Le routage, compléments

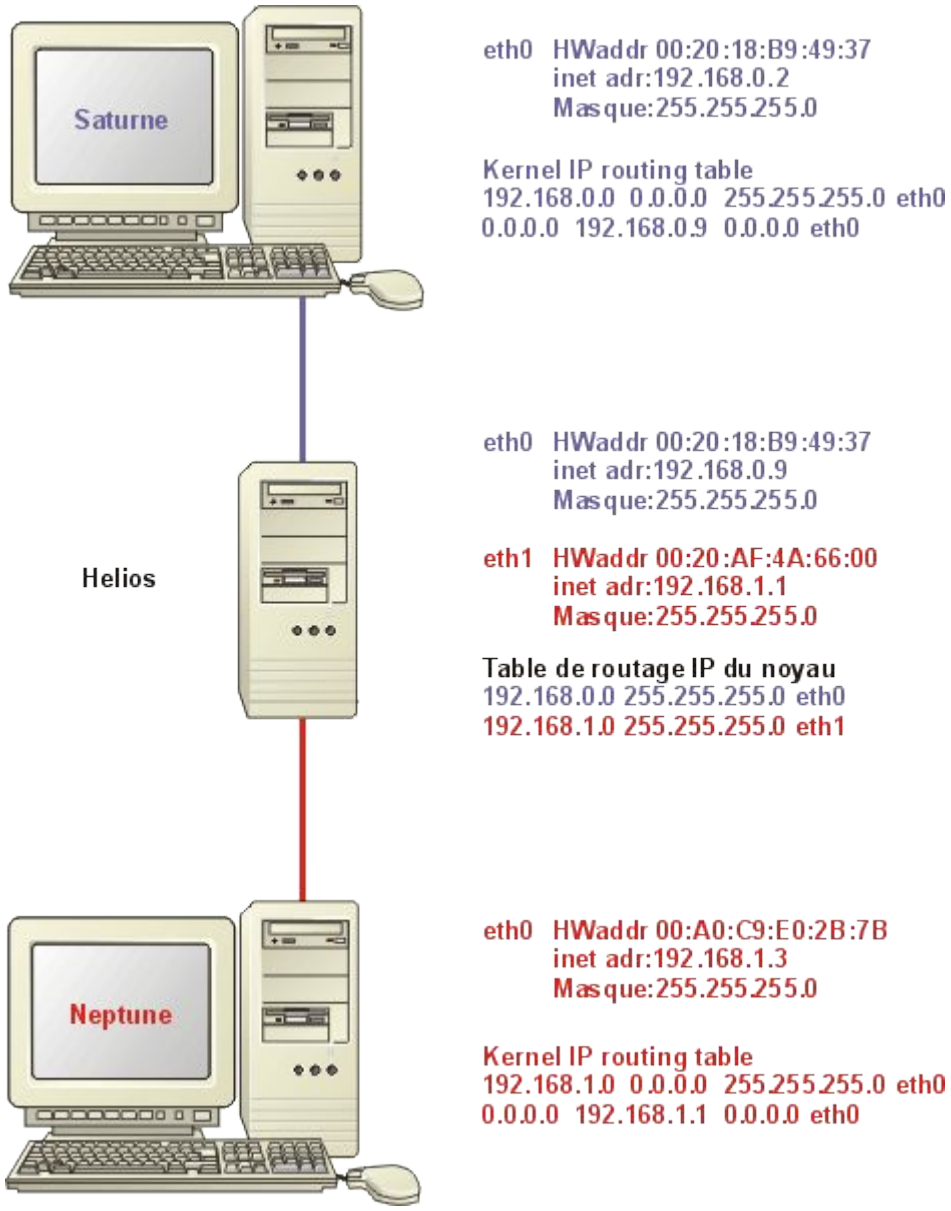
Le routage est une notion pas forcément facile à comprendre.

Ce court chapitre a pour but de donner quelques exemples supplémentaires qui aideront à comprendre, sur des cas concrets, ce qui est expliqué dans le chapitre "le routage" :

- Un cas tout simple d'un routeur entre deux réseaux privés,
- un routeur avec masquage d'adresse, destiné à permettre l'accès des clients d'un LAN à l'Internet, par une connexion permanente unique (partage de connexion Internet),
- un montage un peu plus tordu, mélangeant les deux premiers cas, destiné à montrer comment un paquet peut passer par des sens uniques, passer dans un sens mais pas dans l'autre, emprunter des chemins de traverse...

Plan du chapitre

Une manip. très simple



Dans cet exemple, nous avons un cas extrêmement simple :

- Un réseau bleu, dont l'adresse IP est 192.168.0.0, avec le classique masque 255.255.255.0,
- un réseau rouge, dont l'adresse IP est 192.168.1.0, même masque,
- entre les deux, un routeur qui dispose donc deux deux adresses IP : 192.168.0.9 dans le réseau bleu 192.168.1.1 dans le réseau rouge. Ce routeur s'appelle Helios.

Dans le réseau bleu, Saturne, d'adresse IP 192.168.0.2, envoie un ping sur Neptune, qui se trouve dans le réseau rouge avec l'IP 192.168.1.3

Saturne

Machine sous Debian Woody.

Configuration réseau :

```
saturne:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:00:E8:78:9A:1F
          inet addr:192.168.0.2 Bcast:192.168.0.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:2674831 errors:7 dropped:559 overruns:7 frame:0
          TX packets:1880271 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:100
          RX bytes:2193374824 (2.0 GiB) TX bytes:932807313 (889.5 MiB)
          Interrupt:10 Base address:0xa000

lo        Link encap:Local Loopback...
```

Observez bien les deux adresses qui vont nous être utiles par la suite :

```
HW addr  :00:00:E8:78:9A:1F
inet addr:192.168.0.2
```

Configuration de routage :

```
saturne:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
0.0.0.0 192.168.0.9 0.0.0.0 UG 0 0 0 eth0
```

Notez la route par défaut :
Il faut passer par eth0 et joindre le noeud 192.1686.0.9.

