

Routage sous FreeBSD ou LINUX

Jean-Marc LICHTLE

22 avril 2004

Table des matières

1	Introduction	3
2	Description du matériel mis en oeuvre pour ces essais	3
2.1	jml1	3
2.2	jml2	3
2.3	jml3	3
3	Premier essai, routage entre deux sous-réseaux virtuels	3
3.1	Sous réseaux	4
3.2	Configuration jml2 (LINUX)	4
3.3	Configuration jml1 (BSD pour cet essai)	5
3.4	Activation du routage sous BSD	7
3.5	Et les alias sous LINUX ?	9
4	Routage entre deux réseaux physiques	9
4.1	Résumé de la nouvelle configuration	10
4.2	Configuration de jml2 et jml1	10
4.2.1	Routage vers une branche de réseau	11
4.2.2	Routage vers une adresse IP spécifique	12
4.3	Configuration de jml3	12
4.3.1	Configurer les interfaces réseau	12
4.3.2	Activer le routage	13
5	Routage vers Internet	13
5.1	Résolution des adresses IP	14
5.2	Visibilité depuis l'Internet	14
5.3	Filtrage du trafic	15
5.4	Translation d'adresse	17
6	Suite à donner	17

TABLE DES MATIÈRES

7 LINUX ou FreeBSD ?	18
8 L'auteur	18

1 Introduction

L'objectif est ici de tester les capacités des noyaux LINUX ou BSD à prendre en charge des tâches de routage. Nous allons de fait nous entraîner à utiliser ces deux systèmes d'exploitation pour ce travail. En but final l'idée est de mettre en place un routeur entre un petit réseau privé et l'Internet.

2 Description du matériel mis en oeuvre pour ces essais

Le contexte de cette étude est un réseau Ethernet domestique comme on en trouve certainement de plus en plus. Trois PC sont connectés via un petit hub pour constituer un réseau Ethernet en étoile tout à fait classique. Toutes les cartes employées ici sont des modèles classiques RTL8139 ou Tulip. Ces machines sont nommées jml1, jml2, jml3, dans leur ordre d'achat. Elle sont décrites ci-dessous.

2.1 jml1

Il s'agit d'un vieux Pentium 75 tournant actuellement avec FreeBSD 5.1 en mode texte. Des disquettes de boot permet de lancer un CD KNOPPIX ou un CD Damn Small LINUX en variante à BSD. Ce PC porte une carte Ethernet habituellement configurée à l'adresse 192.168.1.1.

2.2 jml2

Le matériel est ici un peu plus moderne mais accuse tout de même quelques milliers d'heures de vol. Le coeur du système est un P366. La machine peu démarrer en double boot, soit sous LINUX Mandrake 9.1, soit sous un autre système édité du côté de Redmond (USA) mais qui ne sera pas utilisé ici. La carte Ethernet est généralement configurée à l'adresse 192.168.1.2

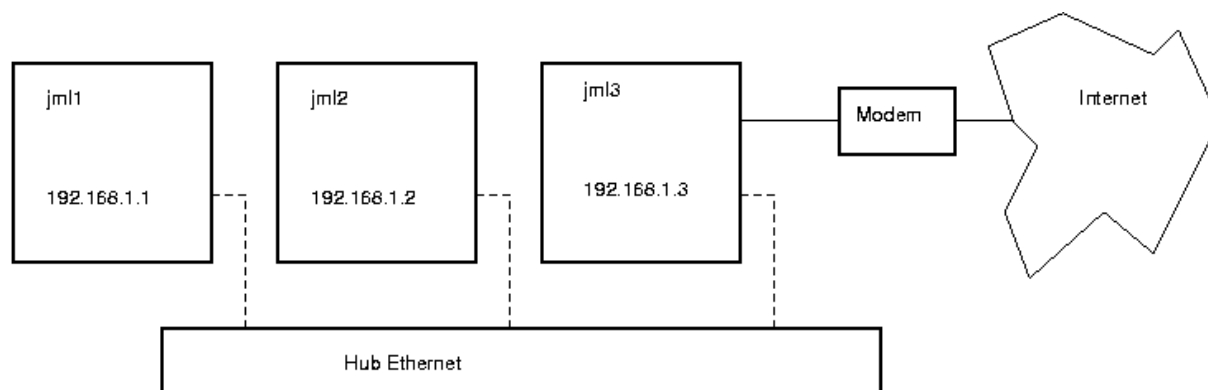
2.3 jml3

La machine la plus puissante, carte Ethernet à l'adresse 192.168.1.3, modem etc.. La combinaison de disques extractibles et de boots multiples permet de faire tourner aussi bien sous LINUX Mandrake 9.2 que Fedora Core 1.B ou FreeBSD 5.2.1. Cette machine ne tourne donc que sous des systèmes libres.

3 Premier essai, routage entre deux sous-réseaux virtuels

L'idée de cet essai m'est venu en étudiant la documentation BSD, plus particulièrement la chapitre relatif au routage. Tous les essais de routage que j'avais pu faire jusque là sous Linux avec d'autres matériels, dans d'autres circonstances n'avaient jamais abouti. J'ai donc décidé que j'allais reprendre tous les tests, cette fois en utilisant la machine jml3 (FreeBSD) pour faire le routage.

FIG. 1 – Topologie normale du réseau testé.



