Linux Networking HOWTO

Joshua Drake

v1.7.0, 29 Décembre 2000

Ceci un document LinuxPorts.Com Document pour le LDP (Linux Documentation Project : Projet de Documentation pour Linux).Il a été soutenu en partie par le Open Source Documentation Fund.

La version actuelle est la 1.7.0, qui est une mise à jour mineure avec quelques corrections grammaticales.

(Traduction et trahison de Jacques.Chion@wanadoo.fr, un grand merci à Jean−Albert Ferrez, Bernard Choppy, Éric Dumas et Jean−Paul Chiron pour leur aide).

NdT : Dans cette version, l'auteur fait appel très souvent à la générosité publique. Trop à mon goût. J'ai laissé volontairement ces appels, car ce n'est pas le but d'un traducteur de dénaturer l'esprit d'un document. À vous de juger.

Table of Contents

1. Comment puis−je apporter mon aide?
- *1.1. Prêter son concours au Net−HOWTO*

2. Historique du document
- *2.1. Retour d'informations*

3. Comment utiliser ce document.
- *3.1. Les conventions utilisées dans ce document*

4. Informations générales concernant le réseau sous Linux.
- *4.1. Informations sur la couche réseau de Linux.*
- *4.2. Où obtenir des informations sur le réseau, non spécifiques de Linux.*

5. Informations générales concernant la configuration réseau
- *5.1. De quoi ai−je besoin pour démarrer ?*
- *5.2. Où mettre les commandes de configuration ?*
- *5.3. Créer vos interfaces réseau*
- *5.4. Configurer une interface réseau. Noyaux 2.0 et 2.2*
- *5.5. Configurer votre solveur de noms*
- *5.6. Configurer votre interface loopback*
- *5.7. Routage*
- *5.8. Configurer vos serveurs réseau et les services.*
- *5.9. Autres fichiers de configuration ayant un rapport avec le réseau*
- *5.10. Sécurité réseau et contrôle d'accès*

6. Informations sur Ethernet
- *6.1. Cartes Ethernet supportées*
- *6.2. Informations générales sur Ethernet*
- *6.3. Utiliser plusieurs cartes Ethernet sur la même machine*

7. Informations relatives à l'IP
- *7.1. Options au niveau du noyau*
- *7.2. EQL − égaliseur de charge à lignes multiples*
- *7.3. Enregistrement IP (IP Accounting) (pour Linux−2.0)*
- *7.4. IP Aliasing*

7.5. IP Pare−feu (Firewall) (pour Linux−2.0)
7.6. Encapsulation IPIP
7.7. IP Masquerade
7.8. IP Transparent Proxy
7.9. IPv6
7.10. Sources de documentation pour IPv6 sous Linux
7.11. IP Mobile
7.12. Multicast
7.13. Mise en forme du trafic − Changer la bande passante allouée
8. DHCP et DHCPD
8.1. Réglage d'un client DHCP pour les utilisateurs de LinuxConf
8.2. Réglage d'un serveur DHCP sous Linux
9. Routage avancé avec Linux−2.2
9.1. Les bases
9.2. Ajouter une route avec les nouveaux outils ip
9.3. Utiliser NAT avec le noyau 2.2
10. Les commandes IP pour les noyaux 2.2 (travail en cours)
10.1. ip
11. Utilisation du matériel courant pour PC
11.1. RNIS
11.2. PLIP pour Linux−2.0
11.3. PPP
11.4. Client SLIP (Antique)
12. Autres technologies réseau
12.1. ARCNet
12.2. Appletalk (AF_APPLETALK)
12.3. ATM
12.4. AX25 (AF_AX25)
12.5. DECNet
12.6. FDDI (Fiber Distributed Data Interface)
12.7. Relais de trames (Frame Relay)
12.8. IPX (AF_IPX)
12.9. NetRom (AF_NETROM)
12.10. Protocole Rose (AF_ROSE)
12.11. Support SAMBA − `NetBEUI', `NetBios', `CIFS'.
12.12. Support STRIP (Starmode Radio IP)
12.13. Token Ring
12.14. X.25
12.15. Carte WaveLan
13. Câbles et câblages
13.1. Câble série NULL Modem
13.2. Câble port parallèle (câble PLIP)
13.3. Câblage Ethernet 10base2 (coaxial fin)
13.4. Câblage Ethernet à paires torsadées
14. Glossaire des termes utilisés dans ce document.
15. Auteurs :
15.1. Actuels
15.2. Passés
16. Copyright.

Chapter 1. Comment puis−je apporter mon aide?

Nous essayons de fournir un ensemble d'informations pour la compréhension de toutes les implantations de réseaux sous Linux. Cependant, cela consomme du temps, et ce document n'est pas une source de revenu. Nous fournissons ces informations en espérant que ce sera utile à la communauté Linux et aux nouveaux convertis à Linux. Nous sommes toujours intéressés par les retours d'information! Nous ajouterons autant que possible tout sujet pertinent dans ce document.

1.1. Prêter son concours au Net−HOWTO

Si vous vouler apporter votre aide, il existe deux voies qui sont d'un très grand secours.

- Achetez un livre OpenBook! Si vous achetez des livres OpenDocs, OpenDocs Publishing rétrocédera une partie de la somme au Fond de Documentation Open Source (Open Source Documentation Fund). Celui−ci aide les auteurs financièrement afin qu'ils continuent à écrire de la documentation pour les projets Open Source.
- *Contribuer financièrement au document*. Avec cette contribution, vous pouvez même demander ce que vous voudriez voir mis à jour, écrit, ou développé dans le document. Pour participer financièrement, contactez s'il vous plaît Command Prompt, Inc. ou Joshua Drake.
- Si vous avez écrit quelque chose sur un sujet auquel vous voudriez apporter votre contribution, envoyez un courrier électronique à poet@linuxports.com

Chapter 2. Historique du document

Le premier document NET−FAQ fut écrit par Matt Welsh et Terry Dawson. Il répondait aux questions fréquemment posées au sujet des réseaux sous Linux, au moment où le LPD (Linux Documentation Project) démarrait tout juste. Il s'agissait alors des toutes premières versions de développement du noyau réseau sous Linux. Le document NET−2−HOWTO, qui succéda au NET−FAQ, fut l'un des premiers documents du LDP HOWTO. Il traitait de ce qui fut appelé « version 2 » (et plus tard « version 3 ») du logiciel réseau du noyau Linux. Ce document prend la suite à son tour et ne traite que de la version 4 du noyau réseau Linux et plus spécialement des versions du noyau 2.x et 2.2.x.

Les versions précédentes de ce document étaient devenues plutôt énormes en raison du grand nombre de sujets abordés. Pour résoudre ce problème, un certain nombre de documents HOWTO ont été créés et traitent de sujets spécifiques. Ce document fait référence à ceux qui sont pertinents et aborde les sujets qui ne sont pas encore couverts par d'autres documents.

2.1. Retour d'informations

Nous apprécions toujours les retours d'informations. Contactez−nous s'il vous plait à: poet@linuxports.com.

Si vous trouvez des erreurs ou bien si vous désirez que l'on ajoute quelque chose, contactez nous.

Ceci vous a intéressé ? Pourquoi ne pas donner 2,50 dollars?

Chapter 3. Comment utiliser ce document.

Ce document est organisé de manière à être lu progressivement. Les premières sections traitent d'informations sur le matériel et peuvent être sautées si cela ne vous intéresse pas. Puis vous trouverez une discussion générale sur les réseaux, et vous devez être certains de l'avoir assimilée avant de poursuivre vers les paragraphes plus spécifiques. Le reste traite l'aspect plus technique, et est regroupé en trois parties principales

: informations sur Ethernet et IP, les technologies concernant le matériel PC le plus courant, et les technologies moins répandues.

La démarche que je suggère pour parcourir ce document est donc la suivante :

Lire les sections générales
: Ces paragraphes s'appliquent à chaque, ou presque, technologie décrite plus loin et il est donc important que vous les ayez comprises. D'autre part, j'espère que de nombreux de lecteurs connaissent déjà le sujet.

Réfléchissez à votre réseau
: Vous devez savoir comment votre réseau est (ou sera) conçu et quels matériels et types de technologies vous utiliserez.

Lisez la section ``Ethernet et IP'' si vous êtes connectés en direct sur un réseau local ou à l'Internet
: Cette section traite de la configuration de base d'Ethernet et des différentes possibilités offertes par Linux, et qui concernent le réseau, telles que le pare−feu, le routage avancé, etc..

Lisez après si vous êtes intéressés par les réseaux locaux à bas coût ou les connexions par téléphone
: Cette section parle de PLIP, PPP, SLIP, et RNIS, les technologies utilisées habituellement sur les stations personnelles.

Lisez la section concernant la technologie qui correspond plus particulièrement à vos besoins.
: Si vos besoins ne concernent pas IP et/ou un matériel courant, vous trouverez à la fin des détails sur les protocoles non−IP et les matériels de communication particuliers.

Configurez votre réseau
: Si vous allez réellement essayer de configurer votre réseau, prenez soigneusement note de tout problème éventuel.

Cherchez de l'aide si nécessaire
: Si vous rencontrez des problèmes qui ne sont pas traités dans ce document, reportez−vous au paragraphe donnant les endroits où l'on peut en obtenir ou bien envoyer des reports de bogues.

Amusez−vous!
: Le réseau est amusant, profitez−en.

3.1. Les conventions utilisées dans ce document

Il n'y a pas de convention spéciale utilisée dans ce document, mais vous devez faire attention à la façon dont les commandes sont spécifiées. En consultant la documentation habituelle d'Unix, toute commande qui doit être tapée est précédée d'une invite du shell. Ce document utilise "*user%*" comme invite pour les commandes ne nécessitant pas de privilèges de superutilisateur, et "*root#*" pour les commandes que l'on doit exécuter comme super−utilisateur (root). Je préfère utiliser "*root#*" à la place du classique "*#*" pour éviter toute confusion avec les extraits de scripts shell, ou le signe dièse est utilisé pour définir les lignes de commentaires.

Lorsque les « Options de Compilation du noyau » sont mentionnées, elles le sont avec le format utilisé par *menuconfig*. Elles devraient donc être compréhensibles même si vous (comme moi) n'êtes pas familiers avec *menuconfig*. Si vous avez un doute sur l'utilisation de certaines options, faites tourner le programme une fois. Cela ne peut que vous aider.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 4. Informations générales concernant le réseau sous Linux.

4.1. Informations sur la couche réseau de Linux.

Il existe un grand nombre d'endroits où l'on peut trouver de bonnes informations sur le réseau Linux.

Il y a un tas de spécialistes disponibles. On peut en trouver une liste sur http://www.linuxports.com/.

Alan Cox, l'actuel mainteneur du code réseau Linux entretient une page web contenant les points principaux du réseau actuel, et ses nouveaux développements, à l'adresse : www.uk.linux.org.

Il existe un groupe de discussion dédié au réseau et, en ce qui le concerne dans la hiérarchie Linux, c'est : comp.os.linux.networking

Il existe une liste de diffusion à laquelle vous pouvez vous inscrire, et où vous pourrez poser des questions en relation avec le réseau Linux. Pour souscrire vous devez envoyer un message par courrier électronique :

```
To: majordomo@vger.rutgers.edu
Subject: (rien du tout)
Message:

subscribe linux-net
```

Souvenez−vous lorsque vous faites part d'un problème d'y inclure le plus possible de détails nécessaires. Plus spécialement indiquez les versions des logiciels que vous utilisez, en particulier la version du noyau, les versions des outils tels que *pppd* ou *dip*, et la nature exacte des problèmes que vous rencontrez. Cela veut dire prendre note de la syntaxe exacte des messages d'erreurs que vous recevez, et les commandes que vous avez exécutées.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

4.2. Où obtenir des informations sur le réseau, non spécifiques de Linux.

Si vous désirez des informations générales de base sur tcp/ip, alors je vous recommande de regarder les documents suivants :

introduction à TCP/IP
: ce document se trouve à la fois sur en version texte et en version postscript.

administration TCP/IP
: ce document se trouve à la fois sur en version texte et en version postscript.

Si vous recherchez des informations plus détaillées je vous recommande chaudement :

" *Inter Networking with TCP/IP, Volume 1 : Principes, Protocoles et Architectures*, par Douglas E. Comer,ISBN 0−13−227836−7, Prentice Hall publications, 3ème édition, 1995."

Si vous voulez apprendre comment écrire des applications réseau dans un environnement compatible Unix, je vous recommande également chaudement :

" *Unix Network Programming* par W. Richard Stevens ISBN 0−13−949876−1, Prentice Hall publications, 1990."

Une deuxième édition de ce livre va apparaitre sur les rayons : le nouveau livre comporte 3 volumes : voyez le site de Prentice Hall pour en savoir plus.

Vous pouvez essayer aussi le groupe de discussions : comp.protocols.tcp−ip.

Une importante source d'informations techniques concernant l'Internet et la suite des protocoles TCP/IP sont les RFC. RFC est l'acronyme de `Request For Comment' et c'est le moyen habituel de soumettre et de s'informer des normes de protocoles Internet. Il y a beauccoup d'endroits où sont stockées ces RFC. Un grand nombre sont des sites ftp, d'autres fournissent des accès WWW avec un moteur de recherche qui fouille dans les bases de données RFC avec des mots−clés particuliers.

Une source possible de RFC est : la base de données RFC de Nexor.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 5. Informations générales concernant la configuration réseau

Vous devez connaître et bien comprendre les paragraphes suivants avant d'essayer de configurer votre réseau. Ce sont des principes de base qui s'appliquent, indépendamment de la nature du réseau que vous voulez mettre en place.

5.1. De quoi ai−je besoin pour démarrer ?

Avant de commencer à construire ou configurer votre réseau, vous aurez besoin de certaines choses. Les plus importantes sont :

5.1.1. Sources du noyau récents (Optionnel).

À noter:

La majorité des distributions actuelles sont livrées avec l'option réseau activée, de sorte que vous n'avez pas besoin de recompiler le noyau. Si vous utilisez du matériel bien connu, tout ira bien. Par exemple: cartes 3COM, cartes NE2000 ou cartes Intel. Cependant si vous devez recompiler le noyau, voyez les informations qui suivent.

Si le noyau que vous utilisez actuellement ne gère pas les types de réseau ou les cartes que vous voulez utiliser, vous aurez besoin des sources du noyau pour pouvoir le recompiler avec les options adéquates.

Pour les utilisateurs des principales distributions comme RedHat, Caldera, Debian ou Suse, ce n'est plus vrai. Tant que vous restez avec un matériel de grande diffusion, il n'est pas nécessaire de recompiler le noyau, à moins que vous n'ayez une exigence très spécifique.

Vous pouvez toujours obtenir les sources du dernier noyau sur : ftp.cdrom.com. Ce n'est pas le site officiel mais ils ont BEAUCOUP de bande passante et BEAUCOUP d'utilisateurs peuvent se connecter en même temps. Le site officiel est kernel.org, mais dans la mesure du possible, utilisez s'il vous plaît celui que je viens de donner. Souvenez−vous que ftp.kernel.org est particulièrement surchargé. Utilisez un miroir. (NdT : et bien sûr ftp.lip6.fr) .

Normalement les sources du noyau doivent être désarchivées dans le répertoire */usr/src/linux*. Pour savoir comment appliquer les patches et compiler le noyau, lisez le Kernel−HOWTO. Pour savoir comment configurer les modules du noyau, lisez le ``Modules−mini−HOWTO''. Enfin, le fichier *README* qui se trouve dans les sources du noyau ainsi que le répertoire *Documentation* donnent de nombreux renseignements au lecteur courageux.

Sauf indication contraire, je vous recommande de vous en tenir à une version stable du noyau (celle avec un chiffre pair en seconde place dans le numéro de version). Les versions de développement (avec un chiffre impair en seconde place dans le numéro de version) peuvent avoir une structure ou autre chose qui peut poser problème avec les logiciels de votre système. Si vous n'êtes pas certain de résoudre ce type de problèmes, avec en plus ceux qui existeraient sur d'autres logiciels, n'utilisez pas de noyau en développement.

5.1.2. Adresses IP : une explication.

Les adresses de protocole Internet (IP) sont composées de quatre octets. La convention d'écriture est appelée `notation décimale pointée'. Sous cette forme chaque octet est converti en un nombre décimal (0−255), en omettant les zéros de tête (à moins que ce nombre ne soit lui−même un zéro) et chaque octet est séparé par le caractère `.'. Par convention, chaque interface d'un hôte ou routeur possède une adresse IP. Il se peut que la même adresse IP soit utilisée sur différentes interfaces d'une même machine, mais, en général, chaque interface possède sa propre adresse.

Les réseaux IP (Protocole Internet) sont des séquences contiguës d'adresses IP. Toutes les adresses d'un même réseau ont des chiffres en commun. La partie d'adresse commune à toutes les adresses d'un réseau s'appelle la `partie réseau' de l'adresse. Les chiffres restants s'appellent `partie hôte'. Le nombre de bits qui sont partagés par toutes les adresses d'un même réseau est appelé masque de réseau (netmask) et c'est le rôle du masque de réseau de déterminer quelles adresses appartiennent à `son' réseau et celles qui ne sont pas concernées. Par exemple :

```
_______________________________________      _______________
Adresse hôte (host address)                  192.168.110.23
Masque de réseau (network mask)              255.255.255.0
Partie réseau (network portion)              192.168.110.
Partie hôte (host portion)                              .23
_______________________________________      ________________
Adresse réseau (network address)             192.168.110.0
Adresse de diffusion (broadcast address)     192.168.110.255
_______________________________________      ________________
```

Toute adresse qui est un `AND bit à bit' avec son masque de réseau révèlera l'adresse du réseau auquel elle appartient. L'adresse du réseau est par conséquent l'adresse de plus petit nombre dans l'ensemble des adresses et a toujours la partie hôte codée avec des zéros.

L'adresse de diffusion est une adresse spéciale que chaque hôte du réseau écoute en même temps que son adresse personnelle. Cette adresse est celle à laquelle les datagrammes sont envoyés si tous les hôtes du réseau sont en mesure de les recevoir. Certains types de données telles que les informations de routage et les messages d'alerte sont transmis vers l'adresse de diffusion de telle sorte que tous les hôtes du réseau peuvent les recevoir en même temps. Il y a deux standards utilisés de manière courante pour définir ce que doit être l'adresse de diffusion. Le plus largement utilisé est de prendre l'adresse la plus haute possible du réseau comme adresse de diffusion. Dans l'exemple ci−dessus ce serait *192.168.110.255*. Pour d'autres raisons, certains sites ont adopté la convention d'utiliser l'adresse de réseau comme adresse de diffusion. En pratique cela n'a pas beaucoup d'importance, mais vous devez être sûrs que tous les hôtes du réseau sont configurés avec la même adresse de diffusion.

Pour des raisons d'administration, il y a quelque temps, lors du développement du protocole IP, des ensembles d'adresses ont été organisés en réseaux et ces réseaux ont été regroupés en ce que l'on a appellé classes. Ces classes donnent un certain nombre de réseaux de tailles standards auxquels on peut assigner des adresses. Ces classes sont :

```


_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
|Classe de |Masque de     | Adresses de réseau            |
```

```
| réseau   |  réseau       |                              |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|    A     | 255.0.0.0     | 0.0.0.0     - 127.255.255.255 |
|    B     | 255.255.0.0   | 128.0.0.0   - 191.255.255.255 |
|    C     | 255.255.255.0 | 192.0.0.0   - 223.255.255.255 |
|Multicast| 240.0.0.0     | 224.0.0.0   - 239.255.255.255 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Le type d'adresse que vous devez utiliser dépend de ce que vous voulez faire exactement. Vous pouvez utiliser une combinaison des actions suivantes pour obtenir l'ensemble des adresses dont vous aurez besoin :

Installer une machine Linux sur un réseau IP existant

Vous devez contacter un des administrateurs du réseau et lui demander les informations suivantes :

◊ Adresse hôte;
◊ Adresse réseau;
◊ Adresse de diffusion;
◊ Masque de réseau;
◊ Adresse de routage; (appelée passerelle sous Windows)
◊ Adresse du serveur de noms de domaine (DNS).

Vous configurerez alors votre réseau Linux à l'aide de ces données. Vous ne pouvez pas les inventer vous−même et espérer que votre configuration fonctionne.

Construire un réseau tout neuf non connecté à l'Internet

Si vous construisez un réseau privé et que vous n'ayez pas l'intention de vous connecter à l'Internet, vous pouvez alors choisir n'importe quelle adresse. Cependant, pour des raisons de sécurité et de fiabilité, il y a quelques adresses de réseau IP réservées à cet usage. Elles sont spécifiées dans la RFC 1597 et sont les suivantes :

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|          ALLOCATIONS POUR RÉSEAUX PRIVÉS                    |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| Classe   | Masque de     | Adresses de réseau              |
| réseau   |  réseau       |                                 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|    A     | 255.0.0.0     | 10.0.0.0    - 10.255.255.255   |
|    B     | 255.255.0.0   | 172.16.0.0  - 172.31.255.255   |
|    C     | 255.255.255.0 | 192.168.0.0 - 192.168.255.255  |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Vous devez d'abord décider de la dimension de votre réseau et choisir ensuite les adresses dont vous avez besoin.

5.2. Où mettre les commandes de configuration ?

Il y a plusieurs possibilités de procédures de démarrage d'un système Linux. Après le démarrage du noyau, celui−ci exécute toujours un programme appelé `*init*'. Ce programme lit le fichier de configuration appelé */etc/inittab* et commence le processus de démarrage. Il y a quelques variantes de *init*, bien que maintenant tout le monde se dirige vers la variante System V (cinq), développée par Miguel van Smoorenburg.

Bien que que le programme *init* soit toujours le même, les réglages du processus de démarrage se font différemment suivant le type de distribution.

Habituellement le fichier */etc/inittab* contient une entrée telle que :

```
si::sysinit:/etc/init.d/boot
```

Cette ligne spécifie le nom du fichier script qui prend en charge réellement la séquence de démarrage. Ce fichier est en quelque sorte équivalent au fichier MS−DOS *AUTOEXEC.BAT*.

Il y a aussi d'autres scripts appelés par le script de démarrage, et souvent le réseau est configuré dans l'un de ceux−ci.

Le tableau suivant peut être utilisé comme guide suivant le système que vous avez :

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Distrib. |Interface Config/Routage           | Initialisation serveur
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Debian   | /etc/init.d/network               | /etc/rc2.d/*
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Slackware| /etc/rc.d/rc.inet1                | /etc/rc.d/rc.inet2
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
RedHat   | /etc/rc.d/init.d/network          | /etc/rc.d/rc3.d/*
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Notez que les distributions Debian et RedHat utilisent tout un répertoire pour les scripts qui mettent en route les services du système (et habituellement l'information ne se situe pas dans ces fichiers, par exemple les systèmes RedHat stockent l'ensemble de la configuration du système sous */etc/sysconfig*, où elle est récupérée par les scripts de démarrage). Si vous voulez saisir les détails du processus de démarrage, je vous conseille de vérifier */etc/inittab* ainsi que la documentation accompagnant *init*. Linux Journal va également publier un article sur l'initialisation des systèmes, et nous pointerons sur lui dès qu'il sera disponible sur le réseau.

La plupart des distributions récentes incluent un programme qui permet de configurer de nombreux types d'interfaces réseau. Si vous en possédez une, regardez si ce programme vous convient au lieu de tenter une configuration manuelle.

