

Projet Réseaux

1. Configuration Réseau

1.1. Topologie et Adressage

Mettre en place les 6 Vms

1. Installer un serveur echo sur VM3 et sur VM3-6.

2. Depuis VM1, se connecter avec un client echo sur le serveur de VM3 et VM3-6

The image shows two terminal windows side-by-side. The left window is titled 'Terminal - etu@VM1-4: /mnt/partage (sur VM1-4)' and shows the user running a script to connect to a server at 172.16.2.186 on port 1235. The script reports 'Network is unreachable' for the first two attempts and then successfully connects on the third attempt to 172.16.2.163 on port 1234, sending 'hello' and receiving 'Bonjour 172.16.2.131! (vous utilisez le port 44136)' and 'hello' in response. The right window is titled 'Terminal - etu@VM3-4: /mnt/partage (sur VM3-4)' and shows the user running an echo server on port 1234. It successfully binds and listens, then accepts connections from 172.16.2.131 on port 44135, receiving 'hey' and 'Terminé.' and then 'hello' and 'hello' in response.

```
etu@VM1-4:~$ cd /mnt/partage/
etu@VM1-4:/mnt/partage$ ./echoclient 172.16.2.186 1235
le nÂ° de la socket est : 3
Essai de connexion Ã 172.16.2.186 (172.16.2.186) sur le p
connection: Network is unreachable
etu@VM1-4:/mnt/partage$ ./echoclient 172.16.2.186 1235
le nÂ° de la socket est : 3
Essai de connexion Ã 172.16.2.186 (172.16.2.186) sur le p
connection: Network is unreachable
etu@VM1-4:/mnt/partage$ ./echoclient 172.16.2.163 1234
le nÂ° de la socket est : 3
Essai de connexion Ã 172.16.2.163 (172.16.2.163) sur le p
reÃ$u: Bonjour 172.16.2.131! (vous utilisez le port 44136)
hello
reÃ$u: > hello

etu@VM3-4:~$ cd /mnt/partage/
etu@VM3-4:/mnt/partage$ ./echoserveur 1234
Ecoute sur le port 1234
le nÂ° de la socket est : 3
Option(s) OK!
bind!
listen!
accept! (4) ip=172.16.2.131 port=44135
[172.16.2.131:44135](4243): 1 :hey
[172.16.2.131:44135](4243): TerminÃ©.
^C
etu@VM3-4:/mnt/partage$ ./echoserveur 1234
Ecoute sur le port 1234
le nÂ° de la socket est : 3
Option(s) OK!
bind!
listen!
accept! (4) ip=172.16.2.131 port=44136
[172.16.2.131:44136](4474): 1 :hello
```

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

Notre client en VM1 ne pouvait pas se connecter au serveur en VM3-6.

2. L'interface virtuelle TUN

2.2. Configuration de l'interface

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.

2. **Routing** : Suite à la **disparition tragique de VM2**, faut-il modifier les informations de routage sur VM1 ? ou sur VM1-6 ?

Il n'y a pas de changements à faire si on supprime directement le dossier de la VM, mais on peut toujours enlever les passerelles et LAN qui vont vers VM2

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

The screenshot shows a terminal window on a VM named 'etu@VM1-6'. The user runs a ping command to 172.16.1.10, which fails with 100% packet loss. Then, the user runs a ping command to 172.16.1.1, which succeeds. Simultaneously, a Wireshark window is open, showing the 'Capture Interfaces' dialog. The 'tun0' interface is selected for capture, with IP 172.16.1.1 and 0 packets/s.

On constate que des paquets sont reçus mais Wireshark lui ne voit rien.

4. Faire un ping 172.16.1.10. Que constatez-vous ?

The screenshot shows a terminal window on a VM named 'etu@VM1-6'. The user runs a ping command to 172.16.1.10, which succeeds. Simultaneously, a Wireshark window is open, showing a list of captured packets. The first packet is an ICMP Echo (ping) request from 172.16.1.1 to 172.16.1.10. The packet details show the Internet Protocol Version 4 and Internet Control Message Protocol.

