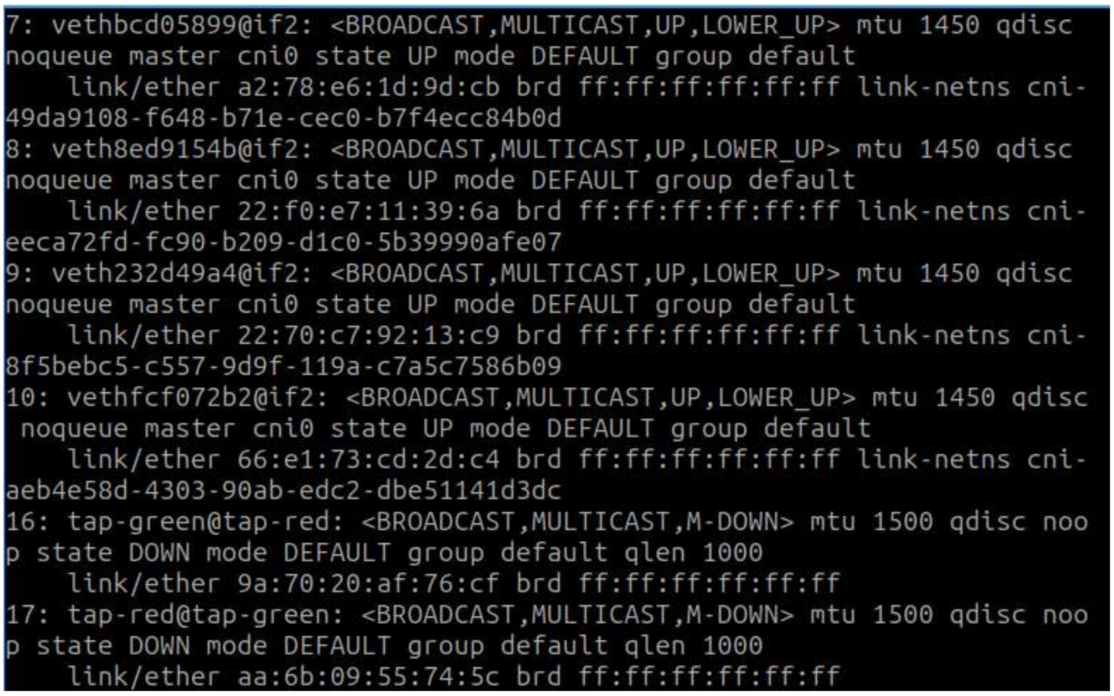
1st Scenario



```
palpitate30@palpitate30-virtualbox:~$ ip netns add netns-red
mount --make-shared /run/netns failed: Operation not permitted
palpitate30@palpitate30-virtualbox:~$ sudo ip netns add netns-red
palpitate30@palpitate30-virtualbox:~$ sudo ip netns add netns-green
palpitate30@palpitate30-virtualbox:~$ ip netns
cni-bbca5849-6b0c-9751-b632-c2e7689953ac (id: 0)
cni-49da9108-f648-b71e-cec0-b7f4ecc84b0d (id: 1)
cni-eeca72fd-fc90-b209-d1c0-5b39990afe07 (id: 2)
cni-8f5bebc5-c557-9d9f-119a-c7a5c7586b09 (id: 3)
cni-aeb4e58d-4303-90ab-edc2-dbe51141d3dc (id: 4)
netns-red
netns-green
palpitate30@palpitate30-virtualbox:~$ ip net # (equivalent)
cni-bbca5849-6b0c-9751-b632-c2e7689953ac (id: 0)
cni-49da9108-f648-b71e-cec0-b7f4ecc84b0d (id: 1)
cni-eeca72fd-fc90-b209-d1c0-5b39990afe07 (id: 2)
cni-8f5bebc5-c557-9d9f-119a-c7a5c7586b09 (id: 3)
cni-aeb4e58d-4303-90ab-edc2-dbe51141d3dc (id: 4)
netns-red
netns-green
```



```
palpitate30@palpitate30-virtualbox:~$ sudo ip link add tap-red type ve
th peer name tap-green
palpitate30@palpitate30-virtualbox:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mo
de DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:38:f8:f0 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
 state DOWN mode DEFAULT group default
    link/ether 02:42:bf:6c:71:44 brd ff:ff:ff:ff:ff:ff
4: flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue
 state UNKNOWN mode DEFAULT group default
    link/ether 42:2b:4a:0e:22:29 brd ff:ff:ff:ff:ff:ff
5: cni0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue stat
e UP mode DEFAULT group default qlen 1000
    link/ether fa:67:27:35:af:bc brd ff:ff:ff:ff:ff:ff
6: vethab274dd8@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 6a:fa:81:1b:00:47 brd ff:ff:ff:ff:ff:ff link-netns cni-
bbca5849-6b0c-9751-b632-c2e7689953ac
```



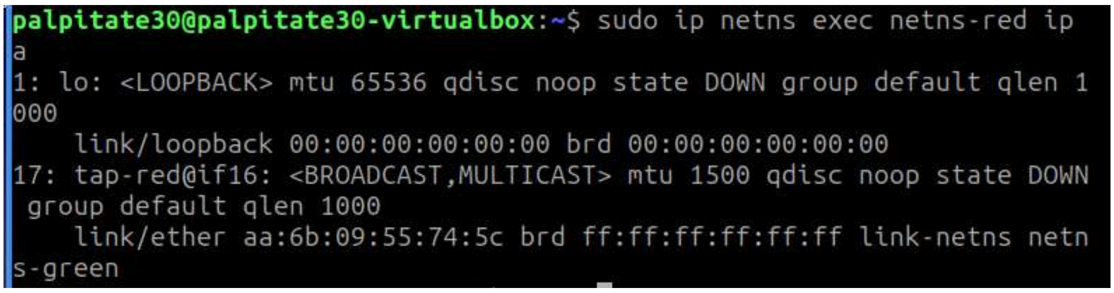
```
7: vethbcd05899@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether a2:78:e6:1d:9d:cb brd ff:ff:ff:ff:ff:ff link-netns cni-
49da9108-f648-b71e-cec0-b7f4ecc84b0d
8: veth8ed9154b@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 22:f0:e7:11:39:6a brd ff:ff:ff:ff:ff:ff link-netns cni-
eeca72fd-fc90-b209-d1c0-5b39990afe07
9: veth232d49a4@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
noqueue master cni0 state UP mode DEFAULT group default
    link/ether 22:70:c7:92:13:c9 brd ff:ff:ff:ff:ff:ff link-netns cni-
8f5bebc5-c557-9d9f-119a-c7a5c7586b09
10: vethfcf072b2@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc
 noqueue master cni0 state UP mode DEFAULT group default
    link/ether 66:e1:73:cd:2d:c4 brd ff:ff:ff:ff:ff:ff link-netns cni-
aeb4e58d-4303-90ab-edc2-dbe51141d3dc
16: tap-green@tap-red: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noo
p state DOWN mode DEFAULT group default qlen 1000
    link/ether 9a:70:20:af:76:cf brd ff:ff:ff:ff:ff:ff
17: tap-red@tap-green: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noo
p state DOWN mode DEFAULT group default qlen 1000
    link/ether aa:6b:09:55:74:5c brd ff:ff:ff:ff:ff:ff
```

1. What do you notice ?

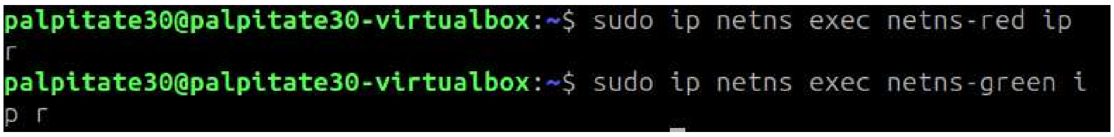
- After creating the tap-red and tap-green veth pair, both interfaces are initially visible in the global namespace and listed when running ip link show.
- After assigning tap-red and tap-green to their respective namespaces (netns-red and netns-green), the interfaces are no longer visible in the

global namespace. They are now isolated within their corresponding namespaces and can only be managed or viewed by running commands inside those namespaces using ip netns exec.

- This behavior reflects the separation and isolation of network resources provided by network namespaces.