Le schéma 1 montre la configuration normale du réseau testé qui sera employée pour cette première expérience :

3.1 Sous réseaux

Il tombe sous le sens que, si trois PC raccordés tous trois sur le même hub doivent travailler ensemble, leurs adresses doivent appartenir au même sous réseau (ici, un réseau de classe C, la famille d'adresses de 192.168.1.1 à 192.168.1.254). Il suffit simplement, pour un tel réseau, de configurer judicieusement les cartes pour que les communications puissent être établies entre les différentes machines. En clair une configuration correcte des cartes est nécessaire et suffisante pour que les communications simples type ping puissent s'établir. A cet endroit il convient d'insister sur un point capital. La configuration d'une adresse IP sur une carte donnée va avoir deux conséquences :

- Fixer l'adresse de la carte (évidemment).
- Ouvrir automatiquement une "route" vers toutes les adresses appartenant au même sous-réseau que la carte configurée.

Cet dernier aspect, et surtout le fait que l'ouverture de cette route soit automatique, va conduire (expérience personnelle) à des oublis plus tard, lorsque la configuration manuelle des routes deviendra nécessaire.

3.2 Configuration jml2 (LINUX)

On peut aussi, et c'est ce que nous allons faire ici, configurer l'un des PC sur un autre sous réseau, par exemple passer jml2 de l'adresse 192.168.1.2 à l'adresse 192.168.2.2 en tapant :

```
ifconfig eth0 down
```

puis

```
ifconfig eth0 192.168.2.2 netmask 255.255.255.0 up
```

3.3 Configuration jml1 (BSD pour cet essai)

La ligne `— down` n'est en fait pas obligatoire. Le fait de reconfigurer la carte à la nouvelle adresse efface tous les anciens réglages, y compris les routes qui étaient associées à cette carte. Pour compléter le paramétrage il faut ajouter une ligne à la table de routage de jml2. Celle-ci contient en effet pour l'instant :

```
[root@jml2 jml]# route
Table de routage IP du noyau
Destination      Passerelle      Genmask          Indic Metric Ref      Use Iface
192.168.2.0      *               255.255.255.0   U        0      0        0 eth0
127.0.0.0        *               255.0.0.0       U        0      0        0 lo
```

Cette table précise explicitement que tout le trafic à destination du sous réseau 192.168.2.0 (192.168.2.1 à 192.168.2.254 plus les adresses particulières ...0 et ...255) sera traité par eth0. Le noyau ne sait donc pas comment traiter le trafic à destination du sous réseau 192.168.1.0. Toute tentative de ping vers ce sous réseau va donc se solder par une succession de lignes

```
From 192.168.2.2 icmp_seq=3 Destination Host Unreachable
```

Pour compléter la table de routage il faut donc taper, sous compte root :

```
route add default eth0
```

ce qui va ajouter la ligne :

```
default          0.0.0.0          U        0      0        0 eth0
```

à la table de routage. Cet ajout informe le noyau d'envoyer tout ce qui ne va pas vers 192.168.2.0 au travers de eth0.

Une autre syntaxe, plus restrictive, aurait consisté à ne "router" que strictement vers le réseau 192.168.1.0 en tapant :

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.2.1
```

Avec pour conséquence cette fois l'ajout de la ligne suivante à la table de routage :

```
192.168.1.0 192.168.2.1 255.255.255.0 UG 0 0 0 eth0
```

Le lecteur se reportera utilement à la lecture des pages man pour des explications complémentaires.

3.3 Configuration jml1 (BSD pour cet essai)

Le PC jml3 va constituer l'autre sous réseau, il est configuré par défaut à l'adresse 192.168.1.1, adresse qui va rester inchangée. Sous BSD (au moins FreeBSD), la commande `route` n'affiche pas la table de routage comme elle le fait sous LINUX. Cette table s'affiche avec `netstat -nr`, syntaxe paraît-il classique sous Unix, en tout cas applicable à IRIX :

3.3 Configuration jml1 (BSD pour cet essai)

```
bash-2.05b$ netstat -nr
Routing tables
```

```
Internet:
Destination      Gateway          Flags    Refs      Use  Netif  Expire
127.0.0.1        127.0.0.1       UH        0         43   lo0
192.168.1        link#1          UC        1         0    r10
192.168.1.3      00:50:fc:70:11:ac UHLW      0        176   lo0
```

```
Internet6:
Destination      Gateway          Flags    Netif  Expire
::1              ::1             UH        lo0
fe80::%r10/64    link#1         UC        r10
fe80::250:fcff:fe70:11ac%r10 00:50:fc:70:11:ac UHL       lo0
fe80::%lo0/64    fe80::1%lo0    Uc        lo0
fe80::1%lo0      link#3         UHL       lo0
ff01::/32        ::1            U         lo0
ff02::%r10/32    link#1         UC        r10
ff02::%lo0/32    ::1            UC        lo0
```

Seule la première partie de cette table nous intéresse (je n'ai pas encore besoin d'ipv6 pour distinguer les machines de mon réseau privé;-)). Elle ne définit aucune route pour le réseau 192.168.2.0. Un ping dans cette direction donnerait donc une série de lignes assez comparable à ce que nous avons tout à l'heure :

```
ping: sendto: No route to host
```

La commande (compte root)

```
route add default 192.168.2.3
```

complète la table de routage comme suit :

