Projet réseau

DHONT Florent – COEFFIC Guénolé – THOMAS Nicolas

Livrable intermédiaire

1. Configuration réseau

Pour cette première itération nous avons eu beaucoup de mal à faire communiquer les 6 VMs. Nous y avons donc beaucoup passé de temps.

Chose faite nous sommes passés à l'installation des serveurs echo sur VM3 et VM3-6.
Pour lancer correctement echo il faut faire :

```
$ dhclient eth0
$ apt install inetutils-inetd
$ update-inetd --add "echo stream tcp6 nowait nobody internal"
$ service inetutils-inetd start
```

Ne pas oublier de faire
```
$ service inetutils-inetd status
```

Et si on a active(excited) :
```
$ service inetutils-inetd restart
```

Avant d'appliquer la configuration avec
```
$ salt-call state.apply
```

On peut ensuite tester avec
```
root@vm1:/home/m1reseaux# telnet fc00:1234:2::36 echo
Trying fc00:1234:2::36...
Connected to fc00:1234:2::36.
Escape character is '^]'.
bonjour
bonjour
salut
salut
```

Et
```
m1reseaux@vm1:~$ telnet fc00:1234:4::3 echo
Trying fc00:1234:4::3...
Connected to fc00:1234:4::3.
Escape character is '^]'.
```

```
bonjour
bonjour
```

N'ayant plus de temps pour la suite nous la ferons pour le prochain rendu.

Livrable finale

Il est possible de faire que le serveur sur VM3 n'écoute que en IPv6 en modifiant le protocol dans les fichiers de configuration il faut ajouter 6 au nom du protocol tel que 'tcp6'

1.2 Un grand malheur !

Tout ceci est bien embêtant.

2. L'interface TUN

```
#!/bin/bash
ip -6 link set tun0 up
ip -6 addr add fc00:1234:ffff::1/64 dev tun0
```

Il est present de le fichier files des VMs (/vagrant/files)

On modifie alors tunalloc.c qui devient

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/ioctl.h>

#include <unistd.h>
#include <fcntl.h>
#include <linux/if.h>
#include <linux/if_tun.h>


int tun_alloc(char *dev)
{
struct ifreq ifr;
```

```
int fd, err;

if( (fd = open("/dev/net/tun", O_RDWR)) < 0 ){
perror("alloc tun");
exit(-1);
}

memset(&ifr, 0, sizeof(ifr));

/* Flags: IFF_TUN - TUN device (no Ethernet headers)
* IFF_TAP - TAP device
*
* IFF_NO_PI - Do not provide packet information
*/
ifr.ifr_flags = IFF_TUN;
if( *dev )
strncpy(ifr.ifr_name, dev, IFNAMSIZ);

if( (err = ioctl(fd, TUNSETIFF, (void *) &ifr)) < 0 ){
close(fd);
return err;
}
strcpy(dev, ifr.ifr_name);
return fd;
}

int main (int argc, char** argv){

int tunfd;
printf("Création de %s\n",argv[1]);
tunfd = tun_alloc(argv[1]);
printf("Faire la configuration de %s...\n",argv[1]);

system("bash /vagrant/configure-tun.sh");
system("ip route add fc00:1234:4::/64 via fc00:1234:ffff::10 dev tun0");

printf("Interface %s Configurée:\n",argv[1]);
system("ip addr");
```

```
printf("Appuyez sur une touche pour terminer\n");
getchar();

return 0;
}
```

Le script shell met donc bien le tunnel avec fc00:1234:ffff::1 et un masque de 64.
Il ajoute également la route

Il ne faut pas modifier les informations sur VM1 mais on modifie la route vers VM1-6 afin de rajouter :

On ajoute à VM1-6

```
ip route add fc00:1234:ffff::/64 via fc00:1234:3::1 dev eth2:
cmd:
- run
```

Pour l'instant il faudra rajouter manuellement

```
$ gcc -o ../../mnt/partage/tunalloc ../../mnt/partage/tunalloc.c
$ ../../mnt/partage/tunalloc tun0
```

Ce qui configurera bien les réseaux via les tunnels

Il faut faire de même pour VM3 et VM3-6.

Avec wireshark (sur VM1) lors du ping6 fc00:1234:ffff::1 on a depuis VM1-6 et VM1:

```
No. Time Source Destination Protocol Length Info
      1 0.000000000 fe80::a00:27ff:fe5e:f04e fe80::a00:27ff:fe0f:7e4e ICMPv6 86 Neighbor Solicitation
for fe80::a00:27ff:fe0f:7e4e from 08:00:27:5e:f0:4e

Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe5e:f04e (fe80::a00:27ff:fe5e:f04e), Dst:
fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
      2 0.000033000 fe80::a00:27ff:fe0f:7e4e fe80::a00:27ff:fe5e:f04e ICMPv6 78 Neighbor Advertisement
fe80::a00:27ff:fe0f:7e4e (sol)

