Rapport du
Projet M1 Réseaux

Quentin Fougereau
Alexandre Putzu
Joachim Neulet

M1 Informatique Luminy
Aix-Marseille Université
2019/2020

Le projet est disponible sur github à l'adresse suivante:
https://github.com/quentinfougereau/projet-reseau-M1

1.Un tunnel via le réseau IPv6

Le but de ce projet est de permettre des communications entre îlots IPv4 au sein d'un monde IPv6. Dans une première partie, on utilisera des tunnels simples au-dessus de IPv6.

2. L'interface virtuelle TUN

2.2. Configuration de l'interface

2. Il y a des modifications à apporter sur les VM1, VM1-6, VM3 et VM3-6.

VM1	VM1-6	VM3	VM3-6
Suppression des routes vers LAN2 et LAN4 par la passerelle VM2	Suppression des routes vers LAN1, LAN2 et LAN4 par la passerelle VM1	Suppression des routes vers LAN1 et LAN3 par la passerelle VM2	Suppression des routes vers LAN1, LAN2 et LAN3 par la passerelle VM3

3. Rien ne s'affiche.

4. On voit des requêtes Ping dont la source est 172.16.2.1 et la destination est 172.16.2.10.

5. Lors de l'envoi d'une requête ping sur l'adresse 172.16.2.1, on n'observe aucun trafic car le système a décidé qu'il n'avait pas besoin de rediriger le paquet puisque l'interface tun0 peut répondre à la requête. Cependant lorsque l'on ping l'adresse 172.16.2.10, on observe bien un trafic sur l'interface tun0. Ici le système voit que l'adresse appartient au réseau 172.16.2.0/28 or tous les paquets envoyés sur le réseau 172.16.2.0/28 passent par l'interface tun0. De plus les paquets sont perdus car il n'existe pas de machine disposant d'une interface avec l'adresse 172.16.2.10.

2.3. Récupération des paquets

3. En executant "ping 172.16.2.1" rien ne s'affiche, comme précedemment.
Par contre, en exécutant "ping 172.16.2.10", des données sont affichées sur la sortie standard. Ce sont des paquets IP sous forme hexadecimal.
Les données observées sur la sortie standard et sur Wireshark sont presque identiques.

4. L'option IFF_NO_PI, permet d'omettre l'information du paquet c'est à dire de se passer des 4 octets (2 octets pour les drapeaux, 2 octets pour le protocole) au début du paquet. Dans la sortie standard il n'y plus les 4 octets (00 08 en hexadécimal) au début de chaque paquet. Les données observées sont identiques à celles observées sur Wireshark.

3. Un tunnel simple pour IPv4

3.1. Redirection du trafic entrant

1.On test la bibliothèque *extremite* qui gère le trafic. On met en place les deux extrémités *ext_in* sur la machine VM1-6 et *ext_out* sur VM1. On observe le trafic suivant sur la sortie standard de la VM1 lorsque l'on effectue un ping sur l'adresse 172.16.2.10 (depuis VM1-6) comme dans la partie précédente.

00 00 08 00 45 00 00 54 fb 48 40 00 40 01 e3 34
ac 10 02 01 ac 10 02 0a 08 00 31 03 08 53 00 15
3f b4 b9 5d 00 00 00 00 03 b0 03 00 00 00 00 00
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37

On retransmet bien le trafic provenant de l'interface tun0 de VM1-6 vers VM1.

3.2. Redirection du trafic sortant

2.On devrait pouvoir observer le trafic qui passe par l'interface tun0 (172.16.2.1) de la machine se trouvant à l'entrée du tunnel (Lors des test on choisit VM1-6 en tant que "entrée" du tunnel) depuis l'interface tun0 (172.16.2.2) de la machine se trouvant en sortie du tunnel (Lors des test on choisit VM1 en tant que "sortie" du tunnel). L'interface tun0 de la machine en sortie redirigera alors le trafic vers le destinataire du paquet.

3. Ce qu'on observe sur l'interface tun0 (172.16.2.2) quand on *ping 172.16.2.10* depuis la machine VM1-6 :

45 00 00 54 0c a3 40 00 40 01 d1 da **ac 10 02 01**
ac 10 02 0a 08 00 5b 67 08 77 00 01 f3 b6 b9 5d
00 00 00 00 1e 39 0a 00 00 00 00 00 10 11 12 13
14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33
34 35 36 37

Ce paquet IP contient les mêmes adresses IP source et destination que la trame vu en 3.1 :
Source : 172.16.2.1 (ac 10 02 01)
Destination : 172.16.2.10 (ac 10 02 0a).

