Administration de réseaux

Marc Baudoin

1

Introduction

L'ADMINISTRATION réseau, de même que l'administration système d'ailleurs, est une discipline qui ne s'enseigne pas. Ceci peut paraître paradoxal puisque ce document est le support d'un cours d'administration réseau, justement. Relativisons les choses, si l'administration réseau ne s'enseigne pas, en revanche, elle s'apprend et le but de ce cours est de donner aux élèves un minimum d'éléments leur permettant par la suite d'orienter leur apprentissage dans la bonne direction.

Pourquoi l'administration réseau ne s'enseigne-t-elle donc pas ? Tout d'abord, parce c'est un domaine bien trop vaste et qui évolue trop rapidement pour que quiconque puisse le dominer de la tête et des épaules. De plus, le nombre de matériels et de logiciels est trop important pour qu'on puisse en faire une étude sérieuse. De toute façon, chaque entreprise a fait ses choix dans ce domaine et les jeunes ingénieurs auront généralement à s'y plier.

Ce cours ne se veut donc pas exhaustif. En particulier, nous n'aborderons pas du tout la configuration des équipements actifs (routeurs, commutateurs, etc.). Celle-ci nécessiterait un cours entier à elle seule et obligerait à faire un choix partial pour tel ou tel constructeur.

En revanche, dans ce cours, nous essaierons de dégager des principes généraux sur la bonne façon d'administrer un réseau. Le champ d'application étant plutôt étendu, nous nous limiterons à quelques technologies fondamentales, applicables aux réseaux IP sur Ethernet :

les réseaux virtuels (VLAN), sans lesquels on ne peut construire de nos jours un réseau moderne et souple ;

Simple Network Management Protocol (SNMP), le protocole d'administration réseau par excellence, que nous étudierons en détail et dont nous discuterons les avantages et les inconvénients ;

les annuaires, en particulier le Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP) et Lightweight Directory Access Protocol (LDAP) ;

la messagerie, avec l'étude des logiciels sendmail et Postfix.

Par ailleurs, le langage de programmation Perl, outil indispensable à tout administrateur réseau, sera rapidement évoqué, son étude pouvant représenter à elle seule l'objet d'un cours entier.

Mais, tout d'abord, il convient de poser les bases sans lesquelles tout travail sérieux est impossible.

2

Les fondements

Nous allons étudier dans ce chapitre quelques principes fondamentaux que tout administrateur réseau doit constamment avoir à l'esprit.

2.1 Les doigts de pied en éventail

Le but d'un réseau informatique est d'assurer le transport des données de manière automatique. Tout l'art de l'administrateur est de faire en sorte que le réseau puisse fonctionner de manière autonome, de façon à minimiser les interventions manuelles. Par exemple, l'utilisation de protocoles de routage dynamique, tels que RIP ou OSPF, permet de pallier aux défaillances d'un petit nombre d'équipements actifs pour peu que des chemins alternatifs existent. L'utilisation de DHCP permet de simplifier la configuration des ordinateurs et offre plus de souplesse pour modifier le plan d'adressage IP. Il faut absolument prendre en considération tout ce qui permet de simplifier le travail de l'administrateur réseau.

Dans le même ordre d'idées, de nombreux services réseau permettent soit l'utilisation simultanée de plusieurs serveurs soit la mise en place d'un serveur de secours, voire de plusieurs, prenant automatiquement la place du serveur principal en cas d'incident sur ce dernier. Il est essentiel de profiter de ces possibilités afin d'améliorer la disponibilité du réseau et de diminuer les interventions d'urgence. À cet effet, il est préférable de placer le serveur de secours dans un autre local, voire dans un autre bâtiment que celui qui héberge le serveur principal. Si cela n'est pas possible, il faut au moins le relier à une alimentation électrique et à un équipement actif du réseau différents de ceux du serveur principal.

Par ailleurs, il convient néanmoins de rester prudent face à ce qui peut simplifier la vie de l'administrateur réseau. Un système qui fonctionne correctement lorsqu'il s'agit de gérer quelques dizaines de machines ou de connexions peut se révéler inadapté pour un nombre plus important. Malheureusement, seules l'expérimentation (quitte à en payer les pots cassés) et l'expérience d'autres administrateurs réseau (qui ne coûte pas cher et permet souvent d'éviter les écueils) permettent d'en avoir le cœur net.

Enfin, on aura beau avoir tous les systèmes redondants du monde, il est néanmoins nécessaire d'effectuer une surveillance rapprochée de son réseau afin d'être au courant des incidents et de pouvoir réagir en conséquence. Parmi les logiciels

de surveillance, on peut citer Nagios [1] qui, outre sa gratuité, a l'avantage d'être particulièrement polyvalent.

2.2 Le matériel

2.2.1 Méfiance face aux fournisseurs

C'est bien connu, les fournisseurs sont là pour vous faire acheter leurs matériels. Ils sont en revanche beaucoup plus discrets lorsqu'il s'agit de résoudre les problèmes que leur utilisation pourrait entraîner. Le souci le plus commun survient lors de la mise à jour des logiciels (en particulier les systèmes d'exploitation des matériels actifs) vers une version plus récente. Certains constructeurs poussent même le vice jusqu'à sortir des versions mineures de leurs logiciels chaque semaine. Dans ces conditions, il est bien évidemment impossible d'effectuer des mises à jour aussi fréquemment.

La démarche la plus sage consiste à bloquer son parc sur une version particulière des logiciels, dont on aura constaté la stabilité, et de ne les mettre à jour que pour de bonnes raisons (cela peut être la disponibilité de nouvelles fonctionnalités, l'amélioration des performances, la correction d'un problème de sécurité, etc.).

2.2.2 La maintenance

Les équipements tombent en panne un jour ou l'autre, c'est dans l'ordre des choses.

En conséquence, tous les matériels doivent disposer d'une maintenance permettant de faire remplacer les pièces défectueuses. Il existe différents délais de remplacement, plus ou moins rapides, et c'est à chacun de définir ce qui convient le mieux à son réseau, en fonction de ses contraintes de fonctionnement et de ses moyens financiers, les délais de remplacement les plus courts étant évidemment les plus coûteux.

Une option intéressante, pour les parcs d'une taille suffisante et suffisamment homogènes, est de disposer d'un équipement supplémentaire de chaque type et de souscrire la maintenance la plus lente possible lorsque ceci est rentable. Il est très improbable d'avoir deux pannes en même temps donc, en cas de défaillance, c'est l'équipement supplémentaire (ou l'un de ses composants) qui remplacera dans un délai très court l'équipement défaillant. Celui-ci sera alors remplacé au titre de la maintenance par un équipement qui deviendra le nouvel équipement de secours.

2.2.3 La valorisation des vieux équipements

Il existe des entreprises spécialisées dans la vente et l'achat d'équipement informatique d'occasion, dont le matériel réseau. Lors de la réforme d'anciens appareils,

1. http://www.nagios.org/

il peut être intéressant d'envisager leur revente à l'une de ces entreprises, certains matériels pouvant avoir une valeur résiduelle non négligeable.

2.3 Un peu d'administration système

L'administration réseau est rarement totalement découplée de l'administration système, pour la simple raison que le bon fonctionnement d'un réseau repose généralement sur un certain nombre de serveurs. Il n'est donc pas inutile de rappeler un certain nombre de règles élémentaires d'administration système.

Un serveur fiable a toujours au moins deux disques. L'un utilisé uniquement pour le système (donc, grosso modo, contenant `/`, `/usr`, `/var` et du swap), l'autre contenant les comptes des utilisateurs (`/home`) et surtout les logiciels recompilés (`/usr/local`) ainsi que les fichiers de configuration de ces logiciels (par exemple `/opt`), de manière à séparer les fichiers résultant d'installation de logiciels, qui sont dans `/usr/local`, et les fichiers générés par des humains, qui sont dans `/toto`. De cette façon, en cas de problème matériel sur le serveur, il est très simple de déplacer le second disque sur une machine installée de la même façon (donc avec un premier disque identique). Une autre approche est de placer `/usr/local` et `/toto` sur un serveur NFS mais on préfère généralement les disques locaux pour éviter les dépendances entre machines.

Un serveur est rarement administré par une seule personne. Il est donc nécessaire de gérer l'accès concurrent aux fichiers modifiables par les administrateurs. Une façon élégante de faire est d'utiliser RCS (voir l'annexe A).

2.4 La sécurité est essentielle

Lorsqu'on parle de sécurité, cela recouvre un domaine très vaste, comprenant entre autres le contrôle d'accès, l'intégrité, la confidentialité et la disponibilité.

Parmi ces aspect, la disponibilité a été traitée au paragraphe 2.1, l'intégrité est assurée en partie par les sommes de contrôle de TCP et d'UDP (rien n'interdisant un intrus placé sur le chemin d'un paquet de modifier les paquets et de recalculer les sommes de contrôle en conséquence), la confidentialité est assurée par IPsec ou par des dispositifs de chiffrement spécifiques.

Le contrôle d'accès, quant à lui, doit être imposé sur tous les équipements actifs et les serveurs, afin que seules les personnes autorisées y aient accès. Il est en effet particulièrement désagréable de voir son travail ruiné par un intrus qui aurait modifié, voire effacé, la configuration de ses équipements. Pour s'en prévenir, outre un contrôle d'accès adéquat, il est préférable de conserver en sûreté les configurations de tous les équipements, afin de pouvoir les restaurer rapidement en cas de problème. En particulier, de nombreux équipements actifs acceptent soit d'être configuré en direct (par l'intermédiaire d'une connexion réseau ou d'une connexion sur port série) soit de télécharger leur configuration, généralement par TFTP. Cette dernière

approche est à privilégier puisqu'elle permet de conserver la trace de la configuration sur le serveur TFTP.

2.5 Les fichiers de configuration

D'ailleurs, quasiment tous les équipements actifs acceptent de télécharger leur configuration, en totalité ou par morceaux, depuis un serveur (généralement par TFTP). Les mauvais administrateurs réseau s'enorgueillissent de pouvoir générer à la main de nombreux fichiers de configuration plus complexes les uns que les autres, au risque d'y introduire des erreurs de syntaxe ou, plus grave, d'avoir à gérer la redondance des informations entre plusieurs fichiers.

En revanche, l'administrateur réseau futé et qui de plus applique le principe des doigts de pied en éventail adopte plutôt un autre principe lorsque cela est possible (attention, ce n'est pas toujours le cas). Il met au point un certain nombre de programmes qui permettent, à partir d'informations élémentaires et non redondantes, de générer les différents fichiers de configuration qui en découlent.

