RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'Etat en Electronique

THÈME

TECHNOLOGIE RESEAU

APPLICATION A L'ETUDE ET A LA MISE EN ŒUVRE D'UN RESEAU INTRANET SOUS LINUX

Proposé et dirigé par :

M^r^ R.SADOUN

Etudié par :

M^r^ B. HAFFAR

M^r^ S. BAKELLI

Promotion 2002

E.N.P 10 Avenue Hassen Badi EL HARRACH - ALGER

الملخص

الهدف من هذا العمل هو دراسة مبادئ و أسس شبكات المعلوماتية على العموم و على الشبكات المحلية – أنترنات – على الخصوص ,و فهم مبادىء و أسس القواعد المستعملة خاصة.

التطبيق تم با استعمال برامج تدعى (البرامج المفتوحة), مثل نظام التشغيل لينيكس ,وفي الخدمات مثل أباش و ساندمايل على سبيل المثال.

كلمات المفتاح

ادارة النظام –لينيكس –شبكات المعلوماتية –الخدمات –أنترانات.

Résumé

L'objectif de ce travail est d'étudier le fonctionnement des réseaux informatiques dans le cas général et de l'Intranet en particulier; la compréhension des mécanismes fonctionnels des différents protocoles mis en jeux était un préalable essentiel.

L'implémentation est faite à l'aide de logiciels dits "open source". On cite le système d'exploitation Linux et comme services Apache et Sendmail par exemple.

Mots Clef

Administration Système -Linux -Réseau Informatique -Services -Intranet.

Abstract

The objective of this work is to study the operation of the data-processing networks in the general case and of the Intranet in particular; the comprehension of the functional mechanisms of the various protocols put in plays was an essential precondition .

The implementation is made using softwares known as " open source ". The system is quoted of Linux exploitation and like services Apache and Sendmail for example .

Words Key

Administration Système - Linux - Data processing network - Services -Intranet.

Dédicace

Je dédie ce modeste travail :

- A mes chers parents et grands parents.
- A ma grande famille et tout mes proche .
- A tous mes amis et collègues d'étude.

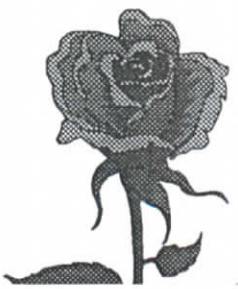
➢ **Brahim**

Dédicace

Je dédie ce modeste travail :

- A mes chers parents et grands parents.
- A mes frères et sœurs.
- A ma grande famille et tout mes proche .
- A tous mes amis et collègues d'étude.

➢ **Salah**



REMERCIEMENTS

Nous tenons tout D'abord à remercier notre Dieu ,puis nos parents pour leur soutien moral et financier ,sans oublier la grande famille.

Nous adressons nos sincères remerciements à notre promoteur Mr R.Sadoun d'abord pour sa proposition du sujet et pour son orientation ;critiques et suggestion dont ils nous ont fait ,pour la documentation et les moyens qu'ils ont mis à notre disposition .

Nos remerciements vont égale à tout les enseignants qui ont contribué à notre formation ,en particulier Mme L.Hammami et Mr A.Belouachrani ,aux personnels de la bibliothèque centrale de l'ENP ;du CCU ;du CERIST en particulier Mr Y.Bakelli et Mr M.Zidelmal.

En fin ,on remercie nos frère ,nos amis ,nos collèges ,et tout personne ayant de loin ou de près contribué à réaliser ce mémoire.

S.BAKELLI
B.HAFFAR

SOMMAIRE

CHAPITRE III — Développement d'un Intranet

Introduction

Notre époque est marquée par l'essor sans cesse croissant de la production, de la diffusion et de la consommation de l'information avec un volume, une densité et un débit qui restent parfois effrayant. Internet représente le vecteur clef associé à cette triptyque.
L'origine de ce concept est liée à celle de l'évolution naturelle de l'ordinateur dont l'objectif implicite consiste en la recherche de plus en plus de puissance de traitement et de plus en plus de fiabilité. Ce qui implique une recherche de redondance, donc de multiplicité. et par, là le développement des outils de leurs intercommunications.

Historiquement, c'est en 1962, alors que le communisme faisait force, que le département de la défense américaine crée l'ARPA dont la mission consiste à créer un réseau de communication capable de résister à une attaque nucléaire. Le concept de ce réseau était posé : permettre aux ordinateurs de la défense américaine de communiquer même en cas de destruction d'une quelconque partie des constituants en s'appuyant sur un réseau d'ordinateurs décentralisé. Il s'agissait donc d'un réseau purement militaire censé être "indestructible". Paul Baran eût l'idée de créer un réseau sous forme d'une grande toile. Il avait réalisé qu'un système centralisé était vulnérable car la destruction du noyau provoquait l'anéantissement des communications. Il mit donc au point un réseau hybride d'architectures étoilées et maillées dans lequel les données se déplaceraient de façon dynamique, en "cherchant" le chemin le moins encombré, et en "patientant" si toutes les routes étaient encombrées. Cependant, malgré ces concepts répertoriés sur onze volumes, le Pentagone refusa à cette époque de valider ce projet. Baptisé ARPANET, il fut repris sept années plus tard afin de relier quatre instituts universitaires américains.

En 1975 le réseau ARPANET était quasiment au point. Le gouvernement américain décida d'en prendre le contrôle en le confiant à une organisation: la United States Defense Communications Agency, renommée par la suite DISA ("Defense Information Systems Agency" traduisez "Agence chargée des sytèmes d'Informations à la Défense").

En parallèle, le développement de concepts de services associés au réseau pris naissance. C'est ainsi qu'en 1972 Ray Tomlinson mit au point un nouveau mode de communication. Appelé courrier électronique, il permettait l'échange d'informations au sein du réseau. Ainsi il était possible de contacter un grand nombre de personnes grâce à un seul message électronique (e-mail). C'est ce même Ray Tomlinson qui mit au point le protocole TCP, permettant d'acheminer des données sur un réseau en les fragmentant en petits paquets.

Tim Berners-Lee, un chercheur britannique du CERN (Centre Européen de la Recherche Scientifique), écrit un document à destination du CERN visant à améliorer le management du système d'information. Ce document, intitulé "Information Management: A Proposal", servira de base un an plus tard à l'établissement du web tel que nous le connaissons aujourd'hui.

Si le Web est connu de tout le monde depuis seulement quelques années, il faut faire remonter son origine à la fin de la deuxième guerre mondiale.

C'est en 1945 qu'apparaît l'idée de lien hypertexte dans les microfilms. Des scientifiques américains mirent au point un boîtier, subtile mélange d'optique, d'électronique et de mécanique, baptisé Memex, capable de créer des liens entre plusieurs microfilms.

Pendant 15 ans environ, on en reste à peu près là en matière d'hypertexte. C'est dans les années 60 que Doug Engelbart mit au point un prototype fabuleux, le "oNLine System" (NLS), capable de parcourir et créer des pages hypertexte, d'écrire des mails, etc.

En 1965, Ted Nelson invente enfin le mot hypertexte. Le tournant de l'hypertexte et du web est finalement pris en 1980, où Tim Berners-Lee, écrit un programme, nommé "Enquire-Within-Upon-Everything", qui permet de créer des liens entre plusieurs documents. Chaque document est en fait muni d'un titre, d'un type et d'une liste de lien. Fin 1992, 26 serveurs web tournent dans le monde.

Cette idée de communication « extra-muros » visant, rappelons le, l'accroissement de puissance et de la fiabilité par le concept de redondance c'est transformé en un fabuleux outil de diffusion et de gestion de l'information à moindre coût. D'où naquis l'idée, somme toute naturelle, d'appliquer ces concepts à des structures beaucoup plus petites, pouvant être fermées, reprenant les idées clefs de l'Internet : réseau et service. Ce nouveau concept, appelé Intranet, représente l'outil optimum de partage et d'échange d'informations dans des réseaux propriétaires.

L'objet de notre mémoire s'inscrit dans ce contexte. Il consiste en la conception et la mise en œuvre d'un réseau Intranet. Pour arriver à cette fin, nous avons structuré notre approche en trois étapes. La première introduisait les principes généraux des réseaux pour nous permettre la compréhension du modèle et des mécanismes fonctionnels qui y sont liés. La seconde nous a introduit de plein pied dans les concepts plus spécialisés d'Internet. On cite le modèle du réseau, l'étude des protocoles associés et celles de ses services. La dernière étape a été consacrée à l'implémentation.

Introduction

Le développement des techniques de télécommunication et leurs mises en œuvre aidant ; cette simple notion de communication s'est étendue à la notion de partage de ressources qui va d'un simple périphérique passif (sortie, stockage ou autre) jusqu'aux processeurs en passant par la mémoire et les informations qui y sont gérées. Les systèmes coopératifs sont alors nés. La notion de sécurité prend alors une dimension des plus importantes.

Pour arriver à ce degrés de développement (Internet représente une très bonne illustration), une technologie spécifique s'est illustrée et son développement a été des plus spectaculaires. Elle s'appuie aussi bien sur les technologies des télécommunications qu'elle est d 'ailleurs en train de transformer (ne parle-t-on pas de téléphonie IP !) que sur celles du traitement de l'information en passant par celle de l'électricité d'une manière générale et de l'électronique en particulier.

Nous nous intéresserons dans notre cas au cas particulier des réseaux locaux et les technologies qui leur sont spécifiques quoique les développements extrêmement rapides auxquels nous assistons rendraient rapidement obsolètes.

I- Présentation des réseaux locaux

L'époque où les réseaux se composaient d'une machine isolée complétée d'un ensemble de terminaux connectés par des liaisons séries est maintenant bien révolue. Les dinosaures qui se raccrochent encore aujourd'hui à cette façon de voir les choses seront démentis par l'évolution. Aujourd'hui, les réseaux d'ordinateurs forment une gigantesque toile d'araignée qui couvre toute la planète.

Les avantages procurés par le réseau sont indéniables. En voici une liste non exhaustive:

- Partage de ressources: par exemple, plus la peine d'avoir une imprimante pour chaque poste, une seule suffit
- Augmentation de la disponibilité et de la fiabilité : plus de gros serveur unique, les serveurs ont des doubles qui peuvent prendre le relais instantanément en cas de problème
- Economie : 10 postes de travail coûtent beaucoup moins cher qu'un seul gros serveur et sont tout autant efficaces si ce n'est plus.
- Communication : par définition, un réseau relie des postes de travail et permet donc l'échange d'informations comme la messagerie et le travail à distance.

Les premiers ordinateurs ont fait leur apparition dans les 40. Son principe n'avait pas trop évolué, c'était des "gros" ordinateurs appelés mainframe et qui étaient en fait surtout gros par leur taille. L'interactivité n'était pas d'actualité. Les programmes étaient saisies sur des cartes perforées à insérer dans un lecteur ; l'ordinateur les lisant comme un orgue de barbarie!. Chaque carte correspondait grosso modo à une instruction et les applications s'exécutaient en général selon le mode batch (encore appelé le traitement par lot). Il n'y avait passage à l'application suivante que lorsque l'application en cours était achevée. Vers 1970 sont apparus les mini-ordinateurs possédant leurs propres ressources (unité centrale, unité de stockage, imprimante), exécutant des applications spécifiques. Comme on avait moins recours à un gros ordinateur central, on parlait alors de traitement décentralisé. Dans le milieu des années 70 est né le besoin de partager, d'abord les imprimantes, puis les données et les unités de stockage et pourquoi pas les programmes ? Le besoin de communiquer entre ordinateurs est alors apparu et l'on a commencé à voir les premiers réseaux d'ordinateurs apparaître.

Pour aborder leur fonctionnement,il faut d'abord étudier le modèle en couches O.S.I qui est en fait sur lui repose la modélisation de tous les mécanismes de communication dans le monde des résaux informatiques. Autrement dit, le modèle O.S.I est une abstraction de protocoles et de services échangés dan un réseau.

II- Protocoles et services

Services et protocoles sont deux concepts différents mais néanmoins complémentaires.

II.1- Services

Le service [1]peut être défini par un ensemble de primitives (opérations) disponibles à des clients donnés. Ces primitives sélectionnent les services à réaliser comme rendre compte par exemple d'une action prise par l'entité paire.

Dans le modèle OSI (qu'on va décrire par la suite), chaque couche fournit des services à la couche qui est au dessus. Les *entités* sont les éléments actifs de chaque couche. Ils se présentent sous forme d'entités logicielle (tel un processus) ou matérielle (puce d'E/S). Les entités de la même couche sur différentes machines sont appelées *entités paires*.
Les couches N et N+1 sont respectivement fournisseur et utilisateur de service. La couche N peut utiliser le service de la couche N-1 pour fournir son service.
Les services sont accessible par des points d'accès aux services appelés SAP : Service Access Point. Chaque SAP est identifié par une adresse unique.
Les services peuvent être fournit selon deux modes : mode avec connexion et mode sans connexion. Dans le mode avec connexion, l'utilisateur établit au préalable une connexion, l'utilise puis la relâche ; l'aspect essentiel de la connexion c'est qu'elle fonctionne comme un tube : l'émetteur pousse des objets d'un coté et le récepteur les récupère dans le même ordre de l'autre coté.
En revanche, dans le mode sans connexion, chaque message porte l'adresse de destination complète et est acheminé à travers le système indépendamment de tout les autres. Normalement, quand deux messages sont envoyés vers la même destination, le premier envoyé est le premier reçu. Cependant, il peut arriver que le premier soit retardé et que le second arrive d'abord. Dans le cas du mode connecté, ce cas est impossible. Chaque service peut être caractérisé par ses qualités. Des services sont fiables s'ils ne perdent jamais de données. La mise en œuvre d'un service fiable se fait par acquittement de chaque message par le récepteur de sorte que l'émetteur soit sûr de son arrivée. Le processus d'acquittement engendre charge et délai supplémentaires souvent bénéfiques mais parfois indésirables .

II.2- Protocoles

Pour communiquer sur un réseau, les machines utilisent un ensemble de règles et de conventions appelées protocoles [1].
Partant du principe de modularité et compte tenu de leur complexité, les protocoles ont été structurés en couches dans le but de faciliter et de contrôler leur implémentation.
L'un des avantages de cette structuration est d'isoler les différents protocoles (correspondants aux différentes couches) pour que tout changement introduit sur l'un deux n'affecte pas les fonctionnalités des autres. Ce modèle offre des interfaces entre les différentes couches afin de permettre aux protocoles d'une couche donnée d'interagir avec ceux des couches qui lui sont directement adjacentes : chaque couche s'appuie sur des services offerts par une couche inférieure et vise à offrir des services à la couche qui lui est supérieure. Les entités utilisent des protocoles afin de mettre en œuvre leurs spécifications de

service .Il est possible de changer de protocole sans pour autant changer la visibilité du service pour l'utilisateur. Ainsi service et protocole sont totalement découplés .

III- Architecture en couches - modèle OSI

Pour assurer et faciliter l'interconnexion de différents types de réseau, il faut réunir les supports physiques. Mais pour pouvoir assurer un bon transfert de l'information avec une qualité suffisante, il faut prévoir une architecture logicielle adéquate.
En partant de ce principe, l'organisation international de normalisation ***ISO*** a formulé un modèle qui fixe une architecture fonctionnelle de la communication par réseau : le modèle ***OSI***[1][2] (*Open System Interconnexion*).(figure 1)
Dans ce modèle, l'ensemble des opérations effectuées par une machine pour communiquer est découpé en sept modules fonctionnels en couches. Les principes ayant conduits à ces 7 couches sont les suivants:
- Une couche doit être créée lorsqu'un nouveau niveau d'abstraction est nécessaire.
- Chaque couche exerce une fonction bien définie.
- Les fonctions de chaque couche doivent être choisies en pensant à la définition de protocoles normalisés internationaux.
- Le choix des frontières entre couches doit minimiser le flux d'informations aux interfaces.
- Le nombres de couches doit être suffisamment grand pour éviter la cohabitation dans une couche de fonctions très différentes et suffisamment petit pour éviter que l'architecture ne devienne difficile à maîtriser. Donc comme le montre la définition d'une couche, chacune à une fonction bien particulière.

Ainsi a été défini 7 différentes fonctions que doit avoir un système , par là les sept différentes couches qui composent le modèle OSI.

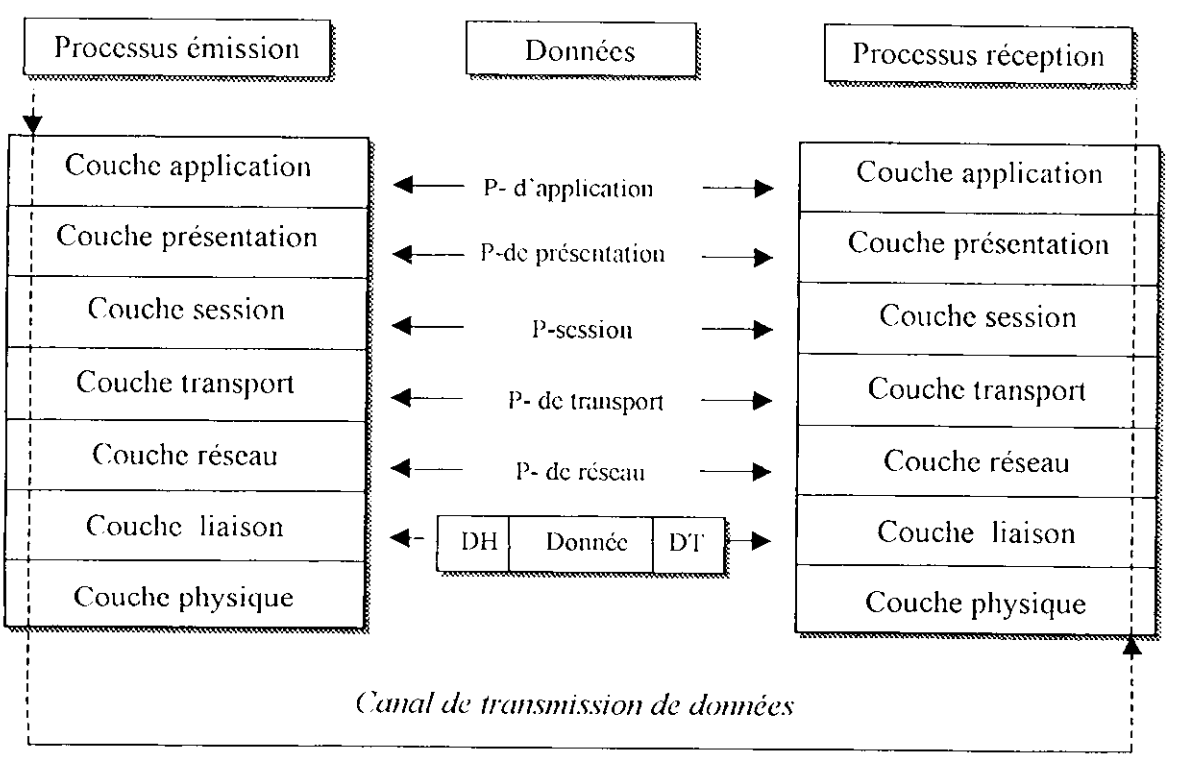


Fig.1 Modèle en couche OSI

III.1- La couche physique

Elle a pour principale fonction d'assurer une transmission transparente de l'information dans le réseau sous la forme de trains d'éléments binaires ou bits [1].
Pour cela, la couche physique devra entre autre définir :

- Les modes de représentation des 0 et des 1 de manière à ce qu'aucune confusion ne soit possible (niveaux de tension, durée d'un bit codage, transcodage des signaux, etc.)
- Les types de connecteurs utilisés aux extrémités d'une liaison (nombre de broches, fonction de chaque broche, etc.)
- La manière dont les liaisons sont établies au début d'une communication, et déconnectées ensuite.
- Le mode de transmission utilisé (unidirectionnel, bidirectionnel, bidirectionnel à l'alternat)
- Le mode de connexion employé (point à point ou multipoint)
- Le mode de traitement des erreurs (détection, correction)

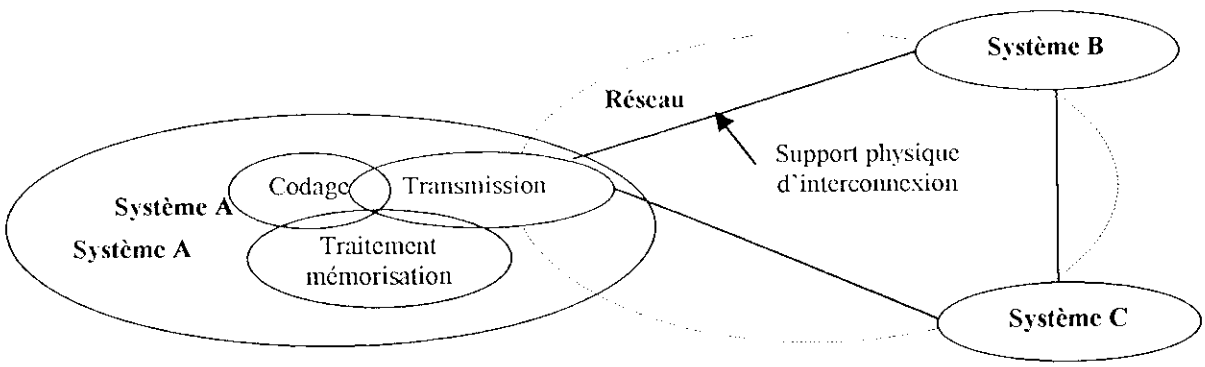


Fig.2 Fonction de base des systèmes téléinformatiques

Comme le montre la figure ci dessus, des concepts fondamentaux ont été mis en œuvre : l'information et les canaux qui la véhiculent avec tous les systèmes de codage et de correction qui sont mis en œuvre pour aussi bien augmenter le débit que pour limiter les perturbations qui peuvent nuire à la qualité de la transmission.

III.1.1- Notion d'information

Le type et le volume des informations, ainsi que le nombre d'utilisateurs simultanés sont des données importantes pour la concrétisation du réseau et selon ces critéres, définir le débit minimum nécesessaire et donc le type de support, la disposition des équipements et la configuration du site d'implatation du réseau conduisant au choix de la topologie [2].

On distingue plusieurs types d'informations pouvant circuler sur un réseau :

- Les informations tel le transfert de fichiers, images fixes, traitement de texte, messagerie, etc.
- Les informations de type multimédia.

III.1.2- Notion de canal

Une ligne de transmission est une liaison entre les deux machines. On désigne généralement par le terme **émetteur** la machine qui envoie les données et par **récepteur** celle qui les reçoit. Les machines peuvent parfois être chacune à son tour réceptrice ou émettrice (c'est le cas généralement des ordinateurs reliés par réseau) [1].

La ligne de transmission, appelé aussi parfois *canal de transmission* ou *voie de transmission*, n'est pas forcément constituée d'un seul support physique de transmission, c'est pourquoi les machines d'extrémités (par opposition aux machines intermédiaires), appelées **ETTD** [2] (*équipement terminal de traitement de données*, ou en anglais *DTE, Data Terminal* Equipment) possèdent chacune un équipement relatif au support physique auxquelles elles sont reliées, appelé **ETCD** [2] (équipement terminal de circuit de données, ou en anglais DCE, Data Communication Equipment). On nomme **circuit de données** (figure.3) l'ensemble constitué des ETCD de chaque machine et de la ligne de données.

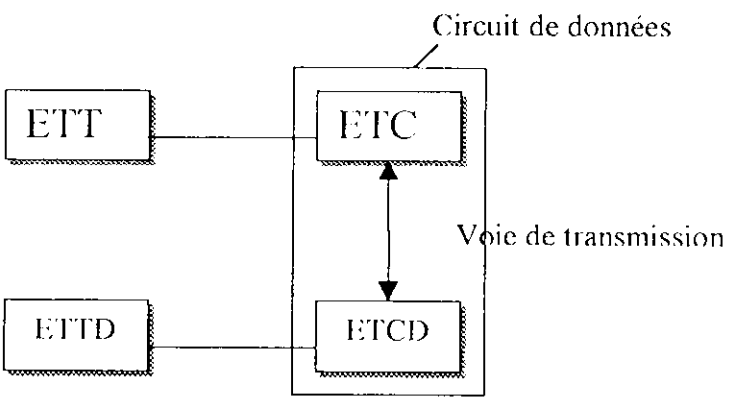


Fig.3 Circuit de données

III.1.3- Perturbations

La transmission de données sur une ligne ne se fait pas sans pertes. Tout d'abord le temps de transmission n'est pas immédiat, ce qui impose une certaine "synchronisation" des données à la réception. D'autre part des bruits ou des dégradations du signal peuvent apparaître [1].

- **les bruits :** sont l'ensemble des perturbations modifiant la forme du signal originel pouvant altérer le contenu fréquentiel . On distingue généralement deux types de bruit:

- **l'affaiblissement** du signal représente la perte de signal en énergie dissipée dans la ligne. L'affaiblissement se traduit par un signal de sortie plus faible que le signal d'entrée et est caractérisée par la valeur :

$$A = 20\log(\text{Niveau du signal en sortie/Niveau du signal en entrée}).$$

L'affaiblissement est proportionnel à la longueur de la voie de transmission et à la fréquence du signal.

- **la distorsion** du signal caractérise le déphasage entre le signal en entrée et le signal en sortie.

III.1.4- Bande passante et capacité

La bande passante d'une voie de transmission est l'intervalle de fréquence sur lequel le signal ne subit pas un affaiblissement supérieur à une certaine valeur (généralement 3db, car 3décibel correspondent à un affaiblissement du signal de 50%, on a donc) [1].

Une ligne de téléphone a par exemple une bande passante comprise entre 300 et 3400 Hertz environ pour un taux d'affaiblissement égal à 3db.
La capacité d'une voie est la quantité d'informations (en bits) pouvant être transmis sur la voie en 1 seconde. La capacité se caractérise de la façon suivante

$$C = W \log2 (1 + S/N)$$

- C capacité (en bps)
- W La largeur de bande (en Hz)
- S/N représente le rapport signal sur bruit de la voie.

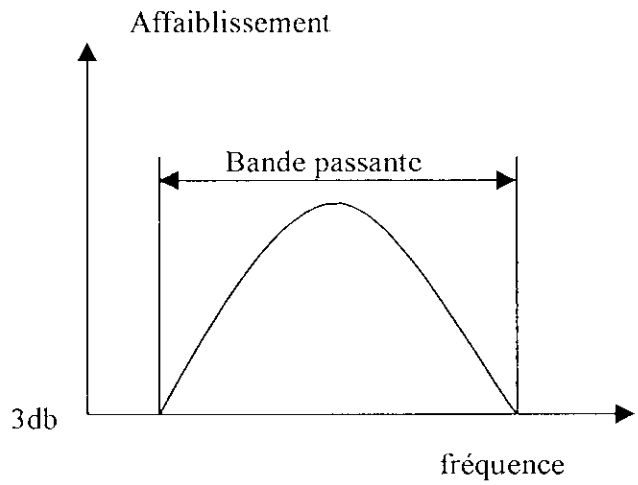


Fig.4 Courbe d'affaiblissement d'un signal

III.1.5- Notion de Codage

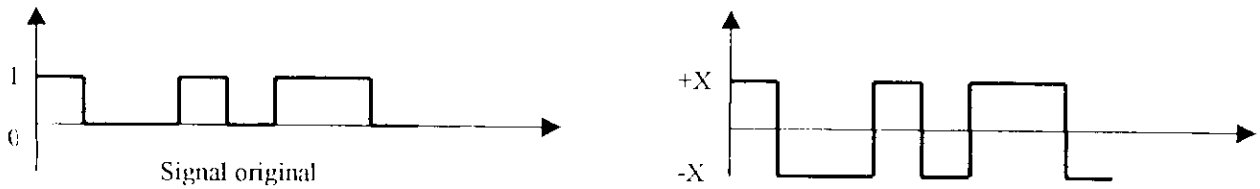
Pour que la transmission soit optimale, il est nécessaire que le signal soit codé de façon à faciliter sa transmission sur le support physique [1]. Il existe pour cela différents systèmes de codage pouvant se classer en deux catégories: [2]

- Le codage à deux niveaux: le signal peut prendre uniquement une valeur strictement négatives ou strictement positive (-X ou +X, X représentant une valeur de la grandeur physique permettant de transporter le signal)
- Le codage à trois niveaux: le signal peut prendre une valeur strictement négatives, nulle ou strictement positive (-X, 0 ou +X)

III.1.6- Divers codages

a- *Codage NRZ*

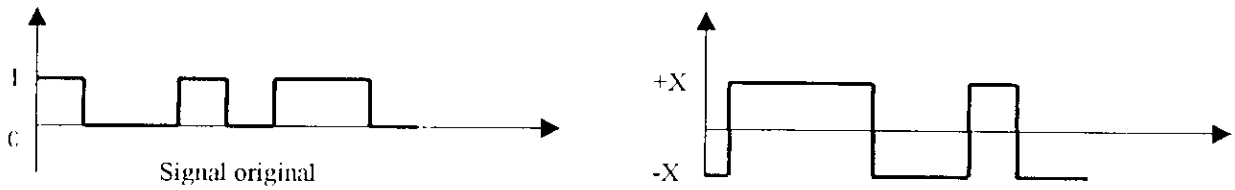
Le codage NRZ (signifiant No Return to Zéro, soit Non Retour à Zéro) est le premier système de codage, car le plus simple. Il consiste tout simplement à transformer les 0 en -X et les 1 en +X, de cette façon on a un codage bipolaire dans lequel le signal n'est jamais nul. Par conséquent, le récepteur peut déterminer la présence ou non d'un signal.



Codage NRZ

b- *Codag NRZI*

Le codage NRZI est sensiblement différent du codage NRZ. Avec ce codage, lorsque le bit est à 1, le signal change d'état après le top de l'horloge. Lorsque le bit est à 0, le signal ne subit aucun changement d'état.



Codage NRZ1

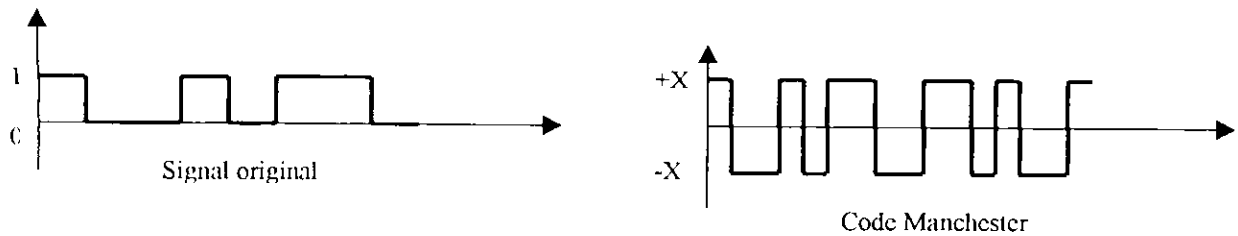
Le codage NRZI possède de nombreux avantages, dont:

- La détection de la présence ou non du signal
- La nécessité d'un faible courant de transmission du signal

Par contre, il possède un défaut: la présence d'un courant continu lors d'une suite de zéro, gênant la synchronisation entre émetteur et récepteur.

c- *Codage Manchester*

Le codage Manchester, également appelé *codage biphasé* ou *PE* (pour *Phase Encode*), introduit une transition au milieu de chaque intervalle. Il consiste en fait à faire un OU exclusif (XOR) entre le signal et le signal d'horloge, ce qui se traduit par un front montant
lorsque le bit est à zéro, un front descendant dans le cas contraire.



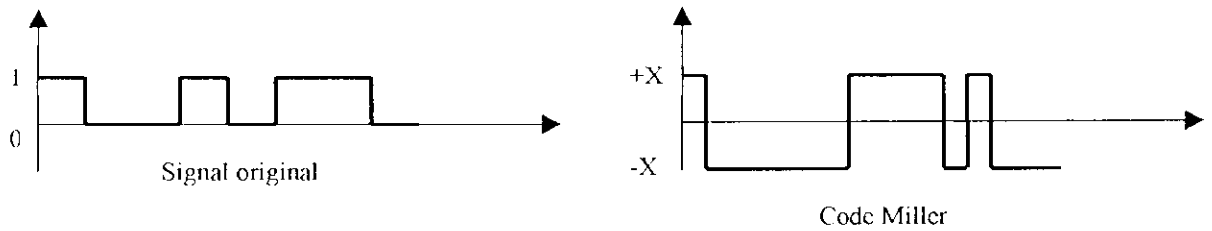
Code Manchester

Le codage Manchester possède de nombreux avantages, dont:

- le non passage par zéro, rendant possible par le récepteur la détection d'un signal.
- un spectre occupant une large bande.

d- ***Codage Delay Mode (de Miller)***

Le codage Delay Mode, aussi appelé code de Miller, est proche du codage de Manchester, à la différence près qu'une transition apparaît au milieu de l'intervalle uniquement lorsque le bit est à 1, cela permet de plus grands débits...



e- ***Codage bipolaire simple***

Le codage bipolaire simple est un codage sur trois niveaux. Il propose donc trois états de la grandeur transportée sur le support physique:

- La valeur *0* lorsque le bit est à 0
- Alternativement X et -X lorsque le bit est à 1

III.1.7- Différents modes de transmission

a- ***transmission en bande de base***

Ce n'est réalisable que pour des transmissions par câbles. Ce mode de transmission permet des débits élevés mais nécessite des distances courtes.

- il n'est pas nécessaire de moduler le signal après codage.
- le signal émis sur la ligne est celui obtenue après le codage.
- l'intérêt de ce codage est le coût peu élevé.
- utilise les différents type de codage cités précédemment.

Les signaux bandes de base [1] sont sujets a une atténuation dont l'importance dépend du support. Ils doivent être régénérés périodiquement en utilisant des répéteurs.

b- ***Transmission en bande transposée***

Pour réaliser une transmission longue distance, on module une onde porteuse sinusoïdale de la forme : [1]

$s(t)=A \cdot \sin(wt+\Phi)$.

avec A :amplitude, F :fréquence, ω :pulsation, Φ :phase initiale.

On distingue différents types de modulation. On cite :

- Modulation d'amplitude : Le signal est modulé en faisant varier l'amplitude $s(t)=A(t) \cdot \sin(\omega t+\Phi)$.
- Modulation de fréquence : $s(t)=A \cdot \sin(2\Pi f(t).t+\Phi)$.
- Modulation de phase : $s(t)=A \cdot \sin(2\Pi ft+\Phi(t))$.

c- *Différents types de standard de transmission*

Plusieurs normes [2],ont été définies pour les réseaux locaux par l'IEEE. La différence essentielle entre ces normes porte sur l'utilisation du support physique de transmission. En revanche elles sont compatibles au niveau de la couche liaison.

Ces normes sont définies en plusieurs parties :

- La norme IEEE 802.1 traite de l'architecture, définit les primitives utilisées et décrit le contexte des normes.
- La norme IEEE 802.2 correspond au protocole LLC de la couche de liaison.
- Les normes de IEEE 802.3 à 802.5 traitent les trois types de réseaux locaux de l'IEEE qui sont le CSMA/CD, le bus à jeton et l'anneau à jeton. Ces normes sont appliquées sur les cartes réseaux.

III.1.8- Cas d'Ethernet

Ethernet [2] (aussi connu sous le nom de *norme IEEE 802.3*) est une technologie de réseau local basé sur le principe que toutes les machines du réseau Ethernet sont connectées à une même ligne de communication, constituée de câble cylindriques. On distingue différentes variantes de technologies Ethernet suivant le diamètre des câbles utilisés:

- 10Base-2: Le câble utilisé est un câble coaxial de faible diamètre.
- 10Base-5: Le câble utilisé est un câble coaxial de gros diamètre.
- 10Base-T: Le câble utilisé est une paire torsadée, le débit atteint est d'environ 10Mbps.
- 100Base-TX: Comme 10Base-T mais avec une vitesse de transmission beaucoup plus importante (100Mbps).

Technologie	Type de câble	Vitesse	Portée
10Base-2	Câble coaxial de faible diamètre	10Mb/s	185m
10Base-5	Câble coaxial de gros diamètre (0.4 inch)	10Mb/s	500m
10Base-T	Double paire torsadée	10 Mb/s	100m
100Base-TX	Double paire torsadée	100 Mb/s	100m

Tableau 1 différants types de câbles.

Ethernet est une technologie de réseau très utilisée car le prix de revient d'un tel réseau n'est pas très élevé

a- *Principe de transmission*

Tous les ordinateurs d'un réseau Ethernet sont reliés à une même ligne de transmission, et la communication se fait à l'aide d'un protocole appelé *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detect* ce qui signifie qu'il s'agit d'un protocole d'accès multiple avec surveillance de porteuse (*Carrier Sense*) et détection de collision).
Avec ce protocole, toute machine est autorisée à émettre sur la ligne à n'importe quel moment et sans notion de priorité entre les machines. Cette communication se fait de façon simple:
Chaque machine vérifie qu'il n'y a aucune communication sur la ligne avant d'émettre
Si deux machines émettent simultanément, alors il y a collision (c'est-à-dire que plusieurs trames de données se trouvent sur la ligne au même moment).

Les deux machines interrompent leur communication et attendent un délai aléatoire, puis la première ayant passé ce délai peut alors ré émettre. Ce principe est basé sur deux contraintes: la taille maximale des paquets de données et le temps d'attente entre deux transmissions.

Le temps d'attente varie selon la fréquence des collisions:

- Après la première collision une machine attend une unité de temps
- Après la seconde collision la machine attend deux unités de temps
- Après la troisième collision la machine attend quatre unités de temps
- ... avec bien entendu un petit temps supplémentaire aléatoire

b- *Topologies standards*

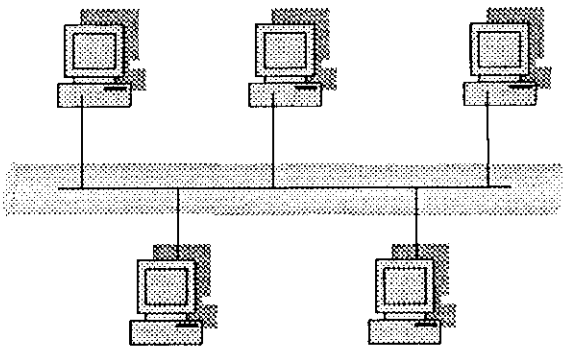
Toutes les architectures réseau dérivent de trois topologies fondamentales : [2]

- En bus.
- En étoile.
- En anneau.