Dans ce cas, la console n'affiche rien mais Wireshark voit des paquets émis.

Expliquez.

Dans le premier cas, l'émetteur est aussi le destinataire, donc rien ne traverse puisque les paquets arrivent avant. Dans le second cas, on ping une IP non affectée, les paquets sortent dans sa direction et Wireshark le voit mais ils ne reviennent jamais car rien n'est là pour répondre.

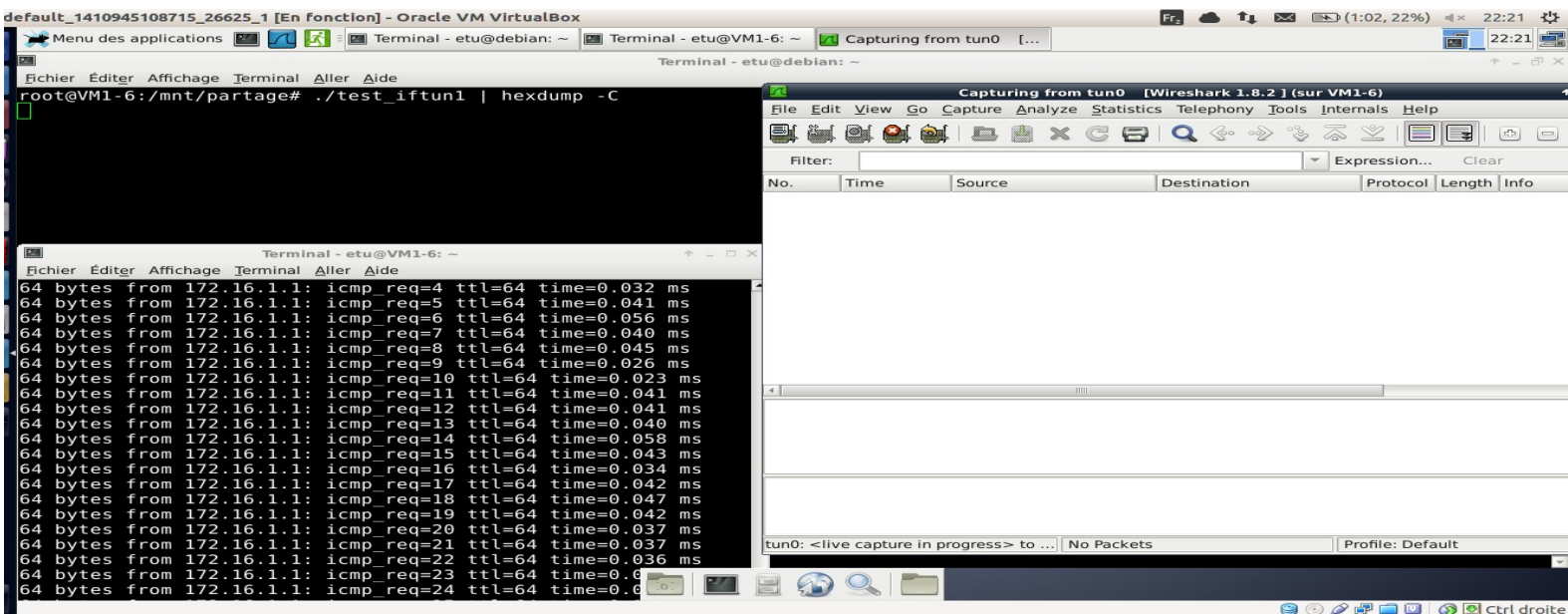
2.3. Récupération des paquets

1. Compléter la bibliothèque iftun avec une fonction avec deux descripteurs de fichiers en paramètres src et dst, qui, étant donné un descripteur srccorrespondant à une interface TUN, recopie perpétuellement toutes les données lisibles sur src dans le fichier décrit par dst.

