Projet Réseaux

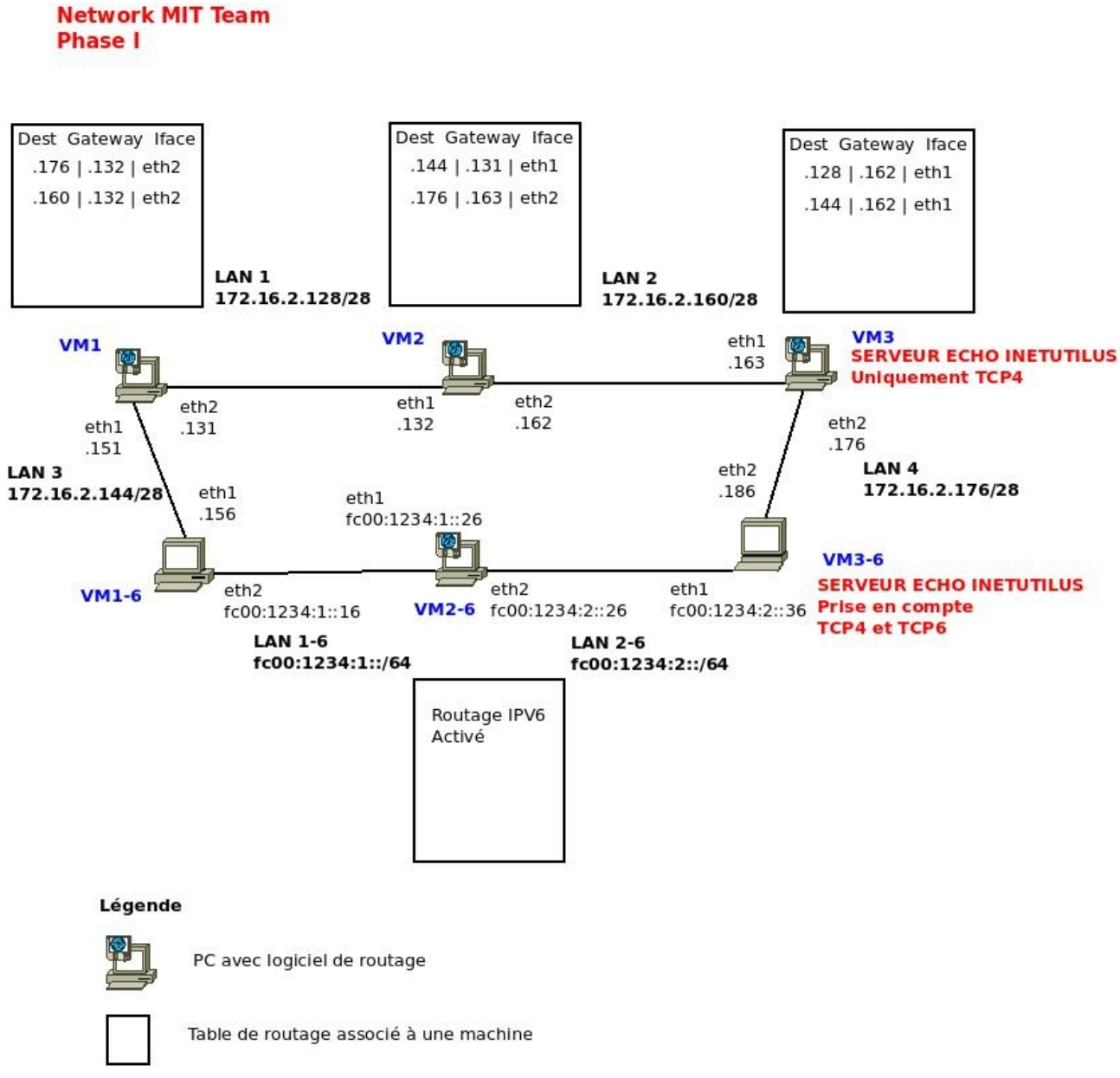
Livrable 1

Nacer Bellil
Adrar Nabil
Khemiri Abdelhak
M1 Informatique - Réseaux

Table des matières

1. Configuration Réseau

1.1 Présentation de la topologie du réseau lors de la 1ère phase



Test du réseau

IPV4 (VM1 VM2 VM3)

VM3 → VM1

root@VM3NetworkProject:/home# ping 172.16.2.131

PING 172.16.2.131 (172.16.2.131) 56(84) bytes of data.