Si les ordinateurs sont connectés les uns à la suite des autres le long d'un seul câble (segment), on parle de *topologie en bus*. Si les ordinateurs sont connectés à des segments de câble qui partent d'un même point (concentrateur), on parle de *topologie en étoile*. Si les ordinateurs sont connectés à un câble qui forme une boucle, on parle de *topologie en anneau*. Ces trois topologies de base sont simples en elles-mêmes. Toutefois, les topologies utilisées dans la pratique combinent souvent les caractéristiques de plusieurs d'entre elles et peuvent donc s'avérer plus complexe.

b.1- *Topologie de type bus*

La topologie en bus (*bus topology*) est également connue sous le nom de bus linéaire. Il s'agit de la méthode la plus simple et la plus fréquente de connexion d'ordinateurs. Cette technique consiste à connecter tous les ordinateurs du réseau les uns à la suite des autres, à l'aide d'un câble unique baptisé tronçon (*trunk*) ou épine dorsale (*backbone*) ou encore segment (*segment*).

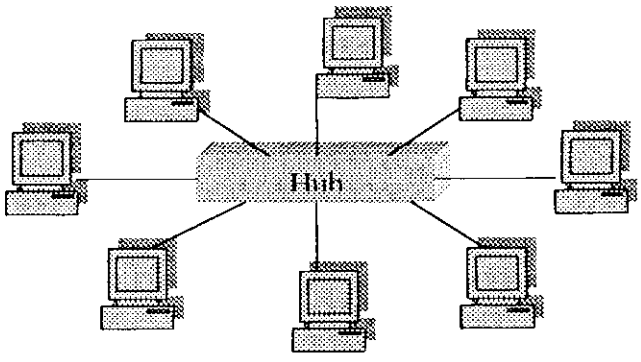


Les données sont émises sur tout le réseau ; elles ne sont acceptées que par l'ordinateur dont l'adresse correspond à celle qui a été codée comme adresse de destination de la trame. Un seul ordinateur à la fois peut envoyer des données.

Inconvénient : En cas de rupture du câble commun, le réseau sera dit hors service car il y aura alors rebond du signal (sauf si l'on y rajoute un bouchon de terminaison)

b.2- *Topologie de type étoile*

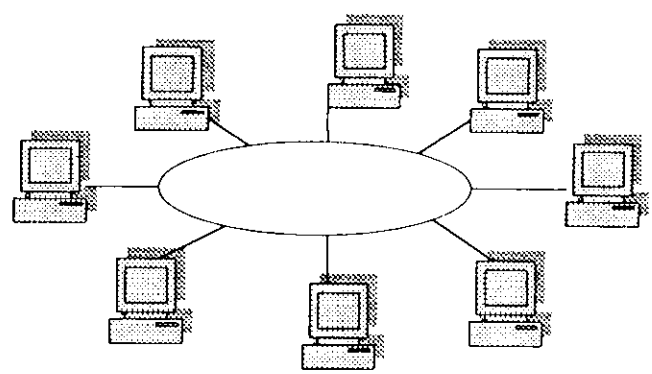
Dans une topologie en étoile (*Star topology*), les ordinateurs sont reliés par des segments de câble à un composant central appelé concentrateur (*hub*). Par l'intermédiaire du concentrateur, les signaux sont transmis depuis l'ordinateur émetteur vers tous les ordinateurs du réseau. Cette topologie date des débuts de l'informatique, lorsque les ordinateurs étaient connectés à un gros système centralisé (Unité



de traitement). Les réseaux en étoile apportent une administration et des ressources centralisées. Cependant, comme chaque ordinateur est relié à un point central, cette topologie exige davantage de câblage dans le cas d'un grand réseau. De plus, si le point central tombe en panne, le réseau tout entier est mis hors service.
Si un ordinateur ou le câble qui le relie au concentrateur est défaillant, seul cet ordinateur sera incapable de recevoir ou d'envoyer des données sur le réseau en étoile. Le reste du réseau continuera à fonctionner.

b.3- ***Topologie de type anneau***

Dans la topologie en anneau (*ring topology*), les ordinateurs sont connectés sur une seule boucle de câble. Il n'y a pas d'extrémités dotées de bouchons de terminaison. Les signaux se déplacent le long de la boucle dans une seule direction et passent chacun des ordinateurs. Contrairement à ce qui se passe avec une topologie en bus passive, chaque ordinateur fait office de répéteur afin d'amplifier le signal et de l'envoyer à l'ordinateur suivant. Dans la mesure où le signal passe par tous les ordinateurs, La panne d'un ordinateur peut avoir une incidence sur l'ensemble du réseau.



Aujourd'hui, de nombreuses topologies sont des combinaisons de bus, d'étoile et d'anneau. (Ex : Internet).

III.1.9- Support de transmission

Au fil de l'évolution des composants électroniques, différents types de câblages ont vu le jour, faisant appel à des technologies différentes [1], le but étant toujours d'atteindre les objectifs suivants:

- Grande bande-passante.
- Possibilité d'utiliser ces câbles sur de longues distances.
- Faible encombrement, facile à poser et à installer.
- Connecteurs simples et résistants.
- Faible coût...

III.1.9.1- Espace guidé

➢ **Les câbles en paire torsadée.**

Une paire torsadée est constituée de deux files torsadées, chacun étant protégé par un isolant électrique en polyéthylène. Un câble en paires torsadées est généralement constitué de quatre paires. Ce nombre permet de supporter les futurs réseau à haut débit comme, par exemple, Ethernet à 100 méga bits/s ou ATM. Ce type de câbles peut transmettre aussi bien des signaux analogiques (téléphonique et vidéo) que numérique (téléphonique numérique, vidéo et réseaux locaux).

On trouve dans le cas des réseaux sous l'appellation **RJ45** .C'est un câble qui ressemble au câble téléphone (mais de meilleure qualité. Les fils du câble sont torsadés 2 par deux en paires. Il y a souvent 4 paires de fils mais seules deux sont utilisées en 10/100Mbits, l'une pour envoyer les données et l'autre pour les recevoir. De même les connecteurs ont 8 broches mais seules les paires sur les broches 1; 2 et 3; 6 sont utilisées.
Il existe trois versions de câbles torsadés :

a- Paire torsadée non blindée (UTP)

Câbles isolés dont les fils sont torsadés les uns autour des autres à raison d'un certain nombre de torsade par mètre. Les torsades permettent de réduire les interférences signal entre les fils. Plus il y a de torsades par mètre, moins il y a de risque de diaphonie. Les paires torsadées non blindées sont classées à en cinq catégories selon la qualité du câble et sa capacité à véhiculer les signaux. Seule l'absence d'isolation ou de blindage différencie les paires torsadées non blindées des paires torsadées blindées.

c- Paire torsadée blindée (STP)

Câbles isolés par des fils torsadés les uns autour des autres (tresse) à raison d'un certain nombre de torsade par mètre. Les torsades permettent de réduire les interférences de signal entre les câbles, plus il y a de torsade par mètre, moins il y a de risque de diaphonie.

b- Paire torsadée blindée (FTP)

Câbles isolés dont le blindage est réalisé par des feuilles d'aluminium en couches successives. Ce feuillard permet de réduire les interférences de signal.

➢ *Câble coaxial*

Le câble coaxial est constitué d'un premier conducteur au cœur du câble, qui conduit le signal électrique, d'un diélectrique (où isolant), d'un deuxième conducteur sous forme de métal tressé assurant le blindage, et enfin d'une gaine de plastique assurant la protection mécanique de l'ensemble.

Les câbles coaxiaux les plus répandues sont classés selon leur impédance caractéristique (qui est pour les courants alternatifs ce que la résistance est au courant continu) :

- 50 ohms pour les transmissions numériques exclusivement.
- 75 ohms pour les signaux analogiques (antennes de télévision) et numérique.

Les premiers sont appelés câbles bande de base, câbles qui véhicule un seul signal numérique composé de "0" matérialisés par une absence de courant et de "1" matérialisés par une présence de courant. Les seconds sont appelés larges bandes car ils peuvent véhiculer plusieurs signaux analogiques à des fréquences différentes (plusieurs chaînes de télévision dans le cas des câbles d'antenne). Dans les deux cas, chaque signal peut être multiplexé dans le temps pour transporter plusieurs informations.

➢ ***Fibres optiques***

Les câbles en fibre optique sont formés d'une fibre très fine en verre ou en plastique, entourée par une gaine protectrice. Les signaux sont transmis dans la fibre sous forme lumineuse. Les signaux lumineux peuvent se propager dans la fibre sur une longue distance, sans qu'il y ait nécessité d'amplification comme avec les câbles de métal. La fibre optique présente une très bonne immunité aux bruits, comparé aux autres supports, et elles possèdent aussi une très large bande passante.

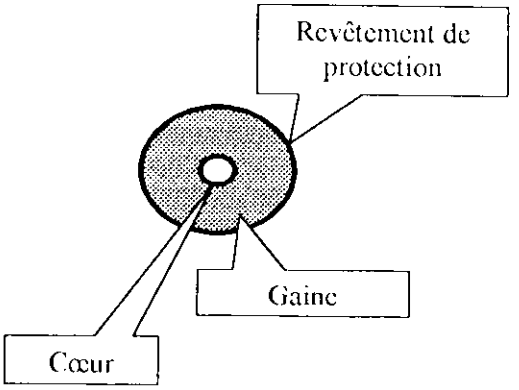


Fig.5 fibre optique

Depuis peu, la fibre optique se répand de plus en plus largement car elle dispose de qualités que non pas les câbles en cuivre, à savoir :

- Une insensibilité aux perturbations électromagnétique ;
- Une vitesse de transmission plus élevée ;
- Une propagation des signaux sur de plus grand de distance.

❖ ***Les différents types de fibres***

Deux grands types de fibres et existent : les multimodes et les monomodes. Elles se distinguent par leur composition et par le mode de cheminement des rayons lumineux.
Dans la catégorie des fibres multimodes, on distinguer celles à gradient d'indice (onde de forme sinusoïdale) et celle à saut d'indice (la réfaction à angle droit). Dans une fibre à gradient d'indice, l'index de réfaction décroît du centre à la périphérie. La vélocité de la lumière est donc plus faible au centre. Dans une fibre a saut d'indice, il n'y a pas de gradation dans l'indice de réfaction.

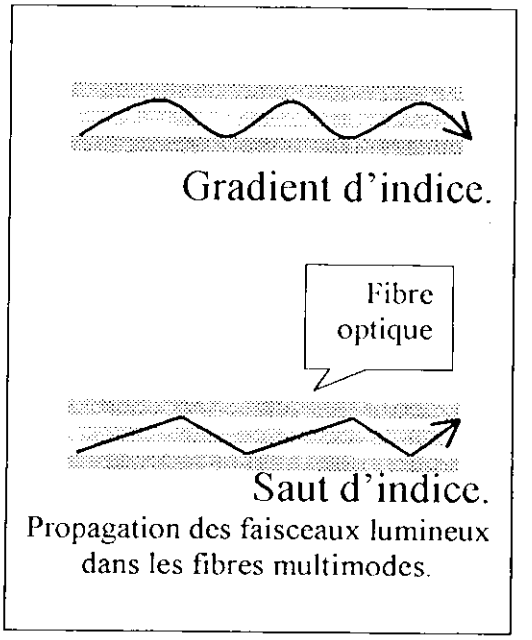


Fig.6 deux type de fibre

L'indice de réfraction dans une fibre monomode peut être constant ou décroissant du centre à la périphérie. Le diamètre du cœur est sensiblement égal à la longueur d'onde du faisceau lumineux, soit moins de 10 microns.

Les fibres monomode sont de plus caractérisées par la longueur de coupure, au-dessus de laquelle le régime de propagation est axial. Ce type de fibre représente ainsi des dispersions chromatiques exprimées en ps/nm par kilomètres dues au fait que les différentes ondes contenues dans le spectre de la source se propagent à des vitesses légèrement différentes. C'est ce facteur qui limite la bande passante sur les fibres monomode.
Les fibres multimodes les plus couramment utilisées en informatiques sont celles que gradient d'indice ayant un cœur d'un diamètre de 62,5 microns et une gaine optique de 125 microns de diamètre. Leur désignation courante est 62,5/125. Les caractéristiques des fibres de ce type decrites par le tableau suivantes :

	Multimode à saut d'indice	Multimode à gradient d'indice	Monomode
Source lumineuse	LED ou laser	LED ou laser	Laser
Bande passante	20 à 200 Mhz.km	200 MHz à 1,5 GHz.km	3 à 50 GHz.km
Diamètre du cœur	De 50 à 125 µ	De 50 à 125 µ	De 2 à 8 µ
Diamètre de la gaine	De 125 à 440 µ	De 125 à 440 µ	De 15 à 60 µ
Application	Informatique Image Téléphonie	Lignes téléphoniques de moyenne portée Image	Lignes de télécommunication longues distances

Tableau 2 Caracteristique Des Fibres Optiques

III.1.9.2 - Espace libre

De plus en plus de types de liaisons telles que les liaisons satellites géostationnaires, les constellations de satellites en orbite basse et les liaisons terrestres sans fils sont utilisées comme support de transmission dans l'Internet. Ces liaisons ont des caractéristiques spécifiques en ce qui concerne le délai, la gigue, le taux d'erreur, mais aussi en terme de symétrie de la liaison et même de sa bi-directionnalité. Le système de communication étant la composante principale du satellite. Il est constitué d'une ou de plusieurs antennes qui reçoivent et émettent sur une large bande de fréquence, d'un jeu d'émetteurs et de récepteurs qui amplifient et retransmettent les signaux reçus. Au fil des ans, la capacité de retransmission des satellites s'est accrue considérablement, alors que les coûts par ligne téléphonique ont baissé de façon spectaculaire.
Deux grandes orientations se détachent dans le domaine : HiperLAN et IEEE 802.11.

a- *HiperLAN*

Le nom est l'abréviation de **High performance radio LAN**. Les fréquences retenues se situent entre 5.15 et 5.30 GHz ainsi qu'une bande de 200 MHz autour de 17 GHz (HiperLAN 2). Les vitesses de transfert devraient être de 10 à 20 Mbps et les communications se font directement de station à station ou par l'intermédiaire d'un nœud central.
Les communications peuvent se faire sur 5 canaux distinctes de priorité différente. L'adaptation du CSMA/CD appelée **EY-NPMA** (Elimination Yield None Preemptive Priority Multiple Access) consiste à scruter les canaux par ordre de priorité jusqu'à trouver un canal libre pour émettre.
Le niveau 2 du modèle OSI est divisée en deux sous-couches, la sous-couche **CAC** (Channel Access Control) qui correspond à la partie physique de la technique d'accès (gestion des problèmes liés au canal hertzien ainsi que toute la transmission et réception) et la sous-couche **MAC** qui correspond à la partie

logique, soit la mise en forme de la trame, le routage interne, les algorithmes de confidentialité, la gestion de priorité (QoS) et l'insertion et le retrait des stations.

b- *IEEE 802.11*

Les communications peuvent se faire soit de station à station, soit par une borne de concentration. Une station ne peut par contre pas relayer les paquets vers une autre station terminale. Les débits sont de 1 ou 2 Mbps selon qu'on utilise le codage FHSS (Frequency Hopping Spread Spectrum) qui utilise un saut de fréquence ou le codage DSSS (Direct Sequence Sequence Spread Spectrum) qui code de façon continue. L'utilisation de l'infrarouge est également possible.
La technique d'accès est plus compliquée, il s'agit du **CSMA/CA** (Collision Avoidance). Pour éviter les collisions, plusieurs temporisateurs propres à chaque station sont attribués se qui limite la probabilités d'avoir deux stations émettant dans les mêmes microsecondes

III.1.10- Modules constituants un réseau

L'interconnexion ne se limite pas au niveau Ethernet, et à un but bien précis qui est de raccorder des réseaux locaux entre eux. Les Matériels utilisés [2] ne sont pas forcement spécifiques à Ethernet.

Les Types de Matériels utilisés sont les suivant :

- Répéteur (repeater).
- Multirépéteur (étoile, hub).
- Routeur (router).
- Pont-routeur (B-Router).
- Passerelle (gateway)
- Commutateur.

➢ ***Répéteur***

Les répéteur (*Repeaters*) sont à comparer à des amplificateurs qui régénèrent le signal et qui permettent ainsi d'étendre la distance maximum de transmission.

➢ ***Hubs (concentrateurs)***

Les Hubs permettent la connexion de plusieurs nœuds sur un même point d'accès sur le réseau, en se partageant la bande-passante totale. La structure physique qui s'en dégage est une étoile, mais la topologie logique reste un bus (pour Ethernet).

➢ ***Routeurs***

Les routeurs sont utilisés pour les réseaux étendus et complexes, lorsque les signaux du réseau peuvent emprunter de nombreux chemins d'accès pour atteindre la même destination. Ils choisissent la plus pratique puis envoient le signal.

➢ ***Ponts(Bridge)***

Ce type d'équipement, assure une segmentation physique et logique du réseau. Seul les paquets destinés à un équipement situé de l'autre côté du Bridge le traverse.

Cela signifie que le trafic local entre les nœuds A et B ne traverse pas le Bridge et n'encombre ainsi pas le segment de droite. Le trafic est *filtré*, les collisions ne sont pas propagées.

➢ ***Ponts-routeurs (B-Routeur)***

Ils fonctionnent de la même façon que les routeurs mais ils présentent en outre les avantages des ponts. Ils fonctionnent comme un routeur avec les protocoles routables et comme un pont avec les protocoles non routables. L'acquisition d'un pont-routeur peut s'avérer d'un meilleur rapport qualité/prix que l'achat d'un pont et d'un routeur séparés.

➢ ***Passerelles***

Matériel qui sert à connecter des réseaux utilisant des protocoles différents de façon à pouvoir faire passer les informations d'un système à l'autre. Les passerelles opèrent généralement au niveau de la couche réseau du modèle OSI.

➢ ***Commutateurs***

C'est un mélange de hub et de pont. Comme le pont il travaille avec l'adresse mac (niveau 1). Par contre le fait d'avoir plusieurs ports lui permet de ne renvoyer les paquets que vers le port où se trouve l'ordinateur. De plus il fait une copie numérique du paquet, ce qui remet à 0 le compteur du nombre de hub. Pour les mêmes raisons que pour le pont on n'utilise pas le switch pour la sécurité.

III.2- La couche liaison de données

La tâche principale de cette couche est d'assurer le contrôle du trafic des données lors de transferts d'informations au niveau des connexions électriques, c'est à dire au niveau des cartes de communication. Elle assure également la transmission sans erreur ou duplication, de plus elle met en forme l'information afin qu'elle soit insérée dans une chaîne de caractères conforme au protocole en vue être envoyée sur le câble via la couche 1.
Elle prend en compte la capacité du canal (débit binaire), offre au utilisateur une grande liberté quant à l'envoie et la réception d'un flux de données et contrôle le volume de mémoire dont disposent les nœuds (buffers ou mémoires tampon) .

III.3- La couche réseau

D'une façon générale le problème de routage se posera pour la grande majorité des réseaux. La solution dépendra d'un certain nombre de caractéristiques de ceux-ci.
Il faut donc considérer :

- La manière dont st pris en compte l'ordre de transmission entre les différents paquets composant un même messages.
- Le type du réseau (point à point, multipoint, local, etc.).
- Le problème de congestion pour chacun des types.

C'est là ou intervient la couche réseau qui a pour rôle de gérer le sous-réseau la façon dont les paquets sont acheminés de la source au destinataire constitue un élément clé de sa conception. Si trop de paquets se trouvent simultanément dans le sous-réseau, il va se créer des engorgements. Le contrôle d'une telle congestion est aussi du domaine de la couche réseau .

III.4- La couche transport

La fonction de base de la couche transport est d'accepter des données de la couche session, de les découper, le cas échéant, en plus petites unités, de les passer à la couche réseau et de s'assurer que tous les paquets arrivent à destination. La couche transport crée une connexion réseau par connexion de transport requise par la couche session. Elle peut cependant en créer plusieurs en cas de gros débit. La couche transport a pour tâche de rendre ce multiplexage transparent à la couche session. La couche transport est une authentique couche de bout en bout, de l'émetteur au destinataire, en d'autres termes, un programme de la machine source soutient une conversation avec un programme similaire sur la machine destinataire en utilisant les messages d'en-tête et de contrôle. En plus, du multiplexage de plusieurs messages sur un canal, la couche transport doit gérer l'établissement et le relâchement des connexions sur le réseau. Cela nécessite un processus d'adressage permettant au processus initiateur d'indiquer avec qui il veut converser. Il existe également un contrôle de flux permettant à la couche transport d'éviter qu'une machine hôte rapide ne sature une machine plus lente.

III.5- La couche session

La couche session constitue un cas à part parmi l'ensemble des couches composant la structure des réseaux ouverts. En particulier la définition de ses caractéristiques implique que dans nombre de réseaux existants, cette couche n'est guère distinguée de la couche transport et dans certains cas, elle n'est même pas mentionnée. Elle permet le transport de données, comme la couche transport, mais elle offre également des services évolués utiles à certaines applications. Une session permet à un utilisateur d'accéder à un système à temps partagé distant ou de transférer un fichier entre deux machines. Les sessions peuvent autoriser le mode bi- ou unidirectionnel du trafic, elle peut aider à savoir qui a la parole en mode unidirectionnel. Un autre service de session est la synchronisation. La couche session permet d'insérer des points de reprise dans le flot de données de façon à ne reprendre après une panne ou une interruption de transfert que le transfert des données suivant le dernier point de reprise.

III.6- La couche présentation

La couche présentation remplit trois fonctions suffisamment courantes et génériques pour n'être pas laissées à la charge de l'utilisateur. Ces fonctions sont :

- Optimise le fonctionnement du réseau, et plus particulièrement économise la capacité des artères de transmission.
- Les problèmes liés à la sécurité ne doivent pas être oubliés. Le développement accéléré des réseaux, rend le phénomène de hacking de plus en plus aigu.
- Le développement des réseaux entraîne de plus en plus fréquemment à l'interconnexion de matériels dont les caractéristiques différent, ce qui conduit obligatoirement à définir des standard pour la communications entre ces matériel.

III.7- La couche application

Etant La couche la plus élevée du modèle OSI, elle est la plus proche de l'utilisateur. En fait c'est la nécessite des organes distant de disposer de programmes dit d'applications qui est à l'origine du développement du concept même du réseau. A cela s'ajoute d'autres fonctions propres à cette couche:

- Le transfert de fichiers.
- L'exécution de travaux à distance.
- la consultation du courrier électronique et de documents.

IV- Application à la conception d'une architecture réseau

IV.1- Les protocoles de communication

Les protocoles sont les programmes qui permettent l'échange d'informations entre des ordinateurs. Les protocoles réseau courants [2] sont les suivants : NetBEUI, TCP/IP, IPX/SPX etc.....

- *IPX/SPX* (Internetwork Packet Exchange/Sequenced Packet Exchange) est un protocole réseau standard pour de nombreux sites. Il gère le routage et peut prendre en charge des applications client-serveur NetWare, où des applications utilisant des sockets et connaissant NetWare communiquent avec des applications IPX/SPX utilisant des sockets.
  - C'est l'interface de la plupart des réseaux Novell.
  - C'est un protocole routable.
  - C'est un protocole robuste, capable de gérer beaucoup d'applications réseau.
  - Il est simple et efficace pour les types de communications pour lesquels il a été conçu (partage de fichiers et d'imprimante).
  - Il est simple à configurer.
- *UDP* : Le protocole User Datagram Protocol (UDP) est défini dans le but de fournir une communication par paquet unique entre deux processus dans un environnement réseau étendu. Ce protocole suppose l'utilisation du protocole IP comme support de base à la communication.
  Ce protocole définit une procédure permettant à une application d'envoyer un message court à une autre application, selon un mécanisme minimaliste. Ce protocole est transactionnel, et ne garantit ni la délivrance du message, ni son éventuelle duplication.
- *TCP/IP* est le protocole le plus répandu dans le monde. La plupart des systèmes d'exploitation supportent ce protocole. Un large éventail de logiciels (des navigateurs Internet aux bases de données client/serveur) sont conçus pour utiliser ce protocole.
  - Il est suffisamment robuste pour supporter les demandes de communications.
  - Il peut router et segmenter le réseau et en soulager le trafic. Ainsi des interfaces bien définies (sockets), il peut contrôler le type de trafic circulant sur le réseau. C'est une des clés de la sécurité, et c'est ainsi fonctionnent les *firewalls.*
  - C'est protocole de transmission multi-usages.
  - Il demande un pue plus de travail pour le configurer et permet ainsi d'un ordinateur de dialoguer avec un autre.
- *NetBEUI* (extension du vieux protocole NetBIOS Extended User Interface) est aux antipodes en terme de standardisation et de robustesse. Ce protocole est généralement utilisé sur de petits réseaux locaux (LAN) de la taille d'un service, comprenant de 1 à 200 clients. La seule méthode de routage qu'il peut utiliser est le routage source Token Ring.
  Ces avantages et inconvénients sont les suivants :
  - Il n'est pas routable, donc il y'a risque de perte d'accès à certains ordinateurs contrairement à TCP/IP.
  - Il est simple et efficace avec les tâches pour lesquels il a été conçu.
  - Il est supporté par une grande variété de systèmes d'exploitation.
  - Incapacité à prendre en compte les transactions d'applications client/serveur.

Il est simple à configurer et est un bon choix pour les petits réseaux locaux.

V- Système d'exploitation et les réseaux

Il n'y pas si longtemps, pour relier une machine à un réseau, il fallait se procurer le paquetage permettant de la configurer, et cela indépendamment du système d'exploitation.

L'évolution de l'informatique a permis au programmeur d'intégrer un noyau dynamique (micro noyau) qui permet l'ajout et la suppression de modules, offrant un environnement fiable et sécurisé. Il permet aussi de gérer d'autres paramètres réseau tel que les ports d'entrée/sortie nécessaire à la communication entre système distant.

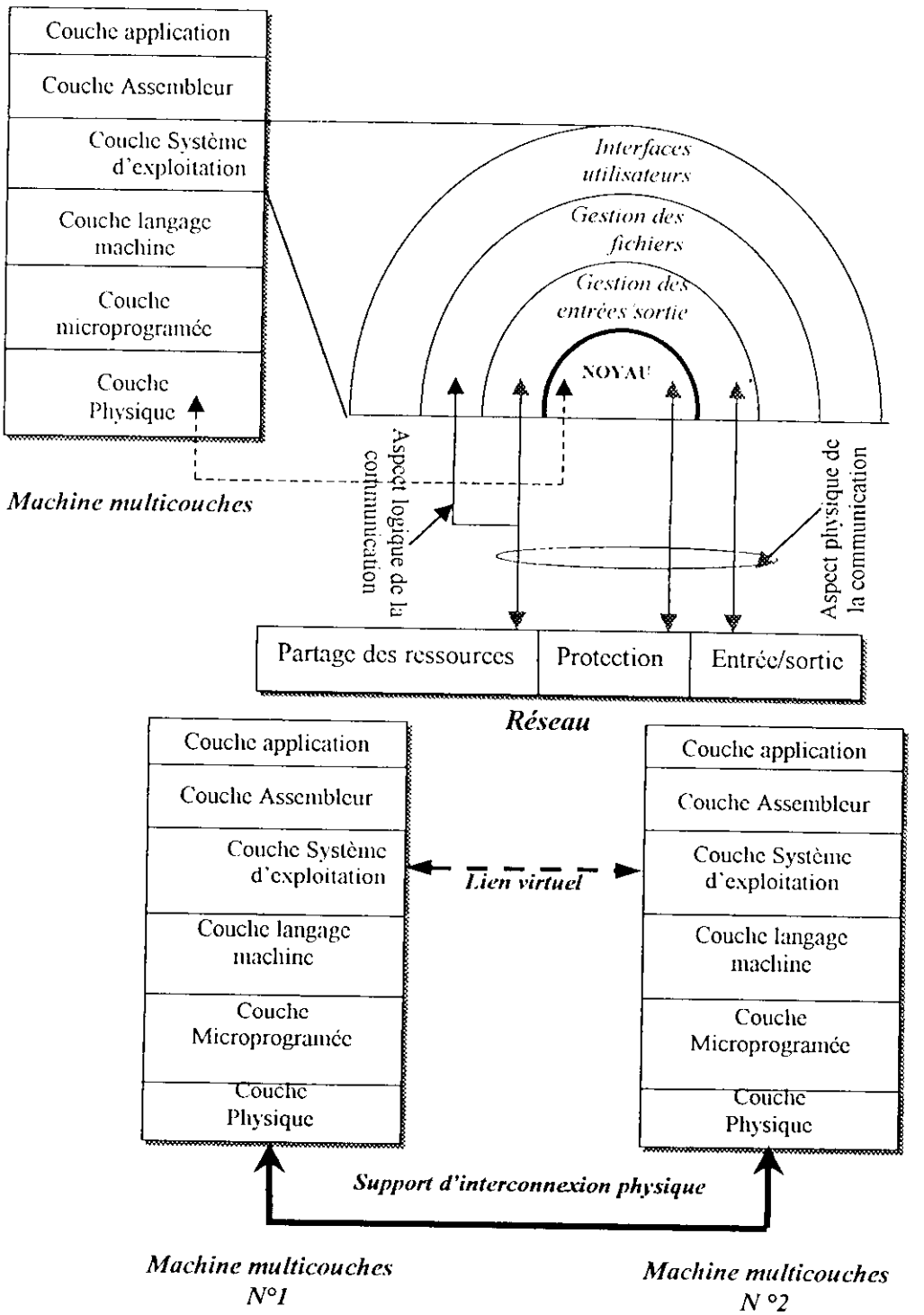


Fig.7 Modélisation des liens entre les divers éléments d'un réseau

Dans notre étude on s'interessera à le système d'exploitation Linux, qui est un clone d'UNIX , et l'étude de LINUX revient a étudier UNIX.

V.1- Présentation d'UNIX

Le système Unix est un système multi-utilisateurs et multi-tâches offrant aux utilisateurs de nombreux utilitaires interactifs [4]. En tant que système d'exploitation, son rôle principal est d'assurer aux différentes tâches et aux différents utilisateurs une bonne répartition des ressources de l'ordinateur (mémoire, processeur, espace disque, imprimante.....) et cela sans intervention des utilisateurs : il les prend totalement en charge et lorsque les demandes sont trop importantes pour être satisfaites rapidement, l'utilisateur le ressent par un certain ralentissement.
Unix est par ailleurs un système où les utilisateurs ont à leur disposition un très grand nombre d'outils permettant d'écrire, mettre au point et documenter leurs programmes .

En résumé, on peut dire que le système est composé de :

- Un noyau assurant la gestion de la mémoire, la gestion des entrées-sorties et l'enchaînement des différentes tâches.
- Un ensemble d'outils de base parmi lesquels :
  - Différents interpréteurs de langages de commande appelés shells.
  - Des commandes permettant la manipulation de fichiers, la gestion des activités du système (les processus) et la communication entre utilisateurs.
  - Des outils de traitement de textes.
  - Des outils généraux de développement : débogueurs, archiveurs, des compilateurs de langages et un éditeur de liens.

V.2- Principales caractéristiques du système

Les principales caractéristiques du système sont :

- Un système hiérarchisé de processus.
- Un système hiérarchisé de fichiers.
- Un ensemble de points d'accès aux services offerts par le noyau à partir d'applications développées dans un langage évolué. Ces points d'accès sont les appels système.
- Un aspect multitâche avec ou sans multi-fenêtrage. Dans le premier cas, l'utilisateur a la possibilité de travailler dans différentes fenêtres de l'écran ; chaque fenêtre se comportant comme un terminal .
- Des langages de commandes qui constituent de véritables langages de programmation, permettant l'écriture de nouvelles commandes complexes .

Unix contient une série de composants réalisant des fonctions bien précises. L'illustration de la figure 9 donne une idée des interactions entre ces diverses fonctions.

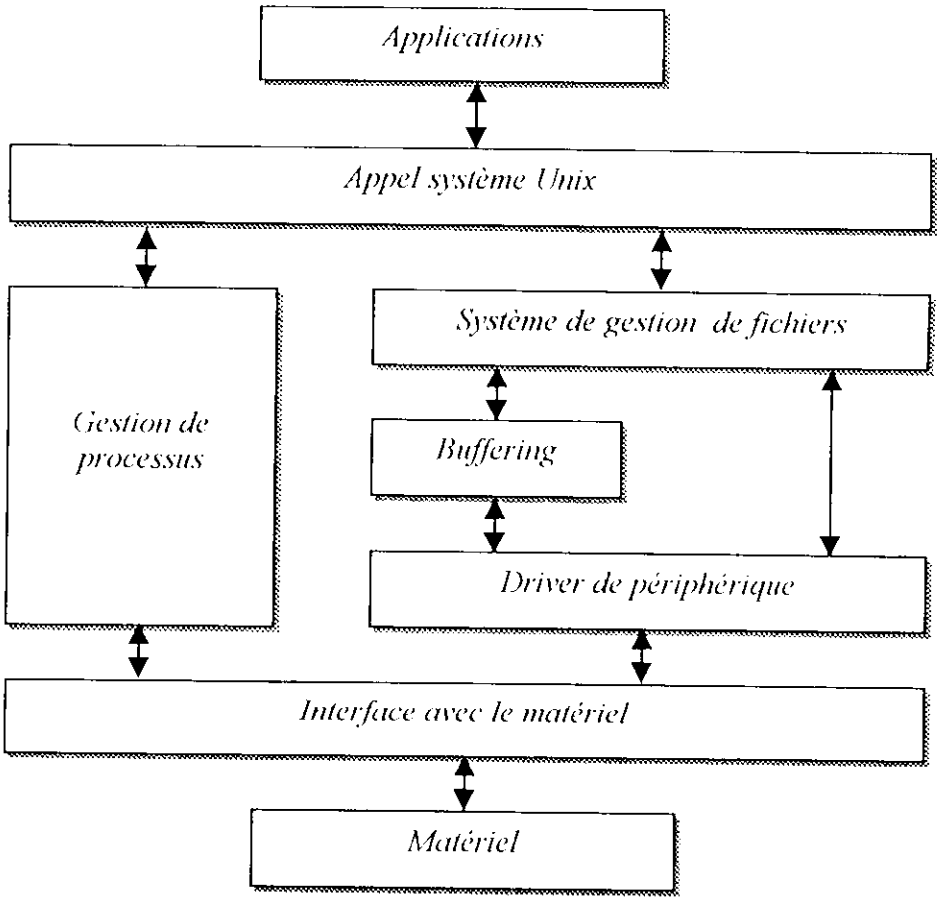


Fig. 8 Structure du système d'exploitation Unix

VI- Notion de service réseau - Model client serveur

VI.1- Présentation de l'architecture d'un système client/serveur

De nombreuses applications fonctionnent selon un environment client/serveur [21], cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en terme de capacités d'entrée-sortie, qui leur fournit des services. Ces services sont des programmes fournissant des données
telles que l'heure, des fichiers, une connexion, ...
Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes. On parle ainsi de client FTP, client de messagerie, ..., lorsque l'on désigne un programme, tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès du serveur (dans le cas du client FTP il s'agit de fichiers, tandis que pour le client messagerie il s'agit de courrier électronique).
Dans un environnement purement Client/serveur, les ordinateurs du réseau (les clients) ne peuvent voir que le serveur, c'est un des principaux atouts de ce modèle.

VI.2- Avantages de l'architecture client/serveur

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont:

- des ressources centralisées: étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction
- une meilleure sécurité: car le nombre de points d'entrée permettant l'accès aux données est moins important
- une administration au niveau serveur: les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés
- un réseau évolutif: grâce à cette architecture ont peu supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modifications majeures

VI.3- Inconvénients du modèle client/serveur

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles:
-un coût élevé dû à la technicité du serveur
-un maillon faible: le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui! Heureusement, le serveur a une grande tolérance aux pannes (notamment grâce au système RAID) .

VI.4- Fonctionnement d'un système client/serveur

Un système client/serveur fonctionne selon le schéma suivant:

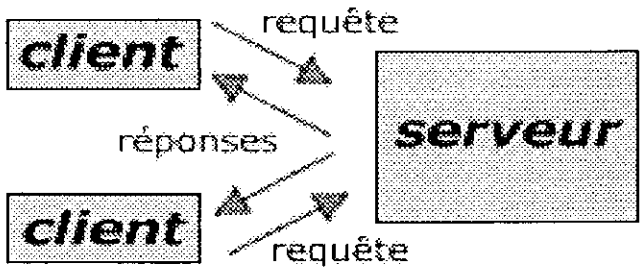


Fig.9 système client /serveur

Le client émet une requête vers le serveur grâce à son adresse et le port, qui désigne un service particulier du serveur. Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port

VI.5- Présentation de l'architecture à 2 niveaux

L'architecture à deux niveaux (aussi appelée *architecture 2-tier*, *tier* signifiant *étage* en anglais) caractérise les systèmes clients/serveurs dans lesquels le client demande une ressource et le serveur la lui fournit directement. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir le service.