Neptune

Également sous Debian Woody.

Configuration réseau :

```
neptune:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:A0:C9:E0:2B:7B
          inet addr:192.168.1.3  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:527923 errors:0 dropped:0 overruns:0 frame:0
          TX packets:422325 errors:0 dropped:0 overruns:0 carrier:0
          collisions:21182 txqueuelen:100
          RX bytes:269846768 (257.3 MiB)  TX bytes:240923195 (229.7 MiB)
          Interrupt:11 Base address:0x5000

lo        Link encap:Local Loopback...
```

Observez bien ici aussi les deux adresses qui vont nous être utiles par la suite :

```
HW addr  : 00:A0:C9:E0:2B:7B
inet addr:192.168.1.3
```

Configuration de routage :

```
neptune:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
0.0.0.0 192.168.1.1 0.0.0.0 UG 0 0 0 eth0
```

Notez là encore la route par défaut :
Il faut passer par eth0 et joindre le noeud 192.168.1.1.

Helios

Sous Mandrake 9.0.

Là, ça va être un peu plus compliqué, il y a deux interfaces réseau :

```
[root@helios root]# ifconfig
eth0      Lien encap:Ethernet  HWaddr 00:20:18:B9:49:37
          inet adr:192.168.0.9  Bcast:192.168.0.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4338949 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4968746 errors:0 dropped:0 overruns:0 carrier:0
          collisions:11750 lg file transmission:100
          RX bytes:503746782 (480.4 Mb)  TX bytes:547552476 (522.1 Mb)
          Interruption:9 Adresse de base:0x5000

eth1      Lien encap:Ethernet  HWaddr 00:20:AF:4A:66:00
          inet adr:192.168.1.1  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4403671 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3762159 errors:0 dropped:0 overruns:0 carrier:0
          collisions:34316 lg file transmission:100
          RX bytes:2763013200 (2635.0 Mb)  TX bytes:2646670391 (2524.0 Mb)
          Interruption:5 Adresse de base:0x210

lo        Lien encap:Boucle locale...
```

Notons les adresses utiles pour chaque réseau :

```
HW addr :00:20:18:B9:49:37
inet adr:192.168.0.9
```

Pour le réseau 192.168.0.0

```
HWaddr 00:20:AF:4A:66:00
inet adr:192.168.1.1
```

Pour le réseau 192.168.1.0

Configuration de routage :

```
[root@helios root]# route -n
Table de routage IP du noyau
Destination Passerelle Genmask Indic Metric Ref Use Iface
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
```

Notez que cette machine n'est qu'un routeur, son unique rôle, simplifié à l'extrême, est d'interconnecter les deux réseaux 192.168.0.0 et 192.168.1.0

Aucune route par défaut n'est indiquée ici.

Tout ce que nous observons, c'est que Helios sait que :

- pour aller dans le réseau bleu (192.168.0.0), il faut envoyer les paquets sur eth0,
- pour aller dans le réseau rouge (192.168.1.0), il faut envoyer les paquets sur eth1.

C'est bien, mais ça ne suffit pas pour que Helios fonctionne comme un routeur. Nous avons un peu besoin d'IPtables :

```
iptables -A FORWARD -j ACCEPT  # c'est pas bien compliqué...
```

Il faut aussi s'assurer que l'on a autorisé le noyau à faire le routage :

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

Et ça va router...

```
saturne:~# ping -c 1 neptune
PING neptune.eme.org (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: icmp_seq=0 ttl=254 time=1.2 ms

--- neptune.eme.org ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.2/1.2/1.2 ms
```

Si le ping passe, c'est que ça route.

Juste pour le fun :

```
neptune:~# ping -c 1 saturne
PING saturne.eme.org (192.168.0.2): 56 data bytes
64 bytes from 192.168.0.2: icmp_seq=0 ttl=254 time=1.1 ms

--- saturne.eme.org ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.1/1.1/1.1 ms
```

Ça marche aussi dans l'autre sens.

Snif.

On regarde tout ça avec des sniffers, l'un sur le réseau bleu et l'autre sur le réseau rouge.

```
saturne:~# ping -c 1 neptune
```

Les traces qui suivent ont été expurgées de ce qui nous est inutile ici.

Tout ce qu'il est important d'observer est surligné

```
Frame 1 ...
Ethernet II
    Destination: 00:20:18:b9:49:37
    Source: 00:00:e8:78:9a:1f
    Type: IP (0x0800)

Internet Protocol, Src Addr: saturne.eme.org
                            (192.168.0.2),
                 Dst Addr: neptune.eme.org
                            (192.168.1.3)
    ...
    Time to live: 64
    Protocol: ICMP (0x01)
    ...
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
...

Frame 2 ...
Ethernet II
    Destination: 00:00:e8:78:9a:1f
    Source: 00:20:18:b9:49:37
    Type: IP (0x0800)
Internet Protocol, Src Addr: neptune.eme.org
                            (192.168.1.3),
                 Dst Addr: saturne.eme.org
                            (192.168.0.2)
    ...
    Time to live: 254
    Protocol: ICMP (0x01)
    ...
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
```

```
Frame 1 ...
Ethernet II
    Destination: 00:a0:c9:e0:2b:7b
    Source: 00:20:af:4a:66:00
    Type: IP (0x0800)

Internet Protocol, Src Addr: saturne.eme.org
                            (192.168.0.2),
                 Dst Addr: neptune.eme.org
                            (192.168.1.3)
    ...
    Time to live: 63
    Protocol: ICMP (0x01)
    ...
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
...

Frame 2...
Ethernet II
    Destination: 00:20:af:4a:66:00
    Source: 00:a0:c9:e0:2b:7b
    Type: IP (0x0800)
Internet Protocol, Src Addr: neptune.eme.org
                            (192.168.1.3),
                 Dst Addr: saturne.eme.org
                            (192.168.0.2)
    ...
    Time to live: 255
    Protocol: ICMP (0x01)
    ...
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
```