Prenons un exemple concret tiré d'une situation réelle. Le DNS, comme vous ne le savez peut-être pas encore (dans ce cas, vous pouvez vous référer au paragraphe 5.1) permet de connaître l'adresse IP associée à un nom de machine et vice versa. Pour cela, le serveur a besoin de deux fichiers de configuration, l'un contenant, pour chaque nom, l'adresse IP correspondante, et l'autre contenant, pour chaque adresse IP, le nom correspondant. Ceci est une vision un peu simplifiée de ce que permet de faire le DNS mais elle couvre son utilisation habituelle. Il est évidemment idiot de gérer ces deux fichiers à la main (sauf lorsqu'on n'a qu'une dizaine de machines à y enregistrer), puisqu'ils contiennent somme toute les mêmes informations. De plus, leur format est adapté à leur interprétation par un logiciel mais pas vraiment à leur rédaction manuelle. Il est plus simple de ne gérer qu'un seul fichier, contenant par exemple des lignes de la forme :

```
nom_de_machine   adresse_IP
```

et de générer à partir de celui-ci les deux fichiers de configuration du DNS au moyen d'un programme maison. Non seulement cette méthode est plus simple et plus rapide (principe des doigts de pied en éventail) mais elle permet également d'avoir des fichiers de configuration exempts d'erreurs de syntaxe (pour peu que le programme maison soit bien conçu). Le programme maison peut également en profiter pour redémarrer le serveur DNS après avoir généré les fichiers de configuration, ce qui évite une manipulation supplémentaire.

Mais on peut aller plus loin. De nos jours, on ne configure plus les paramètres réseau des ordinateurs (adresse IP, masque de sous-réseau, routeur par défaut...) manuellement, on utilise DHCP en attribuant à chaque machine soit une adresse prise au hasard dans un ensemble donné soit une adresse fixe choisie en fonction de l'adresse Ethernet de la machine (cette dernière méthode est d'ailleurs préférable car

elle permet de suivre les incidents plus facilement). Pour cela, le serveur DHCP a besoin d'un fichier de configuration contenant un certain nombre d'informations dont, pour chaque machine, son adresse Ethernet et son adresse IP ou son nom (dans ce cas, le serveur DHCP utilise le DNS pour faire la conversion). En étendant le format du fichier unique utilisé pour le DNS à quelque chose du genre :

```
nom_de_machine   adresse_IP   adresse_Ethernet
```

on peut continuer à utiliser ce fichier pour le DNS (il suffit d'ignorer le dernier champ) et également pour générer le fichier de configuration du serveur DHCP à l'aide d'un deuxième programme maison.

J'arrêterai cet exemple ici, mais on peut encore l'étendre à la configuration des commutateurs.

2.6 Perl

Nous venons de voir que l'administrateur réseau futé est celui qui se simplifie la vie en mettant au point de petits programmes lui permettant de générer des fichiers de configuration complexes à partir de fichiers beaucoup plus simples. Ceci peut, bien entendu, être réalisé au moyen de n'importe quel langage de programmation mais l'un d'eux se révèle particulièrement adapté à ce type de travail, il s'agit de Perl.

Perl, également très utilisé en administration système, est incontournable dès qu'il est question d'analyser et de générer des fichiers au format texte parce que les outils le permettant sont intelligemment inclus dans le langage lui-même. Sorte de mélange de nombreux autres langages, reprenant à chacun ce qu'il sait bien faire, Perl dispose de plus d'une immense bibliothèque de modules permettant de réaliser très simplement à peu près tout ce à quoi l'on peut penser.

Ici n'est malheureusement pas la place pour un cours détaillé sur Perl. Une présentation très rapide est fournie dans le cours B1-3.

3

Les réseaux virtuels (VLAN)

3.1 Historique

Les premiers réseaux Ethernet (on se situe donc en couche 2) étaient conçus à base de câbles coaxiaux raccordés entre eux et connectés aux ordinateurs, si bien que tout signal électrique émis par l'un d'eux était reçu par tous les autres. L'ensemble des machines ainsi reliées entre elles s'appellait un *domaine de collision* (puisque ces machines partageaient le même médium physique).

Dans le cas de grands réseaux locaux, il était impossible d'avoir un seul domaine de collision (pour des raisons d'éloignement géographique, de longueur de câbles, de temps de propagation ou à cause du nombre trop important d'ordinateurs) et il fallait donc concevoir des domaines de collision de taille raisonnable, reliés entre eux par des routeurs. Des ponts Ethernet n'auraient pas suffi car ils augmentent le temps de propagation des signaux électriques, d'où la nécessité de remonter en couche 3. Chacun de ces domaines de collision était également appelé *segment Ethernet*.

À chaque segment Ethernet correspondait donc un sous-réseau IP. Au bout du compte, on aboutissait à un découpage logique calqué très exactement sur le découpage physique du réseau. Cela imposait une proximité géographique des machines si l'on voulait qu'elles appartiennent au même segment Ethernet, ce qui n'est pas nécessairement pratique. Par ailleurs, ceci limitait grandement la mobilité des ordinateurs.

Après l'arrivée des premiers commutateurs, de nouvelles possibilités sont apparues. Compte tenu de l'électronique interne des commutateurs, plus complexe que celles des répéteurs, il devenait possible de disposer de plusieurs segments Ethernet au sein d'un même commutateurs (ce qui était d'ailleurs déjà possible à moindre échelle dans les répéteurs segmentables). Mais, en ajoutant quelques en-têtes supplémentaires aux trames Ethernet, il devenait possible d'étendre la taille de ces segments Ethernet à l'ensemble d'un réseau de commutateurs interconnectés. Les réseaux virtuels (*virtual LAN*, VLAN) étaient nés.

3.2 Principe

Un VLAN est l'équivalement moderne des segments Ethernet de l'ancien temps. Tous les ordinateurs faisant partie d'un même VLAN sont capables de

communiquer entre eux directement sans avoir à passer par un routeur. On ne parle plus de domaine de collision, étant donné qu'il n'y a pas de collisions avec des commutateurs, mais de *domaine de diffusion*, puisqu'une trame de diffusion émise par un ordinateur sera reçue par toutes les machines faisant partie du même VLAN.

Un VLAN peut être local à un commutateur ou s'étendre à un ensemble de commutateurs reliés entre eux. On a donc la possibilité d'organiser la structure logique de son réseau sans avoir à se soucier de sa structure physique, ce qui apporte une souplesse fort appréciable.

Dans le cas où une trame Ethernet doit être transportée d'un commutateur à un autre, il est nécessaire d'y rajouter quelques informations (en particulier le VLAN auquel elle appartient). C'est le but du standard IEEE 802.1Q.

3.3 Le standard IEEE 802.1Q

Approuvé le 8 décembre 1998, le standard 802.1Q du comité 802 [1] de l'IEEE (*Institute of Electrical and Electronics Engineers*) est aujourd'hui le standard de fait pour l'identification des trames faisant partie d'un réseau virtuel.

Il a succédé à ISL (*Inter-Switch Link*), protocole propriétaire développé par Cisco (et également repris par quelques autres constructeurs).

Le principe général est de rajouter dans chaque trame Ethernet destinée à être transmise d'un commutateur à un autre quelques en-têtes supplémentaires contenant en particulier l'identifiant du réseau virtuel auquel elle appartient (VID, *VLAN Identifier*), qui est un numéro sur 12 bits, de 0 à 4094 (4095 est réservé et, en pratique, 0 et 1 sont inutilisés).

L'étude détaillée des 211 pages du standard 802.1Q [2] ne présente que peu d'intérêt et nous allons donc nous concentrer sur son utilisation pratique.

3.4 En pratique

Les constructeurs de commutateurs imposent en général des limitations quant au nombre de réseaux virtuels que leurs matériels peuvent gérer. Les plus petits modèles ne savent souvent en gérer que quelques dizaines alors que les châssis offrent un éventail plus large. Il convient donc d'y prendre garde lors de la définition des VLAN de son réseau et du choix de son matériel.

Il faut également prendre en compte les possibilités offertes par les commutateurs des différents constructeurs. Les plus simples ne permettent de placer un port (et donc les ordinateurs qui y sont connectés) que dans un seul VLAN de manière statique, les plus évolués permettent de choisir le VLAN de manière dynamique en fonction de divers paramètres (adresse Ethernet, protocole, etc.).

1. http://www.ieee802.org/
2. http://standards.ieee.org/getieee802/download/802.1Q-1998.pdf

Mais il faut avant tout déterminer comment découper son réseau et selon quels critères. Il n'y a évidemment pas de règle générale mais l'usage le plus répandu est de calquer l'organisation du réseau sur l'organisation administrative de l'entreprise.

4

Simple Network Management Protocol (SNMP)

SNMP permet à l'administrateur réseau, depuis un poste de contrôle central, d'obtenir et de modifier divers éléments de configuration d'équipements actifs et de logiciels.

SNMP est un paradoxe dans le monde de l'administration réseau. Conçu comme un protocole censé unifier et simplifier la gestion des matériels et des logiciels (ce qui est un but absolument louable), il n'a pas encore réussi à s'imposer auprès des administrateurs pour des raisons parfaitement valables :

- SNMP n'est pas si simple que ça (lisez donc les spécifications des différentes versions de ce protocole pour en avoir le cœur net) ;
- sa mise en œuvre (autrement que pour de petits besoins ponctuels) nécessite des investissements gigantesques en logiciels d'administration ;
- sa sécurité laisse à désirer (c'est un comble pour un protocole dédié à l'administration) bien que la situation se soit bien améliorée avec SNMPv3.

En fait, l'utilisation systématique de SNMP est difficile à justifier lorsqu'on n'administre pas un gigantesque réseau composé de matériels et de logiciels hétérogènes et qu'on n'a pas quelques centaines de milliers d'euros à y consacrer.

Cependant, dans certains cas, SNMP peut tout de même s'avérer fort utile si l'on sait l'utiliser à bon escient.

Ce chapitre ne se veut pas exhaustif, le lecteur désirant approfondir son étude de SNMP pourra se reporter aux URL suivantes :

```
http://www.snmplink.org/
http://www.faqs.org/faqs/by-newsgroup/comp/comp.protocols.snmp.html
```

4.1 Historique

Trois versions successives de SNMP se sont succédées :

- SNMPv1, en 1990, décrit dans les RFC 1155 à 1157 ;
- SNMPv2, en 1996, décrit dans les RFC 1901 à 1908 ;
- SNMPv3, en 1999, décrit dans les RFC 2571 à 2575.

Bien que SNMPv3 soit maintenant assez ancien, rares sont les matériels qui sont capables de le gérer. Nous nous concentrerons donc sur SNMPv2.

4.2 Sécurité

Le protocole permettant la gestion des équipements actifs du réseau devrait logiquement être le plus sécurisé de tous les protocoles avec, en particulier, un contrôle d'accès et une confidentialité irréprochables.

Tous les documents décrivant SNMP (avant la version 3) se contentaient, au paragraphe *Security Considerations*, d'un commentaire laconique :

> *Security issues are not discussed in this memo.*

Et, en effet, le contrôle d'accès était approximatif et la confidentialité inexistante.

SNMPv3, quant à lui, accorde enfin une large part de sa spécification au contrôle d'accès, la confidentialité étant naturellement laissée à IPsec.

4.3 Principes

4.3.1 Les agents

Chaque équipement ou logiciel pouvant être interrogé par SNMP comporte un serveur appelé *agent*. Celui-ci écoute sur le port UDP 161.

4.3.2 Les MIB

Les paramètres pouvant être examinés ou modifiés par l'agent sont définis dans une MIB (*Management Information Base*), qui est un document décrivant ces différents paramètres, leur type (nombre entier, chaîne de caractères...), s'ils sont modifiables ou non (dans l'absolu, indépendamment des droits d'accès), etc.

La MIB la plus utilisée est certainement la MIB-II, décrite dans le RFC 1213, qui définit un ensemble cohérents de paramètres communs à tous les équiments. C'est cette MIB que nous étudierons par la suite.