```
- - - - - - - - - - - - - - - - - - - - - - - - -
Distrib    | Programme de configuration réseau
- - - - - - - - - - - - - - - - - - - - - - - - -
RedHat     | /sbin/netcfg
Slackware  | /sbin/netconfig
- - - - - - - - - - - - - - - - - - - - - - - - -
```

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

5.3. Créer vos interfaces réseau

Sur un grand nombre de systèmes Unix, les périphériques réseau apparaissent dans le répertoire */dev* . Il n'en est pas de même avec Linux. Les périphériques réseau sont créés dynamiquement par les logiciels et ne nécessitent donc pas de fichiers de périphériques.

Dans la majorité des cas, le périphérique réseau est automatiquement créé par le gestionnaire de périphérique lors de son initialisation par le noyau. Par exemple le pilote Ethernet crée les interfaces *eth[0..n]* une par une lorsqu'il détecte votre matériel Ethernet. La première carte Ethernet trouvée devient *eth0*, la deuxième *eth1*, etc.

Cependant, dans certains cas, notamment avec *SLIP* et *PPP*, les périphériques réseau sont créés par un programme utilisateur. Le même mécanisme séquentiel s'applique sur les périphériques, mais ce n'est pas au moment du démarrage du système. La raison en est que, à l'inverse des dispositifs Ethernet, le nombre de périphériques *slip* ou *ppp* actifs peut varier dans le temps. Nous y reviendrons plus tard.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

5.4. Configurer une interface réseau. Noyaux 2.0 et 2.2

Lorsque vous avez tous les programmes requis, votre adresse et les informations réseau, vous pouvez alors configurer vos interfaces. Lorsque nous parlons de la configuration d'interface, nous faisons allusion au processus d'assignation des adresses du périphérique réseau, et au processus de réglage des paramètres configurables. Le programme le plus utilisé pour ce faire est la commande *ifconfig* (interface configure).

Typiquement vous utilisez une commande comme ci−dessous :

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

Dans ce cas je configure l'interface Ethernet `*eth0*' avec l'adresse IP `*192.168.0.1*' et un masque de réseau `*255.255.255.0*'. Le `*up*' qui termine la commande enjoint à l'interface de devenir active, mais il peut être omis, étant par défaut. Pour clore une interface, vous faites juste ``*ifconfig eth0 down*''.

Le noyau suppose certaines valeurs par défaut lorsque l'on configure les interfaces. Par exemple, vous pouvez indiquer une adresse de réseau et une adresse de diffusion, mais si vous ne le faites pas comme nous venons de le faire dans l'exemple ci−dessus, alors le noyau fera certaines hypothèses fondées sur le masque de réseau que vous avez fourni, et si vous ne l'avez pas donnée, sur la classe de l'adresse IP configurée. Dans mon exemple, le noyau considérera que c'est un réseau de classe C et configurera une adresse réseau de `*192.168.0.0*' et une adresse de diffusion de `*192.168.0.255*'.

Il y a de nombreuses autres options pour la commande *ifconfig* . Les plus importantes sont :

up
: active une interface (est fait par défaut).

down
: désactive une interface.

[−]arp
: active ou désactive le protocole de résolution d'adresses sur cette interface.

[−]allmulti
: active ou désactive la réception de tous les paquets multicast matériel (Ndt : Les adresses multicast sont un genre d'adresses de diffusion limitées à un groupe de machine qui n'ont pas nécessairement besoin de se trouver sur le même sous−réseau). Le multicast matériel permet à des groupes d'hôtes de recevoir des paquets adressés vers des destinations spéciales. Ce peut être important si vous utilisez des applications comme la vidéoconférence, mais la plupart du temps on ne l'utilise pas.

mtu N
: ce paramètre permet de régler le *MTU* (Maximum Transfert Unit) sur le périphérique.

netmask <addr>
: ce paramètre permet de fixer le masque de réseau.

irq <addr>
: ce paramètre ne fonctionne qu'avec certains types de matériels, mais vous permet d'en fixer l'IRQ.

[−]broadcast [addr]
: permet d'activer ou de désactiver l'acceptation de datagrammes destinés à l'adresse de diffusion.

[−]pointopoint [addr]
: permet de fixer l'adresse de la machine à l'extrémité d'un lien point−à−point comme pour *slip* ou *ppp*.

hw <type <addr>
: permet de fixer l'adresse matérielle de certains périphériques réseau. Ce n'est pas souvent utilisé pour Ethernet, mais utile pour d'autres types de réseau tels que AX.25.

Avec les versions 2.2 du noyau, il y a un certain nombre d'options que nous n'avons pas énumérées ci−dessus. Parmi les plus intéressantes, citons le tunneling et les options IPV6. Voici les paramètres ifconfig pour les noyaux 2.2.

interface
: Le nom de l'interface. C'est habituellement le nom d'un gestionnaire de périphérique suivi par un numéro d'unité, par exemple eth0 pour la première interface Ethernet.

up
: Ceci provoque l'activation de l'interface. C'est implicitement spécifié si un adresse est affectée à l'interface.

down
: Ceci provoque la désactivation de l'interface.

[−]arp
: Active ou désactive l'utilisation du protocole ARP sur l'interface considérée.

[−]promisc
: Active ou désactive le mode «promiscuous» sur l'interface. S'il est choisi, tous les paquets du réseau seront reçus par l'interface.

[−]allmulti
: Active ou désactive le mode «all−multicast». S'il est choisi tous les paquets multicast du réseau seront reçus par l'interface.

metric N
: Ce paramètre positionne le paramètre «metric» de l'interface.

mtu N
: Ce paramètre positionne le Maximum Transfer Unit (MTU) d'une interface.

dstaddr addri
: Positionne l'adresse IP distante d'un lien point−à−point (tel que PPP). Ce mot−clé est maintenant obsolète; utilisez à la place le mot−clé pointopoint.

netmask addr
: Positionne le masque de réseau IP de l'interface. Donne les valeurs par défaut pour les classes habituelles de masque réseau A, B ou C (provenant de l'adresse IP de l'interface), mais on peut donner n'importe quelle valeur.

add addr prefixlen
: Ajoute un adresse IPv6 à l'interface.

del addr prefixlen
: Enlève une adresse IPv6 de l'interface.

tunnel aa.bb.cc.dd
: Crée un nouveau périphérique SIT (IPv6−in−IPv4), tunnelling vers une destination donnée.

irq addr
: Positionne l'interruption utilisée par ce périphérique. Tous les périphériques ne sont pas capables de changer d'IRQ de manière dynamique.

io_addr addr
: Positionne l'adresse d'entrée−sortie du périphérique.

mem_start addr
: Positionne l'adresse de début de la mémoire partagée utilisée par le périphérique. Seuls quelques périphériques en ont besoin.

media type
: Positionne le port physique ou bien le type de matériel qui doit être utilisé par le périphérique. Tous les périphériques ne peuvent pas changer ce réglage, et ceux qui peuvent le faire diffèrent quant aux valeurs qui peuvent leur être assignées. Les valeurs typiques pour sont 10base2 (thin Ethernet), AUI (transceiver externe) et autres. La valeur spéciale auto peut être utilisée pour dire au gestionnaire de périphérique de détecter automatiquement le périphérique. Encore une fois tous les gestionnaires de périphérique ne peuvent faire ceci.

[−]broadcast [addr]
: Si une adresse est donnée en argument, positionne l'adresse de protocole de diffusion de l'interface. Autrement, positionne (ou efface) le drapeau IFF_BROADCAST de l'interface.

[−]pointopoint [addr]
: Autorise le mode point−à−point pour l'interface, ce qui signifie qu'il existe un lien direct entre deux machines sans que quelqu'un d'autre puisse être à l'écoute. Si une adresse est donnée comme

argument, positionne l'adresse protocole à l'autre extrémité du lien, tout comme le faisait la commande dstaddr, devenue obsolète. Autrement, positionne ou efface le drapeau IFF_POINTTOPOINT de l'interface.

hw class address
: Positionne l'adresse matérielle de l'interface, si le gestionnaire de périphérique supporte cela. Le mot−clé doit être suivi par le nom de la classe matérielle et l'équivalent ASCII de l'adresse matérielle. Les classes matérielles actuellement supportées sont ether (Ethernet), ax25 5AMPR AX.25), ARCnet et netrom (AMPR NET/ROM).

multicast
: Positionne le drapeau multicast de l'interface. Normalement on n'en a pas besoin étant donné que les gestionnaires de périphérique positionne eux−mêmes le drapeau correctement.

address
: L'adresse IP que l'on doit assigner à l'interface.

txqueuelen length
: Positionne la longueur de la file de transmission du périphérique. Il est préférable de la mettre à une valeure faible pour les périphériques les plus lents ayant une latence (liens modem, ISDN) pour empêcher que les grosses masses de transferts comme telnet perturbent le trafic sur l'interface.

Vous pouvez utiliser la commande *ifconfig* pour toutes les interfaces réseau. Quelques programmes utilisateurs comme *pppd* et *dip* configurent automatiquement les périphériques en même temps qu'ils les créent, dès lors l'utilisation manuelle de *ifconfig* n'est pas nécessaire.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

5.5. Configurer votre solveur de noms

Le `*Solveur de Noms*' (Name Resolver) fait partie de la bibliothèque standard de Linux. Sa première fonction est de convertir des noms d'hôtes compréhensibles par l'homme, comme `*ftp.funet.fi*' , en adresses IP compréhensibles par une machine, comme *128.214.248.6*.

5.5.1. Qu'y a−t−il dans un nom ?

Vous êtes probablement familiers avec l'aspect des noms d'hôtes Internet, mais vous ne savez pas comment ils sont composés ou décomposés. Les noms de domaine Internet sont hiérarchisés par nature, c'est−à−dire qu'ils ont une structure arborescente. Un `*domaine*' est une famille, ou un groupe de noms. Un `*domaine*' peut être subdivisé en `*sous−domaines*'. Un `*domaine de premier niveau*' est un domaine qui n'est pas un sous−domaine. Les Domaines de Premier Niveau sont spécifiés dans la RFC−920. Quelques exemples :

COM
: Organisations Commerciales

EDU
: Organisations ayant rapport avec l'Éducation

GOV
: Organisations Gouvernementales (NdT: parfois GOUV en France !)

MIL
: Organisations Militaires

ORG
: Autres organisations

NET
: Organisations ayant un rapport avec l'internet

Nom de Pays
: il existe des codes de deux lettres qui représentent un pays donné.

Pour des raisons historiques la plupart des domaines appartenant à des domaines qui ne sont pas basés sur des noms de pays sont pour les organisations situées aux États−Unis, bien que les États−Unis aient aussi le code de pays `*.us*'. Ce n'est plus vrai pour les domaines *.com* et *.org*, qui sont couramment utilisés par des sociétés hors des États−Unis.

Chacun de ces domaines de premier niveau possède des sous−domaines. Les domaines de premier niveau fondés sur les noms de pays sont divisés ensuite en sous−domaines basés sur les domaines *com*, *edu*, *gov*, *mil* et *org* . Ainsi par exemple, vous finissez par : *com.au* et *gov.au* pour des organisations commerciales ou gouvernementales situées en Australie ; notez que ce n'est pas une règle absolue, car les politiques réelles dépendant de l'autorité qui donne les noms pour chaque domaine.

Le niveau de division suivant représente habituellement le nom de l'organisation. Ces sous−domaines sont variables, souvent ils sont fondés sur la structure en départements de l'organisation mais ils peuvent l'être également sur d'autres critères considérés comme rationnels et compréhensibles par les administrateurs réseau de l'organisation.

La partie tout à fait à gauche du nom est toujours le nom unique assigné à la machine hôte et est appelée le nom d'hôte `*hostname*', la partie de droite du nom est le nom de domaine `*domainname*' et le nom complet s'appelle le nom de domaine complètement qualifié `*Fully Qualified Domain Name*' (ou FQDN).

Si l'on examine l'adresse de la machine de Terry par exemple, le nom pleinement qualifié est `*perf.no.itg.telstra.com.au*'. Cela veut dire que le nom d'hôte est `*perf*' et le nom de domaine `*no.itg.telstra.com.au*'. Le nom de domaine est fondé sur un domaine de premier niveau basé sur son pays, l'Australie et comme son adresse électronique appartient à une organisation commerciale nous avons `*.com*' comme domaine de niveau adjacent. Le nom de la société est (était) `*Telstra*' et notre structure interne de noms est basé sur la structure organisationnelle, dans mon cas, ma machine appartient à l'Information Technology Group, section Network Operations.

Habituellement, les noms sont beaucoup plus courts ; par exemple, mon fournisseur d'accès à l'internet est ``*systemy.it*'' et mon organisation à but non lucratif est ``*linux.it*'', sans sous−domaine *com* ou *org*, aussi mon propre hôte est simplement appelé ``*morgana.systemy.it*'' et *rubini@linux.it* est une adresse électronique valide. Notez que le propriétaire d'un domaine a le droit d'enregistrer les noms d'hôtes aussi bien que les noms de sous−domaine ; par exemple le Groupe d'Utilisateur Linux auquel j'appartiens utilise le domaine *pluto.linux.it*, car les propriétaires de *linux.it* étaient d'accord pour créer un sous−domaine pour ce groupe.

5.5.2. Les informations nécessaires

Vous devez connaître le domaine auquel votre nom d'hôte appartient. Le solveur de nom effectue la traduction en faisant appel à un `*Serveur de Noms de Domaine*'. Vous devez connaître l'adresse IP d'un serveur de nom local que vous pouvez utiliser.

Il y a trois fichiers que vous devez éditer, nous en parlerons chacun à leur tour.

5.5.3. /etc/resolv.conf

Le fichier */etc/resolv.conf* est le fichier principal de configuration de la résolution de noms. Son format est très simple. C'est un fichier texte avec un mot−clé par ligne. Il y a trois mots−clés typiquement utilisés, qui sont :

domain
: ce mot−clé indique le nom de domaine local.

search
: ce mot−clé spécifie une liste d'autres noms de domaine pour rechercher un nom d'hôte.

name server
: ce mot−clé, qui peut être utilisé plusieurs fois, spécifie l'adresse IP d'un serveur de nom de domaine pour la résolution de noms.

Un exemple de */etc/resolv.conf* pourrait ressembler à ceci :

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
name server 192.168.10.1
name server 192.168.12.1
```

Cet exemple spécifie que le nom de domaine par défaut à ajouter aux noms non qualifiés (c'est−à−dire sans domaine) est *maths.wu.edu.au*, et que si l'hôte n'est pas trouvé dans ce domaine on peut aussi essayer le domaine *wu.edu.au* directement. Deux entrées de serveurs de noms sont fournies, chacune d'elles pouvant être appelée par le solveur de noms.

5.5.4. /etc/host.conf

Le fichier */etc/host.conf* sert à configurer certaines choses en vue de modifier le comportement du solveur de noms. Son format est décrit en détail dans la page de manuel `*resolv+*'. Dans le plupart des cas l'exemple suivant vous conviendra :

```
order hosts,bind
multi on
```

Cette configuration indique au solveur de nom de vérifier en premier lieu le fichier */etc/hosts* avant d'essayer un serveur de noms. Cela permet aussi au résolveur de nom de renvoyer toutes les adresses valables d'un hôte trouvé dans le fichier */etc/hosts* au lieu d'en donner simplement la première.

5.5.5. /etc/hosts

Le fichier */etc/hosts* est l'endroit où vous mettez les noms et les adresses IP des hôtes locaux. Si vous mettez un hôte dans ce fichier, alors vous n'avez pas à interroger le serveur de nom de domaine pour obtenir son adresse IP. L'inconvénient est que si l'adresse de cet hôte a changé, vous devez tenir votre fichier à jour. Dans un système bien administré les seuls noms d'hôtes qui apparaissent habituellement sont l'interface loopback, et le nom des hôtes locaux.

```
# /etc/hosts
127.0.0.1      localhost loopback
192.168.0.1    ma.belle.machine
```

Vous pouvez spécifier plus d'un nom d'hôte, comme montré dans la première entrée (qui est standard pour l'interface loopback).

5.5.6. Faire tourner un serveur de noms

Si vous voulez faire tourner un serveur de nom local, vous pouvez le faire facilement. Voyez le le DNS−HOWTO ainsi que tous les documents inclus dans votre version de *BIND* (Berkeley Internet Name Domain).

5.6. Configurer votre interface loopback

L'interface `*loopback*' est un type spécial d'interface qui permet de vous connecter à vous−même. Il y a plusieurs raisons pour faire cela. Par exemple si vous voulez faire des essais de logiciel réseau sans interférer avec quelqu'un d'autre sur votre réseau. Par convention, l'adresse IP `*127.0.0.1*' lui a été assignée. Aussi quelle que soit la machine où vous êtes, si vous ouvrez une connexion telnet vers *127.0.0.1* vous atteindrez toujours l'hôte local.

Configurer l'interface loopback est simple et vous devez vous assurer de l'avoir fait (mais notez que cette tâche est habituellement effectuée par les scripts standards d'initialisation).

```
root# ifconfig lo 127.0.0.1
root# route add −host 127.0.0.1 lo
```

Nous en dirons plus sur la commande *route* dans le prochain paragraphe.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

5.7. Routage

Le routage est un vaste sujet. On peut écrire de grandes quantités de textes sur ce sujet. La plupart d'entre vous ont besoin d'un simple routage, et certains même de rien du tout. Je ne parlerai que des principes du routage. Si vous voulez plus d'informations je vous suggère de vous reporter aux références fournies en début du document.

Commençons par une définition. Qu'est−ce que le routage IP ? Voici celle que j'utilise :

"Le routage IP est le processus par lequel un hôte, ayant des connexions réseau multiples, décide du chemin par lequel délivrer les datagrammes IP qu'il a reçus."

Il peut être utile d'illustrer cela par un exemple. Imaginez un routeur dans un bureau : il peut avoir un lien PPP sur l'Internet, un certain nombre de segments Ethernet alimentant les stations de travail et un second lien PPP vers un autre bureau. Lorsque le routeur reçoit un datagramme de l'une de ses connexions, le routage est le mécanisme utilisé pour déterminer vers quelle interface il doit renvoyer ce datagramme. De simples hôtes ont besoin aussi de routage, tous les hôtes Internet ayant deux périphériques réseau, l'un étant l'interface loopback décrite auparavant et l'autre est celui qui est utilisé pour parler avec le reste du monde, soit un lien Ethernet, soit une interface série PPP ou SLIP.

Ok, alors comment fonctionne le routage ? Chaque hôte possède une liste spéciale de règles de routage, appelée une table de routage. Cette table contient des colonnes qui contiennent au moins trois champs, le premier étant une adresse de destination, le deuxième étant le nom de l'interface vers lequel le datagramme doit être routé et le troisième, qui est optionnel, l'adresse IP d'une autre machine qui transportera le datagramme vers sa prochaine destination sur le réseau passerelle. Sur Linux vous pouvez voir cette table en utilisant la commande suivante :

```
user% cat /proc/net/route
```

ou bien en utilisant l'une des commandes suivantes :

```
user% /sbin/route −n
user% /sbin/netstat −r
```

Le processus de routage est plutôt simple : un datagramme entrant est reçu, l'adresse de destination est examinée et comparée avec chaque entrée de la table. L'entrée qui correspond le mieux à cette adresse est choisie, et le datagramme est renvoyé vers l'interface spécifiée. Si le champ passerelle est rempli, alors le datagramme est renvoyé vers cet hôte via l'interface spécifiée, sinon l'adresse de destination est supposée comme étant sur le réseau supporté par l'interface.

Pour manipuler ce tableau, une commande spéciale est utilisée. Cette commande prend des arguments et les convertit en appels système pour demander au noyau d'ajouter, supprimer ou modifier des entrées dans la table de routage. Cette commande s'appelle `*route*'.

Un exemple simple. Imaginez que vous ayez un réseau Ethernet. On vous a dit que c'est un réseau classe C avec une adresse de *192.168.1.0*. On vous fournit une adresse IP *192.168.1.10* pour votre usage et on vous a dit que *192.168.1.1* est un routeur connecté à l'Internet.

La première étape est de configurer l'interface comme indiqué plus haut. Vous utiliserez la commande :

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Maintenant vous avez besoin d'ajouter une entrée dans la table de routage pour indiquer au noyau que les datagrammes destinés aux hôtes dont les adresses correspondent à *192.168.1.** doivent être dirigés vers le périphérique Ethernet. Vous utiliserez une commande comme ceci :

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Notez l'utilisation de l'argument `*−net*' pour indiquer au programme route que cette entrée est une route réseau. Un autre choix peut être `*−host*' qui est une route spécifique d'une adresse IP.

Cette route vous permettra d'établir des connexions IP avec tous les hôtes sur votre segment Ethernet. Mais qu'en est−il des hôtes IP qui n'y sont pas ?

Il serait compliqué d'ajouter des routes pour chaque réseau destinataire, aussi il y a une astuce utilisée pour simplifier la tâche. L'astuce est appelée route par `*defaut*'. La route par *defaut* s'adapte à toutes les destinations possibles, mais pas très bien, de telle sorte que si il y a une entrée qui correspond à l'adresse requise elle sera utilisée à la place de la route par *defaut*. L'idée de la route par *defaut* est simplement de pouvoir dire `et tout le reste va ici'. Dans l'exemple que j'ai inventé, on utilisera une entrée telle que :

```
root# route add default gw 192.168.1.1 eth0
```

L'argument `*gw*' indique à la commande route que le prochain argument est l'adresse IP, ou le nom, d'une passerelle (gateway) ou d'une machine routeur vers qui tous les datagrammes correspondant à cette entrée seront dirigés pour routage ultérieur.

Ainsi votre configuration complète sera :

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add default gw 192.168.1.1 eth0
```

Si vous regardez bien vos fichiers `*rc*' qui concernent le réseau vous en trouverez au moins un très semblable à celui−ci. C'est une configuration courante.

Examinons maintenant une configuration un peu plus compliquée. Imaginons que nous configurions le routeur examiné auparavant, celui qui avait un lien PPP vers l'Internet et des segments LAN alimentant des stations de travail dans le bureau. Supposons que ce routeur ait 3 segments Ethernet et un lien PPP. Notre configuration de routage ressemblerait à ceci :

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
root# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
root# route add default ppp0
```

Chacune des stations de travail utilisera le format plus simple décrit ci−dessus, seul le routeur aura besoin d'indiquer les routes réseau séparément car pour les stations de travail le mécanisme de routage par *defaut* les capturera toutes, laissant au routeur le soin de les séparer de manière appropriée. Vous pouvez vous demander pourquoi la route par défaut n'utilise pas `*gw*'. La raison en est très simple : les protocoles de lien série comme PPP et SLIP ont seulement deux hôtes sur leur réseau, un à chaque bout. Spécifier à l'hôte que l'autre bout de la liaison est une passerelle est sans objet et redondant, car il n'a pas d'autre choix, aussi vous n'avez pas à indiquer de passerelle pour ce type de connexions réseau. Les autres types comme Ethernet, arcnet ou token ring ont besoin que l'on indique une passerelle car ces réseaux supportent un grand nombre d'hôtes.

5.7.1. Alors, que fait le programme *routed* ?

La configuration de routage décrite ci−dessus est bien adaptée aux réseaux simples où il n'y a que des chemins uniques entre les destinations. Lorsque vous avez un réseau plus complexe les choses deviennent plus compliquées. Heureusement pour la plupart d'entre vous, ce ne sera pas le cas.

Le gros problème est qu'avec le `routage manuel' ou `routage statique' comme décrit ci−dessus, si une machine ou un lien tombe en panne dans le réseau, alors la seule façon de diriger vos datagrammes vers un autre chemin, s'il existe, est d'intervenir manuellement et d'exécuter les commandes adéquates. Naturellement c'est lourd, lent, peu pratique et source de risques. Des techniques variées ont été développées pour régler automatiquement les tables de routage dans le cas d'incidents sur un réseau où il y a plusieurs routes possibles, toutes ces techniques étant regroupées sous le nom de `protocoles de routage dynamique'.

Vous avez peut−être entendu parler des plus courants. Ce sont RIP (Routing Information Protocol) et OSPF (Open Shortest Path First Protocol). RIP est très souvent utilisé sur les petits ou moyens réseaux d'entreprise. L'OPSF est plus moderne et plus apte à gérer de grands réseaux et mieux adapté dans le cas où il y a un grand nombre de chemins possibles à travers le réseau. Les implémentations usuelles de ces protocoles sont : `*routed*' − RIP, et `*gated*' − RIP, OSPF et autres. Le programme `*routed*' est normalement fourni avec votre distribution Linux ou est inclus dans la paquetage `NetKit' décrit auparavant.

Un exemple pour vous montrer comment et où vous pouvez utiliser un protocole de routage dynamique ressemblerait à ceci :

```



    192.168.1.0 /                          192.168.2.0 /
       255.255.255.0                          255.255.255.0
     -                                     -
     |                                     |
     |   /- - -\                /- - -\    |
     |   |     |ppp0   //    ppp0|     |    |
eth0 |- -|  A  |- - - //- - - - -|  B  |- -| eth0
     |   |     |     //         |     |    |
     |   \- - -/                \- - -/    |
     |      \ ppp1            ppp1 /       |
     -       \                    /        -
              \                  /
               \                /
                \              /
                 \            /
                  \          /
                   \        /
```

```
                   \       /
                    \     /
                 ppp0\   /ppp1
                    /- - -\
                    |     |
                    |  C  |
                    |     |
                    \- - -/
                       |eth0
                       |
                  |- - - - -|
                  192.168.3.0 /
                     255.255.255.0
```

Nous avons trois routeurs A, B et C. Chacun supporte un segment Ethernet avec un réseau IP de classe C (masque de réseau 255.255.255.0). Chaque routeur a également une liaison PPP vers chacun des autres routeurs. Ce réseau forme un triangle.

Il est évident que la table de routage sur le routeur A ressemble à ceci :

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
root# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Cela fonctionnera bien jusqu'à ce que le lien entre A et B tombe en panne. Les hôtes sur le segment A (voir le diagramme ci−dessus) ne peuvent pas atteindre les hôtes sur le segment B : leurs datagrammes seront dirigés sur le lien ppp0 du routeur A qui est rompu. Ils pourront encore continuer à parler aux hôtes du segment C, et les hôtes du segment C pourront toujours parler à ceux du segment B car la liaison reste intacte.

