POLYTECH
NANTES

École d'ingénieurs de l'université de Nantes

2014-2015

Spécialité « Informatique »

Réseaux SILR4 / S7

Architectures et protocoles Internet

Rémi Lehn

— 4e année —

UNIVERSITÉ DE NANTES

POLYTECH
Premier réseau national
des écoles d'ingénieurs
polytechniques des universités

Site de la Chantrerie
Rue Christian Pauc - BP 5060
44306 Nantes cedex 3
France
tél. +33 (0)2 40 68 32 00
fax. +33 (0)2 40 68 32 32

Site de Gavy
Gavy Océanis - BP 152
44603 Saint-Nazaire cedex
France
tél. +33 (0)2 40 90 50 30
fax. +33 (0)2 40 90 50 24

www.polytech.univ-nantes.fr

Table des matières

Introduction

Ce cours décrit l'architecture de réseaux IP. Ce protocole est aujourd'hui le protocole le plus utilisé pour la construction de réseaux locaux, de réseaux d'entreprises, de réseaux d'opérateurs ainsi que dans sa vocation initiale, c'est-à-dire, l'interconnexion de réseaux pour la constitution d'un réseau global : Internet [1].

Le terme *intranet* désigne généralement, par raccourci, l'utilisation du protocole *IP*, ainsi que d'autres protocoles ou applications issues de l'Internet, dans le cadre d'un réseau d'entreprise. La différence essentielle entre un tel réseau d'entreprise et l'Internet est que dans le cas d'un réseau d'entreprise, le nombre de nœuds et d'utilisateurs du réseau est a priori connu, de même que les fonctionnalités des applications déployées et leurs accès [2].

Les caractéristiques les plus générales du protocole IP sont la technique de *commutation de paquets*, un fonctionnement asynchrone et sur la base du meilleur effort de chaque élément du réseau, sans garantie de service.

1.0.1 Historique

Historiquement, l'Internet fut d'abord conçu comme un réseau global, avant d'être adopté comme un protocole réseau universel.

L'ARPA

C'est dans le contexte de la bataille technologique qui opposa américains et soviétiques durant la guerre froide [3] qu'Eisenhower, président des États-Unis, crée en 1957, l'*ARPA* (*Advanced Research Project Agency*) [4], dépendant du *DoD* (*Department of Defense*). L'objectif de l'*ARPA* était le développement de technologies spatiales. Lors de la création de la *NASA*, l'année d'après, l'*ARPA* se trouve un nouveau centre d'intérêt : l'informatique, qui est, à cette époque, une technologie naissante.

En 1961, l'*ARPA* dispose d'un (et d'un seul) gros ordinateur, un IBM Q-32, situé à l'Université de Californie. Jack Ruina, directeur de l'*ARPA* recrute Joseph Licklider du *Massachusetts Institute of Technology* (*MIT*), spécialiste des ordinateurs, afin d'exploiter le Q-32 à autre chose que du calcul numérique. Licklider, psychologue de formation, est convaincu que l'ordinateur pourrait permettre aux hommes de communiquer leurs connaissances et leurs ressources, à travers un *réseau informatique*.

Bob Taylor, succédant à Licklider, est nommé directeur de l'*IPTO* (*Information Processing Techniques Office*, branche de l'*ARPA* chargée des recherches en informatique) en 1966. Il lance un

1. Les initiales de ce protocole signifient d'ailleurs *Internet Protocol*.
2. bien sûr, le nombre d'utilisateurs, de machines et les périmètres fonctionnels des applications de l'entreprise sont extensibles.
3. les soviétiques lancent le *Spoutnik* en 1957, les américains créent la *NASA* en 1958 qui fera marcher un homme sur la lune en 1969.
4. L'*Advanced Research Project Agency* (*ARPA*) s'est rebaptisée *Defense Advanced Research Project Agency* (*DARPA*) en 1971, puis à nouveau *ARPA* en 1993 ; depuis 1996, elle s'appelle à nouveau *DARPA*.

programme de développement d'un réseau d'ordinateur, afin de concrétiser les idées de son prédécesseur. Ce programme reçoit un financement de 19 millions de dollars. Bob Taylor réunit une équipe d'experts en informatique, dirigée par Larry Roberts.

La commutation par paquets

Dans les années 60, Larry Roberts et son équipe définissent un *protocole de communication*, afin que les ordinateurs, ne représentant (à l'époque) pas tous l'information de la même façon, puissent dialoguer en un langage commun.