2. Tester avec dst=1 (sortie standard). Comme ce qui est recopié est du binaire, on filtrera la sortie du programme de test avec hexdump.

```
00000000 01 00 08 00 45 00 00 54 78 7a 40 00 40 01 68 03 |...E..Txz@.@.h.|
00000010 01 00 08 00 45 00 00 54 78 7b 40 00 40 01 68 02 |...E..Tx{@.@.h.|
00000020 01 00 08 00 45 00 00 54 78 7c 40 00 40 01 68 01 |...E..Tx|@.@.h.|
00000030 01 00 08 00 45 00 00 54 78 7d 40 00 40 01 68 00 |...E..Tx}@.@.h.|
00000040 01 00 08 00 45 00 00 54 78 7e 40 00 40 01 67 ff |...E..Tx~@.@.g.|
```

3. Refaire ping 172.16.1.1 puis ping 172.16.1.10. Comparer et expliquer. Quel type de trafic voyez-vous? Refaire une capture avec wireshark dans le second cas et comparer avec ce qui est obtenu par votre programme test_iftun.



Un ping 172.16.1.1 ne change rien aux résultats précédents.

The screenshot shows a VirtualBox window titled 'default_1410945108715_26625_1 [En fonction] - Oracle VM VirtualBox'. Inside, there are two windows: a terminal and Wireshark.

Terminal - etu@VM1-6: ~

```

root@VM1-6:/mnt/partage# ./test iftun1 | hexdump -C
00000000 45 00 00 54 78 7a 40 00 40 01 68 03 ac 10 01
00000010 45 00 00 54 78 7b 40 00 40 01 68 02 ac 10 01
00000020 45 00 00 54 78 7c 40 00 40 01 68 01 ac 10 01
00000030 45 00 00 54 78 7d 40 00 40 01 68 00 ac 10 01
00000040 45 00 00 54 78 7e 40 00 40 01 67 ff ac 10 01
00000050 45 00 00 54 78 7f 40 00 40 01 67 fe ac 10 01
00000060 45 00 00 54 78 80 40 00 40 01 67 fd ac 10 01
00000070 45 00 00 54 78 81 40 00 40 01 67 fc ac 10 01
00000080 45 00 00 54 78 82 40 00 40 01 67 fb ac 10 01
00000090 45 00 00 54 78 83 40 00 40 01 67 fa ac 10 01
000000a0 45 00 00 54 78 84 40 00 40 01 67 f9 ac 10 01
000000b0 45 00 00 54 78 85 40 00 40 01 67 f8 ac 10 01
000000c0 45 00 00 54 78 86 40 00 40 01 67 f7 ac 10 01
000000d0 45 00 00 54 78 87 40 00 40 01 67 f6 ac 10 01
000000e0 45 00 00 54 78 88 40 00 40 01 67 f5 ac 10 01
000000f0 45 00 00 54 78 89 40 00 40 01 67 f4 ac 10 01
00000100 45 00 00 54 78 8a 40 00 40 01 67 f3 ac 10 01
00000110 45 00 00 54 78 8b 40 00 40 01 67 f2 ac 10 01
00000120 45 00 00 54 78 8c 40 00 40 01 67 f1 ac 10 01

root@VM1-6:/home/etu# clear

root@VM1-6:/home/etu# ping 172.16.1.10
PING 172.16.1.10 (172.16.1.10) 56(84) bytes of data.