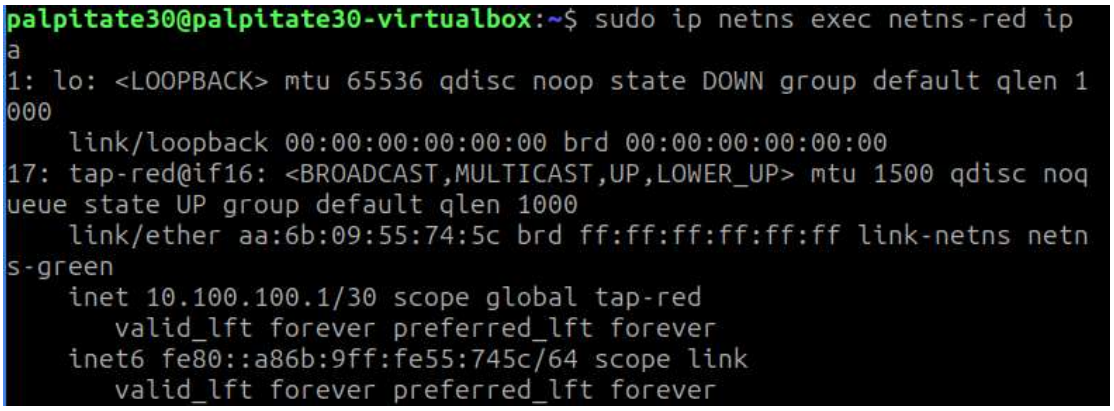
```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-red ip
a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1
000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
17: tap-red@if16: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
 group default qlen 1000
    link/ether aa:6b:09:55:74:5c brd ff:ff:ff:ff:ff:ff link-netns netn
s-green
```



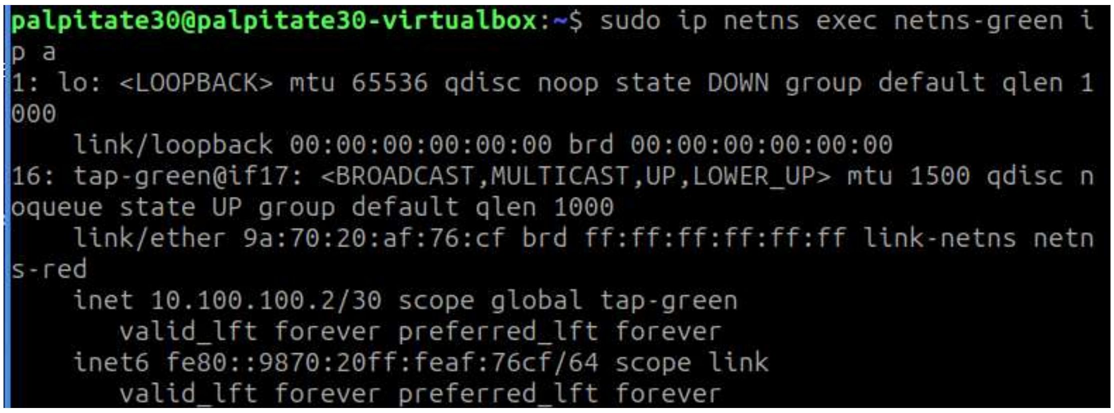
```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-red ip
r
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-green i
p r
```

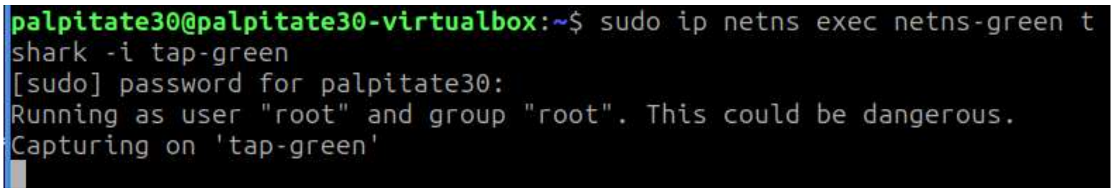
2. What do you notice ?

- Listing the network devices inside netns-red and netns-green shows that each namespace contains its own isolated interfaces (tap-red in netns-red and tap-green in netns-green). These interfaces are in the DOWN state, indicating that they are not yet activated.
- Listing the routes inside each namespace shows no configured routes, meaning no routing or communication paths are currently set up. This confirms that the namespaces are isolated from each other and from the global namespace, requiring explicit configuration to enable connectivity.

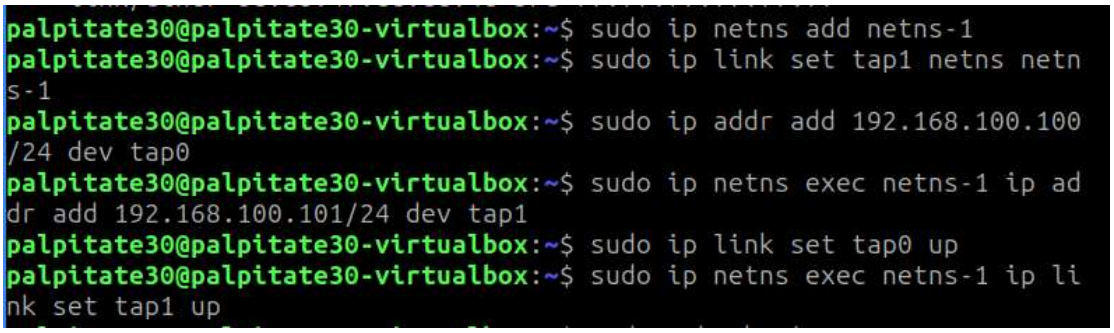


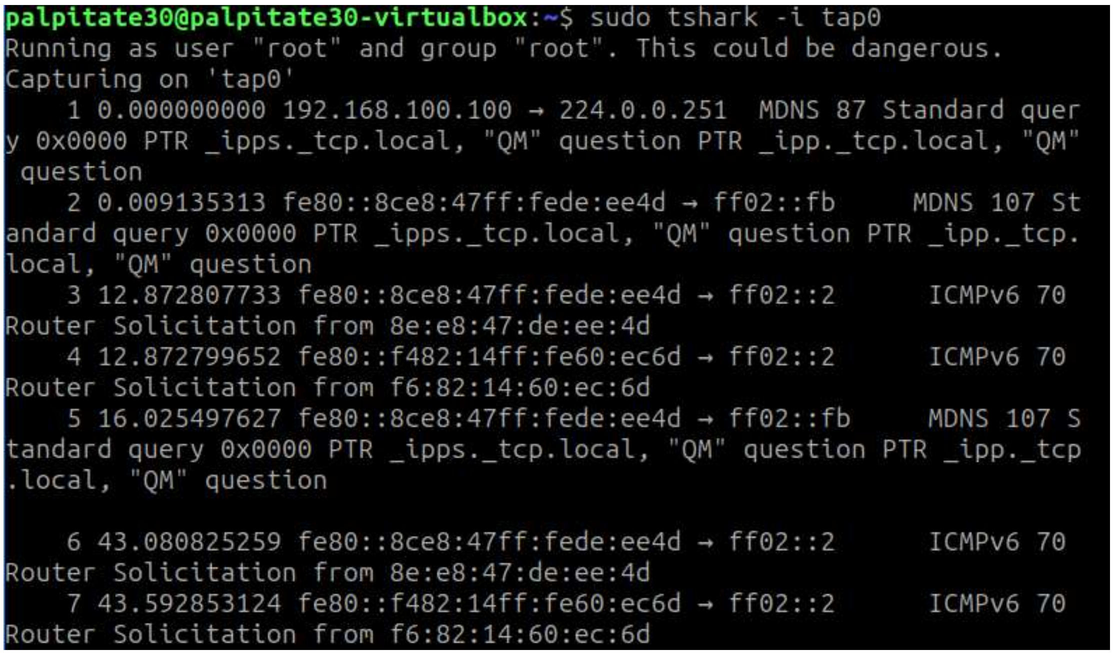
```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-red ip
a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1
000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
17: tap-red@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noq
ueue state UP group default qlen 1000
    link/ether aa:6b:09:55:74:5c brd ff:ff:ff:ff:ff:ff link-netns netn
s-green
    inet 10.100.100.1/30 scope global tap-red
       valid_lft forever preferred_lft forever
    inet6 fe80::a86b:9ff:fe55:745c/64 scope link
       valid_lft forever preferred_lft forever
```

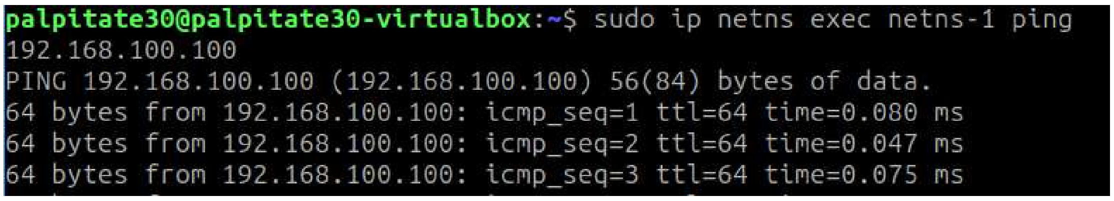

```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-green i
p a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1
000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
16: tap-green@if17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc n
oqueue state UP group default qlen 1000
    link/ether 9a:70:20:af:76:cf brd ff:ff:ff:ff:ff:ff link-netns netn
s-red
    inet 10.100.100.2/30 scope global tap-green
       valid_lft forever preferred_lft forever
    inet6 fe80::9870:20ff:feaf:76cf/64 scope link
       valid_lft forever preferred_lft forever
```


```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-green t
shark -i tap-green
[sudo] password for palpitate30:
Running as user "root" and group "root". This could be dangerous.
Capturing on 'tap-green'
```