```
bash-2.05b$ netstat -nr
Routing tables
```

```
Internet:
Destination      Gateway          Flags    Refs      Use  Netif  Expire
default          192.168.2.3     UGSc     1         0    r10
127.0.0.1        127.0.0.1       UH        0         43   lo0
192.168.1        link#1          UC        2         0    r10
192.168.1.1      link#1          UHLW     2         0    r10
192.168.1.3      00:50:fc:70:11:ac UHLW     0        192   lo0
```

La ligne default indique que le trafic divers doit transiter via r10, nom de la première (et seule) interface Ethernet vers un routeur qui porte l'adresse 192.168.2.3.

3.4 Activation du routage sous BSD

A ce niveau aussi nous aurions pu être plus restrictifs et plutôt que d’essayer d’ouvrir très largement par défaut, on aurait pu limiter le routage au strict nécessaire dans cet exercice, à savoir 192.168.2.0. Je vous laisse chercher éventuellement le syntaxe nécessaire (très simple). Arrivés à ce stade nous avons donc :

- configuré jml2 sur un sous réseau 192.168.2.0 avec eth0 comme route par défaut,
- configuré jml1 sur un sous réseau 192.168.1.0 avec rl0 comme route par défaut.

Il reste donc à faire que jml3 effectue le routage entre ces deux sous réseaux. En effet, pour l’instant tout essai de ping de jml1 vers jml2 ou inversement abouti à un échec.

3.4 Activation du routage sous BSD

Lors de l’installation de FreeBSD j’ai certainement, machinalement, activé le routage par défaut. A ce stade je n’ai donc plus rien à faire. Mais comment vérifier si ce routage est théoriquement activé ? En fait tout va dépendre de la valeur d’une variable interne au noyau BSD. La consultation de ces variables peut simplement se faire avec la commande

```
sysctl -a
```

laquelle déverse un flot de texte sur l’écran ce qui ne nous avance guère ici. Nous cherchons un truc qui parle de routage, “forwarding” en VO non sous-titrée. La commande

```
sysctl -a | grep forward
```

permet donc d’isoler les lignes qui parlent de routage.

```
bash-2.05b$ sysctl -a | grep forward
net.inet.ip.forwarding: 1
net.inet.ip.fastforwarding: 0
net.inet6.ip6.forwarding: 0
```

On retrouve la valeur 1 affectée par défaut à la variable net.inet.ip.forwarding lors de l’installation ce qui confirme que le noyau est bien configuré pour prendre en charge le routage.

Le PC jml3 est équipé d’une interface unique, en effet nous ne configurons pas la seconde interface Ethernet pour ce premier essai. Cette première interface communique sur le réseau 192.168.1.0. Pour lui donner un accès au réseau 192.168.2.0 il suffit de définir une nouvelle adresse sur cette même carte ce qui est possible avec la commande (compte root)

```
ifconfig xl0 inet 192.168.2.1 netmask 255.255.255.0 alias
```

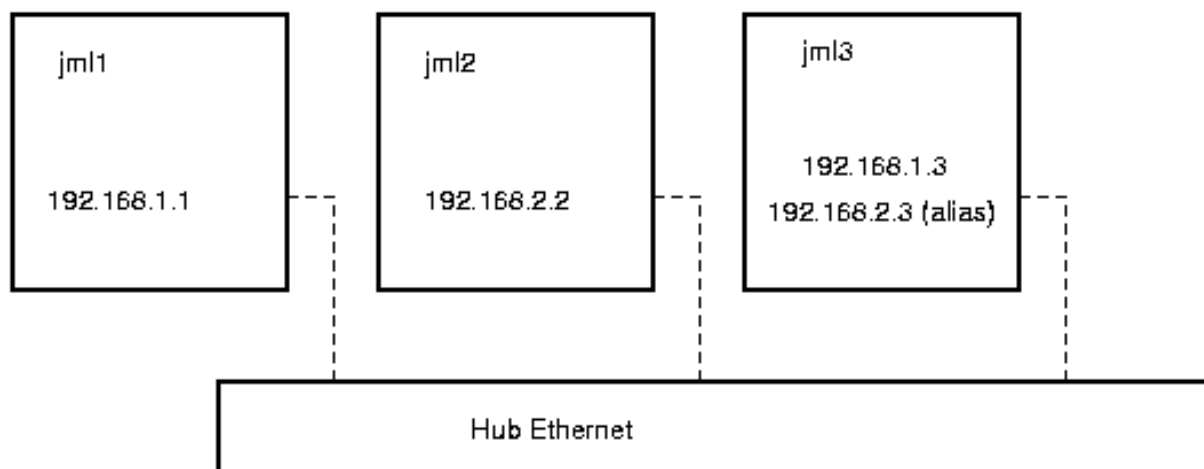
La figure 2 montre le réseau ainsi obtenu, composé de deux réseaux virtuels supportés par un réseau physique commun.

Le résultat d’une commande ifconfig devient :

```
$ ifconfig
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=3<RXCSUM, TXCSUM>
```

3.4 Activation du routage sous BSD

FIG. 2 – Deux réseaux virtuels sur un réseau physique unique.



