

Administration sous UNIX

Chapitre 1 : Introduction

I - UNIX

1) Un peu d'histoire

Années 50-60 :

- mainframes
- ils ne savent pas communiquer
- les SE (Système d'Exploitation) sont rudimentaires
- chaque système à son propre langage
- aucune notion de compatibilité ascendante
- pour chaque nouvelle machine il faut refaire les programmes
=> chaque mise à jour à un coût prohibitif

En 1965, le MIT et le laboratoire Bell conçoivent un système novateur : MULTICS (MULTicomplexed Information and Computing Service). Ce système repose sur le principe du temps partagé entre processus, un assemblage de briques de base et compatibilité ascendante, ce fut un échec.

En 1969, Bell se retire du projet, cependant quelques ingénieurs de Bell continuent leur travaux (Ken Thomson, Denis Ritchie, Doug McIlroy et JF Ossanna). Ils posent les bases d'un système qui fera la base d'UNIX. Thomson et Ritchie commencent à implémenter leur système de fichier (SF). Il manque beaucoup d'outils pour faire un vrai système d'exploitation. Thomson commence à développer ses outils :

- un shell
- les outils pour copier-déplacer les fichiers
- un éditeur de texte
- un assembleur
- ...

UNIX (à mettre en opposition avec MULTICS) était né, le développement s'accélère et Bell Labs commence à comprendre l'intérêt d'un tel système.

Ritchie invente le 'C', Thomson ré-implante UNIX en 'C'.

En 1976, les premiers cours d'UNIX sont données dans une université.

2) Pourquoi utiliser UNIX (par rapport à windows)

- UNIX a été pensé dès le début comme un système multi-utilisateurs
- UNIX est un système réseau
- UNIX a une conception modulaire
- UNIX à l'avantage de l'ancienneté et il fonctionne toujours très bien
- la philosophie d'UNIX a toujours été ouverte, avec un respect des standards.

L'ouverture est poussée à son paroxysme avec le système GNU lancé par Richard Stallman en 1983.

Aujourd'hui UNIX est synonyme de logiciel libre, même si la vérité est plus complexe

- Une documentation abondante existe
- Il existe de très bonne implémentations open source
- UNIX est toujours à la pointe des systèmes d'exploitation

1) Linux

Linux a été inventé par un étudiant finlandais (Linus Torvalds) en 1991.

Il ne s'agit pas d'un SE mais d'un noyau, en conséquence Linux n'est pas utilisable seul.

Rappel (programmation sous Unix) : un SE est une couche d'abstraction entre le matériel et l'utilisateur. Pour simplifier, le SE gère les ressources. Le noyau est la couche la plus basse du SE, c'est un composant logiciel qui est le seul à interagir directement avec le matériel. Un système composé uniquement d'un noyau n'est pas utilisable. Le véritable SE est GNU (GNU is not UNIX) et non Linux. GNU est le projet de développement d'un SE intégralement libre. A l'origine il devait utiliser le noyau Hurd, mais ce dernier est toujours en développement aujourd'hui (depuis 1990).

Avec l'arrivée du noyau Linux en 1991 et avec l'utilisation des outils GNU on obtient un SE. Le SE est GNU/Linux mais par abus de langage on utilise souvent Linux. Une distribution Linux correspond à un SE, elle contient le noyau Linux, les outils GNU ainsi que les outils de haut niveau pour maintenir le système.

II - Qu'est-ce qu'un administrateur?

1) Son rôle

Un administrateur doit permettre le bon fonctionnement des outils de travail et la bonne cohabitation des personnes.

Les aspects relationnels sont importants. Il gère aussi les ressources matériels et logicielles. Il met en place le réseau : choix des configurations matériels, maintenance, surveillance, formation...

2) Aspect relationnel

3 attitudes possibles :

- esclave
- tortionnaire
- interlocuteur

Évidemment la meilleure attitude est la 3ème mais elle nécessite du temps, de la diplomatie, et la justification de ses démarches pour que les utilisateurs comprennent le sens de ces démarches.

3) Aspect technique

L'administrateur doit :

- assurer la maintenance logicielle
- assurer la maintenance matérielle
- rester aux aguets :
 - nouveaux besoins
 - nouvelle technologie
 - surveillance (usage illicite)
 - protection (sécurité, virus)
- gérer les utilisateurs (ouverture/fermeture de compte, droits d'accès...)
- gérer les sauvegardes
- gérer les licences
- inventorier les moyens logiciel et matériel
- gérer les imprimantes, les consommables
- gérer les données (obsolètes, ...)

III - Objectifs du cours

1) Ce dont on ne va pas parler

- aspect relationnel
- différences avec l'administration sous windows

2) Ce dont on va traiter

- administrer une station de travail sous UNIX
- administrer un réseau de station de travail

Chapitre 2 : Choix et installation d'une distribution Linux

I - Préliminaire

La rédaction d'un cahier des charges est nécessaire.

Il sert à planifier le déploiement dans le temps, à évaluer la liste des services que peut fournir la machine, de base de travail.

Il doit être validé par les utilisateurs potentiels.

La première question à se poser est à qui va servir la machine.

Cela détermine plusieurs choses :

- le niveau d'accessibilité de l'interface
- le nombre d'utilisateurs potentiels, ce qui influe sur :
 - l'espace disque nécessaire
 - la technologie utilisée pour l'authentification
- la politique de sécurité

exemple 1 : « La machine va servir aux personnes de passage à la cafétéria pour lire leur mail. »

=> sécurité critique

=> grand nombre d'utilisateurs

=> novices, accentuer la simplicité

exemple 2 : « La machine va servir à un unique étudiant effectuant des travaux sur la simulation d'écoulement liquide. »

=> sécurité normale

=> peu d'utilisateurs

=> utilisateurs expert

La seconde question est à quoi elle va servir.

Cette question permet de déterminer les besoins logiciels et matériels. Une machine destinée à fournir un terminal web ne requiert pas les mêmes besoins qu'une machine destinée à faire tourner des applications de simulation complexe. C'est la réflexion la plus importante à mener. Cette phase se fait en concertation avec les utilisateurs potentiels, c'est l'expression de leurs besoins.

Ex1 : outils minimaux (interface graphique, firefox)

faible nécessité d'espace disque

ex2 : outils de développement

grand besoin d'espace disque

la 3ème question est si elle sera connectée à un réseau. Cette question détermine la politique de sécurité. De plus elle complète les questions précédentes car un certain nombre de services peuvent déjà être présents sur le réseau :

authentification

espace disque

accès à internet

...