Vous avez noté les points fondamentaux :

- Au **niveau IP**, les adresses source et destination sont les **mêmes des deux côtés** de Helios,
- au **niveau Ethernet**, il n'en va pas de même :
  - Dans la trame 1 **(echo request)** :
    - **côté bleu** :
      - l'adresse MAC source est bien celle de Saturne,
      - l'adresse MAC **destination** est celle du routeur **Helios** côté bleu,
    - **côté rouge** :
      - l'adresse MAC **source** est celle du routeur **Helios** côté rouge,
      - l'adresse MAC destination est bien celle de Neptune,
  - Dans la trame 2 **(echo reply)** :
    - **côté rouge** :
      - l'adresse MAC source est bien celle de Neptune,
      - l'adresse MAC **destination** est celle u routeur **Helios** côté rouge,
    - côté bleu :
      - l'adresse MAC **source** est celle du routeur **Helios** côté bleu
      - l'adresse MAC de destination est bien celle de Saturne.

Autrement dit :

- Lorsque Saturne doit envoyer son "echo request" à Neptune :
  - ICMP (couche 4) prépare son paquet et le refile à la couche IP (couche 3),
  - IP constate que la cible n'est pas dans son réseau, cherche le routeur à joindre pour passer dans le réseau rouge. Il n'y a qu'une passerelle par défaut : 192.168.0.9.
    Advienne que pourra, on va lui envoyer le paquet, mais les adresses IP source et destination ne sont pas modifiées. En fait, IP va rechercher l'adresse MAC d'Helios dans son réseau et trouve que 192.168.0.9 a pour adresse MAC 00:20:18:b9:49:37.
    IP va donc déposer cette adresse de destination, avant de refiler le paquet à la couche 2.
  - Ethernet va donc, sans se poser de question gérer ce paquet comme s'il s'adressait à Helios.
  - Helios reçoit ce paquet, le fait remonter jusqu'au niveau IP côté bleu (192.168.0.0),
  - là, la couche IP constate que Helios n'est pas le destinataire, puisque c'est Neptune (192.168.1.3),
  - Netfilter passe le paquer à la couche IP rouge (192.168.1.0)
  - La couche IP rouge va chercher l'adresse MAC de Neptune, puisqu'elle sait que Neptune est dans le même réseau qu'elle.

Et voilà le travail.

Un peu de philosophie

Jouons avec traceroute :

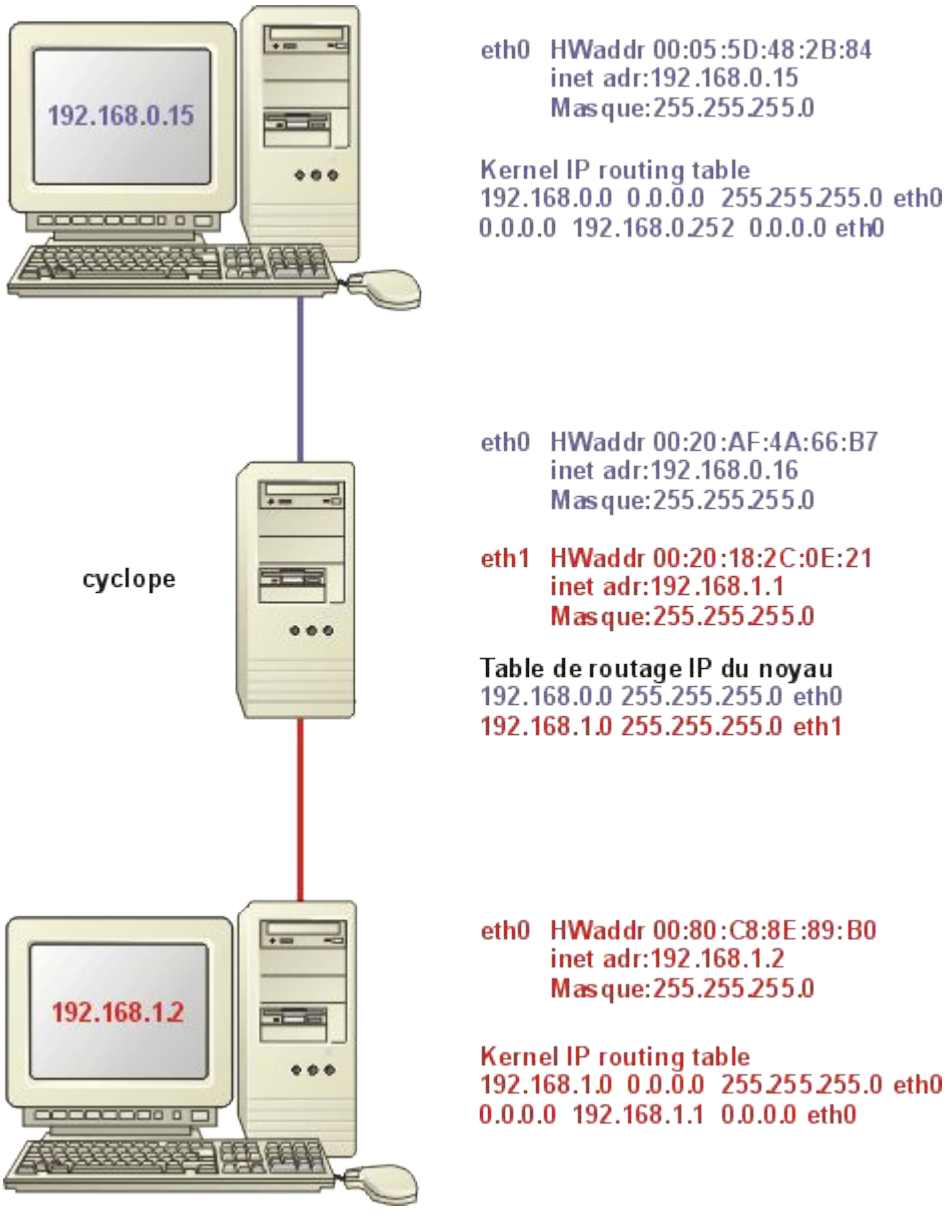
```
saturne:~# traceroute -I -n neptune
traceroute to neptune.eme.org (192.168.1.3), 30 hops max, 38 byte packets
1 192.168.0.9 0.824 ms 0.741 ms 0.740 ms
2 192.168.1.3 1.246 ms 0.922 ms 0.968 ms


neptune:~# traceroute -I -n saturne
traceroute to saturne.eme.org (192.168.0.2), 30 hops max, 38 byte packets
1 192.168.1.1 0.902 ms 0.705 ms 0.675 ms
2 192.168.0.2 1.108 ms 1.033 ms 1.008 ms
```