Frame 2: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e), Dst:
fe80::a00:27ff:fe5e:f04e (fe80::a00:27ff:fe5e:f04e)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
      3 286.007555000 fc00:1234:3::16 fc00:1234:ffff::1 ICMPv6 118 Echo (ping) request id=0x08e0,
seq=1, hop limit=64 (reply in 4)
```

Frame 3: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fc00:1234:3::16 (fc00:1234:3::16), Dst: fc00:1234:ffff::1 (fc00:1234:ffff::1)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
4 286.007621000 fc00:1234:ffff::1 fc00:1234:3::16 ICMPv6 118 Echo (ping) reply id=0x08e0, seq=1, hop limit=64 (request in 3)

Frame 4: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fc00:1234:ffff::1 (fc00:1234:ffff::1), Dst: fc00:1234:3::16 (fc00:1234:3::16)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
5 287.009725000 fc00:1234:3::16 fc00:1234:ffff::1 ICMPv6 118 Echo (ping) request id=0x08e0, seq=2, hop limit=64 (reply in 6)

Frame 5: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fc00:1234:3::16 (fc00:1234:3::16), Dst: fc00:1234:ffff::1 (fc00:1234:ffff::1)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
6 287.009756000 fc00:1234:ffff::1 fc00:1234:3::16 ICMPv6 118 Echo (ping) reply id=0x08e0, seq=2, hop limit=64 (request in 5)

Frame 6: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fc00:1234:ffff::1 (fc00:1234:ffff::1), Dst: fc00:1234:3::16 (fc00:1234:3::16)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
7 288.008827000 fc00:1234:3::16 fc00:1234:ffff::1 ICMPv6 118 Echo (ping) request id=0x08e0, seq=3, hop limit=64 (reply in 8)

Frame 7: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fc00:1234:3::16 (fc00:1234:3::16), Dst: fc00:1234:ffff::1 (fc00:1234:ffff::1)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
8 288.008863000 fc00:1234:ffff::1 fc00:1234:3::16 ICMPv6 118 Echo (ping) reply id=0x08e0, seq=3, hop limit=64 (request in 7)

Frame 8: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fc00:1234:ffff::1 (fc00:1234:ffff::1), Dst: fc00:1234:3::16 (fc00:1234:3::16)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
9 289.008222000 fc00:1234:3::16 fc00:1234:ffff::1 ICMPv6 118 Echo (ping) request id=0x08e0, seq=4, hop limit=64 (reply in 10)

Frame 9: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fc00:1234:3::16 (fc00:1234:3::16), Dst: fc00:1234:ffff::1 (fc00:1234:ffff::1)
Internet Control Message Protocol v6

No. Time Source Destination Protocol Length Info
10 289.008262000 fc00:1234:ffff::1 fc00:1234:3::16 ICMPv6 118 Echo (ping) reply id=0x08e0, seq=4, hop limit=64 (request in 9)

```
Frame 10: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fc00:1234:ffff::1 (fc00:1234:ffff::1), Dst: fc00:1234:3::16
(fc00:1234:3::16)
Internet Control Message Protocol v6


No. Time Source Destination Protocol Length Info
     11 291.010200000 fe80::a00:27ff:fe0f:7e4e fc00:1234:3::16 ICMPv6 86 Neighbor Solicitation for
fc00:1234:3::16 from 08:00:27:0f:7e:4e


Frame 11: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e), Dst:
fc00:1234:3::16 (fc00:1234:3::16)
Internet Control Message Protocol v6


No. Time Source Destination Protocol Length Info
     12 291.010723000 fc00:1234:3::16 fe80::a00:27ff:fe0f:7e4e ICMPv6 78 Neighbor Advertisement
fc00:1234:3::16 (rtr, sol)


Frame 12: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fc00:1234:3::16 (fc00:1234:3::16), Dst: fe80::a00:27ff:fe0f:7e4e
(fe80::a00:27ff:fe0f:7e4e)
Internet Control Message Protocol v6


No. Time Source Destination Protocol Length Info
     13 296.017935000 fe80::a00:27ff:fe5e:f04e fe80::a00:27ff:fe0f:7e4e ICMPv6 86 Neighbor Solicita-
tion for fe80::a00:27ff:fe0f:7e4e from 08:00:27:5e:f0:4e


Frame 13: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe5e:f04e (fe80::a00:27ff:fe5e:f04e), Dst:
fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e)
Internet Control Message Protocol v6


No. Time Source Destination Protocol Length Info
     14 296.018099000 fe80::a00:27ff:fe0f:7e4e fe80::a00:27ff:fe5e:f04e ICMPv6 78 Neighbor Advertise-
ment fe80::a00:27ff:fe0f:7e4e (sol)