```

Wireshark 1.8.2 [sur VM1-6]

Filter:
 Expression... Clear

No.	Time	Source	Destination	Protocol	Length	Info
16	15.12008400	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
17	16.12804400	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
18	17.13682500	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
19	18.14394200	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
20	19.15203500	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
21	20.16004700	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
22	21.16797500	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
23	22.17594000	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
24	23.18393700	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
25	24.19211200	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
26	25.19994200	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
27	26.20801600	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request
28	27.21603900	172.16.1.1	172.16.1.10	ICMP	84	Echo (ping) request

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
 Raw packet data
 Internet Protocol Version 4, Src: 172.16.1.1 (172.16.1.1), Dst: 172.16.1.10 (172.16.1.10)
 Internet Control Message Protocol

0000 45 00 00 54 78 7a 40 00 40 01 68 03 ac 10 01 01 E..Txz@.@.h....
 0010 ac 10 01 0a 08 00 f1 71 17 e8 00 01 6c 21 65 54q...lleT
 0020 26 2c 0c 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 &.....
 0030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23!.."
 0040 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 \$%&'()*+,-./0123
 0050 34 35 36 37 4567

File: "/tmp/wireshark_tun0_201411... Packets: 28 Displayed: 28 ... Profile: Default

Le principe est le même que lors des tests précédents pour un ping 172.16.1.10 (on ping vers l'inconnu), mais on remarque que les 5 premiers paquets envoyés et affichés dans la console sont identiques à ceux que Wireshark a pu lire de 172.16.1.1 vers 172.16.1.10.

4. A quoi sert l'option IFF_NO_PI ? Que ce passe-t-il si vous ajoutez cette option lors de la création de l'interface ?

Elle permet de ne pas fournir les informations sur le paquets, elle enlève les 4 premiers octets.

Sans IFF_NO_PI :

```

00000000 01 00 08 00 45 00 00 54 78 7a 40 00 40 01 68 03 |...E..Txz@.@.h|
00000010 01 00 08 00 45 00 00 54 78 7b 40 00 40 01 68 02 |...E..Tx{@.@.h|
00000020 01 00 08 00 45 00 00 54 78 7c 40 00 40 01 68 01 |...E..Tx|@.@.h|
00000030 01 00 08 00 45 00 00 54 78 7d 40 00 40 01 68 00 |...E..Tx}@.@.h|

```

Avec IFF_NO_PI :

```
00000000 45 00 00 54 78 7a 40 00 40 01 68 03 ac 10 01 01 |E..Txz@.@.h....|
00000010 45 00 00 54 78 7b 40 00 40 01 68 02 ac 10 01 01 |E..Tx{@.@.h....|
00000020 45 00 00 54 78 7c 40 00 40 01 68 01 ac 10 01 01 |E..Tx|@.@.h....|
00000030 45 00 00 54 78 7d 40 00 40 01 68 00 ac 10 01 01 |E..Tx}@.@.h....|
```

3. Un tunnel simple pour IPv4

Le but de cette partie est de créer un tunnel simple encapsulant un trafic IPv4 dans des paquets TCP/IPv6.

3.1. Redirection du trafic entrant

On utilisera par défaut le port 123.

Dans cette partie, on créera une bibliothèque `extremite` qui gèrera le trafic entre extrémités du tunnel.

1. Ecrire une fonction `ext-out` qui crée un serveur écoutant sur le port 123, et redirige les données reçues sur la sortie standard.

2. Ecrire une fonction `ext-in` qui ouvre une connexion TCP avec l'autre extrémité du tunnel, puis lit le trafic provenant de `tun0` et le retransmet dans la socket.

3. Quelle est l'adresse IPv6, noté `IPOUT`, de VM3-6 ? Comment pourrait-on l'obtenir simplement ? Automatiser ?

Nous avons configuré `IPOUT` manuellement, `fc00:1234:2::36`. On peut lui en donner une automatique avec `radvd` comme on l'a vu en tp.

4. La commande `nc` (`netcat`) permet de transférer des données sur un port réseau. Pour le support IPv6 sur vos VMs, il faudra installer éventuellement `netcat6`. L'option `-u` permet d'envoyer en UDP (puisque votre tunnel est unidirectionnel pour l'instant).