64 bytes from 172.16.2.131: icmp_req=1 ttl=63 time=4.00 ms

VM1 → VM3

etu@VM1NetworkProject:~$ ping 172.16.2.163

PING 172.16.2.163 (172.16.2.163) 56(84) bytes of data.

64 bytes from 172.16.2.163: icmp_req=1 ttl=63 time=1.60 ms

En résumé aprés tout les tests ping effectué sur l'intégralité des deux réseau Ipv6 et Ipv4 on obtient :

Côté IPV4 :

A partir de VM1

	VM1-6	VM2	VM3	VM3-6
Réponse au Ping ?	**OUI**	**OUI**	**OUI**	**OUI**

A partir de VM2

	VM1-6	VM1	VM3	VM3-6
Réponse au Ping ?	**OUI**	**OUI**	**OUI**	**OUI**

A partir de VM3

	VM1-6	VM1	VM2	VM3-6
Réponse au Ping ?	**OUI**	**OUI**	**OUI**	**OUI**

A partir de VM1-6

	VM1	VM2	VM3	VM3-6
Réponse au Ping ?	**OUI**	**OUI**	**OUI**	**OUI**

A partir de VM3-6

	VM1	VM2	VM3	VM1-6
Réponse au Ping ?	**OUI**	**OUI**	**OUI**	**OUI**

Côté IPV6 :

A partir de VM1-6

	VM2-6	VM3-6
Réponse au Ping ?	**OUI**	**OUI**

A partir de VM2-6

	VM1-6	VM3-6
Réponse au Ping ?	**OUI**	**OUI**

A partir de VM3-6

	VM1-6	VM2-6
Réponse au Ping ?	**OUI**	**OUI**

1.1.3 depuis VM1, se connecter avec un client `echo` sur le serveur de VM3 et VM3-6

le client telnet VM1 arrive bien a se connecter sur le serveur echo situé sur VM3

tcp 0 0 172.16.2.163:echo 172.16.2.131:59810 ESTABLISHED

le client telnet VM1 arrive bien a joindre le serveur echo situé sur VM3-6

root@VM1NetworkProject:/home/etu# telnet 172.16.2.186 echo

Trying 172.16.2.186...

Connected to 172.16.2.186.

Escape character is '^]'.

tcp6 0 0 [UNKNOWN]:echo [UNKNOWN]:46518 ESTABLISHED

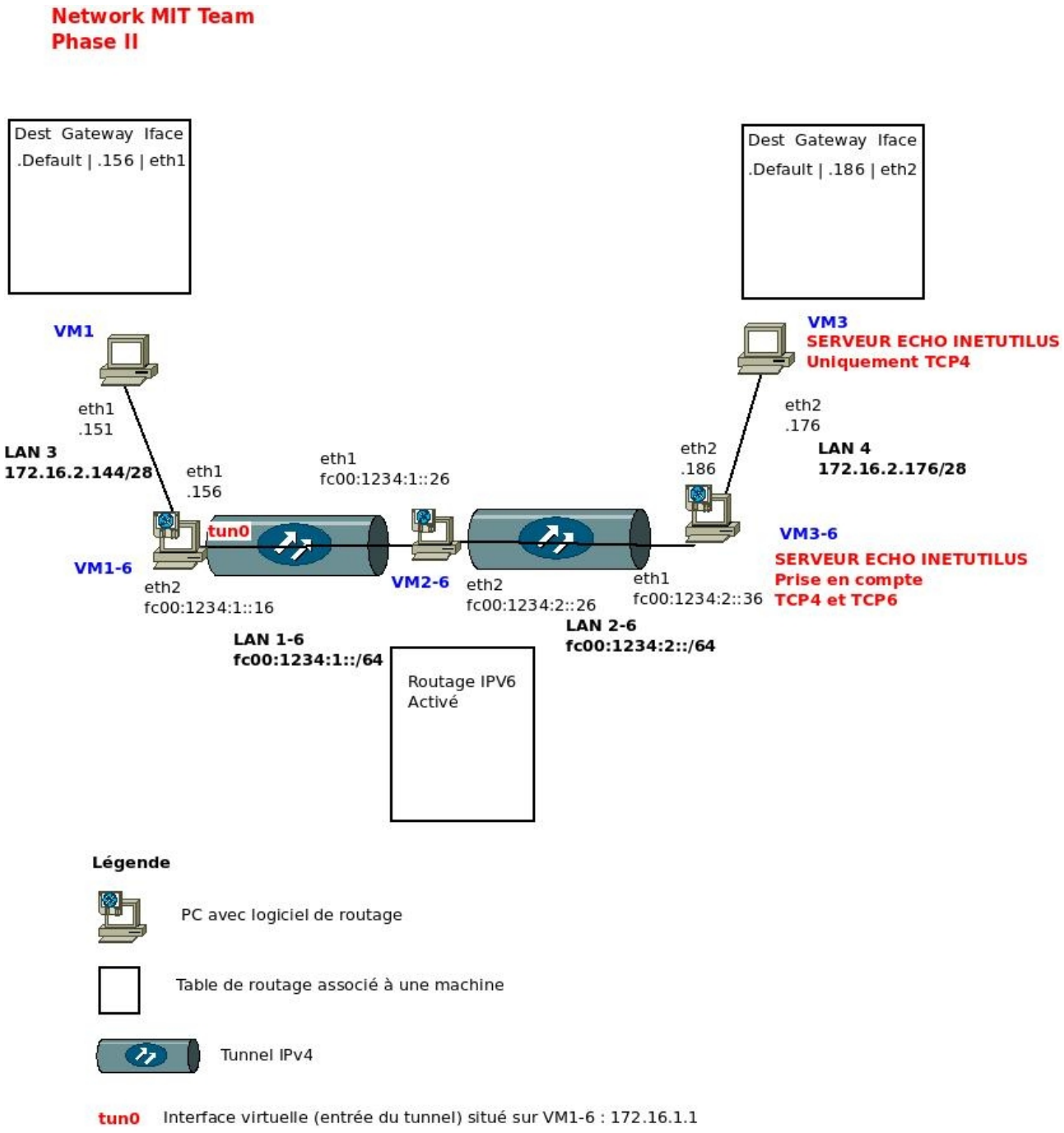
1.1.4 Peut-on faire que le serveur n'écoute qu'en IPv6 sur `VM3-6` ?