2nd Scenario

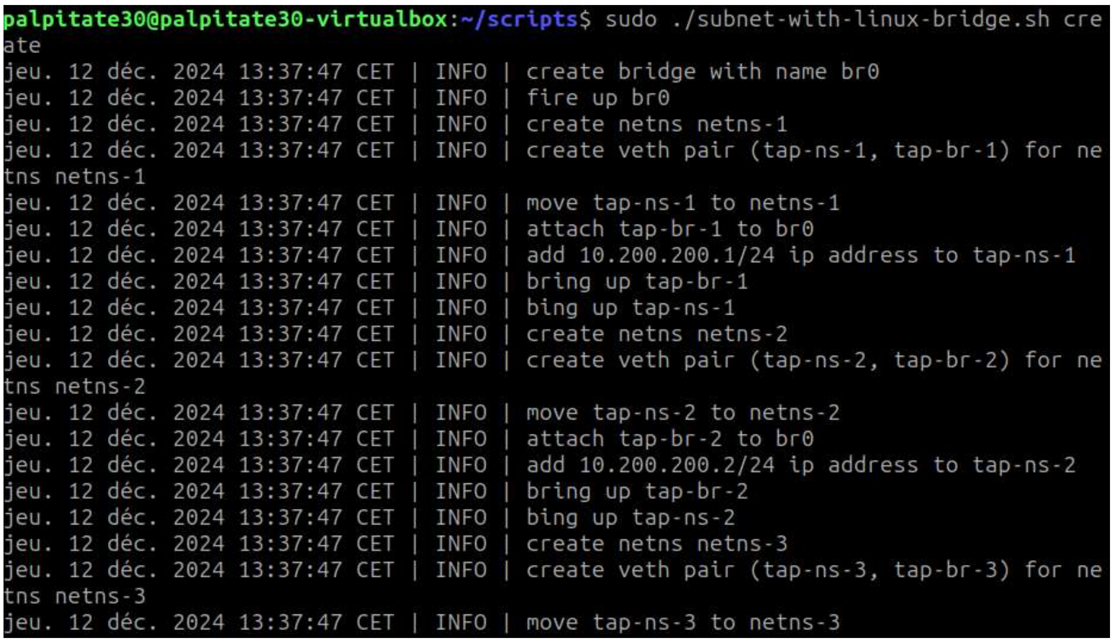

```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns add netns-1
palpitate30@palpitate30-virtualbox:~$ sudo ip link set tap1 netns netn
s-1
palpitate30@palpitate30-virtualbox:~$ sudo ip addr add 192.168.100.100
/24 dev tap0
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-1 ip ad
dr add 192.168.100.101/24 dev tap1
palpitate30@palpitate30-virtualbox:~$ sudo ip link set tap0 up
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-1 ip li
nk set tap1 up
```


```
palpitate30@palpitate30-virtualbox:~$ sudo tshark -i tap0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'tap0'
    1 0.000000000 192.168.100.100 → 224.0.0.251  MDNS 87 Standard quer
y 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM"
 question
    2 0.009135313 fe80::8ce8:47ff:fede:ee4d → ff02::fb     MDNS 107 St
andard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.
local, "QM" question
    3 12.872807733 fe80::8ce8:47ff:fede:ee4d → ff02::2      ICMPv6 70
Router Solicitation from 8e:e8:47:de:ee:4d
    4 12.872799652 fe80::f482:14ff:fe60:ec6d → ff02::2      ICMPv6 70
Router Solicitation from f6:82:14:60:ec:6d
    5 16.025497627 fe80::8ce8:47ff:fede:ee4d → ff02::fb     MDNS 107 S
tandard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp
.local, "QM" question

    6 43.080825259 fe80::8ce8:47ff:fede:ee4d → ff02::2      ICMPv6 70
Router Solicitation from 8e:e8:47:de:ee:4d
    7 43.592853124 fe80::f482:14ff:fe60:ec6d → ff02::2      ICMPv6 70
Router Solicitation from f6:82:14:60:ec:6d
```

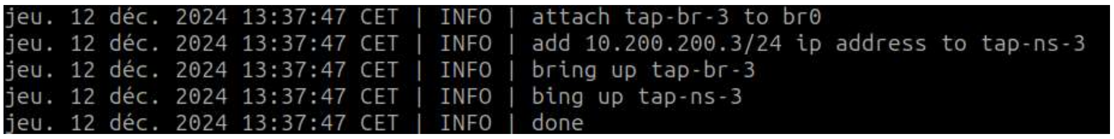


```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec netns-1 ping
192.168.100.100
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 192.168.100.100: icmp_seq=3 ttl=64 time=0.075 ms
```

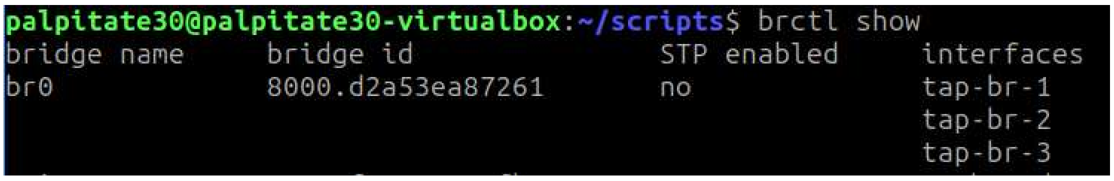
2.2 Create a virtual sub-network using Linux bridge



```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ./subnet-with-linux-bridge.sh cre
ate
jeu. 12 déc. 2024 13:37:47 CET | INFO | create bridge with name br0
jeu. 12 déc. 2024 13:37:47 CET | INFO | fire up br0
jeu. 12 déc. 2024 13:37:47 CET | INFO | create netns netns-1
jeu. 12 déc. 2024 13:37:47 CET | INFO | create veth pair (tap-ns-1, tap-br-1) for ne
tns netns-1
jeu. 12 déc. 2024 13:37:47 CET | INFO | move tap-ns-1 to netns-1
jeu. 12 déc. 2024 13:37:47 CET | INFO | attach tap-br-1 to br0
jeu. 12 déc. 2024 13:37:47 CET | INFO | add 10.200.200.1/24 ip address to tap-ns-1
jeu. 12 déc. 2024 13:37:47 CET | INFO | bring up tap-br-1
jeu. 12 déc. 2024 13:37:47 CET | INFO | bing up tap-ns-1
jeu. 12 déc. 2024 13:37:47 CET | INFO | create netns netns-2
jeu. 12 déc. 2024 13:37:47 CET | INFO | create veth pair (tap-ns-2, tap-br-2) for ne
tns netns-2
jeu. 12 déc. 2024 13:37:47 CET | INFO | move tap-ns-2 to netns-2
jeu. 12 déc. 2024 13:37:47 CET | INFO | attach tap-br-2 to br0
jeu. 12 déc. 2024 13:37:47 CET | INFO | add 10.200.200.2/24 ip address to tap-ns-2
jeu. 12 déc. 2024 13:37:47 CET | INFO | bring up tap-br-2
jeu. 12 déc. 2024 13:37:47 CET | INFO | bing up tap-ns-2
jeu. 12 déc. 2024 13:37:47 CET | INFO | create netns netns-3
jeu. 12 déc. 2024 13:37:47 CET | INFO | create veth pair (tap-ns-3, tap-br-3) for ne
tns netns-3
jeu. 12 déc. 2024 13:37:47 CET | INFO | move tap-ns-3 to netns-3
```

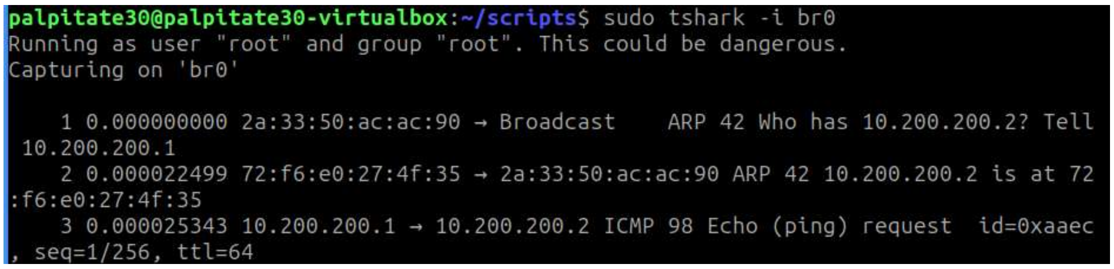


```
jeu. 12 déc. 2024 13:37:47 CET | INFO | attach tap-br-3 to br0
jeu. 12 déc. 2024 13:37:47 CET | INFO | add 10.200.200.3/24 ip address to tap-ns-3
jeu. 12 déc. 2024 13:37:47 CET | INFO | bring up tap-br-3
jeu. 12 déc. 2024 13:37:47 CET | INFO | bing up tap-ns-3
jeu. 12 déc. 2024 13:37:47 CET | INFO | done
```