1. Mettre en place une extrémité `ext-in` et une extrémité `ext-out`.

2. Tester avec un pair `ping 172.16.1.10` pour injecter du trafic comme dans la partie précédente.

Ecoute sur le port 123

le n° de la socket est : 3

Option(s) OK!

bind!

listen!

accept! (4) ip=fc00:1234:1::16 port=34032

```
00000000 45 00 00 54 c4 e3 40 00 3f 01 19 57 ac 10 02 97 |E..T..@.?.W...|
00000010 ac 10 02 b7 08 00 af 84 11 d3 00 02 43 6c 6a 54 |.....CljT|
00000020 9a e2 03 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 |.....|
00000030 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 |.....!"#|
00000040 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 |$%&'()*+,-./0123|
00000050 34 35 36 37 45 00 00 54 c4 e5 40 00 3f 01 19 55 |4567E..T..@.?.U|
00000060 ac 10 02 97 ac 10 02 b7 08 00 be 83 11 d3 00 04 |.....|
```

3. Tester avec un pair nc6 pour injecter du trafic application.

Avec le mot « salut » depuis VM1-6

```
00000260 45 00 00 22 74 cb 40 00 40 11 6b d4 ac 10 01 01 |E.."t.@.@.k....|
00000270 ac 10 01 0a 93 22 00 7b 00 0e be 27 73 61 6c 75 |.....".{...'salu|
```

4. Tester avec un pair nc6 sur des VMs différentes.

Avec le mot « hey » depuis VM1-4

```
00000240 45 00 00 20 c4 a6 40 00 3f 11 19 b8 ac 10 02 97 |E.. ..@.?......|
00000250 ac 10 02 b7 d8 c5 00 7b 00 0c e7 b6 68 65 79 0a |.....{....hey.|
```

3.2. Redirection du trafic sortant

1. Compléter la fonction ext-out de la bibliothèque extrémité pour créer une extrémité qui lit le trafic provenant de la socket TCP et le retransmet dans le tun0 local.

2. Que se passe-t-il lorsque vous écrivez dans le descripteur correspondant à tun0 ?

Le paquet passe ensuite dans VM3-4 si c'est l'IP qu'on a spécifiée.