```
inet 192.168.1.3 netmask 0xffffffff broadcast 192.168.1.255
inet6 fe80::201:2ff:fe5b:5c4c%xl0 prefixlen 64 scopeid 0x1
inet 192.168.2.3 netmask 0xffffffff broadcast 192.168.2.255
ether 00:01:02:fb:5c:4c
media: Ethernet autoselect (100baseTX)
status: active
lp0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
inet 127.0.0.1 netmask 0xff000000
```

On voit bien que la carte Ethernet répond maintenant à deux adresses, l'ancienne 192.168.1.3 et la nouvelle 192.168.2.3, situées chacune sur un des sous réseaux créé. A compter de cet instant les pings de jml2 vers jml1 vont pouvoir circuler joyeusement. Une vérification du rôle de routeur pris par jml3 peut être faite très simplement en tapant (sur jml3 bien-sûr et sous compte root !):

```
# sysctl -w net.inet.ip.forwarding=0
```

ce qui conduit à la réponse suivante :

```
net.inet.ip.forwarding: 1 -> 0
```

Il est facile de vérifier que le routage vient d'être interrompu. faire la modification inverse rétablit le routage.

3.5 Et les alias sous LINUX ?

J'ai cru longtemps que la définition d'alias sous LINUX était impossible. Rien dans la page man de ifconfig si ce n'est une note précisant que les statistiques d'interfaces en alias n'existaient plus depuis le noyau 2.2 (allez comprendre !). C'est en fouillant un jour, par hasard, dans Webmin que j'ai trouvé un lien nommé "add virtual interface" qui m'a conduit vers la définition d'une autre adresse IP pour une carte donnée. Un petit ifconfig -a en mode texte et j'avais l'orthographe que je cherchais.

En fait il est tout aussi facile de définir une adresse alias sous LINUX que sous FreeBSD, seule la syntaxe change très légèrement : (exemple pour jml3 sous Mdk 9.2 partant de la configuration normale que j'utilise quotidiennement)

```
[root@jml3 jml]# ifconfig eth0:0 192.168.2.3
```

Cette ligne suffit à définir une nouvelle adresse pour la carte eth0 ce que confirme la commande ifconfig -a :

```
[root@jml3 jml]# ifconfig -a
eth0      Lien encap:Ethernet  HWaddr 00:50:FC:70:11:AC
          inet adr:192.168.1.3  Bcast:192.168.1.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:390 errors:0 dropped:0 overruns:0 frame:0
          TX packets:972 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:100
          RX bytes:50948 (49.7 Kb)  TX bytes:149908 (146.3 Kb)
          Interruption:5 Adresse de base:0x9000

eth0:0    Lien encap:Ethernet  HWaddr 00:50:FC:70:11:AC
          inet adr:192.168.2.3  Bcast:192.168.2.255  Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interruption:5 Adresse de base:0x9000
```

4 Routage entre deux réseaux physiques

Nous allons employer ici la seconde carte Ethernet et un câble croisé. Nous avons donc maintenant la possibilité de créer deux réseaux bien distincts :

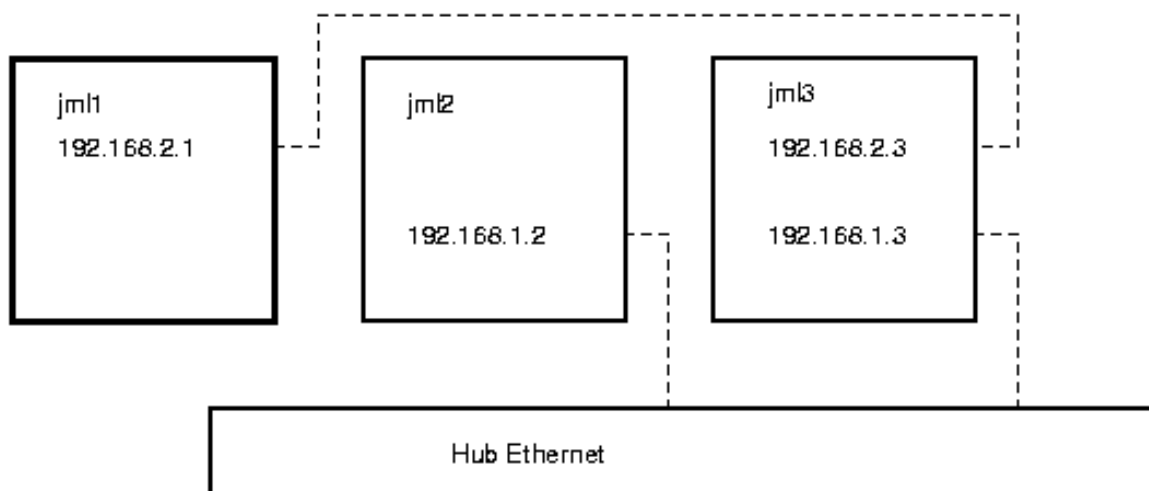
- Un réseau d'adresse 192.168.1.0 constitué des machines jml1, jml3 et du hub.
- Un réseau d'adresse 192.168.2.0 constitué de jml3 dans laquelle est montée la nouvelle carte et jml2 connecté à jml3 par le câble croisé

La figure 3 montre la nouvelle configuration.

Pour les besoins de l'affaire l'adresse de la nouvelle carte sur jml3 est fixée à 192.168.2.3 et l'adresse de jml2 est passée à 192.168.2.2 avec le ifconfig qui va bien.

4.1 Résumé de la nouvelle configuration

FIG. 3 – Deux réseaux physiques.



4.1 Résumé de la nouvelle configuration

Le PC jml3 est maintenant équipé de deux cartes réseaux. Pour varier je décide cette fois de faire tourner jml3 sous KNOPPIX 3.3 ce qui donnera l'occasion de détailler plus bas la configuration des cartes réseau..

Le PC jml2 tourne toujours sous Mandrake 9.1 et est configuré sur le réseau ..2. à l'adresse 192.168.2.2.

Le PC jml1 tourne cette fois sous KNOPPIX 3.2.

4.2 Configuration de jml2 et jml1

Dans les deux cas la configuration revient simplement à paramétrer correctement l'interface eth0 et la table de routage du noyau. Pour bien visualiser ce qui doit être fait je détaille à l'extrême le raisonnement : (Exemple jml2 configuré par défaut à 192.168.1.2)

Pour commencer, et pour faire le vide, faisons "tomber" l'interface etho avec :

