

Infrastructures Réseaux

Ouarafana Badr / Joulain Vincent

Avril 2023

Contents

1 L'architecture réseau	3
2 Mise en place du tunnel	6
3 Protocole L2TPv3 et technologie VxLAN	8
3.1 Layer 2 Tunneling Protocol	8
3.2 Virtual eXtensible Local Area Network	8
3.3 Différences	8
3.4 Chiffrement du traffic	9
3.5 Open vSwitch et VxLAN sous Linux	9
3.6 Et MPLS dans tout ça ?	9
4 Modes IP/UDP L2TPv3 et Tunnel GRE	9
4.1 Analyse de l'encapsulation IP	9
4.2 Analyse de l'encapsulation UDP	11
4.3 Mise en place du tunnel GRE	12
5 Chiffrement IPsec	14
5.1 Comparaison des performances avec Iperf	15
6 Accès à internet	15
6.1 Redirection du trafic DHCP	16

7	Gestion du traffic entre les VLAN	17
7.1	IP tables	17
7.2	Policy Routing	18

1 L'architecture réseau

En premiers temps une architecture réseau avec les “namespaces” à été mise en place avec les **Vlan** sans les tunnels :

On commence par créer les namespaces, les switchs et les liens:

```
ip netns add r1
ip netns add r2
ip netns add rA
ip netns add rB
ip netns add p1
ip netns add p2
ip netns add p3
ip netns add p4
ip netns add internet

ovs-vsctl add-br internet
ovs-vsctl add-br resB
ovs-vsctl add-br resC
ovs-vsctl add-br resD
ovs-vsctl add-br resE

#r1
ip link add r1-eth0 type veth peer name resC-r1
ip link add r1-eth1 type veth peer name resD-r1
#rA
ip link add rA-eth0 type veth peer name resC-rA
ip link add rA-eth1 type veth peer name internet-rA
#rB
ip link add rB-eth0 type veth peer name resB-rB
ip link add rB-eth1 type veth peer name internet-rB
#r2
ip link add r2-eth0 type veth peer name resB-r2
ip link add r2-eth1 type veth peer name resE-r2
#p1
ip link add p1-eth0 type veth peer name resE-p1
#p2
ip link add p2-eth0 type veth peer name resE-p2
#p3
ip link add p3-eth0 type veth peer name resD-p3
#p4
ip link add p4-eth0 type veth peer name resD-p4
```

Pour automatiser le travail, les noms des interfaces seront stockés dans tes tableaux

```
interfaces_root=( "resC-r1" "resD-r1" "resC-rA" "internet-rA" "resB-rB" "internet-rB" "resB-r2" \
"resE-r2" "resE-p1" "resE-p2" "resD-p3" "resD-p4")

interfaces_netns=( "r1-eth0" "r1-eth1" "rA-eth0" "rA-eth1" "rB-eth0" "rB-eth1" "r2-eth0" "r2-eth1" \
"p1-eth0" "p2-eth0" "p3-eth0" "p4-eth0")
```

Avec l'instruction suivante, le nom de la hôte ou le réseau sera extrait de l'interface dans une nouvelle chaîne de caractère.

```
$(echo $interface | sed 's/-.*//')
```

Afin qu'on puisse itérer sur toutes les interface, pour accrocher les liens avec les hôtes:

```

for interface in "${interfaces_netns[@]}"
do
host=$(echo $interface | sed 's/-.*//')
ip link set $interface netns $host
done
# activer les interfaces du namespace root
for interface_res in "${interfaces_root[@]}"
do
res=$(echo $interface_res | sed 's/-.*//')
ovs-vsctl add-port $res $interface_res
done
# activer les interfaces des namespaces
for interface in "${interfaces_root[@]}"
do
ip link set dev $interface up
done
# activer les interfaces des names spaces
for host in r1 r2 rA rB p1 p2 p3 p4
do
ip netns exec $host ip link set dev lo up
done
# activer les interfaces de loopback
for interface in "${interfaces_netns[@]}"
do
host=$(echo $interface | sed 's/-.*//')
ip netns exec $host ip link set dev $interface up
done

```

Après avoir mis en place toute l'architecture réseau, nous allons procéder à la configuration des hôtes.

```

#configuration ip des routeurs
#r1
ip netns exec r1 ip addr add 172.16.1.253/24 dev r1-eth0
ip netns exec r1 ip r add default via 172.16.1.254
#r2
ip netns exec r2 ip addr add 172.16.2.253/24 dev r2-eth0
ip netns exec r2 ip r add default via 172.16.2.254
#rA
ip netns exec rA ip addr add 172.16.1.254/24 dev rA-eth0
ip netns exec rA ip addr add 10.87.0.1/24 dev rA-eth1
ip netns exec rA ip route add 172.16.2.0/24 via 10.87.0.2
ip netns exec rA ip route add 192.168.100.0/24 via 10.87.0.2
ip netns exec rA ip route add 192.168.200.0/24 via 10.87.0.2
#rB
ip netns exec rB ip addr add 172.16.2.254/24 dev rB-eth0
ip netns exec rB ip addr add 10.87.0.2/24 dev rB-eth1
ip netns exec rB ip route add 172.16.1.0/24 via 10.87.0.1
ip netns exec rB ip route add 192.168.100.0/24 via 10.87.0.1
ip netns exec rB ip route add 192.168.200.0/24 via 10.87.0.1
# ip forwarding
ip netns exec r1 sysctl net.ipv4.conf.all.forwarding=1
ip netns exec r2 sysctl net.ipv4.conf.all.forwarding=1
ip netns exec rA sysctl net.ipv4.conf.all.forwarding=1
ip netns exec rB sysctl net.ipv4.conf.all.forwarding=1

#Creation Vlan
#P1
ip netns exec p1 ip link add link p1-eth0 name p1-eth0.100 type vlan id 100
ip netns exec p1 ip addr add 192.168.100.1/24 brd 192.168.100.255 dev p1-eth0.100
ip netns exec p1 ip link set dev p1-eth0.100 up
#P2
ip netns exec p2 ip link add link p2-eth0 name p2-eth0.200 type vlan id 200
ip netns exec p2 ip addr add 192.168.200.1/24 brd 192.168.200.255 dev p2-eth0.200
ip netns exec p2 ip link set dev p2-eth0.200 up
#P3

