# TABLE DES MATIERES



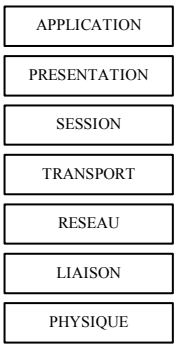
**. : Bibliographie : .**

Hors-Série Linux Magazine MISC n° 8 et 12 : http://www.miscmag.com/
Sécurité Optimale Edition CampusPress
Administration Linux Edition Eyrolles
Tutorial Iptables (Oskar Andreasson) : http://www.netfilter.org/documentation/

## Rappels

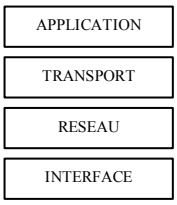
### Modèle OSI

Le modèle OSI est un modèle de référence pour la compréhension de la pile de protocoles et les services mis en œuvre dans le réseau :



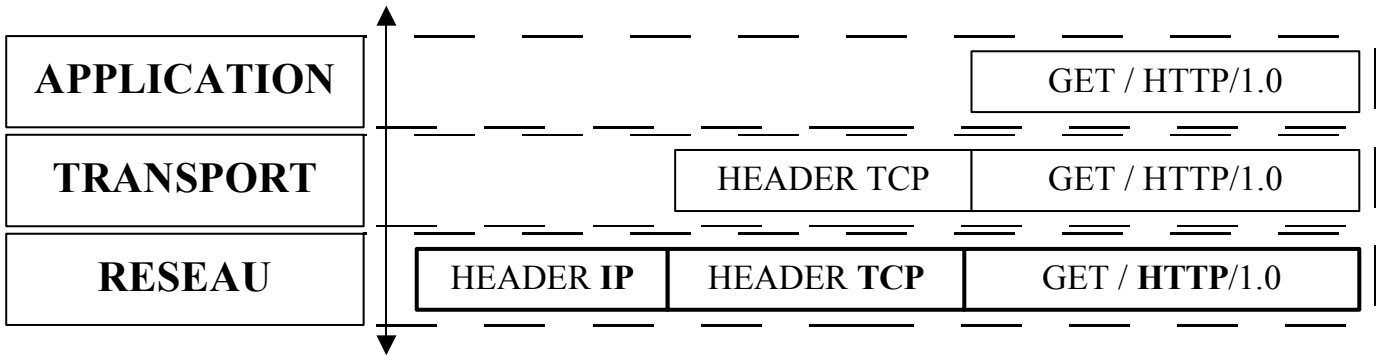

### Modèle TCP/IP

Le modèle DoD plus spécifiquement utilisé dans l'architecture TCP/IP est le suivant :




### Encapsulation

A cela s'ajoute, le principe d'encapsulation des protocoles :



# Introduction

Un système pare-feu (*firewall*) est un dispositif conçu pour examiner et éventuellement bloquer les échanges de données entre réseaux.

C'est donc un élément de sécurité d'un réseau qui peut être :

- un ordinateur
- un routeur
- un matériel propriétaire

Dans tous les cas, un système pare-feu est une combinaison d'éléments matériels et logiciels (propriétaires, *shareware* ou *freeware*).

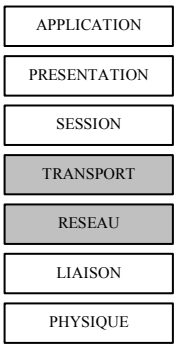
# Principe

Le pare-feu joue le rôle de filtre et peut donc intervenir à plusieurs niveaux du modèle OSI.

Il existe trois types de principaux de pare-feu :

- filtrage de paquets
- filtrage de paquets avec état (*firewall stateful*)
- *proxy*

## Filtrage de paquets




Les pare-feu de filtrage de paquets sont généralement des routeurs qui permettent d'accorder ou de refuser l'accès en fonctions des éléments suivants :

- l'adresse source
- l'adresse destination
- le protocole
- le numéro de port

### Limites de la technique

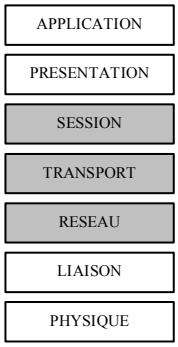
- **Manque de souplesse**

Si cette approche offre une défense simple et efficace, elle manque de souplesse pour que sa mise en place en protection d'un réseau, dans la majorité des cas, ne s'avère pas bloquante pour le bon fonctionnement des systèmes du réseau concerné. Pour contourner ce défaut, l'administrateur est rapidement contraint à autoriser trop d'accès pour que son *firewall* offre encore la moindre protection réelle.