Chaque élément d'une MIB est défini par un nom et un numéro (comme d'habitude, on retrouve cette dualité, les noms étant plus faciles à manipuler pour nous, pauvres humains, et les nombres plus faciles à manipuler par les ordinateurs). Ainsi, dans la MIB-II, l'objet `sysContact`, indiquant le nom de la personne responsable d'un équipement, est défini ainsi :

```
sysContact OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The textual identification of the contact person
            for this managed node, together with information
            on how to contact this person."
    ::= { system 4 }
```

SYNTAX indique le type de l'objet selon la syntaxe ASN.1 (*Abstract Syntax Notation One*).

ACCESS indique le mode d'accès à l'objet. Les valeurs suivantes sont possibles :

- `read-only`
- `read-write`
- `write-only`
- `not-accessible`

STATUS indique si l'objet doit obligatoirement être présent dans toute implémentation de la MIB ou pas. Les valeurs suivantes sont possibles :

- `mandatory`
- `optional`
- `obsolete`

DESCRIPTION est un texte destiné à l'administrateur et qui décrit l'objet.

La dernière ligne de la définition attribue à l'objet `sysContact` le numéro 4 dans le groupe `system`. En effet, les MIB sont hiérarchisées à la manière des répertoires dans un système de fichiers. Le nom complet de cet objet au sein de la MIB-II est donc `system.sysContact` (le point est utilisé comme séparateur) ou bien `system.4` ou bien `1.sysContact` (`system` a pour numéro 1) ou, encore moins lisible, `1.4`.

La MIB-II fait elle-même partie d'une hiérarchie plus large :

```
.iso.org.dod.internet.mgmt
.1.3.6.1.2
```

- La racine de la hiérarchie n'a pas de nom et est désignée simplement par un point.
- `iso` (1) désigne l'*International Organization for Standardization*.
- `org` (3) a été créé par l'ISO à l'intention de divers organismes.
- `dod` (6) a été attribué au ministère de la Défense des États-Unis (*Department of Defense*).
- `internet` (1) regroupe tout ce qui touche à l'Internet.
- `mgmt` (2), enfin, est utilisé pour les standards de l'IAB.

Sous `.iso.org.dod.internet.mgmt`, la MIB-II a pour nom `mib-2` et pour numéro 1, de telle sorte que l'objet `sysContact` a pour nom absolu :

```
.iso.org.dod.internet.mgmt.mib-2.system.sysContact
.1.3.6.1.2.1.1.4
```

En pratique, cet objet est d'un type discret (ce n'est pas un tableau et il ne contient donc qu'une valeur) donc on lui ajoute un `.0` final, ce qui donne comme nom absolu :

```
.iso.org.dod.internet.mgmt.mib-2.system.sysContact.0
.1.3.6.1.2.1.1.4.0
```

Les objets pouvant contenir plusieurs valeurs se voient ajouter à leur nom un `.1` pour la première valeur, `.2` pour la deuxième et ainsi de suite.

Somme toute, SNMP est simple !

4.3.3 Les opérations

Sans rentrer dans le détail du protocole et des formats de paquets, il est néanmoins intéressant de savoir un minimum comment fonctionne le dialogue entre un agent SNMP et un logiciel d'administration.

Il existe *grosso modo* quatre types de messages :

Get permet de récupérer un objet bien précis.

Get next renvoie l'objet suivant (dans l'ordre lexicographique) celui passé en paramètre. Ceci est particulièrement utile pour passer en revue toute une MIB.

Set permet de modifier la valeur d'un objet.

Trap permet à un agent d'envoyer un signal au logiciel d'administration et ce de son propre chef. Les trois types de messages précédents étaient envoyés par le logiciel d'administration à l'agent, celui-ci fonctionne en sens inverse. Il est très utile pour effectuer une surveillance passive du réseau, les équipements se plaignant si nécessaire.

4.3.4 Les communautés

Dans SNMPv2, le contrôle d'accès est fait en fournissant dans chaque message un mot de passe appelé *communauté*. Il existe généralement deux communautés, l'une utilisée pour les accès en lecture (c'est par défaut la chaîne de caractères `public`), l'autre utilisée pour les accès en écriture (c'est par défaut la chaîne de caractères `private`). Il est évidemment essentiel de modifier ces valeurs et de les tenir secrètes.

4.4 net-snmp

La plupart des équipements réseau intègrent un agent SNMP. Certains logiciels également, de même que les systèmes d'exploitation commerciaux. Les UNIX libres, quant à eux, utilisent la suite logicielle net-snmp. Celle-ci regroupe un agent, divers outils d'interrogation ainsi qu'une bibliothèque de fonctions C permettant de construire des agents SNMP ou d'intégrer des capacités d'interrogation dans d'autres logiciels.

Nous étudierons dans ce paragraphe l'agent net-snmp ainsi que les outils qu'il fournit.

4.4.1 Récupération des sources

Les sources sont disponibles à l'URL :

```
http://net-snmp.sourceforge.net/
```

4.4.2 Compilation

Une fois l'archive décomprimée, la compilation s'effectue de manière classique :

```
% ./configure
creating cache ./config.cache
[...]
creating config.h
% make
[...]
```

4.4.3 Installation

Après avoir pris l'identité du super-utilisateur :

```
# make install
```

Ceci installe :
- un ensemble de programmes dans `/usr/local/bin` ;
- des bibliothèques dans `/usr/local/lib` afin de pouvoir communiquer par SNMP depuis des programmes en C ;
- les fichiers d'en-tête correspondants dans `/usr/local/include/ucd-snmp` ;
- divers fichiers dans `/usr/local/share/snmp` (fichiers de configuration, MIB) ;
- des pages de manuel dans `/usr/local/man`.

4.4.4 Configuration

L'agent SNMP se configure au moyen du fichier `/usr/local/share/snmp/snmpd.conf`. Bien que non indispensable (l'agent fonctionne fort bien sans ce fichier), il est préférable d'y définir au minimum les paramètres d'emplacement et de contact, ainsi que les droits d'accès (c'est-à-dire les communautés ou des droits d'accès plus complexes) :

```
syslocation     ENSTA - salle verte
syscontact      Séraphin Lampion <seraphin@lampion.org>

rocommunity     public
rwcommunity     private
```

Nous ne détaillerons pas ici la configuration de droits d'accès plus fins. Néanmoins, dans des situations réelles, il est indispensable d'étudier en détail la documentation des agents SNMP afin de limiter au strict nécessaires les possibilités d'accès.

4.4.5 Lancement de l'agent SNMP

L'agent SNMP se lance sous l'identité du super-utilisateur. Compte tenu des différentes actions qu'il peut être amené à accomplir, il n'est pas envisageable d'en restreindre les privilèges.

Les options les plus utiles sont :

-V pour journaliser (dans le fichier `/var/log/snmpd.log` par défaut) les messages reçus et émis par l'agent ;

-d pour journaliser également le contenu des datagrammes (cette option est rarement utilisée, sauf dans des buts pédagogiques) ;

-A pour journaliser à la suite du fichier existant et ne pas en écraser l'ancien contenu ;

-L pour journaliser à l'écran plutôt que dans un fichier (utile uniquement pendant la mise au point du fichier de configuration) ;

-f pour que l'agent ne se mette pas en arrière-plan (utile uniquement pendant la mise au point du fichier de configuration).

En pratique, l'agent se lance ainsi dans une configuration de production :

```
# /usr/local/sbin/snmpd -V -A
```

Lors de la phase de mise au point, il se lance plutôt ainsi :

```
# /usr/local/sbin/snmpd -V -L -f
```

4.4.6 Interrogation d'agents SNMP

4.4.6.1 Parcours d'une MIB

L'outil `snmpwalk` permet de parcourir une MIB (la MIB-II par défaut) au moyen de l'opération *get next*. Il suffit de lui indiquer le nom de la machine et la communauté permettant d'y accéder en lecture :

```
% snmpwalk -v 2c -c public localhost
SNMPv2-MIB::sysDescr.0 = STRING: NetBSD
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.7
SNMPv2-MIB::sysUpTime.0 = Timeticks: (155481919) 17 days, 23:53:39.19
SNMPv2-MIB::sysContact.0 = STRING: Marc Baudoin <babafou@babafou.eu.org>
SNMPv2-MIB::sysName.0 = STRING: babafou.babafou.eu.org
SNMPv2-MIB::sysLocation.0 = STRING: maison
[...]
```

4.4.6.2 Lecture d'un objet

L'outil `snmpget` permet de récupérer la valeur d'un objet bien précis (par défaut dans la MIB-II) :

```
% snmpget -v 2c -c public localhost sysContact.0
SNMPv2-MIB::sysContact.0 = STRING: Séraphin Lampion <seraphin@lampion.org>
```

4.4.6.3 Modification d'un objet

L'outil `snmpset` permet de modifier la valeur d'un objet bien précis (par défaut dans la MIB-II) :

```
% snmpset -v 2c -c private localhost system.sysContact.0 s toto
SNMPv2-MIB::sysContact.0 = STRING: toto
```

4.5 MRTG

Voici un cas pratique, très simple et néanmoins fort utile d'utilisation de SNMP. MRTG est un logiciel permettant d'interroger divers types d'appareils (routeurs, commutateurs, ordinateurs) et de représenter sous forme graphique le trafic réseau sur chacune de leurs interfaces. Vous pouvez en voir quelques exemples à l'URL suivante :

> http://netadm.pasteur.fr/mrtg/

Les graphiques sont réalisés en interrogeant les équipements à intervalle régulier (par défaut, toutes les cinq minutes) par SNMP.

4.5.1 Récupération des sources

Les sources sont disponibles à l'URL :

> http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/

4.5.2 Compilation

Une fois l'archive décomprimée, la compilation s'effectue de manière classique :

```
% ./configure
creating cache ./config.cache
[...]
creating config.h
% make
[...]
```

4.5.3 Installation

Après avoir pris l'identité du super-utilisateur :

```
# make install
```

Ceci installe :
- divers programmes dans `/usr/local/mrtg-2/bin` ;
- de la documentation dans `/usr/local/mrtg-2/doc` ;
- des fichiers additionnels dans `/usr/local/mrtg-2/lib` ;
- des pages de manuel dans `/usr/local/mrtg-2/man`.

MRTG fonctionne également fort bien sans qu'il y ait besoin de l'installer.

4.5.4 Configuration

MRTG se configure au moyen d'un fichier qui peut être généré automatiquement grâce au programme `cfgmaker` fourni avec MRTG :

```
% cfgmaker public@équipement > mrtg.cfg
```

On peut surveiller ainsi plusieurs équipements différents, il suffit de concaténer leurs fichiers de configuration en un seul fichier qui servira de configuration globale à MRTG.

Quelques paramètres peuvent être rajoutés en tête du fichier de configuration global :

```
WorkDir: /machin/bidule/mrtg
Language: french
RunAsDaemon: Yes
Options[_]: growright, bits
```

- Le paramètre `WorkDir` indique le chemin d'accès du répertoire dans lequel MRTG va générer ses fichiers HTML et ses images.
- Le paramètre `Language` demande à MRTG de parler français.
- Le paramètre `RunAsDaemon` indique à MRTG qu'il va fonctionner en tâche de fond (plutôt que d'être lancé par `cron`).
- Les deux valeurs du paramètre `Options` demandent à MRTG respectivement d'afficher un axe des abscisses orienté de gauche à droite (il fait l'inverse par défaut) et d'indiquer des valeurs de trafic en bits par seconde (il compte en octets par seconde par défaut).

