

Un réseau virtuel pour se faire la main

Mathieu DECORE

4 décembre 2000

Table des matières

1	Introduction	3
2	Choix des adresses IP, de réseau et de diffusion	3
2.1	Choix de l'adresse IP	3
2.2	Choix de l'adresse de réseau	5
2.3	Choix de l'adresse de diffusion	5
2.4	Résumé de la configuration de notre réseau	6
3	Attribuer des adresses IP à chaque hôte et mettre à jour la table de routage	8
3.1	Configurer le noyau	9
3.2	Configuration de base de notre réseau	10
3.3	Ajout des autres sous-réseaux	11
4	Tester notre réseau virtuel	12
4.1	Premiers tests	12
4.2	Le fichier <code>/etc/hosts</code>	13
4.3	Le fichier <code>/etc/networks</code>	15
4.4	Débugger une interface	16
5	Configurer un DNS pour notre réseau local	17
6	Configurer la distribution de mél pour notre réseau local	17
7	Configurer un serveur Web par domaine	17

<i>TABLE DES MATIÈRES</i>	2
8 Tester l'ip masquerading	17
9 Conclusion	17
10 Pour aller plus loin	17
10.1 Que faut-il changer dans le cas d'un vrai réseau?	17
10.2 Le Virtual hosting	19

1 Introduction

Le but de cette documentation est de construire sur une machine un réseau virtuel afin de tester les services réseau classiques : DNS, échange de mél, serveurs Web, ... Cela permet de bien comprendre la mise en place de ces services, en ne travaillant que sur une machine non reliée physiquement à d'autres machines. Ainsi, toute personne possédant un ordinateur, sur lequel tourne une version récente de Linux (noyaux 2.2.x), peut s'exercer à configurer les services réseaux classiques. Comme le réseau est amusant, pourquoi s'en priver ?

La section 2 précise la façon dont les paramètres seront choisis. Il y a des parties techniques, qui peuvent être sautées. Le tableau 1 résume les paramètres choisis, et peut suffire pour comprendre la suite de cette documentation.

La section 3 montre comment configurer les interfaces et la section 4 comment tester notre réseau, et les fichiers de base à modifier.

Les sections suivantes montrent comment mettre en place les services réseau classiques : serveur de noms, échange de méls, serveurs web. . .

2 Choix des adresses IP, de réseau et de diffusion

2.1 Choix de l'adresse IP

Comme on est en réseau local, on peut choisir ce que l'on veut comme adresse IP. La seule norme est que cette adresse doit comporter quatre nombres compris entre **0** et **255**, et séparés par un point (“.”). Par exemple, *1.2.3.4* serait reconnu comme une adresse valide, et tout fonctionnerait bien. Seulement voilà, on va faire les choses proprement, et prendre des bonnes habitudes. Notre objectif est de s'entraîner à configurer le plus fidèlement possible à un vrai réseau. Et un vrai réseau peut être connecté (de façon occasionnelle ou permanente) à l'Internet. Que se passerait-il alors si notre réseau local avait comme adresse IP *1.2.3.4*, en ayant une connexion vers l'extérieur ? Il y aurait un risque que des paquets¹ soient envoyés à cette adresse sur l'Internet. Et si une machine a pour adresse officielle *1.2.3.4* sur l'Internet, ces mêmes paquets risquent de lui arriver, ce qui peut le gêner.

1. La diffusion des informations se fait par suite de paquets, de taille bien définie. Le premier paquet indique combien de paquets il va envoyer, et les autres paquets contiennent l'information ainsi que le numéro du paquet. Ainsi, les paquets peuvent être transmis dans un ordre quelconque, emprunter des routes différentes plus ou moins longues. Si un paquet est mal transmis, le destinataire envoie alors un message à l'expéditeur en lui disant quels paquets il n'a pas ou mal reçus, qu'il renvoie aussitôt.

Pour éviter ce problème, il a été convenu que certaines adresses seraient réservées aux réseaux privés. Ainsi, si on a une adresse de réseau privé, en cas de problème, les paquets ne généreront personne.

Suivant le nombre d'hôtes que l'on désire insérer dans notre réseau, il existe plusieurs classes d'adresses.

Ajouter une explication partie hôte / réseau d'une adresse IP.