Un autre problème est que l'information doit pouvoir être retransmise par des noeuds intermédiaires (des *routeurs*) du réseau, dans le cas ou émetteur et récepteur sont éloignés. Cette retransmission doit être fiable –l'*ARPA* est une organisation militaire, la sécurité et la fiabilité sont donc des données du problème– malgré les pannes fréquentes des ordinateurs à l'époque.

Bob Taylor et Larry Roberts décident que l'architecture la plus fiable pour leur réseau est l'architecture *distribuée* ; ils imaginent un réseau dans lequel plusieurs chemins pourraient exister entre l'émetteur et le récepteur, et les nœuds intermédiaires sur chacun des chemin pourraient "décider" eux-mêmes, du meilleur chemin à prendre. Cette solution a pour avantage que si un des chemins possibles s'interrompt, une bifurcation peut être prise. Ainsi, le réseau reste opérant même si une partie de celui-ci tombe en panne. C'est "le réseau qui résiste à une attaque nucléaire" [5].

Leonard Keinrock, du *MIT*, inventeur de la théorie de la *commutation de paquets* au début des années 60 finit par convaincre Larry Roberts et son équipe à utiliser la commutation de paquets plutôt que la *commutation de circuits* (sur laquelle repose le téléphone, notamment) pour rendre opérationnelle son idée de réseau distribué. Afin de justifier les arguments de la commutation de paquets contre la commutation de circuits, en 1965, Larry Roberts organisa avec Thomas Merrill du *MIT*, la connexion du Q-32 de l'*UCLA* avec le TX-2 du *MIT* par l'intermédiaire d'une ligne téléphonique [6]. L'expérience démontra que les ordinateurs fonctionnant en temps partagé pouvaient lancer des programmes et récupérer les données nécessaires sur une machine distante, mais, que la technique de commutation de circuits représentée par la ligne de téléphone reliant les deux ordinateurs n'était pas adaptée à la construction d'un réseau informatique. La conviction de Leonard Keinrock était confirmée.

Ainsi, les messages constituant les communications entre ordinateurs seraient découpés en paquets de taille fixe, acheminés par des ordinateurs reliés entre eux. Afin de ne pas surcharger les ordinateurs connectés, les opérations sur les paquets (découpage en paquets, ré-assemblage de paquets, acheminement des paquets (*routage*)) sont effectués par des ordinateurs dédiés à cette tâche (*IMP* ou *Interface Message Processors*). En 1968 l'*ARPA* lança un appel d'offre pour développer l'architecture logique des *IMP*.

L'ARPANET

Une petite société conseil, *Bolt Beraneck & Newman* (*BBN*) de Cambridge, Massachusetts, fut choisie par l'*ARPA* pour concevoir les *IMP*. Franck Heart, spécialiste de l'informatique temps réel et Bob Kahn, mathématicien, spécialiste de la communication de l'information, tous deux de *BBN*, réalisèrent, avec un budget d'un million de dollars attribué par l'*ARPA*, quatre *IMP* : un pour l'Université de Californie à Los Angeles (*UCLA*) (livré le 1^{er} Septembre 1969), un pour le *Stanford Research Institute* (*SRI*) (livré le 1^{er} Octobre 1969), un pour l'Université de Californie à Santa Barbara (*UCSB*) (livré le 1^{er} Novembre 1969), un pour l'Université de l'Utah (livré le 1^{er} Décembre 1969).

Les quatre universités avaient pour mission de développer les interfaces entre les *IMP* et leurs ordinateurs. Des étudiants de ces quatre universités, (parmi eux, un certain Vinton Cerf), se regroupèrent pour former le *Network Working Group* (*NWG*), dirigé par Steve Crocker, de l'*UCLA*. Parmi les tâches du *NWG*, la plus urgente était de développer le protocole de communication

5. La rumeur selon laquelle l'*ARPANET* et par la suite l'*InterNet* ont été construits pour résister à une attaque nucléaire est cependant fausse, [Leiner et al., 1998].

6. Ce qui représente le premier réseau informatique inter-métropolitain jamais construit

entre les ordinateurs des universités et les *IMP*. Ils réalisèrent le *Network Control Protocol* (*NCP*), achevé fin 1970 à cette fin.