```

```

ip netns exec p3 ip link add link p3-eth0 name p3-eth0.100 type vlan id 100
ip netns exec p3 ip addr add 192.168.100.2/24 brd 192.168.100.255 dev p3-eth0.100
ip netns exec p3 ip link set dev p3-eth0.100 up
#P4
ip netns exec p4 ip link add link p4-eth0 name p4-eth0.200 type vlan id 200
ip netns exec p4 ip addr add 192.168.200.2/24 brd 192.168.200.255 dev p4-eth0.200
ip netns exec p4 ip link set dev p4-eth0.200 up
#r1
ip netns exec r1 ip link add link r1-eth1 name r1-eth1.100 type vlan id 100
ip netns exec r1 ip addr add 192.168.100.254/24 dev r1-eth1.100
ip netns exec r1 ip l set dev r1-eth1.100 up

ip netns exec r1 ip link add link r1-eth1 name r1-eth1.200 type vlan id 200
ip netns exec r1 ip addr add 192.168.200.254/24 dev r1-eth1.200
ip netns exec r1 ip l set dev r1-eth1.200 up
#r2
ip netns exec r2 ip link add link r2-eth1 name r2-eth1.200 type vlan id 200
ip netns exec r2 ip addr add 192.168.200.253/24 dev r2-eth1.200
ip netns exec r2 ip l set dev r2-eth1.200 up

ip netns exec r2 ip link add link r2-eth1 name r2-eth1.100 type vlan id 100
ip netns exec r2 ip addr add 192.168.100.253/24 dev r2-eth1.100
ip netns exec r2 ip l set dev r2-eth1.100 up

```

Et au final on obtient les résultats suivant :

Nota : Comme le tunnel entre les deux routeurs n'a pas encore été mis en place, le ping ne peut pas être établi entre **r1** et **p1**.

```

tmux 21x56
+ _INFRARESS git:(master) X netns r1
user@user-vm:~/Desktop/INFRARESS [r1] ip a
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
    inet 172.17.0.1/128 scope host
        valid_lft forever preferred_lft forever
2: p3-eth0.100@p3-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 52:6a:5e:29:b4:d0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.254/24 brd 192.168.100.255 scope global p3-eth0.100
        valid_lft forever preferred_lft forever
        inet6 fe80::50a:5eff:fe29:b4d0/64 scope link
            valid_lft forever preferred_lft forever
3: r1-eth1.200@r1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 52:6a:5e:29:b4:d0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.254/24 brd 192.168.200.255 scope global r1-eth1.200
        valid_lft forever preferred_lft forever
        inet6 fe80::50a:5eff:fe29:b4d0/64 scope link
            valid_lft forever preferred_lft forever
137: r1-eth1@r1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 12:ea:42:c0:29:b4 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.16.1.251/24 brd 172.16.1.255 scope global r1-eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::10a4:42ff:fedc:e43/64 scope link
            valid_lft forever preferred_lft forever
138: r1-eth1@r1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 52:6a:5e:29:b4:d0 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.254/24 brd 192.168.100.255 scope global r1-eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::50a:5eff:fe29:b4:d0/64 scope link
            valid_lft forever preferred_lft forever
139: r1-eth1@r1-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 52:6a:5e:29:b4:d0 brd ff:ff:ff:ff:ff:ff
tcpdump: listening on r1-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
02:46:43.124465 [ether:00:0c:00:10:ca] > [ether:00:00:00:00:00], ethertype 802.1Q (0x8100), length 46: [vlan 100, p 0, ethertype ARP], Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.100.254 tell 192.168.100.6, len gth 28
02:46:43.124495 [ether:00:0c:00:10:ca] > [ether:00:00:00:00:00], ethertype 802.1Q (0x8100), length 46: vlan 100, p 0, ethertype ARP, Ethernet (len 6), IPv4 (len 4), Reply who-has 192.168.100.254 ls-at 52:6a:5e:29:b4:d0, length 28
02:46:43.124567 [ether:00:0c:00:10:ca] > [ether:00:00:00:00:00], ethertype 802.1Q (0x8100), length 102: [vlan 100, p 0, ethertype ICMP], (tos 0x0, ttl 64, id 31368, offset 0, flags [DF], proto ICMP (1)), seq 1, length 64
02:46:43.124579 [ether:00:0c:00:10:ca] > [ether:00:00:00:00:00], ethertype 802.1Q (0x8100), length 102: vlan 100, p 0, ethertype ICMP, (tos 0x0, ttl 64, id 21908, offset 0, flags [none], proto ICMP (1), length 64)
02:46:48.291915 52:6a:5e:29:b4:d0 > [ether:00:00:00:00:00], ethertype 802.1Q (0x8100), length 46: vlan 100, p 0, ethertype ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.100.6 tell 192.168.100.254, len gth 28
02:46:48.292533 [ether:00:00:00:00:00] > [ether:00:0c:00:10:ca], ethertype 802.1Q (0x8100), length 46: vlan 100, p 0, ethertype ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.100.6 ls-at f6:4a:ec:c6:10:ca, length 28

```

On remarque que le ping est passé de **r1** à **p3**, et les trames sont taggées avec l'étiquette **VLAN100**.

