

Rapport Namespace

Ricardo Dos Santos

April 8, 2025

Ce rapport est écrit sans IA générative

Contents

I	Introduction	3
II	Topologie	3
III	Préparation d'un hôte	3
IV	Policy Routing et les horreurs du double nat	6
V	Routeur virtuel bridgé	7
VI	Instancier plusieurs routeurs virtuels	8

I Introduction

Les namespaces sont un moyen de virtualiser au niveau logiciel plusieurs instances séparées de pile TCP/IP sur un PC typique comparé à des machines virtuelles.

Ce document parle des namespaces réseau Linux, et ce document parlera d'instancier, à l'aide des namespaces réseau, des routeurs virtuels.

Chaque script sera dans ce git. À noter, que le script de la partie Routeur virtuels et Préparation d'un hôte sont considéré équivalent (moins la partie bridge, il faut partir du principe que si l'on retire la partie bridge, le script sera valide pour préparer un hôte).

II Topologie

La topologie est telle que dans la figure 1:

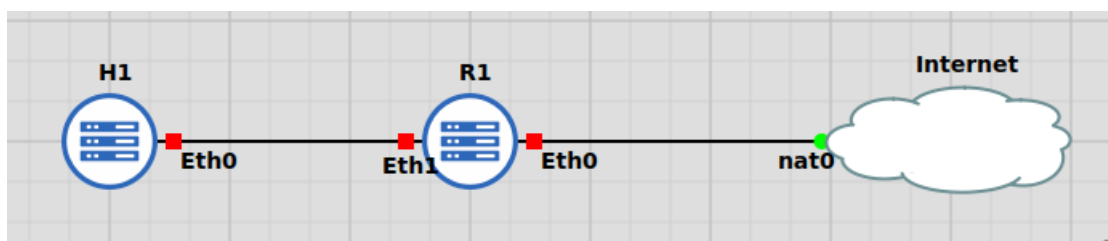


Figure 1: Topologie du labo

La topologie de la figure 1 ne contient qu'un seul hôte dans un LAN (local area network) dédié, avec un routeur (Un pc linux débian) et un accès à internet sur l'une des interfaces du routeur (ici eth0).

L'accès SSH est fait via l'interface mgmt0 de chaque hôte, cependant au delà de cet accès, aucun trafic ne sera routé via cette interface.

III Préparation d'un hôte

Tout d'abords, il faut préparer l'hôte (ici H1) a utilisé R1 sans namespace.

Pour ce faire, nous allons commencé par activé l'*IP_forward* dans R1 via la commande:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Sans cette commande, il est impossible de router des paquets à travers l'hôte, cette commande autorise le kernel a routé les paquets qu'il reçoit.

Ensuite, il faut configurer les IPs des interfaces de R1 et H1. Pour ce faire, la configuration est la suivante:

- L'interface *eth0*, celle du côté *internet provider*, sera configuré via DHCP. L'un des moyens disponible est le suivant:

```
dhclient -v eth0
```

La commande ci-dessus invoque un processus qui fait les requêtes DHCP pour l'interface passée en paramètre (DHCP offer, DHCP request, ect.).

Une route par défaut sera installé via l'interface *eth0* qui recevra l'adresse via DHCP.

- Pour ce labo, il suffira d'ajouter statiquement l'adresse IP des interfaces (*eth1* pour R1, *eth0* pour H1). L'adresse réseau du LAN dans lequel se trouve H1 est choisi arbitrairement à 10.0.0.0/24 (donc une adresse réseau privée).
- La route par défaut de H1 est l'adresse de l'interface *eth1* de R1:

```
ip route add default via 10.0.0.1
```

Cette commande ajoute de manière éphémère (la route disparaît au redémarrage) une route par défaut passant par l'adresse arbitraire de R1 qui est 10.0.0.1/24.

Finalement, nous allons ajouté une règle *nftables* qui dit que chaque paquet routé sur *eth0* dans R1 vera son adresse source modifiée à celle de *eth0*. En d'autres termes, un Network Address Translation (NAT).

Pour ce faire, voici la règle à appliquer:

```
#!/usr/sbin/nft -f

flush ruleset

table ip nat {
    chain masq {
        type nat hook postrouting priority 100;
        oifname "eth0" counter masquerade
    }
}
```

Ci-dessus, chaque ligne s'explique comme suit:

- *#!/usr/sbin/nft -f*: Ce script, quand lancer, utilise la commande *nft -f* (au même titre qu'un script bash utiliserait *#!/bin/bash*, c'est équivalent)
- *flush ruleset*: Supprimer chaque règles établie auparavant
- *table ip nat*: Ajout d'une table de la famille IP, dont le nom est "nat"
- *chain masq*: Ajout d'une chaîne nommée "masq"
- *type nat hook postrouting priority 100*;; Attaché un hook nat après le routage de chaque paquet avec une priorité de 100.

En d'autres termes, appliqué les principes du nat à chaque paquet après routage de ce dernier. La priorité permet de donné un ordonnancement si cette chaîne a plusieurs règles.