Si A peut parler à C et si C peut toujours parler à B, pourquoi A ne routerait−il pas ses datagrammes pour B via C, et laisser ensuite C les envoyer à B ? C'est exactement le type de problèmes que les protocoles de routage dynamique comme RIP sont en mesure de résoudre. Si chacun des routeurs A, B et C utilisent un démon de routage (NdT: démon est une francisation familière du vocable informatique anglais daemon, qui signifie Disk And Extension MONitor, c'est à dire qui n'est pas invoqué manuellement mais attend en tâche de fond que quelque chose se passe, que quelque condition se produise. Ce terme fut introduit au départ sous CTSS (Compatible Time Sharing System), un ancêtre du système MULTICS, lui−même parent d'UNIX (voir la traduction de René Cougnenc de `Le système Linux' de M. Welsh et L. Kaufman chez O'Reilly International Thomson), alors leurs tables de routage seront automatiquement réglées pour refléter le nouvel état du réseau même si l'une des liaisons est défectueuse. Configurer un tel réseau est simple, sur chaque routeur vous devez seulement faire deux choses. Dans ce cas, pour le routeur A :

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# /usr/sbin/routed
```

Le démon de routage `*routed*' trouve automatiquement tous les ports actifs vers le réseau quand il démarre et écoute tous les messages sur chacun des périphériques réseau ce qui lui permet de déterminer et de mettre à jour sa table de routage.

C'était une très brève explication du routage dynamique et de son utilisation. Si vous voulez d'avantage d'explications reportez−vous aux références listées en début de document.

Les points importants relatifs au routage dynamique sont :

1. Vous n'avez besoin d'utiliser un démon de routage dynamique que quand votre machine Linux peut choisir entre plusieurs routes pour une destination donnée. C'est la cas par exemple lorsque vous envisagez d'utiliser IP masquerade.

2. Le démon de routage dynamique modifiera automatiquement votre table de routage pour tenir compte des changements survenus dans votre réseau.
3. RIP est adapté aux réseaux de petite et moyenne taille.

5.8. Configurer vos serveurs réseau et les services.

Les serveurs de réseau et les services sont des programmes qui permettent à un utilisateur distant de devenir utilisateur de votre machine Linux. Les programmes serveurs sont à l'écoute des ports réseau. Les ports réseau permettent de demander un service particulier à un hôte particulier et de faire la différence entre une connexion telnet entrante et une connexion ftp entrante. L'utilisateur distant établit une connexion réseau avec votre machine puis le programme serveur, ou démon de réseau, à l'écoute du port, accepte la connexion et s'exécute. Il y a deux façons d'opérer pour les démons de réseau. Les deux sont couramment utilisés en pratique. Ce sont :

autonome
: le programme démon écoute le port réseau désigné et lorsqu'il y a une connexion, il prend lui−même la connexion en charge pour fournir le service.

esclave du serveur inetd
: le serveur *inetd* est un programme démon spécial spécialisé dans la conduite des connexions réseau. Il possède un fichier de configuration qui indique quel programme doit être utilisé lorsqu'une connexion entrante est reçue. Chacun des ports service doit être configuré soit avec le protocole tcp, soit avec le protocole udp. Les ports sont décrits dans un autre fichier dont nous parlerons plus tard.

Il existe deux fichiers importants qui doivent être configurés : */etc/services* qui assigne des noms aux numéros de port et */etc/inetd.conf* qui sert pour la configuration du démon de réseau *inetd* .

5.8.1. */etc/services*

Le fichier */etc/services* est une simple base de données qui associe des noms compréhensibles par l'homme à des ports service compréhensibles par la machine. Son format est tout à fait simple. Le fichier est un fichier texte dont chaque ligne représente une entrée de la base de données. Chaque entrée comprend trois champs séparés par des caractères espace ou tabulation. Ces champs sont :

nom port/protocole alias # commentaire

nom
: un simple mot qui représente le service décrit.

port/protocole
: ce champ est divisé en deux.

port
: un nombre qui spécifie le numéro de port où le service désigné sera disponible. La plupart des services ont des numéros assignés. Ils sont décrits dans la *RFC−1340*.

protocole
: c'est soit *tcp* soit *udp*.

: Il est important de noter qu'une entrée comme *18/tcp* est très différente de *18/udp* et qu'il n'y a pas de raisons techniques que le même service existe sur les deux. Normalement le bon sens prévaut et c'est vraiment pour un service particulier disponible à la fois sur *tcp* et *udp* que vous verrez une entrée pour les deux..

aliases
: Autres noms qui peuvent être utilisés pour se référer à un service.

Tout texte apparaissant après le caractère `*#*' est ignoré et traité comme commentaire.

5.8.1.1. Exemple de fichier */etc/services*.

Toutes les distributions récentes de Linux fournissent un bon fichier */etc/services*. Juste au cas où vous construiriez tout depuis le départ, voici une copie du fichier */etc/services* fourni avec l'ancienne distribution Debian .

```
# /etc/services:
# $Id: Net-HOWTO.sgml,v 1.1.1.1 2003/01/03 02:38:54 traduc Exp $
#
# Network services, Internet style
#
# Notez que c'est la politique actuelle de l'IANA d'assigner un seul numéro
# de port à la fois pour TCP et UDP; ainsi, la plupart des ports ont deux
# entrées même si le protocole ne supporte pas UDP.
# Mis à jour d'après la RFC 1340, ``Assigned Numbers'' (Juillet 1992).
# Il n'y a pas tous les ports, seulement les plus courants.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          QUOTE
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - non assigné
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
bootps          67/tcp                          # BOOTP server
bootps          67/udp
bootpc          68/tcp                          # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
www             80/udp                          # HyperText Transfer Protocol
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5  # Kerberos v5
kerberos        88/udp          kerberos5 krb5  # Kerberos v5
```

```
supdup          95/tcp
# 100 - reserve
hostnames       101/tcp         hostname        # usually from sri-nic
iso-tsap        102/tcp         tsap            # part of ISODE.
csnet-ns        105/tcp         cso-ns          # also used by CSO name server
csnet-ns        105/udp         cso-ns
rtelnet         107/tcp                         # Remote Telnet
rtelnet         107/udp
pop-2           109/tcp         postoffice      # POP version 2
pop-2           109/udp
pop-3           110/tcp                         # POP version 3
pop-3           110/udp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
imap2           143/tcp                         # Interim Mail Access Proto v2
imap2           143/udp
snmp            161/udp                         # Simple Net Mgmt Proto
snmp-trap       162/udp         snmptrap        # Traps for SNMP
cmip-man        163/tcp                         # ISO mgmt over IP (CMOT)
cmip-man        163/udp
cmip-agent      164/tcp
cmip-agent      164/udp
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
nextstep        178/tcp         NeXTStep NextStep       # NeXTStep window
nextstep        178/udp         NeXTStep NextStep       # server
bgp             179/tcp                         # Border Gateway Proto.
bgp             179/udp
prospero        191/tcp                         # Cliff Neuman's Prospero
prospero        191/udp
irc             194/tcp                         # Internet Relay Chat
irc             194/udp
smux            199/tcp                         # SNMP Unix Multiplexer
smux            199/udp
at-rtmp         201/tcp                         # AppleTalk routing
at-rtmp         201/udp
at-nbp          202/tcp                         # AppleTalk name binding
at-nbp          202/udp
at-echo         204/tcp                         # AppleTalk echo
at-echo         204/udp
at-zis          206/tcp                         # AppleTalk zone information
at-zis          206/udp
z3950           210/tcp         wais            # NISO Z39.50 database
z3950           210/udp         wais
ipx             213/tcp                         # IPX
ipx             213/udp
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
ulistserv       372/tcp                         # UNIX Listserv
ulistserv       372/udp
#
# services spécifiques à UNIX
#
```

```
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
route           520/udp         router routed   # RIP
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
klogin          543/tcp                         # Kerberized `rlogin' (v5)
kshell          544/tcp         krcmd           # Kerberized `rsh' (v5)
kerberos-adm    749/tcp                         # Kerberos `kadmin' (v5)
#
webster         765/tcp                         # Network dictionary
webster         765/udp
#
# D'après ``Assigned Numbers'' :
#
#> Les Ports Enregistrés ne sont pas contrôlés par l'IANA et peuvent être
#> utilisés sur la plupart des systèmes par des processus ordinaires
#> ou des programmes exécutés par des utilisateurs ordinaires.
#
#>Les ports sont utilisés dans le TCP [45,106] pour nommer les extrémités
#> des connexions logiques qui transportent les conversations de longue
#> durée. Pour offrir des services à des utilisateurs non connus, un port
#> de service pour contact a été défini. Cette liste spécifie le port utilisé
#> par le processus serveur ainsi que son port de contact. Comme l'IANA ne peut
#> contrôler l'usage de ces ports, on donne ici une liste d'utilisation
#> de ces ports pour être agréable à la communauté.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np     1525/tcp                # Prospero non-privileged
prospero-np     1525/udp
rfe             5002/tcp                # Radio Free Ethernet
rfe             5002/udp                # Actually uses UDP only
bbs             7000/tcp                # BBS service
#
#
# services Kerberos (Project Athena/MIT)
# Notez que ceux-ci sont pour Kerberos v4, et ne sont pas officiels. Les sites
# tournant sous v4 doivent utiliser ceux-ci et annuler les entrées v5 ci-dessus.
#
kerberos4       750/udp         kdc     # Kerberos (server) udp
kerberos4       750/tcp         kdc     # Kerberos (server) tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
krb_prop        754/tcp                 # Kerberos slave propagation
krbupdate       760/tcp         kreg    # Kerberos registration
kpasswd         761/tcp         kpwd    # Kerberos "passwd"
kpop            1109/tcp                # Pop with Kerberos
knetd           2053/tcp                # Kerberos de-multiplexor
zephyr-srv      2102/udp                # Zephyr server
zephyr-clt      2103/udp                # Zephyr serv-hm connection
zephyr-hm       2104/udp                # Zephyr hostmanager
```

```
eklogin         2105/tcp                # Kerberos encrypted rlogin
#
# Services non officiels mais nécessaires (pour NetBSD)
#
supfilesrv      871/tcp                 # SUP server
supfiledbg      1127/tcp                # SUP debugging
#
# Services protocole de délivrance de datagrammes
#
rtmp            1/ddp                   # Routing Table Maintenance Protocol
nbp             2/ddp                   # Name Binding Protocol
echo            4/ddp                   # AppleTalk Echo Protocol
zip             6/ddp                   # Zone Information Protocol
#
# Services Debian GNU/Linux
rmtcfg          1236/tcp                # Gracilis Packeten remote config server
xtel            1313/tcp                # french minitel
cfinger         2003/tcp                # GNU Finger
postgres        4321/tcp                # POSTGRES
mandelspawn     9359/udp        mandelbrot      # network mandelbrot

# Services locaux
```

Dans la réalité, le fichier augmente toujours en taille au fur et à mesure que de nouveaux services apparaissent. Si vous craignez que votre copie soit incomplète, je vous suggère de copier un nouveau fichier */etc/services* provenant d'une distribution récente.

5.8.2. */etc/inetd.conf*

Le fichier */etc/inetd.conf* est le fichier de configuration du serveur démon *inetd* . Il sert à dire à *inetd* ce qu'il doit faire lorsqu'il reçoit une demande de connexion pour un service particulier. Pour les services où vous acceptez une connexion vous devez dire à *inetd* quel démon serveur de réseau doit tourner, et comment.

Son format est aussi très simple. C'est un fichier texte dont chaque ligne décrit un service que vous voulez fournir. Tout texte suivant un `*#* est ignoré et considéré comme commentaire. Chaque ligne contient sept champs séparés par un nombre quelconque d'espaces (espace ou tabulation). Le format général est comme suit :

```
service  type_de_socket  protocole  drapeaux  utilisateur  chemin  arguments
```

service
: est le nom de service applicable à cette configuration, pris dans le fichier */etc/services*.

type_de_socket
: ce champ décrit le type de socket que cette entrée considère comme pertinent. Voici les valeurs qui sont autorisées : *stream*, *dgram*, *raw*, *rdm*, ou *seqpacket*. C'est un peu technique par nature, mais par expérience, presque tous les services basés sur *tcp* utilisent *stream* et presque tous les services basés sur *udp* utilisent *dgram*. Il n'y a que quelques types de serveurs démons spéciaux utilisant d'autres valeurs.

protocole
: le protocole considéré comme valide pour cette entrée. Il doit correspondre à l'entrée appropriée dans le fichier */etc/services* et sera donc soit *tcp* soit *udp*. Les serveurs basés sur Sun RPC (Remote Procedure Call) utilisent *rpc/tcp* ou *rpc/udp*.

drapeaux
: il n'y a en fait que deux valeurs pour ce champ. Celles−ci disent à *inetd* si le programme serveur réseau libère le socket aprés démarrage, et donc si *inetd* peut prendre en compte une des prochaines demandes de connexion, ou bien si *inetd* doit attendre qu'un autre démon serveur tournant déjà prenne en charge la nouvelle demande de connexion. C'est encore compliqué, mais en pratique tous les serveurs *tcp* doivent avoir cette entrée positionnée sur *nowait* et la plupart des serveurs *udp* ont cette

entrée positionnée sur *wait*. Attention il y a quelques exceptions notables, laissez vous guider par l'exemple suivant si vous n'êtes pas sûrs.

utilisateur
: ce champ décrit quel compte utilisateur extrait de */etc/passwd* sera considéré comme propriétaire du démon réseau lorsqu'il est lancé. C'est très utile lorsque vous voulez vous protéger contre les trous de sécurité. Vous pouvez mettre *nobody* comme utilisateur pour une entrée si bien que dans le cas où le réseau comporte une brèche, les dommages éventuels seront minimisés. Cependant habituellement ce champ est réglé sur *root*, car de nombreux serveurs ont besoin des privilèges de root pour tourner correctement.

chemin_de_serveur
: ce champ est le véritable chemin d'accès au programme.

arguments
: ce champ correspond au reste de la ligne et est optionnel. Il sert à indiquer les arguments de commande que vous voulez passer au programme serveur au lancement.

5.8.2.1. Exemple de fichier */etc/inetd.conf*

Comme pour le fichier */etc/services*, toutes les distributions modernes incluent un bon fichier */etc/inetd.conf* pour pouvoir travailler. Ici, pour être complet , vous trouverez le fichier */etc/inetd.conf* de la distribution Debian.

```
# /etc/inetd.conf:  voir inetd(8) pour d'autres informations.
#
# Base de données pour la configuration d'un serveur Internet
#
#
# Modifié pour Debian par Peter Tobias <tobias@et-inf.fho-emden.de>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Services internes
#
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
discard         stream  tcp     nowait  root    internal
discard         dgram   udp     wait    root    internal
daytime         stream  tcp     nowait  root    internal
daytime         dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# Services standards.
#
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec et talk sont des protocoles BSD.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Services Mail, news et uucp.
#
smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
```

```
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
# Pop et autres
#
#pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# `cfinger' est le serveur finger GNU de Debian.  (NOTE : L'implémentation
# habituelle du démon `finger' permet de le faire tourner avec `root'.)
#
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
#
# Le service tftp est fourni principalement pour démarrer. La plupart des sites
# l'utilisent seulement sur les machines servant de `serveurs de boot'.
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i -t 120
#
# Services Kerberos (ils doivent probablement être corrigés)
#
#klogin         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k -x
#kshell         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services tournant UNIQUEMENT sur Kerberos (doivent être probablement corrigés)
#
#krbupdate      stream tcp      nowait  root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# Services RPC
#
#mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
#rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1        dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# Fin de inetd.conf.
ident           stream  tcp     nowait  nobody  /usr/sbin/identd       identd -i
```

5.9. Autres fichiers de configuration ayant un rapport avec le réseau

Il y a de nombreux fichiers relatifs à la configuration réseau sous Linux et qui sont susceptibles de vous intéresser. Vous n'aurez jamais à modifier ces fichiers, mais il est utile de les décrire pour que vous sachiez ce qu'ils contiennent et quelle est leur utilité.

5.9.1. */etc/protocols*

Le fichier */etc/protocols* est une base de données qui donne la relation des numéros id de protocole avec leurs noms. Il est utilisé par les programmeurs pour leur permettre de spécifier les protocoles par leur nom dans les programmes et aussi par quelques programmes tels que *tcpdump* pour pouvoir afficher en sortie des noms au lieu de chiffres. La syntaxe générale de ce fichier est :

```
nom du protocole     numéro    alias
```

Le fichier */etc/protocols* fourni avec la distribution Debian est le suivant :

```
# /etc/protocols:
# $Id: Net-HOWTO.sgml,v 1.1.1.1 2003/01/03 02:38:54 traduc Exp $
#
# Protocoles Internet (IP)
#
#       d'après: @(#)protocols  5.1 (Berkeley) 4/17/89
#
# Mise à jour pour NetBSD basee sur la RFC 1340, Assigned Numbers (July 1992).

ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf    73      RSPF            # Radio Shortest Path First.
vmtp    81      VMTP            # Versatile Message Transport
ospf    89      OSPFIGP         # Open Shortest Path First IGP
ipip    94      IPIP            # Yet Another IP encapsulation
encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.2. */etc/networks*

Le fichier */etc/networks* a une fonction similaire au fichier */etc/hosts*. Il fournit une simple base de données de noms de réseau avec des adresses. Son format diffère en ce qu'il n'y a que deux champs par ligne, et que ces champs sont codés comme ceci :

```
 Nom du réseau   adresse de réseau
```

Un exemple :

```
loopnet    127.0.0.0
localnet   192.168.0.0
amprnet    44.0.0.0
```

Vous obtiendrez le nom du réseau (et NON son adresse) en utilisant une commande telle que *route* dans l'exemple suivant : la destination est un réseau, et ce réseau possède une entrée dans le fichier */etc/networks*.

5.10. Sécurité réseau et contrôle d'accès

Laissez−moi commencer ce paragraphe en vous avertissant que la sécurisation de votre machine et du réseau contre les attaques pernicieuses est un art complexe. Je ne me considère pas du tout comme un expert dans ce domaine et bien que les mécanismes que je vais décrire puissent vous aider, si vous êtes préoccupés par la sécurité, alors je vous recommande d'effectuer vous−même des recherches sur le sujet. Il existe un grand nombre d'excellentes références sur l'Internet qui traitent du sujet, y compris le Security−HOWTO

Une importante règle pratique est : `*N'utilisez pas de serveurs dont vous n'avez pas besoin*'. Un grand nombre de distributions sont livrées avec tout un tas de services déjà configurés et démarrant automatiquement. Pour assurer quand même un minimum de sécurité vous devriez aller dans votre fichier */etc/inetd.conf* et retirez (*placez un `#' au début de la ligne*) toute entrée que vous ne comptez pas utiliser. De bons candidats sont : *shell*, *login*, *exec*, *uucp*, *ftp*, et les services informatifs tels que *finger*, *netstat* and *systat*.

Il y a plein de sortes de sécurité et de mécanismes de contrôle d'accès ; je vais décrire les plus élémentaires.

5.10.1. /etc/ftpusers

Le fichier */etc/ftpusers* est un mécanisme simple qui vous permet d'interdire l'accès de votre machine à certains utilisateurs de ftp. Il est lu par le programme démon (*ftpd*) lorsqu'une connexion ftp est reçue. Le fichier est une simple liste d'utilisateurs qui ne peuvent pas se connecter. Il ressemble à :

```
# /etc/ftpusers - utilisateurs ne pouvant pas se connecter par ftp
root
uucp
bin
mail
```

5.10.2. /etc/securetty

Le fichier */etc/securetty* vous permet de spécifier sur quels fichiers de périphériques *tty root* a le droit de se connecter. Le fichier */etc/securetty* est lu par le programme de connexion (habituellement */bin/login*). Son format est une liste de fichiers de périphériques tty autorisés (sur tous les autres *root* ne peut se connecter) :

```
# /etc/securetty - consoles où root peut se connecter
tty1
tty2
tty3
tty4
```

5.10.3. Le mécanisme de contrôle d'accès des hôtes *tcpd*.

Le programme *tcpd* que vous avez vu dans le fichier */etc/inetd.conf* fournit les mécanismes de contrôle d'accès et de connexion aux services qu'il a pour but de protéger.

Lorsqu'il est invoqué par le programme *inetd*, il lit deux fichiers contenant les règles d'accès et il autorise ou interdit l'accès au serveur qu'il protège.

Il cherche dans ces deux fichiers jusqu'à ce qu'il trouve une correspondance. S'il n'en trouve pas il suppose que l'accès est autorisé. Il recherche dans l'ordre suivant : */etc/hosts.allow*, */etc/hosts.deny*. Je décrirai chacun d'eux plus tard. Pour une description complète référez−vous aux pages de manuel appropriées (*hosts_access(5)* est un bon point de départ).

5.10.3.1. /etc/hosts.allow

Le fichier */etc/hosts.allow* est un fichier de configuration du programme */usr/sbin/tcpd*. Il contient les hôtes dont l'accès est *autorisé (allowed)* et qui peuvent donc utiliser un service de votre machine.

Le format du fichier est très simple :

```
# /etc/hosts.allow
#
# <liste des services>: <liste des hôtes> [: commande]
```

liste des services
: c'est une liste de serveurs, séparés par des virgules, auxquels les règles d'accès s'appliquent. Exemples de serveur : *ftpd*, *telnetd*, et *fingerd*.

liste des hôtes
: c'est une liste de noms d'hôtes, séparés par des virgules (vous pouvez utiliser également des adresses IP). Vous pouvez en plus spécifier des noms d'hôtes ou des adresses IP avec des jokers pour obtenir des groupes d'hôtes. Des exemples : *gw.vk2ktj.ampr.orgi* pour un hôte spécifique, *.uts.edu.au* pour tous les hôtes se terminant par cette chaîne , *44.* pour toutes les adresses IP commençant par ces chiffres. Il y a quelques expressions pour simplifier la configuration, parmi lesquelles : *ALL* pour tous les hôtes, *LOCAL* pour tout hôte dont le nom ne contient pas de `.' c'est à dire appartenant au même domaine que votre machine, et *PARANOID* pour tout hôte dont le nom ne correspond pas avec son adresse (tricherie dans le nom). Il y a enfin une expression qui peut être utile. Il s'agit de *EXCEPT* qui vous permet de fournir une liste avec des exceptions. Nous verrons un exemple plus tard.

commande
: c'est un paramètre optionnel. Ce paramètre est le nom complet d'une commande (avec son répertoire) qui sera exécutée chaque fois qu'il y aura correspondance. Ce peut être par exemple une commande qui essaiera d'identifier qui se connecte, ou de générer un message par courrier ou tout message d'alerte pour l'administrateur système avertissant que quelqu'un est en train de se connecter. On peut y inclure des extensions, par exemple : *%h* donnera le nom de l'hôte qui se connecte ou bien son adresse s'il n'a pas de nom , *%d* le programme démon appelé.

