La IP-WHATIS en français

Jérôme Petazzoni, skaya@mail.dotcom.fr

8 avril 1999

Table des matières

1 Préambule

Ce document n'est pas une "Howto", qui indique comment faire telle ou telle chose (en général sous Linux), mais une "Whatis", c'est-à-dire un petit background culturel pour ceux et celles qui veulent comprendre quelquechose aux réseaux TCP/IP. La plupart de ces informations me viennent du NAG (disponible sur http: //the-floyd.dyndns.org/ skaya/nag/ ou partout ailleurs, et entre autres sous forme imprimée chez O'Reilly). Il existe une excellente traduction franaise de ce livre, qui s'appelle Administration Réseau Sous Linux ; sa portée ne se limitant pas au monde Linux, je ne peux que recommander vivement sa lecture.

Faites moi plaisir : si vous relevez des fautes (de grammaire, d'orthographe, mais aussi de fond), signalez-le moi afin que j'apporte les corrections nécessaires. N'hésitez pas non plus à m'envoyer un mail pour me dire ce que vous pensez de ce document ; si je reois suffisamment de commentaires positifs, a m'incitera à pondre d'autres doc sur ce sujet ou d'autres.

2 Quelques bases techniques

Cette section n'est pas vraiment nécessaire pour utiliser un réseau IP. Mais pour en administrer un, il est indispensable de maîtriser quelques bases ; ne serait-ce que pour dissiper la confusion fréquente entre IP et TCP...

2.1 Le protocole IP

IP veut dire "Internet Protocol". Et Internet veut dire "INTERconnexion of NETworks", ce qui veut dire que IP est particulièrement adapté pour faire communiquer entre eux plusieurs réseaux distincts - par exemple, le Palo Alto Research Center (le célèbre PARC) et le petit LAN de 486 de la maison. Du point de vue de IP, rien ne distingue un serveur Dec Alpha à 1 gigahertz, derrière sa liaison à 100 megabits/seconde, d'un modeste 386 caché derrière une liaison PPP à 2400 bits/seconde.

IP est un protocole orienté "paquet" ; on dit aussi parfois "trame". Cela signifie que pour envoyer des données à quelqu'un, on fait comme pour une lettre ou un paquet postal : on prépare ce qu'on veut envoyer, on met le tout dans une enveloppe, on règle quelques détails concernant l'enveloppe (paiement, adresse du destinataire...) et on envoie. Rien ne nous garantit que l'enveloppe va arriver dans les délais, ni même qu'elle va arriver ; et si on en envoie plusieurs, elles peuvent arriver dans le désordre. Sur un réseau, c'est encore pire : les paquets peuvent même arriver clonés en plusieurs exemplaires!

(NB : la philosophie "paquet" s'oppose à la philosophie du "circuit virtuel", qui est une méthode employée, par exemple, pour le téléphone...)

IP en soit ne sert pas à transporter de l'information directement. En fait, le protocole IP sert de support à d'autres protocoles, les plus courants étants ICMP, UDP et TCP – un peu de la même manière qu'un réseau local Ethernet, qui peut servir à transporter des données IPX ou TCP/IP.

Un paquet IP comporte typiquement l'adresse de son expéditeur, l'adresse du destinataire, et tout le reste dépend du "sous-protocole" utilisé (attention, je dis "sous-protocole" mais certains pourraient, avec raison d'ailleurs, dire "sur-protocole" ; c'est un point de vue tout relatif).

2.1.1 Les adresses

En ce qui concerne les adresses, ce sont les fameux "dotted quads", de 32 bits (soit 4 octets) du genre 134.157.0.129 ; ce qui permet théoriquement plus de 4 milliards d'adresses ; en réalité il y en a beaucoup moins. Explications. Les créateurs du protocole IP on décidé qu'il y aurait 3 classes de réseaux. Les réseaux de classe A, qui seraient peu nombreux, mais permettraient un très grand nombre de machines. A savoir : 127 réseaux (1.0.0.0 à 127.0.0.0) de 16 millions de machines chacun (à peu près). Ensuite, 16320 réseaux de classe B (de 128.0.0.0 à 191.155.0.0), pouvant chacun accueillir 65024 hôtes. Enfin, 2 millions (à peu près) de réseaux de classe C (de 192.0.0.0 à 223.255.255.0) comportant chacun 254 adresses. Les adresses restantes (224.0.0.0 et au dessus) sont réservées pour des usages particuliers comme le multicasting, par exemple (mais c'est une autre histoire).

Notons qu'à chaque fois, quelques adresses ont été réservées pour l'usage interne. Il s'agit du réseau de classe A (10.*.*.*), des réseaux de classe B (172.16.0.0 à 172.31.0.0), et des 256 réseaux de classe C (192.168.*.*) ; ces adresses sont destinées aux réseaux IP non reliés à Internet - ou du moins, pas directement. Quand un routeur voit une telle adresse comme source ou comme destination d'un paquet, il en déduit que c'est un hôte d'un réseau privé qui a laissé s'échaper un paquet, et il ne le fait pas passer. Je parlerai de ces adresses sous le nom d'adresses privées ("private addresses", dans le RFC1918), et un réseau composé de telles adresses sera de la même faon appelé "réseau privé", ou à la rigueur "réseau interne". Ne pas confondre avec le terme "Intranet", qui appartient plus au langage marchand que technique, et qui peut désigner un réseau privé aussi bien qu'un réseau derrière un Firewall, ou encore un réseau non IP!