Fin Décembre 1969, ce réseau, baptisé *ARPANET*, relie entre elles les quatre universités disposant d'un *IMP*. *ARPANET* est le premier réseau informatique utilisant la commutation de paquets. Fin 1970, *ARPANET* connecte une dizaine de sites, puis une vingtaine en 1971, une trentaine en 1972... D'autres réseaux, fonctionnant eux aussi sur le principe de la commutation de paquets sont créés parallèlement dans d'autres pays (exemple du projet *Cyclades*, créé par Louis Pouzin, en France, en 1972, qui reprenait les mêmes principes qu'*ARPANET*).

En Octobre 1972, Bob Kahn, recruté à la *DARPA*, présente pour la première fois une démonstration de l'*ARPANET* au public lors de la Conférence Internationale sur la Communication en Informatique (*International Computer Communication Conference*, (*ICCC*)). Il présenta, lors de cette conférence, la première application importante du réseau *ARPANET* : le courrier électronique (*e-mail*).

Les deux principales utilisation du réseau *ARPANET* sont, à cette époque, le courrier électronique et les forums de discussion. En 1973, le courrier électronique représente les trois quarts du trafic du réseau. Le *SRI* développe le (premier) *Network Information Center* (*SRI-NIC*), dont le rôle est de maintenir *en ligne*[7], les correspondances entre les noms de machine et les adresses réseau, ainsi que les *Request For Comments* (*RFC*), documents décrivant les protocoles des réseaux *ARPANET* puis *InterNet*[8].

L'Internetting Project

La *DARPA* crée en 1973 l'*Internetting Project*, dans l'objectif de regrouper tous les réseaux émergeants en un réseau international[9]. Parallèlement, Vinton Cerf (du *NWG*) crée l'*International Network Working Group*.

C'est sous l'influence de Bob Kahn (*DARPA*) et de Vinton Cerf (*SRI*) que l'*ARPANET* évoluera en l'*Internet*. Même s'il n'y a pas de date de naissance officielle de l'*Internet*[10], un certain nombre d'évolutions caractérisent *Internet*, et notamment l'ouverture du réseau : l'*Internet* doit pouvoir interconnecter différents réseaux, sans que l'architecture ou la conception de ceux-ci aient à être modifiées. De nombreux réseaux, par exemple, *RPNET*, un réseau utilisant des ondes hertziennes pour acheminer les données (*Packet Radio*), *SATNET*, utilisant des ondes hertiennes relayées par satellite et plus tard, *CSNET* (1983) de la *National Science Fundation*, seront reliés à *Internet*.

Internet

En 1973, *NCP* accusait un certain nombre de lacunes pour servir de protocole pour *Internet* : l'impossibilité d'adresser directement les ordinateurs (seuls les *IMP* pouvant être adressés), aucun contrôle d'erreur entre les ordinateurs connectés (seulement un contrôle d'erreur dans les liaisons *IMP-IMP*), aucune gestion des paquets perdus, et surtout, aucune ouverture vers d'autres protocoles. Afin de combler ces lacunes, Bob Kahn décide de construire un nouveau protocole pour remplacer *NCP*. Bob Kahn définit quatre impératifs pour ce protocole :

1. Chaque réseau interconnecté doit être autonome et aucune modification ne doit être nécessaire sur l'architecture de ce réseau pour le connecter à l'*Internet*.
2. L'acheminement des paquets se fait *dans la mesure du possible*. Si un paquet n'arrive pas à destination, l'émetteur du paquet a la charge de le retransmettre.
3. Les réseaux sont interconnectés entre eux par des *boites noires*[11]. Ces boites noires ne font que retransmettre les informations qui leur parviennent.
4. Aucun contrôle global des opérations réseau ne doit exister.

7. C'est à dire sous forme électronique et accessible depuis le réseau.
8. Les premières RFC ont été rédigées par Steve Crocker, membre fondateur du *NWG*, dès 1969. Ces documents, publics, constituent encore aujourd'hui, les références de base des protocoles de l'InterNet.
9. et afin de garder un certain contrôle sur celui-ci, la *DARPA* étant toujours une agence d'origine militaire, et on est toujours dans le contexte de la guerre froide.
10. alors que l'on peut considérer que l'*ARPANET* est né en Octobre 1969, lors de la connexion de l'*UCLA* avec le *SRI*.
11. désignées plus tard sous le nom de *routeurs* ou *passerelles*.