```
ifconfig eth0 down
```

Vérifions le résultat sur la table de routage :

```
[root@localhost jml]# route
Table de routage IP du noyau
Destination      Passerelle      Genmask          Indic Metric Ref     Use Iface
127.0.0.0        *               255.0.0.0       U        0      0       0 lo
```

La table de routage est effectivement vide ce qui est normal. Ajoutons maintenant l'interface eth0, par exemple ifconfig eth0 192.168.2.2 si on reste sur jml2, il vient que la table de routage ressemble maintenant à ceci :

4.2 Configuration de jml2 et jml1

```
[root@localhost jml]# ifconfig eth0 192.168.2.2
[root@localhost jml]# route
Table de routage IP du noyau
Destination      Passerelle      Genmask          Indic Metric Ref     Use Iface
192.168.2.0      *               255.255.255.0   U      0      0      0 eth0
127.0.0.0        *               255.0.0.0       U      0      0      0 lo
```

On constate que la table est complétée et que l'interface eth0 va bien faire transiter le trafic à destination des adresses du réseau 192.168.2.0 à savoir ...1 à ...254 (...0 et ...255 sont réservés). Il suffit maintenant de définir une route par défaut qui ferait transiter tout le trafic hors réseau ..2. vers l'interface 192.168.2.3 de jml3 qui sera (tout à l'heure) de routeur entre les réseaux. La syntaxe et le résultat sont consignés ci-dessous :

```
[root@localhost jml]# route add default gw 192.168.2.3
[root@localhost jml]# route
Table de routage IP du noyau
Destination      Passerelle      Genmask          Indic Metric Ref     Use Iface
192.168.2.0      *               255.255.255.0   U      0      0      0 eth0
127.0.0.0        *               255.0.0.0       U      0      0      0 lo
default          192.168.2.3    0.0.0.0         UG     0      0      0 eth0
```

Voilà, c'est aussi simple que cela, la configuration de l'interface définit la route vers le réseau auquel appartient l'interface, il suffit en fait de rajouter "à la main" la route vers le "Gateway" (=routeur) qui conduit à l'autre réseau. Ici aussi, comme plus haut, nous aurions pu être plus restrictifs en n'ouvrant la route via la passerelle 192.168.2.3 qu'aux adresses IP appartenant au réseau 192.168.2.0. Nous aurions même pu être encore plus limitatifs en réduisant cette route strictement pour un accès à l'IP 192.168.2.1. Les syntaxes correspondantes (et conséquences sur la table de routage) sont détaillées ci-dessous.

4.2.1 Routage vers une branche de réseau

```
[root@localhost jml]# ifconfig eth0 down
[root@localhost jml]# ifconfig eth0 192.168.2.2
[root@localhost jml]# route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.2.3
[root@localhost jml]# route
Table de routage IP du noyau
Destination      Passerelle      Genmask          Indic Metric Ref     Use Iface
192.168.2.0      *               255.255.255.0   U      0      0      0 eth0
127.0.0.0        *               255.0.0.0       U      0      0      0 lo
192.168.1.0      192.168.2.3    255.255.255.0   UG     0      0      0 eth0
```

On commence par faire tomber eth0 pour nettoyer les routes précédemment mises en place puis on ouvre une route vers toutes les adresses 192.168.1.0 à 192.168.1.255 via 192.168.2.3 désigné

4.3 Configuration de jml3

comme routeur vers ces adresses. Cette désignation se retrouve dans la table de routage qui liste maintenant 192.168.2.3 comme passerelle. La notion de passerelle se retrouve d'ailleurs dans le G de UG qui signifie Gateway.

4.2.2 Routage vers une adresse IP spécifique

```
[root@localhost jml]# ifconfig eth0 down
[root@localhost jml]# ifconfig eth0 192.168.2.2
[root@localhost jml]# route add -host 192.168.1.1 gw 192.168.2.3
[root@localhost jml]# route
```

Table de routage IP du noyau

Destination	Passerelle	Genmask	Indic	Metric	Ref	Use	Iface
192.168.2.0	*	255.255.255.0	U	0	0	0	eth0
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
192.168.1.1	192.168.2.3	255.255.255.255	UGH	0	0	0	eth0

La syntaxe se distingue du cas précédent par quelques détails. On précise tout d'abord que la route qu'on souhaite définir est une route vers un PC (host) et non plus un réseau (net). La destination étant une adresse IP unique la définition d'un masque de réseau ne présente plus d'intérêt. En écho on retrouve l'indication host sous forme d'un H dans l'indicateur UGH.

4.3 Configuration de jml3

Ayant fait le tour en détail de tous les aspects de la définition du routage sur les PC des deux branches de réseaux il ne reste plus maintenant qu'à faire le nécessaire pour que jml3 effectue les routages souhaités.

4.3.1 Configurer les interfaces réseau

Dans le cas de KNOPPIX les interfaces réseau ne sont pas configurées par défaut, évidemment ! Pensez donc à utiliser ifconfig à cette fin. Attention, dans certains cas il peut se produire que la détection automatique des cartes s'arrête à la première carte trouvée. Il convient donc de diagnostiquer ce comportement puis de forcer la chargement par le noyau du module prenant en charge la carte manquante. Le diagnostic est assez simple à faire :