4.5.5 Lancement

MRTG se lance en tâche de fond en lui indiquant le nom de son fichier de configuration :

```
% mrtg mrtg.cfg
```

Dans une configuration de production, il conviendra de lancer MRTG sous l'identité d'un utilisateur non privilégié puisqu'il ne nécessite pas les privilèges du super-utilisateur :

```
# su mrtg -c 'mrtg mrtg.cfg'
```

5

Les annuaires

On désigne sous le terme générique *annuaire* tout service permettant d'obtenir des informations à partir d'une base, centrale ou répartie (de même que l'annuaire téléphonique permet d'obtenir le numéro de téléphone à partir du nom de quelqu'un).

Les annuaires sont rarement indispensables mais ils apportent un confort non négligeable soit aux utilisateurs (c'est le cas du DNS) soit à l'administrateur réseau (c'est le cas de DHCP).

Les annuaires peuvent généralement être gérés soit par plusieurs serveurs simultanément soit par un serveur principal et par un ou plusieurs serveurs secondaires qui en prennent le relais en cas de défaillance. Étant donné l'importance que prennent les annuaires dès qu'on commence à les utiliser, il est indispensable de profiter de ces capacités de redondance.

5.1 Domain Name System (DNS)

Le DNS est l'annuaire le plus ancien et certainement le plus utilisé. En effet, dans un réseau IP, chaque machine est repérée par une adresse, qui est un nombre sur 32 bits pour IPv4 et sur 128 bits pour IPv6. Autant les ordinateurs sont plutôt doués pour gérer des nombres, autant nous autres humains le sommes bien moins et il est plus facile pour nous de mémoriser des noms (faites donc l'essai avec vos amis, vous souvenez-vous plus facilement de leurs noms ou de leurs numéros de téléphone ?). Le DNS est un moyen d'associer un nom à chaque adresse IP et vice versa (ceci est une vision réductrice mais elle correspond à l'utilisation principale du DNS).

Le DNS est décrit dans les RFC 1033 à 1035 (datant de novembre 1987). De nombreux autres RFC décrivent des extensions au DNS.

Le DNS utilise les ports UDP et TCP 53.

5.1.1 Un peu d'histoire

Dans les temps anciens, à l'époque où l'Internet n'était composé que d'une poignée de machines sur des sites qui se comptaient sur les doigts des deux mains, on disposait déjà d'un système permettant d'associer des noms aux adresses IP. Il était

basé sur un fichier, très semblable à l'`/etc/hosts` des systèmes UNIX d'aujourd'hui, qui devait être présent à l'identique sur toutes les machines de l'Internet. Pour illustrer ce retour aux sources, considérons le fichier suivant, dont nous utiliserons les informations par la suite :

```
147.250.14.1    guinness
147.250.14.2    blanche
```

Évidemment, l'ajout d'une nouvelle machine impliquait la recopie du nouveau fichier sur toutes les autres et ce système a très vite montré ses limites, d'où l'invention du DNS.

5.1.2 Principes de fonctionnement

Le DNS est une base de données répartie. Par rapport au système précédent, on ne dispose plus d'un fichier unique de correspondances entre adresses IP et noms mais de plusieurs et chaque site maintient les informations correspondant à ses machines et les met à la disposition du reste de l'Internet par un protocole approprié. Pour savoir quel serveur interroger, le système de nommage a été hiérarchisé. À cet effet, l'Internet a été découpé en *domaines*. Un domaine correspond à un ensemble de machines dépendant administrativement de la même entité.

5.1.2.1 Le système de nommage

Toute hiérarchie commençant quelque part, la racine du DNS s'appelle « . ». Sous cette racine ont été créés des *top-level domains* (TLD) permettant d'effectuer un premier rangement des niveau inférieurs. Parmi les TLD, on retrouve :

des domaines fonctionnels créés à l'origine par et pour les américains, ce qui explique leur vision un peu réduite des choses :

- `com` pour les entreprises,
- `edu` pour les universités,
- `gov` pour les institutions gouvernementales,
- `int` pour les organismes internationaux,
- `mil` pour les organismes militaires (l'Internet ayant été inventé par les militaires américains),
- `net` pour les fournisseurs d'accès à l'Internet,
- `org` pour les autres organisations ;

des domaines nationaux créés par la suite selon la norme ISO 3166-1 :

- `au` pour l'Australie,
- `de` pour l'Allemagne,
- `fr` pour la France,
- etc.

Ces TLD ont des sous-domaines, comme par exemple `ensta.fr` qu'on devrait d'ailleurs écrire `ensta.fr.` (notez le point final) si l'on voulait être exact. On remarque donc que le point sert à la fois à désigner le domaine racine et à séparer les différents composants des noms de domaine (un peu comme le `/` pour les chemins d'accès sous UNIX à la différence que les chemins d'accès se lisent de gauche à droite alors que les noms de domaine se lisent de droite à gauche).

L'entité qui a obtenu la *délégation* (c'est-à-dire la gestion) d'un domaine peut alors créer à loisir des noms de machines dans ce domaine et leur associer des adresses IP (puisque c'est tout de même le but du DNS). Lorsqu'on a beaucoup de machines, il peut s'avérer utile de créer des sous-domaines afin de permettre un nommage plus propre en introduisant un ou plusieurs niveaux de hiérarchie supplémentaires. Par exemple, nous travaillerons par la suite avec le domaine `tp.ensta.fr`.

Le *nom qualifié* d'une machine (en anglais, *fully qualified domain name* ou FQDN), c'est-à-dire son nom complet comprenant le domaine, par exemple `guinness.tp.ensta.fr`, ne peux excéder 255 caractères et chacun des composants (les parties entre deux points successifs) ne peut excéder 63 caractères, ce qui laisse tout de même de la marge. Un nom de machine ou un nom de domaine ne peut contenir que des lettres (majuscules ou minuscules, aucune différence n'est faire à ce niveau), des chiffres ou des tirets. Tout autre caractère est interdit.

Un point de vocabulaire : on distingue la zone `ensta.fr` du domaine `ensta.fr`. La zone `ensta.fr` ne contient que les informations concernant les machines situées directement sous `ensta.fr` (donc la machine `guinness.tp.ensta.fr` n'en fait pas partie puisqu'elle appartient à un sous-domaine d'`ensta.fr`), alors que le domaine `ensta.fr` contient la zone `ensta.fr`, les zones correspondants aux sous-domaines d'`ensta.fr` et ainsi de suite.

5.1.2.2 Les serveurs

Chaque zone dispose d'un ou de plusieurs serveurs DNS. S'il y en a plusieurs, l'un d'eux est dit *maître* et les autres sont ses *esclaves* (on parlait de *primaire* et de *secondaires* dans l'ancienne terminologie du DNS). Le serveur maître est le vrai détenteur des informations de la zone, les esclaves se contentent de les recopier.

À chaque modification des informations d'une zone, le serveur maître avertit ses esclaves pour qu'ils puissent se mettre à jour. Toute modification sur le maître se répercute donc très rapidement sur ses esclaves. L'opération de mise à jour s'appelle un *transfert de zone*.

Le serveur maître et ses esclaves font *autorité* sur les zones qu'ils gèrent (c'est-à-dire qu'eux seuls détiennent les tables de correspondance officielles pour ces zones).

5.1.2.3 La résolution de nom

Si j'ai besoin de savoir l'adresse IP associée à une machine de la zone `ensta.fr`, je vais m'adresser à l'un des serveurs qui font autorité pour cette zone. Mais quels sont ces serveurs ? Comme la structure du DNS est hiérarchique, il suffit de le demander

à l'un des serveurs de la zone `fr`. Si je ne les connais pas non plus, je vais demander leurs adresses à l'un des serveurs de la racine. Puisqu'il faut bien qu'une arborescence commence quelque part, ces derniers sont connus (il y en a actuellement treize, répartis sur la planète) et permettent donc de toujours pouvoir descendre l'arbre du DNS. L'ensemble de ce processus s'appelle la *résolution de nom*.

Un processus qui a besoin de convertir un nom de machine en adresse IP n'effectue jamais la résolution de nom lui-même. Pour cela, il s'adresse au serveur DNS de son site, dont l'adresse IP (pas le nom, pour des raisons évidentes) figure dans le fichier `/etc/resolv.conf` (ce fichier peut contenir plusieurs adresses de serveurs de noms, ils sont alors interrogés dans l'ordre jusqu'à obtenir une réponse). C'est le serveur DNS local qui va effectuer la résolution de nom et en renvoyer le résultat au processus demandeur. En prime, le serveur DNS va conserver la réponse dans un cache, afin d'éviter de refaire cette gymnastique s'il reçoit la même requête dans le futur.

Si plusieurs adresses IP différentes sont associées au même nom de machine (ce qui peut arriver, par exemple dans le cas de services redondants), un serveur DNS donné renverra successivement la première, puis la deuxième et ainsi de suite jusqu'à la dernière, puis il reprendra du début. Ce mécanisme s'appelle le *tourniquet* (*round-robin* en anglais) et permet de faire une répartition de charge naturelle entre des machines différentes mais répondant au même nom (ce qui est donc transparent pour l'utilisateur).

5.1.3 Interrogation du DNS avec `dig`

Le DNS est généralement interrogé de manière transparente par divers programme mais il est également possible de l'interroger explicitement, pour tester un serveur ou par simple curiosité. Les principaux programmes permettant ceci sont `dig`, `nslookup` et `host`. Plus spartiate, `dig` est néanmoins plus précis et les puristes le préfèrent aux autres, principalement parce qu'il renvoie des résultats dans un format directement exploitable par un serveur de noms.

Lancé sans argument, `dig` affiche la liste des serveurs de noms de la racine :

```
% dig

; <<>> DiG 9.1.3 <<>>
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 641
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13

;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       253982  IN      NS      I.ROOT-SERVERS.NET.
.                       253982  IN      NS      J.ROOT-SERVERS.NET.
.                       253982  IN      NS      K.ROOT-SERVERS.NET.
.                       253982  IN      NS      L.ROOT-SERVERS.NET.
.                       253982  IN      NS      M.ROOT-SERVERS.NET.
.                       253982  IN      NS      A.ROOT-SERVERS.NET.
.                       253982  IN      NS      B.ROOT-SERVERS.NET.
.                       253982  IN      NS      C.ROOT-SERVERS.NET.
.                       253982  IN      NS      D.ROOT-SERVERS.NET.
.                       253982  IN      NS      E.ROOT-SERVERS.NET.
.                       253982  IN      NS      F.ROOT-SERVERS.NET.
.                       253982  IN      NS      G.ROOT-SERVERS.NET.
.                       253982  IN      NS      H.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
I.ROOT-SERVERS.NET.     474182  IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     474182  IN      A       192.58.128.30
K.ROOT-SERVERS.NET.     474182  IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     474182  IN      A       198.32.64.12
M.ROOT-SERVERS.NET.     474182  IN      A       202.12.27.33
A.ROOT-SERVERS.NET.     474182  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     474182  IN      A       128.9.0.107
C.ROOT-SERVERS.NET.     474182  IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     474182  IN      A       128.8.10.90
E.ROOT-SERVERS.NET.     474182  IN      A       192.203.230.10
F.ROOT-SERVERS.NET.     474182  IN      A       192.5.5.241
G.ROOT-SERVERS.NET.     474182  IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     474182  IN      A       128.63.2.53

;; Query time: 118 msec
;; SERVER: 147.250.1.1#53(147.250.1.1)
;; WHEN: Sun Jan  5 17:19:53 2003
;; MSG SIZE  rcvd: 436
```

On peut remarquer plusieurs choses :

- la réponse est au format des fichiers de configuration des serveurs de nom (qui sera étudié dans le paragraphe 5.1.4), ce qui permet éventuellement de réinjecter ces données sans autre traitement ;
- en conséquence, les lignes commençant par un point-virgule sont des commentaires (mais elles contiennent néanmoins des informations intéressantes pour le lecteur humain) ;
- les noms qualifiés sont tous terminés par le point de la racine.