2 Mise en place du tunnel

Afin de mettre en place le tunnel, les configurations concernant les Vlan pour **r1** et **r2** doivent être supprimées (commentées), et on ajoute le tunnel liant les deux réseaux de la manière suivante :

```
ip netns exec r1 ip l2tp add tunnel remote 172.16.2.253 local 172.16.1.253 encaps ip tunnel_id 3000 peer_tunnel_id
↪ 4000
ip netns exec r1 ip l2tp add session tunnel_id 3000 session_id 1000 peer_session_id 2000
ip netns exec r1 brctl addbr tunnel
ip netns exec r1 brctl addif tunnel l2tpeth0
ip netns exec r1 brctl addif tunnel r1-eth1
ip netns exec r1 ip link set l2tpeth0 up
ip netns exec r1 ip link set tunnel up
ip netns exec r1 ip link add link tunnel name tunnel.100 type vlan id 100
ip netns exec r1 ip link set tunnel.100 up
ip netns exec r1 ip link add link tunnel name tunnel.200 type vlan id 200
ip netns exec r1 ip link set tunnel.200 up
ip netns exec r1 ip addr add 192.168.100.254/24 dev tunnel.100
ip netns exec r1 ip addr add 192.168.200.254/24 dev tunnel.200

ip netns exec r2 ip l2tp add tunnel remote 172.16.1.253 local 172.16.2.253 encaps ip tunnel_id 4000 peer_tunnel_id
↪ 3000
ip netns exec r2 ip l2tp add session tunnel_id 4000 session_id 2000 peer_session_id 1000
ip netns exec r2 brctl addbr tunnel
ip netns exec r2 brctl addif tunnel l2tpeth0
ip netns exec r2 brctl addif tunnel r2-eth1
ip netns exec r2 ip link set l2tpeth0 up
ip netns exec r2 ip link set tunnel up
ip netns exec r2 ip link add link tunnel name tunnel.100 type vlan id 100
ip netns exec r2 ip link set tunnel.100 up
ip netns exec r2 ip link add link tunnel name tunnel.200 type vlan id 200
ip netns exec r2 ip link set tunnel.200 up
ip netns exec r2 ip addr add 192.168.100.253/24 dev tunnel.100
ip netns exec r2 ip addr add 192.168.200.253/24 dev tunnel.200
```

Et on obtient les configurations suivantes pour les deux routeurs :

```
user@user-vn:~/Desktop/INFRA_NESS [r1] ip a
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 brd 0.0.0.0 scope host lo
        valid_lft forever preferred_lft forever
        inet6 fe80::1%lo brd ff:ff:ff:ff:ff:ff scope link
            valid_lft forever preferred_lft forever
2: l2tpeth0: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc fq_codel master tunnel state UNKNOWN group default qlen 1000
    link/ether b6:79:e0:29:1d:e5 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::b479:effff:fe29:1de5/64 scope link
        valid_lft forever preferred_lft forever
3: tunnel.100: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UP group default qlen 1000
    link/ether b6:79:e0:29:1d:e5 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::b479:effff:fe29:1de5/64 scope link
        valid_lft forever preferred_lft forever
4: tunnel.100@tunnel: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UP group default qlen 1000
    link/ether b6:79:e0:29:1d:e5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.254/24 brd 192.168.100.255 scope global tunnel.100
        valid_lft forever preferred_lft forever
        inet6 fe80::b479:effff:fe29:1de5/64 scope link
            valid_lft forever preferred_lft forever
5: tunnel.200: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UP group default qlen 1000
    link/ether b6:79:e0:29:1d:e5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.254/24 brd 192.168.200.255 scope global tunnel.200
        valid_lft forever preferred_lft forever
        inet6 fe80::b479:effff:fe29:1de5/64 scope link
            valid_lft forever preferred_lft forever
197: r1-eth0@fif06: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether d2:9a:47:37:56:44 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.16.1.251/24 brd 172.16.1.255 scope global r1-eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::d9a:47fff:fe37:5604/64 scope link
            valid_lft forever preferred_lft forever
199: r1-eth1@fif98: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master tunnel state UP group default qlen 1000
    link/ether 4e:c0:af:4e:74:c6 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.254/24 brd 192.168.100.255 scope global r1-eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::c0af:4eff:fe74:c6bb/64 scope link
            valid_lft forever preferred_lft forever
user@user-vn:~/Desktop/INFRA_NESS [r2] ip a
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 brd 0.0.0.0 scope host lo
        valid_lft forever preferred_lft forever
        inet6 fe80::1%lo brd ff:ff:ff:ff:ff:ff scope link
            valid_lft forever preferred_lft forever
2: l2tpeth0: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc fq_codel master tunnel state UNKNOWN group default qlen 1000
    link/ether ae:71:ff:07:9a:90 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::act71:ffff:fe07:9a90/64 scope link
        valid_lft forever preferred_lft forever
3: tunnel.100: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UP group default qlen 1000
    link/ether ae:71:ff:07:9a:90 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::act71:ffff:fe07:9a90/64 scope link
        valid_lft forever preferred_lft forever
4: tunnel.100@tunnel: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UP group default qlen 1000
    link/ether ae:71:ff:07:9a:90 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.253/24 brd 192.168.100.255 scope global tunnel.100
        valid_lft forever preferred_lft forever
        inet6 fe80::act71:ffff:fe07:9a90/64 scope link
            valid_lft forever preferred_lft forever
5: tunnel.200: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UP group default qlen 1000
    link/ether ae:71:ff:07:9a:90 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.253/24 brd 192.168.200.255 scope global tunnel.200
        valid_lft forever preferred_lft forever
        inet6 fe80::act71:ffff:fe07:9a90/64 scope link
            valid_lft forever preferred_lft forever
209: r2-eth0@fif08: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 76:00:b3:fe:bb:17 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.16.2.253/24 brd 172.16.2.255 scope global r2-eth0
        valid_lft forever preferred_lft forever
        inet6 fe80::7400:b3ff:fe1b:bb17/64 scope link
            valid_lft forever preferred_lft forever
211: r2-eth1@fif10: <NOQUEUE,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master tunnel state UP group default qlen 1000
    link/ether 60:0c:6cff:fe11:4a:eb brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.100.254/24 brd 192.168.100.255 scope global r2-eth1
        valid_lft forever preferred_lft forever
user@user-vn:~/Desktop/INFRA_NESS [r2]
```

Afin d'analyser les requêtes ARP, on écoute sur les deux routeurs **r1** et **r2** et on effectue un ping de **p2** à **p1**, et on remarque aussi que la trame est étiquetée par le Vlan 200, elle passe par les deux routeurs.