- **Difficulté pour gérer certains protocoles**

Certains protocoles sont particulièrement délicats à gérer à l'aide de cette technique comme FTP (le suivi des échanges FTP est une opération complexe) ou TCP (protocole de type connecté).

## Firewall Stateful




La technologie *stateful* est l'une des deux réponses possibles aux limites du filtre de paquets.

Un *firewall stateful* inclut toutes les fonctionnalités d'un filtrage de paquet, auxquelles il ajoute la capacité de conserver la trace des sessions et des connexions dans des tables d'état interne. Tout échange de données est soumis à son approbation et adapte son comportement en fonction des états.

Cette technique convient aux protocoles de type connecté (TCP). Certains protocoles (UDP et ICMP) posent un problème supplémentaire : aucune notion de connexion n'y est associée. Le *firewall* est donc amené à examiner les paquets, et peut seulement gérer des *timeout*, souvent de l'ordre d'une minute.

### Limites de la technique

- **Risque d'accès illimité**

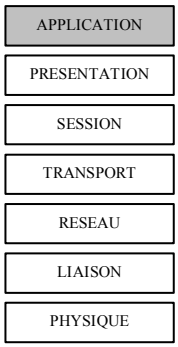
Si cette approche apporte une amélioration certaine par rapport à la technique du filtrage simple de paquets, elle se borne cependant à autoriser ou interdire l'accès à un service donné. Dès lors que l'accès à un service est accordé, celui-ci est illimité.

- **Faille interne au *firewall***

D'autre part, un tel système ne protège aucunement un serveur contre les attaques s'appuyant sur des failles du logiciel utilisé pour fournir le service.

Néanmoins, les pare-feu de type *Stateful* sont considérés, par les administrateurs réseau, comme le technologie minimum pour une sécurisation.

## Proxy




Les *firewalls* de type *proxy* applicatif (ou passerelle applicative) ont pour ambition de répondre aux problèmes soulevés par les filtres de paquets et les *firewall stateful*.

Ces systèmes se substituent au serveur ou au client qu'ils ont pour mission de défendre pour :

- traiter les requêtes et réponses à la place du système à protéger,
- les transmettre, après d'éventuelles modifications
- ou les bloquer

Les pare-feu de ce type jouent le rôle de canal et d'interpréteur en agissant aux niveaux des protocoles de la couche Application.

Cette approche permet, en principe, d'atteindre le plus haut niveau de sécurité.

De nombreux *firewalls* de ce type sont disponible sur le marché, comme Raptor de Axent, Gauntlet de NAI, racheté par Secure Computing, Sidewinder de Secure Computing, et M-Wall de Matranet, pour ne citer que les plus connus.

**Limites de la technique**

Cette technologie, bien que prometteuse, est cependant confrontée à certaines difficultés.

- **Adapatabilité**

En premier lieu, tout protocole transitant par le *firewall* doit être connu de celui- ci, pour pouvoir bénéficier de ses capacités de *proxy*, ce qui exclut l'usage d'applications et protocoles "maison", ou plus simplement peu répandus.

- **Difficulté**

En outre, la gamme de protocoles à examiner étant particulièrement vaste, il est extrêmement délicat d'obtenir un système éliminant réellement toutes les attaques envisageables, les développeurs de *firewalls* applicatifs ne pouvant que difficilement maîtriser tous ces protocoles.

- **Failles du *proxy***

Un tel système a pour rôle, comme les clients et serveurs qu'il défend, d'interpréter, comprendre, et réécrire les requêtes et réponses qu'il reçoit. Or, c'est précisément ce type de fonctionnalités qui est à l'origine d'une vaste majorité des failles applicatives.

- **Performance**

Enfin, le gain de sécurité obtenu à l'aide du *proxy* applicatif se paie en termes de performances : les tâches que ce système a pour rôle de réaliser étant nettement plus complexes que celles du *firewall stateful*, une machine puissante est nécessaire pour faire tourner ce type de logiciel, et il est rare que des débits réseau supérieurs au cinquième des débits gérés par un *firewall stateful* puissent être traités.