Demandons maintenant au DNS l'adresse IP correspondant à la machine `ensta.ensta.fr` :

```
% dig a ensta.ensta.fr.

; <<>> DiG 9.1.3 <<>> a ensta.ensta.fr.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45952
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;ensta.ensta.fr.                        IN      A

;; ANSWER SECTION:
ensta.ensta.fr.         259200  IN      A       147.250.1.1

;; AUTHORITY SECTION:
ensta.fr.               259200  IN      NS      ensta.ensta.fr.
ensta.fr.               259200  IN      NS      nis1.ensta.fr.
ensta.fr.               259200  IN      NS      nis2.ensta.fr.

;; ADDITIONAL SECTION:
ensta.ensta.fr.         259200  IN      A       147.250.1.1
nis1.ensta.fr.          259200  IN      A       147.250.1.21
nis2.ensta.fr.          259200  IN      A       147.250.1.22

;; Query time: 63 msec
;; SERVER: 147.250.1.1#53(147.250.1.1)
;; WHEN: Sun Jan  5 17:17:31 2003
;; MSG SIZE  rcvd: 148
```

La réponse se trouve dans la partie `ANSWER SECTION`.

Si maintenant, nous cherchons le nom qualifié correspondant à l'adresse 147.250.1.1 :

```
% dig -x 147.250.1.1

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45845
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;1.1.250.147.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
1.1.250.147.in-addr.arpa. 259200 IN     PTR     ensta.ensta.fr.

;; AUTHORITY SECTION:
250.147.in-addr.arpa.   259200  IN      NS      ensta.ensta.fr.
250.147.in-addr.arpa.   259200  IN      NS      nis1.ensta.fr.
250.147.in-addr.arpa.   259200  IN      NS      nis2.ensta.fr.

;; ADDITIONAL SECTION:
ensta.ensta.fr.         259200  IN      A       147.250.1.1
nis1.ensta.fr.          259200  IN      A       147.250.1.21
nis2.ensta.fr.          259200  IN      A       147.250.1.22

;; Query time: 14 msec
;; SERVER: 147.250.1.1#53(147.250.1.1)
;; WHEN: Sun Jan  5 17:18:22 2003
;; MSG SIZE  rcvd: 170
```

5.1.4 BIND

Le logiciel utilisé sur quasiment tous les serveurs de noms du monde est *Berkeley Internet Name Domain* (BIND), développé par l'Internet Software Consortium (ISC).

BIND est livré en standard avec tous les systèmes UNIX mais il s'agit rarement de la dernière version. Or il est important de toujours utiliser la dernière version de BIND pour éviter l'exploitation de failles de sécurité connues et pour profiter des dernières fonctionnalités (BIND ayant énormément évolué ces dernières années).

La dernière version de BIND est la 9.7.0, sortie le 16 février 2010. De nombreux serveurs DNS utilisent encore BIND 8 ou BIND 4 (il n'y a pas eu de versions intermédiaires entre BIND 4 et 8) mais seule la version 9 est activement développée par l'ISC.

5.1.4.1 Récupération des sources

Pour installer BIND, l'idéal est de s'en procurer les sources directement chez l'Internet Software Consortium. Elle sont disponibles à l'URL :

`ftp://ftp.isc.org/isc/bind9/`

5.1.4.2 Compilation

Une fois l'archive décomprimée, la compilation s'effectue de manière classique :

```
% ./configure
creating cache ./config.cache
[...]
creating config.h
% make
```

5.1.4.3 Installation

Après avoir pris l'identité du super-utilisateur :

```
# make install
```

Ceci installe :
- des exécutables dans `/usr/local/bin` (dont `dig`) et `/usr/local/sbin` (dont `named`, qui est le serveur de noms).
- des fichiers d'en-tête dans `/usr/local/include` ;
- des bibliothèques dans `/usr/local/lib` ;

5.1.4.4 Configuration

BIND se configure au moyen d'un fichier principal et de plusieurs fichiers auxiliaires. Le fichier principal s'appelle par convention `named.conf` (bien qu'on puisse utiliser un autre nom, je vous recommande de garder celui-ci) et se trouve par défaut dans le répertoire `/etc` (on peut utiliser un autre répertoire et nous allons d'ailleurs le faire par la suite).

Par défaut, BIND est conçu pour fonctionner sous l'identité du super-utilisateur. Un logiciel de cette complexité n'est pas à l'abri d'erreurs de conception et d'anciennes versions comportaient des failles permettant de prendre le contrôle de la machine hébergeant le serveur de noms. Comme BIND peut tout aussi bien fonctionner sous l'identité d'un utilisateur non privilégié (bien que trop peu d'administrateurs prennent la peine de le faire), nous allons donc utiliser cette possibilité.

De même, BIND peut fonctionner dans un système de fichiers restreint par l'appel système `chroot()`. Nous utiliserons également cette option de sécurité.

Il faut maintenant choisir un répertoire où placer les différents fichiers de configuration. C'est également dans ce répertoire que sera confiné le démon `named`. Prenons par exemple `/opt/dns`.

Le fichier `named.conf`, situé dans le précédent répertoire, contiendra :

```
options
{
        directory "/" ;
} ;

# serveurs racine

zone "."
{
        type hint ;
        file "named.root" ;
} ;

# zones maîtres

zone "tp.ensta.fr"
{
        type master ;
        file "master/tp.ensta.fr" ;
} ;

zone "14.250.147.in-addr.arpa"
{
        type master ;
        file "master/14.250.147.in-addr.arpa" ;
} ;

# zones esclaves

zone "ensta.fr"
{
        type slave ;
        file "slave/ensta.fr" ;
        masters
        {
                147.250.1.1 ;
        } ;
} ;
```

Les commentaires sont introduits par un `#` initial (comme en Perl), un `//` initial ou entourés par `/*` et `*/` (comme en C++).

Le groupe `options` spécifie les options générales pour le serveur. Ici, on indique

le répertoire dans lequel les fichiers que l'on indiquera par la suite seront placés. Puisqu'on va mettre les fichiers dans `/opt/dns` et se restreindre à ce répertoire, c'est `/` qu'on indique dans la directive `directory`.

À son lancement, tout serveur DNS a besoin de connaître les serveurs de la racine. Il faut donc les lui fournir dans le fichier `named.root` (disponible à l'URL `ftp://ftp.internic.net/domain/named.root`). Ces données lui serviront uniquement à contacter l'un des serveurs de la racine pour y charger une zone à jour (d'où le type `hint` lié à cette zone).

Chaque zone pour laquelle le serveur est maître est du type `master` et est contenue dans un fichier placé dans le répertoire `master` (on aurait pu appeler ce répertoire tout autrement, voire ne pas mettre les fichiers dans un répertoire particulier, mais ceci permet de ranger les fichiers proprement et d'être cohérent avec la dénomination utilisée dans `named.conf`). Par souci de clarté, chaque fichier aura le même nom que la zone qu'il contient (là encore, ce n'est qu'une convention).

De même, chaque zone pour laquelle le serveur est esclave est du type `slave` et est contenue dans un fichier de même nom que la zone et placé dans le répertoire `slave`. On indique de plus l'adresse IP du serveur maître pour cette zone afin de l'y télécharger.

Les fichiers des zones pour lesquelles le serveur est esclave étant automatiquement remplis par transfert de zone, nous nous intéresserons par la suite aux fichiers des zones pour lesquelles le serveur est maître.

Étudions la zone maître `tp.ensta.fr` :

```
$TTL 1d

@       IN      SOA     ns.ensta.fr. hostmaster.ensta.fr. (
                        2001012800      ; serial
                        ; RFC 1537
                        8h              ; refresh
                        2h              ; retry
                        1w              ; expiry
                        1h )            ; negative caching

                IN      NS      ns.ensta.fr.

localhost       IN      A       127.0.0.1

guinness        IN      A       147.250.14.1
blanche         IN      A       147.250.14.2
```

Les commentaires sont introduits par un `;` et s'étendent jusqu'à la fin de la ligne.

Chaque fichier de zone commence par une directive `$TTL` qui indique la durée de vie (*time to live*) des informations de cette zone lorsqu'elles seront dans le cache d'un

autre serveur DNS. Pour notre zone, les informations resteront dans le cache des autres serveurs pendant une journée (`1d`). Par la suite, toute demande de résolution de nom sera honorée à partir du cache, jusqu'à ce qu'une journée se soit écoulée, auquel cas l'information en sera supprimée.

Le reste du fichier de zone contient des définitions de *resource records* (RR).

Chaque RR tient habituellement sur une ligne de la forme :

```
nom_du_paramètre        IN      RR      valeur(s)_du_paramètre
```

Ici, `IN` indique que le RR est de la classe Internet. Il existe d'autres classes mais, en pratique, elles ne sont pas utilisées.

Le premier RR du fichier est le SOA (*start of authority*), qui indique divers paramètres de la zone.

La première ligne indique :

- le nom de la zone contenue dans le fichier ou `@` pour utiliser le nom de zone tel qu'indiqué dans `named.conf` ;
- la classe Internet ;
- le type du RR (ici SOA) ;
- le nom du serveur maître pour cette zone ;
- l'adresse électronique de la personne responsable de cette zone.

Dans cette adresse, le `@` habituel est remplacé par un point (ce qui implique que la partie gauche de l'adresse ne peut pas en contenir). Le RFC 2142 recommande que cette adresse soit `hostmaster`.

Notez les points terminaux qui indiquent la racine du DNS (à partir de maintenant, faites attention à ne pas les oublier, sauf dans certains cas qui seront détaillés plus loin).

Suivent un ensemble de paramètres numériques, contenus entre parenthèses (ce qui permet de faire tenir le RR sur plusieurs lignes ; cette possibilité n'est d'ailleurs utilisée que pour le SOA) :

Le numéro de série qui est un entier sur 32 bits devant être augmenté à chaque modification de la zone. Par convention (et parce que ça tient dans un entier de 32 bits), on utilise un format AAAAMMJJNN contenant la date de la dernière modification de la zone au format ISO 8601 (AAAA pour l'année, MM pour le mois, JJ pour le jour) suivie d'un numéro d'ordre (NN valant `00` pour la première modification du jour, `01` pour la deuxième, etc.). Une erreur classique consiste à modifier les données de la zone sans augmenter le numéro de série.