Pour tester le serveur **DHCP**, la configuration IP de l'hôte **p1** sera supprimée, puis une demande sera envoyée afin d'obtenir une nouvelle configuration. On remarquera alors que la machine obtiendra une nouvelle adresse IP.

La commande pour faire un serveur **DHCP** est la suivante :

```
sudo dnsmasq -d -z -i tunnel.100 -F 192.168.100.1,192.168.100.10,255.255.255.0  
sudo dnsmasq -d -z -i tunnel.200 -F 192.168.200.1,192.168.200.10,255.255.255.0
```

Et avec cette configuration, une communication TCP sera initiée entre **r1** et **p1**, et on obtient le résultat suivant :

3 Protocole L2TPv3 et technologie VxLAN

3.1 Layer 2 Tunneling Protocol

Le Layer 2 Tunneling Protocol (L2TP) permet de créer un tunnel afin de faire transiter des protocoles de couche 2 (ethernet par exemple) par dessus un réseau de couche 3 (Réseau IP).

3.2 Virtual eXtensible Local Area Network

Les Virtual eXtensible Local Area Network (VxLAN) permettent d'inter-connecter des machines/machines virtuelles sur un réseau de couche 2 au travers de la couche 3 au travers des paquets UDP. En d'autres termes, il nous est possible de simuler un unique réseau local composé de plusieurs réseaux distincts par le moyen de tunnels. Dans l'objectif d'administrer le réseau, les VXLAN permettent une segmentation sur 24 bits (contrairement aux VLAN limitées à 16)

3.3 Différences

A première vue les deux solutions semblent proposer la même chose, toutefois elles présentent des cas d'utilisation bien différents. Les VxLAN sont plus scalables et permettent la segmentation en 16 millions de réseaux virtuels contrairement à l'utilisation des VLAN par dessus L2TPv3, nous serions limité par la segmentation offerte par les VLAN (4096 réseaux virtuels) en plus de devoir gérer manuellement chacun des tunnels mis en place.

L'objectif du protocole L2TPv3 est simplement la mise en place d'un tunnel supportant les protocoles de couche 2 tandis que l'utilisation des VxLAN à pour objectif l'administration de réseaux de plus grande échelle tels que des data centers, cloud storage...

L2TPv3 est plus simple à mettre en place et offre plus de libertés mais devient vite désuet sur des architectures de plus grande ampleur.

3.4 Chiffrement du traffic

Les technologies L2TPv3 et VxLAN n'offrent pas de chiffrement des données par défaut. Il est donc fortement conseillé de coupler l'utilisation de ces tunnels avec des solutions proposant de la sécurité.

Plusieurs sont envisageables :

- IPsec (Internet Protocol security)
- SSL/TLS
- MACsec (Media Access Control security)

IPSec et SSL/TLS permettent d'assurer la sécurité par chiffrement au niveau de la couche 3 et MACsec au niveau de la couche 2.

3.5 Open vSwitch et VxLAN sous Linux

Nous pouvons nous référer à la documentation d'open vSwitch afin de mettre en place un Vxlan sous Linux :

```
# création du Switch OVS
ovs-vsctl add-br br0

# création du VxLAN (définition des addresses et des ports à l'autre bout du tunnel, à faire aux niveau de deux
↪ entrées)
ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=vxlan \
options:remote_ip=[IP_ADDRESS] options:key=flow options:dst_port=[PORT]
```

3.6 Et MPLS dans tout ça ?

Multiprotocol label switching (MPLS) est une technologie complètement différente dans le sens où elle ne profite pas de la couche 3 afin de fonctionner (technologie de couche 2.5). De plus l'objectif n'est pas exactement le même. MPLS fonctionne grâce à l'ajout d'étiquettes sur les paquets qui vont définir le chemin que va prendre celui-ci. Dans un réseau privé tel que celui d'une entreprise, cela permet de router "au plus rapide" les paquets afin d'augmenter l'efficacité du réseau. Cependant si l'entreprise utilise du cloud storage (fourni par un tiers) pour certains de ses services ou cherche à faire augmenter la taille de son réseau, MPLS va rapidement poser des problèmes de scalabilité (pas accès aux serveurs de l'entreprise tierce, temps de mise en place conséquent sur des réseaux de grande ampleur)

4 Modes IP/UDP L2TPv3 et Tunnel GRE

4.1 Analyse de l'encapsulation IP

En prenant cet exemple de communication, des échanges (à travers les protocoles serveur socat, ping, ARP) seront instaurés entre différentes machines et routeurs, conduisant aux conclusions suivantes :