TCP/IP

Au printemps 1973, Bob Kahn demanda à Vinton Cerf, qui avait implémenté *NCP* au *SRI*, de développer avec lui ce nouveau protocole. La première version de ce protocole [Cerf & Kahn, 1974] fut présentée à une rencontre spéciale de l'*INWG*, qui eut lieu à l'Université du Sussex en Septembre 1973. Les caractéristiques suivantes furent exposées :

— Les communications entre deux processus consistent en une longue séquence de caractères (appelés *octets*, en français dans le texte). La position de chaque *octet* dans la séquence permet de l'identifier.
— Le contrôle de trafic est réalisé en utilisant des *fenêtres glissantes* et des *acquittements*. L'ordinateur destination des caractères peut décider le moment de l'acquittement et les acquittements peuvent être cumulés pour les paquets arrivés et non acquittés.
— Les machines reliées au réseau ont une adresse globale sur le réseau. Cette adresse est représentée par un nombre codé sur 32 bits. Les 8 premiers bits représentent le réseau et les 24 bits restants représentent la machine connectée au réseau [12].

L'article de Cerf et Kahn présentant l'*Internet* décrit un protocole, appelé *TCP* (pour *Transmission Control Protocol*), qui correspond à tous les services de transport et de routage du réseau *Internet*. *TCP*, à cette époque, décrit plusieurs services de transport, allant de l'acheminement d'une séquence de caractères totalement sure (*circuit virtuel*) à l'acheminement de *datagrammes*, messages simples envoyés d'une machine à une autre et implémentés directement par le service de commutation de paquets, ce qui implique des pertes, corruptions ou mélanges éventuels dans les datagrammes.

La première fonctionnalité de *TCP* qui à être développée fut l'établissement de circuits virtuels, ce qui permit de développer des applications de transfert de fichiers (*FTP*, *File Transfer Protocol*) et de connexion à distance (*Telnet*). L'établissement de circuits virtuels était inadéquat pour d'autres applications (par exemple, le transfert de la voix par réseau, développé à partir des années 70) pour lesquelles, et pour des raisons de performances, les pertes de paquets ne devaient non pas être corrigées au niveau du protocole de transport, mais par les applications elles-mêmes.

Afin de répondre à ces besoins, le protocole *TCP* sera réorganisé en deux protocoles, à partir de mars 1978 : *IP* (*Internet Protocol*) prenant en charge l'adressage et le routage des paquets et, séparément *TCP*, réalisant uniquement le contrôle de trafic et la correction des paquets perdus pour établir des circuits virtuels.

Pour les applications n'ayant pas besoin de *TCP*, un autre protocole, *UDP* (*User Datagram Protocol*), sera ajouté, permettant un accès direct au service fourni par le protocole *IP*.

Mise en place de l'Internet

La *DARPA* confia, à partir de 1974, au *SRI* (Vinton Cerf), à *BBN* (Ray Tomlinson) et à l'*UCLA* (Peter Kirstein) la réalisation de *TCP*, et son intégration dans leurs trois environnements informatiques. Un an après, les trois implémentations différentes étaient fontionnelles et pouvaient communiquer. Trois réseaux furent reliés par le protocole *TCP* : *ARPANET*, *RPNET* et *SATNET*.

Le développement de nouveaux réseaux

D'autres types de réseaux faisaient leur apparition ; le développement du système *Unix*, créé dans les *Bell Laboratories* d'*AT&T* en fut un vecteur prédominant. L'intégration du protocole de communication *UUCP* (*Unix to Unix CoPy*), permit à de nombreux réseaux d'émerger : *USENET* (1979), *EUnet* (*European Unix NETwork*) (1982), *JUNET* (*Japan Unix NETwork*), ... *UUCP* étant cependant basé sur la *commutation de messages*, ces réseaux étaient essentiellement utilisés pour le courrier électronique. *USENET* permit cependant le développement d'une nouvelle application sur le réseau : les forums de discussion en ligne.

BITNET (*Because It's Time NETwork*), basé sur des protocoles fournis par *IBM* et *CSNET* (*Computer Science Network*) furent créés pour les besoins des universités n'ayant pas une connexion directe à *ARPANET*. *CSNET* était directement financé par la *NSF* (*National Science Fundation*).

12. Il s'est avéré assez rapidement, et notamment lors de l'apparition des réseaux locaux dans la fin des années 70, que l'hypothèse selon laquelle il n'y aurait jamais plus de 256 réseaux connectés par l'*Internet* ait à être revue.