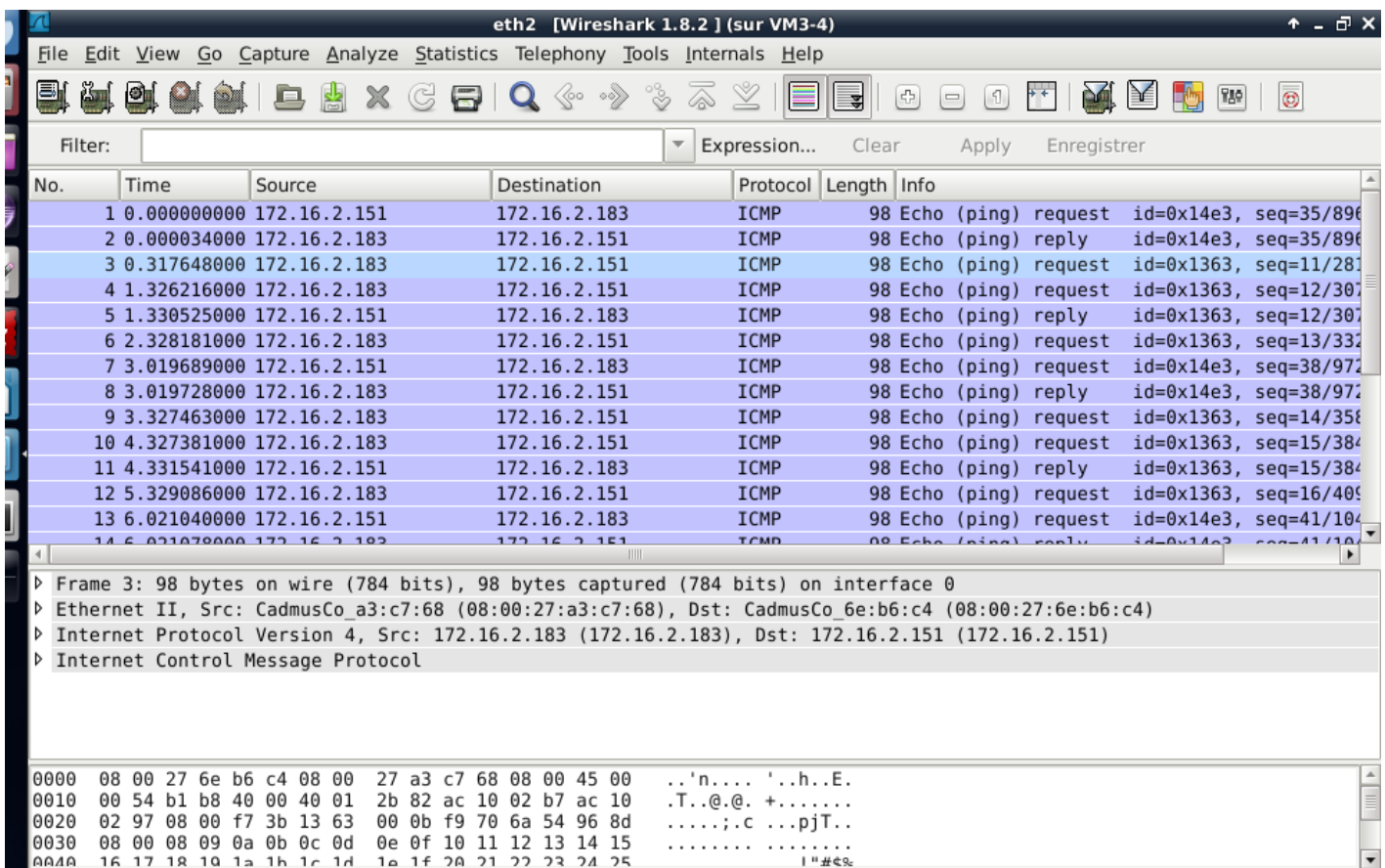
3. Vérifier avec l'un de vos paquets capturés dans le cas précédent. Pensez à le modifier éventuellement (avec un éditeur hexadécimal) pour que cela corresponde à tous vos paramètres réseau.