Classe du réseau	Plage d'adresse du réseau	Nombre de réseaux	Nombre d'hôtes par réseau
Classe A	1.0.0.0 à 127.0.0.0	256 (?)	1.6 million
Classe B	128.0.0.0 à 191.255.0.0	16320	65024
Classe C	192.0.0.0 à 223.255.255.0	2 million	254

Le choix de la classe d'adresse IP dépend donc du nombre d'hôtes que l'on veut insérer. Si on est sûr de ne jamais dépasser **254** postes par sous-réseau (ceci inclut des serveurs, routeurs, imprimantes si elles sont réseau, postes utilisateurs tournant sous Linux ou même Windows, terminaux), on choisira une adresse de réseau **192.168**, comme sous réseaux **192.168.0** à **192.168.255** et comme adresses d'hôtes **192.168.0.0** **192.168.0.1**... **192.168.0.254**. Voir la RFC 1918 [2] pour plus de détails sur l'attribution des adresses IP.

Pour la suite, on décide de prendre une adresse de classe C. Les adresses réservées pour les réseaux privés sont :

Classe du réseau	Plage d'adresse du réseau
Classe A	10.0.0.0 à 10.255.255.255
Classe B	172.16.0.0 à 172.31.0.0
Classe C	192.168.0.0 à 192.168.255.0

On choisit donc comme adresses de réseau **192.168.1.0**, **192.168.100.0** et **192.168.200.0** comme adresses de réseau :

Nom de domaine	Adresse du réseau	Adresse des hôtes
athome.chezmoi	192.168.1.0	192.168.1.X
bienvenue.chezmoi	192.168.100.0	192.168.100.X
cou.chezmoi	192.168.200.0	192.168.200.X

2.2 Choix de l'adresse de réseau

Les deux sections suivantes apportent des précisions sur la façon dont l'adresse de réseau est choisie, ainsi que l'adresse de diffusion. Ces sections ne sont pas indispensables pour la compréhension de la suite de ce document. Ayant choisi des adresses de classe C, on prendra les adresses de réseau et de diffusion calculées à partir des adresses de classe C². C'est juste ce calcul qui est expliqué.

L'adresse IP est divisée en deux parties : la partie réseau, qui indique à quel réseau appartient cette adresse, et la partie hôte, qui indique l'hôte dans ce réseau. La partie réseau est déterminée par le **masque de réseau**. Ce masque détermine sur combien d'octets seront codés le réseau :

- Pour un réseau de classe A, le réseau est codé sur le premier octet de l'adresse IP. Le masque est donc de *255.0.0.0*;
- Pour un réseau de classe B, le réseau est codé sur les deux premiers octets de l'adresse IP. Le masque est donc de *255.255.0.0*;
- Pour un réseau de classe C, le réseau est codé sur les trois premiers octets de l'adresse IP. Le masque est donc de *255.255.255.0*;

On obtient l'adresse de réseau en faisant un ET logique bit à bit de l'adresse IP et du masque de réseau :

	Adresse IP		192	168	1	1
ET	Masque de réseau	ET	255	255	255	0
=	Adresse réseau	=	192	168	1	0

On aurait pu choisir un masque de réseau de classe B avec nos adresses de classe C. Mais on garde les conventions. Donc pour nous, nos **adresses de réseau** sont ***192.168.1.0***, ***192.168.100.0*** et ***192.168.200.0***.

2.3 Choix de l'adresse de diffusion

Il ne reste qu'une adresse à choisir : l'adresse de diffusion. C'est l'adresse écoutée par tous les hôtes du réseau, pour savoir si on leur parle ou pour envoyer des paquets aux autres hôtes du même réseau. Cette adresse doit être la même pour tous les hôtes du réseau. Par

2. Logique, non ? Ben en fait pas tant que ça. Par exemple, je n'ai toujours pas compris pourquoi dans le Guide d'administration réseau sous Linux de Olaf KIRCH, il choisit pour l'exemple une adresse de classe C avec un masque de réseau de classe B. Si quelqu'un peut m'éclairer, toute suggestion est la bienvenue ;-)

convention, on choisit l'adresse de réseau en remplaçant le(s) **0** par **255**. On a donc comme **adresse de diffusion 192.168.1.255**. Techniquement parlant, cela revient à faire un OU logique bit à bit du complément du masque de réseau avec l'adresse de réseau. Ici, cela donne :

NOT	Masque de réseau	NOT	255	255	255	0
=	Complément du masque de réseau	=	0	0	0	255
OU	Adresse réseau	OU	192	168	1	0
=	Adresse de diffusion	=	192	168	1	255

Ce même calcul permet de déterminer les adresses de diffusion des réseaux de quelque classe qu'il soit (seul le masque de réseau change pour un réseau de classe différente).