Changement d'échelle d'Internet

A cette époque, *TCP* n'avait été mis en place que sur des gros ordinateurs à temps partagé. *TCP* était alors trop gros et trop complexe pour fonctionner sur un autre type de machine. David Clarke et son groupe de recherche au *MIT*, écrivirent une version compacte et simplifiée de *TCP* pouvant fonctionner sur le *Xerox Alto*, la première station de travail, développée au *Xerox PARC* (*Palo Alto Research Center*). Le protocole de réseau local *Ethernet* [13], développé également au *Xerox PARC* (et implémenté de façon expérimentale sur des *Xerox Alto*), est inventé par Bob Metcalfe en 1973.

A partir de 1979, la *DARPA* finance le *Computer Systems Research Group* (*CSRG*) de l'Université Californie à Berkeley pour intégrer *TCP/IP* directement dans leur version d'*Unix*, *BSD* (*Berkeley System Distribution*). En 1980, le *CSRG* développera un concept nouveau, permettant de faire représenter par le système l'accès au réseau comme un fichier pour les applications : les *sockets*. *Bill Joy*, ayant travaillé au *CSRG* sur le développement de *BSD* est co-fondateur en 1982 d'une société ayant pour vocation la commercialisation de stations de travail fonctionnant sous *BSD* et intégrant directement le protocole *TCP/IP* : *SUN* (*Stanford University Network*). *SUN* sera à l'origine d'un certain nombre d'applications et de protocoles autour de *TCP/IP* : notamment *RPC* (*Remote Procedure Call*, permettant d'exécuter des programmes à distance) et *NFS* (*Network File System*, permettant le partage de fichiers entre des machines connectées à un réseau).

L'existence de réseau locaux, la possibilité de faire fonctionner le protocole *TCP/IP* sur des stations de travail, l'intégration de *TCP/IP* dans *Unix* et l'émergence de nouveaux réseaux qui seront progressivement interconnectés à *Internet* entraineront un changement d'échelle du réseau, passant de quelques gros ordinateurs interconnectés à une interconnexion de réseaux locaux de stations de travail.

La structure d'*Internet* devra être modifiée pour prendre en compte ce changement d'échelle. D'abord, l'adressage *IP*, sur 8 + 24 bits ne permettant que l'interconnexion de 256 réseaux a été revu. Désormais, 3 classes d'adresses sont définies : une classe A représentant les réseaux d'envergure nationale (peu de réseaux, mais comprenant beaucoup de machines, comme l'*ARPANET*, par exemple), une classe B représentant des réseaux d'envergure régionale et une classe C pour les réseaux locaux (beaucoup de réseaux comprenant peu de machines). Ces classes d'adresses seront abandonnées à partir de 1998, pour être remplacées par des préfixes de taille variable.

Cette évolution est conjointe au développement d'une nouvelle version du protocole IP, IPv6 [14], dont l'objectif est de définir un protocole de réseau adapté à un réseau de la taille de l'Internet. Le protocole IP antérieur à IPv6 est renommé IPv4. IPv4 est encore à l'heure actuelle le protocole réseau majoritairement utilisé sur l'Internet, mais il existe de nombreux équipements capables d'exploiter IPv6 et de grands réseaux IPv6 internationaux ont vu le jour depuis l'introduction de ce protocole.

A l'origine d'*Internet*, le faible nombre de machines connectées permettait de maintenir des tables de toutes les machines avec leurs noms et leurs adresses *IP* sur une seule machine (celle du *SRI-NIC*). Pour permettre de connecter à *Internet* un plus grand nombre de machines, Paul Mockapetris de l'*Information Sciences Institute* de l'Université de Californie du sud (*USC/ISI*) créera en 1984 le *Domain Name System* (*DNS*), reposant sur une architecture distribuée de *serveurs de noms* et permettant de résoudre les correspondances entre les noms et les adresses *IP* des machines connectées à *Internet*.

Le 1^{er} Janvier 1983, l'*ARPANET* change de protocole : *NCP* est abandonné au profit de *TCP/IP*. L'*ARPANET* sera abandonné en 1990.

Les applications

En février 1976, la reine Elizabeth II d'Angleterre envoie un message électronique depuis le *Royal Signals and Radar Establishment (RSRE)* à Malvern. Dans les années 70, le réseau est principalement utilisé pour le courrier électronique. Parmi les premières applications, on trouve également *telnet*, permettant des connexions à distance pour lancer des programmes (1972), et le transfert de fichier.