Un exemple :

```
# /etc/hosts.allow
#
# Permet à tout le monde d'utiliser le courrier
in.smtpd: ALL
# telnet et ftp pour les hôtes de mon domaine et my.host.at.home.
telnetd, ftpd: LOCAL, myhost.athome.org.au
# finger pour tout le monde, mais garde une trace de l'identité.
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

5.10.3.2. /etc/hosts.deny

Le fichier */etc/hosts.deny* est un fichier de configuration du programme */usr/sbin/tcpd*. Ce fichier contient les hôtes qui *n'ont pas l'autorisation* d'accéder à l'un des services de votre machine.

Un exemple simple ressemblerait à ceci :

```
# /etc/hosts.deny
#
# Interdit l'acces aux hotes ayant des noms suspects
ALL: PARANOID
#
# Interdit l'acces a tous les hotes
ALL: ALL
```

L'entrée *PARANOID* est en fait redondante car l'autre entrée interdit tous les cas. L'une ou l'autre entrée devrait convenir, en fonction de vos besoins particuliers.

Mettre *ALL: ALL* par défaut dans le fichier */etc/hosts.deny* puis autoriser certains services, en liaison avec les hôtes que vous avez choisis, dans le fichier */etc/hosts.allow*, est la configuration la plus sûre.

5.10.4. /etc/hosts.equiv

Le fichier *hosts.equiv* est utilisé pour concéder à certains hôtes des droits d'accès leur permettant d'avoir un compte sur votre machine sans fournir de mot de passe. Cela est utile dans un environnement sécurisé où vous contrôlez toutes les machines, sinon ce peut être très risqué. Votre machine est aussi sûre que le moins sûr de vos hôtes de confiance. Pour augmenter la sécurité, n'utilisez pas cette possiblité et encouragez vos utilisateurs à ne pas utiliser le fichier *.rhosts*.

5.10.5. Configurer votre démon *ftp* correctement

Un grand nombre de sites sont intéressés à avoir un serveur *ftp* anonyme pour permettre aux autres de transférer et de récupérer des fichiers sans avoir besoin d'une identification spéciale. Si vous décidez d'offrir ce service soyez certains de configurer votre démon *ftp* de manière adéquate pour les accès anonymes. La plupart des pages de manuel dédiées à *ftpd(8)* décrivent tous les détails pour y arriver. Vous devez toujours vous assurer que vous avez bien suivi les instructions. Un règle importante est de ne pas utiliser une copie de votre fichier */etc/passwd* dans le répertoire */etc* du compte anonyme. Soyez sûrs d'avoir éliminé tous les détails des comptes exceptés ceux qui sont nécessaires, autrement vous serez vulnérables vis à vis de ceux qui maîtrisent les techniques de mise en pièces des mots de passe.

5.10.6. Pare−feu (Firewall) sur le réseau

Ne pas permettre aux datagrammes d'atteindre votre machine ou les serveurs est un excellent moyen de sécurisation. Ceci est abordé en profondeur dans le Firewall−HOWTO et (de manière plus concise) plus loin dans ce document.

5.10.7. Autres suggestions

Voici d'autres suggestions, potentiellement religieuses, à prendre en considération :

sendmail
: en dépit de sa popularité, le démon *sendmail* apparaît avec une effrayante régularité dans les mises en garde concernant la sécurité. Faites comme vous voulez, mais j'ai choisi de ne pas l'utiliser.

NFS et autres services Sun RPC
: soyez circonspects avec eux. Il y a toutes sortes d'exploits possibles avec ces services. Il est difficile de trouver une option pour les services tels que NFS, mais si vous les configurez, soyez prudents envers ceux à qui vous accordez des droits.

Chapter 6. Informations sur Ethernet

Cette section traite d'informations spécifiques sur Ethernet et la configuration des cartes Ethernet.

6.1. Cartes Ethernet supportées

6.1.1. 3Com

- 3Com 3c501 − `à fuir comme la peste' (gestionnaire 3c501)
- 3Com 3c503 (gestionnaire 3c503), 3c505 (gestionnaire 3c505), 3c507 (gestionnaire 3c507), 3c509/3c509B (ISA) / 3c579 (EISA)
- 3Com Etherlink III Vortex Ethercards (3c590, 3c592, 3c595, 3c597) (PCI), 3Com Etherlink XL Boomerang (3c900, 3c905) (PCI) et Cyclone (3c905B, 3c980) Ethercards (gestionnaire 3c59x) et 3Com Fast EtherLink Ethercard (3c515) (ISA) (gestionnaire 3c515)
- 3Com 3ccfe575 Cyclone Cardbus (gestionnaire 3c59x)

- 3Com 3c575 series Cardbus (gestionnaire 3c59x) (ALL PCMCIA ??)

6.1.2. AMD, ATT, Allied Telesis, Ansel, Apricot

- AMD LANCE (79C960) / PCnet−ISA/PCI (AT1500, HP J2405A, NE1500/NE2100)
- ATT GIS WaveLAN
- Allied Telesis AT1700
- Allied Telesis LA100PCI−T
- Allied Telesyn AT2400T/BT (module "ne")
- Ansel Communications AC3200 (EISA)
- Apricot Xen−II / 82596

6.1.3. Cabletron, Cogent, Crystal Lan

- Cabletron E21xx
- Cogent EM110
- Crystal Lan CS8920, Cs8900

6.1.4. Danpex, DEC, Digi, DLink

- Danpex EN−9400
- DEC DE425 (EISA) / DE434/DE435 (PCI) / DE450/DE500 (gestionnaire DE4x5)
- DEC DE450/DE500−XA (dc21x4x) (gestionnaire Tulip)
- DEC DEPCA et EtherWORKS
- DEC EtherWORKS 3 (DE203, DE204, DE205)
- DECchip DC21x4x "Tulip"
- DEC QSilver's (Gestionnaire Tulip)
- Digi International RightSwitch
- DLink DE−220P, DE−528CT, DE−530+, DFE−500TX, DFE−530TX

6.1.5. Fujitsu, HP, ICL, Intel

- Fujitsu FMV−181/182/183/184
- HP PCLAN (séries 27245 et 27xxx)
- HP PCLAN PLUS (27247B et 27252A)
- HP 10/100VG PCLAN (J2577, J2573, 27248B, J2585) (ISA/EISA/PCI)
- ICL EtherTeam 16i / 32 (EISA)
- Intel EtherExpress
- Intel EtherExpress Pro

6.1.6. KTI, Macromate, NCR NE2000/1000, Netgear, New Media

- KTI ET16/P−D2, ET16/P−DC ISA (fonctionne sans cavaliers et avec des options de configuration matérielles)
- Macromate MN−220P (PnP or NE2000 mode)
- NCR WaveLAN
- NE2000/NE1000 (attention aux clônes)
- Netgear FA−310TX (puce Tulip)
- New Media Ethernet

6.1.7. PureData, SEEQ, SMC

- PureData PDUC8028, PDI8023
- SEEQ 8005
- SMC Ultra / EtherEZ (ISA)
- SMC 9000 series
- SMC PCI EtherPower 10/100 (gestionnaire DEC Tulip)
- SMC EtherPower II (gestionnaire epic100.c)

6.1.8. Sun Lance, Sun Intel, Schneider, WD, Zenith, IBM, Enyx

- Adaptateurs Sun LANCEs (noyau kernel 2.2 et suivants)
- Adaptateurs Sun Intel (noyaux kernel 2.2 et suivants)
- Schneider et Koch G16
- Western Digital WD80x3
- Adaptateur intégré Zenith Z–Note / IBM ThinkPad 300
- Ensemble Znyx 312 (gestionnaire Tulip)

6.2. Informations générales sur Ethernet

Les noms de périphériques Ethernet sont `*eth0*', `*eth1*', `*eth2*' etc. La première carte détectée par le noyau devient `*eth0*' et le reste est nommé dans l'ordre de détection.

Une fois que vous avez compilé convenablement votre noyau pour supporter les cartes Ethernet, la configuration des cartes est aisée.

Typiquement vous faites ainsi (ce que la plupart des distributions font automatiquement pour vous, si vous les avez configurées pour supporter votre carte ethernet) :

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

La plupart des gestionnaires Ethernet furent développés par Donald Becker, Donald Becker

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

6.3. Utiliser plusieurs cartes Ethernet sur la même machine

6.3.1. Si le gestionnaire est sous forme de module (habituellement avec les nouvelles distributions)

Le module pourra normalement détecter toutes les cartes installées.

Les informations concernant la détection sont stockées dans le fichier :

/etc/conf.modules.

Supposons qu'un utilisateur possède 3 cartes NE2000, une à l'adresse 0x300, l'autre à 0x240, et la dernière à 0x220. Il faut ajouter les lignes suivantes au fichier /etc/conf.modules :

```
        alias eth0 ne
        alias eth1 ne
        alias eth2 ne
```

```
        options ne io=0x220,0x240,0x300
```

Ceci enjoint au programme *modprobe* de rechercher 3 cartes NE aux adresses spécifiées. De plus cela donne l'ordre dans lequel on doit les trouver et quel périphérique leur est assigné.

La plupart des modules ISA acceptent des valeurs d'E/S séparées par des virgules. Par exemple :

```
        alias eth0 3c501
        alias eth1 3c501
        options eth0 -o 3c501-0 io=0x280 irq=5
        options eth1 -o 3c501-1 io=0x300 irq=7
```

L'option –o permet d'assigner un nom unique à chaque module. La raison en est que vous ne pouvez charger deux copies du même module.

L'option irq= sert à spécifier l'IRQ matériel et l'option io= à spécifier les différents ports entrée–sortie.

Par défaut, le noyau Linux ne peut détecter qu'un seul dispositif Ethernet, et vous devez passer des commandes pour forcer la détection des autres cartes.

Pour apprendre à faire fonctionner vos cartes ethernet sous Linux, voyez le Ethernet–HOWTO.

Chapter 7. Informations relatives à l'IP

Cette section traite d'informations spécifiques à l'IP.

7.1. Options au niveau du noyau

Cette section fournit des informations concernant la mise au point des options IP dans le noyau au moment de l'amorçage. À titre d'exemples, de telles options peuvent être **ip_forward** ou **ip_bootp_agent**. Elles sont utilisées en affectant une valeur à un fichier situé dans le répertoire

```
/proc/sys/net/ipv4/
```

Le nom du fichier est le nom de la commande.

Par exemple, pour obtenir **ip_forward enabled** vous taperez

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

7.1.1. Liste des options IP générales.

- **ip_forward**

  Si **ip_forward** est réglé à 0, il est désactivé. Avec tout autre nombre, il est activé. Cette option est utilisée en conjonction avec des techniques telles que le routage entre interfaces avec le masquage IP..
- **ip_default_ttl**

  C'est la durée de vie d'un paquet IP. Par défaut, elle est de 64 ms.
- **ip_addrmask_agent**

– BOOLÉEN

Répond aux requêtes ICMP ADDRESS MASK. Par défaut TRUE (pour le routeur) et FALSE (pour l'hôte)

- **ip_bootp_agent**

  – BOOLÉEN

  Accepte des paquets ayant une adresse source du type 0.b.c.d et destiné à cet hôte, broadcast ou multicast. Sinon, ces paquets sont ignorés. FALSE par défaut.
- **ip_no_pmtu_disc**

  – BOOLÉEN

  Désactive la recherche du MTU du chemin. FALSE par défaut.
- **ip_fib_model**

  –NOMBRE ENTIER

  0 – (valeur par défaut) Modèle standard. Toutes les routes sont dans la classe MAIN

  1 – Les routes par défaut vont dans la classe DEFAULT. Ce mode devrait être très pratique pour les petits fournisseurs d'accès appliquant une politique de routage.

  2 – Modèle conforme à la RFC1812.

  Les routes interface sont dans la classe MAIN.

  Les routes gateway sont dans la classe DEFAULT.

7.2. EQL – égaliseur de charge à lignes multiples

Le nom du périphérique EQL est `eql'. Avec les sources standards du noyau vous ne pouvez avoir qu'un seul périphérique EQL par machine. EQL permet d'utiliser plusieurs lignes point à point telles que PPP, SLIP ou PLIP comme si c'était un seul lien logique de transport tcp/ip. C'est souvent moins cher d'utiliser plusieurs lignes à faible débit que d'avoir une ligne à haut débit.

Options de compilation du noyau :

```
Network device support --->
    [*] Network device support
    <*> EQL (serial line load balancing) support
```

Pour supporter ce mécanisme la machine à l'autre bout de la ligne doit également supporter EQL. Linux, Livingstone Portmasters et de nouveaux serveurs de ligne supportent des systèmes compatibles.

Pour configurer EQL vous avez besoin des outils eql, disponibles sur : metalab.unc.edu.

La configuration est plutôt directe. Vous commencez par configurer l'interface eql. C'est exactement comme un autre périphérique réseau. Vous configurez l'adresse IP et le mtu en utilissant l'outil *ifconfig* , comme ceci :

```
root# ifconfig eql 192.168.10.1 mtu 1006
```

Ensuite vous devez initialiser manuellement chacune des lignes que vous allez utiliser. Ce peut être toute combinaison de périphériques réseau point à point. La façon d'initialiser les connexions dépend du type de lien, voyez les paragraphes appropriés pour d'autres informations.

Enfin vous devez associer le lien série et le dispositif EQL, cela s'appelle `asservissement' (enslaving) et est réalisé avec la commande *eql_enslave* comme suit :

```
root# eql_enslave eql sl0 28800
root# eql_enslave eql ppp0 14400
```

Le paramètre `*estimated speed*' que vous fournissez à *eql_enslave* ne fait rien directement. Il est utilisé par le gestionnaire EQL pour déterminer comment les datagrammes vont se répartir sur ce périphérique, aussi vous pouvez régler l'équilibrage des lignes en jouant avec cette valeur.

Pour libérer une ligne d'un périphérique EQL, utilisez la commande *eql_emancipate* comme ci−dessous :

```
root# eql_emancipate eql sl0
```

Vous ajoutez le routage comme vous le feriez pour tout lien point à point, sauf que vos routes doivent se rapporter au dispositif *eql* plutôt qu'aux périphériques séries eux−mêmes. Ainsi vous devriez utiliser :

```
root# route add default eql
```

Le gestionnaire EQL fut développé par Simon Janes, *simon@ncm.com*.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.3. Enregistrement IP (IP Accounting) (pour Linux−2.0)

Les possibilités d'enregistrement IP du noyau Linux vous permettent de recueillir et d'analyser les données d'utilisation du réseau. Les données collectées comprennent le nombre de paquets et le nombre d'octets en cumul depuis la dernière remise à zéro. Vous avez à votre disposition une grande variété de réglages pour obtenir les données que vous désirez. Cette option a été enlevée du 2.1.102, car l'ancien dispositif pare−feu basé sur ipfwadm a été remplacé par ``ipfwchains''.

Options de compilation noyau :

```
Networking options  --->
    [*] IP: accounting
```

Après avoir compilé et installé le noyau vous devez utiliser la commande *ipfwadm* pour configurer l'enregistrement IP. Il y a différentes possibilités pour choisir les informations à enregistrer. J'ai pris un exemple simplifié qui pourrait vous être utile; lisez plutôt la page de manuel *ipfwadm* pour plus d'informations.

Scenario : Vous avez un réseau Ethernet qui est relié à l'Internet via une liaison PPP. Sur l'Ethernet vous avez une machine qui offre un grand nombre de services et vous voulez savoir quel trafic est engendré par le trafic ftp et ww, aussi bien que le trafic total tcp et udp.

Vous pouvez utiliser une commande qui ressemble à ceci, qui se présente comme un script shell :

```
        #!/bin/sh
        #
        # Donne les réglages d'enregistrement
        ipfwadm -A -f
```

```
        #
        # Met en place les raccourcis
        localnet=44.136.8.96/29
        any=0/0
        # Ajoute des réglages pour le segment Ethernet local
        ipfwadm -A in  -a -P tcp -D $localnet ftp-data
        ipfwadm -A out -a -P tcp -S $localnet ftp-data
        ipfwadm -A in  -a -P tcp -D $localnet www
        ipfwadm -A out -a -P tcp -S $localnet www
        ipfwadm -A in  -a -P tcp -D $localnet
        ipfwadm -A out -a -P tcp -S $localnet
        ipfwadm -A in  -a -P udp -D $localnet
        ipfwadm -A out -a -P udp -S $localnet
        #
        # Réglages par défaut
        ipfwadm -A in  -a -P tcp -D $any ftp-data
        ipfwadm -A out -a -P tcp -S $any ftp-data
        ipfwadm -A in  -a -P tcp -D $any www
        ipfwadm -A out -a -P tcp -S $any www
        ipfwadm -A in  -a -P tcp -D $any
        ipfwadm -A out -a -P tcp -S $any
        ipfwadm -A in  -a -P udp -D $any
        ipfwadm -A out -a -P udp -S $any
        #
        # Liste les réglages
        ipfwadm -A -l -n
        #
```

Les noms ``ftp−data'' et ``www'' se réfèrent aux lignes du fichier */etc/services*. La dernière commande liste chacune des règles d'enregistrement et affiche le total.

Il est important de noter, lorsque l'on analyse les enregistrement IP, que *les totaux sont incrémentés à chaque fois*, donc pour connaitre les différences vous devez exécuter les opérations mathématiques nécessaires. Par exemple si je veux savoir combien de données ne venaient pas de ftp, telnet, rlogin ou www je dois soustraire les totaux individuels correspondant à chaque port.

```
root# ipfwadm -A -l -n
IP accounting rules
 pkts bytes dir prot source               destination          ports
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 20
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            20 -> *
   10  1166 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 80
   10   572 out tcp  44.136.8.96/29       0.0.0.0/0            80 -> *
  252 10943 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> *
  231 18831 out tcp  44.136.8.96/29       0.0.0.0/0            * -> *
    0     0 in  udp  0.0.0.0/0            44.136.8.96/29       * -> *
    0     0 out udp  44.136.8.96/29       0.0.0.0/0            * -> *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 20
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0            20 -> *
   10  1166 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 80
   10   572 out tcp  0.0.0.0/0            0.0.0.0/0            80 -> *
  253 10983 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> *
  231 18831 out tcp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 in  udp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 out udp  0.0.0.0/0            0.0.0.0/0            * -> *
#
```

7.3.1. Enregistrement IP (IP Accounting) (pour Linux−2.2)

On accède au nouveau code d'enregistrement par des ``chaînes IP pare−feu''. Voir La page d'accueil des chaînes IP pour plus d'informations. Entre autres vous devrez utiliser *ipchains* au lieu de *ipfwadm* pour configurer vos filtres. (d'après *Documentations/Changes* dans les sources du dernier noyau).

7.4. IP Aliasing

Il y a des applications où être en mesure d'affecter plusieurs adresses IP à un seul périphérique réseau pourrait être utile. Certains fournisseurs d'accès à l'Internet utilise souvent cette possibilité pour fournir des offres www et ftp `à la carte' pour leurs clients. Vous pouvez vous référer au mini−HOWTO IP−Aliasing pour plus d'informations.

Options de compilation du noyau :

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
    <*> IP: aliasing support
```

Après avoir compilé et installé le noyau avec le support IP_Alias, la configuration est très simple. Les alias sont ajoutés aux périphériques réseau virtuels associés au périphérique réseau réel. Une simple convention de noms s'applique pour périphériques : *<nom de périphérique> : <numéro de périphérique virtuel>*, par ex. *eth0:0*, *ppp0:10* etc. Notez que le gestionnaire de périphérique ifname:number ne peut être configuré *qu'après* le réglage de l'interface principale.

Par exemple, supposons que vous ayez un réseau Ethernet avec simultanément deux sous−réseaux IP et que vous vouliez que votre machine ait un accès direct aux deux, vous pouvez faire quelque chose comme ceci :

```
        root# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
        root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0

        root# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
        root# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Pour supprimer un alias vous ajoutez simplement un `−' au bout de son nom et et vous faites aussi simplement que ça :

```
       root# ifconfig eth0:0- 0
```

Toutes les routes associées avec cet alias seront enlevées automatiquement.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.5. IP Pare−feu (Firewall) (pour Linux−2.0)

Le pare−feu IP et les publications le concernant sont traités de manière plus approfondie dans le Firewall−HOWTO. Le pare−feu IP vous permet de sécuriser votre machine contre les accès réseau non−autorisés en filtrant, ou acceptant, des datagrammes venant de, ou allant vers, des adresses IP de votre choix. Il y a différentes règles : le filtrage en entrée, le filtrage en sortie, et le filtrage en retransmission. Les règles en entrée s'appliquent aux datagrammes qui sont reçus par un dispositif réseau. Les règles en sortie s'appliquent aux datagrammes qui sont émis par un dispositif réseau. Les règles en retransmission s'appliquent aux datagrammes qui ne sont pas pour cette machine, c'est à dire les datagrammes qui seront reroutés.

Options de compilation noyau :

```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

La configuration du pare−feu IP est réalisée en utilisant la commande *ipfwadm*. Comme mentionné plus haut, la sécurité n'est pas ma spécialité, aussi, bien que je vous présente un exemple utilisable par vous−même, faites des recherches et mettez au point vos propres règlages si la sécurité est importante pour vous.

Utiliser votre machine Linux comme routeur et passerelle pare−feu pour protéger votre réseau local contre les accès non autorisés (venant de l'extérieur) est vraisemblablement l'utilisation la plus courante d'un pare−feu IP.

La configuration suivante est due à Arnt Gulbrandsen, <agulbra@troll.no>.

L'exemple décrit une configuration de pare−feu pour une machine Linux /pare−feu/routeur illustrée par ce diagramme :

```
-                                   -
 \                                  | 172.16.37.0
  \                                 |   /255.255.255.0
   \                 - - - - -      |
    |  172.16.174.30 | Linux |      |
NET =================|  f/w  |- - - |    ..37.19
    |    PPP         | router|      |   - - - -
   /                 - - - - -      |--| Mail  |
  /                                 |  | /DNS  |
 /                                  |   - - - -
-                                   -
```

Les commandes suivantes doivent être normalement placées dans un fichier *rc* de telle sorte qu'elles seront démarrées automatiquement à chaque redémarrage du système. Pour une sécurité maximum, elles devront être effectuées après la configuration des interfaces réseau, mais avant le montage de ces interfaces pour éviter que quelqu'un puisse se connecter pendant que la machine pare−feu redémarre.

```
        #!/bin/sh

        # Nettoie la table des règles de 'Forwarding'
        # Change le réglage par défaut en 'accept'
        #
        /sbin/ipfwadm -F -f
        /sbin/ipfwadm -F -p accept
        #
        # .. et pour 'Incoming'
        #
        /sbin/ipfwadm -I -f
        /sbin/ipfwadm -I -p accept

        # En premier, dévérouille l'interface PPP
        # J'aimerais bien utiliser '-a deny' au lieu de '-a reject -y' mais
        # il serait alors impossible d'établir des connexions également sur
        # cette interface. L'utilisation de -o fait en sorte que tous
        # les datagrammes rejetés sont enregistrés. Cela occupe de l'espace
```

```
# disque avec pour compensation la connaissance sur l'attaque due
# à une erreur de configuration.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Rejette certains types de paquets visiblement faux:
# Rien ne doit venir des adresses multicast/anycast/broadcast s
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# et aucune chose venant du réseau loopback ne doit être vu sur l'air
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# accepte les connexions entrantes SMTP et DNS, mais seules pour
# le serveur  de courrier et le serveur de noms
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS utilise UDP aussi bien que TCP, ce qui l'autorise donc quand
# le serveur de noms est interrogé
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# mais pas de "réponses" arrivant sur les ports dangereux tels que
# NFS et l'extension NFS de Larry McVoy. Si vous utilisez squid
# ajoutez son port ici.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
        -D 172.16.37.0/24 2049 2050

# les réponses aux autres ports utilisateurs sont autorisées
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535

# Rejette les connexions entrantres vers identd
# Nous utilisons 'reject' dans ce cas en sorte qu'il soit dit à l'hôte
# entrant de ne pas persévérer, sinon nous devrons attendre que
# identd s'arrête.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Accepte des connexions sur des services en provenance des réseaux
# 192.168.64 et 192.168.65, qui sont des amis de confiance.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
        -D 172.16.37.0/24 20:23

# accepte et laisse passer tout ce qui vient de l'intérieur
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# rejette la plupart des autres connexions TCP entrantes et les
# enregistre (ajoutez 1:1023 si ftp ne fonctionne pas)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# ... pour UDP également
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

De bonnes configurations pare−feu sont difficiles à faire. Cet exemple peut être un bon point de départ pour vous. La page de manuel *ipfwadm* vous aidera pour savoir comment utiliser cet outil. Si vous voulez configurer un pare−feu, demandez autour de vous et recueillez des avis venant de sources de confiance et prenez contact avec quelqu'un qui est à l'extérieur pour tester votre configuration et en vérifier la fiabilité.

7.5.1. Pare−feu IP (pour Linux−2.2)

On accède au nouveau code d'enregistrement par des ``chaînes pare−feu IP''. Voir La page d'accueil des chaînes IP pour plus d'informations. Entre autres vous devrez utiliser *ipchains/ au lieu de* ipfwadm/ pour configurer vos filtres. (D'après *Documentations/Changes* dans les sources du dernier noyau).

Nous sommes conscients du fait que ce n'est malheureusement plus d'actualité et nous oeuvrons actuellement pour que cette section soit plus à jour. Vous pouvez en espérer une en 1999.

7.6. Encapsulation IPIP

Pourquoi vouloir encapsuler des paquets IP dans d'autres paquets IP? Cela semble bizarre si vous n'avez jamais vu d'applications auparavant. Il y a deux endroits où c'est utilisé : le Mobile−IP et l'IP−Multicast. C'est dans un environnement qui est peut−être le plus largement utilisé et qui est le moins connu : le radio−amateurisme.

Options de compilation du noyau :

```
Networking options  --->
    [*] TCP/IP Networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP tunelling
```

Les périphériques IP tunnel s'appellent `*tunl0*', `*tunl1*', etc..

"Mais pourquoi ?" D'accord. D'accord. Les règles de routage classiques spécifient qu'un réseau IP comprend une adresse IP et un masque de réseau. Ceci fournit un ensemble d'adresses contiguës qui peuvent toutes être routées par l'intermédiaire d'une seule entrée de routage. Cela marche, mais signifie que vous ne pouvez utiliser une seule adresse uniquement lorsque vous êtes connecté à un point du réseau auquelle elle appartient. Dans la plupart des cas, il n'y a pas de problèmes, mais si vous êtes en mouvement alors vous ne pouvez pas rester connecté au même endroit tout le temps. L'encapsulation IP/IP (IP tunneling) vous permet de passer outre cette contrainte en permettant aux paquets destinés à votre adresse d'être enveloppés et redirigés vers une autre adresse. Si vous savez que vous allez opérer depuis un autre réseau IP pour quelques temps, vous pouvez régler une machine qui est chez vous pour accepter des paquets destinés à votre adresse IP et les rediriger vers l'adresse que vous allez utiliser provisoirement.

7.6.1. Une configuration de réseau avec tunneling.

```
 192.168.1.24                          192.168.2.24

     -                                      -
     |      ppp0 =           ppp0 =         |
     |  aaa.bbb.ccc.ddd  fff.ggg.hhh.iii    |
     |                                      |
     |   /- - -\                /- - -\     |
     |   |     |        //      |     |     |
     |- -|  A  |- - - // - - - -|  B  |     |
     |   |     |    //          |     |     |
     |   \- - -/                \- - -/     |
     |                                      |
```

```
    −                                    −
```

Ce diagramme montre une autre raison possible d'utiliser l'encapsulation IPIP : le réseau privé virtuel. Cet exemple présuppose que vous ayez deux machines chacune avec une seule connexion Internet. Chaque hôte a une seule adresse IP. Derrière chacune de ces machines se trouve des réseaux privés locaux configurés avec des adresses IP réservées. Supposez que vous vouliez permettre à chacun des hôtes du groupe A de se connecter à n'importe quel hôte du groupe B, comme s'ils étaient vraiment connectés à l'Internet via un routage réseau. L'encapsulation IPIP vous permettra de le faire. À noter que l'encapsulation ne vous permettra pas de faire en sorte que chacun des hôtes des réseaux A et B puissent parler à n'importe qui sur l'Internet, vous aurez toujours besoin de choses comme le masquage IP pour pouvoir le faire. L'encapsulation est normalement accomplie par une machine fonctionnant comme routeur.

Le routeur Linux `*A*' sera configuré comme suit :

```
        #!/bin/sh
        PATH=/sbin:/usr/sbin
        mask=255.255.255.0
        remotegw=fff.ggg.hhh.iii
        #
        # configuration éthernet
        ifconfig eth0 192.168.1.1 netmask $mask up
        route add −net 192.168.1.0 netmask $mask eth0
        #
        # ppp0 configuration (start ppp link, set default route)
        pppd
        route add default ppp0
        #
        # configuration du périphérique de tunneling
        ifconfig tunl0 192.168.1.1 up
        route add −net 192.168.2.0 netmask $mask gw $remotegw tunl0
```

Le routeur Linux `*B*' sera configuré comme suit :

```
        #!/bin/sh
        PATH=/sbin:/usr/sbin
        mask=255.255.255.0
        remotegw=aaa.bbb.ccc.ddd
        #
        # configuration éthernet
        ifconfig eth0 192.168.2.1 netmask $mask up
        route add −net 192.168.2.0 netmask $mask eth0
        #
        # ppp0 configuration (start ppp link, set default route)
        pppd
        route add default ppp0
        #
        # configuration du périphérique de tunneling
        ifconfig tunl0 192.168.2.1 up
        route add −net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

La commande :

```
root# route add −net 192.168.1.0 netmask $mask0 gw $remotegw tunl0
```

dit : `Envoyer tous les datagrammes destinés à *192.168.1.0/24* dans un paquet d'encapsulation ayant pour adresses de destination *aaa.bbb.ccc.ddd*'.