Frame 14: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e), Dst:
fe80::a00:27ff:fe5e:f04e (fe80::a00:27ff:fe5e:f04e)
Internet Control Message Protocol v6


No. Time Source Destination Protocol Length Info
     15 301.025981000 fe80::a00:27ff:fe0f:7e4e fe80::a00:27ff:fe5e:f04e ICMPv6 86 Neighbor Solicita-
tion for fe80::a00:27ff:fe5e:f04e from 08:00:27:0f:7e:4e


Frame 15: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 3
Ethernet II, Src: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e), Dst: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e), Dst:
fe80::a00:27ff:fe5e:f04e (fe80::a00:27ff:fe5e:f04e)
Internet Control Message Protocol v6


No. Time Source Destination Protocol Length Info
     16 301.026471000 fe80::a00:27ff:fe5e:f04e fe80::a00:27ff:fe0f:7e4e ICMPv6 78 Neighbor Advertise-
ment fe80::a00:27ff:fe5e:f04e (rtr, sol)


Frame 16: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 3
Ethernet II, Src: CadmusCo_5e:f0:4e (08:00:27:5e:f0:4e), Dst: CadmusCo_0f:7e:4e (08:00:27:0f:7e:4e)
Internet Protocol Version 6, Src: fe80::a00:27ff:fe5e:f04e (fe80::a00:27ff:fe5e:f04e), Dst:
fe80::a00:27ff:fe0f:7e4e (fe80::a00:27ff:fe0f:7e4e)
Internet Control Message Protocol v6
```

Depuis VM1 et VM1-6 on a donc bien les packets qui sont transmis et les réponses. Ce n'est pas le cas depuis VM3 et VM3-6 ou nous ne recevons pas les packets.

Si on ping6 fc00:1234:ffff::10 depuis VM1 ou VM1-6 on a les packets qui sont bien transmis à fc00:1234:ffff::1 mais on a pas de réponse.

En modifiant le programme et en éxécutant le programme suivant :
$ ../../mnt/partage/tunalloc tun0 | hexdump -C

Nous obtenons :

```
00000000 60 00 00 00 00 40 3a 40 fc 00 12 34 ff ff 00 00 |`....@:@...4....|
00000010 00 00 00 00 00 00 00 10 fc 00 12 34 ff ff 00 00 |...........4....|
00000020 00 00 00 00 00 00 00 01 80 00 2a 90 27 d5 00 01 |..........*.'...|
00000030 98 dd c3 5f 00 00 00 00 f1 93 04 00 00 00 00 00 |..._............|
00000040 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f |................|
00000050 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f | !"#$%&'()*+,-./|
00000060 30 31 32 33 34 35 36 37 00 00 00 00 00 00 00 00 |01234567........|
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
000004e0 00 00 00 00 00 00 00 00 4c fe ca f3 48 7f 00 00 |........L...H...|
000004f0 00 00 00 00 00 00 00 00 ca 96 93 1c 00 00 00 00 |................|
00000500 01 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 |................|
00000510 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 00 |................|
00000520 e0 10 93 f3 48 7f 00 00 00 00 00 00 00 00 00 00 |....H...........|
00000530 78 cd 90 f3 48 7f 00 00 18 03 40 00 00 00 00 00 |x...H.....@.....|
00000540 00 00 00 00 01 00 00 00 39 05 00 00 01 00 00 00 |........9.......|
00000550 01 00 00 00 00 00 00 00 10 6a ec f3 48 7f 00 00 |.........j..H...|
00000560 c0 95 6a df fc 7f 00 00 08 95 ec f3 48 7f 00 00 |..j.........H...|
00000570 e8 95 6a df fc 7f 00 00 b0 91 ec f3 48 7f 00 00 |..j.........H...|
00000580 01 00 00 00 00 00 00 00 6d 09 cb f3 48 7f 00 00 |........m...H...|
00000590 00 00 00 00 00 00 00 00 10 6a ec f3 48 7f 00 00 |.........j..H...|
000005a0 01 00 00 00 fc 7f 00 00 10 6a ec f3 48 7f 00 00 |.........j..H...|
000005b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
000005d0 78 cd 90 f3 48 7f 00 00 18 03 40 00 00 00 00 00 |x...H.....@.....|
000005e0 00 00 00 00 01 00 00 00 39 05 00 00 01 00 00 00 |........9.......|
000005f0 01 00 00 00 00 00 00 00 10 6a ec f3 48 7f 00 00 |.........j..H...|
00000600 c0 95 6a df fc 7f 00 00 08 95 ec f3 48 7f 00 00 |..j.........H...|
00000610 e8 95 6a df fc 7f 00 00 b0 91 ec f3 48 7f 00 00 |..j.........H...|
00000620 01 00 00 00 00 00 00 00 6d 09 cb f3 48 7f 00 00 |........m...H...|
00000630 00 00 00 00 00 00 00 00 10 6a ec f3 48 7f 00 00 |.........j..H...|
00000640 01 00 00 00 fc 7f 00 00 00 00 00 00 00 00 00 00 |................|
00000650 01 00 00 00 00 00 00 00 ff d1 93 f3 48 7f 00 00 |............H...|
00000660 ff ff ff ff 00 00 00 00 2e 4e 3d f6 ca 27 00 00 |.........N=..'..|
00000670 d0 d3 93 f3 48 7f 00 00 dc 94 6a df fc 7f 00 00 |....H.....j.....|
00000680 00 00 00 00 00 00 00 00 08 95 ec f3 48 7f 00 00 |............H...|
00000690 00 00 00 00 fc 7f 00 00 20 95 6a df fc 7f 00 00 |........ .j.....|
000006a0 8a e4 ee 1c 00 00 00 00 af 05 40 00 00 00 00 00 |..........@.....|
000006b0 00 00 00 00 00 00 00 00 38 f5 d8 03 00 00 00 00 |........8.......|
000006c0 80 7a 90 f3 48 7f 00 00 c0 64 ec f3 48 7f 00 00 |.z..H....d..H...|
000006d0 00 00 00 00 00 00 00 00 90 16 60 00 00 00 00 00 |..........`.....|
000006e0 00 00 00 00 00 00 00 00 90 97 6a df fc 7f 00 00 |..........j.....|
000006f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000700 50 96 6a df fc 7f 00 00 fc 50 cb f3 48 7f 00 00 |P.j......P..H...|
00000710 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000720 e0 96 6a df fc 7f 00 00 b0 16 60 00 00 00 00 00 |..j.......`.....|
00000730 01 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 |................|
00000740 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
000007b0 00 00 00 00 00 00 00 00 00 00 00 00 fc 7f 00 00 |................|
000007c0 a0 11 40 00 00 00 00 00 0b 00 00 00 00 00 00 00 |..@.............|
```

Ainsi maintenant si on fait