II - Choix d'une distribution

Une distribution est un ensemble complet de logiciels permettant d'utiliser GNU/Linux. Un grand nombre de distributions existent (plusieurs centaines), mais il y en a moins d'une dizaine de distributions majeures (Debian, Slackware, Red Hat, Fedora, Mandriva, Suze, Ubuntu). Les différences entre les distributions se situent au niveau de :

- leur système de package
- les outils d'installations et de configurations
- « l'emballage » (boite, documentation, support technique, ...)
- parfois l'utilité (certaines distributions sont spécifiques à un usage donné)
- utilisation sur une vieille machine
- calcul distribué, scientifique
- utilisation sur des ordinateurs spécifiques (machintosh, station SUN, PS3)

Il est possible d'arriver au même résultat quelque soit la distribution, mais on peut gagner du temps en choisissant la bonne au départ. La notion de package est importante dans le monde UNIX. Un package est une archive contenant l'ensemble des fichiers nécessaires au bon fonctionnement d'un programme donné. Il contient aussi d'autres informations comme la liste des dépendances, c'est à dire les autres logiciels (packages) dont a besoin le programme pour fonctionner. Il existe plusieurs standards de packages :

- RPM (Red Hat Package Manager pour Red Hat, Fedora, Mandriva, Suze)
- DEB (Debian, Ubuntu)
- TGZ (Stackware)
- sources (Gentoo)

Il n'y a pas de différences fondamentales entre un .rpm, .deb, .tgz

Particularité des sources : dans la plupart des distributions, les paquets contiennent de logiciels déjà compilés. Or l'architecture des ordinateurs est loin d'être uniforme. Dans une distribution basée sur les sources, les programmes sont compilés lors de leur installation. Cela permet de tenir compte des spécificités matérielles de la machine au prix du temps de compilation.

Remarque : La stackware permet facilement l'utilisation des sources après son installation, c'est plus délicat pour les autres distributions, mais faisable.

Chapitre 3 : UNIX et réseau

I - Fonctionnement d'un réseau

Un réseau fournit un moyen de communiquer entre plusieurs ordinateurs. Pour communiquer, les ordinateurs ont besoin de se mettre d'accord sur la façon de communiquer. On appelle protocole un « langage » utilisable par un ordinateur. Un certain nombre de normes existent et permettent l'interopérabilité des systèmes et ordinateurs.

1) Le modèle TCP/IP

Le modèle TCP/IP (Transmission Control Protocol / Internet Protocol) fournit une modélisation du fonctionnement logique d'un réseau. Il est basé sur la notion de couches. Il permet aux applications de s'abstraire de processus physique qui a lieu pour la transmission de messages.

4	Couche application
3	Couche transport
2	Couche internet
1	Couche d'accès réseau

- La couche application regroupe l'ensemble des logiciels accédant au réseau et tout les mécanismes d'authentification, session.

Ex : navigateur web, client mail...

- La couche transport assure l'acheminement des données à la bonne application. Elle contient les informations contenant l'état de la transmission. Elle assure le multiplexage et démultiplexage des informations à cet effet la notion de port. Chaque application d'une machine écoute sur un port, qui est identifier par un numéro.
- La couche internet assure la segmentation des données et l'adressage
- La couche d'accès réseau est celle de plus bas niveau. Elle est essentiellement matérielle (ex : Ethernet), elle assure l'acheminement des données sur ma liaison, elle fait les conversions analogique/numérique, elle possède des systèmes de contrôle d'erreur.

Passage des couches

application				données	
transport			En tête np	données	
internet		En tête IP	Données IP		
Accès réseau	En tête	trame			Fin trame

Le protocole TCP est un protocole fiable (on sait si le message est reçu ou pas), TCP s'occupe de renvoyer les paquets ou pas. Cependant pour certaines applications tel que la diffusion de flux multimédia ou encore VOIP (Voice Over IP), si un paquet est perdu, il est inutile de le renvoyer. Pour de tels applications on préférera utiliser le protocole de la couche transport UDP. Ce protocole ajoute à IP juste la notion de ports. Quand on envoie un message via UDP, on n'est pas sûr qu'il soit reçu.

2) Réseau local

Un réseau local (LAN - Local Area Network) est un ensemble de machines connectées par un réseau. L'amplitude géographique d'un tel réseau est assez faible (max quelques centaines de mètre). Un réseau permet de partager un certain nombre de ressources. Il est souvent indispensable.

Classiquement, un réseau va fournir les services suivants :

- partage de fichiers
- partage de la connexion internet
- authentification centralisée
- messagerie (e-mail, IM)

Un réseau c'est aussi structurer logiquement les ressources matérielle et les moyens humains.

Internet : la plupart des services fournis par un réseau local, peuvent l'être aussi par internet.

Internet est un ensemble de solutions techniques permettant d'interconnecter des réseaux locaux cependant un LAN n'est pas un milieu hostile en réseau local rapide.

3) Quelques définitions

On appelle service (daemon dans le monde UNIX) un logiciel fournissant des fonctionnalités à d'autres logiciels. Exemple : service web, service mail, service ftp...

On appelle serveur une machine faisant tourner un ou plusieurs services. Par abus de langage on parle aussi de serveur web, serveur mail...

Un serveur est en écoute sur un port donné.

A la réception d'un message, il va :

- l'analyser
- effectuer l'action demandée
- composer la réponse
- envoyer la réponse au client

II - L'authentification

1) objectifs

Le but du mécanisme d'authentification est de reconnaître les utilisateurs. Ceci est un problème clé de tout réseau car chaque utilisateur n'a accès qu'à un sous-ensemble restreint des ressources du réseau. Identifier correctement les utilisateurs est un des défis de l'administrateur. Sous UNIX, un utilisateur est identifié par plusieurs informations :

- un nom
- un mot de passe
- un identifiant numérique (UID)
- un identifiant de groupe (GID)
- un répertoire personnel (home directory)
- un shell par défaut

Le nom et le mot de passe servent au mécanisme d'authentification (login). L'UID et le GID servent à la manipulation de fichiers. Le répertoire personnel et le shell par défaut donnent des informations nécessaires à la phase d'authentification.

2) La phase d'authentification

Un utilisateur rentre un login et un mot de passe par le biais d'un programme de login. Ces informations sont transmises au système d'authentification. Il compare le mot de passe saisi à celui présent dans sa base de données. Il transmet les informations 'succès' ou 'échec' au programme de login. En cas de succès le shell par défaut est exécuté avec les privilèges de l'utilisateur et il est placé dans son répertoire personnel.

3) Principe de fonctionnement

Un système d'authentification fonctionne comme une boîte noire :

- Système basique :
Sur un système non connecté à un réseau, les informations d'authentification sont stockées dans les fichiers '/etc/passwd' et '/etc/shadow'.
'passwd' contient les informations de nom d'utilisateur, de shell par défaut, du répertoire personnel, UID et GID.

Exemple : foo:x:1001:1001:,,,:/home/foo:/bin/bash
(nom d'utilisateur, shadow passwd, UID, GID, home directory, shell)
'shadow' contient les mots de passe cryptés des utilisateurs

Exemple : foo:\$1\$UBPQuBy\$aSJQrauJNQ9JPtyhlzQup/:13881:0:99999:7:::

Quand un utilisateur configure son mot de passe, le système va le crypter généralement avec une méthode non réversible. Quand l'utilisateur se logue, le système d'authentification va crypter le mot de passe donné avec la même méthode, si le résultat du cryptage est identique à celui stocké dans la base, alors le mot de passe est bon.