En fait, il faut juste que tous les hôtes du même réseau aient la même adresse de diffusion. C'est par convention qu'on calcule cette adresse ainsi, mais on pourrait choisir n'importe quelle adresse³.

2.4 Résumé de la configuration de notre réseau

La figure 1 montre le réseau qu'on cherche à simuler.

Le tableau 1 résume tout cela.

Nom du réseau	Adresse du réseau	Nom des hôtes	Adresses des hôtes
athome.chezmoi	192.168.1.0	zecastor, nougat	192.168.1.1, 192.168.1.2
bienvenue.chezmoi	192.168.100.0	soyez, vousetes, tpas	192.168.100.1, 192.168.100.2, 192.168.100.3
cou.chezmoi	192.168.200.0	mais, svp, et	192.168.200.1, 192.168.200.2, 192.168.200.3

TAB. 1 – Résumé du réseau que l'on cherche à simuler.

Le nombre de sous-réseaux et d'hôtes par sous-réseaux est arbitraire, il n'engendre pas plus de difficulté. On a choisi trois sous-réseaux arbitrairement. Il y a en tout huit hôtes, ce qui est le maximum pouvant être supportés par un hub à huit ports (couramment utilisé dans le cadre d'un vrai réseau).

3. A condition d'être sûr que cette adresse ne sera jamais choisie par d'autres réseaux...

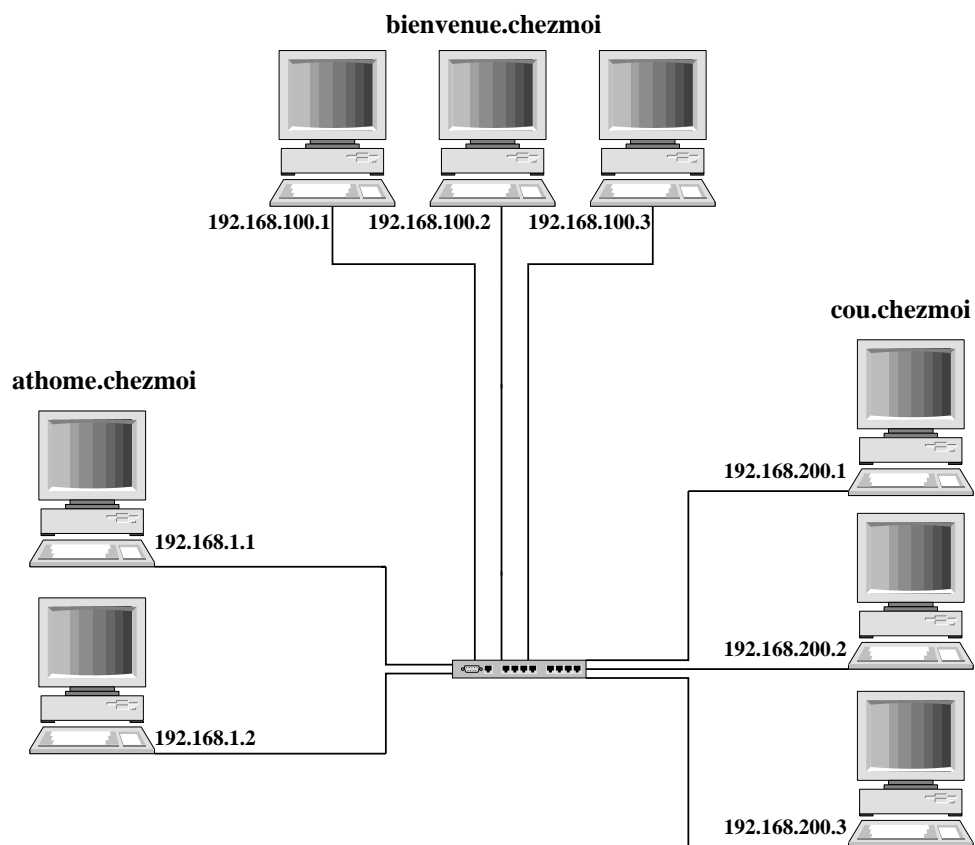


FIG. 1 – Réseau qu'on cherche à simuler

3 Attribuer des adresses IP à chaque hôte et mettre à jour la table de routage

Pour attribuer des adresses IP virtuelles, on va les rajouter à une interface. En effet, une option du noyau appelée “IP Aliasing” permet d’affecter jusqu’à **256** adresses IP à une interface pour Linux 2.0. Les noyaux Linux 2.2 et plus se reconfigurent tout seul au fur et à mesure que des alias sont ajoutés.

Mais quelle interface? Si on a une carte ethernet, on peut activer l’interface *eth0*, si on se connecte avec un modem, on peut activer l’interface *ppp0*. . . Mais nous, on n’a rien de tout ça. Alors? Eh bien il existe une interface, qualifiée de “exotique” par Olaf KIRCH dans le livre **Administration réseau sous Linux** [3]: l’interface *dummy*. Voici à ce sujet un extrait du guide d’administration réseau sous Linux, écrit par ce dernier :

À partir de la version 1.1.16, Linux offre un nouveau type de pilote, le pilote dummy ; que l’on peut traduire par << bidon >>, vous allez voir pourquoi. La question suivante apparaît vers le début de la section concernant les pilotes réseau :

```
Dummy net driver support (CONFIG_DUMMY) [y]
```