```
Ping6 fc00:1234:ffff::1
```

Cela n’affiche rien. Si on fait

```
Ping6 fc00:1234:ffff::10
```

On a donc l’affichages des paquets envoyés dans tun0.

IFF_NO_PI correspond à “ Do not provide packet information “ qui permet de ne pas récupérer les informations du paquet.

Attention pour que les paquets soient bien transmis à tun0 depuis les machines -6 il faut rajouter à la configuration salt :

```
net.ipv6.conf.all.forwarding:
  sysctl:
    - present
    - value: 1
```

3. Un tunnel simple pour IPv6

3.1. Redirection du trafic entrant

Nous créons donc la bibliothèque extremite.c suivante :

```
#include "extremite.h"

/*
    Crée un serveur écoutant sur le port 123
    redirige les données recues sur la sortie tun
 */
void extout(int tun, char * port, char * options) {
    int s, n, len, on = 1;

    struct sockaddr_in client;
    struct addrinfo * resol;
    struct addrinfo indic = {
        AI_PASSIVE,
        PF_INET,
        SOCK_STREAM,
        0,
        0,
        NULL,
        NULL,
        NULL
    };

    int err = getaddrinfo(NULL,port,&indic,&resol);
    if (err<0){
        fprintf(stderr,"Résolution: %s\n",gai_strerror(err));
        exit(2);
    }

    /* Création de la socket, de type TCP / IP */
    if ((s=socket(resol->ai_family,resol->ai_socktype,resol->ai_protocol))<0) {
        perror("allocation de socket");
        exit(3);
    }
    fprintf(stderr,"le n° de la socket est : %i\n",s);

    /* On rend le port réutilisable rapidement /!\ */
```

```
    if (setsockopt(s,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on))<0) {
        perror("option socket");
        exit(4);
    }
    fprintf(stderr,"Option(s) OK!\n");

    /* Association de la socket s à l'adresse obtenue par résolution */
    if (bind(s,resol->ai_addr,sizeof(struct sockaddr_in))<0) {
        perror("bind");
        exit(5);
    }

    // Libération de la mémoire
    freeaddrinfo(resol);
    fprintf(stderr,"bind!\n");

    /* la socket est prête à recevoir */
    if (listen(s,SOMAXCONN)<0) {
        perror("listen");
        exit(6);
    }
    fprintf(stderr,"listen!\n");