/etc/passwd est lisible par tous les utilisateurs.

/etc/shadow n'est accessible que par le root.

Si un utilisateur a accès à plusieurs machines, il doit changer de mot de passe sur toutes les machines ; s'il change de groupe, il doit changer de groupe sur toutes les machines.

- Système avancé :
A partir d'un certain nombre d'utilisateurs, on met en place un système centralisé d'authentification. Cela résout en partie des problèmes liés à l'accès direct aux machines.

III - Notions de permissions UNIX

1) Tout est fichier

UNIX propose une abstraction généralisée de l'ensemble des ressources d'une machine : tout est vu comme un fichier. Évidemment les fichiers « conventionnels » sont des fichiers, de façon plus étonnante, les périphériques sont aussi des fichiers. Pas si étonnant que ça, si on considère une souris : le système lit les informations envoyées par la souris. Ces informations sont une suite de '1' et de '0' ; un fichier est lui aussi une suite de '1' et de '0'. Ce modèle permet un accès uniforme quelque soit la ressource. La partie bas niveau est cachée dans la partie matérielle. Les périphériques apparaissent

tous dans le système de fichier virtuel '/dev'.

2) Fichiers et permissions

Un fichier est un objet dans lequel on peut lire et écrire des données. Il est la propriété d'un utilisateur et d'un groupe. L'utilisateur ne faisant pas forcément partie du groupe. On appelle 'permission' un ensemble de règles limitant l'accès à une ressource donnée. Sur les fichiers il existe trois types de permissions :

- les droits de lecture
- les droits d'écriture
- les droits d'exécution

Un utilisateur peut se voir accorder tel ou tel autre permission. Les permissions peuvent aussi s'appliquer à un groupe. Toutes ces informations sont stockées au niveau du système de fichier (cf inoed, cours de programmation sous UNIX). Cependant, stocker ces informations individuellement pour chaque fichiers nécessite un espace disque important. UNIX propose donc un modèle à taille fixe : les permissions du propriétaire, du groupe et des autres. Pour chacune de ces catégories on peut fixer les trois types de droits.

Exemple : la commande 'ls -la' donne toutes les informations disponibles sur un fichier

Exemple : -rw- r-- r-- 1 matteo matteo 13567 2006-01-29 19:00 main.tex

(permission, nombre de liens, utilisateur, groupe, taille, date de modification, nom du fichier)

Exemple : d-rw- r-- r-- 1 matteo matteo 2006-01-29 19:00 rep

(nombre de sous répertoire)

drwxrwxrwx

de droite à gauche:

- exécution autre
- écriture autre
- lecture autre
- exécution groupe
- écriture groupe
- lecture groupe
- exécution propriétaire
- écriture propriétaire
- lecture propriétaire
- si c'est un répertoire

On peut agir sur les permissions d'un fichier à l'aide de la commande 'chmod'. Le changement du propriétaire et du groupe se fait avec la commande 'chown'. Root est au-dessus des droits de lecture et d'écriture, mais pas d'exécution.

3) Le partage des fichiers

C'est rendre possible l'échange de fichiers entre plusieurs ordinateurs. C'est aussi faire en sorte que les données d'un utilisateur soient accessibles quelque soit le poste utilisé. L'avantage est que les données sont sur un serveur unique (ou pas) ce qui facilite entre autres les sauvegardes, mais il faut protéger le serveur physiquement. On peut utiliser plusieurs méthodes :

- FTP : juste un mécanisme de transfert de fichiers. Un utilisateur peut rapatrier des données en local et renvoyer les versions modifiées sur le serveur.
- NFS : un système de fichier réseau. Un partage réseau est monté de façon transparente. On a une abstraction totale du mode d'accès.
- Samba : implémentation libre du protocole SMB qui est le protocole du monde Microsoft. Avantage : lorsqu'on a un réseau hétérogène avec des machines sous Windows. Inconvénient : les droits ne sont pas directement transposables.

Chapitre 4 : L'organisation logique du système de fichiers

I - L'arborescence « classique »

Sous Unix, on trouve une organisation logique des données

/ : la racine

/bin : exécutable essentiels au système employé par tous les utilisateurs

/boot : fichiers de chargement du noyau, dont le chargeur d'amorce

/dev : points d'entrées des périphériques

/etc : fichiers de configurations

/X11 : fichiers de configuration de Xwindow

/init.d : scripts de démarrage

...

/home : répertoires personnels des utilisateurs

/lib : bibliothèques standards

/mnt : les points de montage des partitions temporaires (CD-ROM, clés USB...) (/média sous

Débian)

/root : répertoire personnel de root

/sbin : exécutable essentiels au système mais dédiés à l'administration ('add user')

/tmp : fichiers temporaire

/usr : hiérarchie secondaire

/X11R6 : est réservé à X

/bin : majorité des fichiers binaires et commandes utilisateurs

/include : en-tête pour les programmes C et C++

/lib : contient la plupart des bibliothèques partagées

/local : données relative aux programmes installées sur la machine local par le root

/bin

/include

/lib

/sbin

/src : source des programmes locaux

/sbin : fichiers binaires non essentiels au système réservé à l'administration

/share : données non dépendante de l'architecture

/src : fichiers de code sources ; c'est ici qu'on trouve les sources du noyau

/var : données versatiles telles que les fichiers de bases de données, les fichiers journaux, les fichiers du spooler d'impression, mails en attente...

II - Les liens

Les liens permettent de donner plusieurs nom à un fichier. Il existe deux types de liens, les liens symboliques et les liens « dur » (physique).

1) Liens symboliques

Un lien symbolique est un pointeur vers un fichier quelconque. On distingue que le fichier est un lien symbolique par la présence d'un '@' devant le nom. Si on supprime le fichier pointé par le lien, le lien n'est pas supprimé, mais il pointe nul part (il apparaît en rouge dans le résultat d'un 'ls'). On peut faire un lien symbolique vers un fichier d'un support amovible.

Syntaxe : `ln -s nom_fichier_à_pointer nom_lien`

2) Liens physique

Un lien physique donne un deuxième nom à un fichier. Les données du fichier ne sont présentes qu'une seule fois sur le support de stockage mais plusieurs fois dans l'organisation logique. On ne peut pas faire un lien physique vers un fichier qui est sur un système de fichier différent.