## Autres possibilités des pare-feu

Les fabricants de pare-feu ont tendance à intégrer un maximum de fonctionnalités :

- **filtrage de contenu** (URL, *spam* mails, code ActiveX, applets Java, …)
- **réseau virtuel privé** (**VPN**) : les VPN permettent de canaliser un trafic sécurisé d'un point à un autre sur des réseaux généralement hostile (Internet par exemple). Checkpoint et Cisco intègrent des services VPN à leur offres de pare-feu.
- **la traduction d'adresse réseau NAT** (*Network Address Translation*) : ce service permet de mettre en correspondance des adresses réservées ou illégales avec des adresses valides. Les premiers dispositifs NAT qui apparaissent en entreprise sont souvent des produits pare-feu.
- **l'équilibrage de charge** : va permettre de segmenter un trafic de façon répartie (orienter le trafic Web et FTP par exemple)
- **la tolérance de pannes** : certains pare-feu, comme CISCO/PIX ou Nokia/Checkpoint supportent ces fonctionnalités (généralement exécution des pare-feu par paire pour permettre un une disponibilité élevée (HA *High-Availability*)
- **la détection des intrusions** (**IDS**) : service nouveau et à la mode !

# Politique de sécurité

## Politique permissive (*open config*)

Cette politique repose sur le principe que par défaut on laisse tout passer puis on va restreindre pas à pas les accès et les services mais la sécurité risque d'avoir des failles.

## Politique stricte (*close config*)

Cette politique repose sur le principe inverse : on commence par tout interdire, puis on décide de laisser seulement passer les services ou adresses désirés ou indispensables.
La sécurité sera meilleure mais le travail sera plus difficile et cela peut même bloquer plus longtemps que prévu les utilisateurs.
C'est évidemment la politique conseillée pour un pare-feu.

# Architecture réseau

Il existe plusieurs **zones de sécurité** commune aux réseaux. Ces zones déterminent un niveau de sécurité en fonction des accès réseaux et donnent les bases de l'architecture.

On considère en général trois zones ou réseaux :

## Réseaux externes

C'est le réseau généralement le plus ouvert. L'entreprise n'a pas ou très peu de contrôle sur les informations, les systèmes et les équipements qui se trouvent dans ce domaine.