Ce pilote ne fait pas grand-chose, mais il est très utile sur des machines isolées ou reliées uniquement en SLIP ou PPP. C’est, en résumé, une interface loopback supplémentaire, qui permet aux machines faisant du SLIP mais n’ayant aucune liaison Ethernet, d’avoir une interface correspondant en permanence à votre adresse IP.

C’est pile poil ce qu’il nous faut. Mais comment affecter plusieurs adresses IP à cette interface? Voici un extrait de l’aide en ligne de `linuxconf` :

Avec l’évolution de l’Internet et la puissance des micro-ordinateurs, il est maintenant possible et utile qu’un ordinateur agisse comme plusieurs, faisant croire qu’il y a plusieurs serveurs en un seul. Un des trucs pour arriver à faire cela est de laisser la machine répondre à plusieurs adresses IP. On les appelle des alias IP.

[...]

Les alias IP sont appelés ainsi parce que ce sont des adresses IP alternatives pour une carte réseau. Quelques logiciels peuvent utiliser différentes configurations en se basant sur l’adresse IP de destination de la requête. Les serveurs web (httpd) et le serveur POP virtuel [...] en font partis.

La **Mini How-To sur la configuration de l’aliasing IP sous Linux** [1] nous renseigne à ce sujet :

3 ATTRIBUER DES ADRESSES IP À CHAQUE HÔTE ET METTRE À JOUR LA TABLE DE ROUTAGE

Premièrement, chargez le module *IP alias* (vous pouvez sauter cette étape si vous avez compilé ce module dans le noyau):

```
/sbin/insmod /lib/modules/$(uname -r)/ipv4/ip_alias.o
```

Deuxièmement, configurez les interfaces *loopback*, *eth0* et tous les numéros IP, en commençant par le numéro IP principal pour l'interface *eth0*:

```
/sbin/ifconfig lo 127.0.0.1
/sbin/ifconfig eth0 up
/sbin/ifconfig eth0 172.16.3.1
/sbin/ifconfig eth0:0 172.16.3.10
/sbin/ifconfig eth0:1 172.16.3.100
```

172.16.3.1 est le numéro IP principal, alors que *.10* et *.100* sont les alias. La magie vient de *eth0:x*, où $x=0,1,3,\dots,n$ pour les différents numéros IP. Le numéro IP principal n'a pas besoin d'alias.

Troisièmement, configurez les routes. D'abord la route pour l'interface *loopback*, puis le réseau, et finalement les numéros IP variés en commençant par celui par défaut (alloué originellement):

```
/sbin/route add -net 127.0.0.0
/sbin/route add -net 172.16.3.0 dev eth0
/sbin/route add -host 172.16.3.1 dev eth0
/sbin/route add -host 172.16.3.10 dev eth0:0
/sbin/route add -host 172.16.3.100 dev eth0:1
/sbin/route add default gw 172.16.3.200
```

C'est tout.