```
[root@localhost jml]# ifconfig -a
```

En tapant ifconfig -a vous obtenez la liste des toutes les interfaces reconnues, que celles-ci soient configurées ou non. En tapant simplement ifconfig vous risquez de laisser dans l'ombre, selon l'OS employé, des cartes reconnues (modules chargés) mais non configurées (par encore d'adresse IP). Avantage de la syntaxe ifconfig -a : elle est fonctionnelle sur LINUX, FreeBSD et d'autres versions d'UNIX. C'est donc, à mon avis, une bonne habitude à prendre.

4.3.2 Activer le routage

Pour que le routage puisse être mis en route il faut donner le feu vert au noyau. Cette autorisation est matérialisée par la valeur contenue dans le fichier `/proc/sys/net/ipv4/ip_forward` soit fixée à 1. A défaut, si cette valeur est 0 il faut la forcer à 1 en tapant :

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

Il sera très simple, tout à l'heure, lorsque tout fonctionnera correctement de vérifier qu'en forçant à nouveau cette valeur à 0 on bloque le processus de routage. Vous aurez noté ici une certaine ressemblance avec BSD, comme un petit air de famille n'est ce pas ?

Voilà, le routage devrait fonctionner maintenant comme le prouvent les échos aux pings de jml1 vers jml2 et inversement. On peut même, depuis jml1, consulter le serveur web installé sur jml2, ou y faire un telnet. Si quelque chose ne fonctionne pas commencez par vérifier les routes établies aux deux extrémités des branches de réseau. Inutile de vous faire de soucis côté routeur, les routes par défaut conviennent. Ne commettez pas l'erreur (que j'ai commise à de nombreuses reprises) de faire un ping sur la carte de sortie du routeur, celle qui donne l'accès à la branche opposée. Un écho vous parviendra mais il ne devra rien au routage. Vous l'obtiendrez aussi bien avec un LINUX configuré pour ne pas router. C'est bien l'écho d'une autre machine, sur une autre branche qu'il faut aller chercher, d'où la nécessité de disposer effectivement de 3 machines pour faire ce test.

Nota : Je reviendrais sur ce point plus loin (si j'y pense), mais il me semble important d'insister à cette endroit sur le fait que FreeBSD a un comportement totalement différent. Le ping sur la carte de sortie est valide avec FreeBSD et ne donne d'écho que si le routage est actif.

L'intérêt de cet exercice ne semble pas évident à priori. Il trouve toutefois une application très pratique dans la connexion de réseaux de nature différentes, Ethernet et Token-ring par exemple. Supposons que vous soyez, par exemple sur votre lieu de travail, sur un réseau Token-ring mais que vous ayez "récupéré" une ancienne station de CAO qui est équipée d'une interface Ethernet. Pas question d'envisager l'achat d'une carte Token-ring, la station est fonctionnelle mais le coût de cet accessoire dépasserait largement le bénéfice à en attendre. Et puis comment expliquer au service informatique que

vous avez récupéré etc....

Un PC sous LINUX et qui tourne sur un tout autre travail, par exemple serveur de fichier ou d'application pour un département, peut très bien, en plus, faire un peu de routage à temps perdu. Étant sur un réseau privé la nécessité de faire un filtrage très poussé n'est pas impérieuse. Un routage simple sans le moindre filtrage comme celui que nous venons de mettre en place peut donc très bien faire l'affaire.

5 Routage vers Internet

Avançons encore un peu dans l'étude du routage avec le cas de l'Internet. Supposons par exemple que l'on veuille utiliser jml3, qui dispose d'un modem RTC sur le port série, pour ouvrir une porte vers Internet aux autres machines du réseau. Simple me direz vous, on fait comme tout à l'heure mais on route entre l'interface réseau et le port série !

Les choses ne sont malheureusement pas aussi simples. Le plus efficace sera ici de faire un petit

5.1 Résolution des adresses IP

FIG. 4 – Routage vers l'Internet.

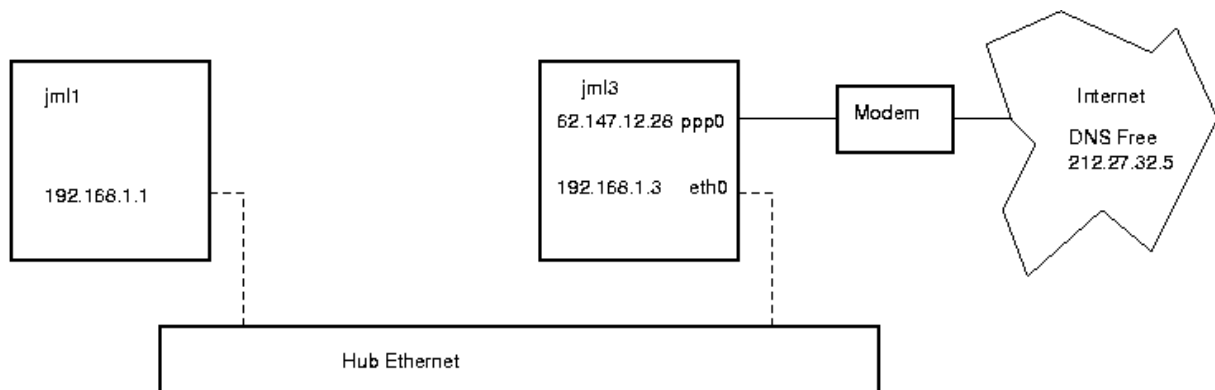


schéma. Supposons que jml1 veuille accéder à Internet via jml3 comme figuré sur le schéma 4. Nous pouvons déjà identifier deux problèmes :

- L'établissement de la connexion conduit à ajouter au fichier `/etc/resolv.conf` de jml3 l'adresse des DNS du provider (Free dans mon cas), modification qui n'est pas faite sur jml1
- La connexion est établie par jml3, c'est donc jml3 qui est vue de l'extérieur, non jml1

5.1 Résolution des adresses IP

Les adresses Internet se présentent généralement sous la forme de chaînes de caractères du genre `http://www.lea-linux.org`. La manipulation des ces chaînes de caractères est facile pour l'homme, pas pour la machine qui utilise des adresse numériques. La conversion doit être faite par un DNS (Domain Name Server). Cette conversion vous est proposée par votre provider Internet lequel vous communique généralement un ou deux adresses de DNS qui font partie de vos paramètres de communication.

A l'établissement de la connexion le fichier `/etc/resolv.conf` est donc modifié pour recevoir les adresses des DNS :