Syntaxe : `ln nom_fichier_à_pointer nom_lien`

3) Quelques commandes de gestion

Pour gérer les répertoires on a les commandes suivantes :

`mkdir nom_fichier` (création d'un répertoire)
`rmdir nom_fichier` (suppression d'un répertoire vide)
`mv repertoire repertoire_d'accueil` (déplacement d'un répertoire)

Pour gérer les fichiers :

`touch nom_fichier` (création d'un fichier vide)
`more nom_fichier` (visualisation d'un fichier page à page)
`rm nom_fichier` (supprime un fichier)
`mv nom_fichier repertoire_d'accueil` (déplacement d'un fichier)
`mv nom_fichier nouveau_nom_fichier` (renommer un fichier)
`cp nom_fichier repertoire_d'accueil [/nouveau nom]` (copie d'un fichier)

4) Définition des points de montage statique

Les fichiers '/etc/dstab' permettent de configurer des points de montages statiques prédéfinies. Lors de l'installation du système de fichier, un fichier par défaut est créé. Ce fichier contient les définitions des points de montage spécifiés lors de l'installation ainsi que ceux pour le lecteur de disques amovibles. Dans ce fichier on trouve aussi les partitions de swap. Le format de fichier est le suivant :

<fichier spécial périphérique> <point de montage> <type de système de fichier> <option> <dump> <ordre fsck>

- fichier spécial périphérique : indique le périphérique dans '/dev'
- point de montage : indique le point de montage dans la hiérarchie logique
- type de système de fichier : indique le type de système de fichiers, ex : ext2, ext3, iso 9960 (CD-ROM), nfs, ntfs, reiserfs, smbfs, vfat (FAT 32)

Si le périphérique est une partition de swap : 'swap'.

On peut spécifier 'auto' pour laisser le noyau choisir le bon format, le succès n'est pas sûr.

- Options : liste des options séparées par des ','. Les options dépendent du type de système de fichier, mais certaines sont connues
 - 'noauto' : le système de fichier ne sera pas monter automatiquement au démarrage ('auto' pour qu'il soit monter automatiquement)
 - 'user' : pour qu'un utilisateur lambda puisse monter le système de fichier ('nouser' pour le contraire)

- 'owner' : le propriétaire du périphérique peut monter le système de fichier (spécifique à Linux)

Parmi les options classique on trouve :

- 'exec' : autorise l'exécution des binaires (sur un système de fichier vfat tous les fichiers deviennent binaire)
- 'rw' : pour autoriser l'écriture
- 'rs' : lecture seul

On peut spécifier comme valeur unique :

- 'defaults' : dans ce cas on a 'rw, suid, dev, exec, auto, nouser, async'
- dump : indique au logiciel 'dunp' la fréquence de sauvegarde (0 pour jamais)
- ordre fsck : indique l'ordre dans lequel 'fsck' doit analyser les systèmes de fichiers

Pour monter un système de fichiers, il n'est pas nécessaire qu'il soit référencer dans fstab (pour peu qu'on soit root). Mais s'il est présent dans fstab, on peut utiliser une syntaxe allégée : `<mount> point_de_montage`

Remarque :

- On ne peut pas démonter un système de fichier qui a au moins un fichier actuellement ouvert. La commande 'lsof' permet de connaître tous les fichiers actuellement ouvert.
- Il existe un daemon particulier HAL (Hardware Abstraction Layer), ce service permet, entre autre, de gérer les connexions à chaud de périphériques tel que les clés USB.

Chapitre 5 : Gestion des processus

I - Visualiser les processus

La commande 'ps' permet de visualiser les processus courants sur le système. Pour lister tous les processus, on utilise l'option 'e' ou 'A'. La commande affiche plusieurs informations :

- PID : l'identificateur de processus
- TTY : le terminal sur lequel s'affiche le processus
- TIME : la durée de traitement du processus
- CMD : le nom de l'exécutable
- ...

Il existe de nombreuses options d'affichage. Selon le système Unix utilisé, l'affichage peut varier. L'outil 'top' permet de visualiser les processus en temps réel, avec notamment leur consommation CPU et mémoire.

II - La priorité

1) Notions sur la priorité

Unix étant un système d'exploitation à temps partagé entre processus, chaque processus a une priorité d'exécution. En théorie, le noyau exécute le processus qui a la plus forte priorité, si deux processus ont la même priorité, ils s'exécutent à tour de rôle. Dans ce contexte, certains processus ne s'exécuteraient jamais. En pratique, c'est plus compliqué que cela. Chaque processus a deux niveaux de priorité, un niveau utilisateur et un niveau noyau. Le niveau utilisateur est le 'nice' (politesse). Plus ce niveau est faible, plus le processus est prioritaire. La politesse va de '-20' (le plus prioritaire) à '19' (le moins prioritaire). Par défaut le 'nice' est à 0. La véritable priorité d'exécution est calculée par le noyau de façon dynamique. Elle tient compte de la politesse mais aussi d'autres paramètres comme la durée depuis la dernière fois où le processus a pu s'exécuter.

2) Changer la politesse

La commande 'nice valeur commande' permet de lancer un programme avec une certaine politesse. La commande 'renice valeur -p pid' permet de changer la politesse du processus pid qui est en cours d'exécution.

3) Arrêter un processus

Pour interrompre un processus, on lui envoie un signal via la commande 'kill'. 'kill pid' envoie le signal 'SIG TERM' au processus identifier par pid. Ce signal lui demande « gentiment » de s'arrêter. Pour forcer l'arrêt d'un processus, on utilise la commande 'kill -9 pid', cette commande tuera la plupart des processus (les processus vitaux sont protégés). Evidemment on ne peut tuer que les processus qui nous appartiennent sauf si on est root.

III - Exécuter en tâche de fond

Par défaut quand on lance un processus celui-ci garde la main sur le shell et on doit attendre qu'il se termine avant de pouvoir rentrer des commandes à nouveau. Il est possible de lancer un processus en tâche de fond, dans ce cas le processus s'exécute mais il vous laisse le shell pour pouvoir exécuter d'autres commandes. Pour lancer un processus en tâche de fond, on ajoute un '&' à la fin de la commande. Pour basculer un processus déjà lancé en tâche de fond on peut faire un 'ctrl+z' pour le mettre en attente puis on tappe 'bg'.

Remarque : un processus en tâche de fond gardera toujours comme sortie standard notre terminal. Si notre processus est très bavard, on va être submerger par les messages.