Si le support IPv6 est activé, les connexions IPv4 et IPv6 seront prises en charge si cela est pris en charge par le système d'exploitation. Si inetd ne devrait accepter des connexions IPv4 ou IPv6, ajouter «4» ou «6» pour le nom du protocole. Par exemple, «tcp4 'accepte uniquement les connexions TCP IPv4 et« tcp6' accepte uniquement les connexions IPv6 TCP.

1.2 Un grand malheur !

Nos deux îlots IPv4 sont séparé par un réseau IPv6 pour pouvoir donc permettre une connexion entre ces deux réseau IPv4 les paquets seront contrains de transiter à travers le réseau IPv6 il serait donc nécessaire de mettre en place un tunnel dans ce réseau IPv6 afin d'encapsuler les paquet IPv4 en entré et de les décapsuler en sortie afin de restituer l'intégrité des paquet Ipv4.

Présentation de la topologie lors de la situation tragique, 2ème phase !



2. L'interface virtuelle TUN

2.1. Création de l'interface

2.1.1 Configurer l'interface `tun0` avec l'adresse 172.16.1.1, mettre un masque adéquat. Ecrire un script `configure-tun.sh` reprenant vos commandes de configuration.

Cf : configure_tun.sh

2.2.2 Routage : Suite à la disparition tragique de VM2, faut-il modifier les informations de routage sur VM1 ? ou sur VM1-6 ?

Initialement VM1 communiqué avec la totalité du réseau IPv4 par routage, désormais la VM1 est contrainte de communiqué par défaut avec son voisin direct qui est VM1-6. Donc oui cette situation tragique oblige à modifier la table de routage de VM1 : Tout les paquets sortant de VM1 doivent être routés vers l'interface IPv4 eth1 de VM1-6.
Ensuite pour respecter le principe de tunnel protocol, nous devons permettre aux paquets entrants sur VM1-6 d'être rediriger sur tun0 plus précisément la station VM1-6 doit router tout les paquets IPV4 qui ne lui sont pas destinés vers le tunnel tun0.