Notez que les configurations sont inversées à l'autre bout. Le périphérique tunnel utilise `gw' dans la commande route comme *destination* du paquet IP où se trouve le datagramme qu'il doit router. Cette machine doit savoir comment `désencapsuler' les paquets IPIP, c'est à dire qu'elle doit aussi être configurée comme périphérique tunnel.

7.6.2. Une configuration d'hôte pour l'encapsulation IPIP.

Ce n'est pas tout un réseau que vous aurez à router. Vous pouvez par exemple ne router qu'une seule adresse IP. Dans ce cas vous devrez configurer le périphérique *tunl* sur la machine `distante' avec sa propre adresse IP et à l'extrémité A n'utiliser qu'une route hôte (avec Proxy Arp) plutôt qu'une route réseau via le périphérique tunnel. Refaisons et modifions notre configuration de manière appropriée. Maintenant nous avons seulement l'hôte `B' qui veut agir comme s'il était à la fois connecté à l'Internet et également au réseau distant supporté par l'hôte `A' :

```
192.168.1/24

    -
    |      ppp0 =                ppp0 =
    |  aaa.bbb.ccc.ddd      fff.ggg.hhh.iii
    |
    |   /- - -\                 /- - -\
    |   |     |       //        |     |
    |- -|  A  |- - - //- - - - -|  B  |
    |   |     |     //          |     |
    |   \- - -/                 \- - -/
    |                         aussi: 192.168.1.12
    -
```

Le routeur Linux `A' sera configuré comme suit :

```
        #!/bin/sh
        PATH=/sbin:/usr/sbin
        mask=255.255.255.0
        remotegw=fff.ggg.hhh.iii
        #
        # configuration éthernet
        ifconfig eth0 192.168.1.1 netmask $mask up
        route add -net 192.168.1.0 netmask $mask eth0
        #
        # configuration de ppp0 (démarre le lien ppp link, règle la route par
        # défaut)
        pppd
        route add default ppp0
        #
        # configuration du périphérique de tunneling
        ifconfig tunl0 192.168.1.1 up
        route add -host 192.168.1.12 gw $remotegw tunl0
        #
        # Proxy ARP pour l'hôte distant
        arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

L'hôte Linux `B' sera configuré comme suit :

```
        #!/bin/sh
        PATH=/sbin:/usr/sbin
        mask=255.255.255.0
        remotegw=aaa.bbb.ccc.ddd
        #
        # configuration de ppp0 (démarre le lien ppp, règle la route par défaut)
        pppd
```

```
        route add default ppp0
        #
        # configuration du périphérique de tunelling
        ifconfig tunl0 192.168.1.12 up
        route add −net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

Ce type de configuration est vraiment typique d'une application IP−Mobile, où un simple hôte veut seulement se balader sur l'Internet et maintenir une adresse IP utilisable tout le temps. Référez−vous au paragraphe Mobile−IP pour avoir plus d'informations et savoir comment faire en pratique.

7.7. IP Masquerade

Un grand nombre de gens ont une simple connexion par téléphone pour aller sur l'Internet. Presque tout le monde ne se voit offrir qu'une seule adresse IP par le founisseur d'accès avec ce type de configuration. Ceci est normalement suffisant pour permettre un accès complet au réseau. IP Masquerade est une astuce intelligente qui vous permet d'avoir plusieurs machines utilisant une seule adresse IP, en faisant croire aux autres hôtes qu'il n'y a que la machine supportant la connexion (NdT : d'où le terme masquerade=duperie, mascarade). Il y a qu'une seule mise en garde, qui est que la fonction masquage ne travaille pratiquement que dans un seul sens : les hôtes masqués peuvent appeler mais ne peuvent accepter ou recevoir des connexions réseau de la part d'hôtes éloignés. Cela signifie que certains services réseau comme *talk* ne peuvent fonctionner et que d'autres, comme *ftp* doivent être configurés pour fonctionner en mode passif (PASV). Heureusement la plupart des services réseau comme *telnet*, World Wide Web et *irc* fonctionnent correctement.

Options de compilation du noyau :

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Normalement votre machine Linux supportant un lien SLIP ou PPP se comportera comme si elle était toute seule. De plus elle peut avoir un autre périphérique réseau configuré, par exemple une carte Ethernet, avec des adresses réseau réservée. Les hôtes masqués seront ceux du second réseau. Chacun de ces hôtes aura l'adresse IP du port Ethernet réglée comme passerelle ou routeur par défaut.

Une configuration typique ressemble à ceci :

```
−                                   −
 \                                  | 192.168.1.0
  \                                 |   /255.255.255.0
   \                 − − − − −      |
    |                | Linux | .1.1 |
NET =================| masq  |− − − |
    |    PPP/slip    | router|      |   − − − −
   /                 − − − − −      | −| hôte |
  /                                 |  |      |
 /                                  |   − − − −
−                                   −
```

7.7.1. Masquerading avec IPFWADM (Noyaux 2.0.x)

Les commandes adéquates pour cette configuration sont :

```
# Routage réseau pour éthernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Route par défaut pour le reste de l'internet.
route add default ppp0
#
# Fait en sorte que tous les hôtes du réseau 192.168.1/24  soient masqués.
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

7.7.2. Masquerading avec IPCHAINS

Cela ressemble à l'utilisation avec IPFWADM mais la structure de la commande change:

```
         # Routage réseau pour ethernet
         route add -net 192.168.1.0 netmask 255.255.255.0 eth0
         #
         # Route par défaut vers le reste de l'internet.
         route add default ppp0
         #
         # Fait en sorte que tous les hôtes sur le réseau 192.168.1/24 soient
         # masqués.
        ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

Vous pouvez obtenir plus d'informations sur IP Masquerade sur la Page d'informations sur l'IP Masquerade. Il existe également un document *très* détaillé qui est le ``IP−Masquerade−mini−HOWTO'' (qui donne en plus des renseignements pour configurer d'autres systèmes d'exploitation pour fonctionner avec un serveur de masquage linux).

Pour obtenir des informations concernant les applications de IP Masquerade, voyez la page Applications IPMASK.

7.8. IP Transparent Proxy

IP transparent proxy est un procédé qui vous permet de rediriger des serveurs ou des services destinés à une autre machine vers les services de votre machine. Typiquement c'est utile lorsque vous avez une machine Linux routeur et qui fournit aussi un serveur proxy. Vous redirigerez toutes les connexions à ce service distant vers le serveur proxy local.

Options de compilation du noyau :

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
        [*] Network firewalls
        ....
        [*] TCP/IP networking
        ....
        [*] IP: firewalling
        ....
        [*] IP: transparent proxy support (EXPERIMENTAL)
```

La configuration du dispositif transparent proxy est réalisé en utilisant la commande *ipfwadm*.

Par exemple :

```
ipfwadm −I −a accept −D 0/0 telnet −r 2323
```

Cet exemple fera en sorte que toutes les tentatives de connexion vers le port *telnet* (23), de n'importe quel hôte, seront redirigées vers le port 2323 de ce même hôte. Si vous utilisez un service sur ce port, vous pouvez rediriger des connexions telnet, les enregistrer ou exécuter tout ce qui bon vous semble.

Un exemple plus intéressant est la redirection de tout le trafic *http* au travers d'un cache local. Cependant, le protocole utlisé par les serveurs proxy diffère du protocole natif de http : quand un client se connecte à *www.server.com:80* et demande *chemin/page*, quand il se connecte au cache local il contacte *proxy.local.domain:8080* et recherche *www.server.com/chemin/page*.

Pour filtrer une demande *http* au travers du proxy local, vous devez pouvoir adapter le protocole en insérant un petit serveur, appelé *transproxy* (vous pouvez le trouver sur la toile). Vous pouvez choisir de faire tourner *transproxy* sur le port 8081, et exécuter la commande :

```
ipfwadm −I −a accept −D 0/0 80 −r 8081
```

Alors le programme *transproxy* recevra toutes les connexions devant aller vers des serveurs externes et les passera au proxy local après avoir corrigé les différences de protocole.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.9. IPv6

À peine pensez−vous avoir commencé à comprendre comment fonctionne le réseau IP, que les règles ont changé ! IPv6 est l'abbréviation de version 6 du `Protocole Internet' (version 6 de IP). Il fut développé initialement pour calmer les inquiétudes de la communauté Internet. Les utilisateurs s'inquiétaient d'une pénurie proche d'adresses IP pouvant être allouées. Les adresse IPv6 sont codées sur 16 octets (128 bits). IPv6 inclut un certain nombre d'autres changements, la plupart du temps des simplifications, qui rendront les réseaux IPv6 plus facilement gérables que les réseaux IPv4.

Linux a déjà une implémentation IPv6 qui marche, mais pas encore complètement, dans la série des noyaux *2.2.**.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.10. Sources de documentation pour IPv6 sous Linux

IPv6−HOWTO

IPv6 pour Linux.

Projet RPM Linux IPv6

FAQ/HOWTO IPv6

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.11. IP Mobile

Le terme "mobilité IP" décrit la possibilité qu'un hôte a de transférer sa connexion réseau d'un point de l'Internet vers un autre sans changer d'adresse IP ou sans perdre la connectivité. Normalement quand un hôte IP change de point de connexion, il change aussi d'adresse IP. La mobilité IP résoud ce problème en allouant une adresse IP fixe à l'hôte qui se déplace et en utilisant une encapsulation IP (tunneling) avec routage automatique pour s'assurer que les datagrammes qui lui sont destinés seront routés vers l'adresse effectivement utilisée à ce moment.

Un projet est en cours en vue de fournir un paquetage complet d'outils Linux pour la mobilité IP. L'état de ce projet et les outils peuvent être obtenus sur : Linux Mobile IP Home Page. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.12. Multicast

L'IP Multicast permet de router simultanément des datagrammes IP vers un certain nombre d'hôtes se trouvant sur des réseaux différents. Ce mécanisme est exploité pour fournir sur l'Internet des applications prenant de la bande passante, telles que les transmissions audio et video et autres nouvelles applications.

Options de compilation du noyau :

```
Networking options  --->
        [*] TCP/IP networking
        ....
        [*] IP: multicasting
```

Un ensemble d'outils et quelques modifications de la configuration réseau sont nécessaires. Pour plus d'informations sur le support multicast pour Linux, voyez le Multicast−HOWTO.html

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

7.13. Mise en forme du trafic − Changer la bande passante allouée

Le metteur en forme de trafic est un gestionnaire de périphérique qui crée de nouvelles interfaces; celles−ci sont limitées au point de vue trafic selon les réglages de l'utilisateur, et se connectent aux périphériques de réseau physiques pour la transmission réelle, et peuvent donc être utilisées comme route vers l'extérieur en vue de trafic réseau.

Le metteur en forme fut introduit sur Linux−2.1.15 et ensuite sur Linux−2.0.36 (il apparut dans le *2.0.36−pre−patch−2* distribué par Alan Cox, l'auteur du dispositif de mise en forme et le mainteneur de Linux−2.0).

Le metteur en forme de trafic ne peut être compilé qu'en tant que module, et se configure à l'aide du programme *shapecfg* avec des commandes comme :

```
        shapecfg attach shaper0 eth1
        shapecfg speed shaper0 64000
```

Ce metteur en forme de trafic ne peut contrôler que la bande passante du trafic sortant, car les paquets sont transmis par le metteur en forme si l'on se réfère aux tables de routage; ainsi, le fonctionnement suivant ``un routage par adresse de départ'' peut aider à limiter la bande passante totale d'hôtes spécifiques utilsant un routeur Linux.

Linux−2.2 possède déjà le support pour un tel routage et si vous en avez besoin pour Linux−2.0, voyez le patch de Mike McLagan, sur *ftp.invlogic.com*. Lisez le fichier *Documentation*networking/shaper.txt pour plus d'informations.

Si vous voulez faire (une tentative de) mise en forme pour les paquets entrants, essayez *rshaper−1.01* (ou plus récent), sur ftp.systemy.it.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 8. DHCP et DHCPD

DHCP est l'acronyme de «Dynamic Host Configuration Protocol» (Protocole de configuration dynamique d'un hôte). La création de DHCP a rendu la configuration du réseau avec plusieurs hôtes extrêmement simple. Au lieu de configurer chaque hôte séparément vous pouvez assigner tous les paramètres communs à l'ensemble des hôtes utilisant un serveur DHCP.

Chaque fois qu'un hôte démarre il diffuse un paquet sur le réseau. Ce paquet est un appel vers tous les serveurs DHCP situés sur le même segment pour configurer l'hôte.

DHCP est extrêmement utile pour assigner des choses comme l'adresse IP, le masque de réseau et la passerelle de chaque hôte.

8.1. Réglage d'un client DHCP pour les utilisateurs de LinuxConf

Sous linux, démarrez le programme linuxconf en tant que super−utilisateur. Ce programme est disponible avec toutes les versions de redhat et fonctionne sous X aussi bien qu'en mode console. Il fonctionne également avec les distributions Suse et Caldera.

```
Select Networking
----------------->Basic Host Information
----------------->Select Enable
----------------->Set Config Mode DHCP
```

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

8.2. Réglage d'un serveur DHCP sous Linux

Récupérez DHCPD s'il n'est pas déjà installé sur votre machine. Télécharger DHCPD

Note Brève: SOYEZ CERTAINS D'AVOIR L'OPTION MULTICAST INSTALLÉE DANS LE NOYAU.

Si vous n'avez pas de distribution binaire pour votre version de linux, vous devrez compiler DHCPD.

Éditez /etc/rc.d/rc.local pour prendre en compte l'ajout d'une route vers 255.255.255.255.

Extrait du fichier README de DHCPd :

Afin que dhcpd fonctionne correctement avec des clients difficiles (par exemple Windows 95), il doit être en mesure d'envoyer des paquets vers l'adresse 255.255.255.255. Malheureusement, Linux insiste pour changer l'adresse 255.255.255.255 en l'adresse de diffusion du sous−réseau local (ici, 192.5.5.223). Il en résulte une violation du protocole DHCP, et alors que de nombreux clients DHCP ne s'aperçoive pas de ce problème, d'autres (par ex, tous les clients DHCP Microsoft) le font. Les clients ayant ce problème apparaîtront comme ne voyant pas les messages DHCPOFFER venant du serveur.

Sous le compte root, tapez ce qui suit :

route add −host 255.255.255.255 dev eth0

Si le message suivant apparait :

255.255.255.255: Unknown host

Essayez d'ajouter la ligne suivante à votre fichier /etc/hosts :

255.255.255.255 dhcp

Puis réessayez :

route add −host dhcp dev eth0

8.2.1. Options de DHCPD

Maintenant vous devez configurer DHCPd. Pour cela vous devez créer ou éditer le fichier /etc/dhcpd.conf. Il existe une interface graphique pour la configuration de dhcpd sous linuxconf. On configure et règle ainsi DHCPD très facilement.

Si vous voulez le configurer à la main, suivez les instructions qui suivent. Je suggère de le faire au moins une fois à la main. Cela vous aidera pour faire des diagnostiques, ce qu'une interface graphique ne peut vous offrir. Malheureusement Microsoft n'y croit pas.

Le plus facile est d'assigner des adresses IP de manière aléatoire. Ci−dessous un exemple de fichier de configuration montrant le type de réglage.

```
# Exemple de /etc/dhcpd.conf
# (ajoutez vos commentaires ici)
default-lease-time 1200;
max-lease-time 9200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.org";
subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.10 192.168.1.100;
range 192.168.1.150 192.168.1.200;
}
```

Cela permet au serveur DHCP d'assigner au client une adresse IP comprise entre 192.168.1.10 et 192.168.1.100 ou bien 192.168.1.150 et 192.168.1.200.

Une adresse IP sera allouée pendant 1200 secondes si le client ne demande pas plus. Autrement l'allocation maximum permise sera 9200 secondes. Le serveur envoie les paramètres suivants au client :

Utilisez 255.255.255.0 comme masque de sous−réseau Utilisez 192.168.1.255 comme adresse de diffusion Utilisez 192.168.1.254 comme passerelle par défaut Utilisez 192.168.1.1 et 192.168.1.2 comme serveurs

DNS.

Si vous spécifiez un serveur WINS pour vos clients Windows, vous devez insérer l'option suivante dans le fichier dhcpd.conf.

option netbios−name−servers 192.168.1.1;

Vous pouvez aussi assigner des adresses IP spécifiques basées sur une adresse ethernet *MAC*, par exemple

```
  host haagen {
     hardware ethernet 08:00:2b:4c:59:23;
     fixed−address 192.168.1.222;
}
```

Cela assignera l'adresse IP 192.168.1.222 au client ayant une adresse ethernet *MAC* de 08:00:2b:4c:59:23.

8.2.2. Démarrage du serveur

Dans la plupart des cas, l'installation de DHCP ne crée pas le fichier dhcpd.leases. Dès lors, avant de démarrer le serveur, vous devez créer un fichier vide :

touch /var/state/dhcp/dhcpd.leases

Pour démarrer le serveur DHCP, tapez simplement (ou bien insérez−le dans les scripts de démarrage)

/usr/sbin/dhcpd

Cela démarre dhcpd sur le dispositif eth0. Si vous devez le démarrer sur un autre dispositif, tapez simplement

/usr/sbin/dhcpd eth1

Si vous voulez tester une configuration bizarre vous pouvez démarrer dhcpd en mode débogage. En tapant la commande suivante, vous pourrez voir exactement ce qui se passe sur le serveur.

/usr/sbin/dhcpd −d −f

Démarrez un client et jetez un coup d'oeil sur la console du serveur. Vous verrez apparaître un grand nombre de messages de débogage.

C'est fini

Chapter 9. Routage avancé avec Linux−2.2

Le noyau 2.2 a accru les possibilités de routage de Linux de manière certaine. Malheureusement la documentation pour exploiter ces nouvelles possibilités est presque impossible à trouver, même si elle existe.

J'y ai passé un peu de temps et ai été en mesure de faire un petit quelque chose. J'en mettrai plus quand j'aurai le temps et l'aide nécessaire pour comprendre tout ce qui a été dit.

Dans les versions du noyau 2.0 et antérieures, Linux utilise la commande *route* standard pour positionner les routes dans une seule table de routage. Si vous aviez tapé *netstat −rn* à l'invite vous auriez pu voir un exemple.

Dans les noyaux récents (2.1 et au−delà) vous avez une autre option. Cette option est autorisée et vous permet d'avoir plusieurs tables de routage. Les nouvelles règles donnent beaucoup plus de souplesse sur la façon de manipuler les paquets. Vous pouvez choisir entre des routes basées non seulement sur l'adresse de destination, mais aussi l'adresse de départ, TOS, ou un périphérique de réception.

9.1. Les bases

Afficher la table de routage :

ip route

Maintenant sur ma machine cela donne la sortie suivante :

```
207.149.43.62 dev eth0  scope link
207.149.43.0/24 dev eth0  proto kernel  scope link  src 207.149.43.62
default via 207.149.43.1 dev eth0
```

La première ligne :

207.149.43.62 dev eth0 scope link est la route vers l'interface

La deuxième :

207.149.43.0/24 dev eth0 proto kernel scope link src 207.149.43.62 est la route qui dit *tout ce qui va vers 207.149.43.0 doit aller vers 207.149.43.62*.

La troisième :

default via 207.149.43.1 dev eth0 est la route par défaut.

9.1.1. Utiliser les informations

Maintenant que nous avons vu une table de routage de base, voyons comment l'utiliser. Tout d'abord lisez name="the Policy routing text." Si vous êtes embêtés, ne vous en faites pas −− c'est un texte confus. Il vous donne tout ce que les nouvelles tables de routage peuvent faire.

9.2. Ajouter une route avec les nouveaux outils ip

Dans la section précédente, nous avons mentionné comment afficher la table de routage et comment comprendre les bases. Impeccable, la sortie ressemble de très près à la syntaxe que nous devons utiliser pour mettre en place la table de routage correspondant exactement à nos besoins.

```
ip route add 207.149.43.62 dev eth0  scope link
ip route add 207.149.43.0/24 dev eth0 proto kernel scope link src 207.149.43.62
ip route add 127.0.0.0/8 dev lo  scope link
ip route add default via 207.149.43.1 dev eth0
```

Comme vous pouvez le constater, les entrées et sorties sont presque les mêmes, sauf le *ip route add* au début de chaque ligne.

Note: Je suis conscient que la documentation sur le routage avec les noyaux 2.2 fait cruellement défaut. Dans les faits, je pense que TOUT LE MONDE en est conscient. Si vous avez une petite expérience, contactez−nous s'il vous plaît à poet@linuxports.com nous aimerions obtenir les informations que vous avez pour nous aider à étoffer notre documentation!

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

9.3. Utiliser NAT avec le noyau 2.2

Le système de traduction d'adresse réseau (NAT: Network Address Translation) ressemble plutôt au « grand frère » standardisé du système de masquage IP de Linux. Il est décrit en détail dans la RFC−1631 sur votre archive RFC la plus proche. NAT fournit des possibiltés que IP Masquerade ne sait pas faire, ce qui le rend plus apte à une utilsation de routeur pare−feu pour un réseau d'entreprise et des installations de plus grande dimension.

Une implémentation alpha de NAT pour le noyau 2.0.29 de Linux a été développée par Michael.Hasenstein, *Michael.Hasenstein@informatik.tu−chemnitz.de*. La documentation et l'implémentation de Michael se trouve sur :

la page web sur l'adressage réseau sous Linux

L'empilement TCP/IP du noyau 2.2, qui a été grandement amélioré, inclut les fonctionnalités de NAT. Ce système semble rendre obsolète le travail de Michael Hasenstein (Michael@informatik.tu−chemnitz.de).

Pour le rendre opérationnel vous devez activer dans le noyau CONFIG_IP_ADVANCED_ROUTER, CONFIG_IP_MULTIPLE_TABLES (pour le système de routage) et CONFIG_IP_ROUTE_NAT (pour un NAT rapide). De plus, si vous voulez utiliser un réglage plus fin de NAT, vous devez activer le pare−feu (CONFIG_IP_FIREWALL) et CONFIG_IP_ROUTE_FWMARK. Pour faire fonctionner effectivement ces possibilités incluses dans le noyau, vous aurez besoin du programme «ip» de Alexey Kuznyetsov récupéré sur ftp://ftp.inr.ac.ru/ip−routing/.

Datagrammes NAT entrants

Maintenant pour traduire les adresses des datagrammes entrants, on utilise la commande suivante :

```
        ip route add nat <ext-addr>[/<masklen>] via <int-addr>
```

Ceci fait qu'un paquet entrant destiné à l'adresse "ext−addr" (l'adresse visible sur l'internet extérieur) aura son champ d'adresse converti en "int−addr" (l'adresse de votre réseau interne, derrière votre passerelle ou pare−feu). Le paquet est alors routé suivant la table de routage locale. Vous pouvez traduire soit une adresse hôte unique, soit des blocs complets. *Exemples:*

```
      ip route add nat 195.113.148.34 via 192.168.0.2
      ip route add nat 195.113.148.32/27 via 192.168.0.0
```

La première commande rend l'adresse interne 192.168.0.2 accessible en tant que 195.113.148.34. Le second exemple montre une réallocation du bloc 192.168.0.0−31 en 195.113.148.32−63. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 10. Les commandes IP pour les noyaux 2.2 (travail en cours)

10.1. ip

Si vous avez les outils iproute2 déjà installés, exécutez la commande ip, ce qui vous permettra d'afficher la syntaxe de base.