```
[root@localhost etc]# cat resolv.conf
nameserver 127.0.0.1
nameserver 212.27.32.5 #kppp temp entry
nameserver 213.228.0.168 #kppp temp entry
```

Les lignes ajoutées sont repérables aux commentaires qui les balisent.

Ces ajouts n'étant pas faits sur jml1 il convient donc d'ajouter ces lignes de façon permanente (mode root).

5.2 Visibilité depuis l'Internet

Lors de la connexion le provider "prête" une des adresses qu'il détient à l'utilisateur. Cette adresse est facilement repérable lorsque vous êtes connecté en utilisant `ifconfig -a`. Vous devriez

5.3 Filtrage du trafic

voir apparaître une nouvelle interface, ici `ppp0` puisque j'utilise une connexion via modem RTC.

```
ppp0      Lien encap:Protocole Point-à-Point
          inet adr:62.147.12.28  P-t-P:192.168.254.254  Masque:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:3
          RX bytes:156 (156.0 b)  TX bytes:97 (97.0 b)
```

L'adresse 62.147.12.28 ne m'est attribuée que pour la durée de la connexion. A la connexion suivante l'adresse sera très probablement l'une des autres adresses du catalogue du provider. Toutes ces adresses sont connues sur Internet, le PC `jml3`, dès sa connexion, sera donc connu. Plus prosaïquement la réponse à une requête émise par `jml3` reviendra correctement via le provider (Free dans ce cas) vers `jml3`.

Une requête émise par `jml1` va immédiatement poser un problème d'adresse de départ. `jml1` fait en effet partie d'un réseau privé, avec une adresse en 192.168... connue comme étant une adresse privée et qui ne sera donc jamais routée sur Internet. A supposer qu'on change l'adresse de `jml1` pour adopter une adresse routable, comment la réponse à la requête trouvera-t-elle le chemin de retour ? A aucun moment `jml1` ne sera connue des DNS d'Internet, ni ceux du provider, ni les autres !

C'est là qu'intervient la "translation d'adresse". De quoi s'agit-il ? Le concept est finalement très simple. L'émetteur situé sur le réseau privé formule sa demande via le routeur. Le routeur, au passage, modifie le message routé et lui affecte comme adresse de retour son adresse propre. En même temps il note sur son petit calepin qu'une requête vient de sortir et que, en toute bonne logique avant la fermeture des bureaux, il devrait voir passer une réponse adressée au destinataire du message initial à lui-même puisque vu de l'extérieur le demandeur est le routeur. Au passage de cette réponse il suffira de rayer l'adresse de retour et de mettre celle du demandeur initial destinataire final de la réponse. Tout le secret du "masquerading", expression anglophone pour désigner la translation d'adresse est consigné dans ces quelques lignes, du moins le principe de base. La concrétisation est un peu plus difficile puisque qu'il n'est pas possible de faire simplement de la translation d'adresse mais qu'il faut, au préalable, mettre en place un filtrage de trafic, la translation d'adresse étant un module de ce filtrage.

5.3 Filtrage du trafic

Le filtrage du trafic nécessite, du moins pour les noyaux 2.4, l'emploi du module `ip_tables`. La commande `lsmod` permet de lire la liste des modules chargés par le noyau. Un `lsmod | grep ip_tables` permet de vérifier si, par hasard (selon distribution), le module `ip_tables` qui va prendre en charge le filtrage ne serait pas déjà chargé. Dans un tel cas vous devriez obtenir une réponse du genre :

```
[root@localhost jml]# lsmod |grep ip_tables
```

5.3 Filtrage du trafic

```
ip_tables          15072    1 [iptables_filter]
[root@localhost jml]#
```

A défaut de réponse satisfaisante charger le module avec :

```
[root@localhost jml]# insmod ip_tables
Using /lib/modules/2.4.22-10mdk/kernel/net/ipv4/netfilter/ip_tables.o.gz
```

Voilà, le plus dur est fait. Il reste maintenant à fixer les règles de filtrage. Ces règles, pour l'instant vides, peuvent être visualisées très simplement comme suit :