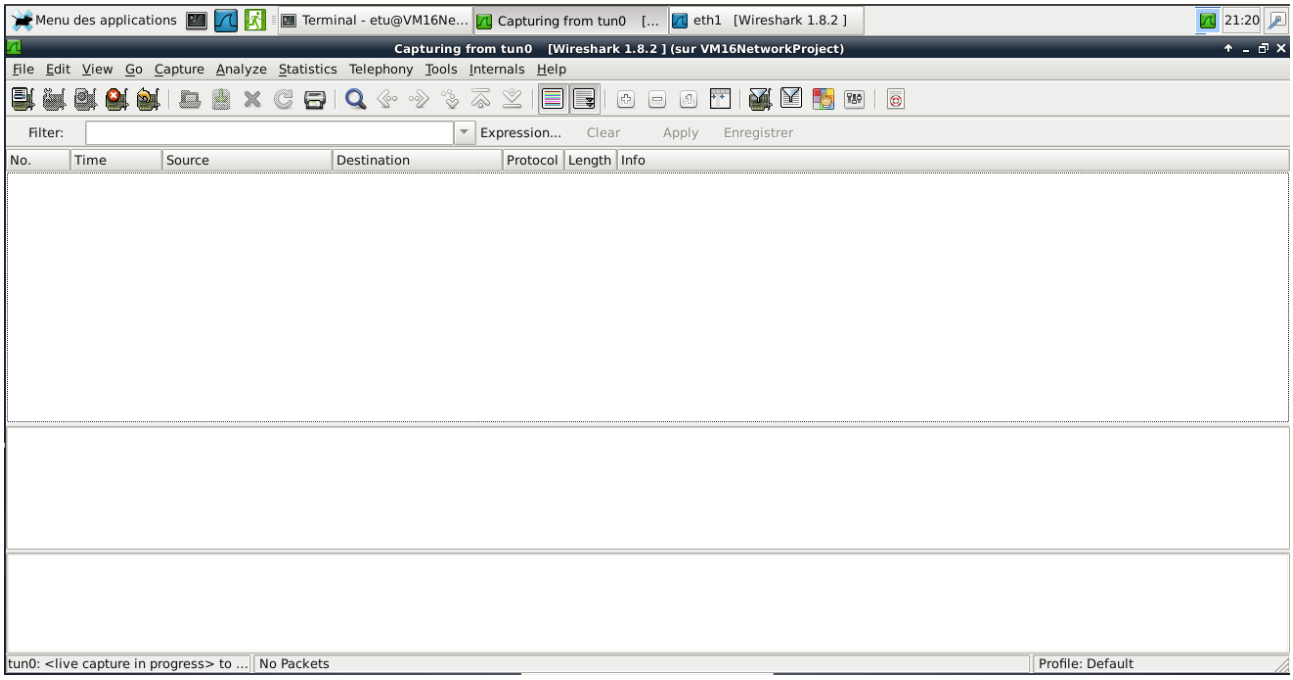
2.2.3 Faire un `ping 172.16.1.1`. Donner la capture sur `tun0` (avec `wireshark`). Que constatez-vous ?

A partir de VM1-6 on ping correctement 172.16.1.1 (Ps : tun0 est joignable en direct)

2.2. Configuration de l'interface

WIRESHARK SUR EN ÉCOUTE SUR tun0

Ping de (VM1) eth1 → tun0 (VM1-6)



Aucun trafic n'est intercepté sur l'interface tun0 **MAIS** le résultat de la commande ping dans VM1 nous indique que tun0 a répondu,