2.1.2 Un mot sur le routage

C'est IP qui est responsable de l'acheminement des paquets ; c'est-à-dire que des paquets contenant des données ICMP, TCP ou UDP seront routés exactement de la même faon (sauf exceptions vicieuses, du genre un firewall qui empêche certains paquets de passer ; mais c'est un autre problème). De faon générale, un hôte ne sait pas en détail comment faire parvenir ses paquets à destination (NB : hôte désigne tout ce qui a une adresse IP : un serveur, une station de travail, un routeur, un hub intelligent, une cafetière... si si, je suis sérieux!). Dans le pire des cas, il suffit à un hôte de savoir comment joindre son routeur, sa "gateway". Ensuite, c'est ce routeur qui prendra des décisions, et qui passera les paquets à telle ou tel autre routeur, et ce jusqu'à ce qu'on arrive au réseau de la machine de destination. Exactement comme pour une lettre à la poste : celui qui poste la lettre ne sait pas forcément par o elle va passer. Elle va être acheminée de centre de triage en centre de triage, jusqu'à arriver au bureau de poste du destinataire o le facteur s'occupe de la dernière étape.

En général, les routeurs qui font un "vrai" travail de routage (par opposition au travail plus simple pour relier 2 réseaux, par exemple) ont des tables plus ou moins importantes qui les aident à savoir par o faire transiter les paquets, et ces routeurs échangent des tas d'informations entre eux afin de mettre ces tables à jour ; ainsi, si quelque part, quelquechose tombe en panne, le traffic sera redirigé par un autre chemin sans que les utilisateurs s'en rendent compte (enfin, eux ils verront peut être que a rame à fond, mais les applications n'ont pas conscience de ce genre de choses - quoi que... - et elles ne planteront pas - quoi que...).

Le routage "tout bête", comment a marche? Prenons l'exemple d'un hôte sur un réseau local, lequel réseau local est relié à Internet par un routeur. On va dire que l'adresse de cet hôte est 192.168.6.201. L'adresse du réseau est donc 192.168.6.0, et on admettra que le routeur a pour adresse 192.168.6.3. Cet hôte aura une table de routage primitive, indiquant : "si tu as des paquets pour 192.168.6.201, ils sont pour toi, prends les. Si tu as des paquets pour 192.168.6.*, ils sont pour quelqu'un sur le réseau local, envoie les sur la carte réseau. Si tu as des paquets pour *.*.*.*, c'est pour le monde extérieur, envoie les au routeur, 192.168.6.3, et il se débrouillera". Noter que l'ordre est important : si on avait mis la dernière entrée en premier, tous les paquets auraient été dirigés vers le routeur, MAIS l'hôte n'aurait jamais réussi à joindre ce routeur, puisqu'il n'est pas dans la table de routage. Ou plutôt : il y est, mais comme le système de routage s'arrête à la première correspondance trouvée, il ne serait jamais arrivé à l'entrée correspondante. (Pour tout faire passer par le routeur, il aurait fallu remplacer 192.168.6.* par 192.168.6.3).

Petite note technique concernant l'ordre d'examen des différentes routes : il arrive aussi qu'on examine d'abord les routes les plus "précises", c'est-à-dire en général celles qui pointent sur des hôtes, avant d'examiner celles, plus générales, désignant des réseaux. La "précision" d'une route est donnée par son masque.

NB : un routeur qui relie plusieurs réseaux entre eux a une adresse sur chaque réseau. Une même machine peut donc avoir plusieurs adresses IP, a ne pose pas de problème. En général, on a une adresse par interface (c'est à dire par carte réseau/modem/etc.), mais on peut aussi avoir plusieurs adresses par interface - cela sert en général à faire du Virtual Hosting, voir plus bas.

Inversement, rien n'empêche d'assigner la même adresse à plusieurs interfaces – mais c'est pour des situations bien particulières. Cela ne permet pas, par exemple, de réaliser de la répartition de charge (*load balancing*) ; il faut des drivers (voire des cartes réseaux) étudiés pour.

2.2 Les protocoles portés par IP

Il y en a un bon paquet... (sans jeu de mots). Pour en avoir une petite liste non exhaustive, regarder le fichier /etc/protocols sous Unix... Je vais parler ici de ceux qui sont le plus souvent utilisés dans mon entourage.

2.2.1 ICMP

Internet Control Message Protocol sert, comme son nom l'indique, à passer des messages de contrôle. Hum. Voilà qui est clair. C'est le protocole utilisé, par exemple, par les fantastiques programmes "ping" et "traceroute" (accessoirement, ce dernier utilise aussi UDP). Le premier envoie des paquets à un hôte, ces paquets lui demandant "réponds moi" ; et cet hôte, bienveillant, lui répond ; le programme affiche lorsqu'il reoit ces réponses, et affiche aussi le temps écoulé entre la question et la réponse. Cela permet de tester une connexion IP quand rien ne marche : si on arrive à "pinger" une machine mais qu'à part a, elle ne répond pas, c'est probablement qu'elle est légèrement plantée ou qu'on lui a dit de ne pas répondre ; si on n'arrive pas à la "pinger", c'est qu'elle est probablement éteinte ; et enfin, si on n'arrive pas à "pinger" le routeur qui dessert cette machine, hum, alors c'est normal que la machine elle même ne réponde pas (à moins bien sr que le routeur ait été informé de ne pas répondre aux pings...).

Autre utilisation d'ICMP : la transmission d'informations de routage. Si un routeur doit transmettre un paquet qui a emprunté un détour, il peut envoyer un message ICMP à l'expéditeur pour lui indiquer une route plus directe. Cela arrive, par exemple, quand le routeur reçoit un paquet venant d'un réseau, destiné à ce même réseau. Ce mécanisme n'est efficace que pour des topologies simples ; ensuite, on utilise les programmes `gated` ou `routed`, par exemple.

On peut aussi utiliser ICMP sur un firewall, lorsqu'un hôte externe essaie de contacter un hôte interne alors qu'on a bloqué l'accès : on envoie alors un message pour signifier "accès refusé, passez votre chemin". Cela se traduit en général par un "no route to host".