Bien entendu, nous passons toujours par Helios au "hop" 1, quelque soit le sens du traceroute, mais nous le voyons du côté bleu avec l'IP 192.168.0.9 et du côté rouge avec l'IP 192.168.1.1.

Ça vous étonne ?

Routage avec MASQUERADE

Plus fort. Nous reprenons ici le principe du routage entre un réseau local utilisant des adresses IP privées et une connexion Internet, avec une "vraie" IP, une qui peut se promener de réseau en réseau en dehors de notre microcosme.



Pour réaliser la manipulation, nous réalisons quelque chose de très similaire à ce que nous avons vu en page précédente, seules les adresses MAC et IP changent, de même que le nom de la passerelle.

Notre routeur "cyclope", qui connecte nos réseaux 192.168.1.0 et 192.168.0.0 va, ici, réaliser un masquage d'adresse pour 192.168.1.0.

Autrement dit :

- 192.168.1.0 joue le rôle du réseau local,
- 192.168.0.0 joue le rôle de l'Internet.

Ici, cyclope ne va pas se comporter en routeur "normal", il fera du masquage d'adresse de tout ce qui vient du réseau 192.168.1.0 et sort par eth0.

Avec IPtables, nous utilisons la commande suivante :

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

192.168.1.2 envoie un ping sur 192.168.0.15 :

Côté réseau 192.168.1.0 (LAN)

```
# Le ping...

Frame 1 ...
Ethernet II, Src: 00:80:c8:8e:89:b0,
              Dst: 00:20:18:2c:0e:21
    Destination: 00:20:18:2c:0e:21 (192.168.1.1)
    Source: 00:80:c8:8e:89:b0 (192.168.1.2)

# Au niveau Ethernet (niveau 2)
#    - La source est la vraie source
#    - la destination devient le routeur
# Jusque là, tout est normal.

    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.1.2,
                   Dst Addr: 192.168.0.15

# Au niveau IP (niveau 3)
#    - La source est la vraie source
#    - la destination est la vraie destination
# Assez banal, finalement...


    Version: 4
    ...
    Protocol: ICMP (0x01)
    Header checksum: 0xb847 (correct)
    Source: 192.168.1.2
    Destination: 192.168.0.15
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)

# Et la réponse...

Frame 2 ...
Ethernet II, Src: 00:20:18:2c:0e:21,
              Dst: 00:80:c8:8e:89:b0
    Destination: 00:80:c8:8e:89:b0 (192.168.1.2)
    Source: 00:20:18:2c:0e:21 (192.168.1.1)

# Au niveau Ethernet, le routeur répond
# au client, c'est encore normal.

    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.0.15,
                   Dst Addr: 192.168.1.2

# Au niveau IP, de ce côté-ci, on a bien
# l'impression que le serveur a répondu directement
# au client, tout semble normal.
# En fait, MASQUERADE a bien fait son travail.

    Version: 4
    ...
    Protocol: ICMP (0x01)
    Header checksum: 0x25b4 (correct)
    Source: 192.168.0.15
    Destination: 192.168.1.2
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
```

Côté réseau 192.168.0.0 (Pseudo Internet)

```
# Le ping...

Frame 1 ...
Ethernet II, Src: 00:20:af:4a:66:b7,
              Dst: 00:05:5d:48:2b:84
    Destination: 00:05:5d:48:2b:84 (192.168.0.15)
    Source: 00:20:af:4a:66:b7 (192.168.0.16)

# Au niveau Ethernet (niveau 2)
#    - La source devient le routeur
#    - la destination est la vraie destination
# Jusque là, tout est normal.

    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.0.16,
                   Dst Addr: 192.168.0.15

# Au niveau IP (niveau 3)
#    - La source devient le routeur (192.168.0.16)
#    - la destination reste la vraie destination
# Il y a donc eu remplacement de l'adresse
# de la source par celle du routeur (MASQUERADE)...

    Version: 4
    ...
    Protocol: ICMP (0x01)
    Header checksum: 0xba39 (correct)
    Source: 192.168.0.16
    Destination: 192.168.0.15
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)

# Et la réponse...

Frame 2 ...
Ethernet II, Src: 00:05:5d:48:2b:84,
              Dst: 00:20:af:4a:66:b7
    Destination: 00:20:af:4a:66:b7 (192.168.0.16)
    Source: 00:05:5d:48:2b:84 (192.168.0.15)

# Au niveau Ethernet, le serveur répond au routeur
# Là encore, c'est normal.

    Type: IP (0x0800)
Internet Protocol, Src Addr: 192.168.0.15,
                   Dst Addr: 192.168.0.16

# Mais au niveau IP, le serveur répond aussi
# au routeur. C'est logique, à cause de MASQUERADE,
# le serveur croit que c'est le routeur qui
# a envoyé le ping.

    Version: 4
    ...
    Protocol: ICMP (0x01)
    Header checksum: 0x25a7 (correct)
    Source: 192.168.0.15
    Destination: 192.168.0.16
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
```