Chapitre 6 : Configuration d'un pare-feu

I - Introduction

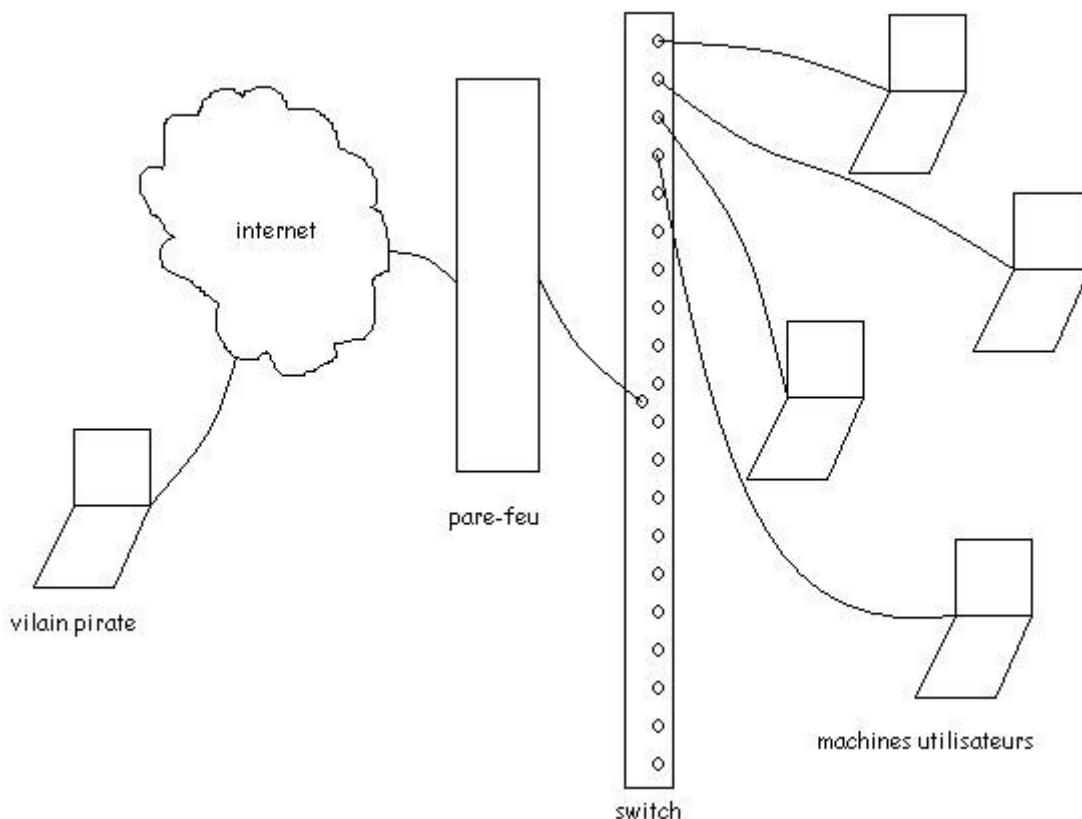
1) A quoi sert un pare-feu ('firewall' en anglais)

A rien si nous vivions dans un monde parfait. Dans un monde parfait les logiciels n'ont pas de bugs et surtout personne ne cherche à s'introduire dans notre machine. Un pare-feu est une application qui filtre les connexions entrantes et sortantes d'une machine. Pour rappel, un service (web, e-mail, ...) est en écoute sur un port donné. S'il n'y a aucun système de filtrage, n'importe quel machine relié en réseau peu potentiellement accéder à ce service. Mais on peut vouloir restreindre ce service à seulement quelques machine du réseau afin d'augmenter la sécurité. Normalement si le service est bien sécurisée on peut se passer de pare-feu. On parle aussi de pare-feu pour désigner une machine qui protège une partie du réseau du reste du monde. Cette machine peut très bien être un PC (classique avec au moins deux cartes réseaux) ou bien un matériel spécifique conçu pour.

2) Principes

Architecture

L'architecture simplifier du réseau est la suivante :



Ici les machines utilisateurs peuvent communiquer entre elles, sans restrictions. Mais pour pouvoir accéder à internet, le pare-feu doit autoriser la connexion. De la même manière, le pare-feu empêche les connexions extérieures d'entrer sur le réseau sauf si elles sont explicitement autorisées (cf VI - 4)). Un pare-feu logiciel peut très bien être installé sur une machine terminale.

Type de firewall

Il existe trois types de pare-feu :

- Filtre des paquets : le pare-feu analyse les paquets IP. On peut filtrer suivant les adresses de sources et de destinations. Sur le protocole transport utilisé (TCP, UDP, ou ICNP). Sur les numéros de ports.
- Mémoire d'état : Ajoute au précédent le fait que le pare-feu conserve la mémoire des connxions établies.
- Pare-feu applicatif : filtrage sur la couche application.

Nous nous intéresserons aux deux premiers.

3) Fonctions annexes

Un pare-feu (au sens large) peut aussi intégrer des fonctions supplémentaires :

- La mise en cache : le 'pare-feu' peut mémoriser les dernières données consultées sur internet et ainsi ne pas avoir à effectuer de nouvelles requêtes (proxy)
- Filtrage au niveau du contenu : mots clés, vidéos, sons exécutables...
- Détections d'intrusions : repérer que quelqu'un cherche à pénétrer le système et bloquer automatiquement sont adresse IP. Un logiciel de pare-feu qui intégrerait ces fonctions irait à l'encontre du principe Unix, « un logiciel ne fait qu'une seule chose mais il la fait bien ». Par contre sous Unix, il existe des passerelles entre les pare-feu et les systèmes de détections d'intrusions par exemple.

II - Comment fonctionne TCP et UDP?

Pour bien configurer un firewall, il est nécessaire de bien comprendre le fonctionnement de TCP et d'UDP.

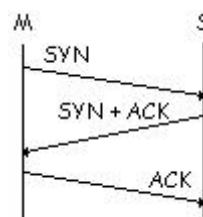
1) TCP

TCP est un protocole qui fonctionne en mode connecté et est « fiable ». L'en tête d'un paquet TCP contient des drapeaux (valeur binaire) indiquant la nature du paquet. Les drapeaux sont :

- URG : signal la présence de données urgente (les routeurs n'en tiennent plus compte)
- ACK : signal que le paquet est un accusé de réception (ACKnowledgement)
- PSH : données à envoyer tout de suite (PuSH)
- RST : rupture anormal de connection (ReSeT)
- SYN : demande de SYNchronisation ou établissement de connection
- FIN : demande de FIN de connection

Établissement de connection :

Supposons qu'une machine M cherche à initialiser une connection avec un serveur S. Une fois la connection établie, M et S peuvent communiquer. On utilise le drapeau ACK pour confirmer la bonne réception des paquets. Des mécanismes sont utilisés pour assurer le bon acheminement des données.



Pour terminer la connection, on envoie un paquet avec le drapeau FIN.

2) UDP

UDP fonctionne en mode non connecté et est aussi considéré comme non fiable. On ne peut pas initier de connexion pour transmettre des informations, on envoie juste le paquet en espérant qu'il arrive. C'est à la couche application de s'assurer que les données sont bien arrivées.

III - Sous Unix - GNU/Linux

Nous allons seulement nous intéresser aux pare-feu sous GNU/Linux

1) Quel pare-feu sous Linux?

La question ne se pose pas. En effet le logiciel de filtrage est directement intégré au noyau Linux, il se nomme Netfilter. Pour administrer Netfilter, on utilise la commande *iptables*. Netfilter s'utilise aussi bien pour une machine terminale que pour une passerelle.