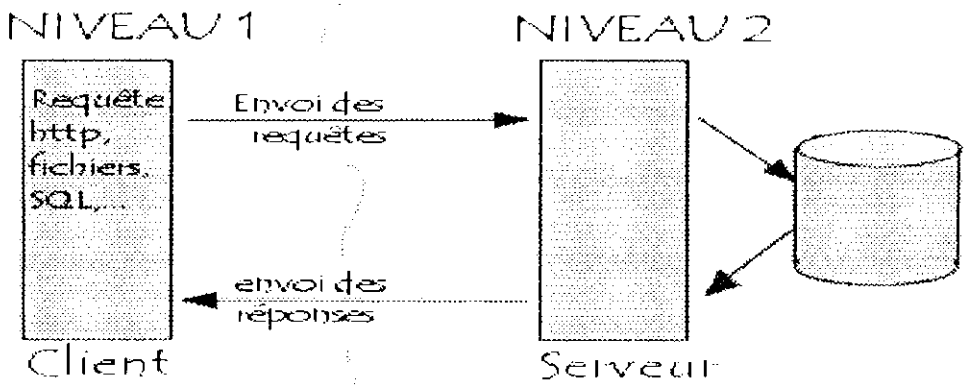


Fig. 10 L'architecture à deux niveaux

VI.6- Présentation de l'architecture a 3 niveaux

Dans l'architecture à 3 niveaux (appelées *architecture 3-tier*), il existe un niveau intermédiaire, c'est-à-dire que l'on a généralement une architecture partagée entre:
Le client: le demandeur de ressources

1. Le serveur d'application (appelé aussi middleware): le serveur chargé de fournir la ressource mais faisant appel à un autre serveur
2. Le serveur secondaire (généralement un serveur de base de données), fournissant un service au premier serveur

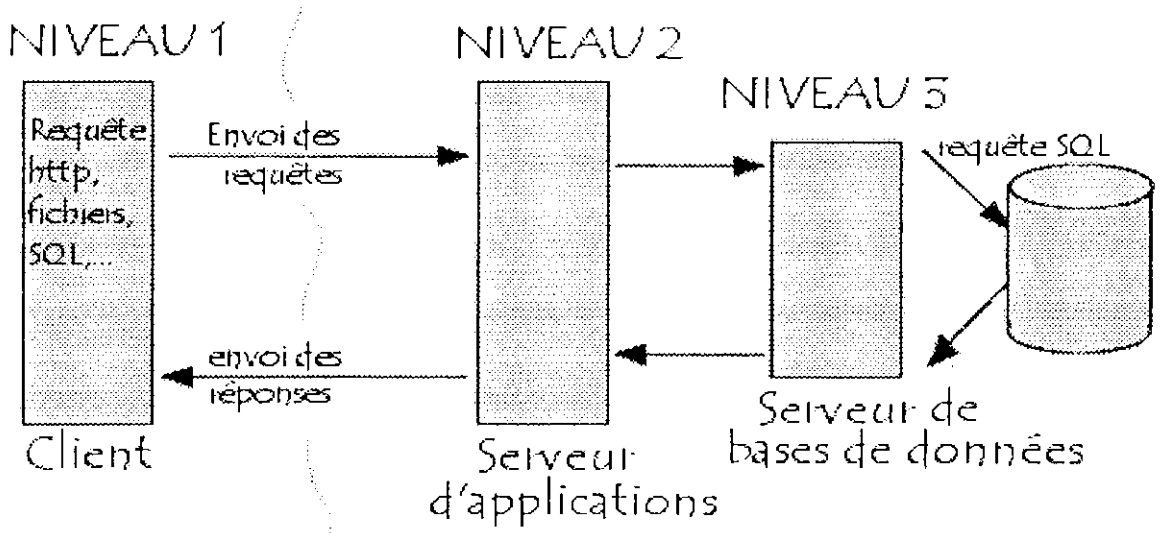


Fig. 11 L'architecture à trois niveaux

Etant donné l'emploi massif du terme d'architecture à 3 niveaux, celui-ci peut parfois désigner aussi les architectures suivantes:

- Partage d'application entre client, serveur intermédiaire, et serveur d'entreprise
- Partage d'application entre client, base de données intermédiaire, et base de données d'entreprise

VI.7- Comparaison des deux types d'architecture

L'architecture à deux niveaux est donc une architecture client/serveur dans laquelle le serveur est polyvalent, c'est-à-dire qu'il est capable de fournir directement l'ensemble des ressources demandées par le client. Dans l'architecture à trois niveaux par contre, les applications au niveau serveur sont délocalisées, c'est-à-dire que chaque serveur est spécialisé dans une tâche (serveur web/serveur de base de données par exemple). Ainsi, l'architecture à trois niveaux permet:

- une plus grande flexibilité/souplesse
- une plus grande sécurité (la sécurité peut être définie pour chaque service)
- de meilleures performances (les tâches sont partagées)

VI.8- L'architecture multi-niveaux

Dans l'architecture à 3 niveaux, chaque serveur (niveaux 1 et 2) effectue une tâche (un service) spécialisée. Ainsi, un serveur peut utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service. Par conséquence, l'architecture à trois niveaux est potentiellement une architecture à N niveaux...

VII- Conclusion

Dans ce chapitre on a abordé les concepts et les notions fondamentales des réseaux locaux ; autrement dit les mécanismes des 2 couches basses du model O.S.I. Le choix de l'étude des réseaux locaux est justifié par le fait qu'un réseau Intranet est un réseau local. Le prochain chapitre constituera le complément spécialisé à cette présentation pour aborder les services implémentés dans un réseau propriétaire type Internet.

Introduction

C'est vers 1980 qu'est apparu le réseau Internet, tel qu'on le connaît maintenant, lorsque l'ARPA (*Advanced Research Project Agency*) commença à faire évoluer les ordinateurs de ses réseaux de recherche vers les nouveaux protocoles TCP/IP et qu'elle se mit à subventionner l'université de Berkeley pour qu'elle intègre TCP/IP à son système d'exploitation Unix BSD (Berkly software Distribution). Ainsi, la quasi totalité des départements d'informatique des universités américaines put commencer à se doter de réseaux locaux qui, en quelques années, seront interconnectés entre eux.

I- Le protocole TCP/IP

La figure suivant montre les relations entre les couches TCP/IP [22] et celles définies dans le modèle ISO [5].

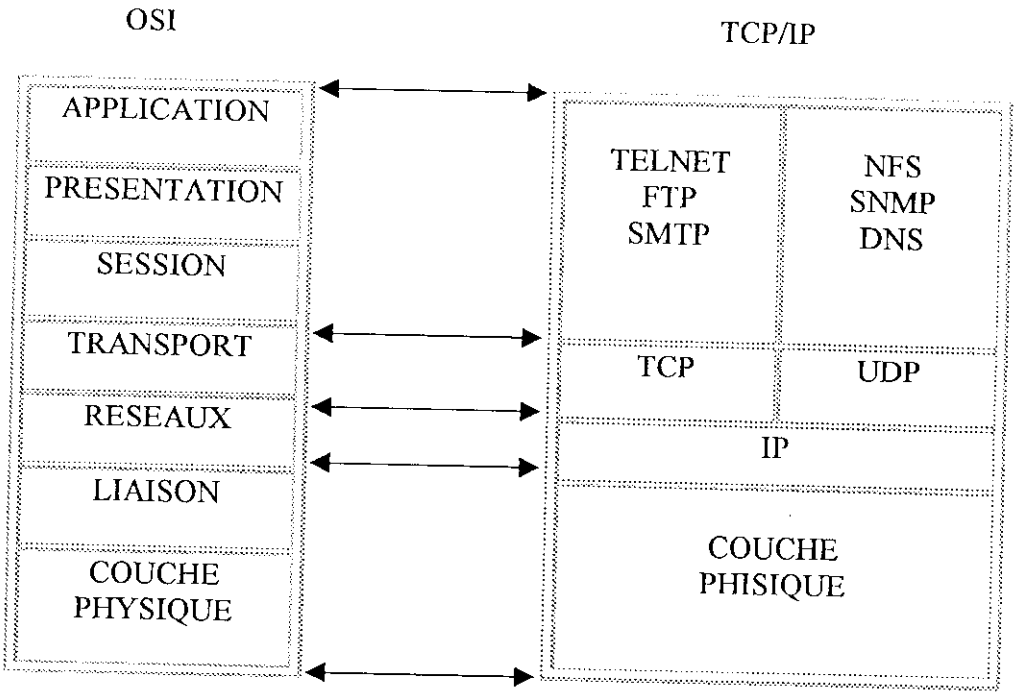


Fig. 1 Comparaison couches OSI et TCP/IP.

II- Le Protocole TCP

Le protocole TCP est défini dans le but de fournir un service de transfert de données de haute fiabilité entre deux ordinateurs raccordés sur un réseau de type paquets commutés.
TCP s'intègre dans une architecture multicouche de protocoles, juste au-dessus du protocole Internet IP. Ce dernier permet à TCP l'envoi et la réception de segments de longueur variable, encapsulés dans un paquet Internet appelé datagramme. Le datagramme Internet dispose des mécanismes permettant l'adressage d'un service TCP source et un destinataire, quelles que soient

leur position dans le réseau. Le protocole IP s'occupe aussi de la fragmentation et du réassemblage des paquets TCP lors de la traversée de réseaux de plus faibles caractéristiques. Le protocole IP transporte aussi les informations de priorité, compartimentation et classification en termes de sécurité relatives aux segments TCP. Ces informations se retrouvent alors transmises de bout en bout de la communication.

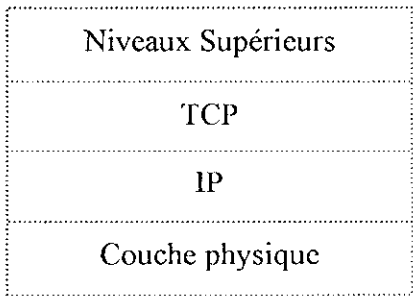


Fig. 2 Couches de protocoles

II.1- Modèle de service TCP

Pour que deux ordinateurs puissent établir une connexion entre eux, il faut créer deux points de connexion sur chaque ordinateur appelé socket. Chaque socket possède:

- un numéro (ou adresse) constitué de l'adresse IP de l'ordinateur hôte.
- d'un port constitué d'un nombre de 16 bits local à cet ordinateur.

Pour obtenir un service TCP, on doit établir une connexion de manière explicite entre deux sockets:

- l'un sur la machine émettrice.
- l'autre sur la machine réceptrice.

Toutes les connexions TCP sont :

- Bidirectionnelles: les données peuvent circuler dans les deux sens simultanément.
- Point à point: chaque connexion est définie par deux extrémités et deux seulement. Pas de multidistribution (multicasting) et de diffusion (broadcasting).

II.2- Fonctionnement

TCP est conçu pour fournir un service de transmission de données sécurisé entre deux machines raccordées sur un réseau de paquets. Pour pouvoir assurer ce service même au dessus d'une couche de protocole moins fiable, les fonctionnalités suivantes sont nécessaires:

- Transfert de données de base
- Correction d'erreur
- Contrôle de flux
- Multiplexage
- Gestion de connexions
- Priorité et Sécurité

En-tête du segment TCP:

0	8	16	23	31
Port Source			Port destination	
Numéro de séquence				
Numéro d'accusé de réception				
Long en-tête TCP	U A P R S F R C S S Y I G K H T N N		Taille de fenêtre	
Total de contrôle			Pointeur d'urgence	
Options (0,1 ou plusieurs mots de 32 bits).				
Données (optionnelles)				

Fig. 3 datagramme TCP

- **Port Source / Port Destinataire**

Identifient les extrémités locales de la connexion. Chaque ordinateur décide lui-même de la façon dont il alloue ses propres ports à partir du 256ème. Un port et l'adresse IP de son ordinateur forment une adresse unique de 48 bits. Le doublet (numéro de socket source, numéro de socket destination) identifie la connexion.

- **Numéro de séquence**

De longueur de 32 bits, ce numéro du datagramme TCP permet à l'ordinateur destinataire de remettre les paquets dans le bon ordre.

- **Numéro d'accusé de réception**

Indique le prochain octet attendu, et non le dernier octet correctement reçu (32 bits).

- **Longueur de l'en-tête TCP**

Indique combien de mots 32 bits contient l'en-tête, indique aussi le point de départ des données au sein du segment.

- **Champ de 6 bits non utilisé**

A l'origine ce champ était conçu pour permettre la correction de problèmes au sein du protocole, la bonne conception de TCP fait que ce champ n'est pas utilisé.
Champ de 6 drapeaux d'un bit chacun:

- URG: est positionné à 1 si le pointeur d'urgence est en cours d'utilisation.
- ACK: est positionné à 1 pour indiquer la validité du Numéro d'accusé de réception.
- PSH: signifie pushed, indique on destinataire de remettre les données à l'application concernée sans les mettre dans la mémoire tampon.
- RST: réinitialise la connexion après un arrêt brutal de l'ordinateur et rejette les segments erronés.
- SYN: sert à établir la connexion.
- FIN: sert à libérer la connexion.

➢ **Taille de fenêtre**

Indique combien on peut transmettre d'octets après l'octet acquitté.

➢ **Total de Contrôle**

Permet d'assurer une très grande fiabilité en rejetant les segments lorsque ce champ est différent de zéro.

➢ **Pointeur d'urgence**

Indique où l'on peut trouver les données urgentes.

➢ **Options**

Permet de rajouter des possibilités comme la taille maximale d'un segment que peut recevoir un ordinateur ou la demande de renvoi de certains paquets que le destinataire n'a pas reçu, évitant à l'hôte de devoir réexpédier tout les paquets même si il en a reçu correctement.

Le fait que TCP soit un protocole fiable s'explique par le fait qu'il assure la transmission totale des paquets émis par un ordinateur. Lorsqu'un paquet n'arrive pas à destination et que l'émetteur ne reçoit pas d'accusé de réception, il doit donc remettre le paquet. La question qui se pose est le temps au bout duquel TCP doit remettre le paquet, ceci implique la gestion des temporisations.

III- Le protocole Internet (Internet Protocol ou IP)

Le but des adresses IP est de fournir un service de communication universel permettant à toute machine de communiquer avec toute autre machine de l'interconnexion Une machine doit être accessible aussi bien par des humains que par d'autres machines. Pour cela une machine doit pouvoir être identifiée par :

- Un nom (mnémotechnique pour les utilisateurs).
- Une adresse qui doit être un identificateur universel de la machine.
- Une route précisant comment la machine peut être atteinte.

Pour remédier a ces contraintes il a été choisit un adressage binaire compacte qui assure un routage efficace. L'Internet adresse ou IP adresse est composé de 32 bits. On y distingue le netid et le hostid.

Le netid identifie un réseau et un hostid identifie une machine sur ce réseau. L'interface utilisateur concernant ls adresses IP consiste en la notion de quatre entiers décimaux séparés par un point, chaque entier représentant un octet de l'adresse IP : 10000000 00001010 00000010 00011110 est écrit : 128.10.2.30.

Il réalise les fonctionnalités de la couche réseau selon le modèle OSI. ee situe au cœur de l'architecture TCP/IP qui met en œuvre un mode de transport fiable (TCP) sur un réseau en mode non connecté. Le service offert par le protocole IP est dit non fiable (pas de réponse de la machine receveuse) c'est-à-dire que la remise des paquets est non garantie. Il n'y a pas de connexion ente les datagrammes (chaque paquets est traité indépendamment les uns des autres) et la diffusion d'un message est faite pour le mieux (best effort, les paquets ne sont pas éliminés sans raison). Le protocole IP définit :

1. L'unité de donnée transférée dans les interconnexions (datagramme).
2. La fonction de routage.
3. Les règles qui mettent en œuvre la remise de paquets en mode non connecté.

III.1- Composition d'un datagramme

L'unité de base dans un réseau Internet est le datagramme codé sur 32 bits qui est constituée d'un en-tête et d'un champ de données :

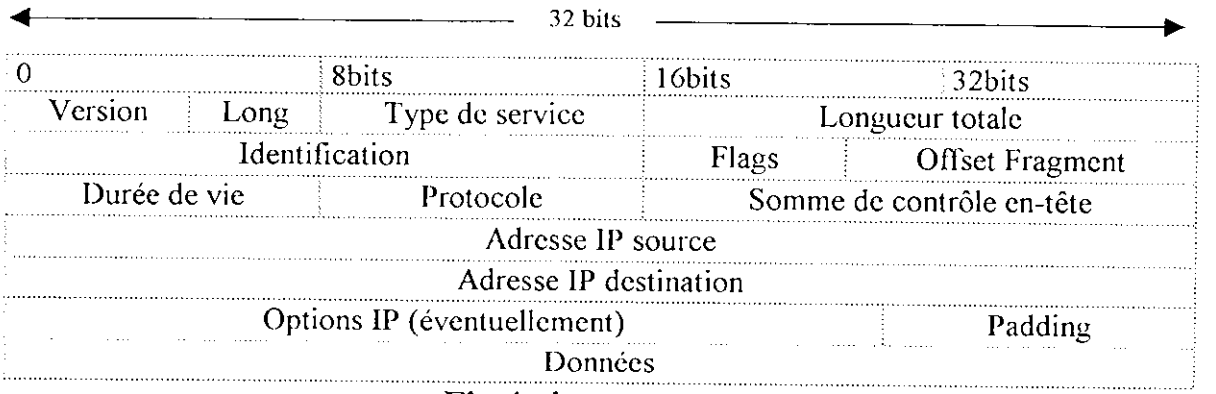


Fig.4 datagramme IP

- **VERS**
  numéro de version de protocole IP.
- **HLEN**
  longueur de l'en-tête en mots de 32 bits, généralement égal à 5(pas d'option)
- **Type de service**
  indique comment le datagramme doit être géré, Précédence | D | T | R | Inutilisé :
  - Précédence : 3 bits qui définissent la priorité du datagramme ; en général ignoré par les machines et passerelles.
  - Bits D, T, R : indiquent le type d'acheminement désiré du datagramme, permettant à une passerelle de choisir entre plusieurs routes (si elles existent) : D signifie délai court, T signifie débit élevé et R signifie grande fiabilité.
- **Longueur total**
  longueur total du datagramme (en-tête plus données).
- **Identification**
  permet à l'ordinateur destinataire de déterminer à quel datagramme appartient le fragment reçu.
- **Flags (Drapeau)**
  indique si le routeur a le droit de fragmenter ou non le datagramme
- **Offset Fragment**
  précise la localisation ou le déplacement dans le datagramme courant.
- **Durée de vie**

ce champ indique en seconde, la durée maximale de transit du datagramme sur l'internet. La machine qui émet le datagramme définit sa durée de vie. Les passerelles qui traitent le datagramme doivent décrémenter sa durée de vie du nombre de secondes que le datagramme a passé pendant son séjour dans la passerelle ; lorsque celle-ci expire le datagramme est détruit et un message d'erreur est renvoyé à l'émetteur.

- **Protocole**

ce champ identifie le protocole de niveau supérieur dont le message est véhiculé dans le champ données du datagramme : 6 TCP ; 17 UDP ; 1 ICMP.

➢ **Somme de contrôle de l'en-tête**

permet de détecter les erreurs survenant dans l'en-tête, et par conséquent l'intégrité du datagramme.

➢ **Options**

le champ d'options est facultatif et de longueur variable. Une option est définie par un champ octet : C| classe d'options | Numéro d'option :

- Copie (C) indique que l'option doit être recopiée dans tous les fragments (C=1) ou bien uniquement dans le premier fragment (C=0).
- Les bits classe d'option et numéro d'option indiquent le type de l'option et une option particulière de ce type :
  - Enregistrement de route : est utilisé pour enregistrer l'itinéraire. Chaque routeur donne son adresse au datagramme
  - Routage strict prédéfini par l'émetteur : prédéfini le routage qui doit être utilisé dans l'interconnexion en indiquant la suite des adresses IP dans l'option. Le chemin spécifié ne tolère aucun autre intermédiaire. Le routage lâche permet le transit par d'autre intermédiaire
  - Horodatage : chaque routeur joint son adresse et en plus une horodate (date universelle)

➢ **Padding (Bourrage)**

les options n'ayant une taille standard, il faut rajouter des octets pour que le datagramme soit valide.

III.2- Adressage

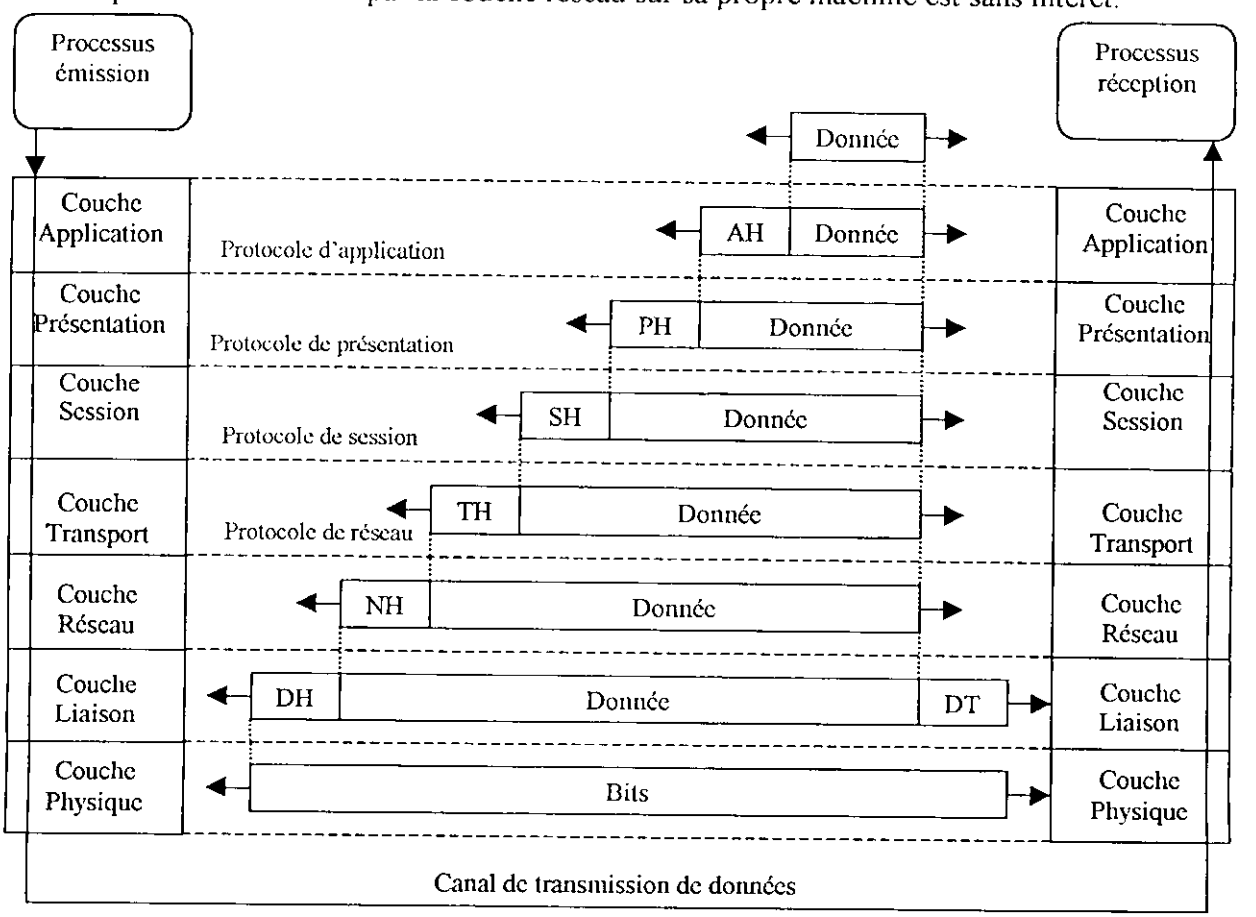
Une distinction doit être faite entre noms, adresses, et chemins. Un nom indique ce que nous cherchons. Une adresse indique où cela se trouve. Un chemin indique comment y aboutir. Le protocole Internet s'occupe essentiellement des adresses. C'est à des protocoles de niveau plus élevé (ex., hôte-vers-hôte ou application) que revient la tâche de lier des noms à des adresses. Le module Internet déduit de l'adresse Internet une adresse réseau local. La tâche qui consiste à transcrire l'adresse de réseau local en termes de chemin (ex., sur un réseau local ou dans un routeur) revient au protocole de bas niveau.

III.2.1- Fragmentation

La fragmentation du datagramme Internet devient nécessaire dès lors qu'un datagramme de grande taille arrive sur une portion de réseau qui n'accepte la transmission que de paquets plus courts. Un datagramme Internet peut être spécifié "non fractionnable" Un tel datagramme Internet ne doit jamais être fragmenté quelques soient les circonstances. Si un datagramme Internet non fractionnable ne peut être acheminé jusqu'à sa destination sans être fragmenté, alors il devra être rejeté. La fragmentation, la transmission et le réassemblage à travers un réseau local hors de vue d'un module de protocole Internet est appelée fragmentation Intranet .

IV- Le passage d'une couche i à une couche i-1 dans le cas d'un réseau TCP/IP

La figure ci-dessous montre un exemple de la façon dont les données peuvent être transmises en utilisant le modèle OSI [1]. Le processus émetteur doit émettre certaines données vers le processus récepteur. Il délivre les données à la couche application qui lui accole un en-tête application AH (qui peut être nul) et délivre l'item résultant à la couche présentation.
La couche présentation peut transformer cet item de différentes façons, éventuellement rajouter un en-tête et délivrer le résultat à la couche session.
Il est important de comprendre que la couche présentation ne connaît pas et ne doit pas connaître l'existence éventuelle de AH qui fait pour elle partie des données utilisateur.
Ce processus se répète, jusqu'à ce que les données atteignent la couche physique, là elle sont effectivement transmises à la machine réceptrice. Inversement les en-tête sont éliminés lorsque les données remontent les différentes couches jusqu'au processus récepteur.
Ce concept est tel que la transmission est horizontale, alors qu'en réalité elle est verticale. Prenons l'exemple d'une donnée reçu par la couche transport émettrice de la couche session, elle lui accole l'en-tête de transport et l'envoie à la couche transport réceptrice. De son point de vue le fait de passer cette donnée par la couche réseau sur sa propre machine est sans intérêt.



AH : En-tête d'application
DH : En-tête de liaison de données
TH : En-tête de transport
NH : En-tête de réseau
SH : En-tête de session
PH : En-tête de présentation
DT : En-tête de fin de trame

Fig.5 Exemple d'utilisation du modèle OSI

V- Services Internet

Les services Intranet [3] se subdivisent en deux catégories : Les services utilisateurs, qui offrent des ressources et des applications destinés à l'utilisateur final, et les services réseau qui assurent la cohérence de l'environnement de réseau et lui permettent de faire fonctionner. Le rôle fondamental d'un Intranet, est de fournir des informations des informations et des services aux utilisateurs au moyen des systèmes reposants sur le HTML.

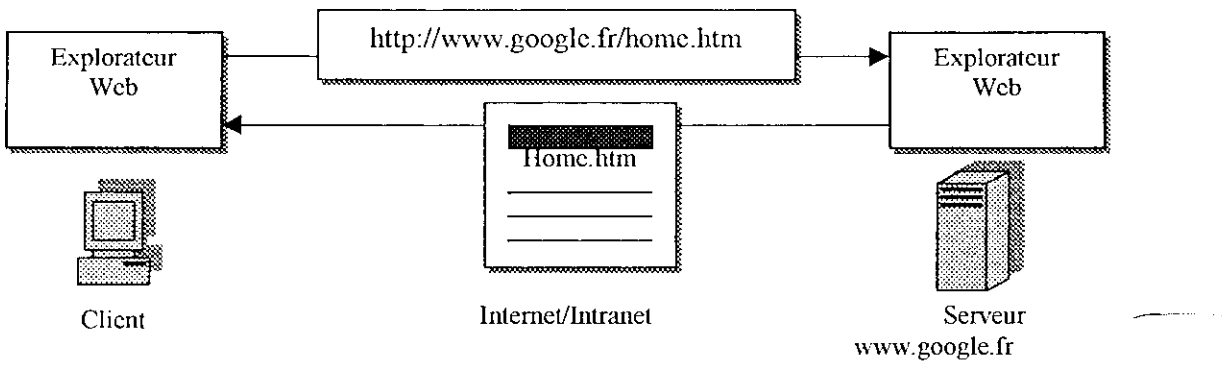


Fig.6 : Récupération de pages HTML sur Internet/Intranet

Les services offerts par un Intranet de service se subdivisent en quatre grandes catégories :

✓ **Stockage et partage de l'information :** Le rôle de base d'un Intranet est d'offrir un environnement sur et facile d'accès tout pour les personnes qui souhaitent publier des pages HTML que pour les lecteurs de celles-ci.
Les services de stockage et d'accès se composent des serveurs de fichiers (serveurs FTP),des serveurs de données (base de données) et des serveurs de documents (serveurs Web).

✓ **Communication et collaboration :** Le courrier électronique et les logiciels groupeware deviennent de plus en plus universels. Les systèmes de courrier électroniques en RTF (Rich Text Format)et de conversation en ligne intégrant du son et des séquences vidéo permettent de faire évoluer facilement les techniques de communication utilisées jusqu'à présent. En associant ces outils au calendrier et aux systèmes de planification de réseaux, les Intranets permettent de mettre en place un environnement de collaboration englobant toutes les modes de communication électronique.

✓ **Navigation :** les services de navigation sont fondamentaux sur un Intranet. l'utilisateur doit en effet pouvoir trouver rapidement ce qu'il cherche. Un Intranet devra donc faciliter à la moindre information placée sur le réseau. Il devra également lui permettre de formuler une requête unique qui génère une liste organisée recensant toutes les informations correspondantes, situées sur les différents serveurs de l'entreprise et sur l'Intranet.

✓ **Accès aux applications** : Sur un Intranet, il suffit d'une interface pour accéder à des bases de données existantes, à des stocks d'informations et à des application installées antérieurement. Les nouvelles applications peuvent ensuite être crées en CGI, puis développées sur n'importe quelle plate-forme pour l'ensemble des environnements d'ordinateurs de bureau et

de serveurs, quelque que soit le système d'exploitation utilisé. Toutes les commandes logiques des applications clientes sont téléchargées au moment ou l'utilisateur accède à l'application, et elles sont mises à jour automatiquement.

VI- Les différentes types de serveurs

- **Serveur DNS** (Domain Name System)

Chaque interface de réseau connectée à un réseau TCP/IP est identifiée au moyen d'une adresse IP unique de 32 bits. Un nom (appelé nom de la machine hôte) peut être ainsi attribué à tout périphérique ayant une adresse IP.
DNS est un système hiérarchique distribué permettant de traduire des noms de machines hôtes en adresses IP. Dans DNS, il n'existe pas de base de données centrale contenant la totalité des informations d'hôtes Internet. Les informations sont réparties parmi des milliers de serveurs de noms organisés en une structure hiérarchique identique à la hiérarchie du système de fichiers UNIX. Chaque niveau est appelé domaine. DNS possède un domaine racine au haut de la hiérarchie des domaines ; un groupe de serveurs de noms appelés serveurs racines sert celui-ci.
Les domaines du niveau supérieur sont localisés directement sous le domaine racine.

La figure 7 illustre la hiérarchie du domaine. Le domaine racine se trouve en haut de la hiérarchie ; les domaines du niveau supérieur sont situés directement sous le domaine racine ; dans cette figure, seuls les domaines organisationnels de niveau supérieur sont indiqués. Les serveurs racines disposent uniquement des informations complètes concernant les domaines du niveau supérieur qui utilisent des pointeurs vers les serveurs des domaines de deuxième niveau (nih situé sous gov et nuts sous com, sont les domaines de deuxième niveau illustrés dans la figure).

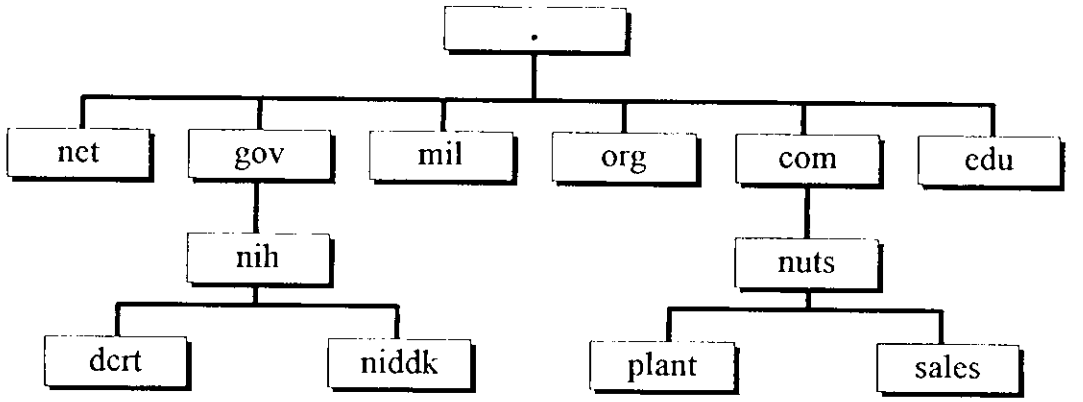


Fig.7 Hiérarchie de Domaine

Aucun serveur, ni même les serveurs racines, ne possède les informations complètes concernant tous les domaines ; les pointeurs leur précisent quels sont les serveurs ayant ces informations. Dès lors, si les serveurs racines se trouvent dans l'impossibilité de répondre à une question, ils peuvent toujours interroger le serveur approprié.

➢ Serveur DHCP

Un serveur DHCP (Dynamic Host Configuration Protocol) à pour rôle de distribuer des adresses IP à des clients pour une durée déterminée. Au lieu d'affecter manuellement à chaque hôte une adresse statique, ainsi que tous les paramètres tels que (serveur de nom, passerelle par défaut, nom du réseau...), un serveur DHCP alloue à un client, un bail d'accès au réseau, pour une durée déterminée (durée du bail). Le serveur passe en paramètre au client toutes les informations dont il a besoin.
La distribution des adresses par le serveur DHCP aux clients sous la forme de paramètres, montre bien que, tous les nœuds critiques du réseau (serveur de nom primaire et secondaire, passerelle par défaut...) doivent avoir une adresse IP statique. Si celle-ci variait, le processus, dans l'état, ne serait pas réalisable. Ce processus est mis en œuvre quand vous ouvrez une session chez un Provider (fournisseur d'accès Internet) par modem. Le fournisseur d'accès, vous alloue une adresse IP de son réseau le temps de la liaison. Cette adresse est libérée, donc de nouveau disponible, lors de la fermeture de session.

➢ Serveur Web

Un serveur Web est un ordinateur qui fait fonctionner un logiciel spécial dont le but est de répondre à des demandes de documents fournies par des clients Web (qui sont des logiciels Web). L'ordinateur en lui-même peut être tout, du mainframe au PC, mais généralement un serveur Web est une machine Unix spécialement configurée pour stocker les pages Web et les rendre disponibles sur le Net.
Les documents Web (fichiers HTML, ou tout autre fichier graphique ou multimédia) sont stockés sur une partie hard. Le logiciel serveur permet principalement aux utilisateurs extérieurs de recevoir les fichiers pour les exploiter par leur propre logiciel.

Le client Web (browser) communique avec un serveur Web à travers une ou plusieurs connexions TCP. Le port de notoriété publique du serveur Web est le port TCP 80, le protocole utilisé par le client et le serveur sur la connexion TCP est HTTP.
Un serveur Web donné «peut pointer » sur d'autres serveurs Web par le truchement des liens hypertextes. Ces liens peuvent également pointer vers d'autres objets que des serveurs Web : un serveur Web peut pointer sur un serveur FTP ou un serveur Telnet, par exemple.
La figure suivante constitue un schéma simplifié du fonctionnement du Web :

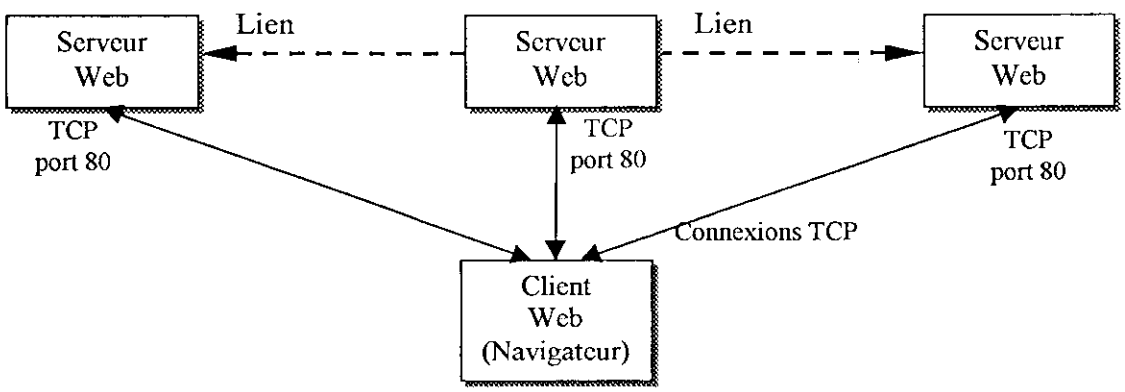


Fig. 8 Organisation d'un WEB client/serveur

➢ Serveur proxy et firewall

a- Définition d'un firewall

Dans un bâtiment, un coupe-feu (firewall) est conçu pour éviter qu'un incendie ne se répande d'une partie de l'immeuble aux autres. En théorie, un firewall Internet sert à la même chose : il empêche les dangers provenant de l'Internet de se répandre à l'intérieur de votre réseau. En pratique, il ressemble plus aux douves d'un château médiéval. Il sert à plusieurs choses :

- Il restreint l'accès à un point précis.
- Il empêche toute intrusion malveillante.
- Il restreint la sortie à un point précis.
- Un firewall Internet est le plus souvent installé au point où votre réseau interne protégé est connecté à Internet. Voir la figure

Tout le trafic provenant de l'Internet ou partant de votre réseau interne passe à travers le firewall, qui a donc la possibilité de vérifier si ce trafic est conforme à la politique de sécurité du site. Ces politiques sont parfois extrêmement restrictives et d'autres assez ouvertes.

b- Définition du serveur proxy

Un serveur proxy est un serveur spécialisé qui tourne généralement sur un firewall. Sa principale fonction est de permettre l'accès à Internet pour les clients d'un réseau local où chaque client peut avoir un accès complet au web sans compromettre la sécurité.
Le serveur proxy écoute les requêtes des clients internes et les envoie aux serveurs Internet éloignés. Il lit les réponses depuis l'Internet et les achemine aux clients internes concernés, il joue le rôle d'un mandataire (Middle–Man).

Généralement, tous les clients d'un même réseau local utilisent le même serveur proxy. Ceci rend possible et efficace le stockage de tous documents demandés par les clients pour une futur utilisation. Les utilisateurs d'un proxy doivent ressentir qu'ils obtiennent les réponses directement depuis les serveurs éloignés.

Les organisations utilisant des adresses privées pour leur réseau local peuvent aussi profiter de l'Internet puisque le proxy est visible pour l'Internet d'un coté et le réseau privé interne de l'autre coté. Le schéma suivant illustre une utilisation du serveur proxy.

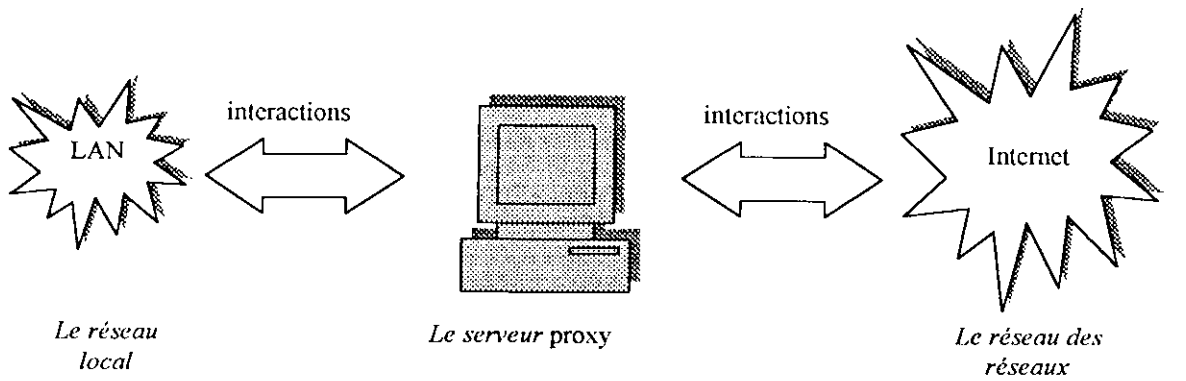


Fig.9 Utilisation d'un serveur proxy

L'ordinateur qui exécute le proxy est connecté à la fois à Internet et au réseau interne. On dit qu'il est multidomicilié (Multihome Host). Il peut être connecté directement à Internet (accès permanent) ou y accéder à distance par une ligne téléphonique ou autre. Du coté interne, il doit avoir une adresse IP permanente. Proxy peut autoriser ou interdire des requêtes en se basant sur le protocole utilisé. Par exemple il peut permettre les demandes FTP mais prohiber ceux de HTTP pour telle ou telle machine. On vient de citer l'un des avantages de proxy que nous verrons en détails juste après.