```
[root@jd Net4]# ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
where OBJECT := { link | addr | route | rule | neigh | tunnel |
                  maddr | mroute | monitor }
      OPTIONS := { −V[ersion] | −s[tatistics] | −r[esolve] |
                   −f[amily] { inet | ipv6 | dnet | link } | −o[neline] }
```

Il y a plusieurs options disponibles :

−V, −Version donne la version de l'utilitaire ip que vous employez puis vous rend la main.

−s, −stats, −statistics donne plus d'informations concernant le périphérique spécifié. Vous pouvez mentionner plusieurs fois cette option pour afficher plus d'informations.

−f, family suivi d'un nom identifiant la famille de protocole tel que : inet, inet6 ou link spécifie la famille de protocole à utiliser, inet désignant le standard IPv4 (le standard internet actuel), inet6 désignant IPv6 (révolutionnaire, un standard internet qui ne sera jamais implanté), et link (un lien physique). Si vous ne donnez pas d'options, la famille de protocole est devinée et s'il n'y a pas assez d'informations, ip reviendra aux réglages par défaut.

−o, −oneline indique la sortie de chaque enregistrement de périphérique en une seule ligne.

−r, −resolve utilise le résolveur du système (par exemple DNS), pour imprimer les noms réels associés aux adresses IP.

OBJECT C'est l'objet/périphérique que l'on veut gérer ou bien sur lequel on veut obtenir des informations. Les types de périphériques qui sont compris par l'implémentation actuelle sont :

- link −− Le périphérique réseau, par exemple eth0 ou ppp0
- address −− L'adresse IP (IP ou IPv6) du périphérique spécifié
- neigh −− L'entrée de cache ARP ou NDISC
- route −− L'entrée de la table de routage
- rule −− Les règles de la base de données de la politique de routage
- maddress −− L'adresse de multidiffusion
- mroute −− L'entrée de cache de la route de multidiffusion
- tunnel −− Faire ou non de l'encapsulation IP

Le nombre d'options possibles avec chaque type d'objet est fonction de la nature de l'action à entreprendre. Comme règle de base, il est possible d'*ajouter*, de *supprimer*, ou de montrer le ou les objets, mais parmi ceux−ci tous ne permettront pas d'utiliser des commandes supplémentaires. Bien sûr, une commande d'aide est disponible pour chaque objet et lors de son utilisation, celle−ci donnera une liste des conventions de syntaxe disponibles pour l'objet en question.

Si vous ne spécifiez pas de commande, c'est celle par défaut qui sera exécutée. Celle−ci donne la liste des objets, ou bien, si ce n'est pas possible, vous obtiendrez une aide de base.

ARGUMENTS est la liste des arguments qui peuvent être donnés lors de l'exécution de la commande. Le nombre d'arguments dépend de la commande et de l'objet. Il existe deux types d'arguments :

Les drapeaux consistant en un mot−clé suivi d'une valeur. Pour la commodité, chaque argument possède quelques valeurs par défaut, qui peuvent être omises pour une utilisation plus facile. Par exemple le paramètre *dev>* est pris par défaut pour une commande telle que *ip link.*

Les erreurs... rendons grâce aux codeurs intelligents Toutes les actions induites par les commandes sont dynamiques. Si la syntaxe est incorrecte, il n'y aura pas de changement dans la configuration du système.

Comme toujours, il existe une exception : la commande *ip link* , utilisée pour changer certains paramètres d'un périphérique.

Il est difficile de donner la liste de tous les messages d'erreur (en particulier les erreurs de syntaxe), mais normalement leur signification est explicite suivant le contexte de la commande. Les erreurs les plus fréquentes sont : 1. Le réseau n'est pas configuré dans le noyau. Le message est : Cannot open netlink socket: Invalid value (ne peut ouvrir la socket : valeur incorrecte).

2. RTNETLINK n'est pas configuré dans le noyau. Dans ce cas on obtiendra l'un des messages suivants, selon la commande : Cannot talk to rtnetlink: Connection refused (ne peut dialoguer avec rtnetlink: connexion refusée) Cannot send dump request: Connection refused (ne peut envoyer une demande de vidage (dump): connexion refusée).

3. L'option CONFIG_IP_MULTIPLE_TABLES n'a pas été choisie lors de la configuration du noyau. Dans ce cas toute tentative d'utilisation de la commande ip échouera, par exemple :

jd@home $ ip rule list RTNETLINK error: Invalid argument dump terminated (erreur de règle ip dans la liste RTNETLINK: argument incorrect, vidage (dump) terminé).

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 11. Utilisation du matériel courant pour PC

11.1. RNIS

Le Réseau Numérique à Intégration de Service (RNIS) (en anglais ISDN: Integrated Services Digital Network) est une série de normes donnant les spécifications d'un réseau de données numériques à usage général. Un `appel' RNIS crée un service synchrone de données point à point vers la destination. RNIS est généralement délivré sur une ligne à haut débit divisée en un certain nombre de canaux discrets. Il y a deux types de canaux, les `canaux B' qui transportent effectivement les données utilisateurs, et un canal unique appelé `canal D' qui est utilisé pour envoyer les informations de contrôle pendant l'échange RNIS en vue d'établir des appels et autres fonctions. En Australie, par exemple, RNIS peut être fourni sur une liaison 2 Mps qui est divisée en 30 canaux B discrets de 64 kps et un canal D de 64 kps. N'importe quel nombre de canaux peuvent être utilisés en même temps et ceci dans toutes les combinaisons possibles. Vous pouvez par exemple établir 30 appels différents de 64 kps vers 30 destinations différentes, ou bien 15 appels de 128 kps chacun vers 15 destinations différentes (2 canaux utilisés par appel), ou seulement un petit nombre d'appels, le reste étant inactif. Un canal peut être utilisé pour des appels entrant ou sortant. Le but initial de RNIS était de permettre aux sociétés de Télécommunications de fournir un seul service de données pouvant délivrer soit le téléphone (avec une voix numérisée) ou bien des services de données vers votre domicile ou votre bureau sans avoir à effectuer de changements pour obtenir une configuration spéciale.

Il y a plusieurs façons de connecter votre ordinateur à un service RNIS. L'une consiste à utiliser un dispositif appelé `Adaptateur de Terminal' qui se branche sur l'unité de terminal réseau que votre opérateur de télécommunications a installé au moment de l'obtention de votre service RNIS, et qui présente des interfaces séries. L'une de ces interfaces est utilisée pour entrer les commandes pour établir les appels et la configuration, et les autres sont reliées aux périphériques réseau qui utiliseront les circuits de données quand la connexion sera faite. Linux peut travailler avec ce type de configuration sans modification, vous devez juste traiter le port de l'adaptateur de terminal comme vous traitez tout périphérique série. Une autre façon, qui est la raison d'être pour le support RNIS dans le noyau, vous permet d'installer une carte RNIS dans votre machine Linux et le logiciel Linux prend en charge les protocoles et fait les appels lui−même.

Options de compilation noyau :

```
ISDN subsystem  --->
        <<*> ISDN support
        [ ] Support synchronous PPP
        [ ] Support audio via ISDN
        < > ICN 2B and 4B support
        < > PCBIT-D support
        < > Teles/NICCY1016PC/Creatix support
```

L'implémentation Linux de RNIS supporte différents types de cartes internes RNIS. Il y a celles énumérées dans les options de configuration noyau :

- ICN 2B and 4B
- Octal PCBIT−D
- Teles ISDN−cards et compatibles

Certaines de ces cartes ont besoin de logiciels devant être téléchargés pour les rendre opérationnelles. Il y a un utilitaire séparé pour le faire.

Tous les détails pour configurer le support RNIS Linux se trouvent dans le répertoire */usr/src/linux/Documentation/isdn/* et un document FAQ dédié à *isdn4linux* est disponible sur www.lrz−muenchen.de (vous pouvez cliquer sur le drapeau anglais pour obtenir la version anglaise).

Note au sujet de PPP. L'ensemble des protocoles PPP peut travailler sur des lignes série synchrone ou asynchrone. Le démon PPP `*pppd*' couramment distribué pour Linux ne supporte que le mode asynchrone. Si vous désirez utiliser les protocoles PPP avec votre service RNIS vous aurez besoin d'une version spéciale. Les détails pour la trouver se trouvent dans la documentation mentionnée ci−dessus.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

11.2. PLIP pour Linux−2.0

Les noms de périphériques PLIP sont `*plip0*', `*plip1*', `*plip2*'.

Options de compilation du noyau :

```
Networking options  ---i>
    <*> PLIP (parallel port) support
```

PLIP (Parallel Line IP) est, comme SLIP, utilisé pour fournir une connexion réseau *point à point* entre deux machines, sauf qu'il est conçu pour utiliser les ports parallèles de votre machine au lieu des ports séries. Parce qu'il est possible de transmettre plus d'un bit en même temps avec un port parallèle, on peut atteindre des plus hautes vitesses avec l'interface *PLIP* qu'avec une sortie série standard (un schéma de câblage est donné plus loin dans ce document). De plus, même le plus simple des ports parallèles, le port imprimante, peut être utilisé, au lieu d'acheter un UART 16550AFN relativement cher pour vos ports séries. PLIP utilise beaucoup de CPU en comparaison d'une liaison série et ce n'est sûrement pas un bon choix si vous avez la possibilité d'avoir des cartes éthernet pas chères, mais ça fonctionne lorsque rien d'autre n'est disponible, et ça fonctionne très bien.

Les gestionnaires PLIP entrent en compétition avec les autres gestionnaires du matériel branché sur le port parallèle. Si vous voulez utiliser les deux, vous devez alors les compiler en tant que modules pour pouvoir choisir quel port vous voulez utiliser pour PLIP et quel port pour l'imprimante. Voyez le document « Modules−mini−HOWTO » pour plus d'informations sur la configuration des modules noyau.

Attention, notez que certains portables utilisent des circuits qui ne peuvent pas fonctionner avec PLIP car ils n'autorisent pas certaines combinaisons dont PLIP a besoin et que les imprimantes n'utilisent pas.

L'interface Linux *PLIP* est compatible avec le *Gestionnaire PLIP Crynwyr Packet* et ceci signifie que vous pouvez connecter votre machine Linux avec une machine DOS tournant avec n'importe quel logiciel TCP/IP via *PLIP*.

Dans la série des noyaux 2.0.* les gestionnaires de périphérique PLIP sont affectés aux ports e/s et IRQ comme suit :

```
device  i/o addr    IRQ
- - -   - - - -     - - -
plip0   0x3BC        5
plip1   0x378        7
plip2   0x278        2
```

Si vos ports parallèles ne correspondent pas aux combinaisons précédentes alors vous pouvez changer les IRQ en utilisant la commande *ifconfig* avec le paramètre `*irq*'. N'oubliez pas de valider les IRQ pour vos ports imprimantes dans votre ROM BIOS s'il supporte cette option. Un autre moyen consiste à spécifier les options ``*io=*'' et ``*irq=*'' sur la ligne de commande de *insmod*, si vous utilisez les modules. Par exemple :

```
root# insmod plip.o io=0x288 irq=5
```

Le fonctionnement de PLIP est contrôlé par deux temporisations de dépassement de temps, dont les valeurs par défaut devraient convenir la plupart du temps. Vous devrez peut−être les augmenter si vous avez un ordinateur particulièrement lent, auquel cas les valeurs devant être augmentées se trouvent sur l'*autre* ordinateur. Il existe un programme appelé *plipconfig* qui permet d'effectuer ces réglages sans recompiler le noyau. Il est fourni avec de nombreuses distributions Linux.

Pour configurer une interface *plip*, vous devez invoquer les commandes suivantes (ou les *ajouter* à vos scripts d'initialisation) :

```
 root# /sbin/ifconfig plip1 localplip pointopoint remoteplip
 root# /sbin/route add remoteplip plip1
```

Dans ce cas, le port utilisé est celui qui a l'adresse 0x378 ; *localplip* et *remoteplip* sont les adresses IP utilisées sur le câble PLIP. Je les mets personnellement dans la base de données */etc/host* :

```
 # entrées plip
 192.168.3.1 localplip
 192.168.3.2 remoteplip
```

Le paramètre *pointopoint* a la même signification que pour SLIP, c'est−à−dire qu'il spécifie l'adresse de la machine à l'autre bout de la liaison.

Dans la plupart des cas vous pouvez traiter l'interface *PLIP* comme si elle était une interface *SLIP*, sauf que ni *dip* ni *slattach* ne doivent, ou ne peuvent, être utilisés.

Plus d'information sur PLIP peut être obtenu avec le document ``PLIP−mini−HOWTO''.

11.2.1. PLIP pour Linux−2.2

Durant le développement des versions 2.1 du noyau, le support concernant les ports parallèles s'est amélioré.

Options de compilation du noyau :

```
General setup --->
    [*] Parallel port support
```

```
Network device support --->
    <*> PLIP (parallel port) support
```

Le nouveau code concernant PLIP se comporte comme l'ancien (on utilise les mêmes commandes *ifconfig* et *route* comme dans le paragraphe précédent), mais l'initialisation du système est différente en raison du support port parallèle amélioré.

Le ``premier'' périphérique PLIP est toujours appelé ``plip0'', premier signifiant celui qui est détecté en premier par le système, comme pour les périphériques Ethernet. Le port parallèle utilisé de fait est l'un de ceux qui sont disponibles, comme indiqué dans */proc/parport*. Par exemple, si vous n'avez qu'un seul port parallèle, vous n'aurez qu'un seul répertoire appelé */proc/parport/0*.

Si votre noyau ne détecte pas l'IRQ utilisée par votre port parallèle, ``*insmod plip*'' échouera ; dans ce cas, vous écrivez juste le chiffre adéquat dans */proc/parport/0/irq* et vous invoquez de nouveau *insmod*.

Une information complète sur la gestion des ports parallèles est disponible dans le fichier *Documentation/parport.txt*, qui se trouve dans les sources du noyau.

11.3. PPP

En raison de la nature de PPP, sa taille, sa complexité, et sa souplesse son propre HOWTO a été créé. Le PPP−HOWTO est toujours un document du LDP mais son site officiel est sur le site LinuxPorts à la section PPP. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

11.4. Client SLIP (Antique)

Les fichiers de périphériques SLIP sont nommés `*sl0*', `*sl1*', etc. Le premier configuré étant `*0*' et les autres s'incrémentant au fur et à mesure de leur configuration.

Options de compilation du noyau :

```
Network device support  ---i>
    [*] Network device support
    <*> SLIP (serial line) support
    [ ]  CSLIP compressed headers
    [ ]  Keepalive and linefill
    [ ]  Six bit SLIP encapsulation
```

SLIP (Serial Line Internet Protocol) vous permet d'utiliser TCP/IP avec une ligne série, ce peut être un téléphone et un modem, ou tout autre ligne dédiée. Bien sûr pour utiliser SLIP vous devez avoir accès à un *serveur SLIP* dans votre entourage. De nombreuses universités et de sociétés fournissent des accès SLIP de par le monde.

SLIP utilise les ports séries de votre machine pour transporter les datagrammes IP. Pour cela il doit prendre le contrôle du périphérique série. Les noms de périphériques SLIP sont *sl0*, *sl1*, etc. Comment ceux−ci correspondent avec vos périphériques série ? Le code réseau utilise ce que l'on nomme un appel *ioctl* (i/o control) pour transformer les périphériques série en périphériques SLIP. Il y a deux programmes qui peuvent faire cela, ce sont *dip* et *slattach*.

11.4.1. dip

dip (Dialup IP) est un programme élégant capable de régler la vitesse du dispositif série, de demander à votre modem d'appeler l'autre extrémité de la ligne, de vous connecter automatiquement au serveur distant, de chercher des messages qui vous ont été envoyés par le serveur et d'en extraire des informations telles que votre adresse IP et de faire le *ioctl* nécessaire pour basculer votre port série en mode SLIP. *dip* est très flexible quant à l'utilisation de scripts et grâce à ceci vous pouvez automatiser vos procédures de connexion.

On peut le trouver sur : metalab.unc.edu.

Pour l'installer faites :

```
user% tar xvfz dip337o-uri.tgz
user% cd dip-3.3.7o
user% vi Makefile
root# make install
```

Le fichier *Makefile* suppose l'existence d'un groupe nommé *uucp*, mais vous pouvez le changer en *dip* ou *SLIP*, selon votre configuration.

11.4.2. slattach

slattach au contraire de *dip* est un programme très simple, très facile à utiliser, mais qui n'a pas la sophistication de *dip*. Il n'a pas la possiblité d'accepter des scripts, tout ce qu'il fait étant de configurer votre périphérique série en périphérique SLIP. Il suppose que vous avez toutes les informations nécessaires et que la liaison série est établie avant de l'invoquer. *slattach* est idéal quand vous avez une liaison permanente avec votre serveur, comme un câble physique ou une ligne dédiée.

11.4.3. Quand utiliser quoi ?

Vous devriez utiliser *dip* lorsque votre liaison vers la machine qui est votre serveur SLIP est un modem, ou tout autre lien intermittent. Vous devriez utiliser *slattach* quand vous avez une ligne dédiée, peut−être un câble, entre votre machine et le serveur et qu'il n'y a pas d'action spéciale nécessaire pour garder la ligne en activité. Voir la section `Connexion SLIP permanente' pour plus de détails.

Configurer SLIP est analogue à la configuration d'une interface Ethernet (voir la section `Configurer un périphérique Ethernet' ci−dessus). Cependant, il existe quelques différences.

Tout d'abord, les liens SLIP ne sont pas des réseaux Ethernet en ce sens qu'il n'y a que deux hôtes sur le réseau, un à chaque extrémité de la liaison. À la différence de l'Ethernet qui est disponible dès que vous êtes câblé, avec SLIP, en fonction du type de lien que vous avez, vous serez amené à initialiser votre connexion réseau d'une manière spéciale.

Si vous utilisez *dip*, alors cela ne sera pas fait au moment du démarrage de la machine, mais plus tard, quand vous serez prêt à utiliser la liaison. Il est possible d'automatiser la procédure. Si vous utilisez *slattach* vous voudrez probablement ajouter une section dans votre fichier *rc.inet1*. Ceci sera décrit bientôt.

Il y a deux types principaux de serveurs SLIP : serveurs avec adressage IP dynamique et serveurs avec adressage IP statique. Presque tous les serveurs SLIP vous demanderont à la connexion d'utiliser un nom d'utilisateur et un mot de passe quand vous composez le numéro. *dip* peut prendre en charge la connexion automatiquement.

11.4.4. Serveur SLIP statique avec une ligne téléphonique et DIP

Le serveur SLIP statique est celui qui vous fournit une adresse IP qui reste exclusivement la vôtre. À chaque fois que vous vous connectez à ce serveur, vous configurez votre port SLIP avec cette adresse. Le serveur SLIP statique répond à votre appel par modem, vous demande probablement un nom d'utilisateur et un mot de passe, et ensuite dirige tous les datagrammes destinés à votre adresse au travers de cette connexion. Si vous avez un serveur statique, alors vous mettez des entrées pour votre nom d'hôte et votre adresse IP (puisque vous savez ce qu'elle sera) dans votre fichier */etc/hosts*. Vous devez aussi configurer d'autres fichiers comme : *rc.inet2*, *host.conf*, *resolv.conf*, */etc/HOSTNAME* et *rc.local*. N'oubliez pas qu'en configurant *rc.inet1*, vous n'avez pas besoin d'ajouter de commandes spéciales pendant la connexion SLIP puisque c'est *dip* qui fait tout le dur labeur à votre place en configurant votre interface. Vous avez besoin de donner à *dip* les informations adéquates et il configure l'interface pour vous après avoir demandé au modem d'établir l'appel et de vous connecter au serveur.

Si votre serveur SLIP fonctionne comme cela alors vous pouvez directement aller à la section `Utiliser Dip' pour apprendre à configurer *dip* convenablement.

11.4.5. Serveur SLIP dynamique avec une ligne téléphonique et DIP

Le serveur SLIP *dynamique* vous alloue une adresse IP de manière aléatoire, à partir d'un groupe d'adresses, à chaque fois que vous vous connectez. Cela signifie qu'il n'y a aucune garantie d'avoir la même adresse à chaque fois, et que celle−ci peut être utilisée par quelqu'un d'autre après la déconnexion. L'administrateur réseau qui a configuré le serveur SLIP a assigné un groupe d'adresses que le serveur SLIP peut utiliser quand il reçoit un appel entrant. Il prend alors la première adresse inutilisée, guide l'appelant au travers du processus de connexion et envoie un message de bienvenue contenant l'adresse IP qu'il a allouée et continue d'utiliser cette adresse tout le temps de l'appel.

Configurer ce type de serveur revient à configurer un serveur statique, sauf que vous devez ajouter une étape pour obtenir l'adresse IP allouée par le serveur puis configurer le périphérique SLIP avec celle−ci.

Encore une fois, *dip* fait le sale boulot et les nouvelles versions sont suffisamment élégantes pour non seulement établir la connexion, mais aussi pour lire l'adresse IP inscrite dans le message de bienvenue et la stocker de telle sorte que vous puissiez configurer votre périphérique SLIP avec.

Si votre serveur SLIP fonctionne ainsi, alors vous pouvez aller à la section `Utiliser DIP' pour savoir comment configurer *dip* de manière adéquate.

11.4.6. Utiliser DIP

Comme expliqué plus haut, *dip* est un programme puissant qui simplifie et automatise le processus de composition d'un numéro vers un serveur SLIP, se connecte dessus, démarre la connexion et configure les périphériques SLIP à l'aide des commandes *ifconfig* et *route* appropriées.

Essentiellement, pour utiliser *dip* vous écrivez un `script dip' qui est tout simplement une liste de commandes que *dip* comprend et qui lui dit comment réaliser chacune des actions que vous voulez qu'il fasse. Voyez le fichier *sample.dip* fourni avec *dip* pour avoir une idée de la manière dont il travaille. *dip* est vraiment un programme puissant, avec beaucoup d'options. Au lieu de regarder chacune d'elles, il vaut mieux jeter un coup d'oeil dans la page de manuel, le fichier README et les fichiers d'exemple qui sont fournis avec votre version de *dip*.

Vous pouvez noter que le script *sample.dip* suppose que vous utilisez un serveur SLIP statique, aussi vous connaissez votre adresse IP à l'avance. Pour les serveurs SLIP dynamiques, les nouvelles versions de *dip* incluent une commande que vous pouvez utiliser pour lire et configurer automatiquement votre périphérique SLIP avec l'adresse IP donnée par le serveur dynamique. L'exemple suivant est une version modifiée du fichier *sample.dip* fourni avec *dip337j−uri.tgz* et qui est probablement un bon point de départ pour vous. Vous pouvez le sauvegarder sous le nom de */etc/dipscript* et l'éditer pour l'adapter à votre configuration :

```
#
# sample.dip    Programme de support pour connexion IP.
#
#       Ce programme (devrait montrer) montre comment utiliser DIP
#       Il devrait fonctionner avec des serveurs dynamiques de type Annex,
```

```
#       et si vous utilsez un serveur avec adresse statique utilsez alors le
#       fichier sample.dip livré avec le paquetage dip337-uri.tgz.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Auteur:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Après, postionner l'adresse et le nom de l'hôte distant.
# Ma machine s'appelle 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Positionne le masque de réseau sur sl0 à 255.255.255.0
netmask 255.255.255.0
# Règle le port série et la vitesse.
port cua02
speed 38400

# Reset le modem et la ligne de terminal.
# Cela semble poser problème à certains!
reset

# Notez! Valeurs "standards" prédéfinies de "errlevel":
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# Vous pouvez les changer en faisant un grep dans *.c avec "addchat()"...

# On se prépare pour numéroter.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# Nous sommes connectés. Nous nous enregistrons sur le système.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n
loggedin:

# Maintenant nous sommes enregistrés.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Demande au serveur de basculer en mode SLIP
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Obtenir et ajuster notre adresse IP grâce au serveur.
# Ici nous supposons qu'après le basculement du serveur en mode SLIP, celui-ci
# nous donne l'adresse IP
#   mode that it prints your IP address
get $locip remote 30
if $errlvl != 0 goto prompt_error

# réglage des paramètres SLIP.
```

```
get $mtu 296
# S'assurer que "route add -net default xs4all.hacktic.nl" sera fait
default

# Dire bonjour, et en avant!
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for sliplogin to fire up...
goto error

login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

L'exemple précédent suppose que vous appeliez un serveur SLIP *dynamique* ; si vous appelez un serveur SLIP *statique*, alors le fichier sample.dip fourni avec *dip337j−uri.tgz* devrait vous convenir.

Quand on donne à *dip* la commande *get $local*, il cherche dans le texte venant de l'extrémité de la ligne une chaîne de caractères ressemblant à une adresse IP, c'est à dire des ensembles de nombres séparés par des caractères `.'. Cette modification fut mise en place plus spécialement pour les serveurs SLIP *dynamiques*, afin que le processus de lecture de l'adresse IP fournie par le serveur soit automatisé.

L'exemple ci−dessus crée automatiquement une route par défaut via votre liaison SLIP, et si ce n'est pas ce que vous voulez, car vous avez une connexion Ethernet qui devrait être votre route par défaut, alors enlevez la commande *default* du script. Après que le script ait fini de tourner, tapez la commande *ifconfig*, et vous verrez que vous avez un périphérique *sl0*. C'est votre périphérique SLIP. Si le besoin s'en fait sentir, vous pouvez modifier manuellement sa configuration, après que la commande *dip* soit finie, en utilisant les commandes *ifconfig* et *route*.

Notez que *dip* vous permet de choisir parmi différents protocoles en utilisant la commande *mode*, l'exemple le plus courant étant *cSLIP* pour utiliser SLIP avec compression. Notez encore que les deux extrémités de la liaison doivent être d'accord, aussi assurez−vous que ce que vous avez choisi est en accord avec les réglages du serveur.

L'exemple montré ci−dessus est plutôt robuste et devrait faire face à la plupart des erreurs. Référez−vous à la page de manuel de *dip* pour plus d'informations. Naturellement, vous pouvez, par exemple, modifier le script pour réaliser des choses comme recomposer le numéro vers le serveur si la connexion n'a pas été faite au bout d'un certain temps, ou même essayer une série de serveurs si vous avez accès à plus d'un d'entre eux.