On a capturé le trafic passant par r1 sur l'interface r1-eth0 lors d'un ping effectué de p1 à p3, et voici le résultat obtenu :

```

# !/bin/bash
# INFRA_RES gti (master) X netns r1
tmux split-pane -t 17 [17:1787]
user@user-vn-5:~ [r1] sudo tcqdisc -nlwrx -l ri=rthot not tq
user@user-vn-5:~ [r1] sudo tc qdisc add dev eth0 type ENODEB (Ethernet) capture size 26144 bytes
17:22:14.867855 02:bfb:de:a7:38 > 86:96:e5:85:0f:7c, ethertype IPv4 (0x0800), length 88: (tos 0x0, ttl
02, id 35561, offset 0, flags [none], proto unknown (115), length 74)
17:22:16.2535 > 17:16.2.2535 lproto ipri 115 54
0x0800: 4500 0802 8000 0000 0000 3673 9404 ac10 0f2d E..J...>s.....
0x0801: ac10 0f1d 0000 0000 e8e9 0000 0000 0000 ffff ffff
0x0802: ffff f0ee 5011 2ac0 8100 0064 0006 0001 .P....d....
0x0803: 8000 0004 0001 fa06 5011 2ac0 c088 6400 ..P...+..d...
0x0804: 0000 0000 0000 0000 c088 6400 .....d....
12:22:14.867855 02:bfb:de:a7:38 > 86:96:e5:85:0f:7c, ethertype IPv4 (0x0800), length 88: (tos 0x0, ttl
64, id 37282, offset 0, flags [none], proto unknown (115), length 74)
17:16.1.2535 > 17:16.2.2535 lproto ipri 115 54
0x0800: 4500 0804 910c 0000 4073 8bba ac10 0f1d E..J..l..>s.....
0x0801: ac10 0f2d 0000 0700 0000 0000 0000 0001 .P....d....
0x0802: ffff f0ee 5011 2ac0 8100 0064 0006 0001 ..P...+..d....
0x0803: 8000 0004 0001 fa06 5011 2ac0 c088 6400 ..P...+..d...
0x0804: 0000 0000 0000 0000 c088 6400 .....d....
12:22:14.869078 02:bfb:de:a7:38 > 86:96:e5:85:0f:7c, ethertype IPv4 (0x0800), length 144: (tos 0x0, ttl
64, id 35854, offset 0, flags [none], proto unknown (115), length 130)
17:16.1.2535 > 17:16.2.2535 lproto ipri 115 111
0x0800: 4500 0802 8000 0000 3673 9404 ac10 0f2d E..J...>s.....
0x0801: ac10 0f1d 0000 0000 e8e9 0000 0000 0000 c088 7c90 ..P....d....
0x0802: c088 f0ee 5011 2ac0 8100 0064 0000 4500 ..P...+..d....
0x0803: 8000 0004 0001 fa06 5011 2ac0 c088 6400 ..P...+..d....
0x0804: 0000 0000 0000 0000 0000 0000 0000 0000 d400 1674 ..L.....
0x0805: 0000 2640 0000 0000 0000 1001 1213 1415 ..>0.....
0x0806: 0000 1617 1819 1a0b 1c1b 1e1f 2021 2223 2425 ..!%#...
0x0807: 0000 2627 2809 2a2b 2c2d 2e2f 3031 3233 3435 ..!%#...
0x0808: 0000 0000 0000 0000 0000 0000 0000 0000 0000 67 ...
12:22:15.819903 02:bfb:de:a7:38 > 86:96:e5:85:0f:7c, ethertype IPv4 (0x0800), length 116: (tos 0x0, ttl
64, id 35854, offset 0, flags [none], proto unknown (115), length 102)
17:16.1.2535 > 17:16.2.2535 lproto ipri 115 111
0x0800: 4500 0806 8000 0000 3673 941f ac10 0f2d E..J...>s.....
0x0801: ac10 0f1d 0000 0000 e8e9 0000 0000 3333 0000 ..P....d....
0x0802: 0002 7a40 b687 b671 8100 0064 8000 0000 .z...q..d....
0x0803: 0000 0000 0000 0000 0000 0000 7880 ..z....x....
0x0804: fbf7 f871 f781 0000 0000 0000 0000 0000 ..P....d....
0x0805: 0000 0000 0002 8590 afce 0000 0000 0101 ..P....d....
0x0806: 0aa6 b667 b671 ...q....d....
12:22:17.367251 02:bfb:de:a7:38 > 86:96:e5:85:0f:7c, ethertype IPv4 (0x0800), length 112: (tos 0x0, ttl
62, id 35864, offset 0, flags [none], proto unknown (115), length 98)

```

En analysant un paquet avec Scapy, on peut remarquer que les adresses source et destination sont visibles, mais le protocole utilisé est inconnu et les données encapsulées sont illisibles (peut-être de l'ICMP ou du ARP). Toutefois, en examinant le paquet de plus près, on peut conclure qu'il s'agit en fait d'une encapsulation IP dans IP, où un paquet ICMP, ARP ont été encapsulés dans un paquet IP.

Pour confirmer l'analyse, les adresses MAC des routeurs et des machines peuvent être vérifiées pour voir si elles correspondent à l'encapsulation IP dans IP que nous avons observée. Si les adresses MAC confirment l'encapsulation, cela renforcerait notre conclusion selon laquelle le paquet est un paquet ICMP encapsulé dans un paquet IP.

4.2 Analyse de l'encapsulation UDP

Afin de créer un tunnel en mode encapsulation UDP, les commandes à utiliser sont les suivantes :

```
ip netns exec r1 ip l2tp add tunnel remote 172.16.2.253 local 172.16.1.253 encapsulation udp tunnel_id 3000 peer_tunnel_id  
↪ 4000 udp_sport 5000 udp_dport 6000

ip netns exec r2 ip l2tp add tunnel remote 172.16.1.253 local 172.16.2.253 encapsulation udp tunnel_id 4000 peer_tunnel_id  
↪ 3000 udp_sport 6000 udp_dport 5000
```

Par la suite les mêmes manipulations pour analyser le trafic intercepté seront effectués, on obtient le résultat suivant :

En analysant le trafic avec scapy, on remarque bien que contenu de notre paquet à été encapsulé dans UDP. Si nous extrayons les données contenues dans le payload UDP nous retrouvons bien les paquets ARP et ICMP que nous avons envoyé.

4.3 Mise en place du tunnel GRE

Configuration du tunnel sur les routeurs 1 et 2 :

```
ip netns exec r1 ip link add eoip1 type gretap remote 172.16.2.253 local 172.16.1.253 nopmtudisc
ip netns exec r1 ip link set dev eoip1 up
ip netns exec r1 brctl addbr tunnel
ip netns exec r1 brctl addif tunnel eoip1
ip netns exec r1 brctl addif tunnel r1-eth1
ip netns exec r1 ip l set dev tunnel up
ip netns exec r1 ip link add link tunnel name tunnel.100 type vlan id 100
ip netns exec r1 ip link set tunnel.100 up
ip netns exec r1 ip link add link tunnel name tunnel.200 type vlan id 200
ip netns exec r1 ip link set tunnel.200 up
ip netns exec r1 ip addr add 192.168.100.254/24 dev tunnel.100
ip netns exec r1 ip addr add 192.168.200.254/24 dev tunnel.200

ip netns exec r2 ip link add eoip1 type gretap remote 172.16.1.253 local 172.16.2.253 nopmtudisc
ip netns exec r2 ip link set dev eoip1 up
ip netns exec r2 brctl addbr tunnel
ip netns exec r2 brctl addif tunnel eoip1
ip netns exec r2 brctl addif tunnel r2-eth1
ip netns exec r2 ip l set dev tunnel up
ip netns exec r2 ip link add link tunnel name tunnel.100 type vlan id 100
ip netns exec r2 ip link set tunnel.100 up
ip netns exec r2 ip link add link tunnel name tunnel.200 type vlan id 200
ip netns exec r2 ip link set tunnel.200 up
ip netns exec r2 ip addr add 192.168.100.253/24 dev tunnel.100
ip netns exec r2 ip addr add 192.168.200.253/24 dev tunnel.200
```