La morale de cette histoire est la suivante :

- Du côté du LAN, nous avons l'impression d'utiliser un routeur "normal", comme vu en page

précédente,

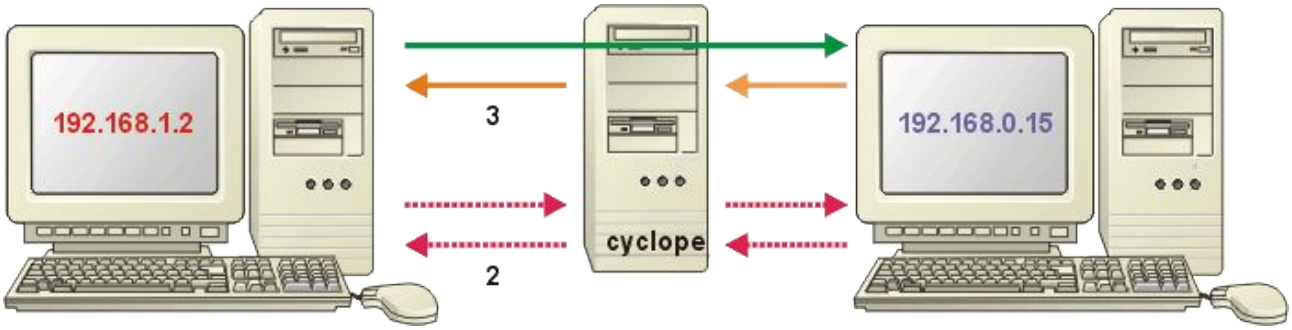
- du côté Internet, les clients du LAN sont masqués par le routeur. Côté Inet, tout semble provenir du routeur lui-même.

Ce n'est pas pour rien que l'on appelle cette technique : du masquage d'adresse (MASQUERADE) !

Pour finir de mettre les points sur les i :

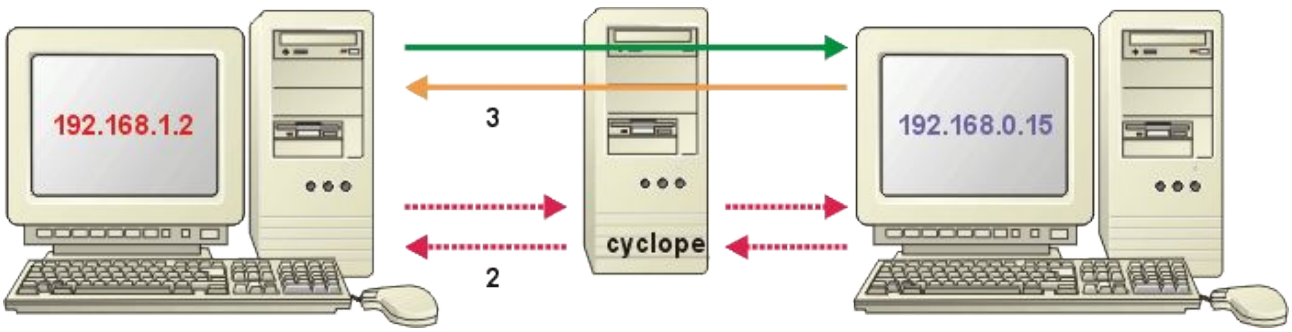
Dans un routage par masquage d'adresse :

- au niveau 2 (Ethernet) :
  - les requêtes sont envoyées du client sur le routeur, puis du routeur sur le serveur,
  - les réponses sont envoyées du serveur sur le routeur, puis du routeur sur le client.
- au niveau 3 (IP) :
  - les requêtes semblent aller directement du client sur le serveur,(mais le routeur remplace, au passage, l'adresse IP du client par la sienne)
  - les réponses vont du serveur au routeur, puis du routeur au client, mais le client a l'impression que la réponse lui parvient directement, parce que le routeur change son adresse de destination par celle du client initial.



Dans un routage simple, c'est plus simple :

- au niveau 2 (Ethernet) :
  - les requêtes sont envoyées du client sur le routeur, puis du routeur sur le serveur,
  - les réponses sont envoyées du serveur sur le routeur, puis du routeur sur le client.
- au niveau 3 (IP) :
  - les requêtes vont directement du client sur le serveur,
  - les réponses vont directement du serveur au client.



Jouons un peu...

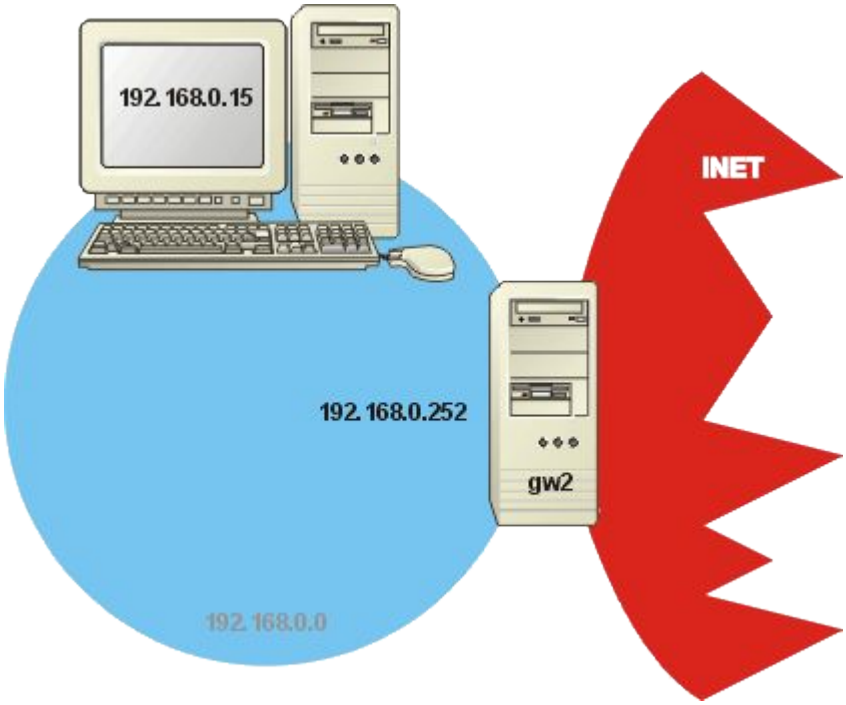
C'est lorsque l'on croit avoir tout bien compris que l'on tombe parfois dans le panneau...