13. A l'origine, ce protocole était destiné à des réseaux utilisant des ondes radio, sans support physique, d'où le nom *Ether*–net.

14. IPv6 a d'abord été référencé sous le terme *IPng*.

USENET a rendu les forums de discussion en ligne populaires. En 1986, le protocole *NNTP* (*Network News Transfer Protocol*) permettra de faire circuler les *news* (messages à des forums de discussions) de manière efficace par le protocole *TCP/IP*.

En 1990, *Archie*, le premier moteur de recherche sur internet est créé. En 1991, *Gopher* est créé par Paul Lindner et Mark P. Mac Cahill de l'Université du Minnessota. *Gopher* permet d'organiser et de présenter l'information d'une manière arborescente en utilisant des liens *hypertexte*. En 1991 également, Brewster Kahle de *Thinking Machine Corporation* réalise *WAIS* (*Wide Area Information Servers*), permettant de relier des serveurs d'information textuelle agrémentée d'images, au moyen de liens hypertexte à travers le réseau.

Le réseau Internet a bien supporté le tant redouté "passage à l'an 2000" (le 1^{er} Janvier 2000), certains comportements étranges ont été relevés, par exemple, le serveur de date de l'Observatoire National de la Marine Américaine (*USNO* (*United States Naval Observatory*) [15]), qui maintient la date et l'heure "officielle" pour les Etats Unis, a annoncé la date "1 Janvier 19100" le 1^{er} Janvier 2000 (source : [Zakon, 2000]).

Le World Wide Web (WWW) : En 1990, Robert Cailliau et Tim Berners-Lee, du *CERN* (*Centre Européen pour la Recherche Nucléaire*), en Suisse, créent le *World Wide Web* (*toile mondiale*). Le *World Wide Web*, reprenant certains concepts de *Gopher* et *WAIS*, permet de mettre à disposition sur des serveurs connectés à *Internet*, des informations contenues dans des pages de texte agrémentées d'images et de documents multimédia (sons, animations, video, ...) et contenant des liens hypertexte vers d'autres pages. Un nouveau protocole, destiné au transfert de l'information hypertexte est créé par le *CERN* : le protocole *HTTP* (*HyperText Transfer Protocol*). Le *CERN* crée également un nouveau langage de description de page, le *HTML* (*HyperText Markup Language*), langage à balises inspiré du langage *SGML* (*Standard Generalized Markup Language*).

Ils écrivent un logiciel de serveur pour le *World Wide Web* : le *CERN httpd*. Tim Berners-Lee écrit en 1990 un client expérimental, fonctionnant sur *NeXTStep*, pour le *WWW*, il s'appelle *WorldWideWeb* à l'origine, et sera renommé *Nexus* par la suite pour éviter la confusion entre le concept de *World Wide Web* et le client lui-même.

Avant que les clients graphiques ne soient répandus (avant 1992), le *WWW* n'est, pour la plupart des utilisateurs, qu'une interface en mode texte accessible par *telnet* sur le site `info.cern.ch`. Les liens hypertextes sont alors représentés par des numéros qui doivent être tapés par les utilisateurs pour changer de page.

Un certain nombre de clients *WWW* plus ou moins expérimentaux seront développés à travers le monde au tout début des années 90, la plupart fonctionnant sous *Unix*. Le plus important d'entre eux, *ViolaWWW* écrit par Pei Wei, un étudiant de Stanford était devenu le plus répandu et recommandé officiellement par le *CERN*.

Le *CERN* développa, conjointement avec le *National Center for Computer Applications* (*NCSA*), un client *WWW* : *Mosaic* qui sera rendu public à partir de 1992. *Mosaic* est une réelle nouveauté pour le grand public, car il permet d'afficher des pages exprimées en langage *HTML* incluant les documents multimédia qu'elles contiennent, et permet d'utiliser des protocoles d'application d'*Internet* pour transmettre ou récupérer de l'information (*HTTP*, *FTP* et *NNTP* notamment). Les liens hypertexte peuvent être suivis simplement en pressant un bouton de la souris en ayant le curseur de la souris sur une partie du lien. En 1994, une société privée, *Netscape*, crée elle aussi un client *WWW* [16], fortement inspiré de *Mosaic*. Ce client servira de base au client *Firefox*, logiciel libre [17] développé à partir de la fin des années 90. À partir de 1993, Microsoft développe son propre client WWW, *Internet Explorer*, lui aussi basé sur *Mosaic*. Internet Explorer fut [18] livré conjointement aux systèmes d'exploitation *Windows* de *Microsoft*.