- *oifname "eth0" counter masquerade*: Compter le nombre de fois que l'interface *eth0* exécute un "masquerade", donc un remplacement de l'adresse source par celle de l'interface (donc *eth0*).

Avec tout cela, il est possible de ping internet depuis H1:

```
root@h1:~# traceroute 1.1.1.1
traceroute to 1.1.1.1 (1.1.1.1), 30 hops max, 60 byte packets
 1  10.0.0.1 (10.0.0.1)  1.093 ms  0.741 ms  0.632 ms
 2  172.21.1.1 (172.21.1.1)  1.886 ms  3.058 ms  2.984 ms
```

Figure 2: Screenshot de traceroute depuis H1

IV Policy Routing et les horreurs du double nat

Désormais, l'hôte R1 aura un namespace qui servira de routeur, nommé R2, connecté par des paires d'interface virtual ethernet (veth).

Voici la topologie au niveau de R1:

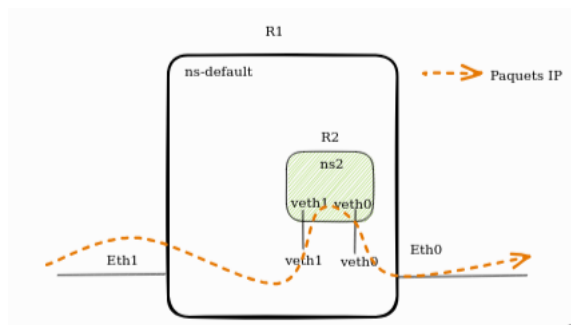


Figure 3: Screenshot de la représentation de R1 avec namespace, sourcé de l'énoncé du labo

Cette fois-ci, voici la configuration de R1:

- La route par défaut est veth1 dans R1.
- La route par défaut de R2 est veth0.
- L'adresse de *eth1* est inchangée (10.0.0.1/24).
- L'adresse de *veth1* est 192.168.1.1/24.
- L'adresse de *veth0* est 192.168.0.1/24.
- L'adresse de *eth0* est reçue via dhcp, et la route par défaut via *eth0* doit être supprimée si elle existe.
- Une table de routage customisé qui envoie tous les paquets provenant de *veth0* vers *eth0* en tant que route par défaut.

Ceci est fait grâce aux commandes suivantes:

```
echo 100 custom >> /etc/iproute2/route/tables
ip rule add iif veth0 table custom
ip route add default via 172.21.1.1 table custom
```

Ces trois lignes de codes font les actions suivantes dans l'ordre:

- Ajout de la table custom avec une priorité de 100
- Ajout d'une règle qui dit que tout paquet venant de *veth0* doit se référer à la table custom de routage
- Ajout d'une route par défaut via 172.21.1.1 (La passerelle internet) dans la table custom

Pour ce qui est de R2, voici la configuration:

- *veth0_r2* a l'adresse 192.168.0.2/24
- *veth1_r2* a l'adresse 192.168.1.2/24
- Une route par défaut allant vers *veth0* de R1
- Une route vers 10.0.0.0/24 passant par *veth1* de R1 est faite afin que les paquets puissent revenir vers H1

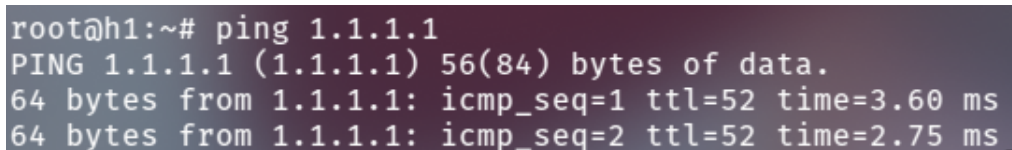
En plus de toutes ces configurations, nous devons pêché et faire non pas un NAT, mais **deux NATs**.

Pour ce faire, nous pouvons reprendre le script nftables du premier point pour R1. Pour R2, il suffit de remplacer *eth0* par le *veth* de sortie du namespace (ex: *veth0_r2*).

Ceci est dû au fait que l'action de naté ne se déroule qu'au sein de la pile dans lequel le nat doit se passer.

Ce qui signifie que la sortie du namespace et la sortie de R1 forment leur propre pile, auquel les paquets de H1 n'appartiennent pas à cette pile TCP/IP. De ce fait, il faut nater l'adresse en sortant de R2 afin que les paquets soient naté à la sortie de R1.

Le fait de router par R2 affecte le *Time to Live*(TTL), qui sera plus bas qu'un ping ne passant que par R1.



```
root@h1:~# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=52 time=3.60 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=52 time=2.75 ms
```

Figure 4: Screenshot des pings après configuration d'un double nat, après beaucoup de douleur à comprendre qu'il fallait faire un double nat

V Routeur virtuel bridgé

Afin d'éviter des règles de routages afin de renvoyer les paquets dans R2 depuis R1, nous pouvons utiliser des bridges linux.

Les bridges linux sont des switches virtuels, chaque interface passe en mode promiscuous.

Le mode promiscuous change les comportements de l'interface, qui accepte chaque trame

qui lui est envoyé, à l'inverse du comportement par défaut qui n'acceptait que les trames qui lui étaient directement dédiées.