On utilise le serveur proxy pour différentes raisons comme :

1. Fournir l'accès à Internet pour un réseau privé.
2. Implémenter un cache pour les documents les plus demandés.
3. Permettre et restreindre l'accès des clients à Internet.
4. Faciliter l'enregistrement de l'activité Internet des clients.
5. Permettre l'accès contrôlé des tiers aux serveurs situés l'intérieur du réseau local : Reverse proxy.

➢ **Serveur de messagerie**

Le serveur de messagerie permet l'envoi et la réception de message et de documents à toutes personnes disposant d'une adresse électronique. Les messages sont stockés dans une boite aux lettre qui est en général de capacité limitée . l'adresse électronique est composé de :

- du nom de l'utilisateur .
- du séparateur : @ .
- du site sur lequel se trouve l'utilisateur.

Les tâches de transferts et de réception du courrier sont des tâches qui s'exécutent généralement en arrière plan. La diffusion du courrier s'effectue en fonction de l'adresse électronique du ou des destinataires, en utilisant les protocoles suivant :smtp, pop, imap, mime.

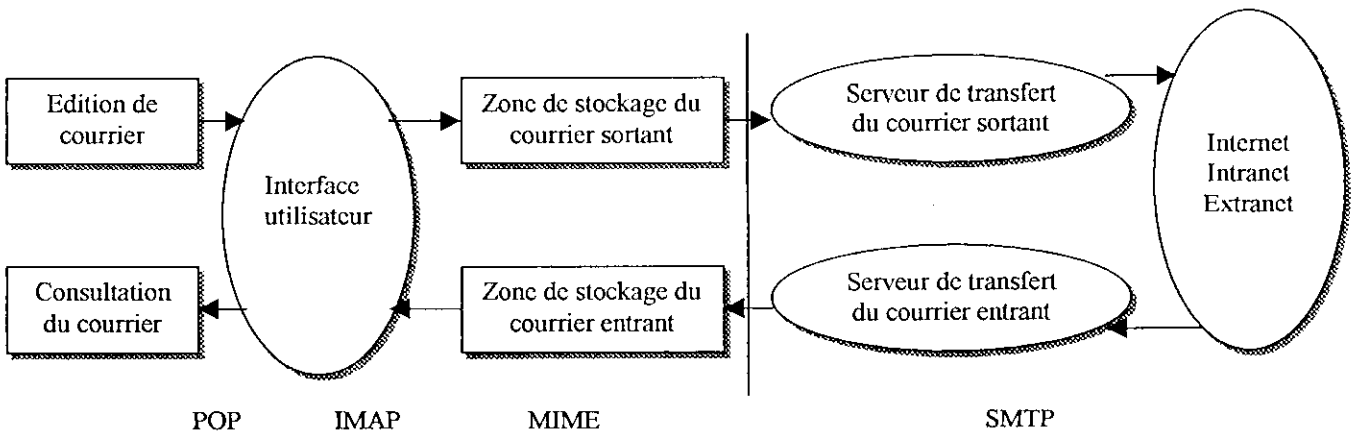


Fig.10 principe des messageries.

➢ **Serveur FTP**

FTP (File Transfer Protocol) est une application communément utilisée. C'est un standard d'Internet pour le transfert de fichier (copier un fichier complet d'un système à un autre).

Comme TELNET, FTP a été conçu dés le début pour fonctionner entre des machines différentes, exécutant des systèmes d'exploitations différents.

➢ **Serveur Samba**

Samba est un ensemble de programmes qui permettent de connecter à un serveur LINUX, des stations fonctionnant sous des systèmes divers : Windows 3.11, Windows 9x, Windows NT et autres.
Le serveur Linux est en mesure de se conduire comme un serveur de fichiers capables d'offrir les services habituels sur un réseau :

- partage de fichiers et de répertoires.
- partage d'imprimantes.
- respect des comptes utilisateurs.
- gestion des permissions d'accès.

VII- Conclusion

Dans ce chapitre, nous avons défini les principes fondamentaux du réseau Internet, le protocole de communication les plus répandus TCP/IP, et cité quelques services de base.
Pour une meilleur compréhension de tout ces aspects théoriques, il fallait concrétiser un Intranet (qui set basé sur les même protocoles et technologies) de manière à être confronté à certains aspects non percevable théoriquement.
Le chapitre suivant traite essentiellement sur l'implémentation d'un réseau Intranet avec l'utilisation de systèmes d'exploitation Linux.

Introduction

L'intranet est un réseau interne qui applique la technologie Internet à la communication interne de l'entreprise tout en se basant sur la technologie des réseaux locaux, afin d'améliorer la productivité et le transfert des informations. L'intranet, c'est donc un réseau Internet ramené à l'échelle de l'entreprise. Il est basé sur les mêmes outils et technologies qu'Internet. Cependant il existe deux différences fondamentales entre l'Internet et l'Intranet et qui sont :

- ✓ Les utilisateurs de l'Intranet sont constitués exclusivement d'une population connue et identifiable, ce n'est pas le cas pour Internet.

- ✓ Le réseau Intranet est administré et contrôlé par une seule entité qui permet une grande maîtrise des infrastructures réseaux et du débit disponible ce qui est pratiquement impossible sur l'Internet.

Les bénéfices apportés sont :

- ✓ Comme tout réseau informatique, l'Intranet permet le partage des ressources informatiques coûteuses (programmes, applications,....), et réduire ainsi le coût d'investissement.
- ✓ L'Intranet met de grandes quantités d'informations à la disposition d'un grand nombre d'utilisateurs.
- ✓ L'Intranet permet de réduire le temps d'édition et de révision des documents, il améliore la communication.
- ✓ L'Intranet réduit de façon spectaculaire les coûts papier, etc.

Un Intranet est tout simplement l'application à un réseau d'entreprise (donc à usage restreint et d'envergure parfaitement délimitée) des standards et outils utilisés sur Internet.

L'Intranet repose (comme l'Internet) sur architecture client/serveur, et utilise (dans notre cas) les protocoles de communication TCP/IP pour les transmissions des données.

I- Les services d'Intranet

Les services présents dans un réseau Intranet sont les mêmes de celles de l'Internet et ils fonctionnent d'une manière quasi identique. La seule différence est due à l'échelle du réseau.

Les éléments qui font la différence entre l'Internet et l'Intranet cité au-dessus, sont les seuls paramètres qui vont différencier entre le fichier de configuration d'un service qui va tourner sur l'Internet, et celui qui va tourner sur un Intranet, par conséquence leur comportements va être différents.

Dans le présent chapitre, on développe l'implémentation d'un réseau Intranet en se basant sur les produits opensource suivants :

- ✓ DHCP : V 2.0pl3-3.
- ✓ BIND8 : V 8.2.3-34.
- ✓ APACHE : V 1.3.12-93.
- ✓ SENDMAIL : V 8.10.2-24.
- ✓ proFTP : V 1.2.6.
- ✓ SAMBA : V 2.0.7-49.

I- Introduction

DHCP signifie Dynamic Host Configuration Protocol[3]. Il s'agit d'un protocole qui permet à un ordinateur qui se connecte sur un réseau d'obtenir dynamiquement (c'est-à-dire sans intervention particulière) sa configuration (principalement, sa configuration réseau). On n'a qu'à spécifier à l'ordinateur de se trouver une adresse IP tout seul par DHCP. Le but principal étant la simplification de l'administration d'un réseau.

Le protocole DHCP sert principalement à distribuer des adresses IP sur un réseau, mais il a été conçu au départ comme complément au protocole BOOTP (Bootstrup Protocol) qui est utilisé par exemple lorsque l'on installe une machine à travers un réseau (BOOTP est utilisé en étroite collaboration avec un serveur TFTP sur lequel le client va trouver les fichiers à charger et à copier sur le disque dur). Un serveur DHCP peut renvoyer des paramètres BOOTP ou de configuration propres à un hôte donné.

II- Théorie

II.1- Fonctionnement du protocole DHCP

Il faut dans un premier temps un serveur DHCP qui distribue des adresses IP. Cette machine va servir de base pour toutes les requêtes DHCP, aussi elle doit avoir une adresse IP fixe. Dans un réseau, on peut donc n'avoir qu'une seule machine avec adresse IP fixe, le serveur DHCP.

Le mécanisme de base de la communication est BOOTP (avec trame UDP). Quand une machine est démarrée, elle n'a aucune information sur sa configuration réseau, et surtout, l'utilisateur ne doit rien faire de particulier pour trouver une adresse IP. Pour faire ça, la technique utilisée est le broadcast (diffusion) pour trouver et dialoguer avec un serveur DHCP, la machine va simplement émettre un paquet spécial de broadcast (broadcast sur 255.255.255.255 avec d'autres informations comme le type de requête, les ports de connexion...) sur le réseau local. Lorsque le serveur DHCP recevra le paquet de broadcast, il renverra un autre paquet de broadcast (on n'oublie pas que le client n'a pas forcement son adresse IP et que donc il n'est pas joignable directement) contenant toutes les informations requises pour le client.
On pourrait croire qu'un seul paquet peut suffire à la bonne marche du protocole. En fait, il existe plusieurs types de paquets DHCP susceptibles d'être émis soit par le client pour le ou les serveurs, soit par le serveur vers un client :

a- **DHCPDISCOVER**
- Pour localiser les serveurs DHCP disponibles.

b- **DHCPOFFER**
- Réponse du serveur à un paquet DHCPDISCOVER, qui contient les premiers paramètres.

c- **DHCPREQUEST**
- Requête diverse du client pour par exemple prolonger son bail.

d- **DHCPACK**
- Réponse du serveur qui contient des paramètres et l'adresse IP du client.

e- **DHCPNAK**
- Réponse du serveur pour signaler au client que son bail est échu ou si le client annonce une mauvaise configuration réseau.

f- **DHCPDECLINE**
- Le client annonce au serveur que l'adresse est déjà utilisée.

g- **DHCPRELEASE**
- Le client libère son adresse IP.

h- **DHCPINFORM**
- Le client demande des paramètres locaux, il a déjà son adresse IP.

Le premier paquet émis par le client est un paquet de type DHCPDISCOVER. Le serveur répond par un paquet DHCPOFFER, en particulier pour soumettre une adresse IP au client. Le client établit sa configuration, puis fait un DHCPREQUEST pour valider son adresse IP (requête en broadcast car DHCPOFFER ne contient par son adresse IP). Le serveur répond simplement par un DHCPACK avec l'adresse IP pour confirmation de l'attribution. Normalement, c'est suffisant pour qu'un client obtienne une configuration réseau efficace, mais cela peut être plus ou moins long selon que le client accepte ou non l'adresse IP...

II.2- Les baux

Pour des raisons d'optimisation des ressources réseau, les adresses IP sont délivrées avec une date de début et une date de fin de validité. C'est ce qu'on appel un bail. Un client qui voit son bail arriver à terme peut demander au serveur une prolongation du bail par un DHCPREQUEST. De même, lorsque le serveur verra un bail arrivé à terme, il émettra un paquet DHCPNAK pour demander au client s'il veut prolonger son bail. Si le serveur ne reçoit pas de réponse valide, il rend disponible l'adresse IP.

C'est toute la subtilité du DHCP : on peut optimiser l'attribution des adresses IP en jouant sur la durée des baux. Le problème est là : si aucune adresse n'est libérée au bout d'un certain temps, plus aucune requête DHCP ne pourra être satisfaite, faute d'adresses à distribuer.

Sur un réseau où beaucoup d'ordinateurs se branchent et se débranchent souvent (réseau d'école ou de locaux commerciaux par exemple), il est intéressant de proposer des baux de courte durée. A l'inverse, sur un réseau constitué en majorité de machines fixes, très peu souvent rebootées, des baux de longues durées suffisent. On n'oublie pas que le DHCP marche principalement par broadcast, et que cela peut bloquer de la bande passante sur des petits réseaux fortement sollicités.

III- Configuration

III.1- Préparation

Une fois le serveur installé, on saisie ifconfig -a.
On doit obtenir :

```
eth0    Link encap:10Mbps Ethernet  HWaddr 00:48:54:6F:C4:62
        inet addr:192.65.146.119  Bcast: 192.65.146.225  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:2875542 errors:0 dropped:0 overruns:0
        TX packets:218647 errors:0 dropped:0 overruns:0
        Interrupt:11 Base address:0x210
```

Si le mot MULTICAST n'apparaît pas, on doit recompiler notre noyau avec le support multicast. Sur la plupart des systèmes, ça ne devrait pas être nécessaire.
Ensuite, on ajoute une route pour 255.255.255.255. D'après le fichier README du DHCPd : Pour que dhcpd fonctionne correctement avec des clients DHCP pointilleux (comme celui de Windows 95), il doit pouvoir envoyer des paquets IP avec une adresse de destination de 255.255.255.255. Malheureusement, Linux insiste pour changer 255.255.255.255 en l'adresse de diffusion du réseau (ici 192.5.5.223). Ceci constitue une violation du protocole DHCP et, alors que beaucoup de clients DHCP ignorent ce problème, certains (par exemple tous ceux de Microsoft) le remarquent. Les clients qui ont ce problème sembleront ne pas voir les messages DHCPOFFER du serveur.
Pour cela, on tape :

route add -host 255.255.255.255 dev eth0

Si on voit un message d'erreur :

"255.255.255.255: Unknown host"

on essaye d'ajouter la ligne suivante à notre fichier /etc/hosts :

255.255.255.255 tout-le-monde

Ensuite, on essaye :

route add -host tout-le-monde dev eth0

ou

route add 255.255.255.0 dev eth0

eth0 désigne bien sûr l'interface réseau que on utilise. Si on en utilise une autre, faites les changements nécessaires.

La plupart du temps, on voudrait assigner des adresses IP aléatoirement. Cela peut se faire de la façon suivante [10]:

```
# Exemple de /etc/dhcpd.conf
# (On ajoute nos commentaires ici)
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.65.146.255;
option routers 192.65.146.1;
option domain-name-servers 192.65.146.119;
option domain-name "elec";
subnet 192.65.146.0 netmask 255.255.255.0 {
range 192.65.146.10 192.65.146.100;
range 192.65.146.150 192.65.146.200;
}
```

III.2- le serveur

Le serveur DHCP distribuera aux clients des adresses IP dans les intervalles 192.65.146.10-192.65.146.100 et 192.65.146.150-192.65.146.200. L'attribution de l'adresse IP se fera pour une durée de 600 secondes si le client ne demande pas une durée différente, la durée

maximale du bail étant de 7200 secondes. Le serveur va également "conseiller" au client d'utiliser le masque de sous-réseau 255.255.255.0, l'adresse de diffusion 192.65.146.255, 192.65.146.1 comme routeur/passerelle et 192.65.146.119 comme serveur DNS.
Si on doit spécifier un serveur WINS pour nos clients Windows, on doit inclure l'option netbios-name-servers :

```
option netbios-name-servers 192.65.146.40;
```

On peut aussi assigner des adresses IP spécifiques en utilisant l'adresse ethernet du client

```
host Windows {
hardware ethernet 08:00:2b:4c:59:23;
fixed-address 192.65.146.222;
}
```

Cela assignera l'adresse IP 192.168.1.222 à un client dont l'adresse ethernet est 08:00:2b:4c:59:23.

On peut aussi mélanger ces différentes possibilités, par exemple si on veut donner des adresses IP statiques à certains clients (les serveurs, par exemple) et des adresses IP dynamiques à d'autres (par exemple les ordinateurs portables de certains utilisateurs). Il existe de nombreuses autres options: adresse du serveur NIS, du serveur d'horloge... Si on a besoin de ces options, on lit la page de manuel de dhcpd.conf.

III.3- Démarrage du serveur

Il ne reste qu'une chose à faire avant de démarrer le serveur. La plupart des installations de DHCPd ne créent pas par défaut de fichier dhcpd.leases. Ce fichier est utilisé par DHCPd pour stocker des informations à propos des attributions en cours de validité. Il est en format texte brut, donc on peut lire pendant le fonctionnement de DHCPd. Pour le créer, :

```
# touch /var/state/dhcp/dhcpd.leases
```

Sur la ligne de commandes. Cela va créer un fichier vide (de taille nulle). Certaines vieilles versions de dhcpd 2.0 plaçaient ce fichier en /etc/dhcpd.leases. On n'a pas à y changer quoi que ce soit; c'est dhcpd qui le manipulera lui-même. Si on voit un message d'erreur disant que le fichier n'existe pas, on ignore le et on passe à l'étape suivante.

On peut maintenant invoquer le serveur DHCP. On se contente de taper (ou de rajouter dans les scripts de démarrage) :

```
/usr/sbin/dhcpd
```

Cette commande invoquera dhcpd sur l'interface eth0. Pour l'utiliser sur une autre interface, on le précise simplement sur la ligne de commande, par exemple :

```
/usr/sbin/dhcpd eth0
```

Pour vérifier que tout fonctionne correctement, on doit d'abord activer le mode de débogage et mettre le serveur en avant-plan. On peut le faire en tapant :

```
/usr/sbin/dhcpd -d -f
```

Ensuite, on active un de nos clients et on regarde la console de notre serveur. On voit apparaître un certain nombre de messages de débogage. Si tout se passe bien, on a terminé.

On quitte dhcpd et on le relance sans les options -d -f. Si on veut qu'il soit lancé au démarrage, on rajoute dhcpd par exemple au fichier /etc/rc.d/rc.local.

IV- conclusion

On peut lancer DHCP sur une machine avec plusieurs cartes Ethernet ,DHCP permet de distinguer les interfaces physiques, si bien que la dernière version de DHCPd devrait fontionner avec 2 interfaces Ethernet à la fois sous Linux. Cette possibilité est toutefois en phase de développement bêta.

I- Introduction :

Au début de l'histoire d'Internet, la correspondance entre le nom (les noms s'il y a des synonymes ou ``alias'') et l'adresse (il peut y en avoir plusieurs) d'une machine est placée dans le fichier /etc/hosts, présent sur toutes les machines dotées d'une pile Arpa :

```
# Host Database
# This file should contain the addresses and aliases
# for local hosts that share this file.
# In the presence of the domain name service or NIS, this file may
# not be consulted at all; see /etc/host.conf for the resolution order.
#
#
127.0.0.1          localhost localhost.my.domain
#
# Imaginary network.
#10.0.0.2          myname.my.domain myname
#10.0.0.3          myfriend.my.domain myfriend
#
# According to RFC 1918, you can use the following IP networks for
# private nets which will never be connected to the Internet:
#
#     10.0.0.0      -  10.255.255.255
#     172.16.0.0    -  172.31.255.255
#     192.168.0.0   -  192.168.255.255
#
# In case you want to be able to connect to the Internet, you need
# real official assigned numbers.  PLEASE PLEASE PLEASE do not try
# to invent your own network numbers but instead get one from your
# network provider (if any) or from the Internet Registry (ftp to
# rs.internic.net, directory `/templates').
#
```

Au début des années 1980 c'est le NIC (Network Information Centre) qui gère la mise à jour continuelle de cette table (HOSTS.TXT), avec les inconvénients suivants :

- Absence de structure claire dans le nommage d'où de nombreux conflits entre les noms des stations.
- Centralisation des mises à jour qui entraîne :
    1. Une lourdeur du processus de mise à jour
    2. Un trafic réseau (ftp) en forte croissance (N^2 si N est le nombre de machines dans cette table) et qui devient rapidement ingérable au vu des bandes passantes de l'époque, et surtout jamais à jour compte tenu des changements continuels.

Au milieu des années 1980 (1986) la liste officielle des hôtes contient 3100 noms et 6500 alias. La forte croissance du nombre des machines, a rendu obsolète cette approche.

Le service DNS [3] est un système organisé de manière hiérarchique, sous forme d'arbre. La racine est désignée par ``.'' et s'appelle ``la racine''. En dessous de . se trouvent un certain nombre

de TLD (*Top Level Domains*); les plus connus sont ORG, COM, EDU, NET et FR, mais il y en a beaucoup d'autres. Tout comme un arbre, il a une racine avec des branches qui en partent.
On reconnaît dans le DNS un arbre de recherche, avec des noeuds, des arrêtes et des feuilles.

Le service de résolution de nom sous Linux est assuré par un programme appelé named. Il fait partie du paquetage ``**Bind 8**" (Berkley Internet Name Domain), géré par Paul Vixie pour l'Internet Software Consortium.

II- Théorie

II.1- Système hiérarchisé de nommage

II.1.1- Domaine et zone

Le réseau peut être considéré comme une hiérarchie de domaines. L'espace des noms y est organisé en tenant compte des limites administratives ou organisationnelles. Chaque nœud, appelé un domaine, est baptisé par une chaîne de caractères et le nom de ce domaine est la concaténation de toutes les étiquettes de nœuds lues depuis la racine, donc de droite à gauche. Par exemple :

.dz	Domaine principale dz .
.edu.dz	Domaine edu.dz sous domaine de dz .
.enp.edu.dz	Domaine enp.edu.dz, sous domaine de edu.dz .

Par construction, tout nœud est un domaine, même s'il est terminal, c'est à dire n'a pas de sous domaine. Un sous domaine est un domaine à part entière et, exceptée la racine, tout domaine est un sous domaine d'un autre.

Bien que le serveur de noms, ``**D**omain **N**ame **S**erver" fasse référence explicitement au concept de domaine, pour bien comprendre la configuration d'un tel service il faut également comprendre la notion de ``**zone**".

Une zone est un point de délégation dans l'arbre DNS, autrement dit une zone concerne un sous arbre du DNS dont l'administration lui est propre. Ce sous arbre peut comprendre plusieurs niveaux, c'est à dire plusieurs sous domaines. Une zone peut être confondue avec le domaine dans les cas les plus simples.

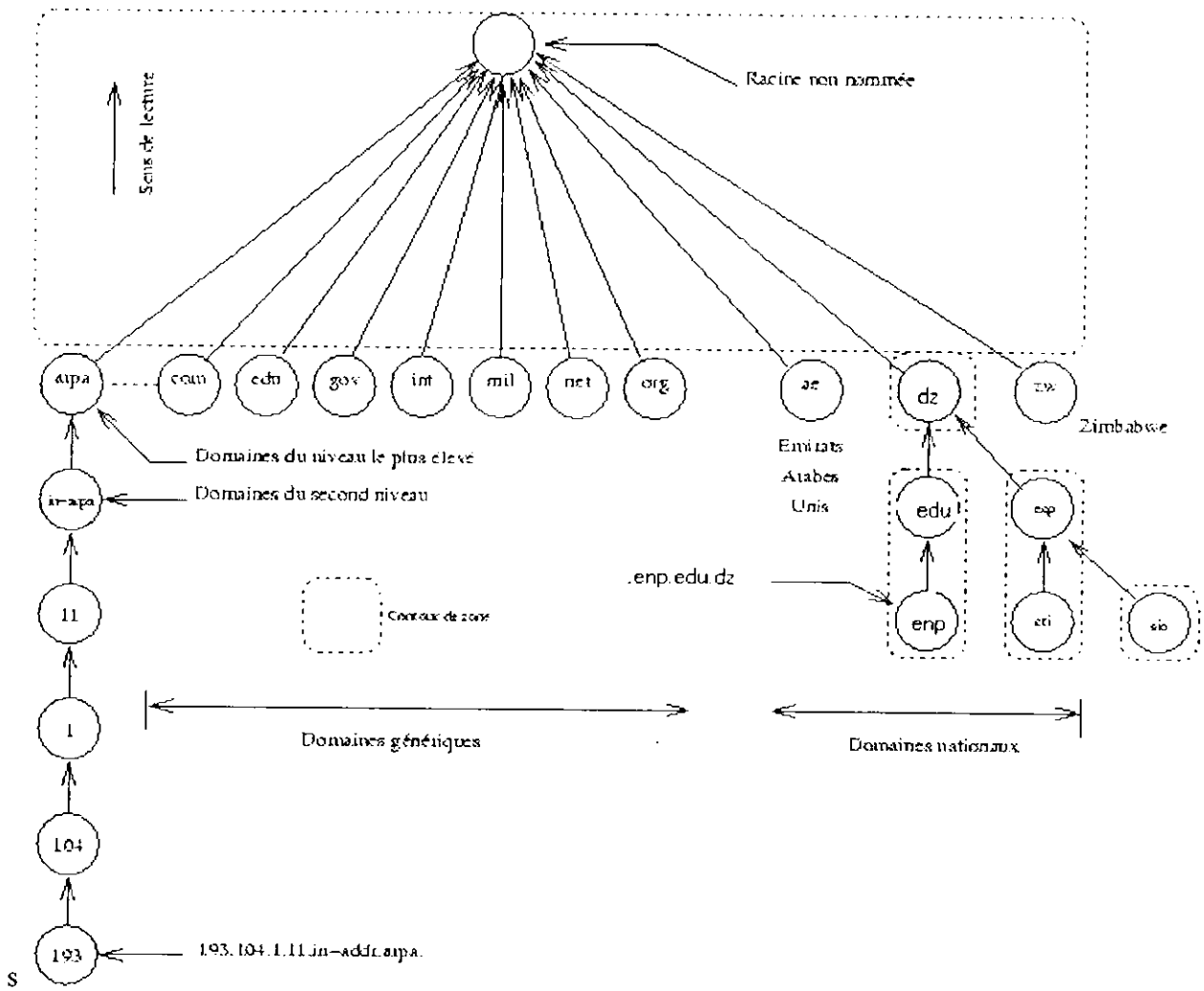
Dans les exemples ci-dessus, on peut parler de zone enp.edu.dz puisque celle-ci est gérée de manière autonome par rapport à la zone edu.dz.
Le serveur de noms est concerné par les ``zones". Ses fichiers de configuration précisent la portée de la zone et non du domaine.

Chaque zone doit avoir un serveur principal (``primary server") qui détient ses informations d'un fichier configuré manuellement ou semi manuellement (DNS dynamique). Plusieurs serveurs secondaires (``secondary server") reçoivent une copie de la zone via le réseau et pour assurer la continuité du service.

Le fait d'administrer une zone est le résultat d'une délégation de pouvoirs de l'administrateur de la zone parent et se concrétise par la responsabilité de configurer et d'entretenir le champ SOA (``start of authority") de cette zone.

II.1.2- Hiérarchie des domaines

Cette organisation du nommage pallie aux inconvénients de la première méthode :
- Le NIC gère le plus haut niveau de la hiérarchie, appellé aussi celui des ``top levels".
- Les instances régionales du NIC gèrent les domaines qui leur sont dévolus. Par exemple le ''NIC Canada'' gère le contenu de la zone « **.ca** » . Le nommage sur deux lettres des pays est issu de la norme ISO 3166.
- Chaque administrateur de domaine (universités, entreprises, associations, entités administratives,...) est en charge de son domaine et des sous domaines qu'il crée. Sa responsabilité est nomminative vis à vis du NIC. On dit aussi qu'il a l'autorité sur son domaine (``authoritative for the domain") .



S

Fig. 1 Hiérarchie des domaines

- Les éventuels conflits de nommage sont à la charge des administrateurs de domaine. Du fait de la hiérarchisation, des machines de même nom peuvent se trouver dans des domaines différents sans que cela pose le moindre problème.

- Chaque domaine entretient une base de données sur le nommage de ses machines. Celle-ci est mise à disposition de tous les utilisateurs de l'Internet.
- Chaque site raccordé de manière permanente procède de cette manière, ainsi il n'y a pas une base de données pour l'Internet mais un ensemble structuré de bases de données formant une gigantesque **base de données distribuée**.

II.2- Fonctionnement du DNS

II.2.1- Le " Resolver "

Le ``resolver" désigne un ensemble de fonctions placées dans la bibliothèque standard qui font l'interface entre les applications et les serveurs de noms. Par construction les fonctions du ``resolver" sont compilées avec l'application qui les utilise .

Le ``resolver" applique la stratégie locale de recherche, définie par l'administrateur de la machine, pour résoudre les requêtes de résolution de nom. Pour cela il s'appuie sur son fichier de configuration **/etc/resolv.conf** et sur la stratégie locale d'emploi des possibilités (serveur de noms, fichier **/etc/hosts**, NIS,...).

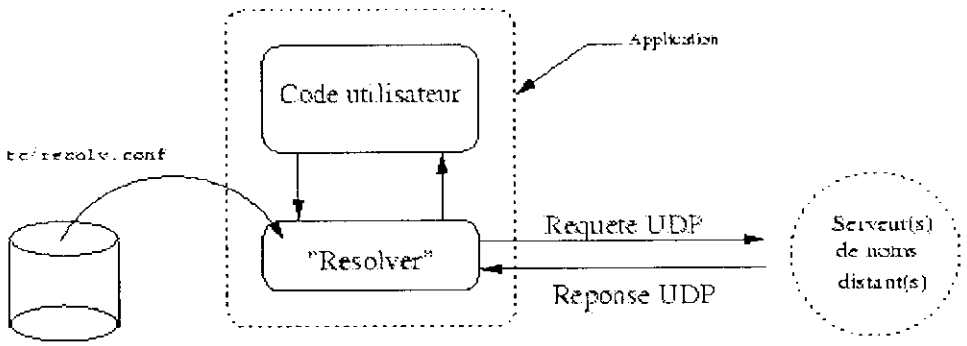


Fig. *2* La stratégie du Resolver.

Le fichier **/etc/resolv.conf** doit préciser au moins le domaine local assorti de quelques directives optionnelles . Exemple de contenu d'un tel fichier :

```
Domain elec
nameserver 192.65.146.119
nameserver 192.65.146.2
```

II.2.2- Stratégie de fonctionnement

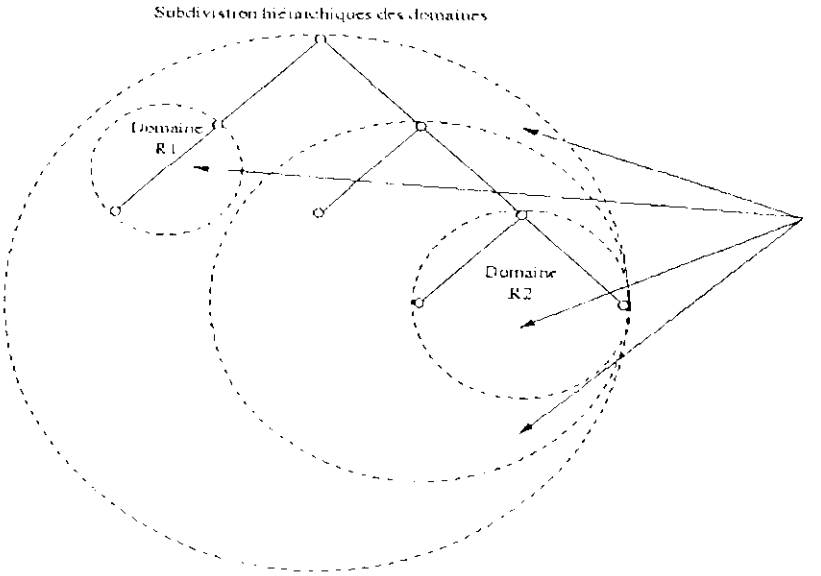


Fig. 3 Subdivision hiérarchiques des domaines.

Cette figure illustre le fait que chaque serveur de noms a la maîtrise de ses données mais doit interroger ses voisins dès qu'une requête concerne une zone sur laquelle il n'a pas l'autorité de nommage.

Un hôte du domaine ``R2" qui veut résoudre une adresse du domaine ``R1" doit nécessairement passer par un serveur intermédiaire pour obtenir l'information. Cette démarche s'appuie sur plusieurs stratégies possibles, résumées dans les paragraphes suivants :

a- Interrogation locale :

La figure ci-dessous illustre la recherche d'un nom dans le domaine local.

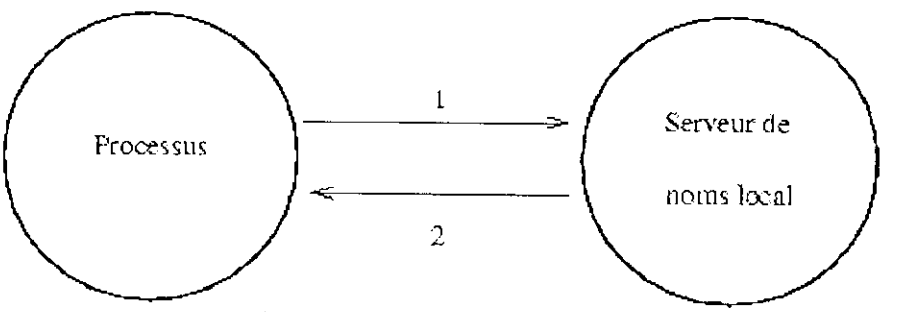


Fig.4 Interrogation locale.

Un processus (``browser" http par exemple) recherche l'adresse d'un nom de serveur. Sur les machines Unix cela se traduit par l'appel à la fonction gethostbyname. Cette fonction est systématiquement présente dans la bibliothèque standard et donc est accessible potentiellement à tout exécutable lors d'une compilation.

La fonction gethostbyname fait systématiquement appel au ``resolver'' déjà cité. C'est donc toujours en passant par ce mécanisme que les processus accèdent à l'espace de noms. Le ``resolver'' utilise une stratégie générale à la machine (donc qui a été choisie par son administrateur) pour résoudre de telles requêtes :

1. Interrogation du serveur de noms (DNS) si présent
2. Utilisation des services type ``YP'' (NIS) si configurés
3. Utilisation du fichier /etc/hosts

Enfin quelle que soit les architecture logicielle le ``resolver'' est configuré à l'aide du fichier /etc/resolv.conf.

Sur la *figure* 4 :

1. Le processus demande l'adresse IP d'un serveur. Le ``resolver'' envoie la demande au serveur local.
2. Le serveur local reçoit la demande, parce qu'il a l'autorité sur le domaine demandé (le sien), il répond directement au ``resolver''.

b- Interrogation distante :

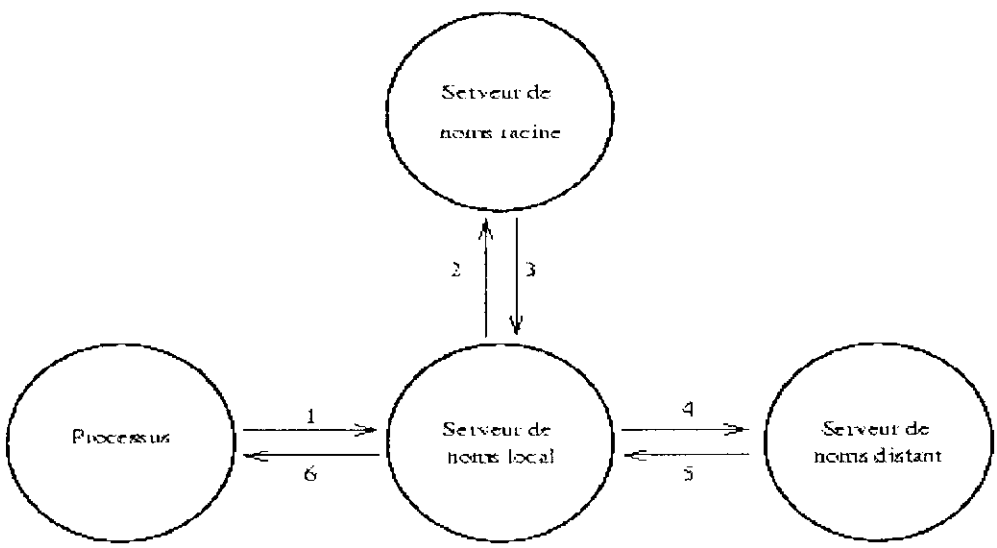


fig. 5 Interrogation distante.

1. Un processus demande l'adresse IP d'une machine. Le ``resolver'' envoie sa requête au serveur local.
2. Le serveur local reçoit la requête et dans ce deuxième cas il ne peut pas répondre directement car la machine n'est pas dans sa zone d'autorité.

Pour lever l'indétermination il interroge alors un serveur racine pour avoir l'adresse d'un serveur qui a l'autorité sur la zone demandée par le processus.

3. Le serveur racine renvoie l'adresse d'un serveur qui a officiellement l'autorité sur la zone
4. Le serveur local interroge ce nouveau serveur distant.
5. Le serveur distant renvoie l'information demandée au serveur local.
6. Le serveur local retourne la réponse au ``resolver''

Remarques :

- Un mécanisme de cache accélère le processus ci-dessus. Si un processus redemande la même machine distante on se retrouve dans le cas d'une interrogation ``locale'', du moins pendant la durée de validité des données.
- Si un processus demande une machine du même domaine que la précédente (mais pas du même nom !), les étapes 2 et 3 deviennent inutiles et le serveur local interroge alors directement le serveur distant.

La durée de vie de l'adresse du serveur distant est elle aussi assujettie d'une date limite d'utilisation.

- Dans le cas général les serveurs racines ne voient pas plus de 1 ou deux niveaux en dessous. Ainsi, si un processus demande A.B.C.D.net :
    1. Le serveur racine donne l'adresse d'un serveur pour D
    2. Le serveur pour D donnera peut être l'adresse d'un serveur pour C et ainsi de suite jouera le rôle de serveur racine de l'étape précédente.

c- Interrogation par "procuration" :

Le processus de recherche décrit au paragraphe précédent ne convient pas dans tous les cas, notamment vis à vis des deux critères suivants :

1. Sécurité d'un domaine
2. Conservation de la bande passante

La *figure 5* montre le serveur local qui interroge directement les serveurs distants, cette démarche pose des problèmes de sécurité dans le cas d'un domaine au sein duquel seuls un ou deux serveurs sont autorisés à le faire.

Le trafic destiné au serveur de noms peut consommer une partie non négligeable de la bande passante, c'est pourquoi il peut être stratégique de concentrer les demandes vers un seul serveur régional et donc de bénéficier au maximum de l'effet de cache décrit précédemment.

II.2.3- Hiérarchie de serveurs

Si tous les serveurs de noms traitent de données d'un format identiques, leur position dans l'arborescence leur confère un statut qui se nomme :

Serveur racine :

(``root name server'') Un serveur ayant autorité sur la racine de l'espace de nommage. Actuellement il y a 13 serveurs de ce type :

[A...M].ROOT-SERVERS.NET

Serveur primaire :

(``primary server'') Un serveur de noms qui a l'autorité pour un ou plusieurs domaines. Il lit ses données dans un fichier stocké sur disque dur, à son démarrage.

L'administrateur du (des) domaine(s) met à jour les informations des domaines concernés depuis cette machine.

Serveur secondaire :

(``secondary server'') Dans le cas d'une panne ou d'un engorgement du serveur primaire, les serveurs secondaires reçoivent en prévision une copie de la base de données.

- Stratégiquement ils sont en dehors du domaine, dans le meilleur des cas. Il peut y avoir plusieurs serveurs secondaires.
- Au démarrage ils reçoivent les informations du serveur primaire, ou du disque dur s'ils ont eu le temps de les y stocker au précédent arrêt du serveur, et si elles sont encore valides.