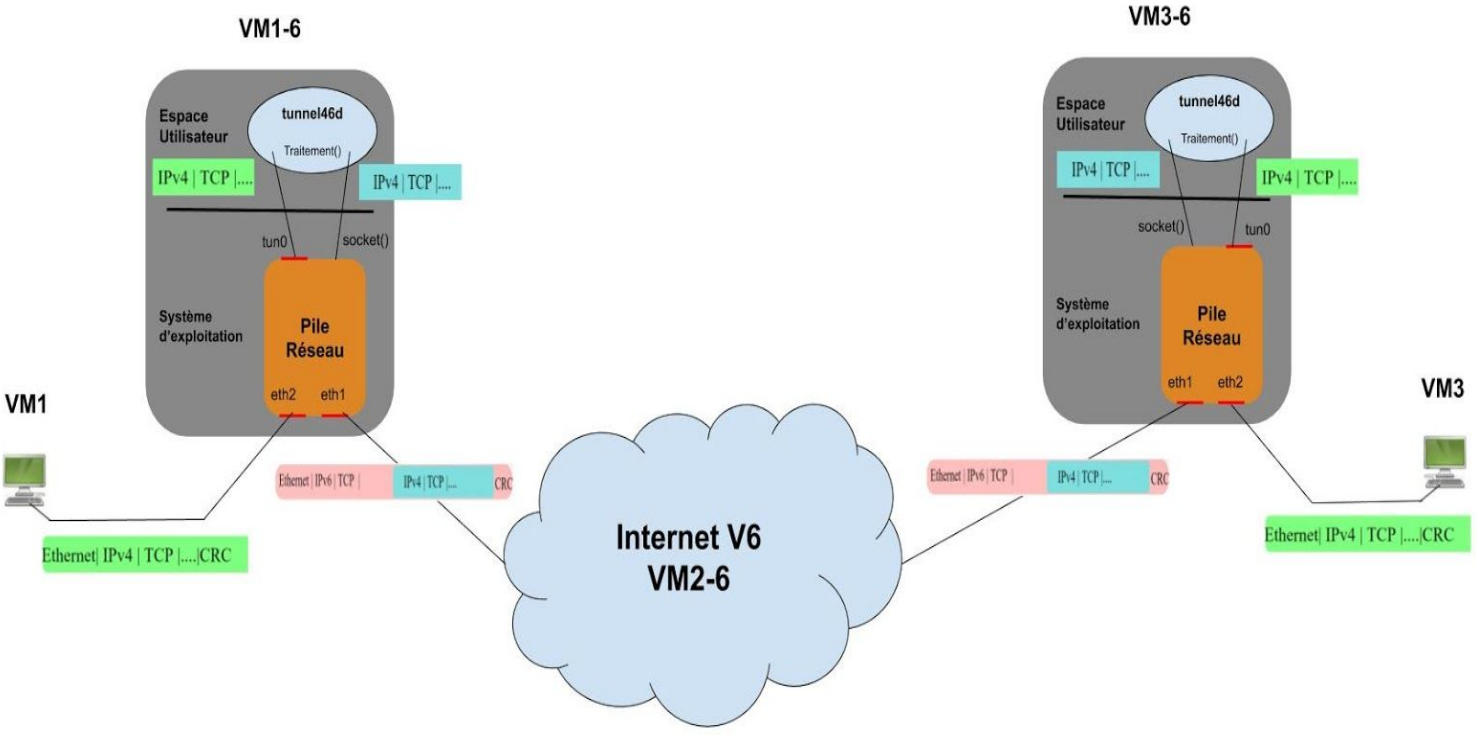
Nous observons donc que les paquets qui passent par tun0 sur VM1-6 (entrée) sont redirigés vers tun0 sur VM1 (sortie).

3.3. Intégration Finale du Tunnel

Après avoir testé le fonctionnement des redirections du trafic entrant et sortant, on modifie la bibliothèque afin d'avoir un flux bidirectionnel sur de l'IPv6.On modifie dans un premier temps les fonctions ext-in et ext-out pour que celles-ci fonctionnent sur le réseau IPv6.
Pour configurer la socket qui nous permettra de communiquer avec l'autre extrémité du tunnel, on remplace la structure **`struct sockaddr_in my_addr`**
(qui permet de communiquer sur le réseau IPv4) par **`struct sockaddr_in6 my_addr`** qui nous permet de communiquer sur IPv6.
Pour pouvoir avoir un flux bidirectionnel, on exécute ext-out et ext-in en parallèle sur chaque extrémité du tunnel.