En 1994, l'aboutissement de ces technologies permet de commander une pizza sur *Internet*.

Le langage HTML (*HyperText Markup Language*), mis en œuvre notamment par Tim Berners-Lee conjointement aux premiers développements du *World Wide Web* dès le début des années

15. `http://www.usno.navy.mil/`
16. Il s'appelle *Netscape* à l'origine, puis *Netscape Navigator* et se prononce *Mozilla*.
17. Mosaic et Netscape n'étaient pas des logiciels libres.
18. un accord très récent entre la Microsoft et communauté européenne concernant la vente liée d'*Internet Explorer* et des systèmes *Windows* permet aux utilisateurs de *Windows 7* des pays membres de la communauté européenne de choisir le client WWW installé par défaut sur le système lors de son installation.

90 a connu de nombreuses évolutions. Parmi celles-ci, citons le développement de langages spécifiques à la présentation (*Cascading StyleSheets* ou *CSS*) à partir de 1994, le langage *JavaScript*, initialement développé par *Netscape* à partir de 1995, puis adopté comme standard. À partir de 1998, les nouvelles possibilités de HTML 4, combinées à CSS et JavaScript permettent de rendre les pages HTML plus dynamiques [19] mais c'est surtout le développement puis la standardisation de XML (standardisé depuis février 1998), ainsi que le développement de CSS 2, couplé à JavaScript qui formeront les bases de ce qui est aujourd'hui appellé *Web 2.0*.

Le *World Wide Web* sert également de support à toute une panoplie d'applications interactives. Un certain nombre d'interfaces de programmation (à l'instar de *CGI*, *Common Gateway Interface*, développé à partir de 1992) et de langages spécialisés (par exemple *PHP Personal Home Pages*, développé à partir de 1994) ont été spécifiquement créés dans ce but. Plus récemment, des protocoles ont été standardisés pour le développement de *services web*. Ils s'appuient généralement sur le langage XML. Le plus connu d'entre eux est SOAP (*Simple Object Access Protocol* [20]), permettant la descriptions de services et leur interface d'accès, ainsi que les méthodes de sérialisation de données permettant ces accès.

Administration d'Internet

A la fin des années 70, la croissance d'*Internet* a poussé Vinton Cerf, jusqu'alors chargé d'organiser le réseau, à créer plusieurs entités de coordination : l'*International Cooperation Board* (*ICB*), chargé de coordiner les activités avec les pays européens, sur des thèmes de recherches sur les communications par satellites notamment et l'*Internet Configuration Control Board* (*ICCB*), chargé de la gestion des activités sur le réseau *Internet*.

En 1983, lorsque Barry Leiner devient directeur du programme de recherche sur *Internet* à la *DARPA*, il décide que l'*ICCB* n'était plus en mesure d'assumer tout seul la gestion du réseau. Il restructure l'*ICCB* en un ensemble de *Task Forces*, chacune spécialisée dans un domaine technologique particulier. L'*Internet Activities Board* (*IAB*) fut créé pour coordonner les différentes *Task Forces*.

En 1985, une des *Task Forces*, l'*Internet Engineering Task Force* (*IETF*), dont le rôle était de résoudre tous les problèmes techniques et opérationnels du réseau, devenait préominante par rapport aux autres *Task Forces*. Une nouvelle restructuration eut lieu afin de répondre à cette acroissement de l'*IETF* : l'*IETF* fut décomposée en groupes de travail, coordonnés par l'*Internet Engineering Steering Group* (*IESG*). Les autres *Task Forces* (autres que l'*IETF*) furent regroupées et coordonnées par l'*Internet Research Task Force* (*IRTF*).

En 1992, une nouvelle réorganisation de l'*IAB* eut lieu : l'*Internet Activities Board* fut renommé en *Internet Architecture Board* (avec les mêmes initiales). L'*Internet Society* (*ISOC*) fut créée pour coordonner les activités de l'*IAB*, l'*IETF* et l'*IESG*. Le *World Wide Web Consortium* (*W3C*) fut créé pour coordonner les activités autour du *World Wide Web*.

En 1993, l'*InterNIC* (*Inter-Network Information Center*) fut créé pour maintenir des informations en ligne sur le réseau et gérer les enregistrements de noms de domaines sur le réseau.