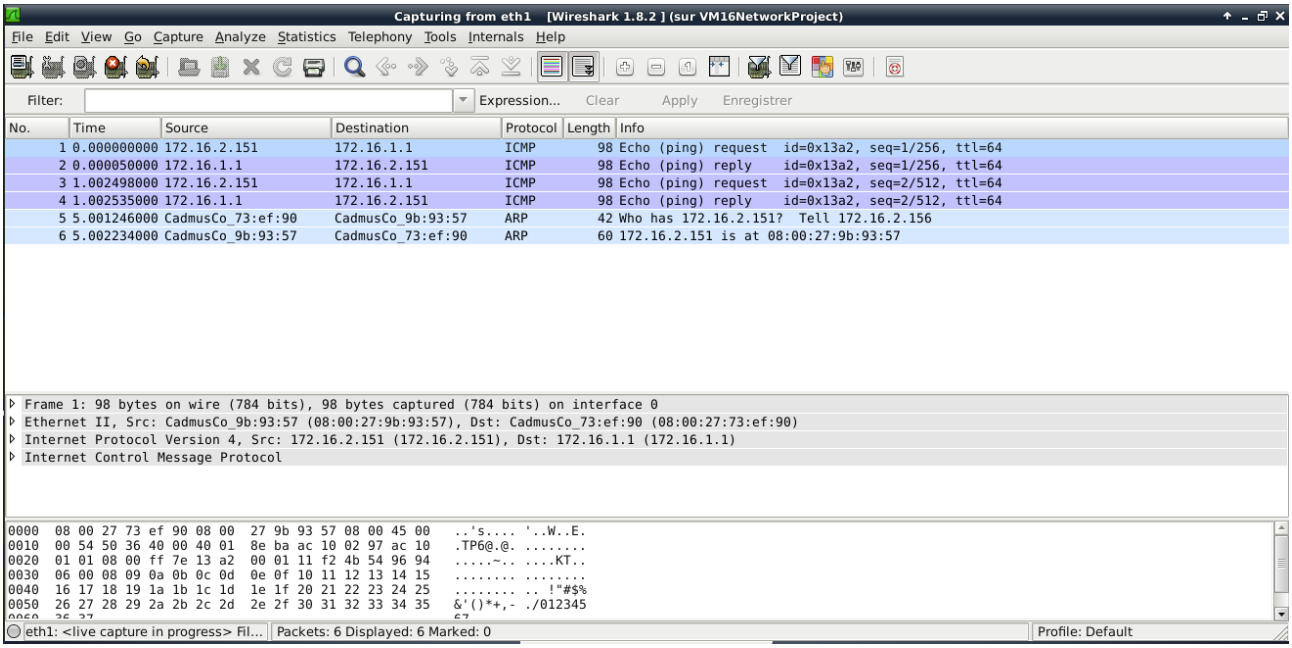
```
root@VM1NetworkProject:/home/etu# ping -c2 172.16.1.1
PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_req=1 ttl=64 time=0.810 ms
64 bytes from 172.16.1.1: icmp_req=2 ttl=64 time=0.937 ms

--- 172.16.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.810/0.873/0.937/0.070 ms
```

Voyons ce qu'a intercepté wireshark en écoute sur l'interface eth1 de VM1-6

WIRESHARK SUR EN ÉCOUTE SUR ETH1 DE VM1-6

Ping de (VM1) eth1 → tun0 (VM1-6) SUR ETH 1



Surprise !! On peut voir (sur l'interface ETH1 de VM1-6) que l'interface tun0 (ie : 172.16.1.1) a **répondu et donc reçus le ping de VM1 !**

Comment se fait-il que aucun paquet n'ai donc été intercepté sur tun0 si celui-ci a reçus et répondu a un ping ?

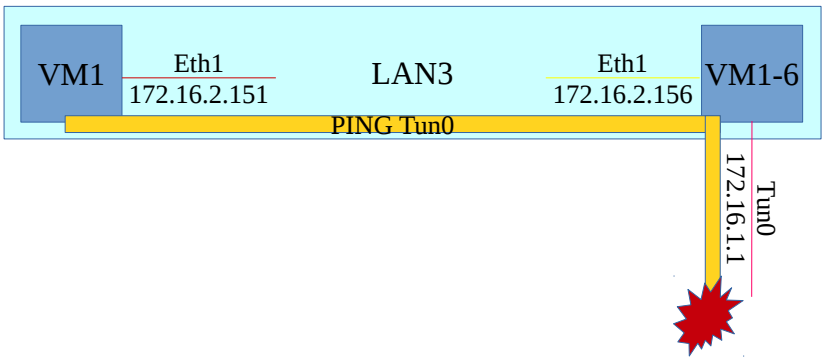
En regardant la sortie de wireshark (tun0), nous voyons ... rien. Il n'y a pas de trafic passant par l'interface. Cela est vrai: puisque nous sommes entrain de ping l'adresse IP de l'interface, le système d'exploitation décide correctement qu'aucun paquet doit être envoyé "sur le fil", et le noyau lui-même réponds à ces pings puisqu'il voit que cette interface n'est pas DOWN.

C'est exactement ce qui se passerait si on un « pingé » (a partir de VM1) l'adresse IP d'une autre interface (par exemple eth2 de VM1): aucun paquet ne serai envoyé. Cela peut paraître évident, mais pourrait être une source de confusion au début (pour nous en tout cas).

En effectuant un ping de tun0 a partir de VM1-6, on retombe sur l'exemple donné plus haut aucun paquet intercepté sur eth1 et tun0, mais on peut voir qu'ils sont envoyés et reçus sur l'interface loopback.

Faire `ping 172.16.1.10`. Que constatez-vous ?

Quand on ping 172.16.1.10 depuis VM1 ou VM1-6 on s'attend à aucune réponse. C'est en effet le cas puisque le paquet ICMP request est transmis à VM1-6 qui le route sur l'interface tun0. Or pour l'instant le processus attaché a tun0 ne fait aucun traitement sur les données reçus et ne renvoi rien.