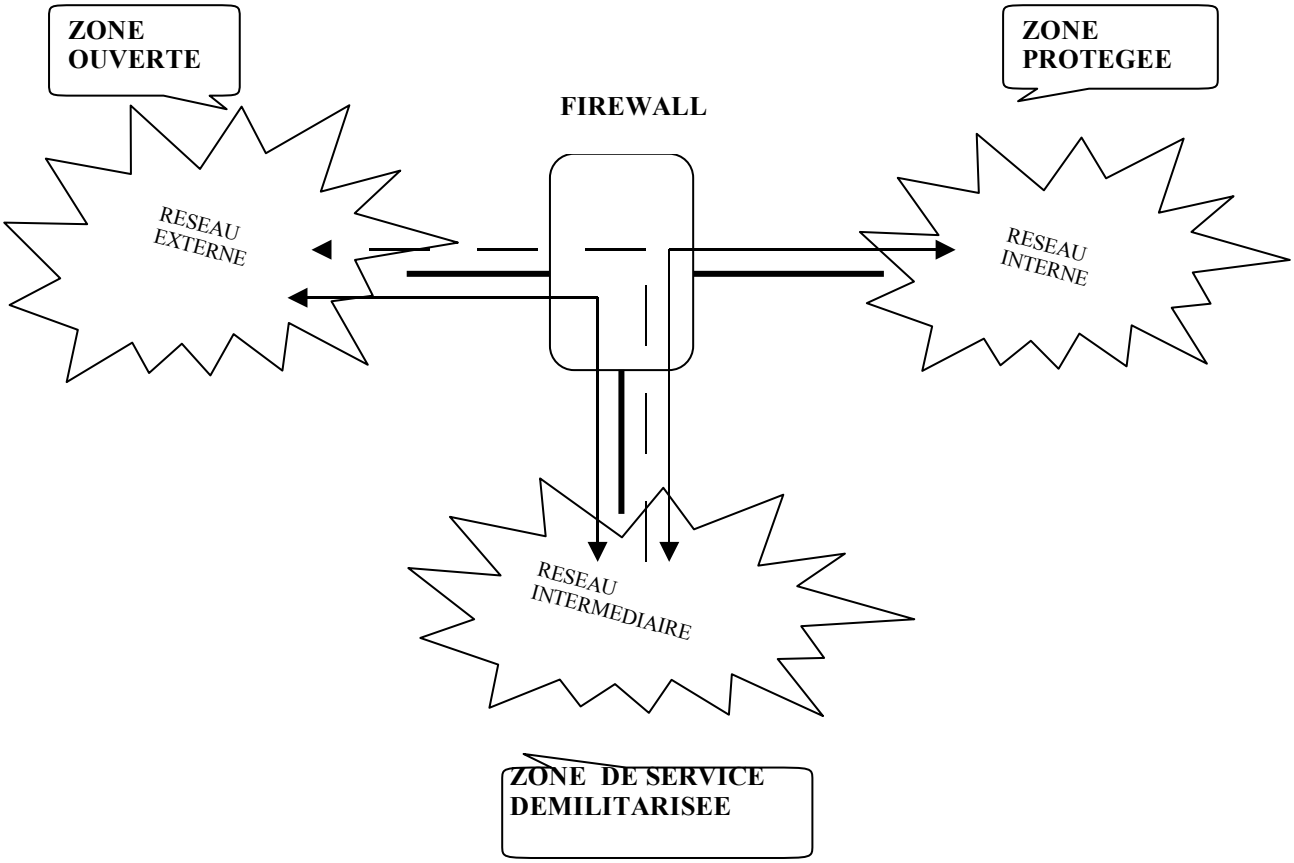
## Réseaux internes

Les éléments de ce réseau doivent être sérieusement protégés. C'est souvent dans cette zone que l'on trouve les mesures de sécurité les plus restrictives et c'est donc le réseau le moins ouvert.

## Réseaux intermédiaires

Cette zone est un compromis entre les deux précédentes. Ce réseau est composé de services fournis aux réseaux internes et externes. Les services publiquement accessibles (serveurs de messagerie, Web, FTP et DNS le plus souvent) sont destinés aux utilisateurs internes et aux utilisateurs par Internet. Cette zone, appelée **réseau de service** ou de **zone démilitarisée** (**DMZ** *De-Militarized Zone*), est considérée comme la zone moins protégée de tout le réseau.

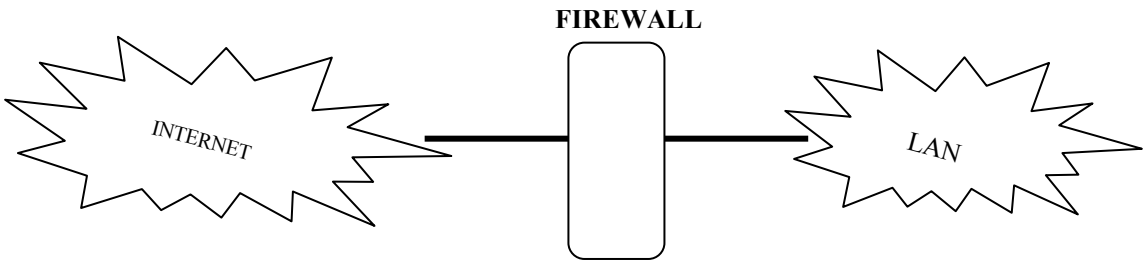
## Exemple d'architecture réseau à trois zones



## La machine rempart (*Bastion Host*)

Le rôle principal qu'une machine tient sur un réseau sécurisé est celui de la machine Bastion ou *Bastion Host* en anglais.

Il s'agit d'une machine directement exposée aux attaques. Ainsi, un serveur ayant une adresse IP publique, et par conséquent accessible depuis Internet, est assimilé à une machine Bastion.




*Remarques :*
Nous considérons ici que la menace vient uniquement d'Internet et non pas de l'intérieur du réseau (ce qui est bien sûr restrictif). Ce type de serveur est donc critique pour la confidentialité, l'intégrité et la disponibilité du réseau. Une attention particulière doit être portée sur sa sécurité. En effet, si cette machine venait à être compromise, la sécurité du réseau (totalement ou en partie) serait menacée.

L'exemple le plus connu de machine Bastion nous est fourni par le pare-feu voire le routeur d'accès au réseau.