2.2.2 UDP

Un peu plus évolué, le User Datagram Protocol sert, lui, à transporter des données. Pour une fois. Il offre un autre avantage sur ICMP : il comprend la notion de "port" ; c'est-à-dire que chaque hôte qui comprend le protocole UDP peut recevoir et émettre en utilisant 65536 ports différents. Ce qui permet de mieux différencier les connexions et d'éviter aux données des différents programmes de communication de se mélanger joyeusement. Ainsi, au lieu d'être obligé d'avoir une machine pour gérer le serveur NFS, une pour le yellow page, une dernière pour le DNS, on leur assigne chacun un port bien défini ; et lorsqu'on souhaite s'adresser au service DNS, on sait qu'il faut s'adresser au port 53.

En effet, certains de ces ports sont à peu près standardisés ; i.e. on y trouve quasiment toujours les mêmes services. Pour une liste, consulter /etc/services sous Unix. Noter aussi que sous Unix, les ports en dessous du port 1024 ne sont utilisables que par l'utilisateur privilégié ("root"), ce qui peut consituer une forme très primitive d'authentification (si une connexion vient d'un port en dessous du port 1024, c'est qu'elle a été établie par un programme privilégié, et inversement, lorsqu'on s'adresse à une machine sur un de ces ports, on sait que le programme qui prend la connexion en charge n'est pas celui de n'importe quel utilisateur, mais bien celui assigné par l'administrateur de la machine).

Les inconvénients d'UDP sont ceux inhérents à IP, ce qui signifie que les paquets peuvent arriver en retard, ou dans le désordre, ou en plusieurs exemplaires, ou ne pas arriver du tout. Il faut donc prendre en compte ce facteur d'incertitude quand on compte l'utiliser... En pratique, sur un réseau local, UDP est fiable quasiment à 100% (sauf en cas de congestion du réseau), mais dès que quelques routeurs viennent se mêler à l'histoire, ne plus trop compter sur ce taux d'erreur.

2.2.3 TCP

Ah, on entre dans les choses intéressantes. Tout comme UDP, le Transmission Control Protocol dispose de 65536 ports. Attention, ces ports sont complètement indépendants des ports UDP! On peut avoir un programme utilisant le port 80 en UDP et un autre l'utilisant en TCP, sans aucun problème. Cela ne devrait pas étonner ceux qui ont compris que les ports ne dépendent pas d'IP, mais de TCP ou UDP.

Le gros point fort de TCP est qu'il est orienté connexion, alors que UDP était orienté datagramme (ou paquet) ; tout comme le téléphone par rapport aux lettres (pour ceux qui sont avec nous depuis le début de cette émission). Ce qui veut dire que les données qu'on envoie arriveront, et en plus elles arriveront dans l'ordre et en un seul exemplaire. Par contre, elles arriveront un petit peu moins vite (du moins, quand il y a des erreurs à corriger) et on perd un peu en débit ; mais c'est peu de chose (à peine 1 % sur un réseau 10 megabits/seconde). En fait, il y a 3 cas o il est intéressant d'utiliser de l'UDP à la place du TCP : quand o il est important que les messages arrivent vite (exemple : les jeux) ; quand les messages à échanger sont tellement brefs qu'établir une connexion TCP prendrait trop de temps (exmple : le DNS) ; et quand on veut un protocole "stateless", c'est à dire que le serveur peut redémarrer sans causer la déconnexion des clients (exemple : le NFS).

2.3 Le support matériel

La connaissance intime du fonctionnement exact du matériel n'est pas indispensable pour bien configurer son réseau ; c'est nécessaire uniquement pour dépanner des conflits d'IRQ ou d'autres aberrations propres au monde PC. Cependant, je vais ici parler un tout petit peu des différents supports physiques ou non qui peuvent faire transiter des paquets IP.

2.3.1 Ethernet (ou réseau local en général)

C'est le plus souvent utilisé dans les entreprises, les universités, les écoles, et les petits réseaux locaux qu'on se bricole à la maison. Le débit peut être modeste (entre 100 et 1000 ko/seconde pour du 10 megabits) ou plus important (plusieurs megaoctets par seconde en 100 megabits) ; avec la baisse récente des prix du matériel 100 megabits/seconde, ce dernier devient envisageable pour les particuliers.

Ces débits sont parfaitement adaptés à du partage de fichiers légers, de la messagerie, etc. ; et avec du bon matériel, on peut envisager des applications plus exigeantes. Un détail intéressant : sur un réseau local, on peut envoyer des paquets en "broadcast", c'est à dire faire en sorte que toutes les machines du réseau local reoivent ce paquet, sans pour autant charger davantage le réseau. Mais pour cela il faut utiliser un protocole orienté paquet, ce qui exclut TCP.

La configuration typique d'un hôte en réseau local est celle évoquée plus haut : une entrée de routage pour le réseau local, o se trouve le point d'accès au monde extérieur, et une entrée pour diriger les paquets vers ce point d'accès. Parfois, on trouve un deuxième routeur permettant de joindre un autre réseau (imaginons une entreprise avec 2 réseaux distincts, dont l'un relié à Internet ; on peut imaginer pas mal de possibilités pour le routage : si le routeur qui relie les 2 réseaux entre eux est celui qui relie le tout à Internet, c'est comme avant ; par contre si il n'est visible que depuis l'un des réseaux, il peut être configuré pour rediriger tous les paquets destinés à l'autre réseau vers l'autre routeur, reliant les 2 réseaux ; ou alors chaque hôte peut avoir dans sa table de routage le routeur pour Internet ET le routeur qui relie les deux réseaux).