Bon, ben il ne nous reste plus qu'à configurer **d'abord** notre réseau de base ainsi que d'activer l'interface *dummy*. On pourra après rajouter toute les adresses que l'on veut dessus, et mettre à jour la table de routage.

3.1 Configurer le noyau

Avant toute chose, il faut s'assurer qu'on a un noyau contenant tout ce dont on a besoin :

```
General setup --->
```

3 ATTRIBUER DES ADRESSES IP À CHAQUE HÔTE ET METTRE À JOUR LA TABLE DE ROUTE

```
[*] Networking support (CONFIG_NET)
```

```
Networking options --->
```

```
[*] TCP/IP networking (CONFIG_INET)
```

```
[*] IP: aliasing support (CONFIG_IP_ALIAS)
```

```
Network device support --->
```

```
[*] Network device support (CONFIG_NETDEVICES)
```

```
<*> Dummy net driver support (CONFIG_DUMMY)
```

3.2 Configuration de base de notre réseau

La configuration de base de notre réseau se fait de manière classique à l'aide des commandes `ifconfig` et `route`. On configure d'abord l'interface de bouclage, puis l'interface *dummy*:

```
ifconfig lo 127.0.0.1
route add -net 127.0.0.0
ifconfig dummy 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up
route add -net 192.168.1.0
```

On affecte une des trois adresse prise par convention sur l'interface *dummy*, par exemple *192.168.1.1*. Comme on affecte une adresse de classe C, `ifconfig` peut déterminer lui même le masque de réseau (*netmask*) et l'adresse de diffusion (*broadcast*), qui suivent les conventions d'une adresse de classe C. On les a quand même précisés pour montrer comment attribuer un masque de réseau non conventionnel. L'argument *up* est lui aussi optionnel.

On peut vérifier que tout est bien configuré à l'aide des commandes `ifconfig` et `route`:

```
nougat:~# ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            UP LOOPBACK RUNNING  MTU:3924  Metric:1
            RX packets:44 errors:0 dropped:0 overruns:0 frame:0
            TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
            Collisions:0

dummy      Link encap:Ethernet  HWaddr 00:00:00:00:00:00
            inet addr:192.168.1.1  Bcast:192.168.100.255  Mask:255.255.255.0
            UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
```

3 ATTRIBUER DES ADRESSES IP À CHAQUE HÔTE ET METTRE À JOUR LA TABLE DE ROUTAGE

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
Collisions:0
```

```
nougat:~# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.1.0      *                255.255.255.0   U        0      0      0 dummy
nougat:~#
```

Les deux interfaces sont bien activées (*UP RUNNING*), avec comme adresse IP *127.0.0.1* et *192.168.1.1*. Les adresses du type *192.168.1.X* appartiennent au réseau *192.168.1.0*, associé à l'interface *dummy* d'après le résultat de la commande `route`.

3.3 Ajout des autres sous-réseaux

Maintenant que l'interface *dummy* est activée, on peut lui attribuer de nouvelles adresses :

```
ifconfig dummy:1 192.168.100.1
ifconfig dummy:2 192.168.100.2
ifconfig dummy:3 192.168.100.3
ifconfig dummy:4 192.168.200.1
ifconfig dummy:5 192.168.200.2
ifconfig dummy:6 192.168.200.3
```

On peut mettre à jour la table de routage :

```
route add -net 192.168.100.0
route add -net 192.168.200.0
```

On peut vérifier que tout est bien configuré à l'aide des commandes `ifconfig` et `route` :

```
nougat:~# ifconfig dummy:1
dummy:1  Link encap:Ethernet  HWaddr 00:00:00:00:00:00
         inet addr:192.168.100.1  Bcast:192.168.100.255  Mask:255.255.255.0
         UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         Collisions:0
```