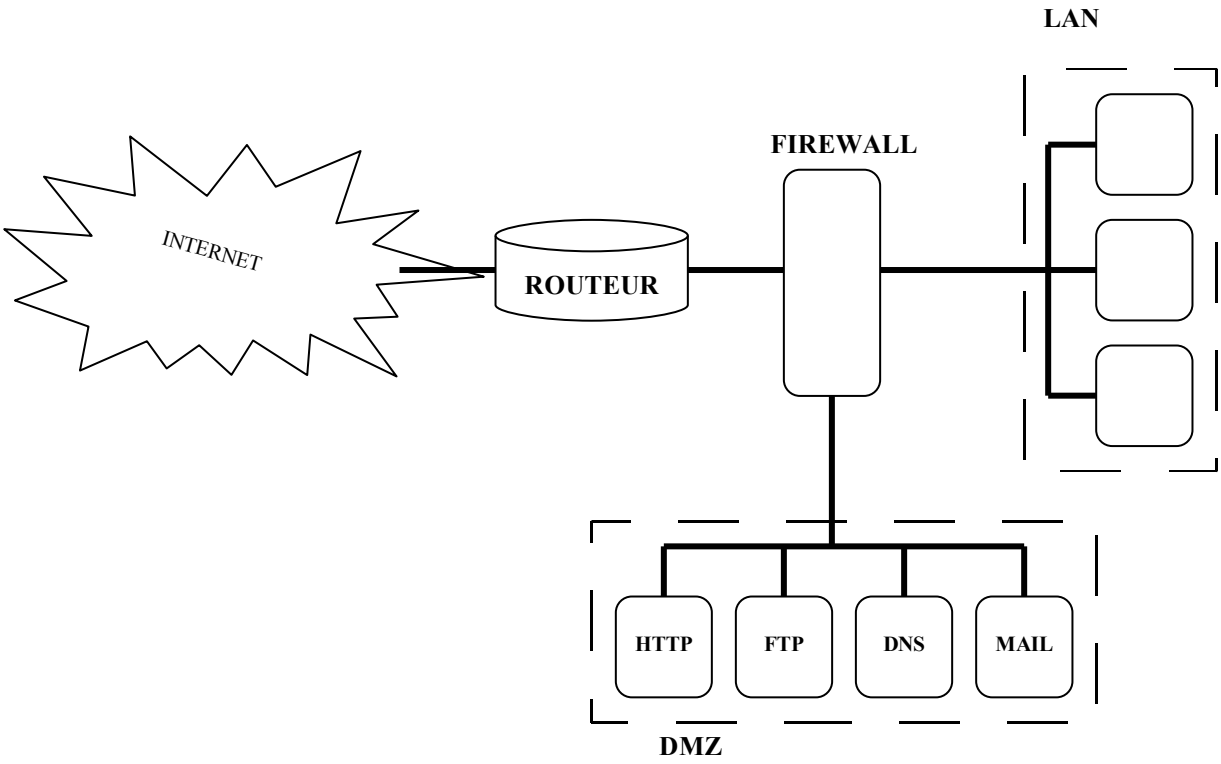
Cependant, les serveurs publiques Web et FTP, les serveurs DNS ou de Mail sont vus aussi comme des Bastions.

*Besoin :*
La nécessité de mettre en place une architecture réseau sécurisée apparaît alors évidente : l'isolement des machines Bastion du réseau interne devient indispensable.

## La zone démilitarisée (*DeMilitarized Zone*)

Avant de connecter un serveur sur Internet, se pose la question de la sécurité de cette machine et du réseau auquel elle appartient. Comme nous l'avons vu précédemment, ces serveurs sont des machines Bastion et doivent être isolés du réseau. C'est à cette problématique que répond la zone démilitarisée (ou **DMZ**).

La DMZ fait partie des principes fondamentaux de la sécurité réseau. Cette zone va jouer le rôle d'espace intermédiaire entre le réseau interne, dit de confiance, et un réseau non maîtrisé, donc potentiellement dangereux. La DMZ isole les machines publiques (Web, DNS, FTP, Mail, ...) du réseau interne. Cette séparation est effectuée et contrôlée par un pare-feu :




La mise en place d'une DMZ est la première étape de la sécurisation du réseau et des machines. Le simple fait d'avoir une DMZ ne fait pas la sécurité du réseau. La DMZ est nécessaire mais non suffisante.

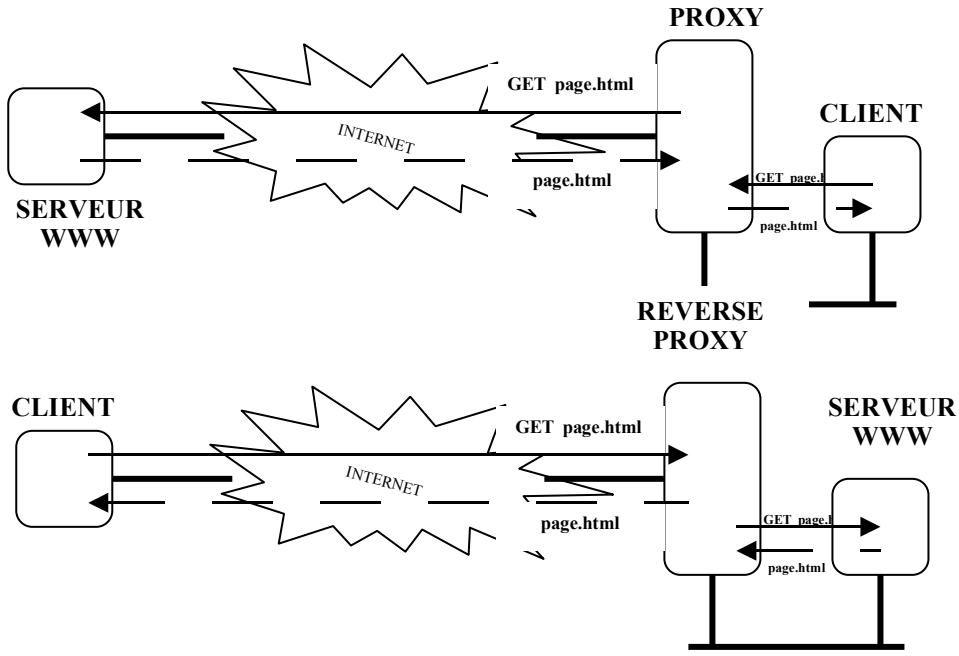
*Besoin :*
L'ajout d'un autre type de barrière renforce également la sécurité : il s'agit d'un contrôle au niveau des protocoles de la couche application (FTP, HTTP, SMTP,...)