La durée de rafraîchissement qui était utilisée par les esclaves pour savoir à quel intervalle ils devaient interroger le maître pour savoir si une zone avait été modifiée. Maintenant, le maître avertit immédiatement ses esclaves de toute modification donc ce paramètre n'est plus utilisé.

La durée de nouvel essai qui est utilisée par les esclaves pour savoir, après une tentative échouée de transfert de zone, au bout de combien de temps ils peuvent la retenter.

La durée d'expiration qui est utilisée par les esclaves pour savoir, s'ils ne peuvent pas mettre à jour leurs zones auprès du maître, au bout de combien de temps ils doivent effacer leurs données.

Le TTL négatif qui indique aux caches combien de temps ils doivent conserver les réponses négatives.

Le RFC 1537 explique le choix de ces valeurs (sauf pour la dernière, dont la signification a changé récemment).

On trouve ensuite les RR NS, qui indiquent quels sont les serveurs de nom pour la zone. Aucune différence n'est faite ici entre maître et esclaves mais l'habitude veut qu'on fasse figurer le maître en première position.

Notez que, dans ce RR, la première colonne est laissée vide. Dans ce cas, sa valeur est prise de la première colonne du RR précédent.

Viennent enfin les RR contenant les correspondances entre noms de machines et adresses IP. Ici, il s'agit d'une zone directe donc les RR sont de type A. Notez que l'on n'a pas ajouté le nom de la zone aux noms de machines et que l'on n'a pas mis de points terminaux. Dans ce cas, le nom de la zone est automatiquement rajouté.

Un tel RR :

```
guinness.tp.ensta.fr    IN      A       147.250.14.1
```

se verrait donc transformé en :

```
guinness.tp.ensta.fr.tp.ensta.fr.       IN      A       147.250.14.1
```

en raison de l'absence du point terminal. Il convient donc d'y être attentif.

Étudions maintenant la zone maître `14.250.147.in-addr.arpa`. Il s'agit d'une zone inverse (c'est-à-dire contenant les correspondances entre adresses IP et noms de machines) :

```
$TTL 1d

@       IN      SOA     ns.ensta.fr. hostmaster.ensta.fr. (
                        2001012800      ; serial
                        ; RFC 1537
                        8h              ; refresh
                        2h              ; retry
                        1w              ; expiry
                        1h )            ; negative caching

                IN      NS      ns.ensta.fr.

1       IN      PTR     guinness.tp.ensta.fr.
2       IN      PTR     blanche.tp.ensta.fr.
```

Tout d'abord, le nom de cette zone est particulièrement étrange. Le domaine `in-addr.arpa` n'est utilisé que pour les zones inverses. Et comme la hiérarchie des domaines se lit de droite à gauche, les adresses IP dans les zones inverses sont écrites dans ce sens, à l'envers du sens normal de lecture.

Pour le reste, ce fichier de zone est très semblable à un fichier de zone directe, à la différence qu'on utilise des RR de type PTR et qu'on n'écrit en première colonne que le dernier octet de l'adresse IP (la suite étant complétée par `14.250.147.in-addr.arpa` en l'absence du point final).

Une délégation se fait tout simplement en indiquant dans la zone mère des RR NS pour sa zone fille. Il faut également indiquer l'adresse IP des serveurs de noms (problème d'œuf et de poule). Ainsi, pour déléguer une zone `b1-4.tp.ensta.fr`, on peut rajouter au fichier de la zone `tp.ensta.fr` :

```
b1-4            IN      NS      ns.b1-4
ns.b1-4         IN      A       147.250.14.51
```

5.1.4.5 Lancement

Pour pouvoir utiliser les ports UDP et TCP 53, le serveur de noms a besoin d'être lancé sous l'identité du super-utilisateur.

Il est recommandé d'utiliser les options suivantes :

-t qui indique le répertoire auquel restreindre le serveur ;

-c qui indique le nom du fichier de configuration relativement à ce répertoire ;

-u qui indique le nom de l'utilisateur sous l'identité duquel tournera le serveur de noms après son démarrage.

Ainsi, le serveur de noms peut se lancer par la commande :

```
# named -t /opt/dns -c /named.conf -u dns
```

5.1.5 Quelques liens

- `http://www.isc.org/services/public/F-root-server.html` ou tout ce que vous avez toujours voulu savoir sur un serveur de la racine sans avoir jamais osé le demander.
- `http://www.internic.net/` L'InterNIC est l'organisme qui gère les zones `.com`, `.org` et autres.
- `http://www.nic.fr/` L'AFNIC est l'organisme qui gère la zone `.fr`.
- `http://www.eu.org/` EU.org, délégation gratuite de domaines.
- `http://www.isc.org/software/bind/` Le site officiel de BIND.
- `http://www.esil.univ-mrs.fr/~lafirme/named/` Installer un serveur DNS sécurisé.
- `http://www.acmebw.com/askmr.htm` Ask Mr. DNS.

5.1.6 Quelques bizarreries

Les noms de domaine très longs sont-ils utiles ?

```
http:
 //www.llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch.co.uk
http:
   //www.fautvraimentetreconpouravoiruneadresseinternetaussilongue.com/
http://3.
   14159265358979323846264338327950288419716939937510582097494459 2.com/
```

5.2 Lightweight Directory Access Protocol (LDAP)

LDAP est l'annuaire qui monte en ce moment. Il est principalement utilisé pour centraliser les bases d'authentification ou les carnets d'adresses électroniques mais il s'agit d'un annuaire généraliste pouvant être utilisé pour presque n'importe quoi.

LDAP est décrit dans le RFC 2251 et utilise le port TCP 389.

5.2.1 Principes

LDAP est l'adaptation dans le monde IP de DAP (*Directory Access Protocol*), qui permet d'accéder à des annuaires X.500 dans le monde ISO. C'est ce qui explique la structure particulières des bases LDAP et leur système de nommage.

À la différence des systèmes de gestion de bases de données (SGBD), les bases LDAP sont optimisées pour la lecture. En conséquence, elles peuvent aisément être répliquées pour assurer une meilleure disponibilité.

Une base LDAP est composée d'entrées. Chaque entrée regroupe un certain nombre d'attributs et a un DN (*Distinguished Name*) qui doit être unique au sein de l'ensemble de la base. Chacun des attributs a un type et une ou plusieurs valeurs.

Chaque entrée a un attribut spécial indiquant la *classe* de l'entrée. Celle-ci contrôle quels attributs sont autorisés dans l'entrée. Ainsi il existe une classe « organisation », une classe « personne », etc.

Les entrées sont définis dans un format de fichier spécial appelé LDIF (*LDAP Data Interchange Format*) et décrit dans le RFC 2849.

Quelques exemples permettront d'y voir plus clair.

```
dn: dc=ensta,dc=fr
objectclass: organization
o: ENSTA
postalAddress: 32, boulevard Victor
```

On définit ici un objet du type « organization » défini par un DN utilisant le domaine DNS de l'organisation au moyen d'éléments DC (*Domain Component*). On indique égalament le nom de l'organisation et son adresse postale.

La structure d'un annuaire LDAP est hiérarchique. Une fois l'organisation définie, on peut définir différents services en son sein :

```
dn: ou=sports,dc=ensta,dc=fr
objectclass: organizationalUnit
ou: sports
```

En dessous de l'entrée précédente, on définit donc le service « sports », dont on peut maintenant indiquer les membres :

```
dn: cn=Gerard Perbal,ou=sports,dc=ensta,dc=fr
objectclass: inetOrgPerson
cn: Gerard Perbal
sn: Perbal
uid: perbal
mail: perbal@ensta.fr
```

Les trois classes que nous venons de voir sont les plus fréquemment utilisées (surtout la dernière) dans les annuaires LDAP, bien qu'il en existe de nombreuses autres.

Les attributs possibles pour ces classes sont les suivants :

organization

- `o` (obligatoire)

- businessCategory
- description
- facsimileTelephoneNumber
- location (l)
- postalAddress
- seeAlso
- telephoneNumber

organizationalUnit

- ou (obligatoire)
- businessCategory
- description
- facsimileTelephoneNumber
- location (l)
- postalAddress
- seeAlso
- telephoneNumber

inetOrgPerson

- commonName (cn) (obligatoire)
- surname (sn) (obligatoire)
- businessCategory
- carLicense
- departmentNumber
- description
- employeeNumber
- facsimileTelephone
- Number
- givenName
- mail
- manager
- mobile
- organizationalUnit (ou)
- pager
- postalAddress
- roomNumber
- secretary
- seeAlso
- telephoneNumber
- title
- labeledURI
- uid

La hiérarchie précédente est basée sur le DNS, simplement parce que ceci assure l'unicité du nommage, mais on peut également définir une hiérarchie géographique :

```
dn: o=ENSTA,c=FR
objectclass: organization
o: ENSTA
postalAddress: 32, boulevard Victor
```

Dans cet exemple, c indique le code du pays et o le nom de l'organisme.

Ici se termine cette courte introduction à LDAP mais vous trouverez ici de quoi approfondir vos connaissances :

> http://www-sop.inria.fr/semir/personnel/Laurent.Mirtain/ldap-livre.html

5.2.2 OpenLDAP

Le serveur LDAP le plus répandu sous UNIX est OpenLDAP [1], dont la dernière version est la 2.4.21.

5.2.2.1 Récupération des sources

Les sources sont disponibles à l'URL :

> ftp://ftp.openldap.org/pub/OpenLDAP/openldap-release/

5.2.2.2 Compilation

Une fois l'archive décomprimée, la compilation s'effectue de manière classique :

```
% ./configure
creating cache ./config.cache
[...]
creating config.h
% make
```

5.2.2.3 Installation

Après avoir pris l'identité du super-utilisateur :

```
# make install
```

1. http://www.openldap.org/

5.2.2.4 Configuration

OpenLDAP est un logiciel bien trop complexe pour que sa configuration soit décrite en détail ici. Dans le cadre de ce cours, vous pouvez vous reporter à `http://www.openldap.org/doc/admin/quickstart.html`.

Pour la documentation complète d'OpenLDAP (un peu légère par endroits, d'ailleurs), vous pouvez vous référer à `http://www.openldap.org/doc/admin/`.

5.2.2.5 Tests

Outre le programme `ldapsearch`, fourni avec OpenLDAP mais d'un usage peu pratique, il est possible d'interroger une base LDAP avec le client graphique `http://biot.com/gq/` ou avec le carnet d'adresses du module de courrier électronique de Netscape.

5.2.2.6 Cas pratique

Vous trouverez à l'URL `http://www.hsc.fr/ressources/breves/openldap.html` une utilisation typique de LDAP consistant à centraliser une base d'authentification.

5.3 Autres annuaires

Network Information System (NIS), auparavant appelé Yellow Pages (YP) est un système créé par Sun Microsystems et permettant le partage de divers bases d'informations (notamment `/etc/passwd` et `/etc/group`) au sein d'un ensemble de machines sous UNIX. Il est aujourd'hui en voie de disparition au profit de LDAP.