II.2.4- Conversion d'adresses IP en noms

On dit aussi questions inverses (``inverse queries''). Il faut reconsidérer la *figure 1*. À gauche de la figure on distingue un domaine un peu particulier ``arpa''.

Toutes les adresses sont exprimées dans "in-addr.arpa " , et du fait de la lecture inverse de l'arbre, les adresses IP sont exprimées en ``mirroir'' de la réalité.

Par exemple pour l'enp :

146.65.192.in-addr.arpa (Classe C 192.65.146)

Chaque administrateur de domaine est aussi en charge de l'administration de sa ``zone reverse'', portion du domaine ``arpa''.

III- Configuration :

Le fichier /etc/named.conf. se présente sous la forme d'une succession de paragraphes, le premier servant à configurer le chemin du répertoire de base de configuration de BIND[10]. Les autres définissent chacun une zone. Chaque zone possède ses propres paramètres, ainsi que son propre fichier de configuration dans lequel on définis les adresses des machines contenues dans cette zone (une zone définie généralement un domain).

Nous allons donc d'abord configurer le premier paragraphe comme ceci:

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

Nous pouvons maintenant configurer les différentes zones.

- La première zone à configurer est la zone "root", qui est en fait un pointeur vers un fichier contenant les adresses des serveurs DNS qui régissent l'univers d'Internet. Ces serveurs sont ceux qui sont utilisés pour maintenir à jour les tables de conversion adresse IP vers nom, et inversement. Le deuxième paragraphe est donc le suivant:

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

- La deuxième zone est la zone locale, celle qui permet à la machine de convertir les adresses quand elle parle d'elle-même à elle-même. Voici:

```
zone "0.0.127.IN-ADDR.ARPA" {
  type master;
  file "127.0.0.zone";
};
```

On note bien l'absence de ``." à la fin des noms de domaine de ce fichier. Elle signifie que nous allons définir la zone 0.0.127.in-addr.arpa, que nous sommes son serveur principal et que tout est stocké dans un fichier appelé 127.0.0.zone
Comme on le voit, la ligne 3 indique un fichier nommé 127.0.0.zone qu'il va falloir créer dans le répertoire /var/named. Ce fichier contiendra ceci:

```
@       IN      SOA    elec. root.elec. (
                1       ; Serial
                8H      ; Refresh
                2H      ; retry
                1W      ; Expire
                1D)     ; Minimum TTL
                NS      elec.
1               PTR     localhost.
```

- La zone de conversion nom vers adresse IP

Revenons maintenant au fichier /etc/named.conf. Définissions maintenant la zone qui va correspondre à notre domaine elec:

```
zone "elec" {
        notify no;
        type master;
        file "elec";
};
```

La ligne notify no est très importante, c'est elle qui dira si oui ou non on souhaite informer les serveurs DNS root d'une éventuelle mise à jour dans leurs tables de correspondance adresse (IP - nom de machine). Si on exécute notre serveur DNS sur une machine qui est reliée à Internet par modem par exemple, il va de soit que ce n'est pas utile (le domaine qu'on va crée sera fictif). Là encore, il faut créer un fichier /etc/named/elecr:

```
@       IN      SOA             labo11.elec. hostmaster.elec. (
                                1               ; serial,
                                8H              ; refresh, seconds
                                2H              ; retry, seconds
                                1W              ; expire; seconds
                                1D)             ; minimum, seconds
;
                NS              labo11.elec.
                MX      10      labo11.elec.    ; Primary Mail Exchanger
1               PTR             localhost.
localhost       A               127.0.0.1
Labo11.elec.    A               192.65.146.119
Windows.elec.   A               192.65.146.40
```

Ce ``fichier de zone" (``zone file"), servira donc pour la conversion des noms de machines en adresses IP.Il contient des ``resource records" (RRs) : un SOA RR, un NS RR et un PTR RR. SOA est l'abréviation de ``Start Of Authority" (Origine de l'Autorité). (A=adresse),Le ``@" est une notation spéciale qui désigne l'origine.NS est le ``resource records" pour le serveur de noms (NS = Name Server), Il n'y a pas de @ au début de la ligne, il est implicite, puisque la ligne d'avant commence avec un ``@". Donc, la ligne NS peut aussi s'écrire comme suit :

```
0.0.127.in-addr.arpa.    IN       NS       labo11.elec
```

Elle dit au service DNS quelle machine est le serveur de noms pour le domaine (0.0.127.in-addr.arpa), c'est labo11.elec. Labo11 est le nom habituel des serveurs de noms, tout comme www. pour les serveurs Web, mais c'est simplement une habitude, on peut choisir n'importe quel nom.
Et finalement le PTR dit que l'adresse 1 dans le sous réseau 0.0.127.in-addr.arpa, donc 127.0.0.1 est appelé localhost.
Le champ SOA est le préambule de *tous* les fichiers de zone, et il doit y en avoir exactement un dans chaque fichier de zone. Ce champ SOA décrit la zone, son origine (une machine appelée labo11.elec), qui est responsable de son contenu, de quelle version du fichier de zone il s'agit (serial : 1), et quelques autres paramètres pour le cache et les serveurs DNS secondaires.
Si la machine possède déjà une adresse IP, et qu'elle n'est pas de ce type, on doit la modifier. On édite pour cela le fichier /etc/rc.conf, et on modifie la ligne contenant

```
ifconfig_interface="inet 192.65.146.119 netmask 255.255.255.0"
```

En remplaçant adresse_IP par une adresse du type décrit plus haut ici ce n'est pas le cas. Il faut que cette adresse soit en concordance avec le fichier de zone du serveur DNS. De même pour la ligne

hostname="labo11.elec".

Revenons à ce fichier de zone. La ligne contenant MX indique le nom du serveur de mail principal du domaine. Les lignes à partir de localhost sont les correspondances noms de machines - adresses IP. Une fois ajoutées tous les noms des machines dans ce fichier, nous pouvons retourner au fichier /etc/named . conf

- Zone de conversion adresse IP vers nom

Il ne reste plus qu'une zone à ajouter, la zone permettant de convertir les adresses IP en noms de machines:

```
zone "0.168.192.IN-ADDR.ARPA" {
        notify no;
        type master;
        file "192.168.0";
};
```

On crée le fichier /etc/named/192.65.146 :

```
@       IN      SOA     labo11.elec. hostmaster.elec. (
                        1998021151      ; Serial, todays date + todays serial
                        8H              ; Refresh
                        2H              ; Retry
                        1W              ; Expire
                        1D)             ; Minimum TTL
                NS      labo11.elec.

119             PTR     labo11.elec.
40              PTR     windows.elec.
```

Ce fichier suit à peu près la même logique que le fichier de zone elec. pour terminer la configuration ,il faut modifies les trois fichiers restons resolv.conf, host.conf et hosts

- **Les fichiers resolv.conf, host.conf et hosts** :

Il faut maintenant dire aux machines de l'intranet où se trouve le serveur DNS pour pouvoir l'interroger correctement, lors d'une requête nslookup par exemple.
On édite le fichier /etc/resolv.conf et on met les deux lignes:

```
domain elec
nameserver 192.65.146.119
```

Puis le fichier /etc/host.conf, qui va servir à dire comment traiter une requête de conversion d'adresse, à savoir soit en interrogeant d'abord le contenu du fichier /etc/hosts, soit en interrogeant le serveur DNS, soit en interrogeant le serveur NIS. Ce fichier décrit la stratégie ou dans quel ordre traiter la requête. On introduit :

```
order hosts, bind
```

ce qui signifie que d'abord essayer /etc/hosts ,ensuite le serveur DNS.

Pour finir, on regarde ce que contient le fichier /etc/hosts de nos machines. Il ne faut pas qu'il y ait de contradiction avec le serveur DNS, on met donc simplement l'adresse de bouclage dans ce fichier:

```
127.0.0.1 localhost
```

On sauvegarde la configuration de named.conf et des autres fichiers. Notre serveur DNS est maintenant configuré.

III.1- Démarrage le démon de DNS

Pour le bon fonctionnement de dns, on doit démarrer le démon : named, selon l'une des deux méthodes suivantes :

a- Démarrage manuel

En tant que root, il suffit de taper la commande suivante :

```
# /sbin/init.d/named start
```

Dns est alors prêt à accepter des connexions.

b- Démarrage au lancement du système:

Pour que le serveur soit chargé au démarrage de la machine, il faut modifier le fichier /etc/rc.conf en lui ajoutant ceci:

```
# Lancement du serveur DNS
named_enable="YES"
```

Une fois ceci fait, donc au démarrage du système, le serveur de nom est alors prêt à accepter des connexions.

III.2- Teste du serveur DNS

Pour tester si notre serveur DNS fonctionne correctement, nous allons utiliser la commande **nslookup**

On tape donc nslookup au prompt. Il doit nous afficher ceci:

```
Default Server: labo11.elec
Address: 192.65.146.119
>
```

Le prompt > qui s'affiche permet d'entrer des commandes. On Saisis par exemple, pour tester la conversion nom de machine vers adresse IP: windows.elec. Il doit nous afficher:

```
> windows.elec
Server: labo11.elec
Address: 192.65.146.119

Name: windows.elec
Address: 192.65.146.40

>
```

On essaye ensuite la conversion inverse adresse IP vers nom de machine en tapant par exemple: 192.65.146.40. Il doit afficher:

```
> 192.65.146.40
Server: labo11.elec
Address: 192.65.146.119

Name: windows.elec
Address: 192.65.146.40

>
```

On quitte nslookup avec la commande **exit**

IV- Conclusion

A travers ce qui vient d'être décrit, on constate l'intérêt d'un serveur de nom. Si dans un réseau de faible d'envergure, il peut être aisé de gérer un pool d'adresse IP. Ce qui n'est plus le cas dès que le parc devient important.
A l'heure actuelle le protocole IP version 4 est utilisé. Pour des raisons évidentes de manque d'adresses IP du à l'explosion du nombre de machines interconnectées, il sera remplacé par le protocole IP version 6, ce qui va changer totalement la façon de noter les adresses IP.
Par exemple :

IP v4 : 192.65.146.119
IP v6 : 3ffe:0400:0100:f200::/56

I- Introduction

HTTP (Hypert Text Transfer Protocol) est le protocole d'échange de documents entre un client et un serveur[3].
C'est le protocole le plus utilisé en volume depuis 1995, devant FTP, NNTP et SMTP. Il accompagne l'explosion de l'utilisation du système global d'information ``World Wide Web".
Depuis 1990, date d'apparition du "Web", le protocole HTTP évolue doucement mais sûrement. La première version déployée largement a été la 1.0 sortie en Mai 1996. Depuis le début du mois de Janvier 1997 la version 1.1 est apparue, deux fois plus volumineuse pour tenir compte des nouvelles orientations de l'usage du service.
Aujourd'hui ce protocole est tellement répandu que pour bon nombre de nouveaux utilisateurs du réseau, Internet c'est le ``web".

Techniquement, l'usage de ce protocole se conçoit comme une relation entre un client et un serveur. Le client, appelé génériquement un ``browser", ``User Agent" ou encore *butineur de toile*, interroge un serveur connu par son ``url .
Par exemple la chaîne de caractères http://www.enp.edu.dz / est une url; il suffit de la transmettre en tant qu'argument à un quelconque outil d'exploration et celui-ci vous renverra (si tout se passe comme prévu) ce qui est prévu sur le serveur en question pour répondre à cette demande (car il s'agit bien d'une requête comme nous le verrons plus loin dans ce chapitre).
Le serveur, supposé à l'écoute du réseau au moment où la partie cliente l'interroge, utilise un port connu à l'avance. Le port 80 est dédié officiellement au protocole http, mais ce n'est pas une obligation (cette décision est prise à la configuration du serveur). L'url qui désigne un serveur peut contenir dans sa syntaxe le numéro de port sur lequel il faut l'interroger, comme dans :
http://www.enp.edu.dz : 80/

II- Théorie

II.1- Exemple d'échange avec http

Le transport des octets est assuré par TCP; ce qui nous autorise des essais de connexion avec le client tcp à tout faire ; telnet. Bien entendu on pourrait utiliser un ``browser" plus classique, mais celui-ci gérant pour nous le détail des échanges qu'il ne serait plus possible d'examiner par la suite.

$ telnet localhost 80	→ Ce qui est saisie par l'utilisateur
Trying... Connected to localhost. Escape character is '^]'.	} Réponse du serveur :
GET / HTTP/1.1	→ Ce qui est saisie par l'utilisateur

Et la Réponse du serveur et

```
HTTP/1.1 200 OK                                                    } 1

Date: Fri, 01 Mar 2002 10:59:06 GMT
Server: Apache/1.3.12 (Unix) (SuSE/Linux) mod_fastcgi/2.2.2        } 2

        ........

<ADDRESS>Apache/1.3.12 Server at Laboo11.elec Port 80
</ADDRESS>                                                         } 3
........

Connection closed by foreign host.
```

cette réponse du serveur, que l'on peut décomposer en trois parties :

1. Un code de retour (HTTP)
2. Un en-tête MIME
3. Des octets, ici ceux d'une page écrite en HTML.

Notons également la déconnexion à l'initiative du serveur, en fin d'envoi de la page HTML.

II.2- Structure d'un échange

L'exemple qui précède est typique d'un échange entre le client et le serveur : une question du client génère une réponse du serveur, le tout lors d'une connexion TCP qui se termine lors de l'envoi du dernier octet de la réponse (clôture à l'initiative du serveur).
Le serveur ne conserve pas la mémoire des échanges passés, on dit aussi qu'il est sans état, ou ``stateless".
La question et la réponse sont bâties sur un modèle voisin : le message HTTP.

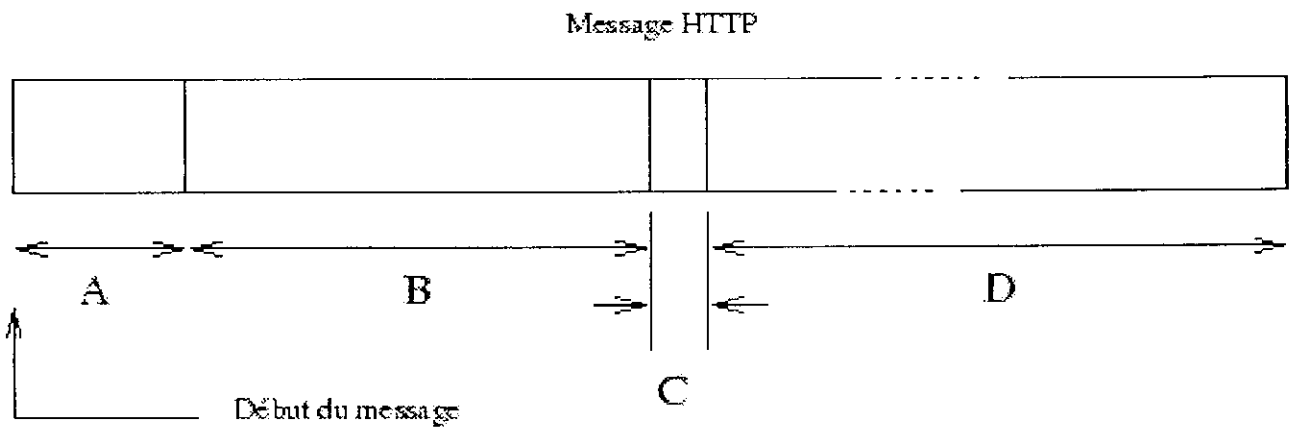


Fig.1 Message HTTP

Les parties A, B et C forment l'en-tête du message et D le corps.

- **A**

La première ligne du message, est soit la question posée (``request-line''), soit le statut de la réponse (``status-line'').

- La question est une ligne terminée par CRLF, elle se compose de trois champs :

Une méthode

à prendre dans GET, HEAD, ou POST.

GET

Plus de 99% des requêtes ont cette méthode, elle retourne l'information demandée dans l'URL (ci-dessous).

HEAD

La même chose que GET, mais seul l'en-tête du serveur est envoyé. Utile pour faire des tests d'accessibilité sans surcharger la bande passante. Utile également pour vérifier de la date de fraîcheur d'un document (information contenue dans l'en-tête).

POST

Cette méthode permet d'envoyer de l'information au serveur, c'est typiquement le contenu d'un formulaire rempli par l'utilisateur.

Une ressource

que l'on désigne par une URL,par exemple http://www.enp.edu.dz / .

La version du protocole

sous forme HTTP-*Numéro de version*. Par exemple HTTP/1.1 !

- La réponse. Cette première ligne n'est que le statut de la réponse, les octets qui la détaillent se trouvent plus loin, dans le corps du message. Trois champs la composent, elle se termine par CRLF :

La version du protocole

Sous forme HTTP-*Numéro de version*, comme pour la question.

Statut

C'est une valeur numérique qui décrit le statut de la réponse. Le premier des trois digits donne le sens général :

1*xx*	N'est pas utilisé ``Futur Use''
2*xx*	Succès, l'action demandée a été comprise et exécutée correctement.
3*xx*	Redirection. La partie cliente doit reprendre l'interrogation, avec une autre formulation.
4*xx*	Erreur coté client. La question comporte une erreur de syntaxe ou ne peut être acceptée.
5*xx*	Erreur coté serveur. Il peut s'agir d'une erreur interne, due à l'OS ou à la ressource devenue non accessible.

Phrase

C'est un petit commentaire (``Reason- Phrase'') qui accompagne le statut, par exemple le statut 200 est suivi généralement du commentaire ``OK'' !

➢ **B**

C'est une partie optionnelle, qui contient des informations à propos du corps du message. Sa syntaxe est proche de celle employée dans le courrier électronique, et pour cause, elle respecte aussi le standard MIME.

Un en-tête de ce type est constitué d'une suite d'une ou plusieurs lignes (la fin d'une ligne est le marqueur CRLF) construite sur le modèle :

Nom_de_champ : *Valeur_du_champ* CRLF

Éventuellement le marqueur de fin de ligne peut être omis pour le séparateur

" ; " .

Exemple d'en-tête MIME :

Date: Mon, 01 Mar 2002 10:59:06 GMT
Server: Apache/1.3.12 (Unix)
Last-Modified: Sat, 10 Nov 2001 16:13:02 GMT
ETag: "1381-8b-3bed520e"
Accept-Ranges: bytes
Content-Length: 79
Connection: close
Content-Type: text/html

Date:

C'est la date à laquelle le message a été envoyé. Bien sûr il s'agit de la date du serveur, il peut exister un décalage incohérent si les machines ne sont pas synchronisées

Server:

Contient une information relative au serveur qui a fabriqué la réponse. En générale la liste des outils logiciels et leur version.

Content-type:

Ce champ permet d'identifier les octets du corps du message.

Content-length:

Désigne la taille (en octets) du corps du message, c'est à dire la partie D de la *figure 1*.

Last-modified:

Il s'agit de la date de dernière modification du fichier demandé, comme l'illustre le résultat de la commande **ll** .

-rw-r--r-- 1 web doc 139 Nov 10 17:13 index.html

ETag:

C'est un identificateur du serveur, constant lors des échanges. C'est un moyen de maintenir le dialogue avec un serveur en particulier, par exemple quand ceux-ci sont en grappe pour équilibrer la charge et assurer la redondance.

➢ **C**

Une ligne vide (CRLF) qui est le marqueur de fin d'en-tête. Il est donc absolument obligatoire qu'elle figure dans le message. Son absence entraîne une incapacité de traitement du message, par le serveur ou par le client.

➢ **D**

Le corps du message. Il est omis dans certains cas, comme une requête avec la méthode GET ou une réponse à une requête avec la méthode HEAD.

C'est dans cette partie du message que l'on trouve par exemple les octets de l'HTML, ou encore ceux d'une image...

Le type des octets est intimement lié à celui annoncé dans l'en-tête, plus précisément dans le champ Content-Type.

Par exemple :

Content-Type : "text/html " Le corps du message contient des octets à interpréter comme ceux d'une page écrite en HTML.

Content-Type : "image/jpg " Le corps du message contient des octets à interpréter comme ceux d'une image au format jpeg

II.3- URIs et URLs

Le succès du ``web" s'appuie largement sur un système de nommage des objets accessibles, qui en uniformise l'accès, qu'ils appartiennent à la machine sur laquelle on travaille ou distants sur une machine en un point quelconque du réseau (mais supposé accessible). Ce système de nommage universel est l'**url** " Uniform Resource Locator" dérivé d'un système de nommage plus général nommé **uri**
" Universal Resource Identifier" .

La syntaxe générale d'un(e) url est de la forme :

<scheme>:<scheme-specific-part>

Succinctement, la " scheme" est une méthode que l'on sépare à l'aide du caractère " : " d'une chaîne de caractères ascii 7 bits dont la structure est essentiellement fonction de la ``scheme" qui précède et que l'on peut imaginer comme un argument.
Une ``scheme" est une séquence de caractères 7bits. Les lettres "a" à "z", les chiffres de "0" à "9", le signe "+", le "." et le "-" sont admis. Majuscules et minuscules sont indifférenciés.
Exemples de ``schemes" : **http, ftp, file, mailto**, ...Il en existe d'autres non indispensables pour la compréhension de cet exposé.
Globalement une url doit être encodée en ascii 7 bits sans caractère de contrôle (c'est à dire entre les caractères 20 et 7F), ce qui a comme conséquence que tous les autres caractères doivent être encodés.
La méthode d'encodage transforme tout caractère non utilisable directement en un triplet formé du caractère "%" et de deux caractères qui en représentent la valeur hexadécimale. Par exemple l'espace (20 hex) doit être codé **%20**.
Un certain nombre de caractères, bien que théoriquement représentables, sont considérés comme non sûrs (unsafe) et devraient être également encodés de la même manière que ci-dessus. Ce sont :

% < > " # { } | \ ^ ~ [ ] `

Pour un certain nombre de ``schemes" (**http**...) certains caractères sont réservés car ils ont une signification particulière. Ce sont :

; / ? : @ = &

Ainsi, s'ils apparaissent dans l'url sans faire partie de sa syntaxe, ils doivent être encodés.

II.3.1- Scheme http

Une url avec la "scheme" **http** bien formée doit être de la forme :

http://<host>:<port>/<path>?<searchpath>

``path" et ``searchpath" sont optionnels.

host : c'est un nom de machine ou une adresse IP.

port : le numéro de port. S'il n'est pas précisé, la valeur 80 est prise par défaut.

path : c'est un sélecteur au sens du protocole **http**.

Searchpath : c'est ce que l'on appelle la ``query string", autrement dit la chaîne d'interrogation.

À l'intérieur de ces deux composantes, les caractères **/ ; ?** sont réservés, ce qui signifie que s'ils doivent être employés, ils doivent être encodés pour éviter les ambiguïtés.
Le « **?** » marque la limite entre l'objet interrogeable et la ``query string". À l'intérieur de cette chaîne d'interrogation le caractère + est admis comme raccourci pour l'espace (ascii 20 hex). Il doit donc être encodé s'il doit être utilisé en tant que tel.
De même, à l'intérieur de la ``query string" le caractère = marque la séparation entre variable et valeur, le **&** marque la séparation entre les couples **variable = valeur**.

Exemple récapitulatif :

http://www.google.fr/search?q=cours+r%E9seaux&hl=fr&start=10&sa=N

Notez le "é" codé **%E9**, c'est à dire le caractère de rang 14 x 16 + 9 = 233. On peut également observer quatre variables **q, hl, start** et **sa** dont la signification peut être partiellement devinée, mais dont le remplissage reste à la charge du serveur en question.
Le rôle de la chaîne ``**search**" est celui de ce que l'on appelle une CGI ou ``Common Gateway Interface", c'est à dire un programme qui effectue le lien entre le serveur interrogé et des programmes d'application, ici une recherche dans une base de données.

II.4- Architecture interne du serveur Apache

Au mois de mars 2002, d'après le ``Netcraft Web Server Survey" le serveur le plus utilisé (55%) est très majoritairement celui du projet Apache[6] [7] [8].
D'après ses auteurs, le serveur Apache est une solution de continuité au serveur du NCSA (National Center Supercomputing Applications). Il corrige des bugs et ajoute de nombreuses fonctionnalités, particulièrement un mécanisme d'API (Application Programming Interface) pour permettre aux administrateurs de sites de développer de nouveaux modules adaptés à leurs besoins propres.
Plus généralement, tout ce qui n'est pas strictement dans les attributions du serveur (gestion des processus, gestion mémoire, gestion du réseau) est traité comme un module d'extension.
Le serveur Apache introduit également la possibilité de serveurs multi-domaines (domaines virtuels), ce qui est fort apprécié des hébergeurs de sites.
Maintenant, nous allons présenter l'architecture d'Apache. Nous décrirons aussi les étapes du traitement de chaque requête.

II.4.1- Environnement d'utilisation

Le serveur se met en œuvre simplement. La compilation fournit un exécutable, httpd, qui, pour s'exécuter correctement, a besoin des trois fichiers ASCII de configuration : srm.conf, access.conf, et httpd.conf. C'est en fait dans celui-ci que sont effectués l'essentiel des ajustements locaux à la configuration standard. Lors de l'exécution, trois fichiers sont modifiés :

- **httpd.pid**

  (PidFile) contient le "process ID" du "leader" de groupe; à utiliser avec le signal SIGHUP pour provoquer la re-lecture et le re-démarrage "à chaud" du serveur, ou avec SIGTERM pour mettre fin à son activité.

- **access_log**

  (CustomLog) Qui contient le détail des accès clients.Ce fichier peut devenir très volumineux.

- **Error_log**

  (ErrorLog) Qui contient le détail des accès infructueux et des éventuels problèmes de fonctionnement du serveur.

La *figure 2* synthétise l'environnement d'utilisation.

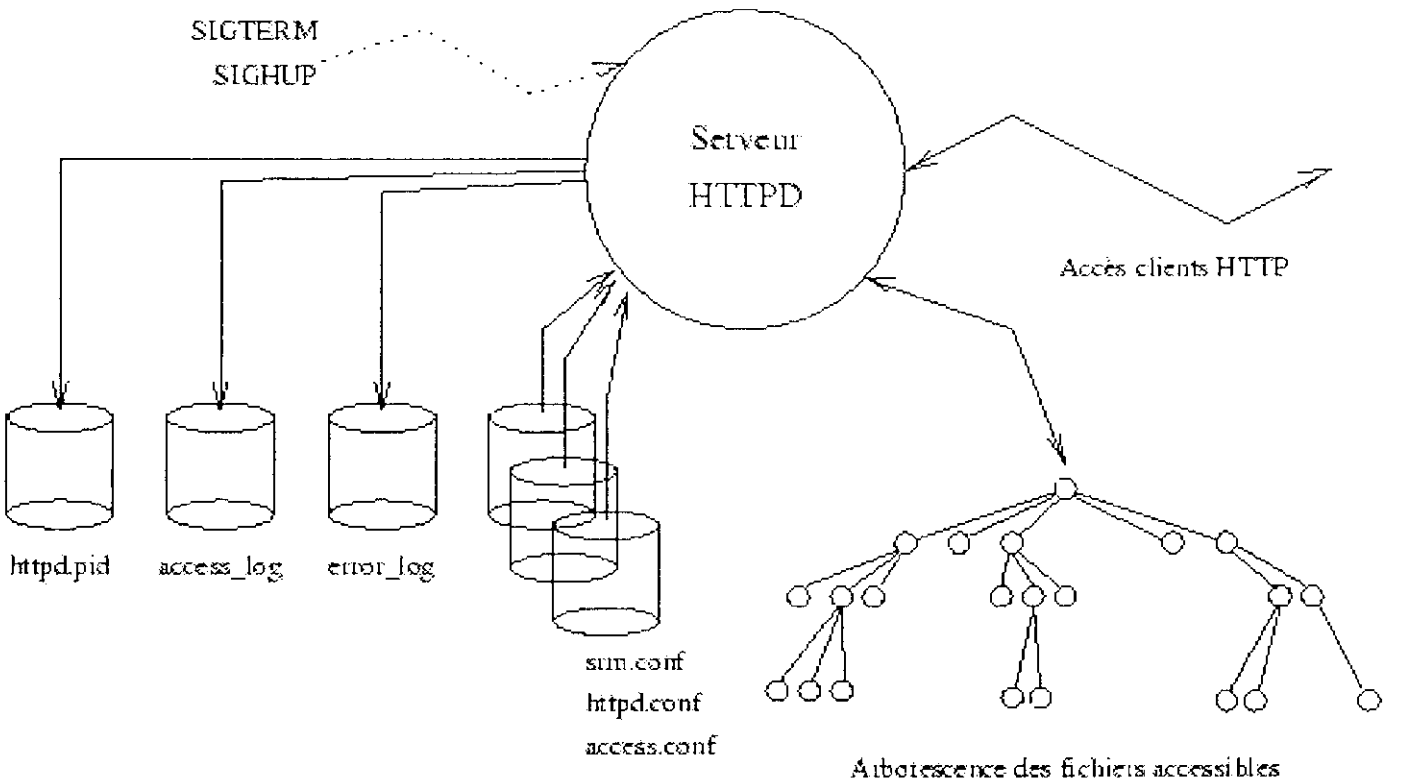


***Fig.2** Serveur APACHE*

II.5- Fonctionnement d'Apache

Une grande partie d'Apache est évidemment dirigée par le fait que c'est un server HTTP qui tourne en tâche de fond. Il lit ses fichiers de configuration, se met en veille et se réveille lors d'une connexion TCP/IP avec un navigateur, il reconnaît les URIs (Universal Ressource Identifier) et les traduit en nom de fichiers ou de scripts et retourne une réponse au navigateur. Les modules jouent un rôle actif dans chacune de ces étapes.
Apache multiplexe ses opérations de manière à pouvoir traiter une nouvelle requête avant d'avoir fini de traiter la précédente. Sur les systèmes Linux (ceux qui nous intéressent dans le cas présent), Apache utilise un modèle multiprocessus dans lequel il lance un groupe de serveurs :

- ✓ un parent unique, qui est responsable de la supervision,
- ✓ un ou plusieurs enfants, qui sont responsables du traitement des requêtes.

II.5.1- Description d'une requête

Voici une description typique d'une requête :
La première ligne en générale contient trois composants : la méthode (GET, POST, HEAD, PUT ou DELETE), l'URI et la version du protocole.

Les champs (headers ou en-têtes) sont les suivants :

- Connection : est une suggestion au serveur web qu'il devrait garder la connexion TCP/IP ouverte après avoir fini cette requête. C'est une optimisation qui améliore les performances sur les pages qui contiennent des images, celles-ci vont être appelées via des requêtes supplémentaires.
- User-Agent donne des informations sur le modèle de navigateur.
- Host : est le nom donné dans l'URI et est utilisé par le système d'hôte virtuel (virtual host) pour sélectionner le bon arbre de documents et le bon fichier de configuration.
- Accept : est une liste de types MIME (Multipurpose Internet Mime Extension) que le navigateur accepte.
- Accept-Language : indique le langage de préférence de l'utilisateur.
- Accept-Charset : indique quels sets de caractères le navigateur est capable d'afficher.
- Accept-Encoding : est une liste de format de compression que le navigateur supporte.

II.5.2- Description d'une réponse

La réponse est similaire à la requête. La première ligne est la ligne de statut contenant trois composants : la version du protocole, le code de la réponse et une version lisible humainement du code de réponse. Viennent ensuite les champs optionnels dont les plus importants sont les suivants :

- Content-Length donne la longueur du document en bytes.
- Content-Type donne le type MIME du document renvoyé suivit du set de caractères utilisé.
- Content-Encoding donne le type de compression du document renvoyé.

Ces champs sont ensuite suivis du document lui-même.

II.6- Cycle de vie d'Apache

Le cycle de vie d'Apache est illustré à la figure suivante.

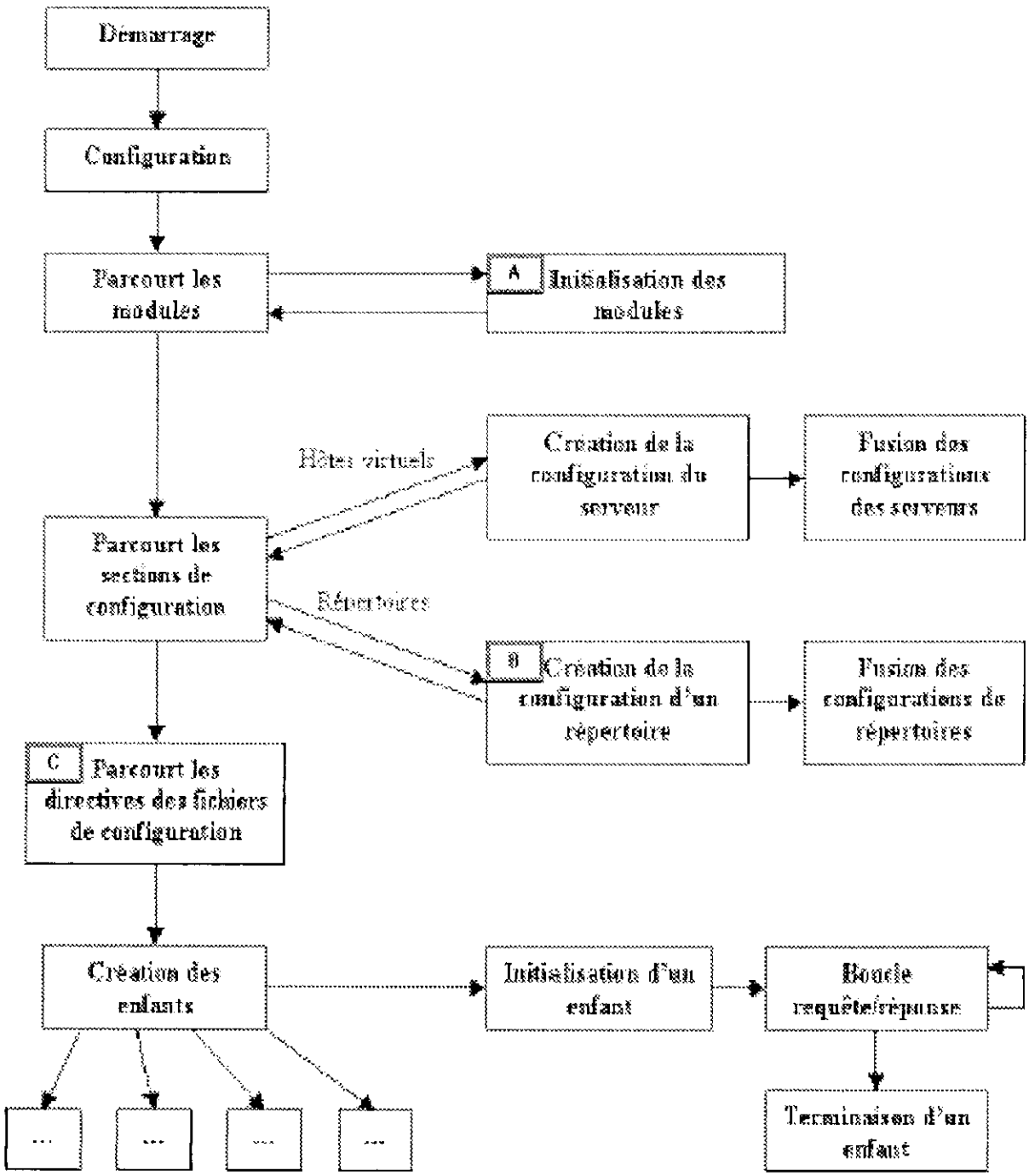


Fig. 3 Cycle de vie d'Apache.

Il démarre, s'initialise, crée plusieurs copies de lui-même et entre dans la boucle de traitement des requêtes. Ensuite, chacune des copies sort de cette boucle et se termine.

II.6.1- Boucle de traitement de requêtes

Maintenant que nous avons quelque connaissances de bases sur l'architecture d'Apache, analysons la partie qui nous intéresse plus particulièrement : le traitement des requêtes.
La boucle de traitement des requêtes est illustrée à la figure suivante

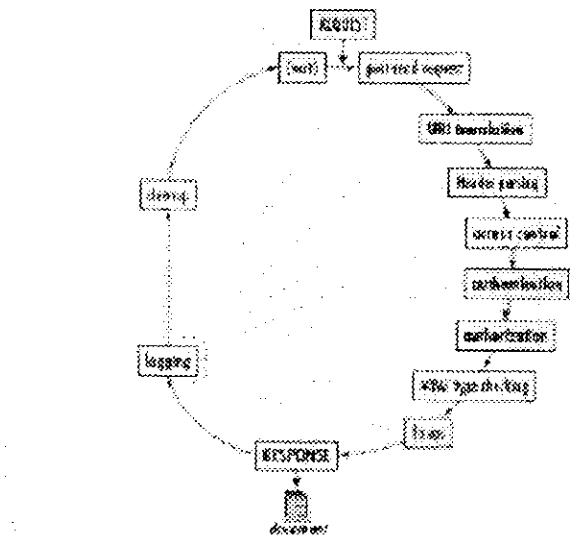


Fig. 4 La boucle de traitement des requêtes.

Le noyau d'Apache gère les aspects les plus communs d'une conversation HTTP: attente de la requête, analyse de la requête et de ses en-têtes, et composition de la réponse.
A chaque boucle, il y a un certain nombre de décisions à prendre et des modules externes peuvent définir des gestionnaires pour améliorer ou remplacer chaque décision. Si aucun gestionnaire n'est défini, Apache retourne à son comportement par défaut.
Chacune de ces décisions est représentée sous forme de phase sur la figure ci-dessus. La plupart des phases sont terminées par le premier gestionnaire qui les traite, sauf pour les phases logging, fixups, access control et authentication, où tous les gestionnaires sont toujours exécutés.

Un gestionnaire pour une phase peut typiquement faire une des trois actions suivantes :

- gérer la requête, et indiquer quand il a fini en retournant la constante OK,
- décliner la gestion de la requête en retournant la constante DECLINE,
- signaler une erreur, en retournant un code d'erreur HTTP.

S'il signale d'une erreur le traitement normal de la requête est terminé et seule la phase de logging est encore exécutée.

- *URI translation*

L'URI peut se référer à un fichier physique, un document généré en temps réel par un script externe ou un document généré par un module interne. Le serveur doit savoir dès le départ quel genre de document il doit gérer.

Les routines de traduction par défaut d'Apache utilisent les directives Alias, ScriptAlias et DocumentRoot pour traduire l'URI en chemin d'accès complet au document demandé. Certains modules externes, comme mod_rewrite, permettent un contrôle plus sophistiqué de cette phase.

♦ *Access control*

Cette phase détermine si l'URI demandé par la requête est accordée en se basant sur des critères non spécifiques aux utilisateurs.

♦ *Authentication*

Cette phase permet de vérifier que l'utilisateur est bien celui qu'il prétend être.