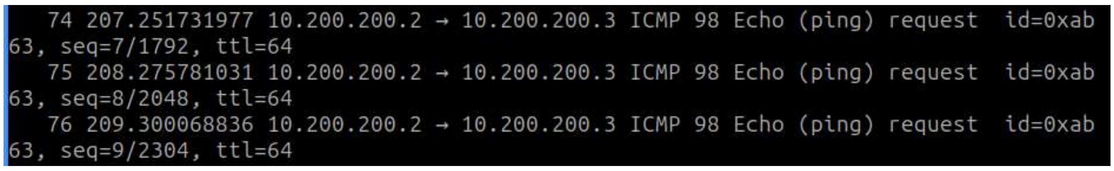
```
palpitate30@palpitate30-virtualbox:~/scripts$ brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.d2a53ea87261       no              tap-br-1
                                                        tap-br-2
                                                        tap-br-3
```

nestn-1 ping nestn-2:



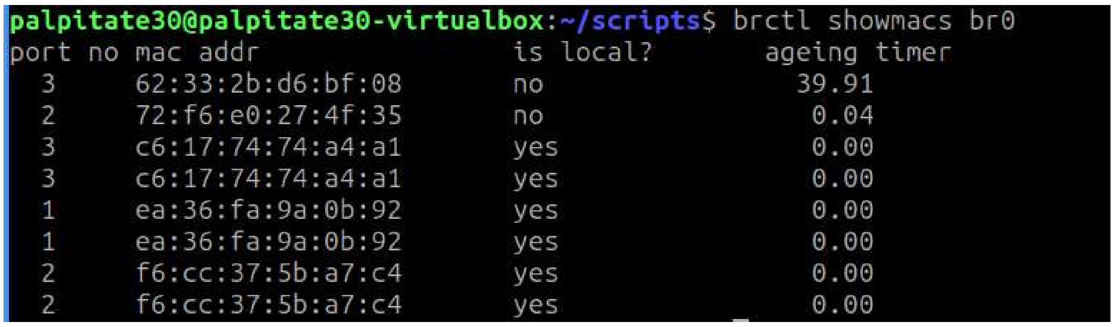
```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo tshark -i br0
Running as user "root" and group "root". This could be dangerous.
Capturing on 'br0'

    1 0.000000000 2a:33:50:ac:ac:90 → Broadcast    ARP 42 Who has 10.200.200.2? Tell
 10.200.200.1
    2 0.000022499 72:f6:e0:27:4f:35 → 2a:33:50:ac:ac:90 ARP 42 10.200.200.2 is at 72
:f6:e0:27:4f:35
    3 0.000025343 10.200.200.1 → 10.200.200.2 ICMP 98 Echo (ping) request  id=0xaaec
, seq=1/256, ttl=64
```

nestn-2 ping nestn-3:



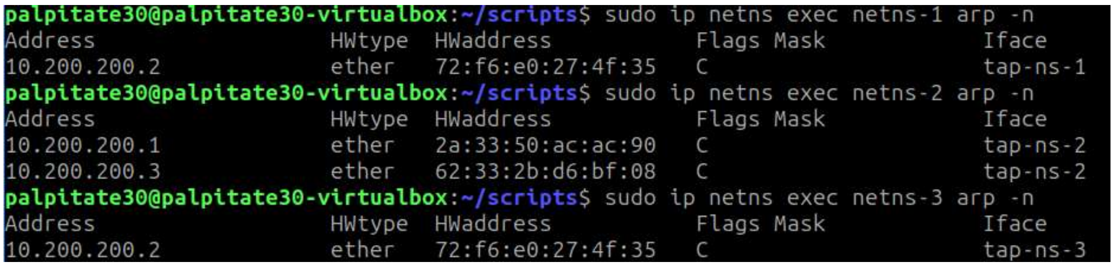
```
   74 207.251731977 10.200.200.2 → 10.200.200.3 ICMP 98 Echo (ping) request  id=0xab
63, seq=7/1792, ttl=64
   75 208.275781031 10.200.200.2 → 10.200.200.3 ICMP 98 Echo (ping) request  id=0xab
63, seq=8/2048, ttl=64
   76 209.300068836 10.200.200.2 → 10.200.200.3 ICMP 98 Echo (ping) request  id=0xab
63, seq=9/2304, ttl=64
```

MAC addresses on br0:



```
palpitate30@palpitate30-virtualbox:~/scripts$ brctl showmacs br0
port no mac addr                is local?       ageing timer
  3     62:33:2b:d6:bf:08       no                39.91
  2     72:f6:e0:27:4f:35       no                 0.04
  3     c6:17:74:74:a4:a1       yes                0.00
  3     c6:17:74:74:a4:a1       yes                0.00
  1     ea:36:fa:9a:0b:92       yes                0.00
  1     ea:36:fa:9a:0b:92       yes                0.00
  2     f6:cc:37:5b:a7:c4       yes                0.00
  2     f6:cc:37:5b:a7:c4       yes                0.00
```

MAC addresses on each netns:



```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ip netns exec netns-1 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.200.200.2             ether   72:f6:e0:27:4f:35   C                     tap-ns-1
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ip netns exec netns-2 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.200.200.1             ether   2a:33:50:ac:ac:90   C                     tap-ns-2
10.200.200.3             ether   62:33:2b:d6:bf:08   C                     tap-ns-2
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ip netns exec netns-3 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.200.200.2             ether   72:f6:e0:27:4f:35   C                     tap-ns-3
```

2.3 Create a virtual sub-network using Open vSwitch virtual switch

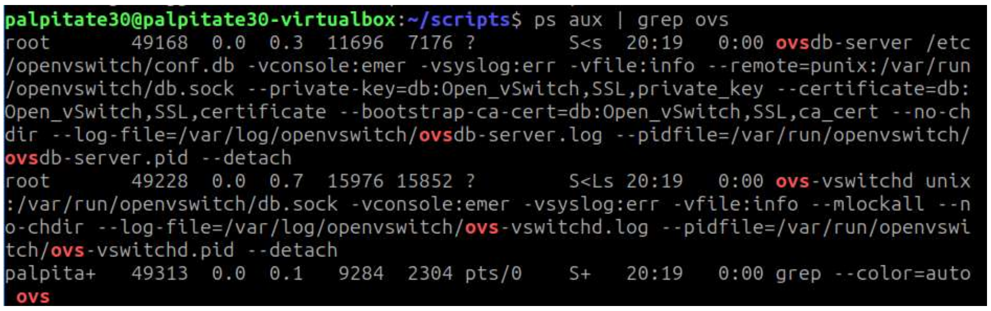
1. What are the main components of OVS ?

   ovs-vswitchd: The daemon responsible for packet forwarding, implementing the data plane.

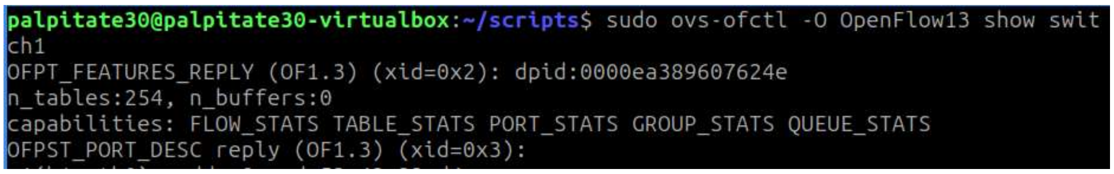
   ovsdb-server: The database server responsible for managing configuration information and interfacing with the control plane.