```
nougat:~# ifconfig dummy:6
dummy:6  Link encap:Ethernet  HWaddr 00:00:00:00:00:00
         inet addr:192.168.200.3  Bcast:192.168.200.255  Mask:255.255.255.0
         UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         Collisions:0

nougat:~# route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
192.168.100.0    *                255.255.255.0    U        0      0      0 dummy
192.168.1.0      *                255.255.255.0    U        0      0      0 dummy
192.168.200.0    *                255.255.255.0    U        0      0      0 dummy
nougat:~#
```

Tout est maintenant bien configuré. On va pouvoir passer à la phase de tests.

4 Tester notre réseau virtuel

Effectuons les test préliminaires, classiques lorsqu'on vient d'installer un réseau. On cherche d'abord à savoir si on peut atteindre les autres hôtes, si ils peuvent recevoir nos paquets... Tous les outils utilisés appartiennent au package *net-tools* (outils réseau), qui sont normalement installés.

4.1 Premiers tests

Pour tester notre nouveau réseau virtuel, on peut essayer de “ping” les adresses IP (appuyer sur **Ctrl-C** pour arrêter) :

```
nougat:~# ping 192.168.100.1
PING 192.168.100.1 (192.168.100.1): 56 data bytes
64 bytes from 192.168.100.1: icmp_seq=0 ttl=255 time=7.0 ms
64 bytes from 192.168.100.1: icmp_seq=1 ttl=255 time=5.6 ms
64 bytes from 192.168.100.1: icmp_seq=2 ttl=255 time=5.6 ms

--- 192.168.100.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5.6/6.0/7.0 ms
nougat:~#
```

Les autres adresses sont atteignables de la même façon.

On peut aussi tenter un `telnet` sur une adresse IP :

```
nougat:~# telnet 192.168.100.2
Trying 192.168.100.2...
Connected to 192.168.100.2.
Escape character is '^]'.
Debian GNU/Linux 2.1 nougat
```

```
nougat login:
```

Evidemment, on aboutit sur l'hôte **nougat**, puisque l'adresse *192.168.100.2* indique qu'il faut s'adresser à la "vraie" adresse de l'interface *dummy*, c'est à dire *192.168.1.1*.

Le nom **nougat** est le nom d'hôte de la machine, qui a été choisit lors de l'installation de Linux. Eh oui, même si on travaille sur une machine isolée, on lui attribue quand même un nom d'hôte ! Ce nom figure dans le fichier `/etc/hostname` :

```
nougat:~# cat /etc/hostname
nougat
nougat:~#
```

4.2 Le fichier `/etc/hosts`

Bien, maintenant qu'on a un réseau configuré, il faut attribuer des noms d'hôtes à chacune des adresses IP ! Cela se fait dans le fichier `/etc/hosts` :

```
#
# Fichier /etc/hosts - Affecte des noms d'hostes a des adresses IP.
#
# Syntaxe : IP      nom d'hote canonique      alias
#
# Pour bouclage
#

127.0.0.1      localhost

#
# Autres hostes
```

```

#
192.168.1.2    nougat.athome.chezmoi    nougat
192.168.1.1    zecastor.athome.chezmoi    zecastor

#
# Hotes du reseau bienvenue.chezmoi (192.168.100.x)
#
192.168.100.1    soyez.bienvenue.chezmoi    soyez
192.168.100.2    vousetes.bienvenue.chezmoi    vousetes
192.168.100.3    tpas.bienvenue.chezmoi    tpas

#
# Hotes du reseau cou.chezmoi (192.168.200.x)
#
192.168.200.1    mais.cou.chezmoi    mais
192.168.200.2    svp.cou.chezmoi    svp
192.168.200.3    et.cou.chezmoi    et

```

On spécifie l'adresse IP, le nom complet (nom d'hôte.nom de domaine) et éventuellement un alias.

On peut vérifier que ça marche :

```

nougat:~# ping tpas
PING tpas.bienvenue.chezmoi (192.168.100.3): 56 data bytes
64 bytes from 192.168.100.3: icmp_seq=0 ttl=255 time=7.8 ms
64 bytes from 192.168.100.3: icmp_seq=1 ttl=255 time=5.8 ms
64 bytes from 192.168.100.3: icmp_seq=2 ttl=255 time=5.8 ms

--- tpas.bienvenue.chezmoi ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5.8/6.4/7.8 ms
nougat:~# ping tpas.bienvenue.chezmoi
PING tpas.bienvenue.chezmoi (192.168.100.3): 56 data bytes
64 bytes from 192.168.100.3: icmp_seq=0 ttl=255 time=6.0 ms
64 bytes from 192.168.100.3: icmp_seq=1 ttl=255 time=5.8 ms
64 bytes from 192.168.100.3: icmp_seq=2 ttl=255 time=5.8 ms