## Le proxy et le reverse proxy

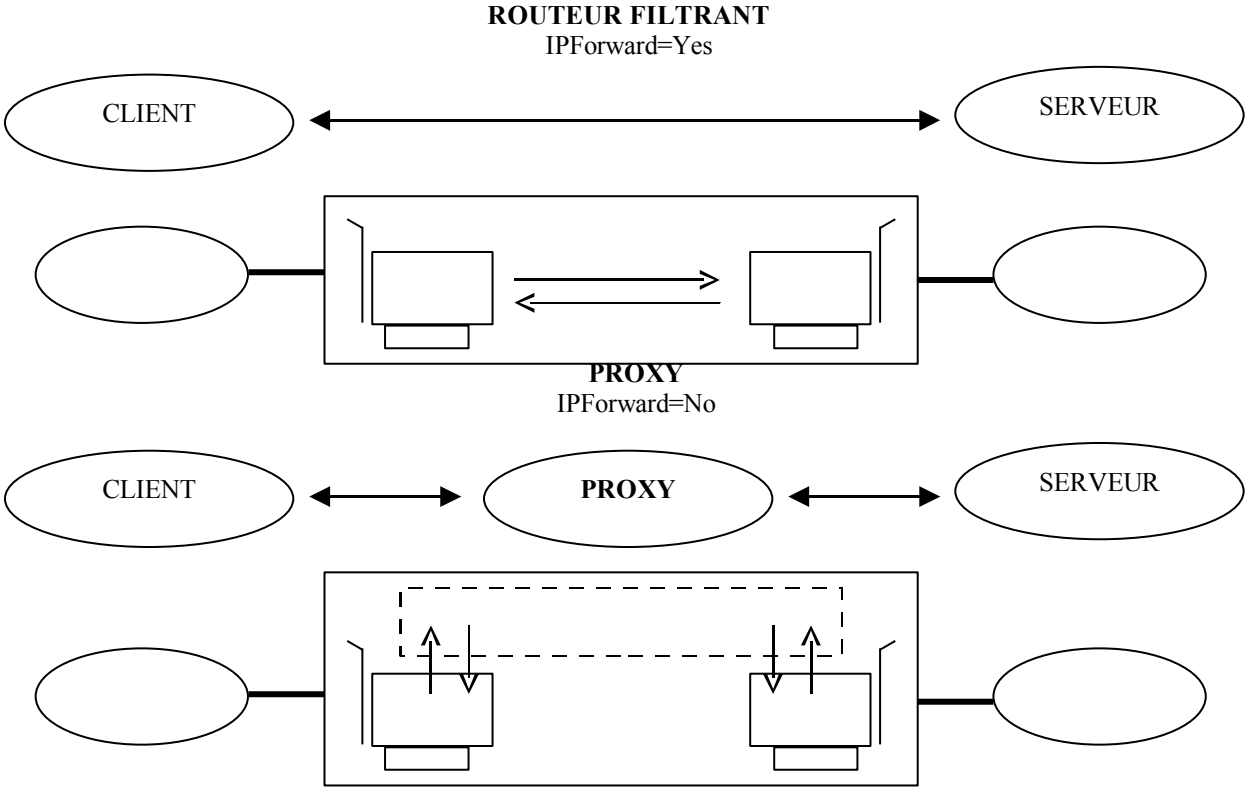
Un *proxy* procure :

- de nouveaux mécanismes de sécurité (filtrage d'URL, authentification des utilisateurs, ... )
- améliorer les performances en offrant une fonction de cache de pages Web.