2.3.2 PPP

Point-to-Point-Protocol est utilisé traditionnellement pour les liaisons par modem. De faon plus générale, à partir du moment o deux machines peuvent communiquer à travers quelquechose qui ressemble à un périphérique en mode caractère, PPP peut marcher (un périphérique en mode caractère, c'est quelquechose que l'on peut utiliser comme un tuyau ; ce qu'on y écrit ressort de l'autre côté dans le même ordre. Exemple : une ligne série). Ainsi, en utilisant pty-redir ou d'autres outils, on peut faire marcher PPP... via une liaison telnet. Les petits futés vont remarquer que c'est un peu inutile puisque si on peut faire un telnet, c'est qu'on a déjà une liaison qui marche ; et bien lisez la fin de ce document au sujet des VPN...

2.3.3 Encapsulation

L'encapsulation, c'est faire passer un protocole dans un autre. Un petit exemple : un hôte est perdu au fin fond d'un réseau IPX. Beurk! Tout passe par IPX, aucun accès TCP/IP en vue. Sauf... sauf à l'autre bout du réseau, derrière 10 routeurs (tous IPX), il y a une machine qui a un accès IP. Cette machine va établir une liaison IPX avec celle qui est isolée, et ensuite tous les paquets IP seront relayés par cette liaison IPX. videmment, cela demande du logiciel spécial de part et d'autre. Un autre intérêt de l'encapsulation est de permettre le "roaming" ; par exemple pour un portable qui se déplace souvent et qui a une adresse IP changeante, et qui veut avoir toujours la même adresse. Il fait comme plus haut : il a un hôte qui se charge de prendre ses paquets (dirigés vers son adresse fixe) et il les redirige par le biais de son adresse mobile.

3 Administration d'un réseau IP

Maintenant, abordons quelques problèmes plus ou moins cruciaux, qui viendront tôt ou tard se poser dès lors qu'on administre un réseau en IP, qu'il s'agisse d'un WAN de plusieurs milliers de postes ou d'un petit réseau local avec 2 ordinateurs en LAN dont un relié en PPP à internet. On va surtout aborder pas mal de "recettes" qui marchent, ou qui marchent moins bien mais qu'il faut connaître parce qu'elles sont utilisées à gauche ou à droite.

3.1 Le pool d'adresses est limité

Première constatation : nous ne sommes plus à l'aube d'Internet, et on manque d'adresses IP. La preuve? Si on en avait assez, on aurait toujours la même adresse en se connectant par son provider habituel. En fait, les providers font une allocation d'adresses IP dynamique : ils ont un stock de, mettons, 200 adresses pour 1000 abonnés (en général, ils ont au moins un réseau de classe C, c'est à dire 254 adresses, mais il faut réserver des adresses pour les routeurs, les serveurs web, mail, ftp... et puis, le provider n'a peut-être que 200 modems, de toute faon). Chaque modem a une adresse IP qui lui est assignée, et quand on appelle le provider, on tombe sur le premier modem disponible - pas forcément toujours le même...

Il est possible de demander une adresse IP statique, mais en général a cote cher. Afin d'avoir une illusion d'adresse IP fixe, on peut utiliser le service dyndns (http: //www.dyndns.org) qui permet d'associer une adresse du genre toto.dyndns.org à une adresse IP, et de changer cette assignation à chaque fois qu'on change d'adresse.

Autre situation dans laquelle on manque d'adresses : prenons le cas d'une petite Université, ou un gros Lycée, qui dispose d'un réseau de classe C mais qui a plus de 300 machines. Ou encore, un particulier avec un accès Internet - donc une seule adresse - mais plusieurs machines à connecter. Plusieurs solutions sont possibles.

3.1.1 DHCP, bootp, rarp...

Le RARP (Reverse Address Resolution Protocol) n'est pas vraiment étudié au départ pour résoudre notre problème, mais il est un peu l'ancêtre des 2 autres. Commenons par le commencement, à savoir l'ARP : Address Resolution Protocol. Sur un réseau Ethernet, quand une machine doit envoyer un paquet à une autre, elle doit connaître l'adresse Ethernet de cette dernière. Les adresses Ethernet sont sur 6 octets, du genre 00: 80: c0: 45: 7a: 99. Afin d'apprendre cette adresse, l'expéditeur utilise le protocole ARP (en gros, il crie très fort sur le réseau : “Qui est 192.168.9.1? ” et l'intéressé lui répond “Moi, 00: 05: 00: cf: a0: 4b”). On a développé plus tard le RARP, qui permet à une machine sans disque dur (ou même avec) qui ne connaît pas son adresse IP, mais qui connaît son adresse Ethernet, de s'autoconfigurer via le réseau avec l'aide d'un serveur.

L'intérêt de ces techniques est d'installer la même configuration à l'octet près sur un ensemble de machine, puis que ces machines puissent se différencier les unes des autres et obtenir les fichiers de configurations qui leur sont propres via le réseau, auprès d'un serveur central.

Traditionnellement, les stations Sun qui doivent booter en “diskless” font une requête RARP, puis, connaissant leur adresse IP et celle de la machine qui a répondu à leur requête RARP, elles demandent via le protocole tftp un fichier à cette machine. Ce fichier, très petit (quelques dizaines de ko) va ensuite être exécuté, et appeler un serveur bootparam en RPC (différent de bootp!), afin d'obtenir l'adresse du serveur qui héberge ses fichiers. Ensuite, le noyau va être récupéré depuis ce serveur via le protocole NFS, et être exécuté. L'enfance de l'art, quoi : -).

Pour simplifier un peu les choses, le protocole bootp est apparu. Il combine le RARP et le bootparam, puisqu'il permet à une machine d'apprendre son adresse IP, et éventuellement une foule de paramètres, comme : l'adresse du serveur DNS, du routeur local, le masque de sous-réseau, le serveur qui héberge ses fichiers, le noyau à utiliser, etc. Sur certaines stations DEC et sur les Powermac (par exemple), la machine fait une requête bootp, puis télécharge le noyau via tftp, et le noyau est exécuté directement.