--- tpas.bienvenue.chezmoi ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 5.8/5.8/6.0 ms

```

```
nougat:~#
```

Si on rajoute les hôtes dans le fichier `/etc/hosts`, on peut aussi essayer un `telnet` sur le nom :

```
nougat:~# telnet tpas.bienvenue.chezmoi
Trying 192.168.100.3...
Connected to tpas.bienvenue.chezmoi.
Escape character is '^]'.
Debian GNU/Linux 2.1 nougat
```

```
nougat login:
```

4.3 Le fichier `/etc/networks`

On peut également remplir le fichier `/etc/networks` pour associer des noms aux adresses de réseaux :

```
#
# Fichier /etc/networks - Affecte des noms de reseau a des adresses de
# sous reseau.
#
# Syntaxe : nom de sous reseau  adresse de sous reseau
#

localnet 127.0.0.0

#
# Le sous reseau local. Toutes les machines ayant comme adresse IP
# 192.168.1.x appartiennent a ce sous reseau.
#

athome.chezmoi 192.168.1.0

#
# Si on veut mettre d'autre sous reseaux, on pourrait faire :
#
#     bienvenue.chezmoi 192.168.2
#
# Ainsi, 192.168.1.1 se termine par athome.chezmoi, et 192.168.2.1 par
# bienvenue.chezmoi.
```

#

```

bienvenue.chezmoi 192.168.100.0
cou.chezmoi 192.168.200.0

```

Cela permet par exemple d'avoir les noms de domaines lors de l'affichage du résultat de la commande `route` :

```

nougat:/# route
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
bienvenue.chezm *                255.255.255.0  U        0      0      0 dummy
athome.chezmoi  *                255.255.255.0  U        0      0      0 dummy
cou.chezmoi     *                255.255.255.0  U        0      0      0 dummy
nougat:/#

```

Si on veut conserver l'affichage des adresses de domaines au lieu des noms, il faut passer l'option `-n` :

```

nougat:/# route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.100.0   0.0.0.0         255.255.255.0  U        0      0      0 dummy
192.168.1.0     0.0.0.0         255.255.255.0  U        0      0      0 dummy
192.168.200.0  0.0.0.0         255.255.255.0  U        0      0      0 dummy
nougat:/#

```

4.4 Débuguer une interface

La commande `netstat` permet d'afficher les statistiques sur les interfaces réseau actives. C'est utile pour voir les paquets perdus, les engorgements :

```

nougat:/# netstat -i
Kernel Interface table
Iface  MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flags
lo     3924  0    382   0     0     0    382   0     0     0 LRU
dummy 1500  0     0     0     0     0     0     0     0     0 BORU
nougat:/#

```

On voit ici que l'interface `lo` a reçu (`RX-OK`) **382** paquets sans erreurs, et en a transmis (`TX-OK`) **382** sans erreurs.

5 Configurer un DNS pour notre réseau local

Bien, maintenant que notre réseau virtuel existe, il faudrait pouvoir tout mettre dans un DNS.

6 Configurer la distribution de mél pour notre réseau local

7 Configurer un serveur Web par domaine

8 Tester l'ip masquerading

9 Conclusion

Finallement notre réseau virtuel a atteint son objectif: on a pu se faire la main avec **une seule machine**, ce qui a l'avantage du gain de place, d'argent, et pas de soucis de cablages, d'installer Linux sur plusieurs machines, compiler les noyaux...

10 Pour aller plus loin

10.1 Que faut-il changer dans le cas d'un vrai réseau?

Bon, tout ça c'est bien joli, ça nous a permis de voir les fichiers de configuration d'un réseau, comment les adapter à son réseau, tester les outils en conditions réelles... Mais maintenant si on a un *vrai* réseau, avec plusieurs machines, des cables et tout et tout, que faut-il changer? Que faut-il rajouter, enlever?

Bien évidemment, la configuration du réseau se fera sans IP Aliasing. De plus, ce n'est pas à l'interface *dummy* qu'on attribuera une adresse IP, mais à l'interface *eth0*, correspondant à la carte Ethernet. Il faudra donc le spécifier lors de la compilation du noyau. Il faut donc identifier précisément son matériel: carte Ethernet ISA ou PCI? De quel type (3COM, NE2000...)? En fonction de ces réponses, il faut activer le bon pilote dans le noyau. La plupart des cartes récentes sont compatibles NE2000. Pour ce type de carte, activer dans

le noyau :