    // while(1) writeSrcInDst(s, 1);

    while(1) {
        fprintf(stderr, "\n\nNOUVEAU MESSAGE\n");
        len = sizeof(struct sockaddr_in);
        if( (n=accept(s,(struct sockaddr *)&client,(socklen_t*)&len)) < 0 ) {
            perror("accept");
            exit(7);
        }

        char hotec[NI_MAXHOST], portc[NI_MAXSERV];

        err = getnameinfo((struct sockaddr*)&client,len,ho-
tec,NI_MAXHOST,portc,NI_MAXSERV,0);
        if (err < 0 ){
            fprintf(stderr,"résolution client (%i): %s\n",n,gai_strerror(err));
        }else{
            fprintf(stderr,"accept! (%i) ip=%s port=%s\n",n,hotec,portc);
        }
        writeSrcInDst(n, 1);
    }
}

void extin(int tun, char * hote, char * port) {
    char ip[NI_MAXHOST];
    struct addrinfo *resol;
    char * tmpdst;
    int _socket;

    while(1) {
```

```
        /* Résolution de l'hôte */
        if (getaddrinfo(hote,port,NULL, &resol) < 0 ){
            perror("résolution adresse");
            exit(2);
        }

        /* On extrait l'addresse IP */
        /* il faut faire l'allocation mémoire */
        sprintf(ip,
            "%s",
            inet_ntoa(((struct sockaddr_in*)resol->ai_addr)->sin_addr));

        /* Création de la socket, de type TCP / IP */
        /* On ne considère que la première adresse renvoyée par getaddrinfo */
        if ((_socket=socket(resol->ai_family,resol->ai_socktype, resol->ai_protocol))<0)
{
            perror("allocation de socket");
            exit(3);
        }
        fprintf(stderr,"le n° de la socket est : %i\n",_socket);

        /* Connexion */
        fprintf(stderr,"Essai de connexion à %s (%s) sur le port %s\n\n",
            hote,ip,port);


        while (connect(_socket,resol->ai_addr,sizeof(struct sockaddr_in))<0) { }

        freeaddrinfo(resol); /* /!\ Libération mémoire */

        writeSrcInDst(tun, _socket);

        close(_socket);
    }

}
```

3.1.3.2 Tester avec un "client" **ping6 fc00:1234:ffff::10** pour injecter du trafic comme dans la partie précédente.

On obtient aucune réponse, ce qui parait normal vu que l'autre extrémité du tunnel n'est pas encore configuré.

3.2. Redirection du trafic sortant

3.2.2 Que devrait-il se passer alors pour ce trafic ?

Tun0 reçoit bien les informations arrivant du paquet, il les retransmet alors au destinataire prévu au départ qui répondre.
Lorsque le tunnel est configuré (extrémité configurées des deux VMs) le ping peut être transmis normalement.

3.2.4 Proposer des tests de connectivité. Tester et vérifier.

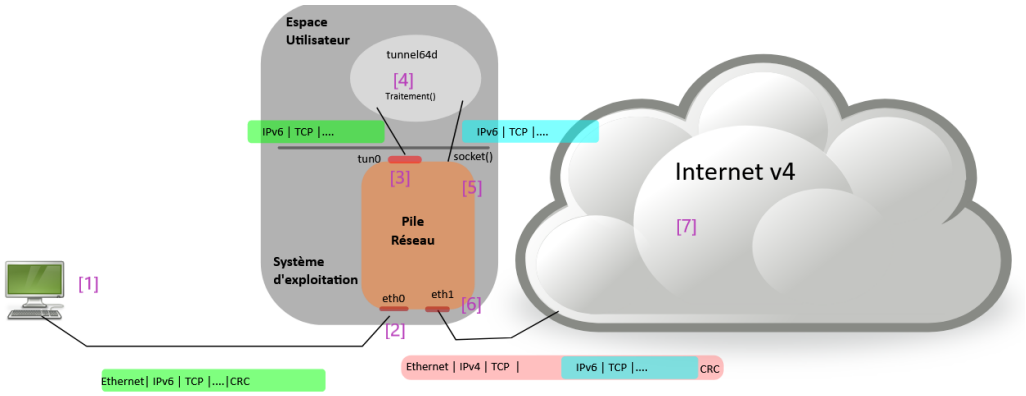
Nous essayons de ping6 depuis VM3 à VM1-6, depuis VM3-6 à VM1-6 et vice-versa. Tous les tests sont alors concluants.

3.3 Intégration finale des données

Sur VM1 et VM3 on lance le programme permettant l'écoute des paquets. Ce programme va lancer en parallèle un programme permettant l'envoi des données présents sur tun0 via un sémaphore.

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

Un packet partant du point [1] transite en IPv6 vers eth0 dans le système d'exploitation ([2]). Le paquet est par la suite transmis à tun0 du même OS ([3]), il est traité dans la partie [4] qui correspond au programme tunaloc. Ensuite ce paquet est envoyé par un le protocole TCP de la socket [5]. Il est envoyé par l'interface eth1 ([6]) en IPv4 vers l'internet IPv4 [7].



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

3.5.1 Créer un exécutable **tunnel64d** qui crée un service offrant un tunnel TCP bidirectionnel.

Nous avons choisi de retirer les commentaires du document pour une question purement pratique.

Notre fichier de configuration est donc :