2. What is the role of these two components: ovs-vswitchd and ovsdb-server ?

- ovs-vswitchd: Handles packet forwarding and switching, enabling high-performance data plane operations.
- ovsdb-server: Maintains the configuration database, allowing administrators and controllers to configure and manage the switch.



```
palpitate30@palpitate30-virtualbox:~/scripts$ ps aux | grep ovs
root       49168  0.0  0.3  11696  7176 ?        S<s  20:19   0:00 ovsdb-server /etc
/openvswitch/conf.db -vconsole:emer -vsyslog:err -vfile:info --remote=punix:/var/run
/openvswitch/db.sock --private-key=db:Open_vSwitch,SSL,private_key --certificate=db:
Open_vSwitch,SSL,certificate --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert --no-ch
dir --log-file=/var/log/openvswitch/ovsdb-server.log --pidfile=/var/run/openvswitch/
ovsdb-server.pid --detach
root       49228  0.0  0.7  15976 15852 ?        S<Ls 20:19   0:00 ovs-vswitchd unix
:/var/run/openvswitch/db.sock -vconsole:emer -vsyslog:err -vfile:info --mlockall --n
o-chdir --log-file=/var/log/openvswitch/ovs-vswitchd.log --pidfile=/var/run/openvswi
tch/ovs-vswitchd.pid --detach
palpita+   49313  0.0  0.1   9284  2304 pts/0    S+   20:19   0:00 grep --color=auto
 ovs
```



```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ovs-ofctl -O OpenFlow13 show swit
ch1
OFPT_FEATURES_REPLY (OF1.3) (xid=0x2): dpid:0000ea389607624e
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS GROUP_STATS QUEUE_STATS
OFPST_PORT_DESC reply (OF1.3) (xid=0x3):
```

3. Which OpenFlow version/s are supported by your OVS virtual switch ?

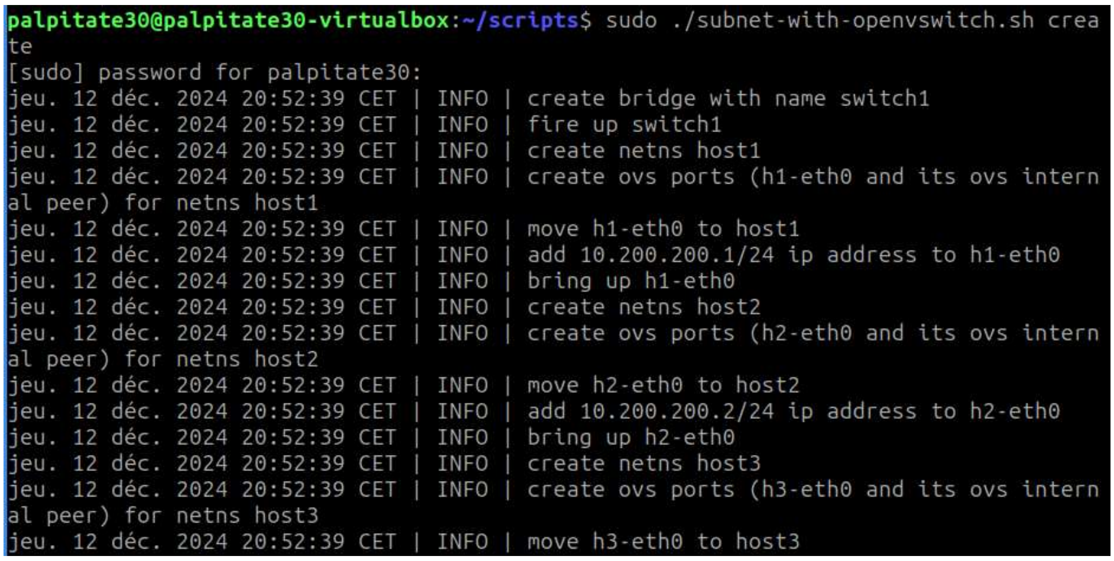
   OpenFlow 1.3

4. Describe the bash script “subnet-with-openvswitch.sh”

   The script is used to create or delete a virtual network, including:

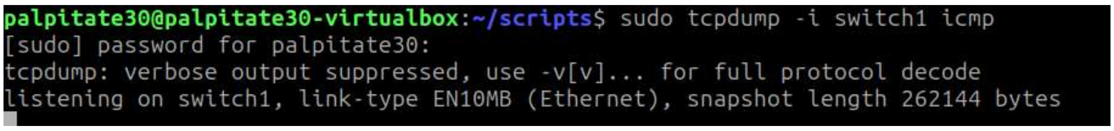
   - 1 virtual switch (switch1).
   - 3 network namespaces (host1, host2, host3).
   - 3 ports connecting the namespaces to the switch, with assigned IPs.

5. Use the provided bash script subnet-with-openvswitch.sh to create the infrastructure



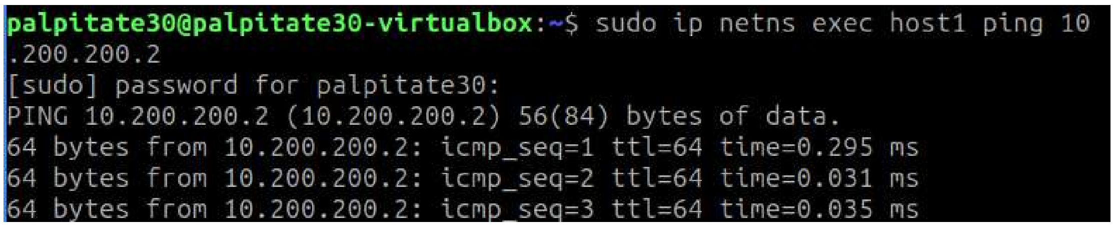
```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ./subnet-with-openvswitch.sh crea
te
[sudo] password for palpitate30:
jeu. 12 déc. 2024 20:52:39 CET | INFO | create bridge with name switch1
jeu. 12 déc. 2024 20:52:39 CET | INFO | fire up switch1
jeu. 12 déc. 2024 20:52:39 CET | INFO | create netns host1
jeu. 12 déc. 2024 20:52:39 CET | INFO | create ovs ports (h1-eth0 and its ovs intern
al peer) for netns host1
jeu. 12 déc. 2024 20:52:39 CET | INFO | move h1-eth0 to host1
jeu. 12 déc. 2024 20:52:39 CET | INFO | add 10.200.200.1/24 ip address to h1-eth0
jeu. 12 déc. 2024 20:52:39 CET | INFO | bring up h1-eth0
jeu. 12 déc. 2024 20:52:39 CET | INFO | create netns host2
jeu. 12 déc. 2024 20:52:39 CET | INFO | create ovs ports (h2-eth0 and its ovs intern
al peer) for netns host2
jeu. 12 déc. 2024 20:52:39 CET | INFO | move h2-eth0 to host2
jeu. 12 déc. 2024 20:52:39 CET | INFO | add 10.200.200.2/24 ip address to h2-eth0
jeu. 12 déc. 2024 20:52:39 CET | INFO | bring up h2-eth0
jeu. 12 déc. 2024 20:52:39 CET | INFO | create netns host3
jeu. 12 déc. 2024 20:52:39 CET | INFO | create ovs ports (h3-eth0 and its ovs intern
al peer) for netns host3
jeu. 12 déc. 2024 20:52:39 CET | INFO | move h3-eth0 to host3
```

6. Start a tshark or a tcpdump on switch1 to listen to ICMP packets



```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo tcpdump -i switch1 icmp
[sudo] password for palpitate30:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on switch1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

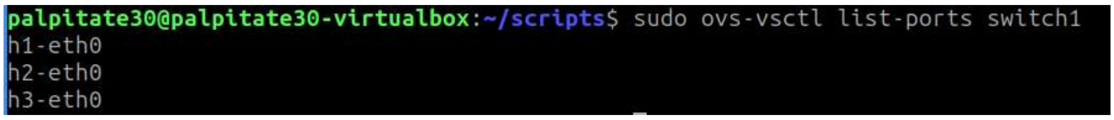
7. Ping a netns-x from a netns-y where x,y are in [1,2,3] (What do you notice ?)



```
palpitate30@palpitate30-virtualbox:~$ sudo ip netns exec host1 ping 10
.200.200.2
[sudo] password for palpitate30:
PING 10.200.200.2 (10.200.200.2) 56(84) bytes of data.
64 bytes from 10.200.200.2: icmp_seq=1 ttl=64 time=0.295 ms
64 bytes from 10.200.200.2: icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from 10.200.200.2: icmp_seq=3 ttl=64 time=0.035 ms
```

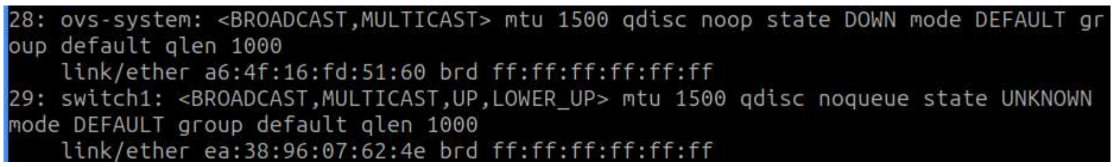
The ping results show that the connection between `host1` and `host2` is successful, indicating that the network is set up correctly. The low latency (0.03ms - 0.3ms) suggests efficient communication within the virtual environment. The TTL value remains constant at 64, meaning the packets are only routed through the virtual switch (`switch1`). This confirms that Open vSwitch is functioning properly, forwarding packets between the namespaces without issues.

8. List ports attached to switch1 using ovs-vsctl

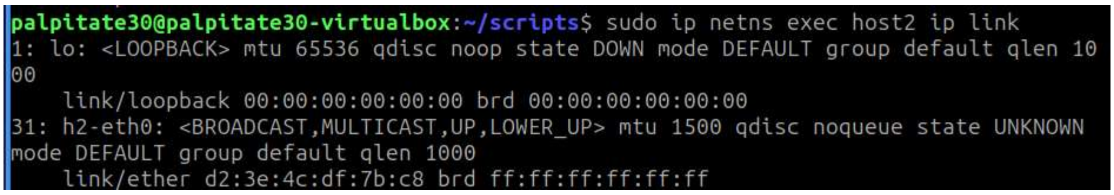


```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ovs-vsctl list-ports switch1
h1-eth0
h2-eth0
h3-eth0
```

9. List network devices using ip link show (What do you notice ?)