Enfin, le DHCP (Dynamic Host Configuration Protocol) est apparu afin de combler quelques lacunes du bootp. Le DHCP permet de transmettre des paramètres du serveur vers le client, mais aussi du client vers le serveur (exemple : sous MS Winzob configuré pour obtenir une adresse IP en DHCP, l'ordinateur envoie au serveur le nom sous lequel il apparaît dans le voisinage réseau). De plus, le DHCP exploite le concept de “bail” (“lease”, en anglais), pour gérer les adresses dynamiques : quand un hôte demande au serveur DHCP de lui fournir une adresse IP, le serveur indique pendant combien de temps cette adresse va être valide. C'est au client de renouveler le bail avant la fin de la période de validité s'il veut conserver son adresse. Les baux peuvent être aussi courts que 15 minutes (pour un banc de test TCP/IP) et aussi longs que plusieurs années (dans une université o le parc ne bouge que tous les ans, aux vacances).

On entend aussi parfois parler de PXE ; le PXE est une méthode de boot reposant entièrement sur le réseau. Ce n'est pas à proprement parler un protocole, puisqu'il utilise le DHCP et le TFTP ; on le trouve sur certaines cartes réseau et sur les PCs “diskless”, sans lecteur de disquette ni lecteur de CD-ROM.

3.1.2 Découpage en “subnets”

Prenons maintenant le cas d'une organisation qui a 1 réseau de classe C attribué, mais qui a physiquement 2 réseaux distincts, avec chacun une petite quantité de machines (disons une centaine maximum). Mais pour des raisons X ou Y, on veut pouvoir contrôler le traffic entre les 2 réseaux ; ou plus simplement, les 2 réseaux sont distincts et reliés par un modem ou n'importe quelle liaison de type point à point. Il est possible de découper une tranche d'adresses en plusieurs sous-tranches, qu'on appelle des subnets... tout simplement.

Pour bien comprendre comment a marche, examinons ce qui se passe dans le cas général, en interne. Le système d'exploitation n'a pas conscience de réseaux de classe A, B, ou C. Ce qu'il connaît, c'est l'adresse de la machine, l'adresse du réseau, le masque du réseau et l'adresse de broadcast. Ces termes n'auront plus de secret dans quelques minutes... L'adresse de la machine : voilà un concept pas très difficile. L'adresse du réseau : il s'agit du préfixe commun à tous les hôtes d'un réseau ; conventionnellement, on met à zéro la partie de l'adresse réservée aux hôtes. Le masque du réseau : c'est l'adresse 255.255.255.255, en mettant à zéro la partie de l'adresse réservée aux hôtes. L'adresse de broadcast : c'est - en général - l'adresse du réseau, en mettant à 1 tous les bits de la partie de l'adresse réservée aux hôtes. Aha, je suis sr que personne n'a compris, aussi voici quelques exemples.

```
<table border="2" cellpadding="2">
  <tr>
    <td align="left">Une machine</td>
    <td align="left">Une autre machine</td>
    <td align="left">R\'eseau</td>
    <td align="left">Masque</td>
    <td align="left">Broadcast</td>
  </tr>
  <tr>
    <td align="left">192.168.6.1</td>
    <td align="left">192.168.6.53</td>
    <td align="left">192.168.6.0</td>
    <td align="left">255.255.255.0</td>
    <td align="left">192.168.6.255</td>
  </tr>
  <tr>
    <td align="left">10.0.51.12</td>
    <td align="left">10.6.0.0</td>
    <td align="left">10.0.0.0</td>
    <td align="left">255.0.0.0</td>
    <td align="left">10.255.255.255</td>
  </tr>
  <tr>
    <td align="left">134.157.96.28</td>
    <td align="left">134.157.65.20</td>
    <td align="left">134.157.64.0</td>
    <td align="left">255.255.192.0</td>
    <td align="left">134.157.127.255</td>
  </tr>
</table>
</center></div>
```

Hum, ceux qui suivent vont halluciner quand ils vont voir la 3ème ligne... Qui est pourtant parfaitement correcte. Si vous comprenez les 2 premières lignes, mais pas la 3ème, regardez ce que donne en binaire cette 3ème ligne :

```
<table border="2" cellpadding="2">
  <tr>
    <td>Une machine</td>
```

```
    <td>134.157.96.28</td>
    <td><u><strong>10000110.10011101.01</strong></u>100000.00011100</td>
  </tr>
  <tr>
    <td>Une autre</td>
    <td>134.157.65.20</td>
    <td><u><strong>10000110.10011101.01</strong></u>000001.00010100</td>
  </tr>
  <tr>
    <td>R\'eseau</td>
    <td>134.157.64.0</td>
    <td><u><strong>10000110.10011101.01</strong></u>000000.00000000</td>
  </tr>
  <tr>
    <td>Masque</td>
    <td>255.255.192.0</td>
    <td><u><strong>11111111.11111111.11</strong></u>000000.00000000</td>
  </tr>
  <tr>
    <td>Broadcast</td>
    <td>134.157.127.255</td>
    <td><strong><u>10000110.10011101.01</u></strong>111111.11111111</td>
  </tr>
</table>
</center></div>
```

C'est un réseau de classe B et demi... Ou plus exactement, un subnet d'un réseau de classe B. On peut ainsi diviser un réseau de 65000 hôtes en 2 réseaux de 32000 et quelques, ou 4 réseaux de 16000 et quelques, ou encore un de 32000 et 2 de 16000, etc. J'espère sincèrement que j'ai été assez clair... Sinon, pas d'inquiétude : quand on gère des réseaux avec des adresses privées, on peut toujours se débrouiller sans subnet, et quand on gère des réseaux avec des "vraies" adresses Internet, eh bien on est censé être assez compétent pour maîtriser a : -). Exercice : en supposant que l'administrateur du site de tout à l'heure s'est vu attribuer les adresses 193.1.2.*, calculer les adresses réseaux, les masques et les broadcast des 2 subnets.