11.4.7. Connexion permanente SLIP utilisant une ligne et slattach

Si vous avez deux machines reliées par un câble, ou si vous êtes suffisamment riche pour avoir une ligne dédiée, ou un autre type de connexion permanente entre votre machine et une autre, alors vous n'avez pas besoin de vous casser la tête avec *dip* pour régler votre liaison série. *slattach* est un utilitaire très simple à utiliser et vous permet d'avoir les fonctionnalités juste nécessaires pour configurer votre connexion.

Puisque votre connexion est permanente, vous ajoutez quelques commandes dans votre fichier *rc.inet1*. Tout ce dont vous avez besoin pour une connexion permanente est de vous assurer que vous avez configuré votre périphérique série à la bonne vitesse et basculer votre périphérique série en mode SLIP. *slattach* vous permet de faire ceci avec une seule commande. Ajoutez ce qui suit à votre fichier *rc.inet1* :

```
#
# Attache une connexion SLIP statique sur une ligne dédiée
#
#  configure /dev/cua0 à la vitesse de 19.2kbps et cslip
/sbin/slattach −p cslip −s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# Fin de SLIP statique.
```

Où :

IPA.IPA.IPA.IPA
: représente votre adresse IP.

IPR.IPR.IPR.IPR
: représente l'adresse IP de l'hôte distant.

slattach alloue le premier périphérique SLIP disponible au périphérique série spécifié. *slattach* démarre avec *sl0*. Par conséquent la première commande *slattach* relie le périphérique *sl0* au périphérique spécifé, puis *sl1* la fois suivante, etc.

slattach vous permet de configurer un certain nombre de protocoles grâce à l'argument *−p*. Dans votre cas vous utilisez soit *SLIP* soit *cSLIP* suivant que vous voulez utiliser la compression ou non. Note : les deux extrémités doivent être d'accord sur l'utilisation de la compression.

11.4.8. Serveur SLIP

Vous avez peut−être une machine connectée au réseau et vous aimeriez que d'autres personnes puissent s'y connecter pour y chercher des services de réseau, alors vous devez configurer votre machine comme serveur. Si vous voulez utiliser SLIP comme protocole de ligne série, vous avez trois possiblités pour configurer votre machine Linux comme serveur SLIP. Ma préférence est la première présentée, *sliplogin*, car elle semble la plus facile à configurer et à comprendre, mais je présenterai un résumé pour chacune, ainsi vous pourrez décider par vous−même.

11.4.9. Serveur SLIP utilisant *sliplogin*.

sliplogin est un programme que vous pouvez utiliser à la place du shell normal de connexion pour les utilisateurs SLIP, et qui convertit la ligne terminal en ligne SLIP. Il vous permet de configurer votre machine Linux soit en *serveur à adresse statique* (les utilisateurs obtiennent toujours la même adresse à chaque connexion), soit en *serveur à adresse dynamique* (les utilisateurs obtiennent une adresse qui n'est pas forcément la même que lors de la connexion précédente).

L'appelant se connecte comme sur un terminal standard, en donnant son nom d'utilisateur et son mot de passe, mais au lieu d'avoir une invite de shell après la connexion, *sliplogin* est exécuté et cherche dans son fichier de configuration une entrée dont le nom correspond à celui de l'appelant. S'il en détecte une, il configure la ligne avec 8 bits de données, et utilise un appel *ioctl* pour basculer celle−ci en ligne SLIP. Quand ce processus est fini, la dernière étape de la configuration prend place, *sliplogin* invoquant un script qui configure l'interface SLIP avec l'adresse IP adéquate, ainsi que le masque de réseau et positionne le routage approprié. Ce script est appelé habituellement */etc/slip.login*, mais tout comme *getty*, si certains appelants nécessitent une initialisation spéciale, alors vous pouvez créer des scripts de configuration appelés */etc/slip.login.loginname* qui seront utilisés à la place du script par défaut.

Il y a quelques fichiers que vous devez configurer pour que *sliplogin* travaille pour vous. Je décrirai comment et où obtenir les logiciels et comment chacun est configuré. Ces fichiers sont :

- */etc/passwd*, pour l'acceptation des utilisateurs entrants;
- */etc/slip.hosts*, qui contient une information spécifique de chaque utilisateur entrant;
- */etc/slip.login*, qui s'occupe de la configuration du routage;
- */etc/slip.tty*, requis uniquement si vous configurez votre serveur avec *allocation d'adresse dynamique* : il contient une table des adresses à allouer.
- */etc/slip.logout*, qui contient les commandes de `nettoyage' après une déconnexion volontaire ou intempestive.

11.4.10. Où obtenir *sliplogin*

Votre distribution contient peut−être déjà le paquetage; si ce n'est pas le cas alors *sliplogin* peut être obtenu sur metalab.unc.edu. Le fichier tar contient à la fois les sources, les binaires précompilés et une page de *manuel*.

Pour s'assurer que seuls les utilisateurs autorisés pourront faire tourner le programme *sliplogin*, vous devez ajouter une entrée dans votre fichier */etc/group* similaire à la suivante :

```
 ..
slip::13:radio,fred
 ..
```

Lorsque vous installez le paquetage *sliplogin*, *Makefile* change le groupe du programme *sliplogin* en *slip*, et cela signifie que seuls les utilisateurs qui appartiennent à ce groupe pourront l'exécuter. L'exemple donné ci−dessus ne permet qu'aux utilisateurs *radio* et *fred* de pouvoir faire tourner le programme *sliplogin*.

Pour installer les binaires dans le répertoire */sbin* et les pages de *manuel* dans la section 8, faites :

```
root# cd /usr/src
root# gzip −dc .../sliplogin−2.1.1.tar.gz | tar xvf −
root# cd sliplogin−2.1.1
root# <..éditez le Makefile si vous n'utilisez pas les shadow passwords..>
root# make install
```

Si vous voulez recompiler les binaires avant de les installer, faites *make clean* avant de faire *make install*. Si vous voulez installer les binaires autre part, vous devez éditer le fichier *Makefile* et le modifier en conséquence.

11.4.11. Configurer */etc/passwd* pour utiliser SLIP

Normalement vous devez créer des noms d'utilisateurs spéciaux, pour ceux qui appellent avec SLIP, dans votre fichier */etc/passwd*. Une convention souvent suivie est d'utiliser le *nom d'utilisateur* de l'appelant préfixée avec la lettre capitale `S'. Ainsi, par exemple, si l'appelant s'appelle *radio* alors vous pouvez créer une entrée dans le fichier */etc/passwd* ressemblant à ceci :

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

Le nom du compte n'a pas réellement d'importance, du moment qu'il ait une signification pour vous.

Note : l'appelant n'a pas besoin de répertoire home spécial car il n'utilisera pas de shell sur la machine, dès lors */tmp* est un bon choix. Notez bien que *sliplogin* est utilisé à la place du shell de connexion normal.

11.4.12. Configurer */etc/slip.hosts*

Le fichier */etc/slip.hosts* est le fichier où *sliplogin* cherche les entrées correspondant au nom de connexion pour obtenir les détails de configuration. C'est le fichier où sont indiqués l'adresse IP et le masque de réseau qui seront assignés à l'appelant et configurés pour leur usage. Des exemples d'entrées pour deux utilisateurs, une statique pour *radio* et l'autre dynamique pour *albert* ressemblent à ceci :

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Les entrées du fichier */etc/slip.hosts* sont :

1. Le nom de connexion de l'appelant.
2. L'adesse IP de la machine serveur, donc de la machine contenant ce fichier.
3. L'adresse IP qui sera attribuée à l'appelant. Si le champ vaut *DYNAMIC* alors l'adresse IP sera allouée suivant les informations contenues dans le fichier */etc/slip.tty* décrit plus loin. *Note :* vous devez utiliser au moins la version 1.3 de sliplogin pour que cela fonctionne.
4. Le masque de réseau assigné à la machine appelante, en notation décimale, par exemple 255.255.255.0 pour un masque de réseau de classe C.
5. Un réglage du mode SLIP qui active/désactive la compression. Les valeurs autorisées sont "*normal*" et "*compressed*".
6. Un paramètre de délai qui spécifie combien de temps la ligne peut rester inactive (aucun datagramme reçu) avant une déconnexion automatique. Une valeur négative désactive cette possiblité.
7. arguments optionnels.

Note : Vous pouvez mettre soit les noms d'hôtes soit les adresses IP en notation décimale pointée pour les champs 2 et 3. Si vous utilisez les noms d'hôtes, alors ces hôtes doivent être résolubles, c'est à dire que votre machine est capable de déterminer une adresse IP pour ces noms d'hôtes, sinon le script échouera pendant l'appel. Vous pouvez le tester en faisant telnet vers un nom d'hôte : si vous obtenez le message `*Trying nnn.nnn.nnn...*' alors votre machine est capable de trouver une adresse ip pour ce nom d'hôte. Si vous obtenez le message `*Unknown host*', alors il n'en a pas. Dans ce cas essayez d'utiliser l'adress IP en notation décimale pointée, ou bien voyez du côté de votre configuration de solveur de noms (voir la section *Résolution de noms*).

Les modes les plus courants de SLIP sont :

normal
: mode SLIP normal non compressé.

compressed
: mode avec compression van Jacobsen des en−têtes (cSLIP)

Bien sûr ils sont mutuellement exclusifs, vous devez utiliser l'un ou l'autre. Pour plus d'informations sur les options disponibles, voir les pages de manuels.

11.4.13. Configurer le fichier */etc/slip.login*.

Après que *sliplogin* ait exploré le fichier */etc/slip.hosts* et ait trouvé une entrée qui convient, il essaye d'exécuter le fichier */etc/slip.login* pour effectivement configurer l'interface SLIP avec son adresse IP et son masque de réseau. L'exemple de fichier */etc/slip.login* fourni avec le paquetage *sliplogin* ressemble à ceci :

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# fichier générique de connexion pour une ligne SLIP. Invoqué par sliplogin
# avec les paramètres:
#     $1       $2       $3    $4, $5, $6 ...
# unité SLIP vitesse    pid   arguments tirés de slip.host
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Notez que ce script utilise seulement les commandes *ifconfig* et *route* pour configurer le périphérique SLIP avec sa propre adresse IP, l'adresse IP de l'hôte distant , le masque de réseau puis crée une route vers l'adresse distante via le périphérique SLIP. C'est–à–dire la même chose que si vous utilisiez la commande *slattach*.

Notez aussi l'utilisation de *Proxy ARP* pour s'assurer que les hôtes placés sur le même segment éthernet que la machine serveur sauront comment atteindre l'hôte qui s'est connecté. Le champ *<hw_addr>* doit être l'adresse matérielle de la carte Ethernet de la machine. Si votre machine serveur n'est pas sur un réseau Ethernet, vous pouvez ignorer cette ligne.

11.4.14. Configurer le fichier */etc/slip.logout*

Quand la connexion s'est arrêtée, assurez–vous que le périphérique série soit revenu à son état normal de telle sorte que les appelants suivants puissent se connecter correctement. Ceci est accompli en utilisant le fichier */etc/slip.logout*. Il est de format très simple et est appelé avec le même argument que le fichier */etc/slip.login*.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Tout ce qu'il fait est de `mettre à zéro' l'interface qui supprimera la route précédemment créée. Il utilise aussi la commande *arp* pour supprimer tout arp proxy en place, encore une fois vous n'avez pas besoin de la commande *arp* dans le script si votre machine serveur ne possède pas de port Ethernet.

11.4.15. Configurer le fichier */etc/slip.tty*

Si vous utilisez une allocation d'adresse ip dynamique (tous les hôtes configurés avec le mot–clé *DYNAMIC* dans le fichier */etc/slip.hosts*) alors vous devez configurer le fichier */etc/slip.tty* pour afficher les adresses qui seront assignées aux ports. Vous n'aurez besoin de ce fichier que si vous voulez que votre serveur alloue des adresses aux utilisateurs de manière dynamique.

Ce fichier est un tableau qui liste les périphériques *tty* supportant les connexions SLIP entrantes et l'adresse ip qui sera assignée aux utilisateurs se connectant à ceux−ci.

Son format est le suivant :

```
# slip.tty    mappage d'adresses tty -> IP pour SLIP dynamique
# format: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Ce que dit ce tableau est que les appelants qui se connectent sur le port */dev/ttyS0* et dont le champ adresse dans le fichier */etc/slip.hosts* vaut sur *DYNAMIC* auront l'adresse *192.168.0.100*.

De cette manière vous n'avez besoin d'allouer qu'une seule adresse par port pour tous les utilisateurs n'ayant pas besoin d'adresse fixe. Ceci vous permet d'avoir le nombre minimum d'adresses nécessaires pour éviter du gaspillage.

11.4.16. Serveur Slip utilisant *dip*

Tout d'abord laissez−moi dire que certaines informations ci−dessous proviennent des pages de manuel de *dip*, où la manière de faire tourner Linux comme serveur SLIP est brièvement décrite. Faites attention aussi que ce qui suit est fondé sur le paquetage *dip337o−uri.tgz* et ne s'applique vraisemblablement pas à d'autres versions de *dip*.

dip possède un mode de traitement des données d'entrée qui permet de localiser automatiquement un utilisateur entrant et qui configure la ligne série comme lien SLIP suivant les informations trouvées dans le fichier */etc/diphosts*. Ce mode est activé en invoquant *dip* avec *diplogin*. Voilà donc comment utiliser *dip* comme serveur SLIP, en créant des comptes spéciaux où *diplogin* est utilisé comme shell de connexion.

La première chose à faire est de créer un lien symbolique comme suit :

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Ensuite vous devez ajouter des entrées à la fois dans vos fichiers */etc/passwd* et */etc/diphosts*. Les entrées que vous devez y mettre sont formatées comme suit :

Pour configurer Linux comme serveur SLIP avec *dip*, vous devez créer quelques comptes SLIP spéciaux pour les utilisateurs, où *dip* (en mode d'entrée) est utilisé comme shell de connexion. Une convention suggérée est d'avoir tous les comptes SLIP commençant avec la lettre `S' majuscule, par exemple `Sfredm'.

Un exemple d'entrée dans */etc/passwd* pour un utilisateur SLIP ressemble à ceci :

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^         ^^        ^^  ^^   ^^   ^^   ^^
|          |         |   |    |    |    \_ _ diplogin comme shell de connexion
|          |         |   |    |    \_ _ _ _  Répertoire personnel
|          |         |   |    \_ _ _ _ _ _ _ Nom complet d'utilisateur
|          |         |   \_ _ _ _ _ _ _ _ _  GID
|          |         \_ _ _ _ _ _ _ _ _ _ _  UID
|          \_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _  Mot de passe chiffré
\_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ Nom de connexion Slip
```

Après la connexion de l'utilisateur, le programme *login* (s'il trouve et accepte l'utilisateur) exécute la commande *diplogin*. *dip*, lorsqu'il est invoqué en tant que *diplogin* sait qu'il sera automatiquement utilisé comme shell de connexion. Quand il est démarré comme *diplogin* la première chose qu'il fait est d'utiliser l'appel de la fonction *getuid()* pour obtenir l'identificateur de l'utilisateur appelant. Il regarde ensuite dans le fichier */etc/diphosts* pour trouver la première entrée qui corresponde soit à l'utilisateur soit au périphérique *tty* où l'appel est entré et se configure lui−même de manière appropriée. Par un choix judicieux : soit de donner à l'utilisateur une entrée dans le fichier *diphosts*, soit de laisser à l'utilisateur la configuration par défaut, vous pouvez construire votre serveur de telle manière que vous puissiez faire cohabiter des utilisateurs ayant des adresses allouées statiquement ou dynamiquement.

dip ajoutera automatiquement une entrée `Proxy−ARP' si elle est invoquée en mode d'entrée, aussi vous n'avez pas à vous soucier d'ajouter de telles entrées manuellement.

11.4.17. Configurer */etc/diphosts*

/etc/diphosts est utilisé par *dip* pour examiner des configurations préétablies concernant des hôtes éloignés. Ceux−ci peuvent être des hôtes se connectant sur votre machine, ou bien des machines sur lesquelles vous vous connectez.

Le format général de */etc/diphosts* est :

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

Les champs sont :

1. *nom de connexion* : comme retourné par getpwuid(getuid()) ou bien le nom de tty.
2. *inutilisé* : pour compatibilité avec passwd
3. *Adresse distante* : adresse IP de l'appelant, soit numérique soit nominative
4. *Adresse locale* : adresse IP de cette machine, soit numérique soit nominative.
5. *Masque de réseau* : en notation décimale pointée
6. *Commentaires* : vous y mettez ce que vous voulez.
7. *protocole* : Slip, CSlip, etc.
8. *MTU* : nombre décimal

Un exemple d'entrée */etc/net/diphosts* pour un hôte distant peut être :

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

qui spécifie une liaison SLIP avec une adresse distante de 145.71.34.1 et un MTU de 296, ou :

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

qui spécifie une liaison compatible cSLIP avec une adresse distante de 145.71.34.1 et un MTU de 1006.

Dès lors, tous les utilisateurs à qui vous permettez d'avoir une connexion avec allocation d'adresse IP statique auront une entrée dans */etc/diphosts*. Si vous voulez que des utilisateurs qui appellent sur un port particulier aient leur adresse allouée dynamiquement, vous devez avoir une entrée pour le périphérique *tty*, mais pas d'entrée pour l'utilisateur lui−même. Vous devez vous souvenir de configurer au moins une entrée pour chaque périphérique *tty* que vos utilisateurs entrants utiliseront pour être sûrs qu'une configuration adéquate soit disponible, indépendamment du modem sur lequel ils se connectent.

Quand un utilisateur se connecte, il recevra une invite normal de login et une demande de mot de passe, pour lesquels il devra entrer son identificateur SLIP et son mot de passe. Si tout est correct, l'utilisateur ne verra pas de message spécial, il devra juste basculer en mode SLIP chez lui et ensuite il sera connecté et configuré avec les paramètres contenus dans le fichier *diphosts*.

11.4.18. Serveur SLIP utilisant l'ensemble *dSLIP*

Matt Dillon *<dillon@apollo.west.oic.com>* a écrit un paquetage qui permet des liaisons SLIP non seulement entrantes mais aussi sortantes. Le paquetage de Matt est une combinaison de petits programmes et de scripts qui prennent en charge les connexions à votre place. Vous aurez besoin de *tcsh* car au moins l'un des scripts en a besoin. Matt fournit une copie binaire de l'utilitaire *expect* car il est aussi nécessaire pour l'un des scripts. Il serait préférable d'avoir une certaine expérience de *expect* pour que ce paquetage soit utile pour vous, mais que cela ne vous décourage pas.

Matt a écrit une bonne procédure d'installation dans le fichier README, aussi je ne me fatiguerai pas à la répéter.

Vous pouvez récupérer le paquetage *dSLIP* sur son site d'origine :

apollo.west.oic.com

```
/pub/linux/dillon_src/dSLIP203.tgz
```

ou bien sur :

metalab.unc.edu

```
/pub/Linux/system/Network/serial/dSLIP203.tgz
```

Lisez le fichier *README* et créez les entrées */etc/passwd* et */etc/group* *avant* de faire *make install*.

Chapter 12. Autres technologies réseau

Les paragraphes suivants traitent de sujets spécifiques concernant des technologies liées au réseau. Les informations qui y sont contenues ne s'appliquent pas forcément aux autres types de technologies réseau. Les sujets sont traités par ordre alphabétique.

12.1. ARCNet

Les noms de fichier périphériques de ARCNet sont `*arc0e*', `*arc1e*', `*arc2e*' ... ou bien `*arc0s*', `*arc1s*', `*arc2s*', etc. La première carte détectée par le noyau devient `*arc0e*' ou `*arc0s*' et les autres sont nommées en suivant dans l'ordre de leur détection. La lettre finale dépend de votre choix : soit un format d'encapsulation de paquets Ethernet, soit un format de paquets suivant RFC1051.

Options de compilation du noyau :

```
Network device support  --->
    [*] Network device support
    <*> ARCnet support
    [ ]   Enable arc0e (ARCnet "Ether-Encap" packet format)
    [ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

Si vous avez construit convenablement votre noyau pour supporter votre carte Ethernet, alors la configuration de la carte est facile.

Typiquement vous devriez utiliser quelque chose comme ceci :

```
root# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

Merci de vous référer aux documents */usr/src/linux/Documentation/networking/arcnet.txt* et */usr/src/linux/Documentation/networking/arcnet−hardware.txt* pour d'autres informations.

Le support ARCNet fut développé par Avery Pennarun, *apenwarr@foxnet.net*. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.2. Appletalk (*AF_APPLETALK*)

Le support Appletalk ne possède pas de noms de périphériques spécifiques car il utilise les périphériques réseau existants.

Options de compilation noyau :

```
Networking options  --->
    <*> Appletalk DDP
```

Le support Appletalk permet à votre machine Linux de dialoguer avec les réseaux Apple. Son utilisation principale est de pouvoir partager des ressources, comme les imprimantes et les disques, entre vos ordinateurs Linux et Apple. Un logiciel supplémentaire est requis, il s'appelle *netatalk*. Wesley Craig *netatalk@umich.edu* représente une équipe appelée le `Research Systems Unix Group' à l'université du Michigan. Celle−ci a élaboré le paquetage *netatalk*, qui fournit un logiciel implémentant la pile protocole Appletalk et quelques utilitaires. Soit ce paquetage *netatalk* vous a été fourni avec votre distribution Linux, soit vous pouvez le récupérer par ftp depuis le site University of Michigan

Pour construire et installer le paquetage, vous faites :

```
user% tar xvfz .../netatalk-1.4b2.tar.Z
user% make
root# make install
```

Vous pouvez éditer le fichier `Makefile' avant de faire appel à *make*, plus précisément pour changer la valeur de la variable DESTDIR qui définit l'endroit où les fichiers seront installés plus tard. Le répertoire par défaut, /usr/local/atalk, semble très raisonnable.

12.2.1. Configurer le support Appletalk.

La première chose à faire pour que tout fonctionne est de vérifier que les entrées adéquates sont présentes dans le fichier */etc/services*. Ces entrées sont :

```
rtmp    1/ddp   # Routing Table Maintenance Protocol
nbp     2/ddp   # Name Binding Protocol
echo    4/ddp   # AppleTalk Echo Protocol
zip     6/ddp   # Zone Information Protocol
```

L'étape suivante consiste à créer les fichiers de configuration Appletalk dans le répertoire */usr/local/atalk/etc* (ou bien à l'endroit où vous avez installé le paquetage).

Le premier fichier à créer est */usr/local/atalk/etc/atalkd.conf*. Initialement ce fichier ne nécessite qu'une ligne qui indique le périphérique supportant le réseau sur lequel sont vos machines Apple :

```
eth0
```

Le programme démon Appletalk ajoutera d'autres détails quand il tournera.

12.2.2. Exporter un système de fichiers Linux avec Appletalk.

Vous pouvez exporter des systèmes de fichiers depuis votre machine Linux vers le réseau en sorte qu'une machine Apple puisse les partager.

Pour cela vous devez configurer le fichier */usr/local/atalk/etc/AppleVolumes.system*. Il y a une autre fichier de configuration appelé */usr/local/atalk/etc/AppleVolumes.default* qui a exactement le même format et qui décrit quels systèmes de fichiers les utilisateurs connectés pourront recevoir avec des privilèges d'invités.

Tous les détails, qui vous diront comment configurer ces fichiers et avec quelles options, peuvent être trouvés dans la page de manuel de *afpd*.

Un simple exemple :

```
/tmp Scratch
/home/ftp/pub "Public Area"
```

Ce qui exportera votre système de fichiers */tmp* comme volume AppleShare `Scratch' et votre répertoire public ftp comme volume AppleShare `Public Area'. Les noms de volume ne sont pas obligatoires, le programme démon pouvant les choisir pour vous, mais ça ne coûte rien de les spécifier quand même.

12.2.3. Partager votre imprimante Linux avec Appletalk.

Partager votre imprimante Linux avec vos machines Apple est très simple. Vous devez faire tourner le programme *papd* qui est le démon protocole d'accès aux imprimantes de Appletalk. Lorsque vous faites tourner ce programme il acceptera les requêtes émanant de vos machines Apple et spoulera le travail d'impression vers votre démon local d'impression.

Vous devrez éditer le fichier */usr/local/atalk/etc/papd.conf* pour configurer le démon. La syntaxe de ce fichier est la même que le fichier habituel */etc/printcap*. Le nom que vous donnez à la définition de l'imprimante doit être conforme au protocole de désignation Appletalk, NBP.

Un exemple de configuration ressemble à ceci :

```
TricWriter:\
   :pr=lp:op=cg:
```

Ce qui fera une imprimante nommée `TricWriter' disponible pour le réseau Appletalk et tous les travaux acceptés seront imprimés sur l'imprimante linux `*lp*' (telle que définie dans le fichier */etc/printcap*) utilisant *lpd*. L'entrée `*op=cg*' indique que l'utilisateur linux `*cg*' est l'opérateur de l'imprimante.