Voyons un cas de figure un peu plus compliqué, mais intéressant.

Dans ce cas, déjà vu en page précédente, gw2 représente un routeur avec masquage d'adresse, de manière à permettre aux hôtes du réseau 192.168.0.0 d'accéder à l'Internet.

Les hôtes du réseau 192.168.0.0 connaissent uniquement :

- la route de leur propre réseau,
- une route par défaut : 192.168.0.252

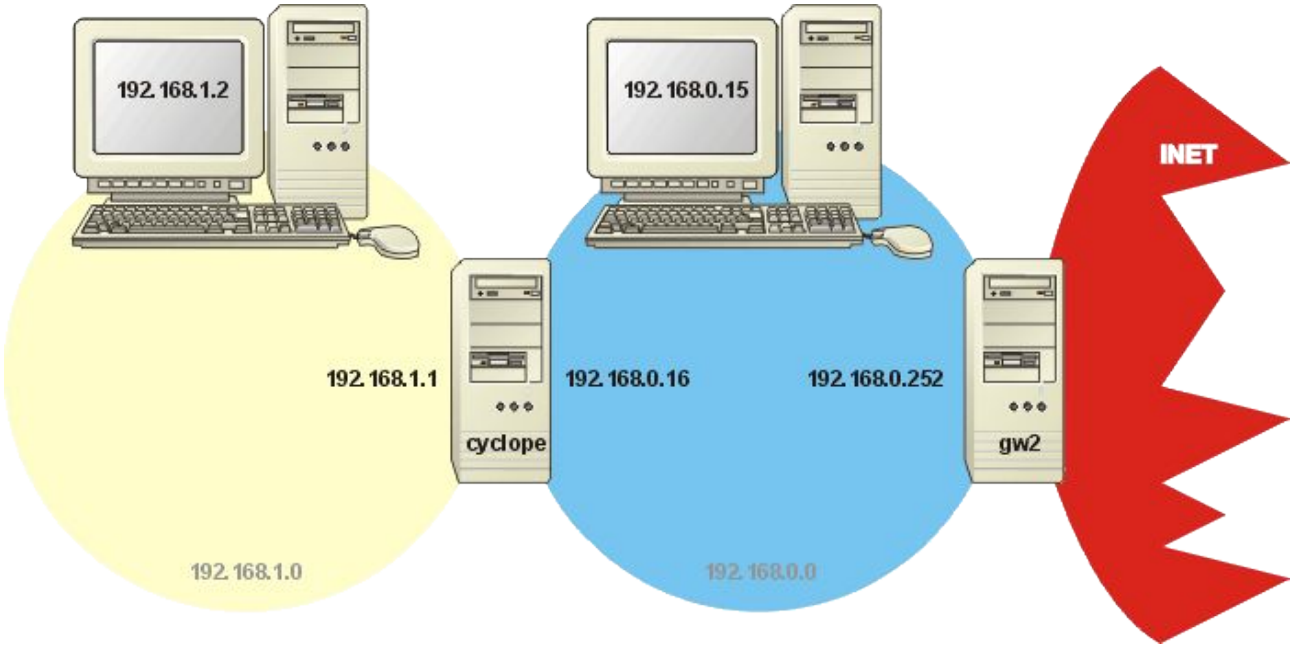


Hormis le détail de masquerade, gw2 agit comme un routeur tout à fait ordinaire. Sa table de routage est simple :

```
ca-marseille-14-119:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
80.8.136.1      0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
0.0.0.0         80.8.136.1      0.0.0.0         UG    0      0        0 ppp0
```

Mais compliquons un petit peu l'architecture...

Nous rajoutons un second réseau , 192.168.1.0, interconnecté au réseau 192.168.0.0 par une autre passerelle, que nous appellerons cyclope. Ce routeur va fonctionner sans masquage d'adresse (ce serait trop facile sinon).



Objectif de la manoeuvre

Nous aimerions bien que les choses se passent de la manière suivante :

- Le réseau jaune doit accéder à l'Internet,
- le réseau jaune doit accéder au réseau bleu,
- le réseau bleu doit accéder au réseau jaune, bien sûr.

Facile, me direz-vous ? Peut-être pas tant que ça...

De 192.168.1.0 vers INET

Il suffit d'indiquer aux hôtes du réseau jaune 192.168.1.1 comme passerelle par défaut. cyclope a une table de routage toujours très simple :

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Bon. un paquet part de 192.168.1.2 à destination, par exemple, de 213.186.35.33.

- L'émetteur, sachant que le destinataire n'est pas dans le réseau jaune, va envoyer le paquet sur cyclope,
- cyclope, voyant que ce n'est pas pour lui, va chercher une route pour joindre le destinataire. Malheureusement, il n'en connaît pas. Ça va se terminer par un "no route to host" ou un truc de ce genre.