3.1.3 Mise en place d'un réseau privé ; d'une DMZ

Mais certains ne trouveront pas leur bonheur dans les solutions évoquées plus haut. Soit parce qu'ils ne peuvent accepter une solution basée sur des adresses dynamiques (toutes les machines doivent pouvoir fonctionner en même temps), ou parce qu'ils ne peuvent diviser en subnet puisque de toute faon ils n'ont pas assez d'adresses IP. La solution est alors de mettre en place un réseau interne, avec - histoire de faire a bien - une DMZ.

Le réseau privé sera un réseau avec des adresses privées (voir au début de ce document pour un rappel à ce sujet). Pour accéder à Internet, les machines du réseau privé utiliseront un système de mandatement (le fameux "proxy") ou de traduction d'adresses (on voit parfois translation, de l'anglais "translate"). De plus amples détails sont disponibles plus loin, mais pour schématiser, elles demanderont à une machine - qui elle, aura une "vraie" adresse IP - de faire le relais pour elles. Quant à la DMZ, il s'agit d'un réseau qui a donc de "vraies" adresses IP, et qui se trouve généralement entre les routeurs d'accès à Internet et le réseau interne. DMZ signifie DeMilitarized Zone. Elle contient en général les serveurs devant être accessibles à la fois depuis l'intérieur et l'extérieur, par exemple : le serveur de messagerie.

Notons qu'en règle générale, une machine située sur le réseau privé ne peut être serveur - au sens conventionnel du terme ; plus simplement, il est techniquement impossible d'établir une connexion depuis l'extérieur vers une de ces machines. Attention, a ne veut pas dire que les données ne passent que dans un sens ; a veut dire que dans le meilleur des cas, une machine de l'intérieur peut établir une connexion vers l'extérieur, et ensuite des données peuvent transiter par cette connexion dans les deux sens ; mais l'inverse est impossible (tout simplement parce que les machines du réseau interne, vues de l'extérieur, n'existent pas puisqu'elles n'ont pas de "vraie" adresse IP!). Et comme pour 99% de serveurs, c'est le client qui établit la connexion...

Voir plus loin pour des explications plus détaillées sur les firewall.

3.2 Le virtual hosting

Changeons un peu de sujet. Le virtual hosting n'est pas une fonctionalité capitale du protocole IP, mais c'est utile de savoir que a existe. Imaginons que l'on veuille héberger 2 serveurs Web, disponibles sous les noms www1.toto.com et www2.toto.com, mais qu'on ne dispose que d'une seule machine (les temps sont durs). C'est possible : il suffit de s'arranger pour que www1.toto.com corresponde à l'adresse, mettons, 192.168.12.1 et www2.toto.com, 192.168.12.2. Ensuite, on va attribuer à la machine qui va héberger les sites, ces 2 adresses IP. Et le programme serveur se débrouillera pour réagir de faon différente selon qu'il est contacté sur une adresse ou sur l'autre. Cela peut marcher pour d'autres choses que les serveurs Web, bien sr.

Note : pour le cas particulier des serveurs Web, il est possible de réaliser du Virtual Hosting avec une seule adresse IP, car le nom du serveur est spécifier dans la requête. Pour les autres protocoles, le Virtual Hosting nécessite généralement plusieurs adresses IP.

4 Quelques mots sur la sécurité et le concept de firewall

Tôt ou tard, il faudra s'occuper du problème de la sécurité, c'est à dire les petits malins qui vont essayer de pénétrer le réseau depuis l'extérieur. Dans le meilleur des cas, cela se traduira par un "nuke" de quelques bécanes, ce qui se résout par un reboot ; mais un intrus pourrait aussi s'introduire sur un ordinateur, et copier des fichiers "intéressants" et généralement secrets (mots de passe, comptabilité d'une entreprise, plans des catacombes, organigramme du RPR, analyse de la robe de Monica Lewinsky...). Afin de permettre aux utilisateurs normaux de continuer de travailler sans avoir à entrer des mots de passes et se soumettre à des scans de rétine toutes les 5 minutes, mais de couper les ponts avec les indésirables venant de l'extérieur, on met en place des firewall.

Un firewall, a peut être un simple routeur intelligent (cisco, ascend...) que l'on peut contrôler à distance, depuis l'intérieur du réseau ; cela peut être aussi un terrible serveur surpuissant équipé d'une panoplie logicielle monstrueuse. La solution la plus intelligente consiste à prendre une machine de performances modestes (inutile de déployer un Pentium pour gérer un traffic de quelques dizaines de ko/seconde, un 386 suffit), mais robuste et stable (pas de cartes mères overclockées, de toute faon le modem reste à 14400 bits/seconde). Le choix le plus important est celui du système d'exploitation. On éliminera d'entrée Windows ou tout autre système de nature instable, ou lent, ou peu souple. Si on opte pour un Unix gratuit et libre, on a le choix entre Linux (se tourner alors vers Debian ou Jurix, en évitant soigneusement RedHat) et *BSD ; NetBSD est plus stable que Linux, et pour mettre l'accent sur la sécurité on choisira OpenBSD. Sinon, le bon vieux DOS peut aussi faire l'affaire, mais les logiciels sont beaucoup moins puissants et assez limités...

De manière générale, un bon firewall doit être strictement impénétrable de l'extérieur. Il doit être impossible de toucher à quoi que ce soit sans être à l'intérieur du site, ou tout du moins sur une liaison sécurisée, cryptée et authentifiée (ssh est pratique pour cela). Pour éviter tout trou de sécurité intempestif, on chassera du firewall tous les serveurs - web, mail, TOUT doit disparaître. Ensuite, on peut se consacrer à la configuration du firewall.