12.2.4. Démarrer Appletalk.

Bon, vous devriez être prêts pour essayer cette configuration de base. Le fichier *rc.atalk* fourni avec le paquetage *netatalk* devrait vous convenir, alors vous faites ceci :

```
# /usr/local/atalk/etc/rc.atalk
```

et tout devrait démarrer et tourner sans problémes. Vous ne devriez voir aucun message d'erreurs et le programme devrait vous envoyer des messages sur la console indiquant chaque étape qui démarre.

12.2.5. Tester Appletalk.

Pour tester si le programme fonctionne correctement, allez sur une des machines Apple, déroulez le menu Pomme, cliquez sur AppleShare, et votre boîte Linux devrait apparaître.

12.2.6. Mises en garde sur Appletalk.

- Vous aurez peut−être besoin de démarrer votre support Appletalk avant de configurer votre réseau IP. Si vous avez des problèmes pour démarrer vos programmes Appletalk, ou si après les avoir démarrés vous avez des ennuis avec votre réseau IP, essayez alors de mettre en route votre programme Appletalk avant de faire démarrer */etc/rc.d/rc.inet1*.
- Le démon *afpd* (Apple Filing Protocol Daemon) SECOUE SÉVÈREMENT VOTRE DISQUE DUR. Derrière les points de montage il crée deux répertoires appelés *.AppleDesktop* et *Network Trash Folder*. Ensuite, pour chaque répertoire auquel vous accédez il crée un sous−répertoire *.AppleDouble* pour pouvoir stocker des fichiers de ressource, etc. Réfléchissez bien avant d'exporter */*, vous aurez besoin de pas mal de temps pour tout nettoyer.
- Le programme *afpd* attend des mots de passe en clair venant des Macs. La sécurité pouvant être un problème, soyez donc attentifs lors de l'utilisation de ce démon sur une machine connectée sur l'Internet et ne vous en prenez qu'à vous−même si quelqu'un de mal intentionné vous fait des misères.
- Les outils de diagnostic existants tels que *netstat* et *ifconfig* ne supportent pas Appletalk. Les informations peuvent être trouvées dans le répertoire */proc/net/* si vous en avez besoin.

12.2.7. Autres informations

Pour en savoir plus sur la configuration de Appletalk pour Linux, référez vous à la page de Anders Brownworth *Linux Netatalk−HOWTO* disponible à l'url thehamptons.com.

12.3. ATM

Werner Almesberger *<werner.almesberger@lrc.di.epfl.ch>* dirige un projet en vue de fournir un support Mode de Transfert Asynchrone (Asynchronous Transfer Mode) pour Linux. Les informations sur l'état du projet se trouvent sur : lrcwww.epfl.ch.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.4. AX25 (*AF_AX25*)

Les noms de périphériques AX.25 sont `*sl0*', `*sl1*', etc. avec les noyaux *2.0.** ou `*ax0*', `*ax1*', etc. avec les noyaux *2.1.**.

Options de compilation du noyau :

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

Les protocoles AX25, Netrom et Rose sont couverts par le document AX25−HOWTO. Ces protocoles sont utilisés par les radio−amateurs du monde entier pour l'expérimentation packet−radio.

L'essentiel du travail d'implémentation de ces protocoles a été réalisé par Jonathon Naylor, *jsn@cs.nott.ac.uk*.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.5. DECNet

Le support pour DECNet est en cours d'élaboration. Vous devriez le voir apparaitre dans l'un des prochains noyaux *2.1.**. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.6. FDDI (Fiber Distributed Data Interface)

Les noms de périphériques FDDI sont `*fddi0*', `*fddi1*', `*fddi2*' etc. La première carte détectée par le noyau s'appelle `*fddi0*' et le reste est nommé dans l'ordre de détection.

Larry Stefani, *lstefani@ultranet.com*, a développé un gestionnaire pour les cartes Digital Equipment Corporation FDDI EISA et PCI.

Options de compilation noyau :

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

Lorsque vous avez construit et installé votre noyau pour supporter le gestionnaire FDDI, la configuration de l'interface FDDI est presque identique à celle d'une interface Ethernet. Vous devez spécifier le nom de l'interface FDDI appropriée dans les commandes *ifconfig* et *route*.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.7. Relais de trames (Frame Relay)

Les noms de périphériques de `relais de trames' sont `*dlci00*', `*dlci01*' etc pour les systèmes d'encapsulation DLCI et `*sdla0*', `*sdla1*' etc pour les FRAD(s) (Frame Relay Access Device).

Le relais de trames est une nouvelle technologie réseau conçue pour s'adapter au trafic de transmission de données `par à coups' ou de nature intermittente. Vous vous connectez à un réseau de ce type en utilisant un dispositif d'accès par relais de trames (FRAD). Les supports Linux relais de trames supportent IP par−dessus celui−ci comme décrit dans la RFC−1490.

Options de compilation noyau :

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

Mike McLagan, *mike.mclagan@linux.org*, a développé le support Frame Relay et les outils de configuration.

À l'heure actuelle le seul FRAD supporté est, à ma connaissance, Sangoma Technologies *S502A*, *S502E* et *S508*. et Emerging Technologies. Leur site se trouve sur ici.

Je voudrais dire quelquechose. J'ai une expérience personnelle avec Emerging Technologies et je vous les recommande pas. Je les ai trouvés absolument pas professionnels et très grossiers. Si quelqu'un d'autre a eu une bonne expérience avec eux, faites le moi savoir. A leur décharge, leur produit est souple d'utilisation et paraît stable.

Pour configurer les systèmes FRAD et DLCI après avoir reconstruit votre noyau, vous aurez besoin des outils de configuration. Ils sont disponibles sur ftp.invlogic.com.

Compiler et installer les outils est très facile, mais le manque de fichier Makefile au premier niveau oblige à le faire à la main :

```
user% tar xvfz .../frad-0.15.tgz
user% cd frad-0.15
user% for i in common dlci frad; make -C $i clean; make -C $i; done
root# mkdir /etc/frad
root# install -m 644 -o root -g root bin/*.sfm /etc/frad
root# install -m 700 -o root -g root frad/fradcfg /sbin
root# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Notez que ces commandes utilisent la syntaxe du shell *sh*, et si vous utilisez *csh* (comme *tcsh*), la boucle *for* sera différente.

Après l'installation vous devez créer un fichier */etc/frad/router.conf* Vous pouvez utiliser cet exemple, qui est une version modifiée de l'un des fichiers donné en exemple :

```
# /etc/frad/router.conf
# C'est un modèle de configuration pour relais de trames.
# Tout y est inclus. Les valeurs par défaut sont fondées sur le code
# fourni avec les gestionnaires DOS de la carte Sangoma S502A.
#
# Une ligne avec '#' est un commentaire
# Les blancs sont ignorés (vous pouvez utiliser des tabulations aussi).
# Les sections [] inconnues et les entrées inconnues sont ignorées.
#

[Devices]
Count=1                 # nombre de périphériques à configurer
Dev_1=sdla0             # nom d'un périphérique
#Dev_2=sdla1            # nom d'un périphérique

# Ce qui est spécifie ici s'applique à tous les périphériques, et peut être
# mis à jour pour chaque carte individuelle.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
#
# MTU=1500              # Taille maximum de l'unité de transfert 4096 par défaut
# T391=10               # valeur de T391  5 - 30, 10 par défaut
# T392=15               # valeur de T392  5 - 30, 15 par défaut
# N391=6                # valeur de N391  1 - 255, 6 par défaut
# N392=3                # valeur de N392  1 - 10,  3 par défaut
# N393=4                # valeur de N393  1 - 10,  4 par défaut

# On spécifie ici les valeurs par défaut pour toutes les cartes
# CIRfwd=16             # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511
```

```
#
#
# Configurations spécifiques
#
#


#
#  Sangoma S502E
#
[sdla0]
Type=Sangoma            # Type de périphérique à configurer, actuellement seul
                        # SANGOMA est reconnu
#
# Spécifique des types 'Sangoma'
#
# cartes S502A, S502E, S508
Board=S502E
#
# Le nom du logiciel de carte en essai pour Sangoma
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# Le nom du logiciel de carte FR
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360                # Port pour cette carte particulière
Mem=C8                  # Adresse de fenêtre mémoire, A0-EE, dépend de la carte
IRQ=5                   # numéro d'IRQ, pas nécessaire pour S502A
DLCIs=1                 # Nombre de DLCI attachés à ce périphérique
DLCI_1=16               # numéro du premier DLCI, de 16 à 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Ce qui est spécifie ici s'applique au périphérique seulement,
# et remplace les valeurs par défaut
#
# Access=CPE            # CPE ou NODE,  CPE par défaut
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal        # Externe ou Interne, Interne par défaut
# Baud=128              # Débit spécifié du CSU/DSU attaché
# MTU=2048              # Taille maximum de l'unité de transfert 4096 par défaut
# T391=10               # valeur de T391   5 - 30, 10 par défaut
# T392=15               # valeur de T392   5 - 30, 15 par défaut
# N391=6                # valeur de N391   1 - 255, 6 par défaut
# N392=3                # valeur de N392   1 - 10,  3 par défaut
# N393=4                # valeur de N393   1 - 10,  4 par défaut

#
# Le second periphérique est une autre carte
#
# [sdla1]
# Type=FancyCard        # Type de périphérique à configurer.
# Board=                # Type de carte Sangoma
# Key=Value             # valeurs spécifiques pour ce type de périphérique


#
# Paramètres de configuration DLCI par défaut.
# Peuvent être écrasés par des configurations spécifiques
#
CIRfwd=64               # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
```

```
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511

#
# Configuration DLCI
# Optionnel. La convention d'appellation est
# [DLCI_D<devicenum>_<DLCI_Num>]
#

[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Drapeaux définis par Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Drapeaux définis par Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=16
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0
```

Lorsque vous avez construit votre fichier */etc/frad/router.conf*, la seule étape restante est de configurer les périphériques eux−mêmes. C'est un tout petit peu plus compliqué que la configuration normale d'un périphérique réseau; vous devez vous souvenir de monter le périphérique FRAD avant les périphériques d'encapsulation DLCI.

```
#!/bin/sh
# Configure le materiel frad et les parametres DLCI
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Montage du dispositif FRAD
ifconfig sdla0 up
#
# Configure les interfaces d'encapsulation DLCI et le routage
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add -net 192.168.10.0 netmask 255.255.255.0 dlci00
#
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add -net 192.168.11.0 netmask 255.255.255.0 dlci00
#
route add default dev dlci00
#
```

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.8. IPX (*AF_IPX*)

Le protocole IPX est la plupart du temps utilisé dans les environnements réseaux locaux Novell NetWare(tm). Linux offre un support pour ce protocole, et peut être configuré pour agir comme extrémité réseau, ou comme routeur pour les environnements réseaux IPX.

Options de compilation du noyau :

```
Networking options  ---i>
    [*] The IPX protocol
    [ ] Full internal IPX network
```

Le protocole IPX et le NCPFS sont traités en détail dans le document IPX−HOWTO. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.9. NetRom (*AF_NETROM*)

Les noms de périphériques NetRom sont `*nr0*', `*nr1*', etc.

Options de compilation du noyau :

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Les protocoles AX25, Netrom et Rose sont décrits dans le document AX25−HOWTO. Ces protocoles sont utilisés par les radio−amateurs dans le monde entier pour l'expérimentation du packet−radio.

L'essentiel du travail d'implémentation a été fait par Jonathon Naylor, *jsn@cs.not.ac.uk*. Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.10. Protocole Rose (*AF_ROSE*)

Les noms de périphériques Rose sont `*rs0*', `*rs1*', etc. . Rose est disponible dans la série des noyaux *2.1.**.

Options de compilation du noyau :

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    <*> Amateur Radio X.25 PLP (Rose)
```

Les protocoles AX25, Netrom et Rose sont expliqués dans le AX25−HOWTO. Ces protocoles sont utilisés par les opérateurs radio−amateur du monde entier pour l'expérimentation du packet−radio.

L'essentiel du travail d'implémentation de ces protocoles a été réalisé par Jonathon Naylor, *jsn@cs.not.ac.uki*.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.11. Support SAMBA − `NetBEUI', `NetBios', `CIFS'.

SAMBA est une implémentation du protocole Session Management Block. Samba permet aux Systèmes Microsoft et autres de monter et d'utiliser vos disques et imprimantes.

SAMBA et sa configuration sont décrits en détail dans le SMB−HOWTO.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.12. Support STRIP (Starmode Radio IP)

Les noms de périphériques STRIP sont `st0', `st1', etc.

Options de compilation du noyau :

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        < > STRIP (Metricom starmode radio IP)
```

STRIP est un protocole conçu spécialement pour un certain type de modems radio Metricom dans le cadre d'un projet de recherche conduit par l'Université de Stanford appelé MosquitoNet Project. Il y a un tas de choses intéressantes à lire, même si vous n'êtes pas directement concerné par le projet.

Les radios Metricom se connectent sur un port série et emploient la technologie à large bande spectrale et peuvent aller jusqu'à 100kbps. Des informations sur ceux−ci sont disponibles sur : Le serveur web de Metricom.

À l'heure actuelle, les outils réseau habituels ne supportent pas le gestionnaire STRIP, vous devez donc télécharger des outils personnalisés à partir du serveur web MosquitoNet. Pour avoir des détails sur les logiciels à utiliser allez voir : MosquitoNet STRIP Page.

En résumé la configuration consiste à utiliser un programme *slattach* modifié pour régler la discipline de ligne d'un périphérique série pour SLIP, puis à configurer le périphérique `st[0−9]' résultant comme vous le feriez pour Ethernet avec une exception importante : pour des raisons techniques STRIP ne supporte pas le protocole ARP , vous devez alors configurer manuellement les entrées ARP pour chacun des hôtes de votre sous−réseau. Cela ne devrait pas être trop contraignant.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.13. Token Ring

Le noms de périphériques Token ring sont `tr0', `tr1' etc. Token Ring est un protocole LAN standard IBM en vue d'éviter les collisions en fournissant un mécanisme qui n'autorise qu'une seule station du LAN à transmettre à un moment donné. Un `jeton' est détenu par une station à un moment donné, et celle−ci est la seule autorisée à émettre. Lorque c'est fait, elle passe le jeton à la station suivante. Le jeton fait le tour de toutes les stations actives, d'où le nom de `Token Ring' (anneau à jeton).

Options de compilation du noyau :

```
Network device support  --->
        [*] Network device support
        ....
        [*] Token Ring driver support
        < > IBM Tropic chipset based adaptor support
```

La configuration de token ring est identique à celle de l'Ethernet à l'exception du nom de périphérique réseau à configurer.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.14. X.25

X.25 est un protocole de circuit basé sur la commutation de paquets défini par le *C.C.I.T.T.* (un groupe de normalisation reconnu par les compagnies de télécommunications dans la plupart du monde). Une implémentation de X.25 et LAPB est en cours dans les noyaux récents *2.1.**.

Jonathon Naylor *jsn@cs.nott.ac.uk* dirige le développement et une liste de diffusion a été créée pour discuter des affaires relatives à X.25 pour Linux. Pour y souscrire, envoyez un message à : *majordomo@vger.rutgers.edu* avec le texte "*subscribe linux−x25*" dans le corps du message.

Les dernières versions des outils de configuration peuvent être obtenues sur le site ftp de Jonathon à ftp.cs.nott.ac.uk.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

12.15. Carte WaveLan

Les noms de périphériques Wavelan sont `*eth0*', `*eth1*', etc.

Options de compilation du noyau :

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        ....
        <*> WaveLAN support
```

La carte WaveLAN est une carte LAN sans−fil à large bande. Elle ressemble beaucoup en pratique à une carte Ethernet et se configure presque de la même manière.

Vous pouvez avoir des informations sur la carte Wavelan sur Wavelan.com.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 13. Câbles et câblages

Ceux qui sont habiles du fer à souder peuvent vouloir fabriquer leurs propres câbles pour relier deux machines Linux. Les schémas de câblage suivants pourront les y aider.

13.1. Câble série NULL Modem

Tous les câbles NULL modem ne se ressemblent pas. Une grosse partie d'entre eux ne font que faire croire à votre ordinateur que tous les signaux appropriés sont présents et échangent les données de transmission et de réception. C'est bien, mais cela signifie que vous devez utiliser le contrôle de flux logiciel (XON/XOFF) qui est moins efficace que le contrôle de flux matériel. Le câble suivant donne la meilleure transmission de signal entre les deux machines et vous permet d'utiliser le contrôle de flux matériel (RTS/CTS).

```
Pin Name  Pin                              Pin
Tx Data    2  - - - - - - - - - - - - - - - - -3
Rx Data    3  - - - - - - - - - - - - - - - - -2
```

```
RTS         4  - - - - - - - - - - - - - - - - -5
CTS         5  - - - - - - - - - - - - - - - - -4
Ground      7  - - - - - - - - - - - - - - - - -7
DTR         20 -\- - - - - - - - - - - - - - - -8
DSR         6  -/
RLSD/DCD    8  - - - - - - - - - - - - - - -/-  20
                                            \-  6
```

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

13.2. Câble port parallèle (câble PLIP)

Si vous avez l'intention d'utiliser le protocole PLIP entre deux machines alors ce câble vous conviendra indépendamment du type de port parallèle installé.

```
Pin Name     pin            pin
STROBE       1*
D0->ERROR    2  - - - - - - 15
D1->SLCT     3  - - - - - - 13
D2->PAPOUT   4  - - - - - - 12
D3->ACK      5  - - - - - - 10
D4->BUSY     6  - - - - - - 11
D5           7*
D6           8*
D7           9*
ACK->D3      10 - - - - - - 5
BUSY->D4     11 - - - - - - 6
PAPOUT->D2   12 - - - - - - 4
SLCT->D1     13 - - - - - - 3
FEED         14*
ERROR->D0    15 - - - - - - 2
INIT         16*
SLCTIN       17*
GROUND       25- - - - - - -25
```

Notes :

- Ne pas connecter les broches marquées avec un astérisque `*'.
- Les masses supplémentaires sont 18,19,20,21,22,23 et 24.
- Si le câble que vous utilisez possède un blindage, il doit être connecté à une des prises DB−25 et *une seule extrémité*.

Attention : un câble PLIP mal branché peut détruire votre carte contrôleur. Soyez attentifs et vérifiez chaque connexion deux fois pour être sûr de ne pas vous créer de travail inutile ou de gros ennuis.

Bien que l'on puisse utiliser des câbles PLIP sur des longues distances, évitez−le si possible. Les spécifications du câble permettent d'avoir une longueur d'environ 1 mètre. Faites attention si vous utilisez de grandes longueurs, car les sources de champs magnétiques élevés comme la foudre, les lignes de puissance et les émetteurs radio peuvent interférer et parfois endommager votre carte contrôleur. Si vous voulez vraiment connecter deux de vos ordinateurs sur une grande distance, utilisez plutôt des cartes Ethernet et un câble coaxial.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

13.3. Câblage Ethernet 10base2 (coaxial fin)

10base2 est un standard de câblage Ethernet spécifiant l'utilisation d'un câble coaxial 50 ohms avec un diamètre d'environ 5 mm. Il faut se souvenir d'un nombre important de règles quand on relie deux machines avec un câblage 10base2. La première est que vous devez utiliser des terminaisons *à chaque extrémité* du câble. Un terminateur est une résistance de 50 ohms qui sert à s'assurer que le signal est absorbé et non réfléchi à l'extrémité du câble. Sans terminaison à chaque extrémité vous pourriez trouver que l'Ethernet n'est pas fiable ou ne marche pas du tout. Normalement vous utilisez des `T' pour interconnecter les machines, en sorte que vous finirez par avoir quelque chose qui ressemble à ceci :

```
|==========T=============T=============T==========T==========|
           |             |             |          |
           |             |             |          |
         _ _ _         _ _ _         _ _ _      _ _ _
         |   |         |   |         |   |      |   |
         _ _ _         _ _ _         _ _ _      _ _ _
```

Les `|' à chaque extrémité représentent une terminaison, les `======' représentent une longueur de câble coaxial avec des prises BNC en bout et les `T' représentent un connecteur en `T'. Gardez la longueur de câble entre les connecteurs en `T' et les cartes Ethernet aussi courte que possible, l'idéal étant que ces connecteurs soient branchés directement sur la carte Ethernet.

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

13.4. Câblage Ethernet à paires torsadées

Si vous n'avez que deux cartes Ethernet avec paires torsadées et que vous voulez les relier, vous n'avez pas besoin de répartiteur. Vous pouvez câbler les deux cartes directement ensemble. Un schéma montrant comment faire est inclus dans le document Ethernet−HOWTO

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 14. Glossaire des termes utilisés dans ce document.

Ci−dessous une liste des termes les plus importants utilisés dans ce document.

ARP
: C'est l'acronyme de *Address Resolution Protocol* (protocole de résolution d'adresses), permettant à une machine du réseau d'associer une adresse IP à une adresse matérielle.

ATM
: C'est l'acronyme de *Asynchronous Transfer Mode* (mode de transfert asynchrone). Un réseau ATM enveloppe les données en blocs de taille standard pour pouvoir les convoyer efficacement d'un point à un autre. ATM est une technologie réseau à commutation de paquets.

client
: C'est habituellement le morceau de logiciel d'un système du côté où se trouve l'utilisateur. Il y a des exceptions, par exemple, dans le système de fenêtres X11 c'est en fait le serveur qui est avec l'utilisateur et le client qui est sur la machine distante. Le client est le programme ou l'extrémité d'un système qui utilise le service fourni par un serveur. Dans le cas de systèmes *d'égal à égal* tels que *slip* ou *ppp* le client se trouve à l'extrémité qui a initialisé la connexion, l'autre extrémité, étant considérée comme le serveur.

datagramme

Un datagramme est un paquet discret de données qui contient les adresses, et qui est l'unité de base de transmission sur un réseau IP. On peut aussi l'appeler `paquet'.

DLCI
DLCI veut dire `Data Link Connection Identifier'(identifieur de connexion de liaison de données), et est utilisé pour identifier une liaison virtuelle unique point à point via un réseau à relais de trames (Frame Relay). Les DLCI sont normalement assignés par le fournisseur de réseau à relais de trames.

Relais de trames
Frame Relay (Relais de trames) est une technologie réseau idéale lorsque l'on a un trafic de nature cahotique ou sporadique. Les coûts peuvent être réduits quand on a de nombreux clients partageant la même capacité réseau et on compte sur le fait que les clients utilisent le réseau à des instants différents.

Adresse matérielle
C'est un nombre qui identifie de manière unique un hôte sur un réseau physique au niveau de la couche accès. Par exemple : *Adresses Ethernet* et *Adresses AX.25*.

ISDN
C'est l'acronyme de *Integrated Services Digital Network*(Réseau Numérique à Intégration de Services=RNIS). Il fournit des moyens standardisés avec lesquels les compagnies de télécommunications peuvent délivrer soit de la voix soit des informations vers des clients. Techniquement c'est un réseau de données à commutation de paquets.

ISP
C'est l'acronyme de `Internet Service Provider' (fournisseur d'accès à l'Internet=FAI). Ce sont des organisations ou des sociétés qui fournissent une connexion réseau à l'Internet au public.

Adresse IP
C'est un nombre qui identifie de manière unique un hôte TCP/IP sur le réseau. Cette adresse est codée sur 4 octets et se présente habituellement sous la forme appelée "notation décimale pointée", où chaque octet est sous forme décimale, avec un point `.' entre chaque.

MSS
Le Maximum Segment Size (*MSS*) (Taille Maximum de Segment) est la plus grande quantité de données qui peut être transmise en une seule fois. Si vous voulez éviter des fragmentations MSS doit être égal à l'en−tête MTU−IP.

MTU
Le Maximum Transmission Unit (*MTU*) (taille maximum de l'unité de transfert) est un paramètre qui détermine le plus long datagramme pouvant être transmis par une interface IP sans avoir besoin d'être fragmenté en unités plus petites. Le MTU doit être plus grand que le datagramme le plus grand que vous voulez transmettre sans être fragmenté. Note : ceci protège de la fragmentation uniquement de manière locale, d'autres liens sur le chemin peuvent avoir un MTU plus petit et les datagrammes seront fragmentés à cet endroit. Les valeurs typiques sont de 1500 octets pour une interface Ethernet, ou de 576 octets pour une interface SLIP.

route
La *route* est le chemin que les datagrammes suivent à travers le réseau pour atteindre leur destination.

serveur
C'est habituellement le morceau de logiciel ou l'extrémité d'un système éloigné de l'utilisateur. Le serveur fournit un service vers un ou plusieurs clients. Des exemples de serveurs sont *ftp*, *Networked File System* (NFS), ou *Domain Name Server* (DNS). Dans le cas de systèmes *égal à égal* comme *SLIP* ou *PPP* le serveur est considéré comme étant l'extrémité de la liaison qui est appelée et l'extrémité appeleante est le client.

fenêtre
La *fenêtre* (window) est la plus grande quantité de données que l'extrémité réceptrice peut accepter à un certain moment.

Chapter 15. Auteurs :

15.1. Actuels

Joshua D. Drake

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

15.2. Passés

Terry Dawson Allessandro Rubini

Ceci vous a intéressé? Pourquoi ne pas donner 2,50 dollars?

Chapter 16. Copyright.

Information de copyright

Note du traducteur

Voir les autres HOWTO traduits en français. Lire également le livre «Administration réseau sous Linux, éditions O'Reilly». Enfin voyez le site www.linux−france.com où vous trouverez de très bons articles décrivant en détail différents points évoqués dans ce document.