6

La messagerie

LA MESSAGERIE est certainement l'un des services réseau les plus utilisés. Il existe de nombreux systèmes de messagerie propriétaires mais aujourd'hui la majorité des messageries utilise le protocole SMTP (*Simple Mail Transfer Protocol*). Quelques messageries d'entreprise utilisent également le protocole X.400 mais uniquement en interne, la communication vers l'extérieur se faisant par l'intermédiaire d'une passerelle SMTP.

SMTP est décrit dans le RFC 5321. Le format des courriers électroniques est décrit dans le RFC 5322. SMTP utilise le port TCP 25.

6.1 SMTP

Le fonctionnement de SMTP est en fait très simple. Étudions une session type :

```
% echo test | mail -v -s test babafou@babafou.eu.org
babafou@babafou.eu.org... Connecting to ensta.ensta.fr.
   via nullclient...
220 ensta.ensta.fr ESMTP The Internet gateway to ENSTA
   (8.9.1/8.9.1) ready at Sun, 11 Feb 2001 14:51:38 +0100 (CET)
>>> EHLO olive13.ensta.fr
250-ensta.ensta.fr Hello IDENT:root@olive13.ensta.fr [147.250.9.23],
   pleased to meet you
250-8BITMIME
250-SIZE
250-ONEX
250-ETRN
250-XUSR
250 HELP
>>> MAIL From:<baudoin@ensta.fr> SIZE=47
250 <baudoin@ensta.fr>... Sender ok
>>> RCPT To:<babafou@babafou.eu.org>
250 <babafou@babafou.eu.org>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 OAA21364 Message accepted for delivery
babafou@babafou.eu.org... Sent (OAA21364 Message accepted for delivery)
Closing connection to ensta.ensta.fr.
>>> QUIT
221 ensta.ensta.fr closing connection
```

L'option `-v` de la commande `mail` permet d'afficher le dialogue entre la machine expéditrice et le serveur de messagerie (attention, il s'agit du serveur de messagerie du site expéditeur, qui se chargera ensuite d'envoyer le message au serveur de messagerie du site destinataire). Les lignes préfixées par `>>>` sont envoyées du client au serveur de messagerie, celles qui commencent par un nombre sont envoyées du serveur au client, les quelques autres sont purement informatives.

À l'établissement de la connexion, le serveur est le premier à parler, affichant quelques informations sur le logiciel qu'il utilise :

```
220 ensta.ensta.fr ESMTP The Internet gateway to ENSTA
   (8.9.1/8.9.1) ready at Sun, 11 Feb 2001 14:51:38 +0100 (CET)
```

Le client se présente ensuite en indiquant son nom qualifé :

```
>>> EHLO olive13.ensta.fr
```

et le serveur lui rend la politesse en indiquant la liste des options qu'il connaît :

```
250-ensta.ensta.fr Hello IDENT:root@olive13.ensta.fr [147.250.9.23],
   pleased to meet you
250-8BITMIME
250-SIZE
250-ONEX
250-ETRN
250-XUSR
250 HELP
```

L'envoi du message proprement dit peut commencer, le client indique l'adresse électronique de l'expéditeur et la taille du message :

```
>>> MAIL From:<baudoin@ensta.fr> SIZE=47
```

Le message, tel que généré par `mail`, est le suivant, vous pouvez recompter, il contient bien 47 caractères :

```
To: babafou@babafou.eu.org
Subject: test

test
```

Le client indique ensuite l'adresse électronique du destinataire :

```
>>> RCPT To:<babafou@babafou.eu.org>
```

et envoie le message :

```
>>> DATA
```

Le message en question est celui indiqué plus haut, avec ses en-têtes et son corps. Le serveur de messagerie rajoutera d'ailleurs d'autres en-têtes pour aboutir au message qui partira effectivement.

Les adresses électroniques indiquées par le client dans son dialogue avec le serveur (par les commandes `MAIL From` et `RCPT To`) constituent ce qu'on appelle les adresses d'enveloppe du message. Elles correspondent généralement aux adresses indiquées en `From`, `To` et `Cc` de l'en-tête du message, sauf dans le cas d'adresses en `Bcc`, auxquel cas celles-ci n'apparaissent nulle part dans l'en-tête mais figurent bel et bien dans l'enveloppe.

6.2 Relation avec le DNS

Comment déterminer quel est le serveur de messagerie du site destinataire ? Le RFC 2219 indique bien qu'il est censé s'appeler `mail.domaine` mais c'est loin d'être le cas sur tous les sites. De plus, comment faire s'il y a plusieurs serveurs ?

Le DNS dispose donc d'un RR de type MX, qui indique quel est le serveur de messagerie associé à une zone :

```
ensta.fr.       IN      MX      10 ensta.ensta.fr.
```

Le serveur de messagerie pour la zone `ensta.fr` est donc la machine `ensta.ensta.fr`. Le nom de cette machine est précédé d'un entier naturel indiquant la priorité relative du serveur. Cet entier est appelé le *poids* du serveur. En effet, une zone peut disposer de plusieurs serveurs de messagerie et il faut pouvoir connaître la priorité de chaque serveur. Plus le poids est petit, plus le serveur est prioritaire. Ainsi, si le serveur de poids le plus faible ne répond pas, le courrier sera envoyé au deuxième, sinon au troisième et ainsi de suite. La valeur des poids n'a pas d'importance, seule en a leur position relative dans l'ordre croissant.

```
pasteur.fr.     IN      MX      0 mail.pasteur.fr.
                IN      MX      10 mail0.pasteur.fr.
```

Dans l'exemple précédent, le courrier sera tout d'abord envoyé au serveur `mail.pasteur.fr`. Si celui-ci ne répond pas, il sera alors envoyé au serveur `mail0.pasteur.fr`.

6.3 Serveurs de messagerie

Il existe de nombreux serveurs de messagerie. Sous UNIX, les plus répandus sont :

- sendmail [1]
- Postfix [2]

et, dans une moindre mesure :

- qmail [3]
- exim [4]

Sendmail, le logiciel de messagerie historique (sa première version remonte au début des années 1980), est encore utilisé sur de très nombreux serveurs de messagerie. Cependant, un système de configuration complexe (même s'il est maintenant masqué par des outils plus simples), une structure monolithique et une longue histoire de problèmes de sécurité (dont le dernier remonte pourtant à 1997) lui font aujourd'hui préférer d'autres logiciels, notamment Postfix.

1. http://www.sendmail.org/
2. http://www.postfix.org/
3. http://www.qmail.org/
4. http://www.exim.org/

6.4 sendmail

La messagerie électronique doit beaucoup à sendmail, qui équipait il y a quelques années la quasi-totalité des serveurs de l'Internet. Les administrateurs de messagerie doivent également beaucoup à sendmail, qui leur a procuré des nuits blanches en grand nombre. En effet, sendmail est réputé pour les nombreux problèmes de sécurité dont il a été affecté. Programme monolithique complexe fonctionnant sous l'identité du super-utilisateur, sendmail a d'ailleurs été l'un des vecteurs de propagation du ver de l'Internet en 1988. Il ne faut cependant pas être mauvaise langue car sendmail n'a pas eu de problème de sécurité connu depuis 1997. En revanche, il a conservé sa structure monolithique complexe et fonctionne toujours sous l'identité du super-utilisateur, ce qui en fait un danger potentiel.

De nombreux UNIX fournissent sendmail en standard mais il s'agit rarement de la dernière version. Bien que les soucis de sécurité semblent maintenant appartenir au passé, il est cependant prudent d'utiliser la version de sendmail la plus récente. Il s'agit actuellement de la version 8.14.3, sortie le 3 mai 2008.

6.4.1 Récupération des sources

Les sources de sendmail sont disponibles à l'URL :

```
ftp://ftp.sendmail.org/pub/sendmail/
```

6.4.2 Compilation

La compilation s'effectue dans le répertoire `sendmail` de la distribution source au moyen de la commande :

```
% sh Build
```

6.4.3 Installation

Après avoir pris l'identité du super-utilisateur :

```
# sh Build install
```

6.4.4 Configuration

Tout administrateur réseau vous dira que la configuration de sendmail est un cauchemar : c'est vrai. Mais seulement quand on ne sait pas comment s'y prendre parce que, finalement, si on ne regarde pas tout ce qu'il y a au-dessous, ce n'est pas bien méchant. Le but de ce cours n'étant pas de détailler la configuration de sendmail, nous allons survoler le célèbre fichier `sendmail.cf` et voir comment le générer plus simplement. Les curieux pourront se reporter à l'ouvrage *sendmail* [9] de la bibliographie.

6.4.4.1 Le fichier `sendmail.cf`

Le fichier de configuration de sendmail s'appelle `sendmail.cf` et se trouve dans le répertoire `/etc/mail` (les versions de sendmail antérieures à la 8.10 utilisaient le répertoire `/etc`).

Comme dans beaucoup de fichiers de configuration, les commentaires sont introduits par un dièse et s'étendent jusqu'à la fin de la ligne.

Par la suite, nous allons prendre le fichier `/etc/sendmail.cf` des machines de l'ENSTA comme exemple.

Le fichier `sendmail.cf` débute généralement par diverses définitions, comme par exemple :

```
DHensta.ensta.fr
```

Ceci définit (c'est ce qu'indique le `D`) une variable appelée `H` et dont la valeur est `ensta.ensta.fr`. Cette variable a une signification particulière pour sendmail, elle indique le nom du serveur de messagerie auquel envoyer tous les messages pour leur expédition.

Le fichier continue par la définition de certaines options :

```
O EightBitMode=pass8
```

Au bout d'un moment, on arrive aux *règles de réécriture*. Sendmail dispose en effet d'un petit langage de programmation lui permettant de modifier (dans ce cas, on dit *réécrire*) les adresses électroniques. L'intérêt des règles de réécriture est aujourd'hui assez limité mais elles étaient indispensables à l'époque où le courrier électronique devait circuler sur des réseaux totalement différents, pas seulement sur l'Internet. Les règles de réécriture servent maintenant principalement à aiguiller le courrier de manière statique (c'est-à-dire sans utiliser les RR MX du DNS), à normaliser les adresses (supprimer les noms de machines en partie gauche pour n'y laisser que le nom de domaine), à rejeter les messages provenant d'expéditeurs indésirables, etc.

Les règles de réécritures sont regroupées en six grands ensembles numérotés de `0` à `5`. En effet, chaque ensemble de règles de réécritures possède un numéro (on peut également lui attribuer un nom dans les versions récentes de sendmail). L'ensemble 3 est introduit ainsi :

```
Scanonify=3
```

Le `S` en début de ligne sert à introduire un ensemble de règles de réécriture. Ici, l'ensemble s'appelle `canonify` et porte le numéro 3. Si l'on n'avait pas précisé `=3`, comme ceci :

```
Scanonify
```

sendmail aurait choisi un numéro d'ensemble non encore attribué.

On aurait également pu indiquer directement :

```
S3
```

Viennent ensuite les règles proprement dites, une par ligne, introduites par `R` :

```
R$-@$+  $1 $2   ceci est un commentaire
```

En regardant bien, on distingue trois colonnes, séparées par des tabulations.

```
R$-@$+      →$1 $2      →ceci est un commentaire
```

La première colonne est comparée à la chaîne de caractères passée en entrée de la règle. Si elle correspond, on la remplace par la deuxième colonne et on passe à la règle suivante, sinon on passe directement à la règle suivante.