3.4. Mise en place du tunnel entre VM1-6 et VM3-6 : Schémas

Le schéma ci-dessous représente le fonctionnement de notre tunnel:



Le tunnel étant bidirectionnel on expliquera que le trajet effectué par un paquet dans un seul sens car la transmission d'un paquet s'effectue de la même manière dans le sens inverse.

On prend ici pour exemple un paquet transmis depuis VM1 vers VM3. Le paquet IPv6 est dans un premier temps envoyé sur le réseau IPv4 depuis VM1 à VM1-6 . Celui-ci est reçu sur l'interface eth2 de VM1-6. Le paquet est ensuite redirigé par la pile réseau sur l'interface tun0. Notre programme prend ensuite le relais. Il créer une connexion tcp avec l'autre extrémité du tunnel c'est à dire VM3-6. La pile réseaux encapsule le paquet IPv4 dans une trame IPv6 et transmet le paquet à la socket qui est connectée au serveur situé sur VM3-6. Le paquet IPv6 contenant le paquet IPv4 est transmis via l'interface eth1 de VM1-6 sur le réseaux LAN1-6. Il est ensuite routé sur le réseau LAN2-6 par VM2-6. Le paquet est alors reçu par l'interface eth1 de VM3-6 et transmis à la socket du serveur puis rediriger vers tun0. La pile réseaux décapsule alors le paquet IPv4 contenu dans la trame IPv6 avant de l'envoyer sur le réseau LAN4 via l'interface eth2 de VM3-6 vers VM3.

Pour que la redirection vers les interfaces tun0 locales soit possible, on ajoute une route ayant comme destination le réseaux ipv4 distant et comme passerelle l'adresse ip de l'interface tun0 locale. L'ajout de cette route est effectué par le script qui configure l'interface tun0 et qui est exécuté par notre programme.

3.5. Mise en place du tunnel entre VM1 et VM3 : Système

On crée un exécutable tunnel46d qui va nous permettre de déployer notre tunnel sur les différentes machine.

Pour fonctionner, celui-ci utilise un fichier de configuration qui nous permettra de configurer, l'adresses de la socket locale et d'obtenir l'adresse de la socket du serveur distant ainsi que du port d'écoute du serveur locale et du serveur distant.

Configuration de VM1-6:

```
# interface tun
tun=tun0
# adresse locale
inip=fc00:1234:1::16
inport=123
options=0
# adresse distante
outip=fc00:1234:2::36
outport=123
tunconfigfile=./config-tun-VM1-6.sh
```

Configuration de VM3-6:

```
# interface tun
tun=tun0
# adresse locale
inip=fc00:1234:2::36
inport=123
options=0
# adresse distante
outip=fc00:1234:1::16
outport=123
tunconfigfile=./config-tun-VM3-6.sh
```

5. Améliorations

Lors du projet, nous avons tenté d'implémenter des améliorations pour notre tunnel.
Nous avons essayé de mettre place la gestion de tunnel via salt. La difficulté majeure était de faire en sorte que salt s'exécute correctement .Cependant lorsque salt exécute des instructions, il attend que celle-ci lui renvoie un code lui indiquant l'échec ou le succès de son exécution.Le problème étant notre programme ne s'arrête "jamais". Il fallait donc faire en sorte que le programme puisse tourner en tâche de fond et demander à salt de ne pas attendre de retour de la part de l'instruction qui exécute le tunnel. Il suffit pour cela de rajouter les lignes suivant aux fichiers config.sls des machines exécutant le tunnel.

```
tunnel:
  cmd.run:
   -  name : /mnt/partage/./tunnel46d  /mnt//partage/config-vmXX.config
   -  bg: true
```

Nous avons cependant rencontré un problème. Lors de l'exécution de notre programme par salt celui-ci n'arrivait pas à configurer l'interface tun0 . La communication entre les deux extrémités était alors impossible.

Nous avons mis en place la gestion de la taille du paquet. A l'entrée du tunnel, on récupère la taille du paquet et on créer un nouveau paquet de taille égale à la taille récupérée. Ceci permet de ne pas envoyer un tampon de taille fixée mais plutôt d'envoyer seulement le paquet avec exactement les données nécessaires.