Interfaces des routeurs 1 et 2 :

```
tmux
tmux(207x55)
projectx@Raider:~/Desktop/INFRA_BESS [r1] ip a
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_lifetimer forever
            preferred_lifetimer forever
            link-layer-brd
            valid_lft forever preferred_lft forever
2: gre@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
3: gretap@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
4: eth0@NONE: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
5: eotap@NONE: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1462 qdisc fq_codel master tunnel state UNKNOWN group default qlen 1000
    link/ether 22:09:c5:de:03:ab brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.254/24 brd ff:ff:ff:ff:ff:ff scope link
            valid_lifetimer forever
            preferred_lifetimer forever
6: tunnel: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1462 qdisc noqueue state UP group default qlen 1000
    link/ether 22:09:c5:de:03:ab brd ff:ff:ff:ff:ff:ff
        inet fe00::c5ff:fe00:1/64 scope link
            valid_lifetimer forever
            preferred_lifetimer forever
7: tunnel_1@tunnel0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1402 qdisc noqueue state UP group default qlen 1000
    link/ether 22:09:c5:de:03:ab brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.254/24 scope global tunnel100
            valid_lifetimer forever
            preferred_lifetimer forever
            link-layer-brd
            valid_lifetimer forever
            preferred_lifetimer forever
8: tunnel_200@tunnel0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1402 qdisc noqueue state UP group default qlen 1000
    link/ether 22:09:c5:de:03:ab brd ff:ff:ff:ff:ff:ff
        inet 192.168.100.254/24 scope global tunnel200
            valid_lifetimer forever
            preferred_lifetimer forever
            link-layer-brd
            valid_lifetimer forever
            preferred_lifetimer forever
9: ri-eth0@fl193: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 5a:94:5a:31:ff:ff brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 172.16.1.251/24 brd ff:ff:ff:ff:ff:ff scope global ri-eth0
            valid_lifetimer forever
            preferred_lifetimer forever
            link-layer-brd
            valid_lifetimer forever
            preferred_lifetimer forever
10: ri-eth1@fl195: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 5a:94:5a:31:ff:ff brd ff:ff:ff:ff:ff:ff link-netnsid 0
        inet 172.16.1.251/24 brd ff:ff:ff:ff:ff:ff scope global ri-eth1
            valid_lifetimer forever
            preferred_lifetimer forever
            link-layer-brd
            valid_lifetimer forever
            preferred_lifetimer forever
11: link/ether 32:08:5a:5e:c0:6a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet fe00::300d:5aff:fe5e:c0da/64 scope link
        valid_lifetimer forever
        preferred_lifetimer forever
projectx@Raider:~/Desktop/INFRA_BESS [r2]
```

ping entre les machines P1 et P3, affichage des paquets dans les routeurs 1 et 2 :

La MTU du tunnel GRE est similaire à celle du tunnel L2TPv3, soit environ 1500 octets. Il nous est cependant possible de la modifier selon nos besoins.

5 Chiffrement IPsec

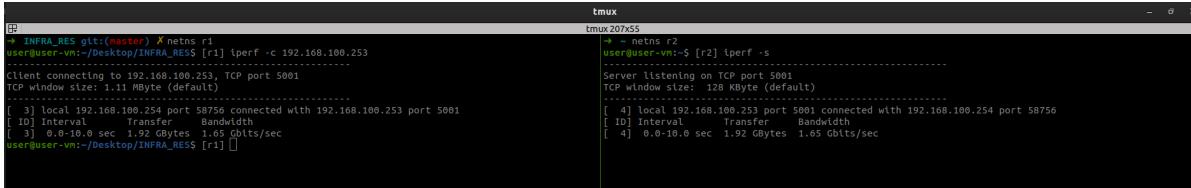
```
ip netns exec r1 ip xfrm state flush
ip netns exec r1 ip xfrm policy flush
ip netns exec r1 ip xfrm state add src 172.16.1.253 dst 172.16.2.253 proto esp spi 0x12345678 reqid 0x12345678 \
mode tunnel auth sha256 0x323730ed6f1b9ff0cb084af15b197e862b7c18424a7cdfb74cd385ae23bc4f17 enc "rfc3686(ctr(aes))"
↪ \
0x27b90b8aec1ee32a8150a664e8faac761e2d305b
ip netns exec r1 ip xfrm state add src 172.16.2.253 dst 172.16.1.253 proto esp spi 0x12345678 reqid 0x12345678 \
mode tunnel auth sha256 0x44d65c50b7581fd3c8169cf1fa0ebb24e0d55755b1dc43a98b539bb144f2067f enc "rfc3686(ctr(aes))"
↪ \
0x9df7983cb7c7eb2af01d88d36e462b5f01d10bc1
ip netns exec r1 ip xfrm policy add src 172.16.2.253 dst 172.16.1.253 dir in tmpl src 172.16.2.253 dst
↪ 172.16.1.253 proto esp reqid 0x12345678 mode tunnel
ip netns exec r1 ip xfrm policy add src 172.16.1.253 dst 172.16.2.253 dir out tmpl src 172.16.1.253 dst
↪ 172.16.2.253 proto esp reqid 0x12345678 mode tunnel

ip netns exec r2 ip xfrm state flush
ip netns exec r2 ip xfrm policy flush
ip netns exec r2 ip xfrm state add src 172.16.1.253 dst 172.16.2.253 proto esp spi 0x12345678 reqid 0x12345678 \
mode tunnel auth sha256 0x323730ed6f1b9ff0cb084af15b197e862b7c18424a7cdfb74cd385ae23bc4f17 enc "rfc3686(ctr(aes))"
↪ \
0x27b90b8aec1ee32a8150a664e8faac761e2d305b
ip netns exec r2 ip xfrm state add src 172.16.2.253 dst 172.16.1.253 proto esp spi 0x12345678 reqid 0x12345678 \
mode tunnel auth sha256 0x44d65c50b7581fd3c8169cf1fa0ebb24e0d55755b1dc43a98b539bb144f2067f enc "rfc3686(ctr(aes))"
↪ \
0x9df7983cb7c7eb2af01d88d36e462b5f01d10bc1
ip netns exec r2 ip xfrm policy add src 172.16.2.253 dst 172.16.1.253 dir out tmpl src 172.16.2.253 dst
↪ 172.16.1.253 proto esp reqid 0x12345678 mode tunnel
ip netns exec r2 ip xfrm policy add src 172.16.1.253 dst 172.16.2.253 dir in tmpl src 172.16.1.253 dst
↪ 172.16.2.253 proto esp reqid 0x12345678 mode tunnel
```