```
28: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT gr
oup default qlen 1000
    link/ether a6:4f:16:fd:51:60 brd ff:ff:ff:ff:ff:ff
29: switch1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
mode DEFAULT group default qlen 1000
    link/ether ea:38:96:07:62:4e brd ff:ff:ff:ff:ff:ff
```



```
palpitate30@palpitate30-virtualbox:~/scripts$ sudo ip netns exec host2 ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 10
00
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
31: h2-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
mode DEFAULT group default qlen 1000
    link/ether d2:3e:4c:df:7b:c8 brd ff:ff:ff:ff:ff:ff
```

The loopback interface in each namespace is now enabled for local communications. Additionally, the primary interfaces (h1-eth0, h2-eth0, and h3-eth0) are in the UP, LOWER_UP state with state UNKNOWN, which is expected for virtual interfaces.

10. Which command is used to add an OVS port to an OVS bridge ?

    To add a port named h1-eth0 to a bridge named switch1, the command would be: sudo ovs-vsctl add-port switch1 h1-eth0

3 Testing a Simple SDN Infrastructure

1. In which language mininet is written ?

   Mininet is written in Python for high-level logic and user interface, and partially in C for performance optimization and low-level networking operations.

2. What does mininet use to create virtual hosts, switches and links ?

   Mininet uses Linux namespaces, Open vSwitch (OVS), and virtual Ethernet pairs (veth pairs) to create virtual hosts, switches, and links. Namespaces isolate hosts, OVS simulates switches, and veth pairs enable communication between hosts and switches.

3. What is a Provider in ONOS ?

   In ONOS, a Provider is a module or abstraction layer responsible for communicating with the underlying network devices through southbound interfaces (e.g., OpenFlow or Netconf). It collects the network's state and applies control logic to these devices, enabling SDN functionality.

4. Which port is used by OpenFlow ?

   OpenFlow uses TCP port 6653 as the standard port, and TCP port 6633 was commonly used in earlier versions.

3.1 Mininet Setup

1. Using the help of the mininet CLI: What are the supported software switches ? ans which is the default one ?

   Mininet supports the following software switches:

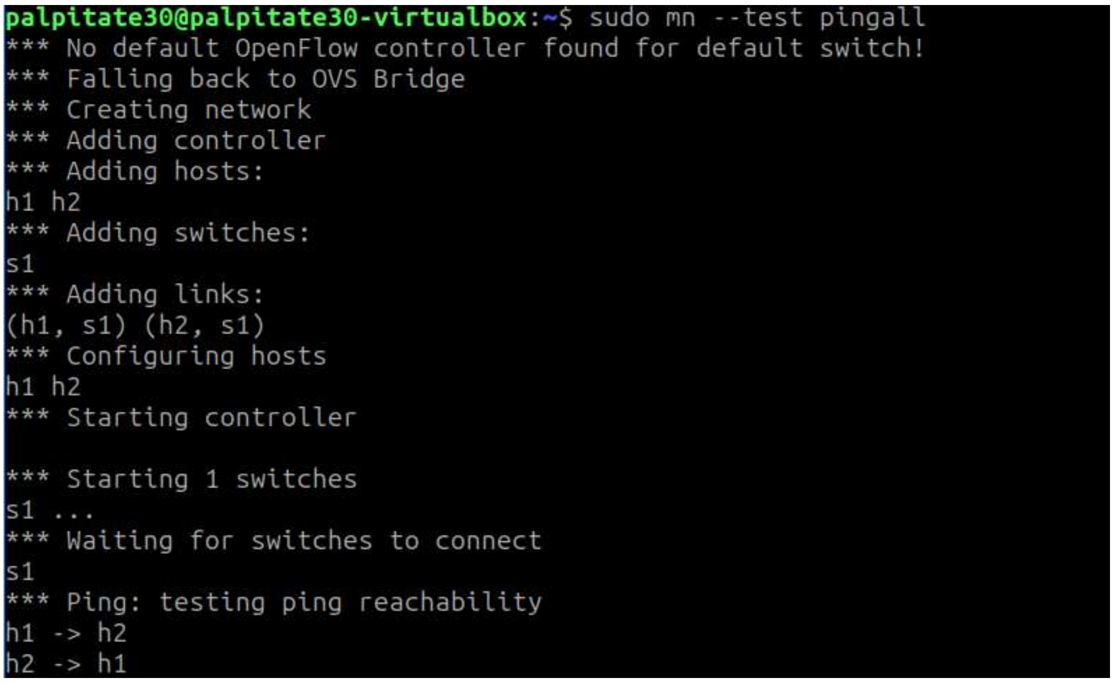
   Open vSwitch (OVS)

   Linux Bridge

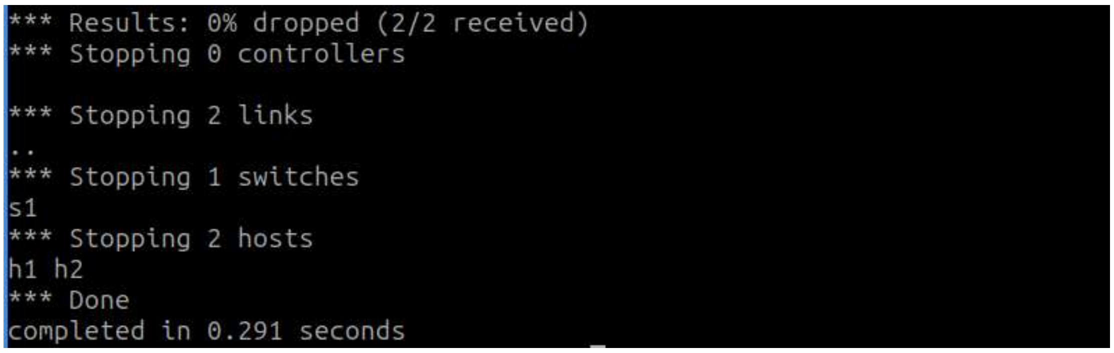
   Indigo Virtual Switch (IVS)

   The default software switch in Mininet is Open vSwitch (OVS).

2. What this test is doing ?



```
palpitate30@palpitate30-virtualbox:~$ sudo mn --test pingall
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
```