```
tun=tun0
inip=172.16.2.131
inport=123
options=empty
outip=172.16.2.163
outport=123
```

Il est present de le fichier files des VMs (/vagrant/files)

Le fichier tunnel64d est présent dans /mnt/partage/

3. Validation fonctionnelle

Sur VM1

```
m1reseaux@vm1:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:f8:33:b7 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef8:33b7/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:12:26:70 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.131/28 brd 172.16.2.143 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe12:2670/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:d2:7e:33 brd ff:ff:ff:ff:ff:ff
    inet6 fc00:1234:3::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed2:7e33/64 scope link
       valid_lft forever preferred_lft forever
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
    inet6 fc00:1234:ffff::1/64 scope global
       valid_lft forever preferred_lft forever
m1reseaux@vm1:~$ ip route
default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
```

172.16.2.128/28 dev eth1 proto kernel scope link src 172.16.2.131

172.16.2.160/28 via 172.16.2.132 dev eth1

m1reseaux@vm1:~$ ip -6 route

fc00:1234:1::/64 via fc00:1234:3::16 dev eth2 metric 1024

fc00:1234:2::/64 via fc00:1234:3::16 dev eth2 metric 1024

fc00:1234:3::/64 dev eth2 proto kernel metric 256

fc00:1234:4::/64 via fc00:1234:ffff::10 dev tun0 metric 1024

fc00:1234:ffff::/64 dev tun0 proto kernel metric 256

fe80::/64 dev eth0 proto kernel metric 256

fe80::/64 dev eth1 proto kernel metric 256

fe80::/64 dev eth2 proto kernel metric 256

Sur VM2

m1reseaux@vm2:~$ ip addr

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default

  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

  inet6 ::1/128 scope host

    valid_lft forever preferred_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

  link/ether 08:00:27:f8:33:b7 brd ff:ff:ff:ff:ff:ff

  inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0

    valid_lft forever preferred_lft forever

  inet6 fe80::a00:27ff:fef8:33b7/64 scope link

    valid_lft forever preferred_lft forever

3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

  link/ether 08:00:27:f9:d8:66 brd ff:ff:ff:ff:ff:ff

  inet 172.16.2.132/28 brd 172.16.2.143 scope global eth1

    valid_lft forever preferred_lft forever

  inet6 fe80::a00:27ff:fef9:d866/64 scope link

    valid_lft forever preferred_lft forever

4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

  link/ether 08:00:27:cb:9a:10 brd ff:ff:ff:ff:ff:ff

  inet 172.16.2.162/28 brd 172.16.2.175 scope global eth2

    valid_lft forever preferred_lft forever

```
   inet6 fe80::a00:27ff:fecb:9a10/64 scope link
      valid_lft forever preferred_lft forever
m1reseaux@vm2:~$ ip route
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
172.16.2.128/28 dev eth1  proto kernel  scope link  src 172.16.2.132
172.16.2.160/28 dev eth2  proto kernel  scope link  src 172.16.2.162
m1reseaux@vm2:~$ ip -6 route
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
```

Sur VM3

```
m1reseaux@vm3:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
   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
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
   link/ether 08:00:27:f8:33:b7 brd ff:ff:ff:ff:ff:ff
   inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
      valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fef8:33b7/64 scope link
      valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
   link/ether 08:00:27:58:97:b4 brd ff:ff:ff:ff:ff:ff
   inet 172.16.2.163/28 brd 172.16.2.175 scope global eth1
      valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe58:97b4/64 scope link
      valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
   link/ether 08:00:27:29:82:af brd ff:ff:ff:ff:ff:ff
   inet6 fc00:1234:4::3/64 scope global
      valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe29:82af/64 scope link
```

```
    valid_lft forever preferred_lft forever
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
  link/none
  inet6 fc00:1234:ffff::10/64 scope global
    valid_lft forever preferred_lft forever
m1reseaux@vm3:~$ ip route
default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
172.16.2.128/28 via 172.16.2.162 dev eth1
172.16.2.160/28 dev eth1  proto kernel  scope link  src 172.16.2.163
m1reseaux@vm3:~$ ip -6 route
fc00:1234:1::/64 via fc00:1234:4::36 dev eth2  metric 1024
fc00:1234:2::/64 via fc00:1234:4::36 dev eth2  metric 1024
fc00:1234:3::/64 via fc00:1234:ffff::1 dev tun0  metric 1024
fc00:1234:4::/64 dev eth2  proto kernel  metric 256
fc00:1234:ffff::/64 dev tun0  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
```

Sur VM1-6