À partir de VM1 :

	Ping tun0 (172.16.1.1)	Ping 172.16.1.10
Paquet affiché dans la console ?	**NON**	**OUI**
Réponse au Ping ?	**OUI**	**NON**

Explication :

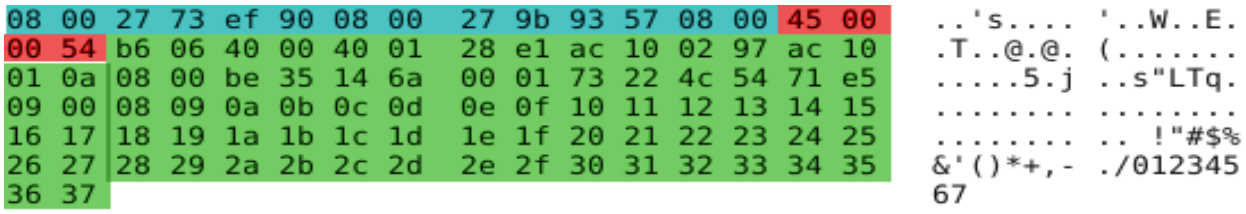
> Comme nous l'avons expliqué le paquet à destination de **tun0** ne va pas être routée vers l'interface tun0 c'est le système qui va répondre au ping de VM1, donc on a bien une réponse au ping mais rien est entrée dans le tunnel donc on obtient aucun affichage dans la console.
>
> Pour le ping de 172.16.1.10 c'est différent, le paquet par de VM1 et arrive sur l'interface eth1 de VM1-6, là le système voit qu'il faut le routé sur l'interface tun0 car ils appartiennent au même sous-réseau. Le paquet ICMP est donc envoyé sur l'interface tun0 à laquelle est attaché note processus ***test_iftun*** ce programme lit ce paquet ICMP et l'affiche à l'écran. On a donc un affichage dans test_iftun mais pas de réponse au ping. (pour avoir la réponse et l'affichage il aurai fallu que notre programme construise un paquet ICMP reply l'envoie a VM1)

2.3. Récupération des paquets

2.3.3

Comme précédemment on a une réponse au ping pour 172.16.1.1 mais pas pour 172.16.1.10 qui « passe dans le tunnel » et est donc afficher à l'écran . On voit des requêtes et réponses ARP et des requêtes ICMP

– Wireshark –



– Console –

```
00000c00  00 00 08 00 45 00 00 54  b6 06 40 00 3f 01 29 e1  |....E..T..@.?.).|
00000c10  ac 10 02 97 ac 10 01 0a  08 00 be 35 14 6a 00 01  |...........5.j..|
00000c20  73 22 4c 54 71 e5 09 00  08 09 0a 0b 0c 0d 0e 0f  |s"LTq...........|
00000c30  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f  |................|
00000c40  20 21 22 23 24 25 26 27  28 29 2a 2b 2c 2d 2e 2f  | !"#$%&'()*+,-./|
00000c50  30 31 32 33 34 35 36 37  00 00 00 00 00 00 00 00  |01234567........|
00000c60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

On peut voir que c'est les même donnée sauf que il n'y a plus de d'en-tête Ethernet car l'interface virtuel tun0 fonctionne comme matériel réseau de niveau 3.

Donc le noyau, à la réception des trames dés-encapsule les entête Ethernet jusqu'à arrivé au en-tête IP puis les route et les envoient à tun0. le noyau envoie le paquet IP brut, pas d'autres têtes sont présentes. (les 0 a la fin servent de « bourrage »)

C'est d'ailleurs ce que l'on peut remarquer en regardant la taille de la trame émise par VM1 qui était de 98 octets et la taille de la trame intercepté sur l'interface tun0 qui était de 84 octets. La taille d'une entête Ethernet étant de 14 octets on obtient bien :

98 – 14 = 84 octets.

La trame est donc la suivante :

Flags [2 octets]
Proto [2 octets]
Raw protocol(IP, IPv6, etc) frame.

2.3.4

IFF_NO_PI est une option qui indique au noyau de ne pas fournir des informations sur les paquets. Le but de IFF_NO_PI est de dire au noyau que les paquets seront paquets IP «pure», sans octets ajoutés. Cela peut poser problème dans le cas de paquets fragmentés car le recepteur ne pourra plus les ré-assemblés

Après avoir effectuée des tests cette option ne modifie rien. En tout cas pour les ICMP Request / Reply.