4. Proposer des tests de connectivité. Tester et vérifier !

3.3. Test du Tunnel

1. Compléter la bibliothèque pour pouvoir avoir un flux bidirectionnel

2. Assurez-vous que les communications sont bien asynchrones.

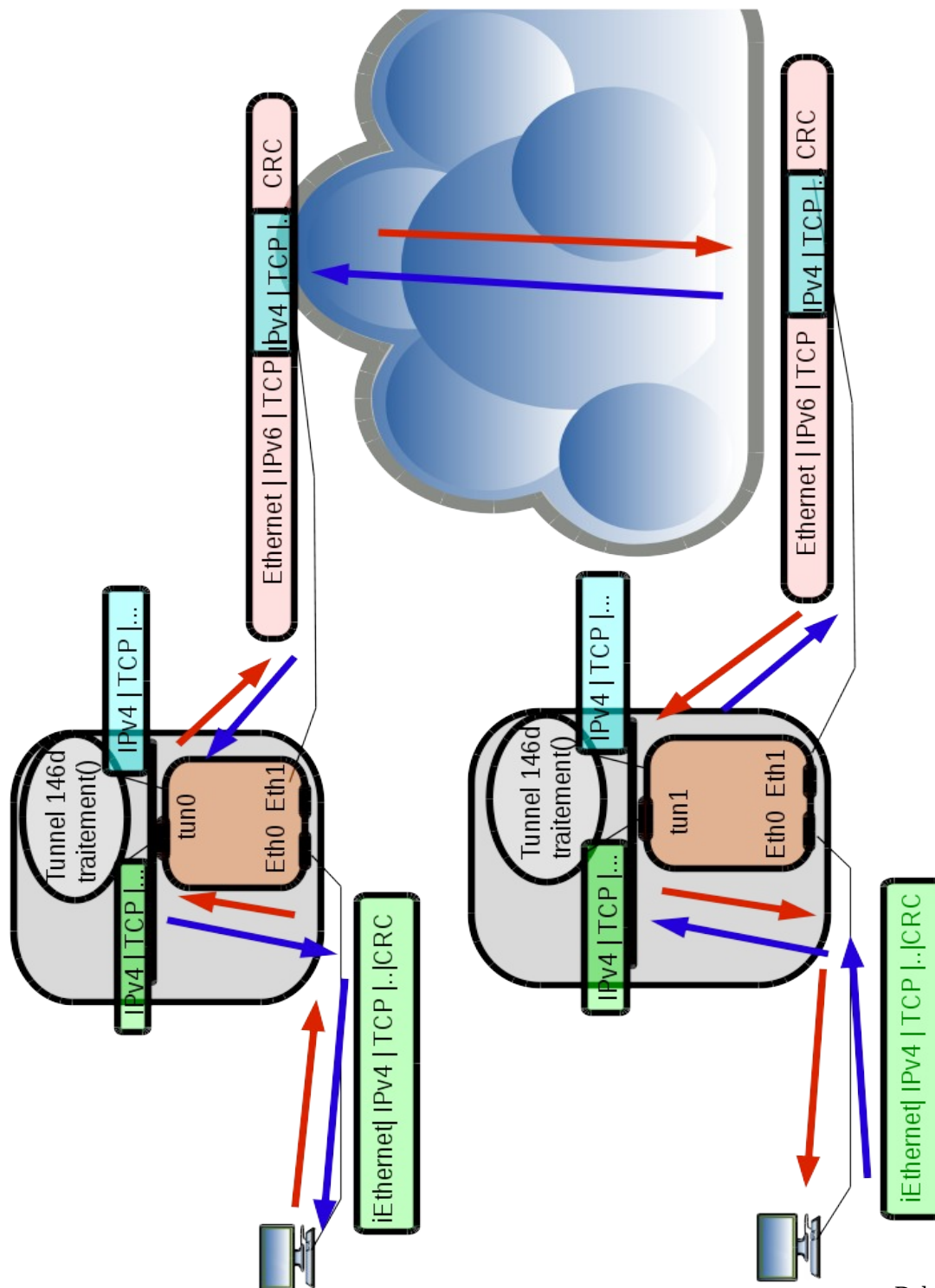


On remarque qu'il n'y a pas vraiment d'ordre, c'est donc bien asynchrone

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

On veut utiliser VM1-6 et VM3-6 pour pouvoir faire un tunnel entre LAN3 et LAN4.

- Compléter le **schéma simplifié** en expliquant **en détail** le parcours complet d'un paquet.