♦ *Authorization*

Elle détermine si l'utilisateur identifié d'une telle requête est autorisé à accéder à une telle URI.

♦ *MIME type checking*

Cette phase est très importante. Apache fait une première supposition sur le type de contenu du document. Sur base de ce type de contenu, Apache va décider comment traiter le document. Il base sa décision sur le nom du document, l'extension du nom de fichier ou sur la configuration du répertoire auquel appartient le fichier.
Il existe trois types de contenu :

- un type MIME (par exemple image/vif),
- un codage (par exemple httpd/unix-directory).
- un libellé de gestion de contenu (par exemple jakarta-servlet ou traceoutput comme nous le verrons plus tard).

Une fois que ce type est déterminé, Apache l'utilise pour sélectionner le gestionnaire de contenu (content handler) et pour générer ou transmettre le document lui-même pendant la phase de réponse.

♦ *Fixups*

Cette phase est utilisée pour modifier la requête en dernière limite, avant que celle-ci ne soit traitée par le gestionnaire de contenu. Mod_trace_output utilise cette phase pour déterminer la suite de son action sur la requête en cours.

♦ *Response*

Une fois qu'Apache a décidé quel module va gérer le contenu, il lui passe l'URI et les informations accumulées sur le document.Typiquement, le module modifie les champs de la réponse HTTP et dit à Apache de les envoyer au navigateur. Ensuite, il crée le contenu de la réponse et l'envoie au navigateur client. La phase de réponse de mod_trace_output est sensiblement plus compliquée.

♦ *Logging*

Apache fournit un système d'enregistrement qui écrit dans des fichiers journaux. En plus des enregistrements par défaut, il est également possible de les personnaliser en fonction des besoins.

♦ *Cleanup*

La requête est terminée, mais il reste peut-être des ressources à libérer.
Les modules peuvent enregistrer des gestionnaires de nettoyage pour libérer les ressources qu'ils ont utilisées, comme les connexions aux bases de données, la mémoire, etc.

II.6.2- Requêtes internes et sous-requêtes

Une requête interne a tout de la requête ordinaire, excepté qu'elle a été générée par Apache. Les sous-requêtes sont un cas particulier des requêtes internes.

II.7- Apache comme étant un serveur Proxy et Cache

Apache fournit tous les éléments essentiels nécessaires qui le rendent utilisable à la foi en tant que serveur Proxy et Cache :

- **Serveur Proxy** : est un serveur spécialisé qui agit en tant qu'intermédiaire entre les clients et les autre serveur Web. Les clients se connectent au serveur Proxy, et lui envoient leurs requêtes, plutôt que de se connecter directement au serveur Web qu'il désirent rejoindre. Le Proxy tente en suite de retracer les ressources que chaque client requiert pou les lui resservir. Les clients et le serveur Proxy résident habituellement sur le même réseau local. Il sert a :
  - ✓ Cacher la topologie et la structure du réseau interne au monde extérieur.
  - ✓ Surveiller le trafic des clients d'Intranet, puisque c'est le point d'accès au monde extérieur.
  - ✓ Coupler avec un Firewall, il renforce la sécurité du réseau local.

- **Serveur Cache** : sert a faire des sauvegardes pour les pages consultées. Si un client demande une page déjà consultée, il va se servir par la copie du cache. Cela économise l'utilisation du réseau local, en épargnant la bande passante Internet, et augmente considérablement la vitesse d'accès Web.

III- Configuration d'apache

Apache est entièrement paramétrable à l'aide de fichiers textes de configuration. Dans la configuration[10] standard, le serveur utilise (lit) d'abord *httpd.conf*, puis *srm.conf*, et enfin *access.conf*.
Le premier définit les attributs généraux du serveur (tels que le numéro de port ou l'utilisateur sous lequel il fonctionne).
Le second définit la racine de l'arborescence de documents et des fonctions spéciales telles que l'analyse du HTML effectuée par le serveur, l'analyse d'implantations d'images internes, etc.
Enfin, *access.conf* concerne les différents cas d'accès.
Mais actuellement et dés la version 1.3.4 d'apache, les trois fichiers ont fusionné en un seul fichier de configuration qui est **httpd.conf.** Ce dernier contient des directives de configuration qui donnent au serveur ses instructions.

Les directives de configuration d'Apache sont regroupées en trois sections :

- Les directives qui contrôlent l'environnement global du serveur apache.
- Les directives qui définissent les paramètres du serveur principal (ou du serveur par défaut), répondant aux requêtes non issues d'hôtes virtuels.

➢ Les directives qui définissent les paramètres relatifs aux hôtes virtuels.
Grâce aux hôtes virtuels, les requêtes Web peuvent être envoyées à différentes URL, mais seront traitées par le même processus serveur. Il ne sera pas fait ici usage d'hôtes virtuels.
Maintenant en va voir quelques directives, selon leur groupe

III.1- Environnement global

ServerType standalone Cette directive définit comment le serveur est exécuté par le système d'exploitation. Elle accepte deux options : standalone ou inetd.

- Le mode standalone est le plus utilisé car il offre de meilleures performances. Dans le cas d'un site très sollicité, le mode standalone sera certainement la seule solution possible.
- Le mode inted est parfois préféré pour des raisons de sécurité malgré le coût important en ressources pour chaque connexion : ni le mode standalone ni le mode inetd ne peuvent assurer une sécurité totale, mais ce dernier étant nettement moins utilisé, a par conséquent moins de chance de subir des attaques. Cependant, certains paramètres avancés d'Apache ne sont pas compatibles avec ce mode, qui risque d'ailleurs de ne plus être supporté dans les versions ultérieures.

ServerRoot "/usr/local/httpd" Répertoire principal sous lequel les fichiers de configuration, d'erreur et log sont stockés.

PidFile logs/httpd.pid Fichier dans lequel le serveur enregistre son PID (Proccess Identification) lorsqu'il est lancé.

Timeout 300 Temps d'attente maximum (en secondes) du serveur pour recevoir une requête, qui s'applique pour chaque bloc de données transféré plutôt que pour l'ensemble du transfert. Ce qui autorise le téléchargement de fichiers importants avec une connexion lente, sans interrompre le transfert.

KeepAlive On Après que l'utilisateur se soit connecté sur le site, il y accèdera probablement de nouveau par la suite. Cette directive, paramétrée sur "on", permet de maintenir la connexion persistante et ainsi de gagner du temps.

MinSpareServers 5 Apache vérifie périodiquement le nombre de serveurs fils en attente de connexion. Si celui-ci est inférieur à MinSpareServers, de nouveaux serveurs fils sont lancés, au rythme d'un par seconde.

MaxSpareServers 10 Représente le nombre maximum de serveurs fils en attente de connexion. Sauf nécessité, il est préférable ne pas utiliser des valeurs élevées pour ces deux dernières directives, afin de ne pas épuiser inutilement les ressources.

StartServers 5 Nombre de serveurs fils lancés au démarrage.

MaxClients 150 Nombre de serveurs pouvant s'exécuter en même temps.

MaxRequestsPerChild 30 Chaque exemplaire fils d'Apache traite ce nombre de requêtes et meurt. Lorsque MaxRequestsPerChild est paramétré à 0, le processus dure jusqu'au redémarrage de la machine ; ce qui est à éviter pour des raisons de sécurité.

III.2- Configuration du serveur principal

Port 80 Le port écouté par le serveur en mode standalone.
ServerAdmin root@Labo11.elec Lorsqu'une erreur survient sur le serveur suite à une requête, il est possible d'envoyer un courrier électronique à l'adresse spécifiée par cette directive.
ServerName http://labo11.elec/ URL du serveur.
DocumentRoot "/usr/local/httpd/htdocs" Répertoire principal contenant l'arborescence de documents à publier.

III.3- Hot virtuel

Afin de faire publier plus d'un site Web a partir d'un seul serveur apache, on utilise la procédure d'hôte virtuel comme illustré dans l'exemple ci-après.
Dans cet exemple, un seul serveur nommé : www.labo11.elec va être servir des clients à partir du répertoire : /home/www, et enregistre des erreurs dans le répertoire:/home/error_log :

```
<VirtualHost www.labo11.elec>
ServerAdmin root@Labo11.elec
DocumentRoot /home/www
ServerName www.labo11.elec
ErrorLog /home/error_log
</VirtualHost>
```

III.4- Démarrage du démon de serveur apache

Pour le bon fonctionnement de serveur, on doit démarrer le démon apache selon l'une des deux méthodes suivantes :

a- Démarrage manuel :

En tant que root, il suffit de saisir la commande suivante :

```
# /sbin/init.d/apache start
```

Le serveur apache est alors prêt à traiter ces requêtes.

b- Démarrage au démarrage du système

Pour que le serveur soit chargé au démarrage de la machine, il faut modifier le fichier /etc/rc.conf en lui ajoutant :

```
# Lancement du serveur web apache
START_HTTPD="YES"
```

VI- Conclusion

Apache est le serveur HTTP le plus utilisé dans le monde. Cela ne serait pas étonnant si en plus d'être très performant, il fait partie des logiciels dits opensource
Mais en fait, qu'est-ce qui différencie vraiment Apache des serveurs propriétaires :

- ✓ Il est totalement paramétrable et contrôlable à l'aide de son fichier de configuration ; simple à modifier car il s'agit d'un fichier texte.
- ✓ Il ne cesse de s'améliorer par de nouvelles versions plus adaptées aux exigences du monde Web grâce à son architecture micro-noyau.
- ✓ Il peut tourner sur la plupart des systèmes d'exploitation dont notamment UNIX/Linux et Windows 2000/NT ; ce qui le rend compatible multi-plateforme.

I- Introduction

Le courrier électronique, ou `` e-mail '' est un des services les plus populaires sur l'Internet, avec le transfert de fichier et les services 3W. C'est aussi un des plus vieux services existants sur les réseaux, bien avant l'apparition d'Internet.
Sa popularité repose sur sa grande souplesse et rapidité d'emploi. Il permet aussi bien les échanges professionnels que les échanges privés. Son mode d'adressage donne la possibilité d'envoyer un courrier à une personne comme à une liste de personnes ou encore à un automate capable de rediffuser vers un groupe donné (``mailing-list'').
De nombreux outils développés (d'abord essentiellement sur le système Unix) autour de ce concept ouvrent un vaste champ de possibilités aux utilisateurs comme la ventilation par thème, le renvoi automatique, le répondeur (pendant les absences), la réception de fax,...
Un des plus connus et des plus utilisés est Sendmail.

Le package Sendmail[3] a été écrit par Eric Allman, en 1980 alors qu'il travaillait à l'Université de Berkeley. Lorsqu'il a quitté Berkeley, il a arrêté de travailler sur Sendmail. Le programme est alors resté en l'état (version 5). Plusieurs améliorations ont alors vu le jour indépendamment, la plus connue étant la version IDA. Depuis 1990, Eric Allman avait repris son travail sur Sendmail [23]et a sorti la version 6, devenue très rapidement la version 8 pour des raisons de numérotation de versions internes à Berkeley.

II- Théorie

II.1- Format des messages

Un message est constitué de deux parties distinctes :

- un *en-tête*

L'en-tête est une suite de champs, Chaque champ est constitué d'un mot-clef (ou commande), du caractère << : >>, et des informations du champ.

- un *corps*

Le corps est le contenu du message proprement dit. Il est séparé de l'en-tête par au moins une ligne vide. Le standard MIME définit dorénavant une structure pour le corps (lignes de 1000 caractères au maximum; les caractères sont tronqués à 7 bits).

II.2- Adresse électronique

Tous les courriers électroniques ont un destinataire précisé par son adresse électronique, ou ``e-mail''. Celle-ci précise le nom du destinataire et le site où il reçoit son courrier électronique.
Le nom du destinataire est une chaîne de caractères.
Par exemple, il est assez fréquent de voir employer le nom complet (prénom et nom de famille) pour désigner l'interlocuteur distant. La conversion ultime entre cette convention et la boîte aux lettres de l'utilisateur est l'affaire du ``bureau de poste le plus proche'', c'est à dire le programme ``sendmail'' pour nous.
Le caractère ``@'' (lire ``at'') sépare l'identificateur du destinataire de la destination.

La destination est peut être vide (il s'agit alors d'un destinataire sur la machine courante, ou d'un ``alias'' que le sendmail local sait traiter), être un nom de machine du domaine local, le nom d'un autre domaine ou d'une machine sur un autre domaine.
Les adresses suivantes ont un format valide :

user1 : Destinataire local.

user2@machine : Destinataire sur une machine du domaine courant.
user3@machine.domaine : Destinataire sur une machine particulière d'un domaine particulier.
user4@domaine : Destinataire sur un domaine particulier (par forcément local).

On devine aisément que la possibilité d'envoyer du courrier sur une machine distante sous entend l'usage du DNS (voir la section serveur DNS).

II.3- Structure d'un système de messagerie

Un système de messagerie Internet contient les composants suivants :

- *agent utilisateur* (ou *user agent*, ou UA).
  Un UA est un programme que l'utilisateur emploie pour composer son message et l'envoyer à l'agent de routage (voir ci-dessous) pour l'injecter dans le système de messagerie. Un UA permet également la lecture du courrier. C'est la phase finale. Il y a un UA à chaque extrémité du système de messagerie.
- *agent de routage des messages*
  Un agent de routage reçoit un message. En fonction de l'adresse du destinataire, il décide de faire appel à un agent de transport de messages dont le but est d'acheminer le message dans la direction du destinataire.
  Les agents de routage sous Unix sont, entre autres, sendmail, kmail et mmdf. Toutefois, le plus répandu est incontestablement Sendmail. Il est l'implémentation d'un agent de routage de messages ainsi que d'un agent de transport spécialisé pour le protocole SMTP .
- *agent de transport des messages* (ou MTA)
  Un agent de transport ou « *mailer* » reçoit un message et une direction. Il l'achemine à l'endroit indiqué. Il faut noter que l'agent de transport des messages ne prend pas de décision quant au routage. Cette décision lui est indiquée par l'agent de routage qui lui transmet le message.
  Un agent de transport de messages est spécialisé pour un type de transmission. Par exemple, il y aura un agent de transport pour le protocole SMTP, un autre pour la remise physique du message (lorsque le courrier arrive dans la boîte aux lettres du destinataire), un troisième pour l'expédition vers un site UUCP, etc.
  Les agents de transport des messages typiques sous Linux sont sendmail (une partie de Sendmail est dédiée au protocole SMTP), uucp pour l'acheminement vers un site UUCP, ou encore /bin/mail pour la remise physique.
- une boîte aux lettres.
  Sur les systèmes Unix, une boîte aux lettres est simplement un fichier dans/var/spool/mail. Dans un tel fichier, tout nouveau message débute par une ligne From (sans caractère << : >>). Cette ligne, qu'on appelle souvent le << From Unix >> sert à délimiter les différents messages.

Le monde de la messagerie Sendmail rend assez difficile une perception nette de chacun de ces composants. En particulier, une confusion peut naître à cause des deux points suivants :

- le programme Sendmail lui-même est un agent de routage plus un agent de transport spécialisé pour le protocole SMTP. Si cette concentration dans un seul programme n'est pas intellectuellement satisfaisante, elle est dictée par un souci d'efficacité.
- le programme /bin/mail est utilisé à la fois comme UA et comme agent de transport de messages (puisque c'est lui qui réalise la remise physique). Ce rôle dual est une erreur de conception et les nouvelles versions de sendmail fournissent un programme distinct pour la remise physique. Certains sites utilisent également le programme procmail.

II.4- Protocole SMTP

Le courrier électronique au sein d'Internet est géré par le protocole SMTP (*Simple Mail Transfer Protocol*) bâti sur TCP (port 25). Il permet d'échanger des messages entre un expéditeur et un (ou plusieurs) destinataire pourvu que leurs adresses soient connues. Il est à noter qu'un mécanisme d'alias permet de définir des équivalences entre adresses; notamment de préciser quelle machine parmi toutes celles d'un même domaine gère réellement le courrier de chaque utilisateur.

Une des caractéristiques principales du protocole SMTP est d'effectuer une remise différée du courrier qui assure que le service sera correctement rendu même si le réseau ou l'ordinateur destinataire sont momentanément en panne ou surchargés.

Un courrier expédié par un utilisateur est d'abord copié dans une mémoire de *spool* accompagné des noms de l'expéditeur, du récepteur, de l'ordinateur destinataire et de l'heure de dépôt. Puis il associe le nom de l'ordinateur destinataire à une adresse IP et tente d'établir une connexion TCP avec le serveur SMTP de client; Si cela réussit, le processus de transfert envoie une copie du message au destinataire qui l'enregistre dans une zone de spool spécifique. Lorsque le client et le serveur se sont confirmés l'envoi et l'enregistrement complet du message, le client supprime sa copie locale. Si le client n'arrive pas à établir une connexion TCP, ou si elle est rompue lors du transfert d'un message, il enregistre l'heure de cette tentative et réessaye quelque temps plus tard d'expédier le message. Il finira par retourner à son expéditeur un message impossible à expédier après un délai important.

Le problème est que SMTP est un protocole limité, par exemple ne permet pas l'échange de courriers contenant des caractères accentués et des besoins ultérieurs qui ne manqueront pas d'apparaître. Pour ce la deux approches complémentaires ont été adoptées en 1992 :

- définition de MIME (Multipurpose Internet Mail Extensions) pour être capable, entre autres, et si besoin est, d'échanger des messages contenant des accents (donc sur 8 bits) ou des données binaires sur un canal SMTP .
- définition de ESMTP (Extended SMTP), extension au protocole SMTP, afin de pouvoir faire évoluer le protocole tout en gardant la compatibilité avec les implémentations actuelles ou à venir.

II.4.1- Commandes

Les mots-clefs (ou commandes) du protocole SMTP sont listés ci-dessous. Chaque mot-clef est cité avec son nom, sa syntaxe (entre parenthèses, sans distinction entre minuscules et majuscules) et sa signification.

- Hello (HELO site-émetteur)

Commence une session SMTP par une identification des deux parties. Le récepteur s'identifie dans sa réponse ;

- Mail (MAIL FROM: expéditeur) Commence une nouvelle transaction. Spécifie l'expéditeur, pour le renvoi des messages d'erreur éventuels ;
- Recipient (RCPT TO: destinataire)
  Spécifie un destinataire. Cette procédure peut être répétée autant de fois que nécessaire. Si le destinataire n'est pas valide, un code d'erreur est renvoyé aussitôt ;
- Data (DATA)
  Envoie le message (en-tête + corps) terminé par une ligne ne contenant qu'un point. Pour ne pas interférer avec le message à transmettre, toute ligne du message contenant un point en première position est modifiée en insérant un point supplémentaire en début. Ce point supplémentaire est supprimé par le récepteur (*hidden dot algorithm*) ;
- Send (SEND FROM: expéditeur)
  Commence une nouvelle transaction (de manière analogue à MAIL) pour envoyer un message sur un terminal ;
  Note : non supporté par sendmail.
- Send or mail (SOML FROM: expéditeur)
  Commence une nouvelle transaction (de manière analogue à MAIL) pour envoyer un message sur un terminal ou dans une boîte aux lettres si le destinataire n'est pas connecté
  Note : non supporté par sendmail.
- Send and mail (SAML FROM: expéditeur)
  Commence une nouvelle transaction (de manière analogue à MAIL) pour envoyer un message sur un terminal et dans une boîte aux lettres ;
  Note : non supporté par sendmail.
- Reset (RSET)
  La transaction courante est annulée, l'ensemble du logiciel est réinitialisé, un nouveau message peut donc être envoyé sur le même canal ;
- Verify (VRFY destinataire)
  Demande au site récepteur de vérifier que le destinataire est une adresse valide ;
- Expand (EXPN destinataire)
  Demande au site récepteur, d'une part de vérifier que le destinataire est une adresse de liste de diffusion valide, et d'autre part de renvoyer les identités des membres de la liste ;
- Help (HELP commande)
  Renvoie la liste des commandes SMTP admises par le site récepteur ;
- Noop (NOOP)
  Ne fait rien ;
- Quit (QUIT)
  Termine la session SMTP ;

II.4.2- Codage

Les agents de transport de messages ne permettent pas d'envoyer des messages composés de caractères accentués (avec le huitième bit), ou composés de données binaires telles que des sons, des images, des fichiers compressés, etc. car la notion de ligne est toujours présente. Pour transmettre ces données, il faut les transformer éventuellement dans une forme compatible avec les agents de transport. Cette transformation, si elle a lieu, doit pouvoir être inversée par le destinataire (ou un nœud intermédiaire). Pour cela, on ajoute un champ (Content –Transfer -

Encoding) à l'en-tête afin de spécifier le type de codage utilisé pour le corps du message (ou le sous-corps s'il s'agit d'un message multipart).

À l'heure actuelle, 5 codages possibles sont définis. Ils sont résumés dans le tableau suivant :

Codage	Informations sur	Codée sur	Transportable avec
7bit	7 bits	7 bits	SMTP
Quoted-printable	8 bits	7 bits	SMTP
Base64	8 bits	7 bits	SMTP
8bit	8 bits	8 bits	ESMTP/8bits
Binary	8 bits	8 bits	ESMTP/binaire

II.5- Le protocole ESMTP

ESMTP (Extended SMTP), extension au protocole SMTP, afin de pouvoir faire évoluer le protocole tout en gardant la compatibilité avec les implémentations actuelles ou à venir.
La nouveau mot-clef, EHLO (les lettres E et H sont inversées), qui introduit un dialogue ESMTP si les deux parties le reconnaissent Il s'agit, pour le client, de spécifier qu'un message contient des caractères 8 bits (accentués ou autres),eu lieu de 7 bits pour SMTP

II.5.1- Définition de MIME (Multipurpose Internet Mail Extensions)

La fonction la plus répandue et la plus simple de MIME est d'autoriser l'usage des caractères accentués (codage sur 8 bits ou plus) à l'intérieur du corps du message (l'en-tête SMTP reste codée sur 7 bits). MIME spécifie :

- le type du message, qui peut être du texte, une image, un son, une vidéo, ou encore un agrégat de plusieurs types .
- le codage du message (7 bits, 8 bits ...).

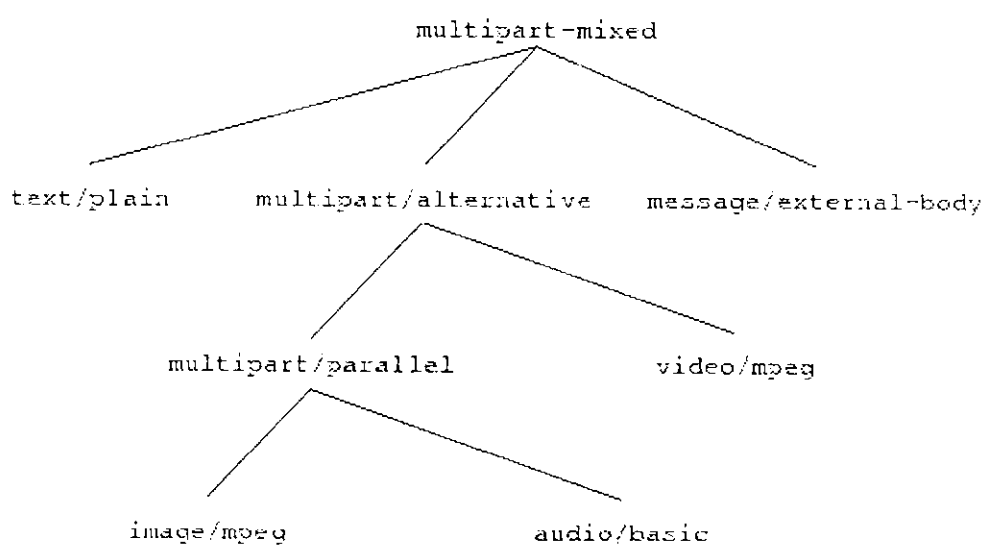


Fig.1 Exemple de structure de document MIME complexe

II.6- Stratégies de transfert

La *figure suivante* illustre la possibilité d'échange entre deux programmes sendmail : la connexion est indirecte. Cela signifie que le sendmail de la station émettrice ne contacte pas directement le sendmail de la station réceptrice mais a traverse les relais ``mailhub ou relay", et lui délivre l'information. Cette architecture est théorique. En pratique il peut y avoir une hiérarchie de ``relay " plus compliquée.

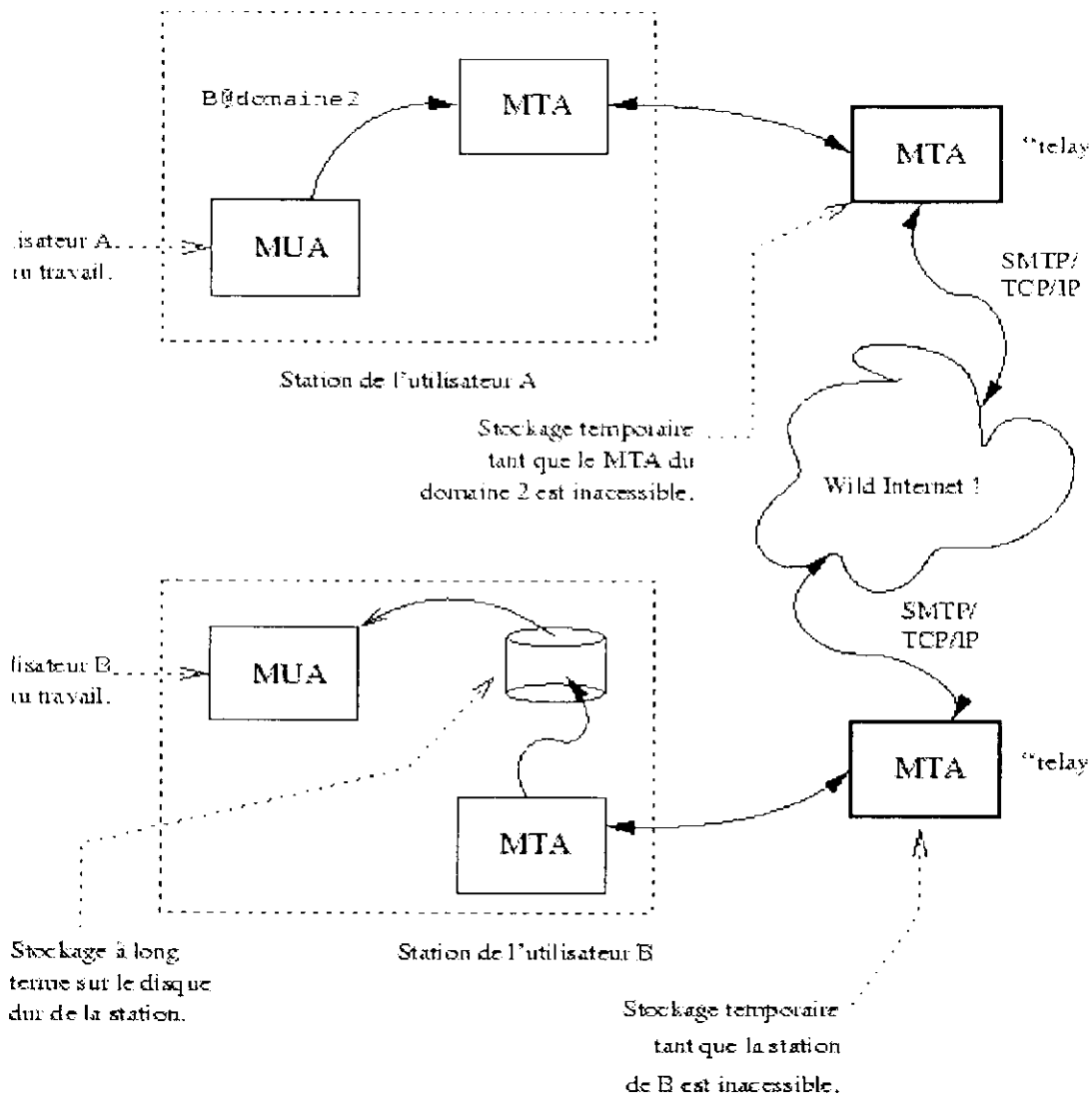


Fig.2 Stratégies de transfert

II.7- Lecture déportée

De plus en plus les utilisateurs lisent leur courrier non pas directement sur leur machine avec un UA local, mais avec un UA *déporté*. Cette lecture est rendue possible grâce aux protocoles de lecture à distance : **POP3** ou **IMAP4**. Dans ce type d'architecture de messagerie, le MTA (par exemple Sendmail sur linux) dépose le courrier dans les boîtes aux lettres des

utilisateurs. La configuration du MTA n'est pas spécifique à la lecture déportée. Lorsqu'un utilisateur désire lire ses messages, il se connecte au serveur POP3 ou IMAP4; ce qui provoque le lancement d'un programme auxiliaire (le démon implémentant le protocole) pour lire la boîte. Cette configuration ne nécessite que l'ajout d'un démon sans modification de la configuration du MTA. Elle est donc très simple à mettre en œuvre.

II.7.1- POP3

POP signifie Post Office Protocol version 3 c'est un protocole d'accès au courrier électronique. *Il s'occupe en faite de l'accès au courrier et non du transport de celui-ci*, le transport du courrier étant assuré par SMTP. L'accès au courrier peut se faire à distance. Le protocole POP fonctionne selon le mode autonome et de plus permet de laisser les messages sur le serveur. L'avantage étant que l'on peut rédiger les messages à émettre sans être connecté et ensuite lors de la connexion les nouveaux messages sont téléchargés et les messages à transmettre peuvent être envoyés. Le problème principal de ce protocole est que si on désire que d'autres personnes aient accès à certains messages il faut les charger sur le poste client puis ensuite; les envoyer aux autres personnes. Pour les personnes accédant par modem, cela rallonge les temps de connexions considérablement.

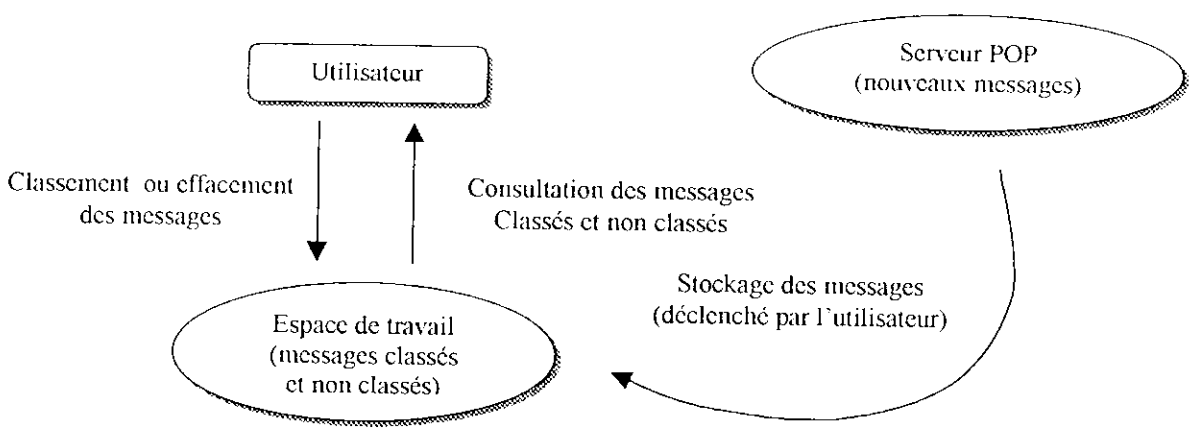


Fig.3 Accès POP3

II.7.2- IMAP

IMAP signifie Internet Message Access Protocol. C'est un protocole d'accès au courrier électronique mais à l'instar de POP, il est capable de fonctionner selon les trois types d'accès. Il faut noter que IMAP est principalement utilisé dans le cas des accès en ligne; c'est à dire que les messages sont traités directement sur le serveur; ce qui facilite la gestion des boîtes aux lettres. En effet, si on désire que d'autres personnes aient accès à certains messages il suffit de les faire suivre dans les boîtes aux lettres des autres personnes, les messages ne seront pas chargés sur le poste client puis ensuite envoyer aux autres personnes. Il est possible de sélectionner les messages que l'on désire consulter. Le problème principal est que le client doit être obligatoirement connecté au serveur. Dans le cas d'une connexion par modem, la facture peut devenir rapidement astronomique.

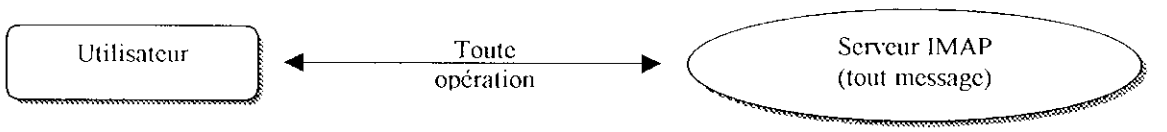


Fig.4 Méthode IMAP

II.8- Interaction avec le DNS

Comme à la présenté, le DNS (*domain name system*) est le mécanisme distribué de nommage dans l'Internet. Le protocole SMTP est bien sûr dépendant du DNS, comme tout protocole utilisant des noms de sites. Mais le DNS comprend une caractéristique conçue spécialement pour le courrier électronique : les *resource records* (RR) de type MX (pour *Mail eXchanger*).

Un RR de type MX associé à un site S a pour but d'indiquer aux sites qui voudraient envoyer un courrier à S de ne pas lui envoyer directement, mais de l'envoyer à une autre machine citée dans le MX.

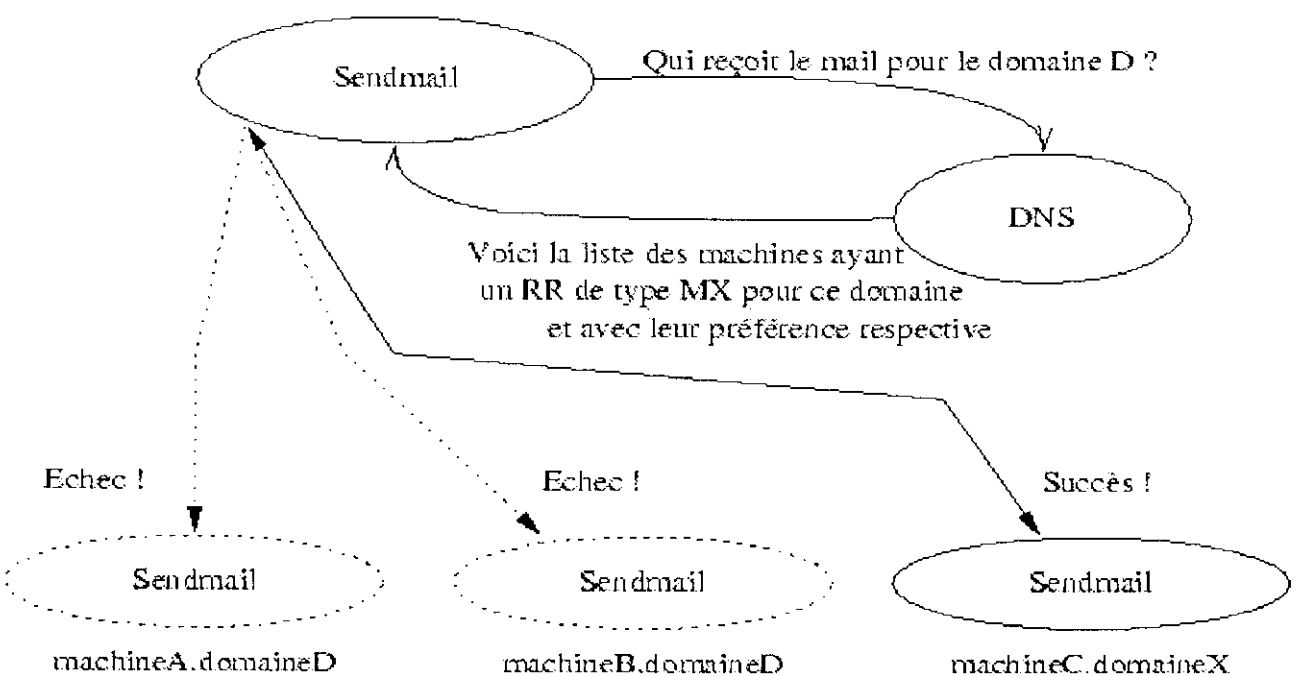


Fig.5 Interaction avec le DNS

Par exemple : Labo11.elec IN MX 10 enp.dz..edu

Ce MX indique que tout courrier adressé à Labo11 doit être en réalité envoyé à ENP. Toute implémentation de SMTP conforme doit obéir à ce MX.
Dans la pratique, un MX set :

- à router un message pour un site n'ayant pas de connectivité Internet
- à recevoir les courriers adressés à un domaine.
- à centraliser la distribution du courrier
- à re-router les courriers en cas de problème

II.9- Règles de réécriture

L'essentiel du travail de Sendmail est réalisé dans les règles de réécriture. Ces règles manipulent l'adresse par un jeu de réécritures analogues, L'objectif des règles de réécriture est de :

- convertir l'adresse du destinataire figurant dans l'enveloppe en une forme canonique facilement manipulable, puis de prendre la décision de routage (c'est-à-dire choisir l'agent de transport) en fonction de l'adresse ;
- prendre chaque champ de l'en-tête contenant une adresse, et la convertir en une forme canonique, puis la réécrire en fonction de l'agent de transport désiré ;
- effectuer certaines actions, comme par exemple la validation d'un relais ou d'une adresse d'émetteur dans le cadre de la lutte anti-*spam(* Le *spam* est un courrier non sollicité envoyé à de très nombreuses personnes, analogue aux prospectus qui inondent nos boîtes aux lettres informatiques).

II.10- Environnement d'utilisation

Sendmail a de multiples relations avec le système d'exploitation comme le montre la *figure ci dessous*

L'activité opérationnelle de sendmail est consignée à l'aide de syslog, qui est le dispositif central d'affichage et de journalisation des messages d'erreur. Ceci explique la présence de la ligne suivante trouvée dans le fichier /etc/syslog.conf pour indiquer ou on peut trouver ces messages :

mail.info /var/log/maillog

UUCP, X.400, SMTP sont autant de moyens de propager le courrier. Le support de transport du courrier n'est pas forcément TCP/IP. On peut opter pour un transport via uucp ou via X.400. La solution uucp, bien qu''`ancienne" est encore de temps en temps employée car elle permet un accès RTC au mail seul, facile à sécuriser car n'utilisant pas une adresse IP et donc interdisant les requêtes http vers l'extérieur. Ces supports peuvent cohabiter au sein d'une même configuration avec autant de ``Mailer" sélectionnés en fonction de l'adresse du destinataire.