```
m1reseaux@vm1-6:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
  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
  inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
  link/ether 08:00:27:f8:33:b7 brd ff:ff:ff:ff:ff:ff
  inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    valid_lft forever preferred_lft forever
  inet6 fe80::a00:27ff:fef8:33b7/64 scope link
    valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
  link/ether 08:00:27:52:3f:e8 brd ff:ff:ff:ff:ff:ff
```

```
   inet6 fc00:1234:1::16/64 scope global
      valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe52:3fe8/64 scope link
      valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
   link/ether 08:00:27:9b:7c:aa brd ff:ff:ff:ff:ff:ff
   inet6 fc00:1234:3::16/64 scope global
      valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fe9b:7caa/64 scope link
      valid_lft forever preferred_lft forever
m1reseaux@vm1-6:~$ ip route
default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
m1reseaux@vm1-6:~$ ip -6 route
fc00:1234:1::/64 dev eth1  proto kernel  metric 256
fc00:1234:2::/64 via fc00:1234:1::26 dev eth1  metric 1024
fc00:1234:3::/64 dev eth2  proto kernel  metric 256
fc00:1234:4::/64 via fc00:1234:3::1 dev eth2  metric 1024
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
```

Sur VM3-6

```
m1reseaux@vm3-6:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
   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
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
   link/ether 08:00:27:f8:33:b7 brd ff:ff:ff:ff:ff:ff
   inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
      valid_lft forever preferred_lft forever
   inet6 fe80::a00:27ff:fef8:33b7/64 scope link
      valid_lft forever preferred_lft forever
```

```
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:10:3a:79 brd ff:ff:ff:ff:ff:ff
    inet6 fc00:1234:2::36/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe10:3a79/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:ec:07:fa brd ff:ff:ff:ff:ff:ff
    inet6 fc00:1234:4::36/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feec:7fa/64 scope link
       valid_lft forever preferred_lft forever
m1reseaux@vm3-6:~$ ip route
default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0  proto kernel  scope link  src 10.0.2.15
m1reseaux@vm3-6:~$ ip -6 route
fc00:1234:1::/64 via fc00:1234:2::26 dev eth1  metric 1024
fc00:1234:2::/64 dev eth1  proto kernel  metric 256
fc00:1234:3::/64 via fc00:1234:4::3 dev eth2  metric 1024
fc00:1234:4::/64 dev eth2  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
```

Depuis VM1-6 on peux ping VM3-6 :

```
m1reseaux@vm1-6:~$ ping6 fc00:1234:4::36
PING fc00:1234:4::36(fc00:1234:4::36) 56 data bytes
64 bytes from fc00:1234:4::36: icmp_seq=1 ttl=62 time=75.8 ms
64 bytes from fc00:1234:4::36: icmp_seq=2 ttl=62 time=30.6 ms
64 bytes from fc00:1234:4::36: icmp_seq=3 ttl=62 time=8.37 ms
64 bytes from fc00:1234:4::36: icmp_seq=4 ttl=62 time=3.68 ms
```

On peut également tester VM1-6 depuis VM3-6 :

```
m1reseaux@vm3-6:~$ ping6 fc00:1234:3::16
PING fc00:1234:3::16(fc00:1234:3::16) 56 data bytes
64 bytes from fc00:1234:3::16: icmp_seq=1 ttl=62 time=6.83 ms
64 bytes from fc00:1234:3::16: icmp_seq=2 ttl=62 time=3.06 ms
```

```
64 bytes from fc00:1234:3::16: icmp_seq=3 ttl=62 time=3.52 ms
```

Et VM3 depuis VM1-6 :

```
m1reseaux@vm1-6:~$ ping6 fc00:1234:4::3
PING fc00:1234:4::3(fc00:1234:4::3) 56 data bytes
64 bytes from fc00:1234:4::3: icmp_seq=1 ttl=63 time=2.54 ms
64 bytes from fc00:1234:4::3: icmp_seq=2 ttl=63 time=2.48 ms
```

Nous n'arrivons pas à faire fonctionner netcat correctement et donc ne pouvons tester à l'envoi du message.

Pour les performances nous avons :