```
*** Results: 0% dropped (2/2 received)
*** Stopping 0 controllers

*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 0.291 seconds
```

(1) Creates a simple network topology with two hosts (h1, h2) and one switch (s1).

(2) Tests connectivity between the hosts using ping.

(3) Confirms successful communication with 0% packet loss.

(4) Cleans up the network after the test.

3.2 ONOS Setup

1. What is the protocol or application running on each of those ports ?

   (1) Port 6653:

   Protocol/Application: OpenFlow

   Used for communication between the ONOS controller and OpenFlow-enabled switches. It is the standard OpenFlow protocol port.

   (2) Port 6640:

   Protocol/Application: OVSDB (Open vSwitch Database Management Protocol)

   Used for managing and configuring Open vSwitch (OVS) instances.

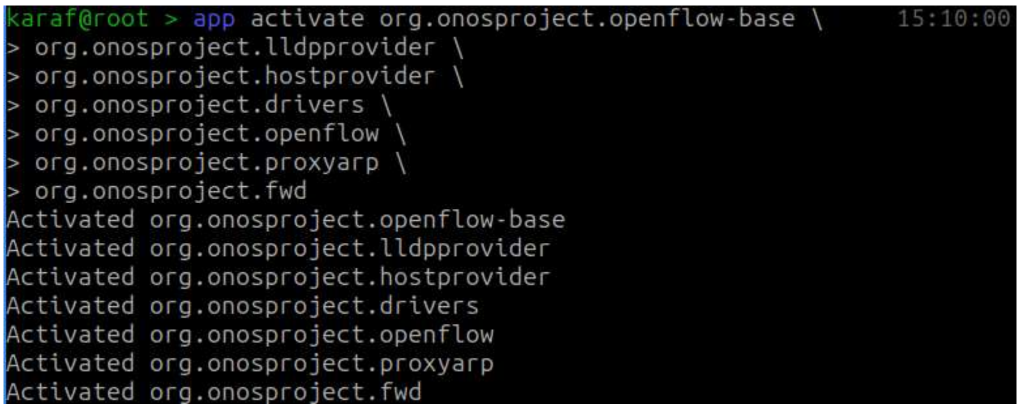
   (3) Port 8181:

   Protocol/Application: ONOS Web UI and REST API

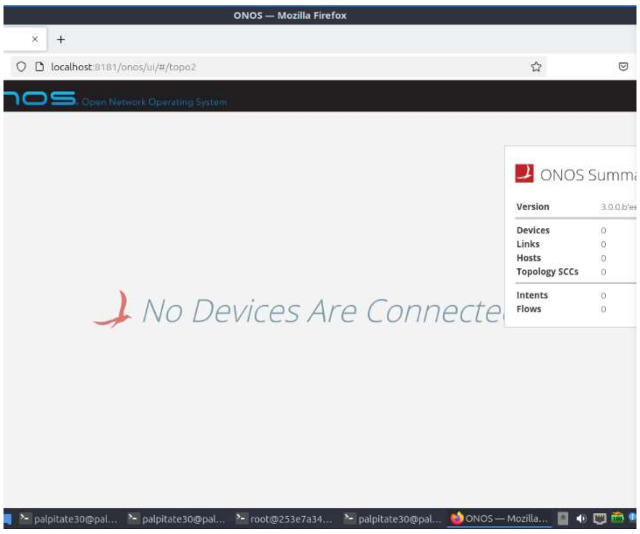
   Hosts the ONOS web-based graphical user interface (GUI) and REST API endpoints.

   (4) Port 8101:

   Protocol/Application: ONOS CLI via SSH



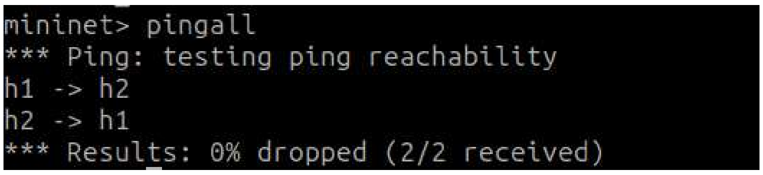
```
karaf@root > app activate org.onosproject.openflow-base \     15:10:00
> org.onosproject.lldpprovider \
> org.onosproject.hostprovider \
> org.onosproject.drivers \
> org.onosproject.openflow \
> org.onosproject.proxyarp \
> org.onosproject.fwd
Activated org.onosproject.openflow-base
Activated org.onosproject.lldpprovider
Activated org.onosproject.hostprovider
Activated org.onosproject.drivers
Activated org.onosproject.openflow
Activated org.onosproject.proxyarp
Activated org.onosproject.fwd
```



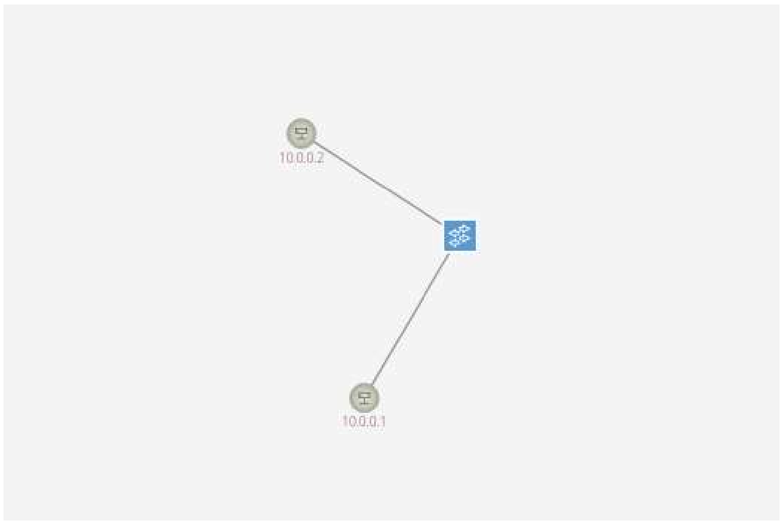
ONOS Summ

Version	3.0.0.b'e
Devices	0
Links	0
Hosts	0
Topology SCCs	0
Intents	0
Flows	0

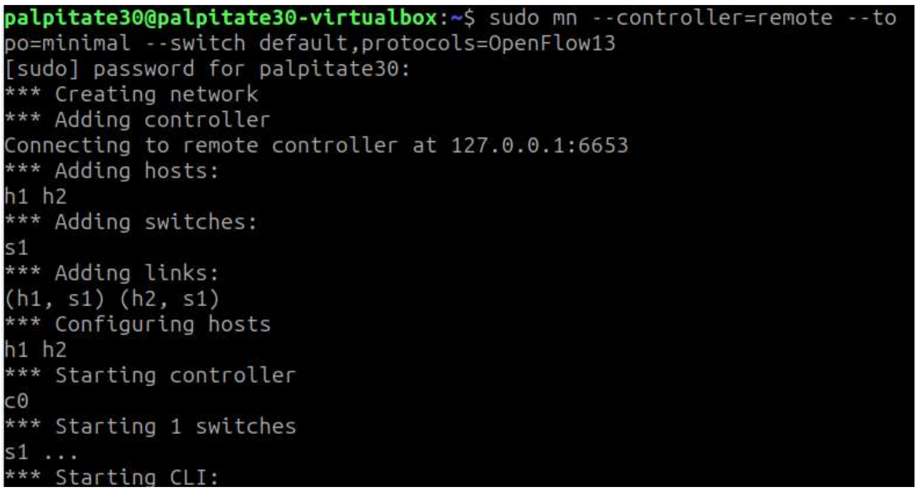
3.3 Network topologies



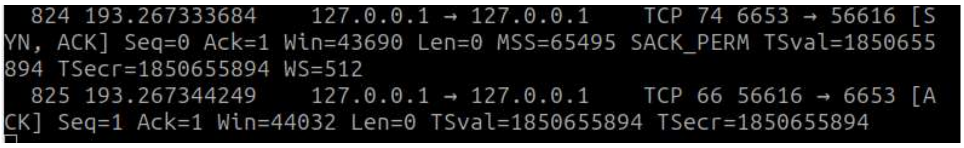
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```



Launch a mininet topology of your choice (minimal may be a good choice):



```
palpitate30@palpitate30-virtualbox:~$ sudo mn --controller=remote --to
po=minimal --switch default,protocols=OpenFlow13
[sudo] password for palpitate30:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** 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:
```