Dernier détail : il faut bien sr se prémunir de toute agression externe, mais il faut aussi prévoir le cas de maladresses internes qui pourraient ouvrir des brèches dans le système de sécurité, exemple : un employé qui lance le serveur Web de Trompe-Page (au hasard) et qui permet du coup à un intrus d'accéder à la totalité du disque dur, et entre autres à des fichiers sensibles.

4.1 Le filtrage

C'est ce qu'on utilise dans le cas o, sans le firewall, toutes les machines seraient accessibles. C'est à dire que chaque machine a une "vraie" adresse IP et est susceptible d'être attaquée, et chaque machine est susceptible de devoir utiliser les services de l'extérieur. Le filtrage consiste à bloquer ou laisser passer des paquets IP suivant certains critères. Attention, il s'agit le plus souvent de critères simples ; ne pas espérer bloquer les paquets "qui contiennent un virus" ou "qui essaient de pirater le site". On peut filtrer :

4.1.1 ...par adresse (niveau IP)

Si l'on a remarqué que des voyous localisés sur piratez.org essaient de s'introduire sur le site, on peut dire au firewall de tout laisser passer, sauf les paquets en provenance et à destination de ce site. Un bon firewall qui fonctionnerait de la sorte interdirait plutôt tous les paquets, sauf ceux qui viennent de sites "srs". En fait, en général les firewalls interdisent tout traffic, sauf le traffic avec certains hôtes de la DMZ - et encore, pas tous.

4.1.2 ...par protocole

L'UDP étant rarement utilisé pour dialoguer avec Internet (on l'utilise majoritairement pour le NFS, un système de partage de fichiers en réseau), on bloque généralement toutes les communications UDP depuis ou vers Internet. De plus, comme la plupart des programmes qui utilisent UDP passent par le système RPC, ils n'utilisent pas forcément toujours les mêmes ports, et on ne peut même pas espérer filtrer par service (voir ci dessous).

Quant à l'ICMP, il peut être utilisé par des intrus pour "sonder" un site, et estimer la topologie interne d'un réseau, en faisant par exemple des calculs de delta sur les différents pings, ou en faisant des traceroute. Certains administrateurs filtrent donc ICMP. De la même manière, au lieu de renvoyer des message ICMP disant "cet hôte est derrière un firewall, passez votre chemin", ils ne renvoient rien du tout (autant laisser l'ennemi dans l'ignorance la plus noire).

4.1.3 ...par service (niveau TCP ou UDP)

Le filtrage par service est possible pour le TCP et l'UDP, mais est surtout utilisé en TCP car les ports UDP peuvent changer (à cause du RPC). Le filtrage par service consiste tout simplement à interdire ou autoriser des connexions en fonction des ports utilisés. Exemple : interdire tous les ports sortants, sauf le 80 (utilisé par les serveurs Web). Inconvénient : si un serveur différent (IRC, par exemple) accepte des connexions sur le port 80, a passera ; et à l'inverse, si un serveur Web attend des connexions sur un autre port, a ne passera pas.

Faon plus élégante de réaliser un accès "Web seulement" : utiliser un serveur mandataire ("proxy", pour ceux qui, comme moi, n'arrivent plus à parler franais). Voir plus bas pour plus de détails. Noter qu'on peut aussi utiliser le bit d'ACK comme règle de filtrage, ce qui permet de n'autoriser que les connexions sortantes (une machine à l'intérieur peut contacter une machine de l'extérieur, mais pas l'inverse). Et évidemment, toutes ces règles peuvent se combiner dans tous les sens, de faon par exemple à tout interdire, sauf toutes les connexions venant d'un autre site (d'une filiale par exemple) et les connexions depuis quelques serveurs bien choisis vers notre serveur SMTP sur le port 25 (protocole et port généralement utlisés pour relayer du traffic e-mail).

4.2 Toujours plus de sécurité

On peut bien sr aller encore plus loin! Les méthodes que je vais évoquer ici sont non seulement parmis les plus sres, mais aussi parmis les plus simples à mettre en oeuvre. Génial, non? Par contre, elles sont aussi les plus restrictives... Tout en laissant aux utilisateurs des machines du réseau privé une marge de manoeuvre que je juge acceptable.

4.2.1 Mandatement ou relais (serveur proxy)

Utilisé le plus souvent de faon consciente pour le Web et le ftp, mais aussi de faon inconsciente pour le DNS, les news et le mail. Prenons un exemple "inconscient", le DNS. Le DNS est le service qui permet de traduire "kangaroo.krypton.fr" en 192.168.5.1. Ce service est en UDP, sur le port 53. Comment faire pour permettre aux machines du réseau privé d'utiliser DNS sans leur ouvrir un port UDP vers le reste du monde? Tout simplement en installant un serveur DNS à l'intérieur du réseau. Comme dans la plupart des cas, c'est déjà fait... Ensuite, il suffit d'autoriser les traffic UDP sur le port 53, mais uniquement pour le serveur DNS. Généralement, les serveurs DNS sont configurés de faon à résoudre d'eux même les adresses locales (le serveur DNS de toto.com connaît les adresses de toutes les machines de son domaine - logique), et à relayer la requête à un autre serveur lorsqu'ils ignorent la réponse. Ils gardent ensuite cette réponse en mémoire pour référence ultérieure.

Autre exemple d'utilisation naturelle d'un système de relais : l'e-mail, et aussi je suppose la plupart des messageries électroniques. Lorsqu'on envoie un message à quelqu'un sur un site, on ne contacte jamais directement la machine de l'utilisateur, ni même parfois le serveur de messagerie du site : on envoie le courrier à son serveur SMTP, et lui va le transmettre au serveur du site du destinataire. Dernier exemple : les forums de discussion USENET (les fameux "newsgroups") ; quand un site monte son propre serveur news, cela permet aux utilisateurs d'y avoir accès sans se connecter à l'extérieur. Tous ces serveurs – DNS, mail, news se trouvent généralement dans la DMZ, ou, pour les petits sites, sur le firewall (ce qui est extrêmement déconseillé!)