Il nous faut donc modifier la configuration de cyclope, pour lui indiquer une route par défaut, qui pointera sur gw2 :

```
~# route add default gw 192.168.0.252 eth0

~# route -n
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.0.252   0.0.0.0         UG    0      0        0 eth0
```

La route par défaut de cyclope est placée, il sait maintenant où envoyer les paquets qui ne vont pas dans le réseau bleu (ni dans le réseau jaune, d'ailleurs).

Et rejouons le scénario :

Re-bon. un paquet part de 192.168.1.2 à destination, par exemple, de 213.186.35.33.

- L'émetteur, sachant que le destinataire n'est pas dans le réseau jaune, va envoyer le paquet sur cyclope,
- cyclope, voyant que ce n'est pas pour lui, va chercher une route pour joindre le destinataire. N'en trouvant pas une explicite, il va envoyer le paquet à gw2,
- celui-ci, ne connaissant pas non plus de route explicite pour ce paquet, va l'envoyer à sa passerelle par défaut, à savoir 80.8.136.1 dans l'exemple.

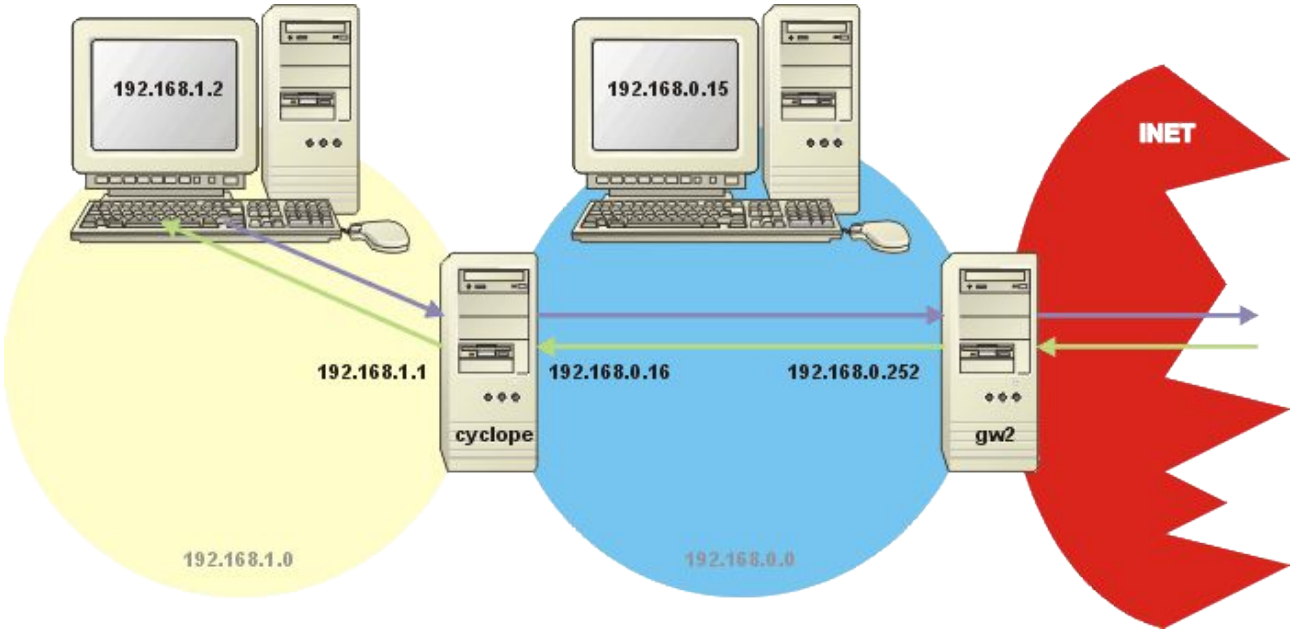
C'est parti. Après, ce n'est plus de notre compétence, mais de celle de notre fournisseur d'accès. Le paquet va arriver à destination.

Mais pour le retour ?

Pour le retour, ça va coincer, vous pensez bien... Lorsque la réponse va arriver sur gw2, après démasquage de l'adresse du destinataire, il lui restera à trouver une route qui mène à 192.168.1.0 et elle n'en connaît pas, donc, retour sur la passerelle par défaut. Ca ne va pas trop bien marcher. Il faut bricoler quelque chose sur gw2 :

```
~# route add -net 192.168.1.0 gw 192.168.0.16 netmask 255.255.255.0 eth1

~#route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
80.8.136.1      0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.1.0     192.168.0.16    255.255.255.0   UG    0      0        0 eth1
0.0.0.0         80.8.136.1      0.0.0.0         UG    0      0        0 ppp0
```

En rajoutant cette route, ça ira mieux. gw2 enverra le paquet à destination de 192.168.1.2 sur cyclope qui, lui, saura joindre le destinataire.



Voilà une bonne chose de faite. Notre réseau jaune est maintenant en mesure d'offrir à ses hôtes l'accès à l'Internet.

De 192.168.1.0 vers 192.168.0.0 (et vice versa)

Ce serait trop beau que ça marche du premier coup...

Sur le réseau bleu, il y a deux passerelles. Si on ne le dit pas à tous les hôtes du réseau bleu, elles ne le découvriront pas toutes seules. Elles ne connaissent, faut-il le rappeler, qu'une route par défaut, qui pointe sur gw2, à savoir 192.168.0.252.

Essayons quand même, depuis 192.168.0.15, de faire un ping sur 192.168.1.2. Ça passera le temps.

```
Envoi d'une requête 'ping' sur 192.168.1.1 avec 32 octets de données :

Réponse de 192.168.1.1 : octets=32 temps=1 ms TTL=255
Réponse de 192.168.1.1 : octets=32 temps<1ms TTL=255
Réponse de 192.168.1.1 : octets=32 temps<1ms TTL=255
Réponse de 192.168.1.1 : octets=32 temps<1ms TTL=255

Statistiques Ping pour 192.168.1.1:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 1ms, Moyenne = 0ms
```

Mince alors, ça marche !!! Comment ce peut-ce ?