Techniquement parlant, le *proxy* est un relais entre le client et le serveur de destination. Dans l'exemple du Web, le navigateur va se connecter au *proxy* et le *proxy* se connecte ensuite au serveur. ainsi les utilisateurs du réseau interne ne sont pas reliés directement à Internet, seul le *proxy* y est connecté. Le *proxy* sera donc placé dans la DMZ.




*Comparaison entre **proxy** et **routeur filtrant** :*



**Fonctions de masquage et de translation d'adresse**

Les deux besoins distincts sont :

- partager une ou plusieurs adresses valides d'un réseau d'interconnexion
- masquer les adresses réelles des postes derrière un routeur ou un pare-feu

Plusieurs techniques sont disponibles pour répondre à ces besoins :

- le masquage d'adresse réseau (*Masquerading*)
- la translation d'adresse réseau (NAT)
- les services d'un proxy

**Mise en situation**

Le besoin se pose essentiellement lorsqu'on interconnecte un réseau public (Internet) avec un réseau privé. La situation courante veut qu'on utilise des adresse privées pour le réseau local (il est rare que l'on puisse obtenir des adresses publiques pour l'ensemble d'un réseau).

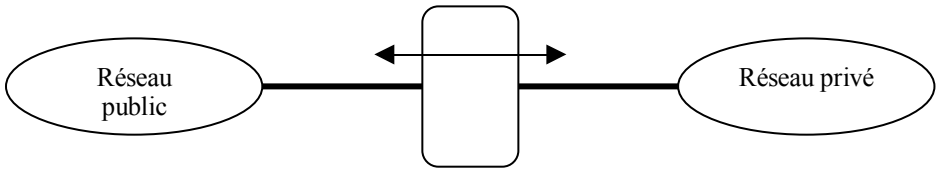
Le réseau privé ne pourra pas utiliser n'importe lequel adressage interne essentiellement à cause des tables de routage et des risques de conflits avec l'extérieur.

Il existe des adresses réservées, qui ne seront donc pas routées, à utiliser pour les réseaux privés :

| | | |
|---|---|---|
| Classe A : 10.0.0.0 | à | 10.255.255.255 |
| Classe B : 172.16.0.0 | à | 172.31.255.255 |
| Classe C : 192.168.0.0 | à | 192.168.255.255 |

*Remarque :*
Ces adresse réservées peuvent être découpées en sous-réseaux privés.

Par contre, les postes privés ont peut être le besoin d'accéder aux services proposés par le réseau public interconnecté :



### IP Masquerade

L'IP masquerade, consiste à masquer les adresses IP du réseau interne et à n'utiliser que l'adresse du routeur sur le réseau d'interconnexion.

Cette technique est utilisée pour connecter tout un réseau local sur l'accès d'un prestataire qui ne fournit qu'une seule adresse IP (souvent dynamique) : partage d'une connexion internet le plus souvent.

La différence majeure entre le MASQ et NAT est que le serveur MASQ n'a besoin que d'une adresse IP valide (celle de l'interface de sortie).

La différence majeure entre le MASQ et un PROXY est que le serveur MASQ n'a pas besoin de changer la configuration des machines clientes. Il faut juste que les machines clientes déclarent leur passerelle par défaut.

### NAT (*Network Adress Translation*)

La machine, ayant a sa disposition un certain nombre d'adresses IP valides, pourra utiliser NAT pour les partager avec un groupe de postes ne possédant que des adresses IP privés, ou réservés ou illégales (au sens RFC).

Lorsqu'une connexion vers l'extérieur est requise par un utilisateur interne, cette machine associe une adresse IP valide disponible a l'IP privée qui requiert la connexion. Lorsqu'une adresse NAT publique reste sans être utilisée pendant un certain laps de temps, l'adresse IP publique retourne dans le groupe des IP publiques disponibles.

Le problème principal du NAT est qu'une fois que toutes les adresses IP publiques sont utilisées, ceux qui veulent obtenir une connexion ensuite ne le pourront pas tant qu'une adresse ne se libère pas.

### Proxy

Un serveur proxy utilise seulement une adresse IP, comme IP MASQ, et sert de passerelle pour les clients du réseau privé.

Toutes les applications clientes doivent supporter les services du proxy (SOCKS par exemple) et être configurées pour les utiliser.

Au delà du masquage d'adresse, le proxy a la possibilité d'ajouter une fonction de cache pour améliorer les performances globales.

### Proxy-ARP

Le proxy-ARP permet de résoudre le problème posé par des pare-feu installés sur un même réseau d'adresse.