Petite note culturelle : on traduit parfois "gateway" par passerelle, et on est alors amené a employé ce terme pour désigner un routeur. Comme me l'a fait remarquer quelqu'un qui semble en connaître assez long sur ce sujet : -), un routeur désigne un élément matériel (avec éventuellement un peu de logiciel) destiné, comme son nom l'indique, à router des paquets, c'est à dire les envoyer vers tel ou tel endroit afin de les approcher de leur destination finale. Exemple de routeur : les plus ou moins petites boîtes magiques portant des étiquettes prestigieuses comme Cisco, Ascend... Ou encore, une machine sous Linux avec le noyau approprié. Une passerelle, par contre, est essentiellement logicielle, exemple : un serveur de messagerie qui relaie du courrier électronique.

Ensuite, les exemples "conscients" : c'est le cas du proxy Web. Dans la plupart des navigateurs actuels, on peut spécifier un "serveur proxy", et au lieu de contacter directement le site, le navigateur demandera à ce fameux serveur de faire la requête à sa place. Là encore, ce serveur est dans la DMZ. Il existe aussi certains firewall (exemple : le système SOCKS) qui permettent d'ouvrir arbitrairement des ports, moyennant un compte sur le firewall. Ainsi on peut communiquer plus librement, mais chaque connexion est enregistrée dans un "log" (fichier journal) qui servira en cas d'incident de sécurité.

4.2.2 Translation d'adresses (masquerading, NAT)

C'est le système fréquemment utilisé sous Linux ou *BSD, et qui a un très grand mérite : il ne demande aucune configuration sur le client (hormis l'adresse IP, réseau et routeur ; et encore, en DHCP cela peut être entièrement automatisé). Cela fonctionne comme un mandatement généralisé. Le serveur responsable de cette opération a dans sa table de routage ou de firewall une entrée du type "masquerade tous les paquets venant du réseau X, destinés au réseau Y, sur le port Z". En général, X désigne le réseau interne, Y le réseau Internet, et Z tous les ports (sauf si on veut restreindre à l'usage du Web, par exemple). Concrètement, lorsque le serveur reoit un paquet venant du réseau X et destiné au réseau Y, il "maquille" ce paquet, en faisant croire qu'il vient de lui-même ; et il note le port source qu'il utilise de son côté. Et ensuite il l'envoie sur Y. Lorsque les réponses arrivent - sur le port source, qu'il a noté - il remaquille les paquets et les renvoie à l'hôte du réseau X, qui n'y voit que du feu.

C'est à ce jour la seule solution que je connaisse qui marche sur toutes les plateformes sans installation d'aucun logiciel particulier. Les gens pointilleux remarqueront un détail : cela ne peut marcher que pour les protocoles o il y a une notion de port, puisque j'ai parlé de port source. Eh bien a marche aussi pour ICMP, mais au lieu de noter le port, le serveur note un numéro de séquence.

La traduction d'adresses, comme tous les autres systèmes de mandatement, a cela de magique qu'un hôte du réseau privé a un accès quasi illimité au monde extérieur, mais que le monde extérieur ne peut en aucun cas s'introduire sur le réseau privé, puisque les hôtes de ce réseau n'existent pas, de l'extérieur. Le seul moyen est alors de s'introduire dans le firewall - ce qui est très difficile, puisque nous avons consciencieusement désactivé tous les serveurs, et tous les moyens d'accès extérieurs.

4.2.3 Utilisation de "jetons"

C'est le principe expliqué plus haut pour les firewall de type SOCKS : lorsqu'un utilisateur veut accéder au monde extérieur, il doit effectuer une certaine manoeuvre (exemple : faire un telnet sur un hôte, se loger et lancer un certain programme), ce qui lui ouvre un accès vers l'extérieur. L'intérêt principal est de pouvoir garder traces de toutes les demandes effectuées par les utilisateurs, et aussi d'avoir un système qui marche pour la plupart des applications (pour un firewall SOCKS, il faut une application "socksifiée", ce qui demande au moins une recompilation - pas de problème sous Unix, mais sous Windows... ouh làlà...). C'est cependant peu pratique de devoir faire un telnet à chaque fois qu'on veut accéder au réseau ; de plus les utilisateurs peuvent être tentés de faire un petit programme qui automatise le telnet en tapant leur login et leur mot de passe ; et quelqu'un pourrait récupérer ce programme et en extraire ces informations, usurpant l'identité de l'utilisateur.

5 Cerise sur le gâteau : le VPN

Cette section est encore en développement. Tout ce que je peux dire, c'est que le VPN permet à des petits réseaux locaux comme ceux que nous avons tous (quelques machines en LAN, avec une liaison PPP à internet) de se relier les uns aux autres afin de communiquer de faon transparente. Cela repose sur l'utilisation de PPP et d'encapsulation des données, et d'une dorsale virtuelle... (envoyez le générique de twilight zone, a le fait!!!)

6 Remerciements

ceux et celles qui m'ont envoyé des corrections. Si vous souhaitez que votre nom apparaisse, envoyez moi un mail (je ne voulais pas le faire sans autorisation).

Spéciale dédicace à : (par ordre analphabètique)

Amfo - c'est lui qui, le premier, a commis l'erreur de me passer la distrib infomagic automne 96. Ensuite, c'est avec l'aide de son NAG en franais que j'ai configuré mon réseau local. Merci Julien...
Krystal - mais tu vas le lire ce putain de NAG ou il faut que je te le fasse bouffer? !?
Shadow - VMS Ope'lato'l Awaiting Inst'luctions : -)

... et tous les garons et les filles qui, ici ou là, sont root sur un Linux, un *BSD, ou n'importequoi d'autre qui a le got d'Unix. (Et il ne suffit pas de taper “root” dans le login windows! Usurpateurs!)