En faisant du ping pong.

ping

- 192.168.0.15 ne connaît pas de route vers 192.168.1.2. Il envoie la balle à sa passerelle par défaut 192.168.0.252 (gw2),
- gw2, lui, connaît une route vers le réseau 192.168.1.0, c'est cyclope (192.168.0.16). Il lui

passe la balle,

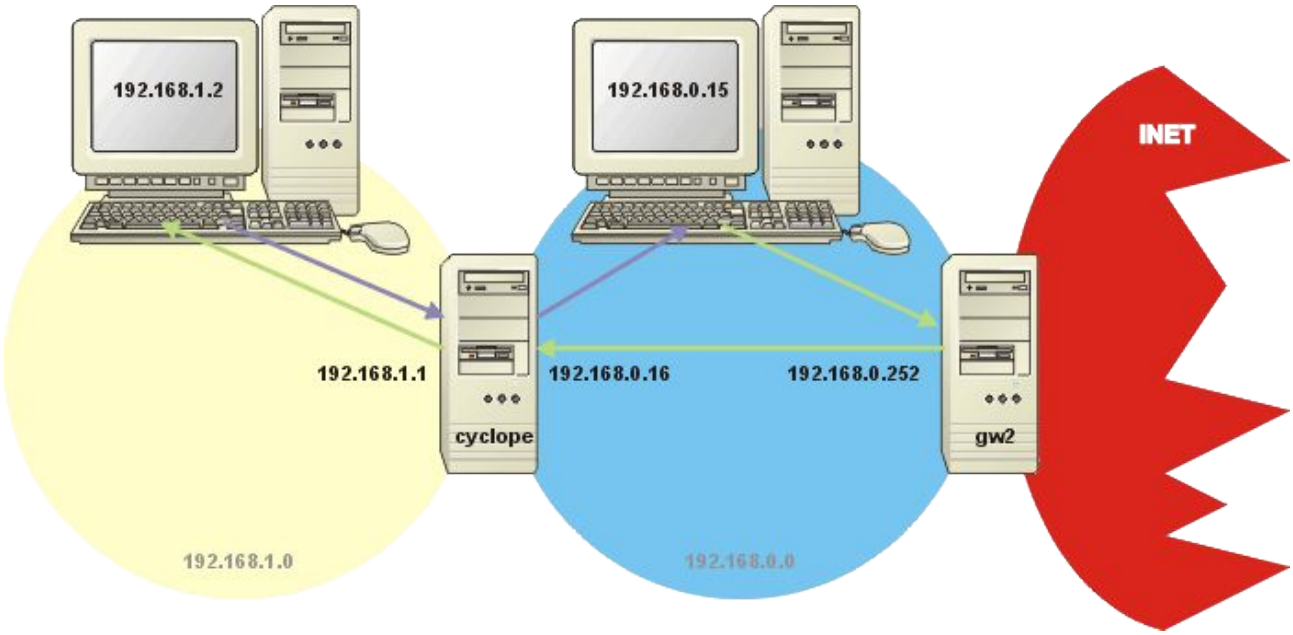
- cyclope envoie à 192.168.1.2.

pong

- 192.168.1.2 répond. Ne sachant pas joindre 192.168.0.15, il envoie à cyclope,
- cyclope sait joindre **directement** 192.168.0.15, il connaît la route vers ce réseau.

C'est une partie triangulaire, en quelque sorte.

Voici graphiquement ce que ça donne, mais dans l'autre sens, ping en vert et pong en violet :



Bien entendu, nous pouvions aussi indiquer à 192.168.0.15 la route directe pour aller dans le réseau 192.168.1.0, mais ce n'est pas forcément facile à faire de façon automatique (DHCP par exemple).

Traceroutes

Pour finir de convaincre l'aimable assistance, voici des traceroutes dans les deux sens.

de 192.168.1.2 vers 192.168.0.15 :

```
~# traceroute -n 192.168.0.15
traceroute to 192.168.0.15 (192.168.0.15), 30 hops max, 38 byte packets
 1  192.168.1.1  1.150 ms  1.601 ms  1.602 ms
 2  192.168.0.15  1.618 ms  1.863 ms  1.932 ms
```

Comme prévu, 192.168.1.2 va joindre 192.168.0.15 en traversant le seul routeur 192.168.1.1 (cyclope, côté 192.168.1.0).

de 192.168.0.15 vers 192.168.1.2 :

```
~# traceroute -n 192.168.1.2
traceroute to 192.168.1.2 (192.168.1.2), 30 hops max, 38 byte packets
 1  192.168.0.252  1.209 ms  0.855 ms  0.839 ms
```

```
2  192.168.0.16  1.531 ms  0.728 ms  2.235 ms
3  192.168.1.2  1.702 ms  1.195 ms  1.184 ms
```

En revanche, pour atteindre 192.168.1.2, 192.168.0.15 passera d'abord par 192.168.0.252 (gw2) puis par 192.168.0.16 (cyclope, côté 192.168.0.0).

Conclusions

- Les routes aller et retour ne sont pas forcément les mêmes,
- des paquets peuvent trouver leur route dans un sens, mais pas dans l'autre,
- sur des structures plus compliquées, ça peut marcher, mais pas forcément de manière efficace. Avec un peu d'imagination, vous trouverez des cas où un paquet peut beaucoup voyager dans votre inter réseau, pour arriver finalement sur la machine juste à côté. Attention donc, lorsque ça marche du premier coup alors même que ce n'était pas du tout évident, avant de s'émerveiller, il vaut mieux essayer de comprendre pourquoi.