2) Les tables

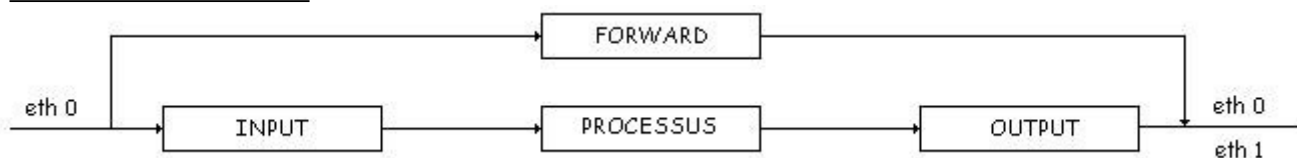
Netfilter se base sur des tables pour fonctionner. Une table est composée de chaînes qui sont elles-mêmes composées de règles. La table 'filter' contient les chaînes pour le filtrage des paquets. C'est la table par défaut. La table 'nat' contient les informations de translation d'adresse. La table 'mangle' sert pour transformer les paquets (utile pour le QOS par exemple). Dans la suite de cette section, on ne s'intéresse qu'à la table filter.

3) Les chaînes

La notion de chaîne est importante pour Netfilter. En effet les règles de filtrage qu'on va appliquer seront contenues dans ces chaînes. Il existe trois chaînes principales :

- input : contient les règles à appliquer sur les paquets entrant de la connexion réseau
- output : contient les règles à appliquer sur les paquets sortant de la connexion réseau
- forward : contient les règles à appliquer sur les paquets traversant le système. Cette chaîne est utilisée pour les machines faisant office de passerelle.

Articulation des chaînes :



Un paquet entrant à destination d'un processus local à la machine devra passer la chaîne INPUT, si la chaîne reconnaît le paquet comme légitime, il sera transmis au processus concerné. C'est la même chose pour la chaîne OUTPUT. Par exemple en supposant que notre machine propose un service web (port 80) et que le service soit public. Il y aura dans 'INPUT' une règle qui indiquera que les paquets TCP à destination des ports 80 sont autorisés et une règle 'OUTPUT' pour que les paquets en provenance du port 80 soit autorisés (la réponse du serveur).

NB : On peut définir ses propres règles pour améliorer la lisibilité des règles (un peu comme on ferait des fonctions dans un programme).

4) La définition des règles

Netfilter fonctionne avec un principe un peu particulier : il n'y a pas de fichier de configuration, à la place on utilise la commande iptables pour gérer les règles.

Politique par défaut :

On peut spécifier pour chaque chaîne quelle est la politique par défaut, c'est à dire qu'il faut autoriser ou rejeter les paquets que ne correspondent à aucune règle. Pour définir la politique par défaut :
`iptables -P <chaîne> <cible>`

Exemple : `iptables -P INPUT DROP`

=> par défaut on rejete les paquets entrant

La cible est généralement soit 'ACCEPT' pour accepter le paquet, soit 'DROP' pour le rejeter.

Ajout de règle :

Bien qu'il n'existe pas de fichier de configuration, l'ordre des règles est importante. En effet, lorsqu'un paquet est à analyser, Netfilter regarde règle après règle jusqu'à trouver une règle qui correspond un paquet.

Syntaxe : `iptables -A <chaîne> <description de la règle> [option] -j <cible>`

ou `iptables -I <chaîne> <numéro de règle> <description> [option] -j <cible>`

Si on utilise -A alors la règle sera ajouté à la fin de la chaîne, -I permet de spécifier un point d'insertion.

Quelques options pour décrire les règles :

-p <protocole> : indique le protocole (tcp, udp, icmp ou all)

-s <source> [masque] : spécifie l'adresse de l'émetteur du paquet, 'masque' est pour spécifier une plage d'adresse

-d <destination> masque : spécifie la destination du paquet

-i <interface> : spécifie sur quelle interface d'entrée la règle s'applique (eth0, uniquement pour la chaîne INPUT)

-o <interface> : spécifie sur quelle interface de sortie la règle s'applique (eth0, uniquement pour la chaîne OUTPUT)

--source-port <port> [:port] : indique un port ou une plage de ports d'origine '--sport' est un alias. Fonctionne qu'avec '-p tcp' ou '-p udp'

--destination-port <port> [:port] : indique un port ou une plage de ports d'origine '--dport' est un alias. Fonctionne qu'avec '-p tcp' ou '-p udp'

--tcp-flags <masque> <comp> : regarde les drapeaux tcp positionnées. <masque> est la liste des drapeaux à analyser (séparé par des ,) et <comp> est la liste des drapeaux qui doivent être positionnés parmi ceux de <masque>

Exemple :

```
iptables -A INPUT -p tcp --tcp-flags SYN, ACK, FIN, RST SYN -j ACCEPT
```

Le drapeau SYN doit être positionné. ACK, FIN, RST doivent être à 0.

--SYN équivalent à '--tcp-flags SYN, ACK, FIN, RST SYN'

NB :

- La plupart des options peuvent être inverser (ex : "n'est pas le protocole udp") en ajoutant !
 - Ex : -p !udp
- Cette liste n'est pas exhaustive. Cf 'man iptables'

Un exemple :

```
iptables -A INPUT -p tcp -dport 80 -j ACCEPT
```

Supprimer une règle :

Pour supprimer une règle, on utilise la commande :

```
iptables -D <chaîne> <numéro de la règle> ou iptables -D <chaîne> <description>
```

Lister les règles :

```
iptables -L
```

Vider les chaînes :

```
Si on souhaite supprimer toutes les règles : iptables -F [chaîne]
```

IV - Translation d'adresse, NAT

NAT = Network Address Translation

1) Objectifs et raison d'être

La version 4 du protocoles IP (Ipv4) est celle qui est la plus répandu sur Internet, le remplaçant étant Ipv6 qui se déploie (trop) lentement. Les adresses dans Ipv4 sont codées sur 4 octets (32 bits), ce qui nous donne $2^{32} = 4\ 294\ 967\ 296$ adresses possibles, ce qui semblait largement suffisant dans les années 70 lorsque le protocole fut conçu. Hors avec l'explosion d'Internet, le nombre d'adresses disponible est insuffisant.

Deux solutions : passage en Ipv6 ou la translation d'adresses. Le principe de la translation d'adresses est de permettre la correspondance entre plusieurs machines adressées en interne avec 1 ou plusieurs adresses publiques.