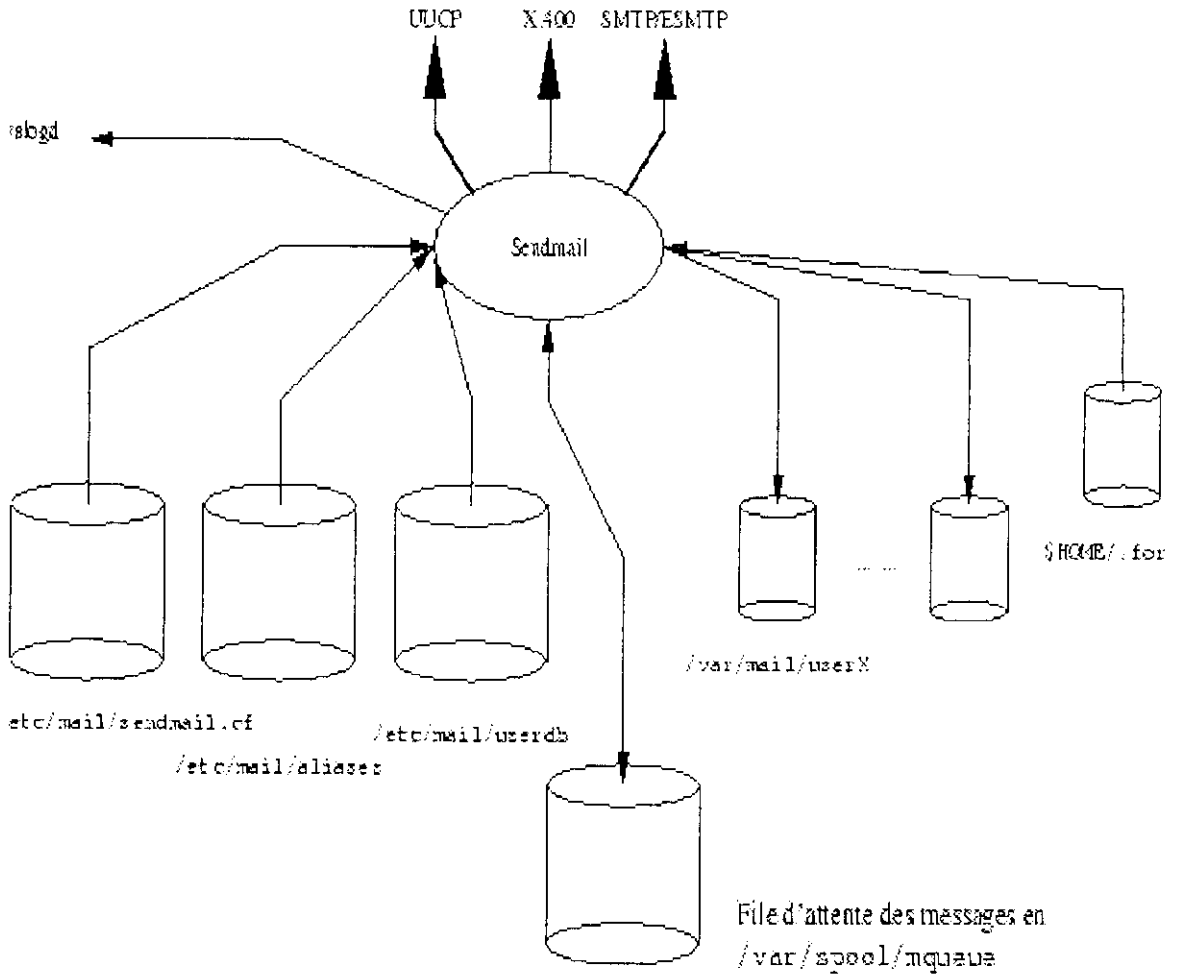


Fig.6 Environnement d'utilisation

- /etc/sendmail.cf est le fichier de configuration principale de sendmail. Tout système linux est livré avec une configuration standard qu'il faut adapter à la situation locale.
- /etc/aliases est une base de synonymes. Quand sendmail reçoit un courrier, il tente de reconnaître le nom du destinataire dans cette base et si c'est le cas de lui appliquer la transformation prescrite.
- /var/spool/mqueue. Dans ce répertoire sont stockés les mails en attente d'une délivrance. Il peuvent y rester plusieurs jours ,c'est un paramètre de la configuration du Sendmail .
- /var/mail/user Chaque utilisateur de la machine locale a une boîte aux lettres (``mail box'') repérée comme un fichier ayant comme nom son login.
  Par exemple /var/spool/mail/root.
  Ce fichier est mis à jour automatiquement par le MTA local en cas d'arrivée de courrier. De même que pour le répertoire de file d'attente, le répertoire des boîtes aux lettres des utilisateurs doit faire l'objet d'une attention particulière de la part de l'administrateur; la prolifération des ``documents attachés'' aux courriers électroniques est un fléaux contre lequel il est difficile de se prémunir sauf à agrandir perpétuellement la taille de la partition /var.

III- Configuration

III.1- Fichiers de configuration

Avec la version 8 de sendmail, nous avons deux niveaux de configuration; le premier avec le fichier **linux.mc**, qui est le fichier par lequel un utilisateur va configurer sendmail[10], et ensuite **/etc/sendmail.cf** qui est le fichier de configuration lu effectivement par sendmail généré à partir de linux.mc par le préprocesseur m4. Le fichier linux.mc est rédigé dans un format sensé être facile à comprendre, alors que sendmail.cf est écrit dans un langage hautement "complexe ". C'est ce dernier fichier qui était directement modifié avec les anciennes versions de sendmail.

III.2- Fichier de configuration sendmail.cf

Le fichier de configuration est organisé comme une série de lignes; le premier caractère de chacune d'elles en précise le type. Une ligne vide ou qui débute par un # est ignoré (commentaire), une ligne qui débute par un espace ou une tabulation est la continuation de la précédente. Et il contient :

- *Des définitions de variables*
  Les variables (*macros* en terminologie Sendmail) ont des noms formés d'un seul caractère. Les lettres minuscules ont une signification spéciale (ce sont les variables réservées de Sendmail), les lettres majuscules sont utilisables librement.

- *Des définitions de classes*
  Une classe est similaire à une variable, à ceci près qu'elle peut contenir plusieurs éléments, et que l'on peut faire des tests d'appartenance ou de non appartenance. Par exemple, une classe peut contenir l'ensemble des sites UUCP à qui nous transmettons le courrier, ou encore l'ensemble des machines pour lesquelles notre machine locale agit comme dépositaire de courrier.

- *Des définitions d'options*
  Les options modifient le comportement de sendmail. Il y a un grand nombre d'options (dont les noms sont une lettre minuscule ou majuscule) qui vont de la spécification du fichier aliases jusqu'au comportement de Sendmail en cas de serveur de noms défaillant.

- *Des définitions de priorités*
  Le champ d'en-tête Precedence est traité spécialement par Sendmail pour indiquer une priorité à donner au message. Ces priorités sont indiquées dans le message par un mot-clef, et la définition des priorités dans le fichier de configuration a pour but d'affecter une valeur numérique à chaque mot-clef.

- *Des définitions d'utilisateurs fiables*
  Les utilisateurs << fiables >> (*trusted users*) sont des utilisateurs ayant le droit d'appeler Sendmail avec l'option -f pour changer l'identité de l'expéditeur. Ceci correspond aux programmes qui envoient des messages pour le compte d'autres utilisateurs. Par exemple, lorsque uucp reçoit un message depuis un autre site UUCP, il appelle Sendmail pour router ce message. Le courrier ne doit alors pas apparaître comme venant de l'utilisateur

uucp, mais comme venant de l'utilisateur original sur le site distant. Autrement dit, cela permet d'assurer la transmission de l'enveloppe lorsque Sendmail est appelé par un autre programme.

- *Des définitions de champs d'en-tête*
  Lorsqu'un message est traité, Sendmail peut ajouter ou modifier certains champs de l'en-tête. Modifier les définitions des champs d'en-tête est un sport délicat qui n'est heureusement quasiment jamais nécessaire.

- *Des définitions de tables de correspondances*
  Il peut arriver qu'on ait besoin d'une table de correspondance (par exemple, pour transformer une adresse login@site en *Prénom.Nom@site*). Ces tables de correspondances, avec la version 8 de sendmail, sont stockées en format binaire db ou dbm, suivant les options de compilation de sendmail.

- *Des définitions d'agents de transport de messages*
  Les agents de transport de messages (les *mailers* en terminologie sendmail) ne sont a priori pas connus. Chaque agent de transport est défini par un nom, un programme à appeler et ses arguments, ainsi que des paramètres (taille maximum des messages, champs d'en-tête nécessaires, etc.).
  Trois agents au minimum doivent être connus :
  - local : utilisé pour la remise physique dans une boîte aux lettres ;
  - prog : utilisé pour la remise physique à un programme ;
  - error : (c'est le sendmail qui le définit automatiquement), mais il peut être utilisé dans le fichier de configuration pour renvoyer un message d'erreur lorsqu'une condition d'erreur a été détectée par les règles de réécriture.

- *Des règles de réécriture*
  Les règles de réécriture sont le cœur de la configuration de Sendmail.

III.3- Exemple de création d'un fichier /etc/sendmail.cf

Pour tout ce qui suit, nous supposons que le nom de notre machine est **L abo11**; le nom de domaine est **elec**, le nom de login est **Etudiant1** et que notre compte chez le fournisseur d'accès est **ENP2002**. Et la machine de fournisseur à laquelle nous envoyons notre courrier s'appelle **mail.com**

Avec l'utilisation des macros m4, la démarche est plutôt simple. Sous root, on crée un fichier linux.mc dans le répertoire /usr/lib/sendmail-cf/cf. Ce fichier se borne à donner des valeurs à certaines constantes et à définir certaines options. Enfin, nous compilerons ce fichier pour qu'il génère /etc/sendmail.cf Sous le compte root, puis déplaçons-nous vers le répertoire /usr/lib/sendmail-cf/cf et, à l'aide de l'éditeur de texte, on crée le fichier linux.mc suivant :

```
include(`../m4/cf.m4')dnl
OSTYPE(`linux')dnl
define(`SMTP_MAILER_FLAGS', `e9')dnl
FEATURE(redirect)dnl
```

```
FEATURE(nocanonify)dnl
FEATURE(always_add_domain)dnl
FEATURE(local_procmail)dnl
GENERICS_DOMAIN(Labo11.elec Labo11 localhost)
FEATURE(genericstable)
FEATURE(masquerade_envelope)dnl
define(`confCF_VERSION', `Sendmail.cf de Labo11 - 1/10/2002')dnl
define(`confCON_EXPENSIVE', 'True')dnl
define(`confME_TOO', `True')dnl
define(`confCOPY_ERRORS_TO', `Postmaster')dnl
define(`confDEF_CHAR_SET', `ISO-8859-1')dnl
define(`confMIME_FORMAT_ERRORS', `True')dnl
define(`SMART_HOST', `smtp8:[mail.com]')dnl
define(`confTO_QUEUEWARN', `24h')
MAILER(local)
MAILER(smtp)
```

Puis on va :

- créer le fichier /etc/genericstable suivant :

Etudiant1: ENP2002@mail.com
root: ENP2002@mail.com
news: ENP2002@mail.com
L'espace suivant ":" est un caractère de tabulation, pas un simple espace.

- modifier, ou créer le fichier /etc/nsswitch.conf pour que chacune de ses entrées ne contienne que l'option files, sauf l'entrée hosts qui contiendra files

- vérifier le fichier /etc/aliases et assurer qu'il contienne au moins les deux entrées suivantes :

MAILER-DAEMON: postmaster
postmaster: root

III.3.1- Quelques explications

La ligne **include(`../m4/cf.m4')** demande l'inclusion dans le fichier des macros m4, nécessaires au traitement.
La ligne **OSTYPE(`...')** provoque l'inclusion des spécificités d'un système d'exploitation particulier (ce qui se trouve entre ` et ' doit être un nom de fichier ayant l'extension .m4 et présent dans /usr/lib/sendmail-cf/ostypes/ : notamment le fichier linux.m4 définit le chemin permettant à sendmail de retrouver le programme procmail, utilisé pour délivrer localement le courrier.
Les lignes **define** servent à affecter des valeurs sous forme de chaînes à des variables qui seront prises en compte par sendmail.
Les lignes **FEATURE** servent à fixer certaines caractéristiques du fonctionnement de sendmail.
Les lignes **MAILER** servent à spécifier les protocoles utilisés pour transporter notre courrier.
Ici, on indique que l'on utilise le transport local et via smtp.

A partir de ce fichier, on peut maintenant générer notre fichier /etc/sendmail.cf. Plaçons nous dans le répertoire /usr/lib/sendmail-cf/cf et exécuton la commande :

```
m4 linux.mc > /etc/sendmail.cf
```

On vérifie à l'aide de la commande suivant ls -l /etc/sendmail.cf que ce fichier ait les permissions 600, qu'il appartient à l'utilisateur et au groupe root.

III.4- Démarrage le démon de sendmail

On peut utiliser les deux méthodes suivantes :

a- Démarrage manuel :

En tant que root, il suffit de taper la commande suivante :

```
# /sbin/init.d/sendmail start
```

Sendmail est alors en fonction.

b- Démarrage au démarrage du système:

Pour que le serveur soit chargé au démarrage du système, il faut modifier le fichier /etc/rc.conf en lui ajoutant :

```
# Lancement du serveur de messagerie Sendmail
SMTP="YES"
```

Une fois ceci fait, donc au démarrage du système, le serveur de messagerie est alors en fonction

VI- Conclusion

Dès qu'il est question de courrier sous Unix, Sendmail est le standard de facto, bien que certains lui préfèrent qmail ou smail. La conception de sendmail fait qu'il est capable d'administrer le courrier à l'échelle d'une structure tout en donnant la possibilité de ne l'utiliser que pour des besoins beaucoup plus modestes : une simple machine Linux reliée épisodiquement à l'Internet par une connexion PPP, par exemple ...

I- Introduction

FTP (File Transfer Protocol) [3] est l'ensemble des applications utilisées pour transférer des fichiers entre différentes machines sur l'Internet. La plupart des systèmes d'exploitation (comme UNIX, VMS, MS-DOS) possèdent une application appelée ftp permettant le transfert de fichiers. Il peut être utilisé à la fois pour l'envoie et la réception de fichiers entre différents ordinateurs.

Ftp fonctionne un peu comme la commande telnet. Par contre, contrairement au service Telnet, FTP peut donner un accès (restreint) à des machines où l'on n'a pas de compte utilisateur.

FTP a subi une grande évolution au fil des ans. Elle inclue la première proposition de mécanisme de transfert de fichiers de 1971 qui avait été développée pour une application sur les hôtes du M.I.T. (Massachusetts Institute of Technology).

Le File Transfer Protocol était désormais défini comme un protocole de transfert de fichiers entre des hôtes d'ARPANET(*Advanced Research Project Agency NETwork*) et dont la fonction première était définie comme le transfert efficace et fiable entre des hôtes pour profiter de l'utilisation d'une capacité de stockage de données distante, en proposant les services suivants :

- ✓ Promouvoir le partage de fichiers (programmes informatiques et/ou données).
- ✓ Encourager l'utilisation indirecte ou implicite (via des programmes) d'ordinateurs distants.
- ✓ Prémunir l'utilisateur contre les variations de formats de stockage de données entre les différents hôtes.
- ✓ Transférer les données d'une façon efficace et fiable.

FTP[5], bien que directement utilisable par un utilisateur depuis un terminal, est néanmoins conçu essentiellement pour être utilisé par des applications.

II- Théorie

II.1- TERMINOLOGIE

Contrôle d'accès : Le contrôle d'accès définit les privilèges utilisateur nécessaires pour utiliser un système, et pour accéder à des fichiers dans ce système. Le contrôle d'accès est nécessaire pour éviter un usage accidentel ou non autorisé de ressources fichiers. Il est dans les prérogatives d'un processus serveur FTP[24] d'invoquer ce contrôle d'accès.

Tailles d'octets : Deux tailles d'octets intéressent FTP : la taille des octets logiques du fichier, et la taille utilisée pour la transmission des données. La taille d'octet pour le transfert est toujours de 8 bits. Cette taille de transfert n'est pas nécessairement l'unité d'enregistrement logique du fichier dans le système, ni la taille des unités logiques permettant l'interprétation des structures de données.

Canal de contrôle : Le chemin de communication entre le USER-PI et le SERVER-PI pour l'échange de commandes et de réponses à commandes. Cette connexion utilise le protocole Telnet.

Canal de données : Une connexion bidirectionnelle (full duplex) sur laquelle les données sont transférées, dans un mode et sous un type particuliers. Les données transférées peuvent être une partie d'un fichier, un fichier entier, ou plusieurs fichiers.

Port de données : Un processus de transfert passif "écoute" sur le **port de données** un ordre de connexion de la part d'un processus de transfert actif émis dans le but d'ouvrir un canal de données.

DTP : Le processus de transfert de données DTP (data transfer process) procède à l'établissement et à la gestion de la connexion. Un DTP peut être passif ou actif.

End-of-Line : La séquence de fin-de-ligne qui définit la séparation entre deux lignes d'impression. Cette séquence est en général composée d'un Retour Chariot (CR = Carriage Return), suivi d'un saut de ligne (LF = Line Feed).

EOF : La condition end-of-file détermine la fin du fichier en cours de transfert.

EOR : La condition end-of-record marque la fin d'un enregistrement de données en cours de transfert.

Correction d'erreur : Une procédure qui permet à un utilisateur de se récupérer suite à certaines erreurs telles qu'une faute du système serveur ou du processus de transfert lui même. Pour FTP, la correction d'erreurs nécessitera un redémarrage de la transmission d'un fichier à partir d'un point de contrôle donné.

Commandes FTP : Un ensemble de commandes comprenant le contrôle des informations transitant entre le USER-FTP et le SERVER-FTP.

File : Une suite ordonnée (séquentielle) de données informatiques (y compris des programmes), d'une longueur arbitraire, et définies parfaitement par un "chemin d'accès".

NVT : Le Network Virtual Terminal défini par le protocole Telnet.

NVFS: Le Network Virtual File System. Un concept qui définit un système de fichiers standard vu à travers le réseau utilisant des conventions standardisées de commandes et de syntaxe de noms de chemins d'accès.

Page : Un fichier peut être structuré comme un ensemble de parties indépendantes appelées pages. FTP supporte la transmission de fichiers discontinus comme une suite de pages indexées indépendantes.

PI : Le Protocol Interpreter (interpréteur de protocole). Les côtés serveur (SERVER) et utilisateur (USER) d'un protocole ont des "rôles" distincts implémentés respectivement dans un SERVER-PI et un USER-PI.

Enregistrement : Un fichier à accès séquentiel peut être structuré comme un certain nombre de portions contiguës appelés enregistrements. Les structures en Enregistrements sont supportées par FTP bien qu'un fichier n'ait nul besoin d'être organisé de cette façon.

Réponse : Une réponse est un acquittement ou une dénégation envoyée par un serveur à l'utilisateur via la connexion de contrôle en réponse à une commande FTP. La forme générale d'une réponse est un code de résultat (pouvant être un code d'erreur) suivi d'une chaîne de caractères. Les codes sont à destination d'agents logiciels, le texte est plus naturellement destiné à des utilisateurs humains.

SERVER-DTP : Le processus qui transmet les données, dans son état "actif" normal, , établit le canal de données sur le port "en écoute". Il établit des paramètres pour le transfert et le stockage, et transfère les données sur commande de son PI. Le DTP peut entrer dans un état "passif" pour attendre, plutôt qu'initier une communication.

Processus serveur FTP : Un processus ou ensemble de processus qui prennent en charge la fonction de transfert de fichiers en coopération avec un processus USER-FTP et, certainement

un autre serveur. La fonction rassemble un interpréteur de protocole (PI) couplé à un processus de transfert de données (DTP).

SERVER-PI : L'interpréteur de protocole serveur "écoute" sur le Port L une communication arrivant d'un USER-PI et établit la connexion pour le canal de contrôle. Il reçoit par celui-ci les commandes FTP de l'USER-PI, y répond, et pilote le SERVER-DTP.

Type : Le type de représentation de données utilisé pour la transmission et le stockage. Le type implique certaines différences entre le temps d'enregistrement et le temps de transfert. Les types de représentation de données définis dans FTP sont décrits dans la Section traitant de l'établissement des canaux de données.

USER-DTP : Le processus de transfert de données "écoute" le port de données en attendant la connexion à un processus SERVER-FTP. Si deux serveurs transfèrent des données entre eux, le processus USER-DTP est inactif.

Processus USER-FTP : Un ensemble de processus et de fonctions incluant un interpréteur de protocole, un processus de transfert de données et une interface utilisateur par laquelle la fonction de transfert de fichier peut être effectuée en coopération avec un ou plusieurs processus SERVER-FTP. L'interface utilisateur met à disposition de l'utilisateur un langage local de commande-réponse.

USER-PI : L'interpréteur de protocole utilisateur instaure le canal de contrôle via son port U avec le processus SERVER-FTP, émet des commandes FTP, et gouverne le USERDTP si ce dernier est impliqué dans le processus de transfert.

II.2- Le modèle FTP

Avec les définitions ci-dessus à l'esprit, le modèle suivant (montré en Figure 1) peut être explicité pour la mise en oeuvre d'un service FTP.

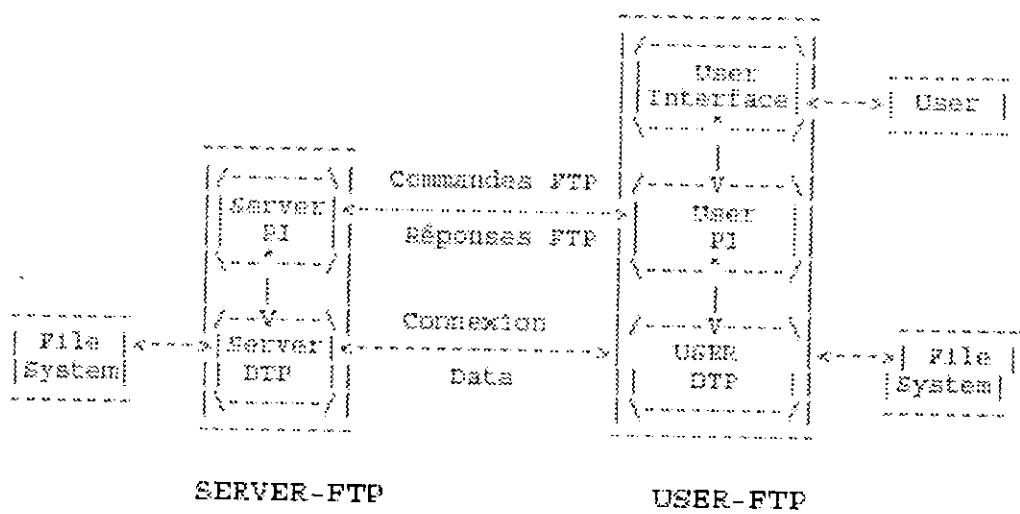


Fig. 1 Modèle d'usage de FTP

NOTES : 1. La connexion de données peut être utilisée dans les deux directions.
2. Il n'est pas nécessaire que le canal de données soit maintenu en permanence.

Dans le modèle décrit en Figure 1, l'interpréteur de protocole utilisateur (USERPI) établi le canal de contrôle. Ce circuit de communication utilise le protocole Telnet. A l'établissement de cette connexion, des commandes FTP standard sont générées par le USER-PI et transmises au processus serveur via le canal de contrôle. Des réponses standardisées sont émises en retour par le SERVER-PI au USER-PI via le canal de contrôle alors établie.

Les commandes FTP spécifient les paramètres du canal de données (port de données, mode de transfert, type pour la représentation, et structure des données) ainsi que la nature du fonctionnement des systèmes de fichiers (enregistrement, lecture, ajout, suppression, etc.). Le USER-DTP ou son délégué se mettra en "écoute" sur le port de données spécifié et le serveur établira le canal de données et effectuera le transfert de fichiers selon les paramètres spécifiés. Il doit être noté que le port de données n'est pas nécessairement sur le même hôte que celui qui a émis les premières commandes FTP par son canal de contrôle, bien que l'utilisateur ou le USER-FTP doive continuer à assurer "l'écoute" sur le port spécifié. Il doit être ici signalé en outre que le canal de données mis en place peut servir simultanément à la lecture et à l'écriture de données.

Une autre situation peut consister en un utilisateur qui souhaite transférer des fichiers entre deux hôtes, les deux étant des hôtes distants différents de celui de l'utilisateur. L'utilisateur établit alors un canal de contrôle vers chacun des deux serveurs et utilise ces canaux pour créer un canal de données entre ces deux hôtes.

De cette façon, les informations de contrôle passent par le USER-PI bien que les données soient transmises ente deux processus serveurs de transfert. Ce qui suit est un modèle de cette interaction entre serveurs.

Fig. 2 Modèle d'interaction entre serveurs

Le protocole demande à ce que les canaux de contrôle soient ouverts tant que dure le transfert de données. Il est de la responsabilité de l'utilisateur de demander la fermeture des canaux de contrôle lorsque l'utilisation du service FTP est terminée. C'est néanmoins le processus serveur qui prend en charge la rupture. Le serveur peut arrêter un transfert de données si le canal de contrôle est coupé sans commande préalable.

II.3- Fonctions de Transfert de donnes

Seul le canal de données permet le transfert effectif des fichiers. Le canal de contrôle n'est utilisé que pour le contrôle des commandes qui indiquent les fonctions qui doivent être exécutées ainsi que les réponses à ces commandes (voir la Section traitant des Réponses FTP). Plusieurs commandes concernent le transfert de données entre hôtes. Ces commandes de transfert incluent

la commande MODE qui spécifie comment les bits de données doivent être transmis ainsi que les commandes STRUcture et TYPE, qui sont utilisées pour définir la manière avec laquelle sont représentées les données. La transmission et la représentation sont des notions indépendantes. Cependant le mode de transmission "Stream" reste dépendant des attributs de structure des fichiers et si le mode de transmission "Compressed" est utilisé, la nature des octets de remplissage dépendra de la représentation des données utilisée.

II.3.1- Etablissement du canal de donnes

Le mécanisme de transfert de données consiste en l'établissement d'un canal de données entre les ports appropriés et de ce fait, en le choix des paramètres de transfert. Le USER et SERVER-DTP disposent tous deux d'un port de données par défaut. Le port "données" par défaut du processus utilisateur est identique à celui utilisé pour le contrôle de la connexion (c-à-d., U). Le port "données" par défaut du processus serveur est le port adjacent à celui utilisé pour le contrôle de la connexion (c-à-d., L-1).

La taille de l'octet transféré est toujours de 8-bits. Cette taille n'a de signification que pendant le processus effectif de transfert des données; elle ne présume en rien de la taille des unités logiques nécessaires pour représenter les données à l'intérieur du système. Le processus de transfert de données à l'état passif (ceci peut être un USERDTP ou un deuxième SERVER-DTP) devra "écouter" son port de données avant de pouvoir émettre une commande de requête de transfert. La commande FTP de requête de transfert détermine le sens du transfert de données. Le serveur, sur réception de la requête, établira la connexion au port "données". Lorsque cette dernière est établie, le transfert de données débute entre les deux DTP, et le SERVEUR-PI émet une confirmation à destination du USER-PI. Toute implémentation FTP doit accepter l'utilisation des ports par défaut, et seul le USER-PI peut invoquer une migration de la connexion vers des ports non standard.

Le processus utilisateur peut demander l'usage d'un autre port "données" par l'intermédiaire de la commande PORT. Par exemple, un utilisateur demande l'impression d'un fichier sur une imprimante en ligne TAC lequel fichier doit être récupéré depuis un troisième hôte. Dans le dernier cas, le USER-PI établit un canal de contrôle avec les deux SERVER-PI. Il est alors demandé à un serveur (par une commande FTP) "d'écouter" une connexion qu'une troisième entité va initier. Le USER-PI émet à destination d'un des SERVER-PI une commande PORT indiquant le port "données" de l'autre connexion. Enfin, il est envoyé aux deux serveurs les commandes de transfert appropriées. La séquence exacte de commandes et de réponses envoyées entre le contrôleur de l'utilisateur et les serveurs est définie dans la section traitant des Réponses FTP.

En général, il est de la responsabilité des serveurs de maintenir le canal de données actif, de l'initialiser et de le clore. L'exception à cette règle est lorsque le USER-DTP envoie des données dans un mode qui implique que la fin de fichier (EOF) correspond à la fermeture de la transmission. Le serveur DOIT fermer le canal de données sous les conditions suivantes :

1. Le serveur a terminé la transmission de données dans un mode ou la fin de fichier est signalée par une fermeture du canal.
2. Le serveur reçoit une commande ABORT de l'utilisateur.
3. La spécification du port "données" est changée par une commande de l'utilisateur.
4. Le canal de contrôle est fermé par une procédure normale ou pour une toute autre raison.

5. Une condition d'erreur irrécupérable est apparue.

Dans tous les autres cas, la fermeture est une prérogative du serveur. L'exercice de la quelle doit être signalée au processus utilisateur par un code de réponse 250 ou 226 seulement.

II.3.2- Gestion du canal de donnes

Ports de données standard : Toute implémentation FTP doit accepter l'usage des ports de données standard, seul un USER-PI pouvant initialiser un canal sur un port autre que standard. Négociation des ports autres que par défaut : Le USER-PI peut spécifier un port de données non standard à "viser" par le serveur via la commande PORT. Le USER-PI peut demander au serveur de s'identifier au serveur "cible" exprimé par ce port non standard via la commande PASV. La connexion étant définie comme une paire d'adresse, ces deux actions sont suffisantes pour obtenir à chaque fois un canal de données différent, bien qu'il soit admis de pouvoir déclencher deux fois ces commandes pour raccorder deux ports non standard à chaque extrémité d'un canal de données.

Réutilisation du canal de données : Lorsque le mode de transfert en "flux" est utilisé, la fin de fichier est indiquée implicitement par une fermeture du canal. Ceci pose un problème évident lorsque plusieurs fichiers doivent être transférés au cours de la même session, dans la mesure où TCP doit "bloquer" la connexion qui vient d'être utilisée pendant un certain temps fixé pour des raisons de fiabilité. De ce fait, une connexion ouverte sous ce mode ne peut pas être réutilisée immédiatement.

On donnera deux solutions à ce problème. La première est de négocier un autre canal sur des ports non standard. La première est de changer le mode de transfert. Commentaire sur les modes de transfert. Le mode de transfert en "flux" est par nature non fiable, dans la mesure où il est impossible de déterminer si un canal est fermé normalement ou non. Les autres modes de transfert (Bloc, Compressé) ne ferment pas le canal après transmission du fichier. Le niveau d'encodage de FTP est suffisant pour que le canal puisse être "surveillé" et que la fin de fichier puisse être RFC959 19 détectée. Ces modes sont donc tout à fait exploitables pour la transmission de multiples fichiers.

II.3.3- Modes de transmission

La considération suivante à prendre en compte pour transférer des fichiers est le choix d'un mode de transmission. FTP définit trois modes :

- un qui formate les données est permet de recommencer la transmission si nécessaire.
- une qui compresse en plus les données pour un transfert plus efficace .
- et un dernier mode qui laisse passer les données avec le moins d'encodage possible.

Dans ce dernier cas, le mode interagit avec les attributs de structure pour déterminer le type de traitement. En mode compressé, le type de représentation détermine essentiellement la nature du bourrage.

Tous les transferts de données doivent s'achever par la transmission d'une séquence de fin-de-fichier (EOF), laquelle peut être explicite ou implicitement déduite de la fermeture du canal. Pour les fichiers de structure de type "enregistrement", tous les marqueurs de fin d'enregistrement (EOR) sont explicites, y compris le dernier. Pour les fichiers transmis selon une structure de pages, la page de type "last-page" sera utilisée pour marquer la fin de la transmission.

Dans le reste de notre développement, octet signifiera "l'octet de transfert" sauf mention contraire explicite.

Dans le but d'obtenir un transfert standardisé, l'hôte émetteur devra traduire sa représentation interne d'une fin de fichier ou fin d'enregistrement dans la représentation préconisée par le protocole pour le mode de transfert et la structure de fichier donnés, l'hôte récepteur effectuant la transcription duale vers sa propre représentation interne. Le champ de comptage d'enregistrements d'un mainframe IBM peut ne pas être reconnu par un autre hôte, l'information de fin d'enregistrement devant alors être transféré comme un code de contrôle à deux octets en mode "flux" ou par marquage de bits dans les descripteurs des modes Bloc ou Compressé. La fin de ligne dans un fichier ASCII ou EBCDIC sans structure d'enregistrement devrait être indiqué par une séquence <CRLF> ou <NL>. Comme ces transformations impliquent un travail supplémentaire dans les hôtes, des systèmes identiques ou similaires s'échangeant des fichiers préféreront utiliser un transfert binaire dans un mode de type "flux".

Les modes de transmission suivants sont reconnus par FTP :

a- MODE "FLUX"

Les données sont transmises comme un flux d'octets. Il n'y a dans ce cas aucune restriction sur la représentation des données utilisée. Des structures type enregistrements sont autorisées.

Dans un fichier d'enregistrements, les séquences EOR et EOF seront toutes deux marquées par un code de contrôle à deux octets. Le premier octet vaudra 0xFF, le caractère d'échappement. Le second octet aura sont bit de poids faible à 1 et des 0 ailleurs pour la marque EOR (le second bit à 1 pour la marque EOF); en somme, l'octet aura la valeur 1 pour l'EOR et 2 pour l'EOF. EOR et EOF peuvent être marqué simultanément dans la dernière séquence en marquant les deux bits dans le même octet (donc, une valeur 3 pour le dernier enregistrement). Si un octet de données devait avoir la valeur OxFF, il devra être répété dans le second octet du code de contrôle.

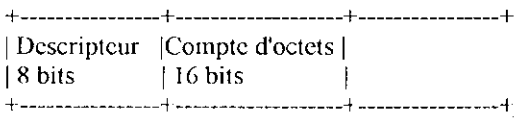
Si la structure est de type "fichier", la séquence EOF sera implicitement marquée par la fermeture du canal. Tous les octets transmis sont donc des octets de données.

b- MODE BLOC

Le fichier est transmis comme une suite de blocs de données précédés d'un ou plusieurs octets d'en-tête. L'en-tête contient un champ de comptage de blocs, et un code de description. Le champ de comptage indique la longueur totale du bloc de données en octets, et indique donc le début du bloc suivant (il n'y a pas de bits de bourrage). Le code de description indique : le dernier bloc du fichier (EOF) le dernier bloc de l'enregistrement (EOR), le marqueur de reprise (voir la Section
traitant de la Récupération d'Erreurs et Reprise de Transmission) ou de données suspectes (c-à-d. qu'il est possible que les données transférées soient erronées et non fiables). Ce dernier code N'EST PAS destiné à implémenter une fonction de contrôle d'erreur sous FTP. Il est motivé par la demande de certains sites d'échanger des classes particulières de données (ex., données sismiques ou météorologiques) en dépit d'erreurs locales qui peuvent survenir (telles que des erreurs de lecture sur des supports magnétiques), pour indiquer que certaines données transmises peuvent être suspectes pour des raisons autres que la transmission. Des structures type enregistrements sont admises dans ce mode, et toute forme de représentation de données peut être utilisée.

L'en-tête consiste en trois octets. Sur ces 24 bits d'information d'en-tête, les 16 bits de poids faible représentent le compte d'octets, les 8 bits de poids fort donnent le code de description selon les définitions ci-dessous.

Bloc d'en-tête



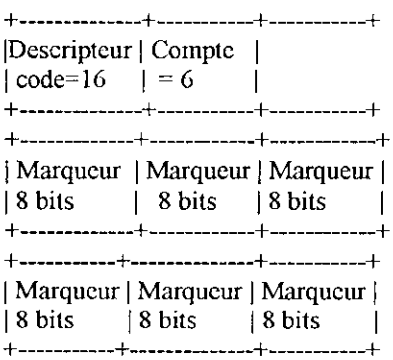
Le descripteur est formé de bits indicateurs. Quatre codes sont actuellement reconnus, dont le nombre représente la valeur décimale du masque.

- **Codes de description**

128 Le bloc est le dernier d'un enregistrement (EOR en fin de bloc)
64 Le bloc est le dernier de la transmission (EOF en fin de bloc)
32 Erreurs probables dans les données de ce bloc
16 Le bloc de données est le premier d'une reprise Grâce à cet encodage, plusieurs situations simultanées peuvent être codées dans un seul bloc. Autant de bits du descripteur que nécessaire peuvent être marqués.

Le marqueur de reprise est émis comme des données d'un multiple entier d'octets de 8-bits représentant des caractères imprimables selon le langage utilisé sur le canal de contrôle (ex., par défaut--NVT-ASCII). <SP> (Espace, dans le langage approprié) ne doit JAMAIS être employé dans un marqueur de reprise.

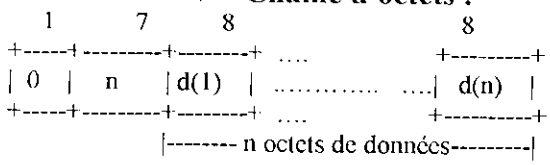
Par exemple, pour transmettre un marqueur de reprise de six caractères, la séquence suivante serait émise :



c- MODE COMPRESSE

Trois classes d'informations doivent être envoyées : des données "littérales", envoyées comme des chaînes d'octets; des données compressées, consistant en des octets "répliqués" ou des octets de bourrage; et des informations de contrôle, émis selon des séquences d'échappement à deux octets. Si n>0 octets (jusqu'à 127) littéraux sont émis, ces n octets doivent être précédés d'un octet dont le bit de poids fort est nul, les 7 autres bits contenant ce nombre n.

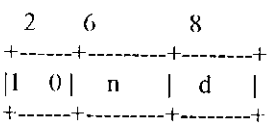
➢ **Chaîne d'octets :**



Chaîne de n octets de données d(1),..., d(n)
Le compte n doit être positif.

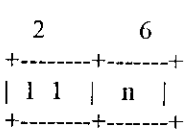
➢ **Octets répliqués :**

Pour compresser une chaîne comportant n répliques de l'octet de données d, les deux octets suivants sont émis :



➢ **Chaîne de bourrage :**

Une chaîne de n octets de bourrage peut être compressée en seulement deux octets, dans lesquels l'octet indiquant la valeur de bourrage change selon le type de représentation de données. Si le type est l'ASCII ou EBCDIC l'octet de bourrage est l'espace <SP> (ASCII code 32, EBCDIC code 64). Si le type est Image ou Local la valeur de bourrage vaut 0.



➢ **Séquence de contrôle :**

Une séquence de contrôle est un octet double, dont le premier est le caractère d'échappement (octet nul) et le deuxième contient les codes de description tels que définis dans le mode Bloc. Le descripteur a la même signification que dans le mode Bloc, et s'applique à la chaîne qui le suit. Le mode compressé est particulièrement utile pour gagner de la bande passante lors de transferts de gros volumes de données, et ce pour un coût CPU assez faible. Il peut être utilisé de façon très efficace pour transmettre des fichiers de sortie d'impression directement formatés.