La chaîne de caractère passée à la règle est tout d'abord découpée en morceaux, aux endroits des `@` et des points, qui ont une signification particulière dans les adresses électroniques. Puis celle-ci est comparée à la première colonne. Dans la comparaison, `$-` correspond à exactement un élément, `$+` à un ou plusieurs éléments et `$*` à zéro ou plusieurs éléments.

Ainsi, l'adresse `babafou@ensta.fr` correspond à `$-@$+` (`babafou` correspond à `$-`, le `@` se correspond à lui-même et `ensta.fr` correspond à `$+`). En revanche, `babafou@ensta.fr` ne correspond pas à `$-@$-` puisqu'`ensta.fr` contient deux éléments.

Si la chaîne de caractère correspond donc à la première colonne, cette chaîne est remplacée par la deuxième colonne. Dans celle-ci, les `$1`, `$2` et ainsi de suite servent à mémoriser les éléments de la chaîne initiale qui ont correspondu à la première colonne. Ainsi, dans notre exemple, `$1` contiendrait `babafou` et `$2` contiendrait `ensta.fr`.

La troisième colonne est un commentaire.

On peut également faire appel à un autre ensemble de règles de réécriture, un peu comme on appellerait un sous-programme :

```
R$-@$+  $>51 $1 $2      ceci est toujours un commentaire
```

Ici, en cas de correspondance réussie, on appelle l'ensemble 51 en lui passant les deux parties de l'adresse électroniques séparées par un espace.

Nous n'irons pas plus loin dans l'étude des régles de réécriture, ceci représentant déjà bien trop de maux de tête pour aujourd'hui.

6.4.4.2 Configuration de sendmail avec `m4`

Comme vous avez pu l'entrevoir, les règles de réécriture sont réservées à une certaine élite capable de comprendre le sendmail dans le texte. Presque plus personne ne gère son serveur de messagerie en allant farfouiller directement dans le `sendmail.cf`. Comme pour tous les fichiers de configuration, on préfère le faire générer automatiquement à partir d'un fichier plus simple.

Pour cela, sendmail est fourni d'origine avec les bons outils, autant les utiliser.

Le répertoire `cf` de la distribution contient dans ses sous-répertoires divers fichiers permettant de construire un `sendmail.cf` au moyen d'un fichier maître grâce au préprocesseur `m4`. Divers exemples de fichiers maîtres se trouvent dans le sous-répertoire `cf` (donc `cf/cf` par rapport au sommet de la distribution).

Considérons le fichier `sendmail.mc` suivant :

```
include('../m4/cf.m4')

OSTYPE(linux)dnl

define('confSMTP_MAILER', 'smtp8')dnl
define('confPRIVACY_FLAGS', 'authwarnings,restrictqrun,restrictmailq,noetrn,noexpn')dnl

FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
FEATURE(always_add_domain)dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_entire_domain)dnl
FEATURE(relay_entire_domain)dnl
define('PROCMAIL_MAILER_PATH', '/usr/local/bin/procmail')dnl
FEATURE(local_procmail)dnl

MASQUERADE_AS(ensta.fr)dnl
MASQUERADE_DOMAIN(ensta.fr)dnl
FEATURE(masquerade_envelope)dnl

MAILER(local)dnl
MAILER(smtp)dnl
```

Il s'agit d'un fichier permettant de générer un `sendmail.cf` fonctionnel pour une configuration classique de serveur de messagerie. La génération se fait en utilisant `m4` :

```
m4 sendmail.mc > sendmail.cf
```

Il serait difficile de détailler ici le contenu de ce fichier, vous en trouverez une description à jour dans le fichier `cf/README`.

6.4.5 Lancement

Sendmail dispose de trop d'options pour toutes les citer ici. Il se lance généralement par la séquence suivante :

```
sendmail -bd -q30m
```

A

Revision Control System (RCS)

RCS désigne un ensemble d'outils permettant de conserver l'historique des modifications apportées à un fichier. En ce sens, il est fort utile afin de suivre les évolutions d'un fichier de configuration et de revenir, le cas échéant à une version antérieure. RCS permet également de gérer l'accès concurrent à un fichier, dans le cas d'un travail en équipe.

RCS est fourni avec la plupart des systèmes UNIX mais il est possible au besoin d'en télécharger les sources à l'URL `ftp://ftp.gnu.org/pub/gnu/rcs/`. La version courante est la 5.7.

A.1 Principe de fonctionnement

RCS stocke la dernière version d'un fichier, ainsi que chacune des modifications permettant de générer les versions précédentes dans un fichier appelé *fichier RCS*. Le nom de ce fichier est construit en ajoutant le suffixe `,v` (oui, oui, avec une virgule, pas un point) au nom du fichier à gérer. Nous utiliserons comme exemple le fichier `toto`, dont le fichier RCS correspondant s'appelle donc `toto,v`.

Par défaut, le fichier RCS est stocké dans le même répertoire que le fichier principal. Ceci peut rapidement multiplier le nombre de fichiers dans ce répertoire et y noyer les fichiers importants. C'est pourquoi RCS stocke plutôt ses fichiers dans un sous-répertoire `RCS` lorsqu'il existe. Il est donc recommandé d'utiliser cette possibilité.

A.2 Premier enregistrement d'un fichier

Afin de pouvoir gérer un fichier avec RCS, il faut tout d'abord créer le fichier RCS (ce qui suppose que le fichier principal existe déjà). Ceci se fait grâce à la commande `ci` (*check in*) :

```
% ci -u toto
toto,v  <--  toto
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> exemple RCS
>> .
initial revision: 1.1
done
```

La création du fichier RCS s'accompagne d'un commentaire (qui, dans un cas réel, sera certainement plus intelligent que celui de l'exemple).

L'option `-u` indique à `ci` qu'il doit conserver le fichier principal (qui aurait été effacé sans cela). En revanche, tout droit d'accès en écriture lui est supprimé (il faudra passer par une autre commande pour pouvoir modifier le fichier).

A.3 Modification d'un fichier

Avant de pouvoir modifier un fichier, il faut utiliser la commande `co` (*check out*) afin de lui rendre les droits en écriture et d'empêcher toute modification simultanée par quelqu'un d'autre (ce qu'indique l'option `-l`) :

```
% co -l toto
toto,v  -->  toto
revision 1.1 (locked)
done
```

On peut alors modifier le fichier à loisir. Toute autre tentative d'utiliser `co` se soldera par un échec, ce qui évite que plusieurs personnes modifient le même fichier au même moment.

Une fois les modifications terminées, on peut les enregistrer dans le fichier RCS et libérer l'accès au fichier :

```
% ci -u toto
toto,v  <--  toto
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' or end of file:
>> description succinte de la modification
>> .
done
```

A.4 Travail en groupe

Comme la commande `co` permet d'éviter l'accès simultané de plusieurs personnes au même fichier, RCS se révèle fort utile lorsqu'on travaille en équipe sur un ensemble de fichiers. À cet effet, il faut que chacun ait accès en écriture au répertoire `RCS` contenant les fichiers RCS. Pour cela, il suffit que les personnes en question appartiennent au même groupe UNIX, de même que le répertoire `RCS`.

A.5 Autres commandes RCS

La commande `rlog` permet d'afficher les commentaires associés à toutes les versions d'un fichier :

```
% rlog ethers
RCS file: RCS/ethers,v
Working file: ethers
head: 1.1240
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 1240;  selected revisions: 1240
description:
Fichier de correspondance adresse Ethernet <-> nom de machine
----------------------------
revision 1.1240
date: 2002/01/31 16:51:23;  author: masson;  state: Exp;  lines: +1 -0
> 00:c0:4f:b4:d3:06     cardamone.pmo.pasteur.fr
----------------------------
revision 1.1239
date: 2002/01/31 15:58:36;  author: masson;  state: Exp;  lines: +1 -1
< 00:02:55:6b:60:44     odile.sig.pasteur.fr
> 00:02:55:2b:60:44     odile.sig.pasteur.fr
----------------------------
[...]
```

La commande `rcsdiff` permet d'afficher les différences entre deux versions d'un fichier (au même format que la commande `diff`) :

```
% rcsdiff -r1.1238 -r1.1240 ethers
===================================================================
RCS file: RCS/ethers,v
retrieving revision 1.1238
retrieving revision 1.1240
diff -r1.1238 -r1.1240
2692a2693
> 00:c0:4f:b4:d3:06     cardamone.pmo.pasteur.fr
2923c2924
< 00:02:55:6b:60:44     odile.sig.pasteur.fr
---
> 00:02:55:2b:60:44     odile.sig.pasteur.fr
```

Bibliographie

Réseaux informatiques

[1] Olivier Bonaventure.
Computer Networking. Principles, Protocols and Practice.
Presses universitaires de Louvain, 2010.
URL : http://inl.info.ucl.ac.be/cnp3.

Simple Network Management Protocol (SNMP)

[2] Douglas Mauro et Kevin Schmidt.
Essential SNMP.
2e édition.
O'Reilly Media, 2005.
URL : http://shop.oreilly.com/product/9780596008406.do.

[3] Mark E. Miller.
Managing Internetworks with SNMP.
3e édition.
IDG Books Worldwide, 1999.

Les annuaires

[4] Gerald Carter.
LDAP System Administration.
O'Reilly Media, 2003.
URL : http://shop.oreilly.com/product/9781565924918.do.

[5] Ralph Droms et Ted Lemon.
The DHCP Handbook.
Macmillan Technical Publishing, 1999.
URL : http://www.dhcp-handbook.com/.

[6] Timothy A. Howes, Mark C. Smith et Gordon S. Good.
Understanding and Deploying LDAP Directory Services.
2e édition.
Addison-Wesley, 2003.
URL : http://www.informit.com/store/product.aspx?isbn=9780672323164.

[7] Cricket Liu et Paul Albitz.
DNS and BIND.
5e édition.
O'Reilly Media, 2006.
URL : http://shop.oreilly.com/product/9780596100575.do.

[8] Jan-Piet Mens.
Alternative DNS Servers. Choice and deployment, and optional SQL/LDAP back-ends.
UIT Cambridge Ltd, 2009.
URL : http://www.uit.co.uk/BK-ADNS/HomePage.
Ce livre est téléchargeable au format PDF : http://mens.de/:/altdnsbook.

La messagerie

[9] Bryan Costales, George Jansen, Claus Aßmann et Gregory Neil Shapiro.
sendmail.
4e édition.
O'Reilly Media, 2007.
URL : http://shop.oreilly.com/product/9780596510299.do.

[10] Kyle D. Dent.
Postfix : The Definitive Guide.
O'Reilly Media, 2003.
URL : http://shop.oreilly.com/product/9780596002121.do.

[11] Philip Hazel.
Exim : The Mail Transfer Agent.
O'Reilly Media, 2001.
URL : http://shop.oreilly.com/product/9780596000981.do.

[12] John Levine.
qmail.
O'Reilly Media, 2004.
URL : http://shop.oreilly.com/product/9781565926288.do.

Revision Control System (RCS)

[13] Don Bolinger et Tan Bronson.
Applying RCS and SCCS.
O'Reilly Media, 1995.
URL : http://shop.oreilly.com/product/9781565921177.do.

Table des matières