```
m1reseaux@vm1-6:~$ iperf3 -6 -c fc00:1234:4::36 -n 1 -l 10
Connecting to host fc00:1234:4::36, port 5201
[  4] local fc00:1234:3::16 port 37454 connected to fc00:1234:4::36 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-0.00   sec   100 Bytes  3.38 Mbits/sec    0   13.9 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-0.00   sec   100 Bytes  3.38 Mbits/sec    0             sender
[  4]   0.00-0.00   sec  10.0 Bytes   338 Kbits/sec                  receiver

iperf Done.
```

Qui répond

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from fc00:1234:3::16, port 37453
[  5] local fc00:1234:4::36 port 5201 connected to fc00:1234:3::16 port 37454
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-0.06   sec  10.0 Bytes  1.38 Kbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-0.06   sec   100 Bytes  13.8 Kbits/sec    0             sender
[  5]   0.00-0.06   sec  10.0 Bytes  1.38 Kbits/sec                  receiver
```

```
m1reseaux@vm1-6:~$ iperf3 -6 -c fc00:1234:4::36 -n 1 -l 2K
Connecting to host fc00:1234:4::36, port 5201
[  4] local fc00:1234:3::16 port 37456 connected to fc00:1234:4::36 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-0.00   sec  20.0 KBytes  42.1 Mbits/sec    0   13.9 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-0.00   sec  20.0 KBytes  42.1 Mbits/sec    0             sender
[  4]   0.00-0.00   sec  10.4 KBytes  21.8 Mbits/sec                  receiver

iperf Done.
```

Qui donne :

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from fc00:1234:3::16, port 37455
[  5] local fc00:1234:4::36 port 5201 connected to fc00:1234:3::16 port 37456
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-0.13   sec  10.4 KBytes   677 Kbits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-0.13   sec  20.0 KBytes  1.31 Mbits/sec    0             sender
[  5]   0.00-0.13   sec  10.4 KBytes   677 Kbits/sec                  receiver
```

```
m1reseaux@vm1-6:~$ iperf3 -6 -c fc00:1234:4::36 -n 1 -l 128K
Connecting to host fc00:1234:4::36, port 5201
[  4] local fc00:1234:3::16 port 37458 connected to fc00:1234:4::36 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-0.00   sec  66.9 KBytes   139 Mbits/sec    0   13.9 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-0.00   sec  66.9 KBytes   139 Mbits/sec    0             sender
[  4]   0.00-0.00   sec  0.00 Bytes  0.00 bits/sec                    receiver

iperf Done.
```

Qui donne :

```
-----------------------------------------------------------
```

```
Server listening on 5201
-----------------------------------------------------------
Accepted connection from fc00:1234:3::16, port 37457
[  5] local fc00:1234:4::36 port 5201 connected to fc00:1234:3::16 port 37458
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-0.08   sec  0.00 Bytes  0.00 bits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-0.08   sec  66.9 KBytes  7.10 Mbits/sec    0             sender
[  5]   0.00-0.08   sec  0.00 Bytes  0.00 bits/sec                  receiver
```

```
m1reseaux@vm1-6:~$ iperf3 -6 -c fc00:1234:4::36 -n 1 -l 1M
Connecting to host fc00:1234:4::36, port 5201
[  4] local fc00:1234:3::16 port 37460 connected to fc00:1234:4::36 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-0.01   sec  66.9 KBytes  53.7 Mbits/sec    0   13.9 KBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-0.01   sec  66.9 KBytes  53.7 Mbits/sec    0             sender
[  4]   0.00-0.01   sec  0.00 Bytes  0.00 bits/sec                  receiver

iperf Done.
```

Qui donne :

```
Accepted connection from fc00:1234:3::16, port 37459
[  5] local fc00:1234:4::36 port 5201 connected to fc00:1234:3::16 port 37460
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-0.04   sec  0.00 Bytes  0.00 bits/sec
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-0.04   sec  66.9 KBytes  14.0 Mbits/sec    0             sender
[  5]   0.00-0.04   sec  0.00 Bytes  0.00 bits/sec                  receiver
```

Les résultats semblent cohérents, on reçoit bien les données avec des débits qui semblent raisonnables. Cependant nous ne comprenons pas totalement ces résultats...

5. Améliorations

Nous avons choisi l'amélioration vers la configuration *salt.*

Pour mettre en place cette gestion du tunnel avec salt il faut résoudre un problème majeur. En effet, notre programme tourne en boucle sur écoute, mais salt demande que le programme termine. Nous avons donc deux ressources qui s'opposent sur ce point.

Pour résoudre ce point, nous avons choisi d'utiliser le service *Screen* disponible sur Linux. Ce package permet de lancer un processus en arrière-plan, puis d'y accéder depuis n'importe quel autre terminal sur la même machine.
Grâce à ce package, nous pouvons lancer notre programme sur un terminal et le laisser tourner de côté. Ensuite, nous pouvons configurer notre tunnel avec salt depuis un autre terminal pour lancer facilement notre programme de réseau.

NOTA BENE

Pour lancer les machines il faut faire

```
$ sudo -s
```

Puis

```
salt-call state.apply
```

La configuration salt permet de lancer directement les serveurs, le temps que package screen se télécharge cela peut prendre un peu de temps.

Une fois les machines configurées on peut tester les différentes fonctionnalités. Cependant nous avons fait le choix de ne pas inclure directement les packages tels que iperf3 et netcat. Il faut donc les installer manuellement.