II.3.4- Récupération d'erreur et reprise de transmission

Il n'existe pas de mécanisme permettant de détecter des bits perdus ou erronés d'un fichier transféré; ce niveau d'erreur est géré au niveau de TCP. Cependant, une procédure de reprise est prévue pour protéger les utilisateurs de défaillances majeures des systèmes (incluant le crash d'un hôte, d'un processus FTP, ou d'un protocole réseau sous-jacent).

La procédure de reprise n'est définie que dans les modes de transfert par bloc ou compressé. Elle demande à l'émetteur des données d'envoyer un marquer particulier dans le flux de données

incluant des informations de reprise. Ces informations du marqueur n'ont de signification que pour l'émetteur, mais doivent consister en des caractères imprimables au sens du langage utilisé

pour le contrôle de la connexion (ASCII ou EBCDIC). Le marqueur peut représenter un comptage de bits, d'enregistrements, ou tout autre information pouvant coder un "point de contrôle". Le récepteur des données, s'il implémente la procédure de reprise, notera la position de ce point au niveau de l'hôte récepteur, et renvoie cette information à l'utilisateur.

Dans le cas d'une faute système, l'utilisateur peut alors enclencher la procédure de reprise en notifiant le point de contrôle. L'exemple suivant illustre l'utilisation de la procédure de reprise. L'émetteur des données insère un bloc de marquage approprié dans le flux de données en un point donné. Le récepteur des données marque le point de contrôle dans son système de fichiers local et indique les derniers points émis et reçus à l'utilisateur, soit directement, soit en utilisant la réponse de code 110 du protocole de contrôle (suivant qui est l'émetteur). Lors d'une faute système, l'utilisateur ou le contrôleur requiert un nouveau transfert à partir du dernier marqueur en émettant le bloc de reprise avec ce marqueur comme argument. La commande de reprise est transmise via le canal de contrôle et est immédiatement suivi de la commande (telle que RETR, STOR ou LIST) qui était en exécution avant la faute système.

II.4- Fonction de transfert des fichiers

Le canal de communication entre le USER-PI et le SERVER-PI est établi comme une connexion TCP entre l'utilisateur et le port standard FTP du serveur.

L'interpréteur de protocole est responsable de l'émission des commandes FTP et de l'interprétation des réponses; le SERVER-PI interprète les commandes, envoie les réponses, et pilote le DTP pour établir le canal de données et transférer les fichiers.

Si le correspondant du processus de transfert (le processus passif) est un USERDTP, alors celui-ci est lui-même piloté par l'intermédiaire de l'interpréteur de protocole de l'hôte USER-FTP; s'il s'agit d'un second SERVER-DTP, alors son contrôle se fait via son propre PI sur commande du USER-PI. Les réponses FTP sont décrites dans la section suivante. Dans la description des quelques commandes de la section présente, il nous est apparu utile d'être explicite sur les réponses à attendre.

II.4.1- Commande FTP

Le protocole FTP suit les recommandations du protocole Telnet pour toutes les communications sur le canal de contrôle. Comme le langage choisi pour la communication sous Telnet peut être une option négociée, toutes les références dans les deux prochaines section se font par rapport au "langage Telnet" et le "code de fin de ligne Telnet" correspondant. De façon courante, on considérera qu'il s'agit du NVT-ASCII et de la séquence respective <CRLF>. Aucune autre spécification du protocole Telnet ne sera citée ici.

Les commandes FTP sont des chaînes de caractères "Telnet" terminées par le "code de fin de ligne Telnet". Les codes de commande sont eux-mêmes des caractères alphabétiques suivis du caractère <SP> (Espace) si d'autres paramètres suivent, et Telnet-EOL dans le cas contraire. Les codes et sémantique des commandes sont décrites dans cette section; la syntaxe détaillée est décrite dans la Section traitant des Commandes, les séquences de réponse sont explicitées dans la Section traitant du Séquencement des Commandes et Réponses, et les scénarios illustrant l'usage typique d'une commande sont donnés en Section traitant des Scénarios FTP Typiques.

Les commandes FTP peuvent être divisées en :

- commandes de contrôle d'accès,
- commandes de paramétrage de transfert,
- et des commandes de service FTP.

Certaines commandes (telles que ABOR, STAT, QUIT) peuvent être émises via le canal de contrôle y compris lorsqu'un transfert est en cours. Certains serveur ne pourront simultanément gérer le canal de contrôle et celui de données, auquel cas certaines actions spéciales devront être faites pour attirer l'attention du serveur. La procédure suivante doit être employée dans cet ordre:

1. Le système de l'utilisateur insère un signal "Interrupt Process" Telnet (IP) dans le flux Telnet.
2. Le système utilisateur envoie un signal "Synch" Telnet.
3. Le système utilisateur tente une commande d'avortement (ex., ABOR) dans le flux de commandes Telnet.
4. Le SERVER-PI, après réception de "IP", inspecte le flux Telnet en attendant EXACTEMENT UNE commande FTP.

II.4.2- Reponses FTP

Les réponses à des commandes FTP sont destinées à assurer une certaine synchronisation des actions impliquées dans un processus de transfert de fichiers, et garantir que le processus utilisateur puisse toujours connaître l'état du serveur.

Chaque commande suscite au moins une réponse, mais plusieurs réponses peuvent être données; dans ce dernier cas, les multiples réponses devront être aisément différentiables. De plus, certaines commandes peuvent être émises groupées en séquence, comme USER, PASS et ACCT, ou RNFR et RNTO. Les réponses témoignent de l'existence d'états intermédiaires si toutes les commandes passées sont exécutées avec succès. L'échec d'une seule étape nécessitera de recommencer toute la procédure.

Une réponse FTP répond consiste en un nombre à trois chiffres (transmis sous forme de trois caractères alphanumériques) suivis d'un texte. Le code numérique est à destination d'automates pour renseigner des dispositions à prendre et de l'état suivant de celui-ci; le texte est plutôt destiné à l'utilisateur humain. Les trois digits du code sont sensés contenir suffisamment d'information pour que le processus utilisateur (USER-PI) n'ait pas nécessité d'examiner la partie texte de la réponse, laquelle peut être soit éliminée, soit transférée à l'interface utilisateur, selon la nécessité. En particulier, le texte émis peut varier de serveur à serveur, et un automate pourrait donc avoir des difficultés à analyser tous les messages possibles.

Une réponse est définie comme contenant le code à 3-digits, suivi d'un Espace <SP>, suivie par une ligne de texte (lorsqu'une longueur maximale de réponse a été définie auparavant), et terminée par le code de fin-de-ligne Telnet. Il y aura des cas cependant, ou le texte sera plus long qu'une simple ligne. Dans ce cas, le texte entier aurait pu être mis entre crochets de sorte que le processus utilisateur puisse savoir quand s'arrête la lecture du texte (c-à-d. arrête l'analyse de l'entrée du canal de contrôle) pour passer à d'autres tâches. Ceci implique l'utilisation d'un format particulier sur la première ligne pour indiquer que d'autres lignes suivent, et un autre format particulier sur la dernière. Au moins une de ces lignes doit présenter le code de réponse. Pour satisfaire tous les avis sur le problème, il a été décidé que le code serait identique sur la première et la dernière ligne.

Ainsi, le format d'une réponse multilignes est tel que la première ligne débute par le code exact de la réponse, suivi d'un tiret "-" (Hyphénation ou "moins"), suivi du texte de la première ligne. La dernière ligne commencera par le même code, suivi immédiatement d'un Espace <SP>, éventuellement du texte, terminé par le code de fin-de-ligne Telnet.

Par exemple:

123-First line
Second line
234 A line beginning with numbers
123 The last line

Le processus utilisateur n'a plus qu'à chercher la deuxième occurrence du code de réponse suivie de l'Espace <SP> en début de ligne, et ignorer les lignes intermédiaires. Si une ligne intermédiaire commence par un nombre de 3-digits, le serveur ajoutera un espace en tête de ligne pour éviter toute confusion.

Ce schéma permet à des routines système standard d'être employées pour générer la réponse (ex. pour la réponse à la commande STAT), avec un marquage supplémentaire "artificiel" en tête de la première et la dernière ligne. Au cas (rare) ou ces routines seraient susceptibles de générer une ligne commençant par 3 digits suivi d'un espace, un caractère neutre (ex. Espace) sera rajouté en tête de chaque ligne.

Les trois digits de la réponse ont chacun une signification particulière. Ceci permet d'implémenter des traitements à réponse du plus simple au plus complexe dans l'USER-PI.

- Le premier digit indique si la commande se termine en succès, échec, ou est incomplète. (Rapport au diagramme d'état), un interpréteur de protocole simpliste pourra déterminer une stratégie d'action à lancer (telles que se retirer, tenter de nouveau, etc.) en se bornant à examiner ce digit. Un processus utilisateur désireux de savoir de quelle nature est l'erreur, (ex. erreur du système de fichiers, erreur de syntaxe dans la commande) pourra examiner le second digit, le troisième étant réservé au degré le plus fin de signalisation (ex., une commande RNTO sans commande RNFR antérieure).
- Le second digit donne une indication sur la nature de la réponse :
- Le troisième digit permet de qualifier encore plus finement les réponses dans chacune des catégories données par le deuxième digit.

II.5- Scénarios FTP tiypiques

Un utilisateur au port U voulant transférer ou recevoir des fichiers d'un serveur S:
En général, l'utilisateur communique avec le serveur via la médiation d'un processus USER-FTP. Ce qui suit peut être pris comme scénario typique. Les "prompts" USER-FTP sont montrés entre parenthèses, '---->' désigne une commande de l'utilisateur U vers l'hôte S, et '<----' désigne une réponse de l'hôte S à l'utilisateur U.

#COMMANDES LOCALES (Utilisateur) ACTION IMPLIQUEE
ftp (host) multics<CR> #Connexion à l'hôte S, port L,
#Etblissement du canal de contrôle.
<---- 220 Service ready <CRLF>.
username Doe <CR> USER Doe<CRLF>---->
<---- 331 User name ok,
need password<CRLF>.
password mumble <CR> PASS mumble<CRLF>---->
<---- 230 User logged in<CRLF>.
retrieve (local type) ASCII<CR>
(local pathname) test 1 <CR> #Le USER-FTP ouvre un fichier local en ASCII.
(for. pathname) test.pl1<CR> RETR test.pl1<CRLF> ---->
<---- 150 File status okay;
about to open data
connection<CRLF>.
#Le serveur établit le canal de données vers le port U.
<---- 226 Closing data connection,
file transfer successful<CRLF>.
type Image<CR> TYPE I<CRLF> ---->
<---- 200 Command OK<CRLF>
store (local type) image<CR>
(local pathname) file dump<CR> #Le USER-FTP ouvre le fichier local
#sous Image.
(for.pathname) >udd>cn>fd<CR> STOR >udd>cn>fd<CRLF> ---->
<---- 550 Access denied<CRLF>
terminate QUIT <CRLF> ---->
#Le serveur ferme toutes les connexions

II.6- Etablissement de la connexion

Le canal de contrôle FTP est établi via TCP entre le port U du processus utilisateur et le port L de l'hôte serveur. Au protocole FTP est en effet assigné le port 21 (25 octal), c'est à dire L=21.

III- Configuration

Dans la configuration de proftpd [14][10],il n'y a qu'un seul fichier qui rentre en ligne de compte, c'est : `/etc/proftpd.conf` .
C'est dans ce fichier que nous allons "tout" définir (enfin presque) concernant la configuration du serveur. Mais ce n'est pas ici que nous allons définir les utilisateurs qui ont accès ou non au serveur .

III.1- Les utilisateurs

Avec proftpd on a le choix dans sa stratégie. On utilise le fichier `/etc/ftpusers` pour définir tous les utilisateurs qui n'ont pas accès au service FTP. Tous les utilisateurs présents dans ce fichier ne pourront donc en aucun cas se connecter au service FTP.
On lit le fichier `/etc/ftpusers` comme il était en ajoutant tous les utilisateurs qui ont un shell (ksh, bash...) sur la machine. Car `proftpd` par défaut n'accepte pas la connexion si le shell de l'utilisateur n'est pas "valide" donc indiqué dans `/etc/shells`.

De plus on peut spécifiez lors dans la configuration de `proftpd.conf` si l'on souhaite que `proftpd` vérifie que les utilisateurs aient oui ou non un shell valide. Cela se fait par la directive : `RequireValidShell` (qui prend comme argument on ou off).
On crée un groupe unique qui servira à placer tous les utilisateurs avec un shell `/bin/false`, par exemple user1 et user2 et on les met dans le groupe ftp.

III.2- Le fichier Proftpd.conf

➢ Contexte de configuration Server Config

Tout d'abord le fichier `proftpd.conf` se divise en plusieurs parties, qui ne sont pas toutes nécessaires (vitales), on a donc plusieurs contextes de configuration :
server config, <Global>, <Anonymous>, <VirtualHost>, <Limit>, <Directory>, .ftpaccess
Donc pour résumé sur les contextes de configuration, on peut avoir un fichier proftpd.conf avec seulement le contexte : server config.

➢ Les directives les plus courantes

On commence par un fichier proftpd.conf ne contenant qu'une directive et le contexte de configuration obligatoire qu'est "server config".

```
# début du fichier profptd.conf
  UseFtpUsers off
# fin du fichier proftpd.conf d'exemple, pour info ce fichier en l'état n'est pas valide
```

On peut dire que tous les contextes de configuration sont forcément incluses dans le contexte "server config" et que l'on peut avoir un contexte comme <Directory> qui soit dans un "sous" contexte comme <Anonymous>. C'est un principe de configuration imbriquée.

Les premières directives que l'on va avoir dans le fichier `proftpd.conf` vont concerner le mode de lancement du serveur et les infos le concernant.

```
ServerName "Server FTP du Labo11"
ServerType standalone
DefaultServer on
```

ServerName => le nom du serveur ftp.
ServerType => est important car il indique que le serveur va démarré manuellement .
DefaultServer => cela est utile par ce que on faite des virtualhost, En fait cette directive vérifie

qu'elle configuration du serveur sera prise en compte (soit server config ou un des virtual hosts, ou anonyme....) .
Ensuite nous passons à des options dont l'utilité est très simple à comprendre :

```
AllowStoreRestart on
Port            21
Umask           022
MaxInstances 30
```

AllowStoreRestart => permet d'autoriser les clients à reprendre les uploads vers le serveur, ce n'est pas cette directive qui leur permet de reprendre quand ils téléchargent depuis le serveur. Port => difine le port TCP utilise par les client pour se connecte

`Umask` => c'est comme dans unix en général, 022 est une valeur qui est bien,
`MaxInstances` => comme c'est indiqué dans le fichier par défaut de `proftpd.conf`, cela sert à spécifier le nombre de processus fils maximum que va gérer (utiliser) `proftpd` .
Ensuite viennent 2 paramètres très important, on va indiquer sous quel utilisateur le serveur FTP est lancé, il ne s'agit donc pas de mettre root :
Le user `nobody` et le group `nogroup` sont les paramètres par défaut

Ensuite vient une option qui est très importante à ce qui veulent partager une petite connexion, il faut limiter le nombre de tentatives de logins :

```
MaxLoginAttempts 3
```

Pour donner d'info précise sur le serveur, l'option suivante soit à on:

```
DeferWelcome on
```

- **Contexte de configuration <Limit> appliqué à notre cas**

Maintenant on va indiquer un contexte de configuration Limit qui va s'appliquer au contexte de configuration server config (partie générale du serveur) plus concrètement, c'est à dire tous ceux qui vont se connecter et qui ne seront pas concernés par un virtualhost. En fait ce contexte de configuration est utile pour en limitant les possibilités (écriture, création de repertoires...)
avec la directives suivantes :

```
<Limit MKD RNFR RNTO DELE STOR >
DenyAll
</Limit>
```

Il est aussi possible d'utiliser des mots clefs comme `READ` et `WRITE` qui englobent plusieurs commandes, et vont limiter l'accès en lecture et en écriture.

- **Contexte de configuration Global**

Là on arrive à une section relativement importante, le contexte de configuration <Global> En effet ce contexte de configuration peut être utilisé à l'intérieur de la "server config" et du contexte de configuration <VirtualHost>
Tout ce qui va être défini dans <Global> va être appliquer à l'ensemble du contexte de configuration dans laquelle <Global> se trouve.

➢ **Contexte de configuration VirtualHost**

Premièrement il faut savoir que <VirtualHost> à la base est prévu pour un serveur par exemple qui pourra être accessible d'un côté par des personnes s'y connectant depuis internet et d'autres qui s'y connecteront depuis le réseau local et l'on souhaite que ceux qui s"y connectent depuis internet n'aient pas accès aux même données que ceux qui s'y connectent depuis le réseau local. Donc ces différents personnes vont se connecter sur le serveur en indiquant une adresse différente. Par exemple l'utilisateur A est chez lui sur internet, il veut se connecter, pour cela il indique l'adresse suivante : ftp.Labo11.A.elec (IP= 192.100.100.100)
Une autre personne, B, va elle se connecter sur ce serveur depuis le réseau local, elle va utiliser l'adresse interne qui sera l'adresse : ftp.Labo11.B.elec (IP= 192.65.146.119)
Ces 2 adresses IP dirigeant vers la même machine.
Il est aussi possible de faire plusieurs <VirtualHost> qui travaillent sur la même adresse IP mais qui ont un port différent ce qui peut être très pratique aussi.

➢ **Les autres contextes de configuration** <Anonymous> **et** <Directory>:

Le contexte de configuration <Anonymous> comme son nom l'indique sert à configurer un accès anonyme au service FTP et le contexte de configuration <Directory>, permet de définir un contexte pour les répertoires

III.3- Démarrage de démon de FTP

Pour le bon fonctionnement de FTP, nous devons démarrer le démon : proftpd , selon l'une des deux méthodes suivantes :

a- Démarrage manuel :

En tant que root, il suffit de taper la commande suivante :

```
# /sbin/init.d/proftpd start
```

FTP est alors prêt à accepter des connexions.

b- Démarrage au démarrage du système:

Pour que le serveur soit chargé au démarrage du système, il faut modifier le fichier /etc/rc.conf en lui ajoutant ceci:

```
# Lancement du serveur FTP
proftpd_enable="YES"
```

Une fois ceci fait, donc au démarrage du système, le serveur FTP est alors prêt à accepter des connexions.

III.4- Test du serveur FTP

L'utilisateur qui possède un compte personnel sur le serveur peut accéder à son compte librement et y faire toutes les manipulation qu'il désire (déposer et de charger des fichiers).
En revanche, nous avons crée un accès annonyme qui donne la possibilité à quiquonque de s'y connecter et de récupérer des fichiers. Mais ils ne pourront rien faire tant que nous n'aurons pas mis à leur disposition des fichiers dans le répertoire /var/ftp/pub.

I- Introduction

Samba est une suite d'applications Unix utilisant le protocole SMB[25] (Server Message Block). La plupart des systèmes d'exploitation, comme Windows et OS/2, prennent en charge les communications réseau à l'aide de SMB. Grâce à ce protocole, *Samba permet à des serveurs Unix de communiquer avec des produits Microsoft Windows*. Ainsi, une machine Unix dotée de Samba peut être vue comme un serveur de réseau Microsoft, en proposant les services suivants :

- Partage d'un ou de plusieurs systèmes de fichiers
- Partage d'imprimantes installées sur le serveur et sur ses clients.
- Assistance des clients dans l'exploration à l'aide du Voisinage réseau.
- Prise en charge de l'authentification de clients se connectant à un domaine Windows.
- Prise en charge de la résolution de serveurs de noms WINS ou assistance dans ce domaine.

Samba est l'enfant d'Andrew Tridgell, qui dirige actuellement l'équipe de développement de Samba (la fameuse Samba Team) depuis sa maison de Canberra, en Australie. Le projet est né en 1991 quand Andrew développa un programme de gestion de fichiers pour son réseau local prenant en charge le protocole de Digital Pathworks. Au fil des mises à jour, ce protocole allait devenir SMB.

Quelques années plus tard, il compléta son serveur SMB et le distribua sur Internet sous le nom de SMB Server. Toutefois, Andrew ne pouvait plus conserver le nom d'un produit appartenant déjà à une autre société. Ne sachant pas comment l'appeler, il essaya la méthode Unix de recherche de nom : grep -i 's.*m.*b' /usr/dict/words La commande produisit le résultat suivant : salmonberry samba sawtimber scramble « Samba » était né.

II- Théorie

II.1- Introduction à SMB

SMB est le protocole qu'emploient les machines Windows 95/98 et NT pour communiquer entre elles. À un niveau élevé, le protocole SMB est relativement simple. Ses commandes permettent d'effectuer toutes les opérations de fichiers et d'impression réalisables sur un disque ou une imprimante locale :

- Ouverture et fermeture de fichiers.
- Création et suppression de fichiers et de répertoires.
- Lecture et écriture d'un fichier.
- Recherche dans des fichiers.
- Gestion d'une file d'impression.

Chacune de ces opérations peut être codée dans un message SMB et transmise à un serveur (ou à partir d'un serveur). Les structures SMB sont une variante des structures de données des appels système DOS standard. Elles sont définies pour fonctionner entre machines d'un même réseau.

II.2- Format SMB

Un message émis par un serveur est le plus souvent une réponse à une requête envoyée par un client. Un message SMB n'est pas aussi complexe qu'on pourrait l'imaginer. Etudions la structure interne d'un tel message. Elle peut être divisé en deux :

- l'en-tête, de taille fixe.
- la chaîne de commande, dont la taille est fonction du contenu du message.

a- *Format d'en-tête SMB* :

Le tableau 1 présente le format d'un en-tête SMB. Les commandes SMB n'utilisent pas tous les champs de l'en-tête SMB. Par exemple, quand il tente de se connecter pour la première fois à un serveur, un client n'a pas encore reçu d'identificateur d'arborescence (TID) (attribué une fois que la connexion a réussi). Aussi un TID nul (0xFFFF) est-il placé dans son champ d'en-tête. Les champs inutilisés sont complétés par des zéros.

Champ	Taille (octets)	Description
0xFF 'SMB'	1	Identificateur de protocole.
COM	1	Code de commande, de 0x00 à 0xFF.
RCLS	1	Classe d'erreur.
REH	1	Réservé.
ERR	2	Code d'erreur.
REB	1	Réservé.
RES	14	Réservé.
TID	2	Identificateur d'arborescence (TID) : identificateur unique pour une ressource utilisée par un client.
PID	2	ID de processus de l'ordinateur appelant.
UID	2	Identificateur d'utilisateur.
MID	2	Identificateur multiplex, utilisé pour router des demandes dans un processus.

Tableau 1 : format d'un en-tête SMB

b-*Format de commande SMB* :

L'en-tête est suivi d'un nombre variable d'octets constituant une commande ou une réponse SMB. Chaque commande, comme Open File (ayant pour identificateur de champ COM, SMBopen) ou Get Print Queue (SMBsplretq), a son propre groupe de paramètres et de données. Comme pour les champs d'en-têtes SMB, certains champs sont facultatifs, selon la commande, par exemple, la commande Get Server Attributes (SMBdskattr) définit à zéro les champs WCT et BCC. Les champs du segment de commande sont répertoriés dans le tableau 2 :

Champs	Taille (octets)	Description
WCT	1	Nombre de mots de paramètres.
VWV	Variable	Paramètres (la taille est donnée par WCT).
BCC	2	Taille des données (en octets).
DATA	Variable	Données.

Tableau 2 : Contenu de la ligne de commande SMB

II.3- Exemple de connexion SMB simple

Etudions une connexion SMB simple. Les données techniques qui suivent ne sont pas indispensables à l'administration de Samba. Ces informations ont pour but de vous familiariser avec les négociations du protocole SMB avec d'autres ordinateurs du réseau. Pour établir une connexion à une ressource, le client et le serveur doivent effectuer les quatre opérations suivantes:

a- Etablissement d'une connexion virtuelle.
b- Négociation de la variante du protocole à utiliser pendant la session.
c- Définition des paramètres de session.
d- Etablissement d'une connexion d'arborescence à une ressource.

Nous examinerons chaque opération à l'aide d'un outil pratique ; le programme tcpdump disponible sur le site Web de Samba.

a- *Etablissement d'une connexion virtuelle :*

Lorsqu'un utilisateur adresse, pour la première fois, une requête d'accès à un disque réseau ou envoie un travail d'impression à une imprimante distante, NetBIOS (Network Basic Input Output System) crée une connexion dans la couche session, établissant ainsi un canal bidirectionnel virtuel entre client et serveur. Pour établir cette connexion, l'un et l'autre n'ont besoin, en réalité, que de deux messages ; ce que montre l'exemple suivant, capturé à l'aide de tcpdump :

```
>>> NBT Packet
 NBT Session Request
 Flags=0x81000044
 Destination =ESCRIME          NameType=0x20 (Server)
Source =WIZZIN                 NameType=0x00 (Workstation)

 >>> NBT Packet
NBT Session Granted
Flags=0x82000000
```

b- *Négociation de la variante du protocole* :

A cette étape, un canal est ouvert entre le client et le serveur. Le premier envoie un message au second pour négocier un protocole SMB. Comme nous l'avons vu, le client met son identificateur d'arborescence (TID) à zéro puisqu'il ne sait pas encore lequel utiliser.
Un identificateur d'arborescence est un numéro représentant une connexion à un partage sur un serveur. La commande du message est SMBnegprot, demande de négociation d'une variante du protocole pendant la session. Notons que c'est le client qui envoie au serveur une liste des variantes qu'il connaît, et non l'inverse. Le serveur répond à la demande SMBnegprot en indexant, à partir de 0, la liste des variantes proposées par le client. Si aucune variante n'est acceptable, la valeur 0xFF est renvoyée. Dans notre exemple, le serveur renvoie la valeur 5, ce qui signifie que le dialecte NT LM 0.12 va être utilisé jusqu'à la fin de la session :

```
>>> NBT Packet
NBT Session Packet
```

```
Flags=0x0
Length=154

SMB PACKET: SMBnegprot (REQUEST)
SMB Command = 0x72
Error class = 0x0
Error code = 0
Flags1 = 0x0
Flags2 = 0x0
Tree ID = 0
Proc ID = 5371
UID = 0
MID = 385
Word Count = 0
Dialect=PC NETWORK PROGRAM 1.0
Dialect=MICROSOFT NETWORKS 3.0
Dialect=DOS LM1.2X002
Dialect=DOS LANMAN2.1
Dialect=Windows for Workgroups 3.1a
Dialect=NT LM 0.12

>>> NBT Packet
NBT Session Packet
Flags=0x0
Length=69

SMB PACKET: SMBnegprot (REPLY)
SMB Command = 0x72
Error class = 0x0
Error code = 0
Flags1 = 0x0
Flags2 = 0x1
Tree ID = 0
Proc ID = 5371
UID = 0
MID = 385
Word Count = 02
[000] 05 00
```

c- *Initialisation de la session et des paramètres de connexion* :

Nous allons maintenant transmettre des paramètres de session et de connexion. Ils contiennent le nom du compte et, le cas échéant, le mot de passe, le nom du groupe de travail. La taille maximale des données qui peuvent être transférées et le nombre de demandes qui peuvent être simultanément en souffrance dans la file d'attente.

Dans l'exemple suivant, la commande Session Setup véhicule une commande SMB supplémentaire. Adjointe au nom de commande, la lettre X caractérise cette propriété. Le code hexadécimal de la seconde commande est indiqué dans le champ Com2. Dans ce cas, la

commande 0x75 correspond à l'exécution combinée de Tree Connect et X. Le message SMBtconX recherche le nom de la ressource dans le tampon (ou buffer) smb_buf. (Il s'agit du dernier champ de la demande suivante.) Dans cet exemple, smb_buf contient la chaîne

\\LABO11\PUBLIC, correspondant au chemin d'accès complet à un répertoire partagé sur le nœud LABO11. Les commandes de type « X » accélèrent les transactions puisqu'une demande peut être faite avant l'émission du résultat de la précédente. Notons que le TID est toujours à zéro. Le serveur fournit un TID au client après l'ouverture de session et l'établissement d'une connexion à la ressource demandée. Le mot de passe est envoyé lors de l'ouverture de connexion. Pour éviter cette brèche de sécurité, nous recourrons ultérieurement aux mots de passe chiffrés :

```
>>> NBT Packet
NBT Session Packet
Flags=0x0
Length=139

SMB PACKET: SMBsesssetupX (REQUEST)
SMB Command = 0x73
Error class = 0x0
Error code = 0
Flags1 = 0x10
Flags2 = 0x0
Tree ID = 0
Proc ID = 5371
UID = 1
MID = 385
Word Count = 13
Com2=0x75
Res1=0x0
Off2=106
MaxBuffer=2920
MaxMpx=2
VcNumber=0
SessionKey=0x1FF2
CaseInsensitivePasswordLength=1
CaseSensitivePasswordLength=1
Res=0x0
Capabilities=0x1
Pass1&Pass2&Account&Domain&OS&LanMan= KRISTIN PARKSTR Windows 4.0
    Windows 4.0
PassLen=2
Passwd&Path&Device=
smb_bcc=22
smb_buf[]=\\LABO11\PUBLIC
```

d- *Connexion à une ressource* :

Pour terminer, le serveur renvoie un TID au client, indiquant que l'utilisateur dispose d'un accès autorisé et que la ressource est disponible. Il initialise le champ ServiceType à la valeur
« A », ce qui correspond à un service de fichier. Les types de service sont les suivants :

- « A » pour un disque ou un fichier.
- « LPT1 » pour une impression placée dans un tampon.
- « COMM » pour une imprimante ou un modem connecté en direct.
- « IPC » pour un tube (ou canal) nommé.

Le résultat est le suivant :

```
>>> NBT Packet
NBT Session Packet
Flags=0x0
Length=78
SMB PACKET: SMBsesssetupX (REPLY)
SMB Command = 0x73
Error class = 0x0
Error code = 0
Flags1 = 0x80
Flags2 = 0x1
Tree ID = 121
Proc ID = 5371
UID = 1
MID = 385
Word Count = 3
Com2=0x75
Off2=68
Action=0x1
[000] Unix Samba 1.9.1
[010] PARKSTR

SMB PACKET: SMBtconX (REPLY) (CHAINED)
smbvwv[]=
Com2=0xFF
 Off2=78
smbbuf[]=
 ServiceType=A
```

Un TID ayant été attribué, le client peut exécuter n'importe quelle commande comme à partir d'un disque dur local. Il peut ouvrir des fichiers, les lire et y écrire des données, les supprimer, créer des fichiers, rechercher des noms de fichiers, etc.

III- Configuration

III.1- Fichier de configuration de base

Samba a un atout majeur. Ses paramètres de configuration [10]sont regroupés dans un seul et même fichier : **smb.conf**. Ce fichier peut être très simple ou, au contraire, extrêmement complexe.
Voici un exemple de fichier de configuration de Samba. Il ressemble à la structure des fichiers de paramétrage .INI de Windows :

```
[global]
log level = 1
max log size =1000
socket options = TCP_NODELAY IPTOS_LOWDELAY
guest ok = no
[homes]
browseable = no
map archive = yes
[printers]
path = /usr/tmp
guest ok = yes
printable = yes
min print space = 2000
[test]
browseable = yes
read only = yes
guest ok = yes
path = /export/samba/test
```

Ce fichier de configuration définit tout d'abord un journal de débogage de base d'une taille maximale d'un méga-octet, optimise les connexions TCP/IP entre Samba et les clients SMB et permet de créer un partage de disque pour tout utilisateur possédant un compte Linux sur le serveur. En outre, chaque imprimante déclarée sur le serveur ainsi qu'un partage en lecture seule associé au répertoire /export/samba/test seront accessibles à partir des postes clients. La dernière partie du fichier est similaire au partage de disque utilisé lors des essais de Samba .

III.2- Structure du fichier de configuration

Examinons la structure de fichier de configuration :

```
[global]
   ...
[homes]
   ...
[printers]
   ...
[test]
```

Les noms figurant entre crochets délimitent des sections (ou rubriques) uniques dans le fichier smb.conf ; chaque nom de rubrique correspond à un nom de partage (ou de service). Ainsi, les rubriques [test] et [homes] désignent chacune un partage de disque unique. Elles contiennent des options associées à des répertoires spécifiques du serveur Samba. Le partage [printers] contient des options propres aux différentes imprimantes du serveur. À l'exception de la section [global], toutes les rubriques du fichier smb.conf définissent des partages de disque ou d'imprimantes accessibles aux clients ouvrant une session avec le serveur Samba. Les autres lignes sont des options de configuration propres au partage considéré. Le contenu d'une rubrique est compris
entre le titre entre crochets et le titre suivant ou bien, s'il s'agit de la dernière rubrique, la fin du fichier de configuration. Chaque option de configuration se présente sous la forme suivante :

option = valeur

Une valeur est attribuée à chaque option du fichier smb.conf. Certains noms d'option de Samba n'ont pas été choisis de manière claire. Ainsi, si read only (lecture seule) est un nom suffisamment clair par lui-même, ce n'est pas le cas de l'ancienne option « public » pour laquelle on a créé le synonyme plus précis, guest ok .
Si on a créé le fichier de configuration manuellement, il est vivement recommandé de le tester pour en corriger les erreurs éventuelles. L'analyseur **testparm** examine la syntaxe du fichier : smb.conf et, le cas échéant, la liste des erreurs détectées pour tel ou tel service actif. Si tout est correct, on démarre les démons du serveur.

III.3- Démarrage des démons de Samba

Pour le bon fonctionnement de Samba, vous devez démarrer deux démons : smbd et nmbd, selon l'une des trois méthodes suivantes :

a- *Démarrage manuel* :

En tant que root, il suffit de taper les commandes suivantes :

```
# /usr/sbin/smbd –D
# /usr/sbin/nmbd –D
```

Samba est alors prêt à accepter des connexions.

b- *Démarrage à l'aide d'un script* :

Pour lancer les processus de Samba comme démons autonomes, on ajoute à nos scripts de démarrage Linux standard, les commandes de la section précédente. L'opération varie selon les distributions.

c- *Démarrage à partir du démon Inetd* :

Inetd est, en quelque sorte, un « super démon » Internet pour système Unix. À l'écoute des ports TCP définis dans /etc/inetd.conf, il exécute à la demande, le programme approprié. L'avantage d'un tel mécanisme est que les démons prêts à répondre à des demandes peuvent être nombreux sans, pour autant, être nécessairement actifs en même temps. Le démon inetd écoute les demandes à la place des autres. La charge entraînée par la création d'un nouveau processus est, par conséquent, modeste, mais on doit éditer deux fichiers de configuration au lieu d'un seul. Cette méthode est pratique si le serveur ne comporte qu'un ou deux utilisateurs ou est surchargé de démons. Il est également aisé d'effectuer une mise à niveau sans gêner une connexion existante. Pour démarrer les démons de Samba à partir de inetd, on ouvre /etc/services à l'aide d'un éditeur de texte. Si elles ne sont pas encore définies, on introduit les deux lignes suivantes :

netbios-ssn 139/tcp
netbios-ns 137/udp

On éditez ensuite /etc/inetd.conf. Recherchons les deux lignes suivantes et si elles n'existent pas dans le fichier, on les ajoute. On s'assure que les chemins d'accès à smbd et à nmbd sont corrects. La syntaxe de ce fichier peut varier légèrement en fonction de la variante Unix du serveur. On s'inspire des lignes suivantes :

netbios-ssn stream tcp nowait root /usr/local/samba/bin/smbd smbd
netbios-ns dgram udp wait root /usr/local/samba/bin/nmbd nmbd

Enfin, on arrête les éventuels processus smbd ou nmbd, puis on envoit un signal de déconnexion (HUP) au processus inetd. À la réception du signal HUP, inetd relit son fichier de configuration. Pour rechercher l'ID du processus, on emploiela commande ps, puis la commande suivante :

kill -HUP id_processus

Samba est maintenant disponible et actif

IV- Conclusion

Samba reste un outil étroitement lié spécialement au protocole SMB ; donc généralement aux machines dotées du système d'exploitation microsoft. Il nous résout d'une manière transparente le problème de compatibilité sur des environnements hétérogènes unix-microsoft.

Conclusion

L'objectif de ce travail est d'étudier le fonctionnement des réseaux informatiques dans le cas général et de l'Intranet en particulier; la compréhension des mécanismes fonctionnels des différents protocoles mis en jeux était un préalable essentiel.

L'implémentation est faite à l'aide de logiciels dits "open source". On cite le système d'exploitation Linux et comme services Apache et Sendmail par exemple. Ce choix est justifié par le faite que:

- Ils sont gratuits et faciles à obtenir.
- Le source et la documentation sont entièrement disponibles sur Internet.
- Dans le cadre pédagogique, le fait qu'il sont open source nous permet d'enlever la contrainte de l'opacité fonctionnelle; ce qui nous permet de mieux comprendre.
- Ils sont est très performants, de qualité et il répond au exigences de sécurité mieux que les logiciels propriétaires.

Du point de vue de l'implémentation, on a réussi à mettre en œuvre tous les services.

Finalement, il est à mentionner à notre sens que les réseaux informatiques sous "open source" constituent un environnement attrayant , passionnant et en plus très prometteur.

BIBLIOGRAPHIE

✓ Ouvrages

1. D.Dromard F.ouzzani D.Seret ,**Réseaux informatiques Cours et exercices,** tome 1 et 2. 1995 ,Eyrolles.
2. Larry L.Peterson Bruce S.Davie ,**Réseau d'ordinateurs,**.1998,VUIBERT.
3. H.Holz B.Schmitt A.Tikart ,**Internet et Intranet sous Linux** ,1999, EYROLLES.
4. G.Tackett D.Gunter L.Brown ,**Linux** ,1995, LE MACMILLAN.
5. F.Alin D.Lafont G.Françions Macary ,**Le Projet Intranet** ,1999,EYROLLES.
6. Rich Bowen Ken Coar ,**Apache Serveur** ,2000 ,CAMPUS PRESS.
7. Ben Laurie Peter Laurie ,**Apache** ,2000 ,OREILLY.
8. S.Spainbour R.Eckstein ,**Webmaster in a nutshell** ,1999 ,OREILLY.

✓ Sites Internet

9. www.suse.com
10. www.linux.org/docs/ldp/howto/HOWTO-INDEX/howtos.html
11. www.freenix.fr/unix/linux/
12. www.linux-france.org/article/index.shtml
13. www.abul.org/infos/doc.php3
14. www.ze-linux.com
15. www.linux-center.org/fr/
16. www.linuxfr.org
17. www.linux.fr
18. www.gershwin.ens.fr/vdaniel/Informatique/Linux/Linux-Doc.html
19. www.linux-france.org/article/web/egraffin/
20. www.apache.org
21. http://dyade.inrialpes.fr/~freyssin/cours/index.html
22. www.info.univ-angers.fr/Reseaux/Intro/Reseau-2
23. www.Sendmail.org
24. www.Proftpd.org
25. www.tldp.org/HOWTO/SMB-HOWTO.html
26. www.Linux-center.org/fr/