Lorsqu'une machine doit communiquer avec une autre sur le même réseau, on a besoin de déterminer l'adresse Ethernet MAC correspondant à son adresse IP. La machine source envoie donc en diffusion générale la question "Quelle est l'adresse MAC de l'interface qui réponds à l'adresse IP 1.2.3.4 ?" par l'intermédiaire du protocole ARP (*Adress Resolution Protocol*). Les requêtes ARP, étant faites par *broadcast*, seront donc normalement bloquée par la machine filtrante. L'activation de la fonctionnalité Proxy-ARP permet de palier à ce problème en demandant explicitement à ce que les requêtes et réponses ARP arrivant par une carte soient propagées vers l'autre et vice-versa.

```
# On active Proxy-ARP pour chacune des 2 interfaces
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
```

## Solutions Linux

### Routage (statique)

Un PC sous GNU/Linux est capable de router des paquets IP sans ajout de logiciels spécifiques. En effet, tout PC/Linux, fonctionnant sous TCP/IP, possède une table de routage accessible par la commande **route**.

Un simple poste de travail devra au minimum indiquer l'adresse de la passerelle (*gateway* au sens IP) du réseau (routeur, *firewall* ou *proxy*).

Maintenant, il est possible, en installant au moins deux interfaces réseaux, de transformer le PC/Linux en **routeur**.

Pour cela, on réalisera le mode opératoire suivant :

_ Installation et configuration des interfaces

Dans le cas de deux cartes réseaux Ethernet, elles seront identifiées par eth0 et eth1 dans le système.

Pour les configurer, on utilisera la commande ifconfig ou les scripts présents dans /etc/sysconfig/network-scripts/ifcg-ethx.

_ Activation des interfaces

Là encore, on utilisera la commande suivante : ifconfig ethx up ou ifup ethx
On peut évidemment aussi les désactiver : ifconfig ethx down ou ifdown ethx

### Configuration du fichier *hosts*

Ajouter un nom pour chaque interface dans le fichier /etc/hosts.

### Valider le *forward* IP

Mettre l'option FORWARD_IPV4 à yes dans le fichier /etc/sysconfig/network.

OU

Exécuter la commande suivante : echo 1 > /proc/sys/net/ipv4/ip_forward

*Remarque :*
Dans ce fichier, on peut indiquer entre autre la route et l'interface par défaut (options GATEWAY et GATEWAYDEV).

### Configuration des noms logiques

Cette étape n'est pas indispensable mais facilitera le travail de configuration des routes.
Pour configurer les noms symboliques des réseaux, on utilisera le fichier /etc/networks.

### Configuration des routes

Pour cela, on utilisera la commande **route** pour configurer la table de routage de la machine.
Visualiser les routes : route
Supprimer une route : route del …
Ajouter une route : route add …

Les routes sont normalement décrites dans le fichier /etc/sysconfig/static-routes.

Le routage statique, mis en œuvre ici, consiste à imposer aux paquets la route à suivre. Seul l'administrateur pourra manuellement modifier la table de routage.

Le routage statique convient pour des sites de taille modeste puisqu'il ne peut pas :

- gérer des changements de topologie complexes
- gérer des problèmes de type engorgement ou panne d'interface

### Tests

Le commande la plus utilisée pour faire un test est la commande ping.

Il faut tout de même remarquer que la plate-forme GNU/Linux propose de nombreux outils supplémentaires. On peut citer les commandes : arp, netstat, traceroute, …

## Routage (dynamique)

Le routage dynamique met en œuvre des algorithmes qui permettent aux routeurs d'ajuster les tables de routage en échangeant des messages avec les autres routeurs. La table de routage sera donc mis à jour en fonction de l'état du réseau (nouveaux routeurs, engorgements, pannes, …).

Ces protocoles dits de routage ont pour but de maintenir la cohérence de la table de routage et non de router. On peut citer les protocoles : EGP (*Exterior Gateway Protocol*), RIP (*Routing Information Protocol*) et OSPF (*Open Shortest Path First*).

Le routage dynamique (et ses protocoles) impose donc l'ajout d'un logiciel supplémentaire : un **routeur logiciel**.

Sous GNU/Linux, on pourra utiliser **GNU Zebra** (http://www.zebra.org/).

### Routeur filtrant *stateful*

Pour les versions 2.4 du noyau Linux, on utilisera **Netfilter/iptables**.

### Proxy

Dans le cas du Web, on peut utiliser **Apache** configuré en *proxy* ou le logiciel **Squid**, un *proxy* avec cache.