```
  824 193.267333684    127.0.0.1 → 127.0.0.1    TCP 74 6653 → 56616 [S
YN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM TSval=1850655
894 TSecr=1850655894 WS=512
  825 193.267344249    127.0.0.1 → 127.0.0.1    TCP 66 56616 → 6653 [A
CK] Seq=1 Ack=1 Win=44032 Len=0 TSval=1850655894 TSecr=1850655894
```

1. What transport protocol is used by OpenFlow ?

   The transport protocol used by OpenFlow is TCP.

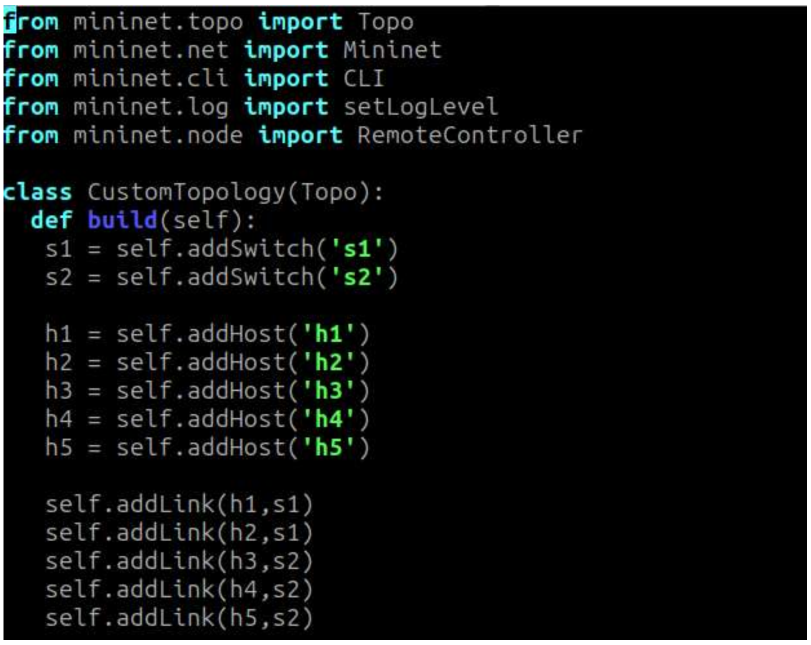
2. Is it the switch or the controller that initiated the session ?

   The session is initiated by the switch.

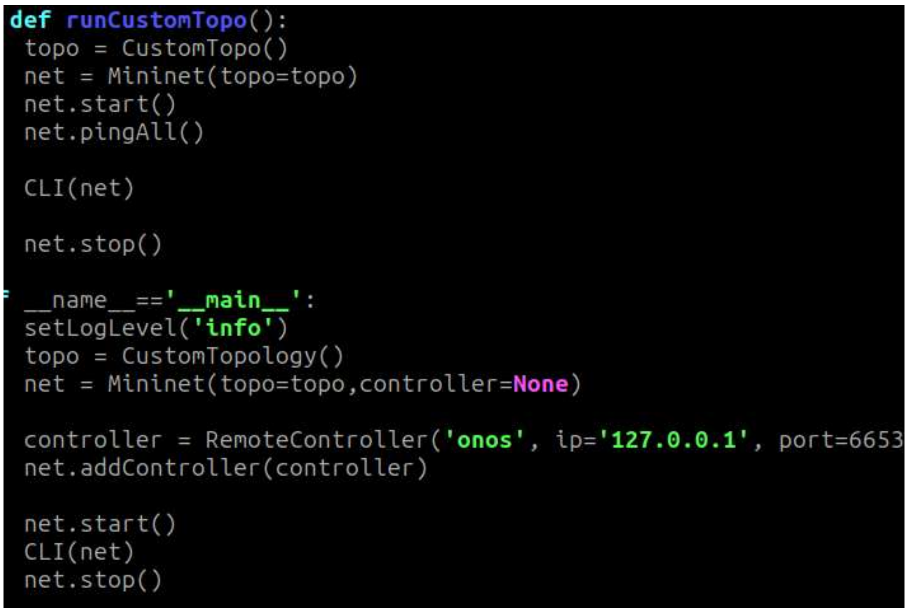
3. What is the first OpenFlow message ?

   The first OpenFlow message is typically a HELLO message, which establishes the connection between the switch and the controller.

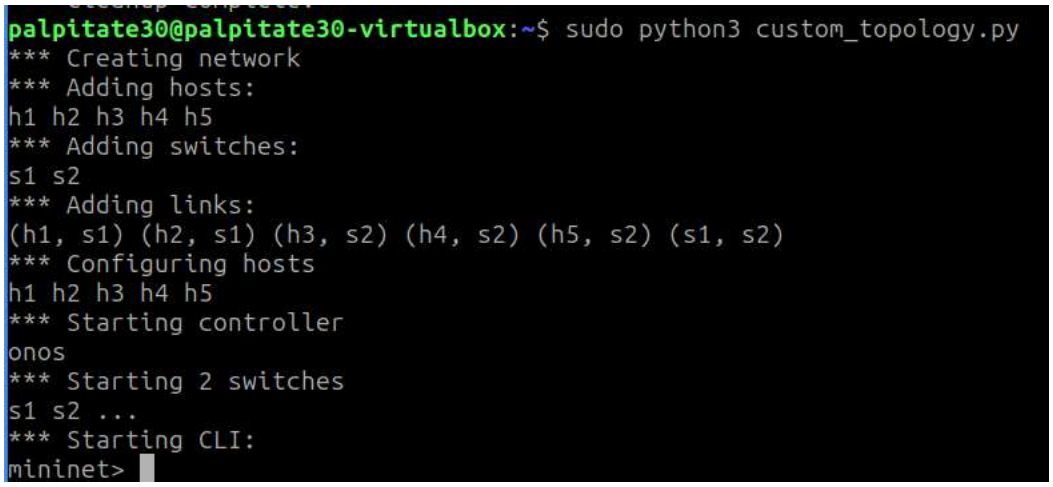
3.4 Create the custom topology



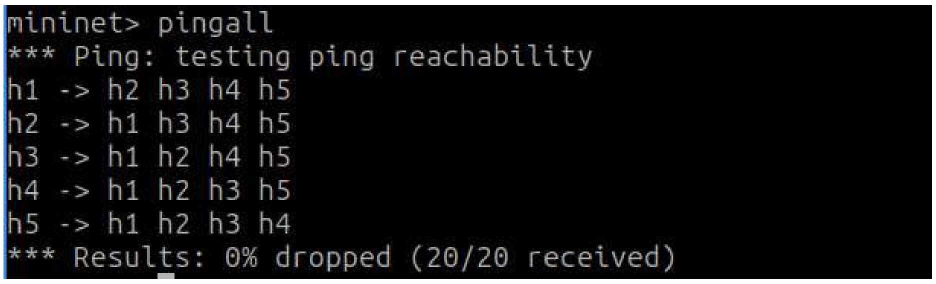
```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.node import RemoteController

class CustomTopology(Topo):
  def build(self):
   s1 = self.addSwitch('s1')
   s2 = self.addSwitch('s2')

   h1 = self.addHost('h1')
   h2 = self.addHost('h2')
   h3 = self.addHost('h3')
   h4 = self.addHost('h4')
   h5 = self.addHost('h5')

   self.addLink(h1,s1)
   self.addLink(h2,s1)
   self.addLink(h3,s2)
   self.addLink(h4,s2)
   self.addLink(h5,s2)
```



```
def runCustomTopo():
 topo = CustomTopo()
 net = Mininet(topo=topo)
 net.start()
 net.pingAll()

 CLI(net)

 net.stop()

 __name__=='__main__':
 setLogLevel('info')
 topo = CustomTopology()
 net = Mininet(topo=topo,controller=None)

 controller = RemoteController('onos', ip='127.0.0.1', port=6653
 net.addController(controller)

 net.start()
 CLI(net)
 net.stop()
```



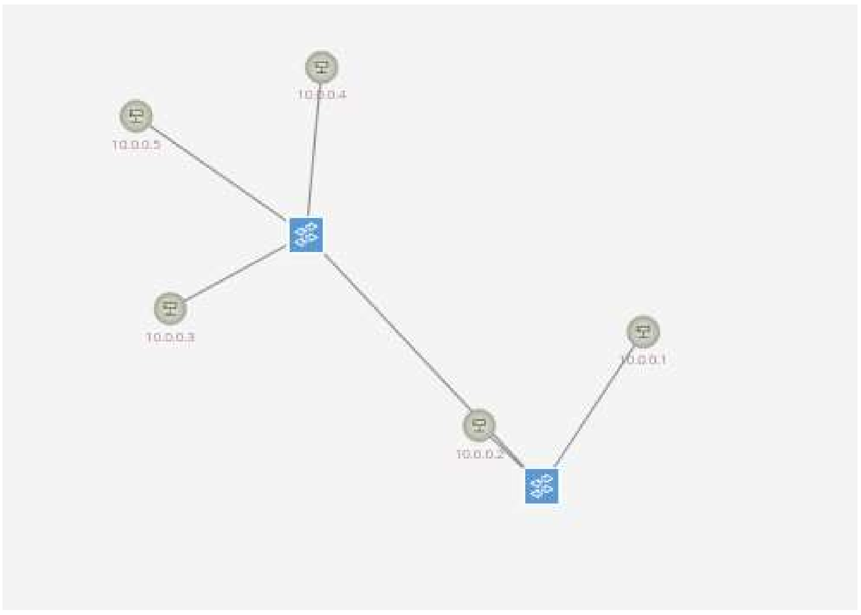
```
palpitate30@palpitate30-virtualbox:~$ sudo python3 custom_topology.py
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
onos
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```



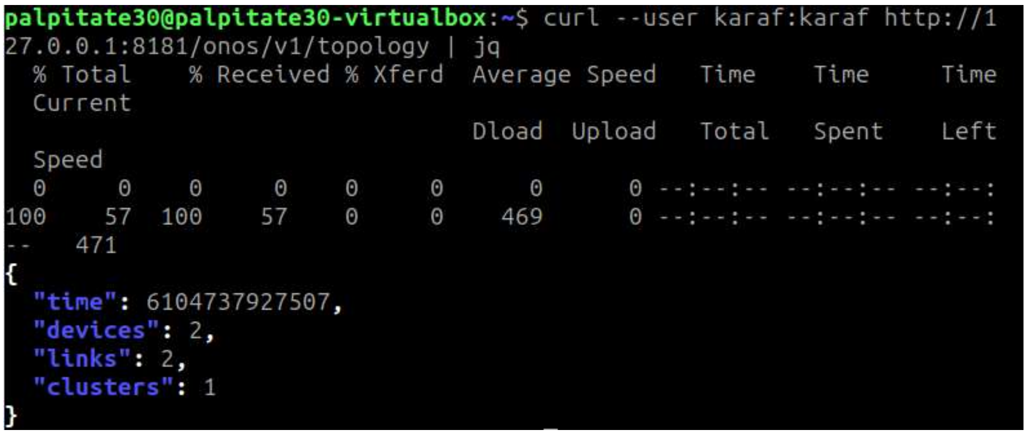
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
```

3.5 ONOS REST API

Using the OpenAPI web UI:



Using cURL:



```
palpitate30@palpitate30-virtualbox:~$ curl --user karaf:karaf http://1
27.0.0.1:8181/onos/v1/topology | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
  Current
                                 Dload  Upload   Total   Spent    Left
  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:
100    57  100    57    0     0    469      0 --:--:-- --:--:-- --:--:
--   471
{
  "time": 6104737927507,
  "devices": 2,
  "links": 2,
  "clusters": 1
}
```