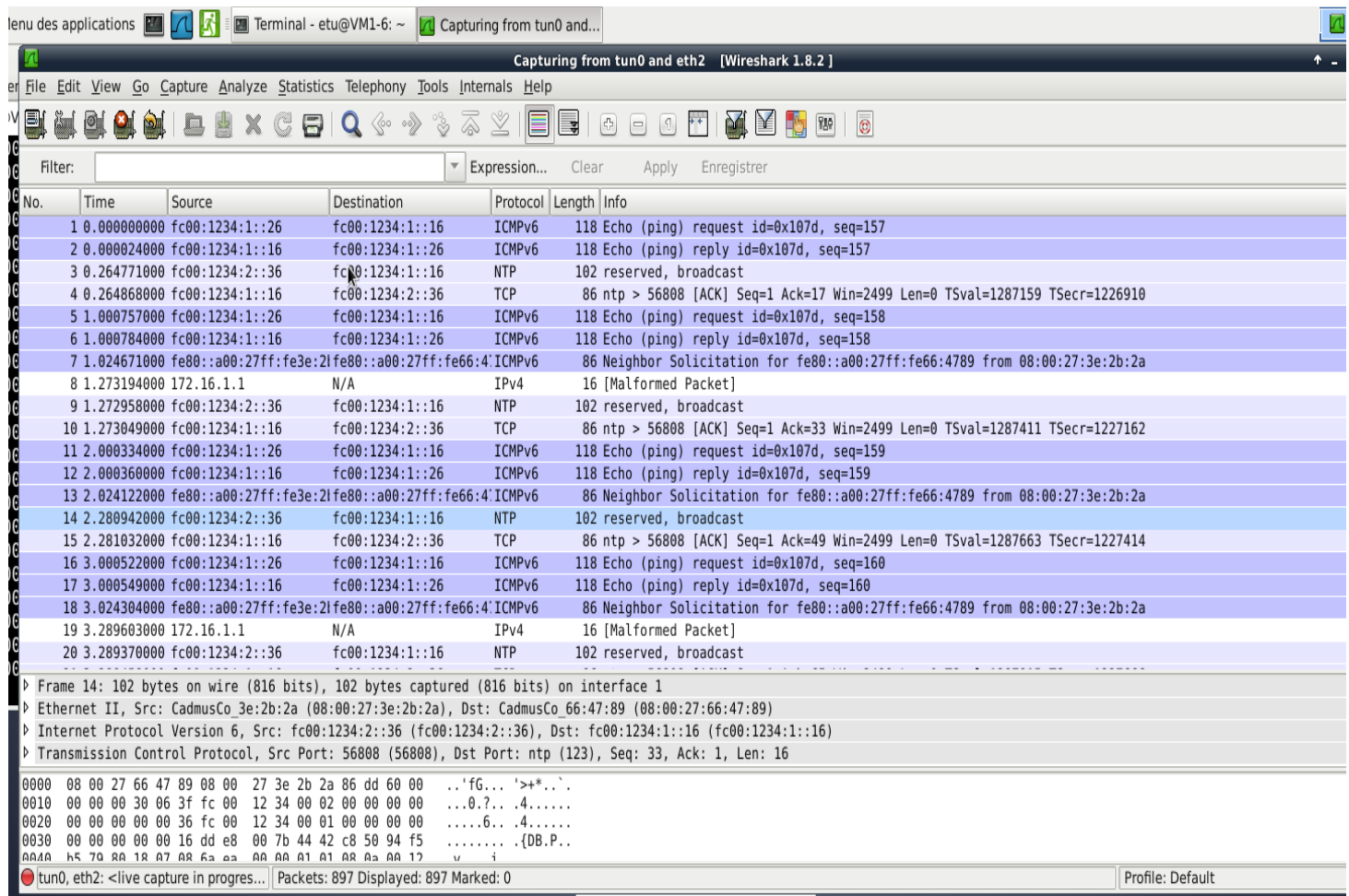


3.5. Mise en place du tunnel entre VM1-6 et VM3-6 : Système

1. Créer un exécutable tunnel46d qui, en s'appuyant sur la bibliothèque `extremite` crée un service offrant un tunnel TCP bidirectionnel.

Il pourra appeler le script de configuration réseau avec ces paramètres une fois que l'interface est créée. Il est également possible de faire des appels systèmes directs avec `ioctl`.

1. Tester en configurant et lançant `tunnel46d` sur deux VMs différentes.
2. Comparer le trafic direct et celui passant par le tunnel.



Les protocoles NTP et TCP sont les ping direct et l'IPv4 est issu du tunnel.

4. Améliorations

Nous n'avons pas eu le temps, malheureusement !