La topologie est la suivante:

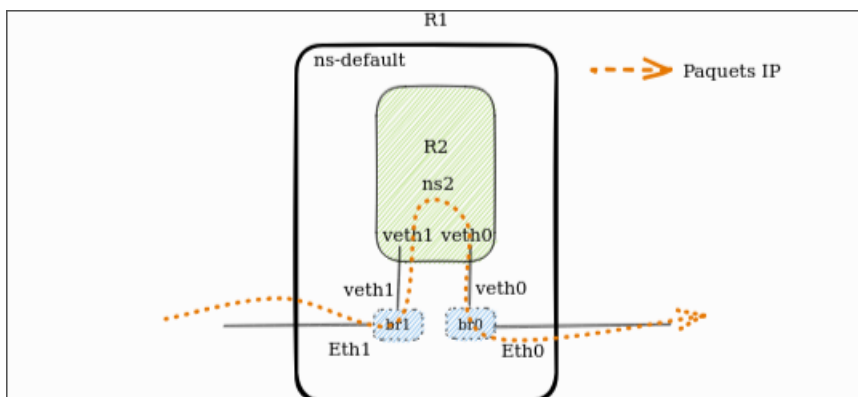


Figure 5: Screenshot de la topologie de R1 sourcé de l'énoncé

Dans cette topologie, R1 ne sert que de switch virtuel au routeur R2.

La configuration de R1 devient la suivante:

- Création des bridges et des paires veth.
- Création du namespace, et déplacement de la paire des veth dans le namespace R2.
- Changer les interfaces en promiscuous de tel manière:

```
ip link set eth1 master br1
ip link set veth1 master br1

ip link set eth0 master br0
ip link set veth0 master br0
```

Finalement, pour configurer R2, nous reprenons la configuration de la première section du document, mais à la place de R1, nous le faisons dans R2.

VI Instancier plusieurs routeurs virtuels

Pour cette partie, il faut choisir entre policy routing et bridgé pour la configuration de deux routeurs virtuels.

Le choix est vite fait: ce sera les bridges. Plus jamais de NAT.

La topologie est la suivante:

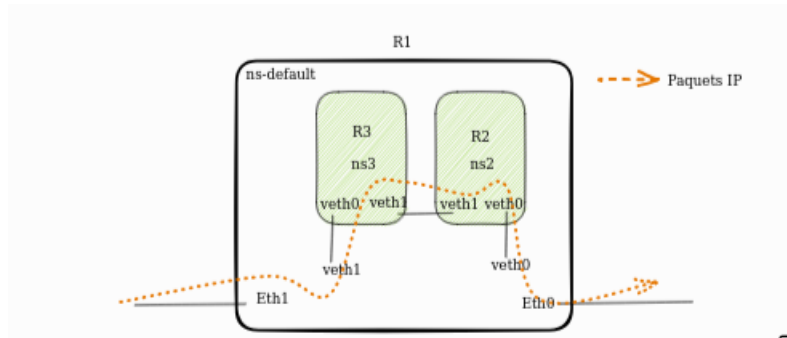


Figure 6: Screenshot de la topologie de R1 sourcé de l'énoncée

La configuration se fait en trois étapes:

- Configuration de R1: Ajout des bridges, ajout des paires veth.

Les paires appartenant à R1 sont master au bridges `br0` et `br1`. Pour finir, les interfaces sont up

- Configuration de R2: Déplacement des veths appartenant à R2, ajout des adresses en `10.0.0.0/24` côté H1 et `192.168.2.0/24` côté R3.

Une route par défaut est rajoutée vers R3 (l'interface doit être déjà configurée et up).

Up des interfaces

- Configuration de R3: Déplacement des veths, ajout d'une route vers `10.0.0.0/24` vers `192.168.2.2/24` (veth côté R2) et appel à `dhclient` pour recevoir une adresse en DHCP côté internet.

Ajout des règles de NAT à la veth de sortie de R3.

Up des interfaces.

Cette configuration fait que chaque routeur est sa propre pile TCP/IP unique et ne demande pas de faire l'affront capital du double NAT.

```

C14:46:36.470155 IP (tos 0x0, ttl 63, id 44566, offset 0, flags [DF], proto ICMP (1), length 84)
  10.0.0.3 > 1.1.1.1: ICMP echo request, id 651, seq 1, length 64
14:46:36.471931 IP (tos 0x0, ttl 54, id 32510, offset 0, flags [none], proto ICMP (1), length 84)
  1.1.1.1 > 10.0.0.3: ICMP echo reply, id 651, seq 1, length 64

2 packets captured
2 packets received by filter
0 packets dropped by kernel
[ 1467.991861] device veth2 left promiscuous mode
root@r1:~# [ 1468.730236] device veth0_r3 left promiscuous mode
[ 1469.428632] device veth2_r3 left promiscuous mode

root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#
root@r1:~#

root@h1:~# ping 1.1.1.1 -c 1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data:
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=3.02 ms

--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.018/3.018/3.018/0.000 ms
root@h1:~#

```

10