```
[root@localhost jml]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

On constate immédiatement l'existence de 3 chapitres, INPUT, FORWARD et OUTPUT. Le chapitre le plus important ici est FORWARD puisque c'est lui qui va définir le comportement du routage d'une interface à la suivante. Activons donc le "FORWARDING" avec :

```
[root@localhost jml]# iptables -A FORWARD
```

L'effet de cette commande est d'autoriser (A = autorise) tout le trafic entre les deux interfaces. Si on liste à nouveau les règles de filtrage il vient maintenant :

```
[root@localhost jml]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
          all  --  anywhere              anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

En appliquant la même règle sur INPUT et OUTPUT on termine de construire une vraie passoire qui offre une voie royale à tous les pings, virus et agressions en tout genre mais qui route indiscutablement. J'attire l'attention du lecteur sur le fait que nous venons simplement de construire un routeur entre deux réseaux physiques. La construction d'un firewall utilisera cette première étape mais fixera des règles de filtrage bien plus restrictives que celles que nous venons de mettre en place ici.

5.4 Translation d'adresse

Le filtrage étant en place nous pouvons maintenant passer au stade suivant, la translation d'adresse. Pour cela il convient de mettre en place la règle de translation en utilisant la syntaxe suivante :

```
[root@localhost jml]# iptables -t nat -A POSTROUTING //  
-s 192.168.1.0/255.255.255.0 -j MASQUERADE
```

Le symbole // indique une coupe de ligne pour la faire rentrer dans le format de page. Ne pas le reproduire. Cette ligne demande simplement au filtre piloté par iptables de prendre en charge le masquage des adresses provenant (-s = source) du réseau privé. La prise en compte de cette nouvelle règle peut être vérifiée comme suit :

```
[root@localhost jml]# iptables -t nat -L  
Chain PREROUTING (policy ACCEPT)  
target      prot opt source                destination  
  
Chain POSTROUTING (policy ACCEPT)  
target      prot opt source                destination  
MASQUERADE  all  --  192.168.1.0/24        anywhere  
  
Chain OUTPUT (policy ACCEPT)  
target      prot opt source                destination  
[root@localhost jml]#
```

Sommes nous au bout de nos peines ? Je pense n'avoir rien oublié donc je réponds cette fois oui à la question

J'espère que ces quelques pages vous aurons permis de passer du niveau utilisateur confirmé à celui d'administrateur de base. Vous êtes maintenant armé pour lire les documents "étouffe-chrétien" sur le sujet. Bonne lecture et pensez à l'auteur de ce texte à chacun de vos progrès.

6 Suite à donner

Aucun document ne peut prétendre couvrir l'intégralité d'un sujet aussi complexe et varié, sauf à imaginer une véritable encyclopédie. L'étude des pages man reste le passage obligé pour aller plus loin. Dans le cas du routage, et du firewalling qui a simplement été évoqué ici, le lecteur pourra se reporter à nombre de howto. J'ai choisi de ne pas alourdir l'exposé par l'énoncé des fichiers à modifier pour rendre les modifications de configuration permanentes. J'ai la saine habitude de ne jamais automatiser au début pour me contraindre à refaire toute la démarche intellectuelle à chaque mise en route. Ne cherchez donc pas plus loin la raison pour laquelle j'insiste si lourdement sur certains points qui semblent être des détails, comme la définition du routage. Faites une erreur sur ce poste et passez, comme ça m'est arrivé, des heures à chercher ce qui ne va pas.

7 LINUX ou FreeBSD ?

Ne comptez pas sur moi pour me lancer dans une dissertation sur le sujet. Si j'ai terminé l'exposé en m'appuyant essentiellement sur LINUX c'est que ce système, pour des raisons de facilité de mise en oeuvre, de volume de la documentation disponible etc. est le plus simple à traiter. Le début du document est essentiellement étayé par des références à FreeBSD. Ceci est lié au fait que j'ai pensé longtemps que la déclaration d'une seconde adresse IP par alias sur une carte Ethernet n'était pas possible sous LINUX. Et comme c'est sous BSD que j'ai obtenu mon premier ping vainqueur je ne pouvais faire autrement que de mettre en lumière le fonctionnement du routage élémentaire sous ce SE. J'ai tenté de mettre en place le filtrage sous BSD, pour l'instant sans succès, mais je vais continuer à gratter la question.

J'ai à l'occasion cité un système d'exploitation de la famille UNIX bien qu'il s'agisse d'un produit commercial. On ne peut, du moins c'est mon avis, travailler sur des sujets comme LINUX ou BSD sans s'intéresser aux autres UNIX, y compris les produits propriétaires, qu'on peut croiser sur sa route. Dans mon cas j'ai simplement été conduit, comme je l'ai expliqué dans le texte, à réaliser un routage entre le réseau Token Ring de mon entreprise vers une ancienne station SGI équipée d'une interface Ethernet.

8 L'auteur

Jean-marc LICHTLE Ingénieur Arts et Métiers promotion CH 73. J'ai décidé en fin d'année 2003 de me cracher dans les mains et de regarder ce qui pouvait se faire de bien du côté des BSD. Quelques semaines plus tard je me dis que j'ai eu une très bonne idée. Comme le dit fort justement Emmanuel DREYFUS dans son bouquin "Cahiers de l'admin BSD" (Eyrolles), "je n'ai jamais tant appris sur LINUX que depuis que je me suis lancé dans l'étude de BSD".