Comme nous pouvons le voir sur la capture ci-dessous, le traffic capturé par les routeurs 1 et 2 est bien chiffré :

5.1 Comparaison des performances avec Iperf

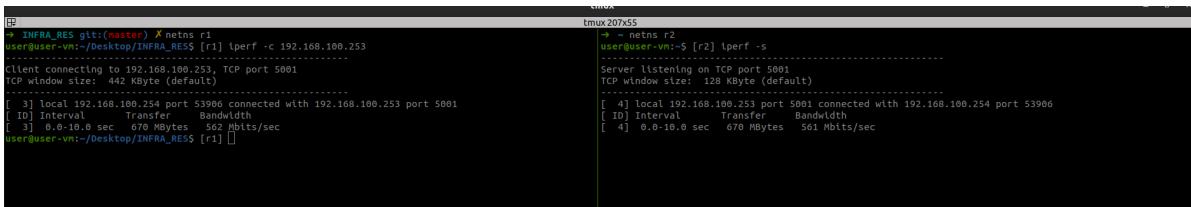
Measure du débit sans IPsec entre les routeurs 1 et 2 (500 Mb/s) :



```
INFRA_RES git:(master) ✘ netns r1
user@user-vn:~/Desktop/INFRA_RES [r1] iperf -c 192.168.100.253
Client connecting to 192.168.100.253, TCP port 5001
TCP window size: 1.11 MByte (default)
[...]
[ 3] local 192.168.100.254 port 5001 connected with 192.168.100.253 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-10.0 sec 1.92 GBytes 1.65 Gbits/sec
user@user-vn:~/Desktop/INFRA_RES [r1] 

tmux
tmux 207x55
→ - netns r2
user@user-vn:~/Desktop/INFRA_RES [r2] iperf -s
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
[...]
[ 4] local 192.168.100.253 port 5001 connected with 192.168.100.254 port 58756
[ ID] Interval Transfer Bandwidth
[ 4] 0.0-10.0 sec 1.92 GBytes 1.65 Gbits/sec
user@user-vn:~/Desktop/INFRA_RES [r2]
```

Meure du débit avec IPsec entre les routeurs 1 et 2 (1.5 Gb/s) :



```
INFRA_RES git:(master) ✘ netns r1
user@user-vn:~/Desktop/INFRA_RES [r1] iperf -c 192.168.100.253
Client connecting to 192.168.100.253, TCP port 5001
TCP window size: 442 Kbyte (default)
[...]
[ 3] local 192.168.100.254 port 53900 connected with 192.168.100.253 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-10.0 sec 678 MBbytes 562 Mbits/sec
user@user-vn:~/Desktop/INFRA_RES [r1] 

tmux
tmux 207x55
→ - netns r2
user@user-vn:~/Desktop/INFRA_RES [r2] iperf -s
Server listening on TCP port 5001
TCP window size: 128 Kbyte (default)
[...]
[ 4] local 192.168.100.253 port 5001 connected with 192.168.100.254 port 53900
[ ID] Interval Transfer Bandwidth
[ 4] 0.0-10.0 sec 678 MBbytes 561 Mbits/sec
user@user-vn:~/Desktop/INFRA_RES [r2]
```

Comme attendu le chiffrement des données nuit aux performances, toutefois nous ne nous attendions pas à un écart d'une telle taille (tout de même une division par 3 du débit).

6 Accès à internet

Pour se connecter à internet, l'interface internet située dans le root sera utilisée comme point d'accès pour les routeurs **rA** et **rB**, tout en étant leur route par défaut.

La première étape consiste à activer cette interface et lui assigner une adresse IP.

```
ip l set dev internet up
ip a add 10.87.0.3/24 dev internet
```

Configurer la table de routage de **rA** et **rB**

```
ip netns exec rA ip r add default via 10.87.0.3
ip netns exec rB ip r add default via 10.87.0.3
```

Enfin, la fonctionnalité de MASQUERADE est activée pour cacher les adresses IP privée des routeurs.

```
iptables -t nat -F
iptables -t nat -A POSTROUTING -s 10.87.0.3/24 -j MASQUERADE
sysctl net.ipv4.conf.all.forwarding=1
```

Afin de permettre l'accès à internet depuis un VLAN, il est nécessaire d'activer la fonctionnalité de MASQUERADE aussi pour les réseaux du VLAN sur le routeur **r1**. De plus, il doit être configuré comme la route par défaut pour les machines **p3** et **p4**.

```
ip netns exec rA iptables -t nat -A POSTROUTING -s 172.16.1.0/24 -j MASQUERADE
ip netns exec r1 iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j MASQUERADE
ip netns exec r1 iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j MASQUERADE

ip netns exec rB iptables -t nat -A POSTROUTING -s 172.16.2.0/24 -j MASQUERADE
ip netns exec r2 iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -j MASQUERADE
ip netns exec r2 iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j MASQUERADE
```

Nota: Il est possible de configurer r1 comme la route par défaut pour p1 et p2 également, si nécessaire. Cela dépend des besoins spécifiques de la configuration.