```
Network device support --->
[*] Network device support (CONFIG_NETDEVICES)
Ethernet (10 or 100Mbit) --->
[*] Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET)
[*] EISA, VLB, PCI and on board controllers (CONFIG_NET_EISA)
<M> PCI NE2000 support (CONFIG_NE2K_PCI)
```

Ou, pour une carte 3COM 509 :

```
Network device support --->
[*] Network device support (CONFIG_NETDEVICES)
Ethernet (10 or 100Mbit) --->
[*] Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET)
[*] 3COM cards (CONFIG_NET_VENDOR_3COM)
[*] 3c509/3c579 support (CONFIG_EL3)
```

Dans tous les cas, lire la Ethernet-HOWTO [4].

On peut ensuite voir si la carte a bien été détectée par le noyau lors du démarrage avec la commande `dmesg` :

```
zecastor:~# dmesg | grep eth0
eth0: RealTek RTL-8029 found at 0x6200, IRQ 12, 52:54:00:E8:70:FF.
zecastor:~#
```

Ou, pour une carte 3Com :

```
$ dmesg | grep eth0
eth0: 3c509 at 0x300 tag 1, 10baseT port, address 00 20 af b7 f4 79, IRQ 10.
eth0: Setting Rx mode to 1 addresses.
$
```

Si la sortie est vide, vérifier qu'il y a bien une entrée pour le type de carte. Pour une carte de type NE2000, chercher *ne2k-pci* :

```
zecastor:~# dmesg | grep ne2k
ne2k-pci.c:vpre-1.00e 5/27/99 D. Becker/P. Gortmaker http://cesdis.gsfc.nasa.
```

```
gov/linux/drivers/ne2k-pci.html
ne2k-pci.c: PCI NE2000 clone 'RealTek RTL-8029' at I/O 0x6200, IRQ 12.
zecastor:~#
```

Ou, pour une carte de type 3Com, chercher *3c*:

```
$ dmesg | grep 3c
eth0: 3c509 at 0x300 tag 1, 10baseT port, address 00 20 af b7 f4 79, IRQ 10.
3c509.c:1.16 (2.2) 2/3/98 becker@cesdis.gsfc.nasa.gov.
$
```

Si rien n'apparaît, reprendre la compilation du noyau. Voir la Kernel-HOWTO [5] pour plus de détails.

Ensuite, il faut bien sur brancher toute les machines (brancher les cables RJ45 non croisés sur le hub, et vérifier qu'une diode par poste est bien allumée).

Et c'est tout !

Tout les outils présentés (*ifconfig*, *route*, *netstat*, *ping*...) pourront être utilisés de la même façon. Les fichiers de configuration (*etc/hosts*, */etc/networks*...) auront la même syntaxe.

Le choix des adresses IP restera le même, des adresses réservées aux réseaux privés.

10.2 Le Virtual hosting

Références

- [1] Harish Pillay : **Mini How-To sur la configuration de l'aliasing IP sous Linux**, 13 Janvier 1997.
ftp://ftp.lip6.fr/pub/linux/french/docs/HOWTO/mini/IP-Alias.gz
- [2] Network Working Group : **RFC 1918 - Address Allocation for Private Internets**, Février 1996.
ftp://ftp.lip6.fr/pub/rfc/rfc/rfc1918.txt.gz
- [3] Olaf Kirch, Terry Dawson : **The Linux Network Administrator's Guide, Second Edition**, Mars 2000.
http://linuxdoc.org/LDP/nag2/index.html

Version 1.0 en Français (les sources seules) :

ftp://ftp.lip6.fr/pub/linux/french/books/nag.french.eoit-1.0.tar.gz

[4] Paul Gortmaker : **Linux Ethernet-Howto**, 5 mai 1999.

ftp://ftp.lip6.fr/pub/linux/french/docs/HOWTO/Ethernet-HOWTO.gz

[5] Brian Ward : **Le HOWTO du noyau Linux (Kernel HOWTO)**, 5 juin 1999.

ftp://ftp.lip6.fr/pub/linux/french/docs/HOWTO/Kernel-HOWTO.gz