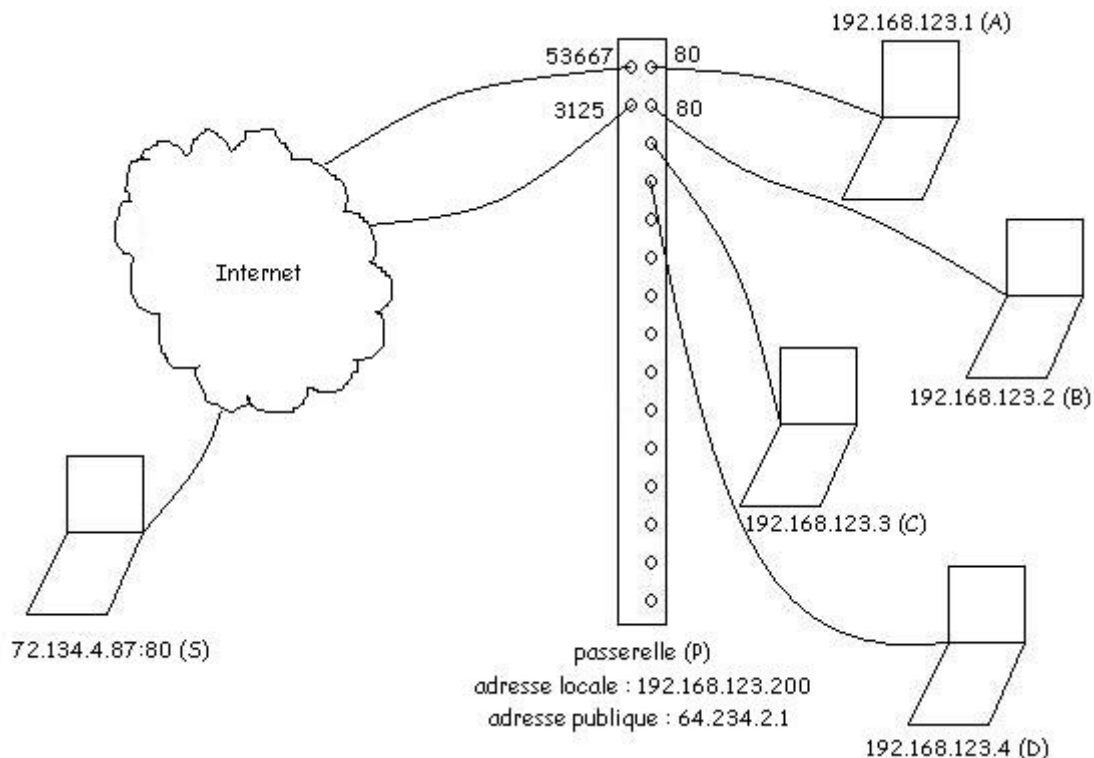
Il existe des plages d'adresses IP réservées pour l'adressage interne :

- 10.0.0.0 à 10.255.255.255
- 172.16.0.0 à 172.31.255.255
- 192.168.0.0 à 192.168.255.255

Ces adresses ne sont pas routables sur Internet.

2) 2 types de NAT

Il existe deux types de NAT. Le NAT statique et le NAT dynamique. On ne va s'intéresser qu'au 2^{ème}. Supposons que nous avons le réseau suivant :



Si la machine 192.168.123.1 A veut se connecter au serveur 72.134.4.87:80 S. A va d'abord initier une connexion avec la passerelle P. Le port source de A serait 4325 et le port de destination 80. A va dire à P qu'elle veut se connecter à S. P va donc initier une connexion avec S via son adresse publique. Ici P va choisir un port quelconque (53667). Quand S va répondre, il répond à P sur le port 53667. P sachant que ce port particulier appartient à sa table NAT, il retransmet l'information à A.

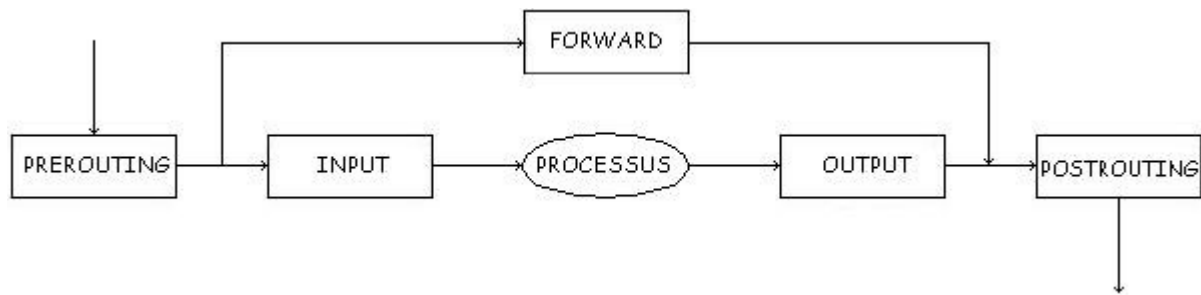
Avantages :

- A partir d'une adresse publique, on peut avoir plusieurs machines qui se connectent à Internet
- Economie effective des adresses publiques
- Sécurité : impossible d'initier une connexion depuis l'extérieur, sauf si on le configure explicitement
- Inconvénient :
 - seul tcp et udp sont pris en compte
 - ralenti la propagation d'IPv6 qui propose d'autres améliorations que le nombre d'adresses disponibles
 - tout les protocoles de la couche application ne peuvent pas passer par un serveur NAT

3) NAT sous Netfilter

Sous Netfilter on a la table 'nat' qui est là pour tout ce qui concerne la translation d'adresse. Elle rajoute deux chaînes à Netfilter :

- PREROUTING : qui traite les paquets avant qu'ils ne soient routés
- POSTROUTING : qui traite les paquets après qu'ils ne soient routés



La commande pour faire de la translation d'adresse est :

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth1 -j MASQUERADE
```

où :

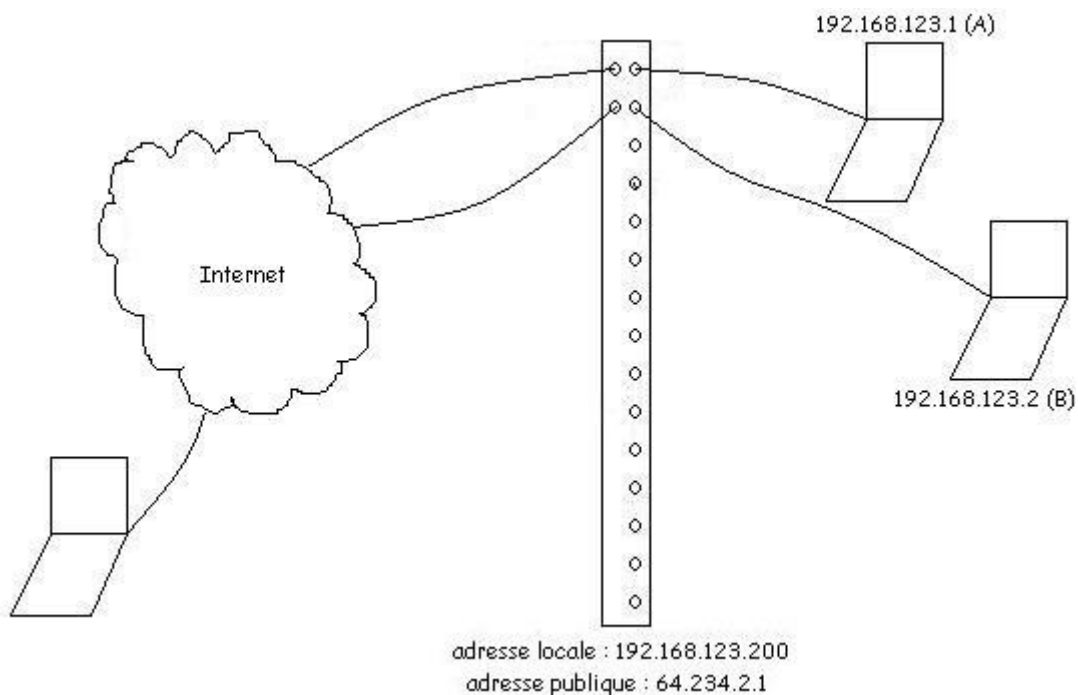
- -s 192.168.0.0/24 indique le sous réseau privé
- -o eth1 indique l'interface de sortie
- -j MASQUERADE est la cible spéciale pour la nat

cela ne suffit pas pour faire fonctionner une machine en tant que passerelle, il faut indiquer dans un fichier spécial qu'on autorise la translation d'adresse. *Echo 1 > /proc/sys/net/ipv4/ip_forward*

Si on souhaite filtrer les connexions réseau qui transitent via la passerelle, on utilise les règles contenues dans la table FORWARD.

4) Redirection de port

On peut grâce à la chaîne PREROUTING rediriger des ports "publique" vers des machines du réseau privé.



Supposons que B propose un service web et qu'on souhaite qu'il soit accessible de l'extérieur (ex : parc). Il faut configurer notre passerelle pour que les paquets à destination du port 80 soient redirigés vers B. Pour cela on utilise la commande :