Ou bien spécifier dans le serveur On peut remarquer que l'accès internet fonctionne.

6.1 Redirection du trafic DHCP

Pour la configuration DHCP intelligente, les demandes seront redirigées vers **r2** en utilisant cette commande :

```
sudo dnsmasq -d -z -i tunnel.100 -F 192.168.100.1,192.168.100.10,255.255.255.0 --dhcp-option=3,192.168.100.253
```

On pourra faire pareillement avec l'interface tunnel.200, en separrant les commandes avec " ; "

```
sudo dnsmasq -d -z -i tunnel.200 -F 192.168.200.1,192.168.200.10,255.255.255.0 --dhcp-option=3,192.168.200.253
```

Obtention d'une adresse IP sur P1 :

Traceroute depuis P1, les paquets passent bien par R2, puis RA et non par R1 :

```
projectx@Raider:~/Desktop/INFRA_RES$ [p1] traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max
 1  192.168.100.253  1,237ms  0,005ms  0,002ms
 2  172.16.2.254  0,003ms  0,002ms  0,002ms
 3  10.87.0.3  0,495ms  0,003ms  0,145ms
 4  164.81.239.254  4,699ms  2,678ms  9,175ms
 5  10.0.9.113  1,769ms  1,619ms  1,832ms
 6  10.0.9.93  1,903ms  15,973ms  3,770ms
 7  10.0.8.33  4,203ms  15,400ms  1,377ms  dans le serveur
 8  193.50.172.102  1,957ms  1,569ms  1,556ms  pour que l'accès internet fonctionne
 9  193.51.189.76  3,448ms  1,866ms  1,961ms
10  193.51.180.72  13,790ms  13,485ms  13,598ms
11  193.51.180.186  15,377ms  15,509ms  13,385ms
12  193.55.204.110  13,304ms  12,951ms  12,733ms
13  193.51.180.118  18,872ms  13,423ms  13,427ms
14  72.14.218.132  13,481ms  13,889ms  13,074ms
15  74.125.244.225  15,338ms  15,358ms  15,672ms
16  216.239.42.241  15,149ms  15,653ms  15,321ms
17  8.8.8.8  19,505ms  21,058ms  13,392ms
```

7 Gestion du traffic entre les VLAN

7.1 IP tables

Effectuer les deux commandes suivantes sur les routeurs 1 et 2 :

```
iptables -t filter -A FORWARD -s 192.168.100.0/24 -d 192.168.200.0/24 -j DROP  
iptables -t filter -A FORWARD -s 192.168.200.0/24 -d 192.168.100.0/24 -j DROP
```

7.2 Policy Routing

Pour ce faire nous avons commencé par créer des tables de routages différentes pour les VLAN 100 et 200 que nous avons ajouté dans `/etc/netns/rX/iprooute2/rt_tables`.

```
100 Vlan100  
200 Vlan200
```

Ensuite il nous faut interdire le traffic d'un VLAN à l'autre sur chacun des routeurs :

```
ip rule add from 192.168.100.0/24 table 100 prio 1  
ip rule add from 192.168.200.0/24 table 200 prio 1  
ip route add prohibit 192.168.200.0/24 table vlan100  
ip route add prohibit 192.168.100.0/24 table vlan200
```

Comme nous pouvons le voir ci-dessous le ping entre Poste 3 et Poste 4 fonctionne avant l'exécution des commandes présentées, une fois celles-ci exécutées les paquets apparaissent comme "filtered" :

```
INFRA_RES:~(master) X netns r1  
user@user-vn:~/Desktop/INFRA_RES [r1] $ sudo ./proh.sh  
+ ip rule add from 192.168.100.0/24 table 100 prio 1  
+ ip rule add from 192.168.200.0/24 table 200 prio 1  
+ ip route add prohibit 192.168.200.0/24 table vlan100  
+ ip route add prohibit 192.168.100.0/24 table vlan200  
user@user-vn:~/Desktop/INFRA_RES [r1] $ ip rule  
0: from all lookup local  
1: from 192.168.100.0/24 lookup vlan100  
1: from 192.168.200.0/24 lookup vlan200  
32766: from all lookup main  
32767: from all lookup default  
user@user-vn:~/Desktop/INFRA_RES [r1] $ ip route  
0: 0.0.0.0/0 via 192.168.100.1 dev eth0  
1: 192.168.100.0/24 dev eth0  
1: 192.168.200.0/24 dev eth0  
32766: 0.0.0.0/0 via 192.168.100.1 dev eth0  
32767: 0.0.0.0/0 via 192.168.200.1 dev eth0  
user@user-vn:~/Desktop/INFRA_RES [r1] $ tmux  
tmux  
user@user-vn:~/Desktop/INFRA_RES [p4] $ sudo ip r add default via 192.168.200.254  
user@user-vn:~/Desktop/INFRA_RES [p4] $ ping 192.168.200.2  
PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.  
64 bytes from 192.168.200.2: icmp_seq=1 ttl=63 time=0.346 ms  
64 bytes from 192.168.200.2: icmp_seq=2 ttl=63 time=0.065 ms  
...  
--- 192.168.200.2 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1006ms  
rtt min/avg/max/mdev = 0.065/0.205/0.346/0.140 ms  
user@user-vn:~/Desktop/INFRA_RES [p3] $ ping 192.168.100.2  
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.  
From 192.168.100.254 icmp_seq=1 Packet filtered  
From 192.168.100.254 icmp_seq=2 Packet filtered  
From 192.168.100.254 icmp_seq=3 Packet filtered  
From 192.168.100.254 icmp_seq=4 Packet filtered  
...  
--- 192.168.100.2 ping statistics ---  
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3050ms  
user@user-vn:~/Desktop/INFRA_RES [p3]
```