```
iptables -t nat -A PREROUTING -p tcp -dport 80 -j DNAT --to-destination 192.168.123.2
```

Notons qu'on pourrait tout à fait rediriger sur d'autres ports : le port publique serait 43732 et sur le serveur web, 80.

5) Comment on fait pour notre pare-feu puisqu'on a pas de fichier de configuration?

La première solution est de faire un script qui est appelé au lancement de la machine.

La seconde solution est d'utiliser une soucouche logiciel qui s'occupe de la configuration de Netfilter.

On peut citer Shorewall qui possède des fichiers de configuration et fonctionne comme un service.

Il existe aussi des interfaces graphique comme Guarddog (non encore compatible avec notre version de la Debian), ou encore Firestarter qui propose encore moins d'options de configuration que le firewall de Windows XP SP2.

V - IPv6

1) Présentation

IPv6 est la version 6 du protocole IP. Les adresses sont codées sur 128 bits, soit 340 282 366 920 938 463 374 607 431 768 211 456 soit environ $3,4 \times 10^{38}$ adresses disponibles ou encore 667 132 000 milliards ($6,67 \times 10^{17}$) adresses par millimètre carré de surface terrestre. Mais IPv6 permet aussi :

- La réduction des tables de routage
- Une amélioration de la sécurité
- Une amélioration de la qualité de service
- Permet la mobilité

2) Cohabitation avec IPv4

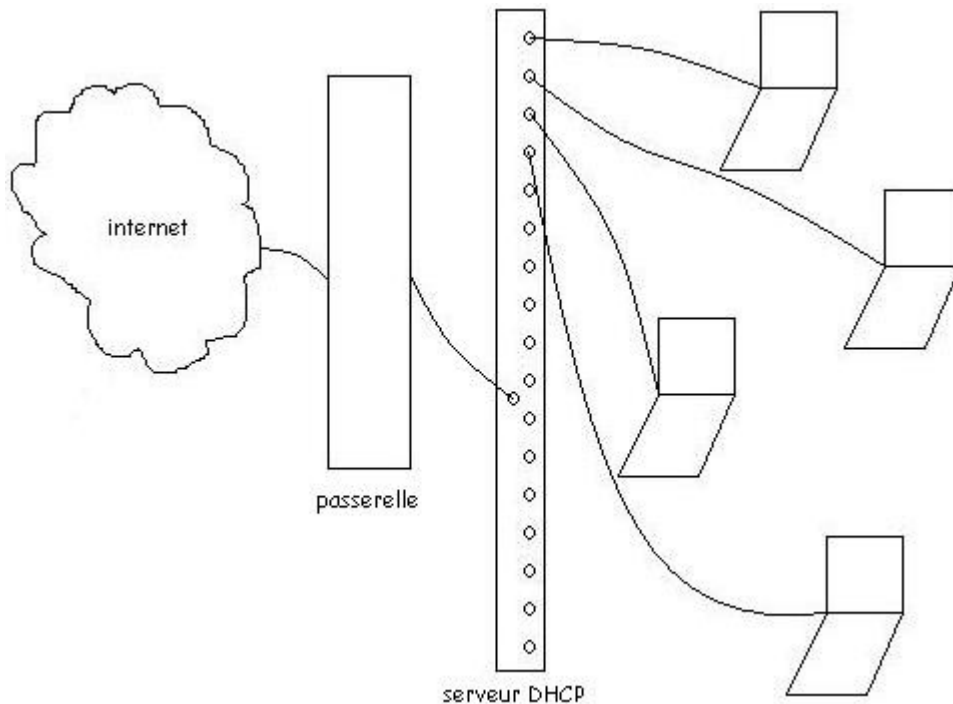
Il est impossible de basculer intégralement tout Internet en Ipv6 d'un seul coup. L'idée est de faire grandir des îlots IPv6 dans un océan d'IPv4. IPv6 est capable d'encapsuler des adresses IPv4, une machine sous IPv6 peut donc accéder à un service IPv4, la passerelle s'occupera d'extraire l'information IPv4. Les serveurs IPv6 possède aussi une pile réseau IPv4, pour que les machines IPv4 puissent y accéder. Enfin les paquets IPv6 peuvent être encapsulés dans des paquets IPv4 pour franchir des parties IPv4 du réseau.

Chapitre 7 : Le protocole DHCP

I - Qu'est-ce que le DHCP

DHCP (Dynamic Host Configuration Protocol) permet d'attribuer automatiquement des adresses IP à des machines d'un réseau.

Exemple :



II - Fonctionnement

Quand une machine se connecte au réseau, elle va faire une requête DHCP. Cette machine n'a pas d'adresse IP et elle ne connaît rien du réseau y compris l'adresse du serveur DHCP. La machine envoie une requête sur le réseau (en UDP et en mode Broadcast) avec comme adresse IP : 0.0.0.0. La requête contient entre autre l'adresse MAC de la machine. Le serveur DHCP va intercepter la requête. Il va regarder si son pool d'adresses contient encore des adresses disponibles et il va renvoyer une réponse DHCP avec une adresse IP disponible plus l'adresse MAC de la machine.

La machine cliente intercepte la réponse, voit que son adresse MAC en fait partie et configure automatiquement son adresse IP avec celle proposée par le serveur. La réponse contient aussi le masque de sous réseau, l'adresse de la passerelle, l'adresse du serveur DNS, le bail (la durée pendant laquelle l'adresse IP est attribuée à la machine).

III - SOUS GNU/Linux

On a à notre disposition le service 'dhcp'.