Évolution du réseau

Depuis 1969 où l'ARPANET interconnectait 4 noeuds, la taille du réseau ARPANET puis Internet a augmenté de manière quasiment exponentielle : durant les années 90, la taille du réseau a pratiquement doublé chaque année ; la taille du réseau a continué a progresser durant les années 2000, mais à un rythme moindre.

En 2009, le réseau *Internet* regroupe plus de 625 millions de machines. Plus de 130 millions de serveurs *WWW* ont été recensés. Plus d'un milliard et demi de personnes dans le monde ont déjà utilisé Internet. Chaque jour, entre 150 et 400 millions de personnes se connectent à *Internet* à travers le monde et échangent plus de 100 peta-octets de données. Les applications de l'Internet concernent principalement (aujourd'hui et en volume) du partage de fichiers, de l'accès à des pages WWW, de la diffusion de contenu audiovisuel, du courrier électronique et des discussions synchrones (voix, images et texte).

19. On parle alors de *DHTML* pour *Dynamic HyperText Markup Language*.
20. Ces initiales sont aujourd'hui obsolètes ; le nom est conservé, mais les initiales perdent leur sens.

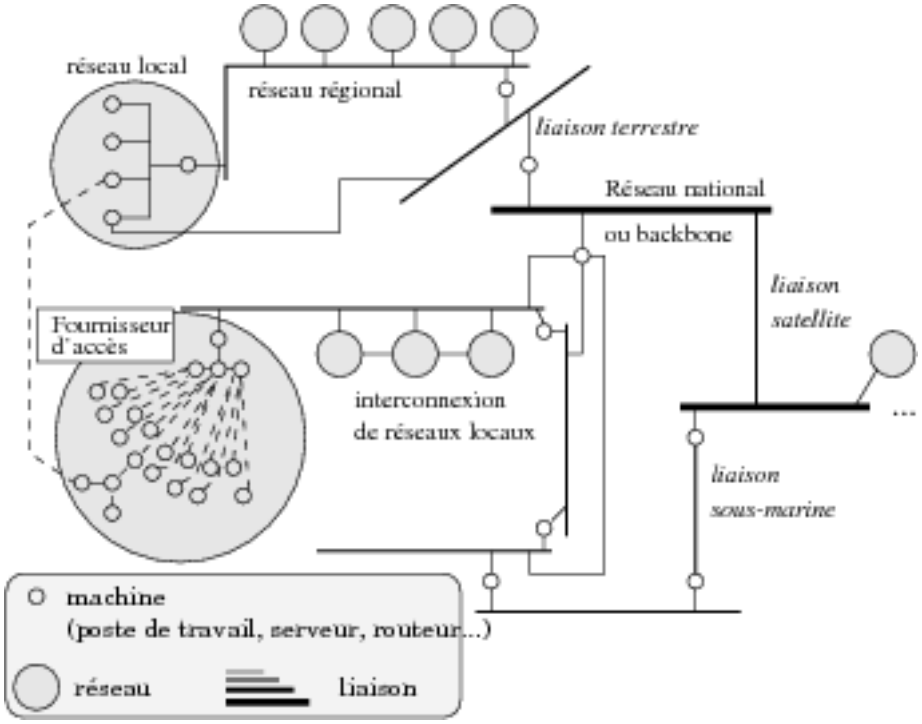


FIGURE 1.1 – Architecture physique d'internet

1.0.2 Architecture de l'internet

Internet est une interconnexion de réseaux, sans contrôle global des opérations. Il a conservé, de ses objectifs d'origine, sa capacité à interconnecter des réseaux de structures différentes. La figure 1.1 présente schématiquement les différentes liaisons qui composent Internet.

Les liaisons qui composent Internet sont très diverses et ont des caractéristiques physiques et des protocoles de bas niveau très différents : on peut les regrouper en deux grandes catégories :

— Les liaisons *point-à-point* : par exemple les liaisons téléphoniques en utilisant des *modems* (*MODulateur DÉModulateur*) pour convertir l'information numérique circulant sur le réseau en signal analogique pouvant passer sur une ligne téléphonique et réciproquement. L'utilisation du réseau téléphonique commuté analogique permet d'atteindre des débits jusqu'à 56 Kbits/s.
Le service *Numeris* de proposé par Orange, utilisant les protocoles *RNIS* (*Réseau Numérique à Intégration de Services*) permet de faire circuler les information sous forme numérique directement. Il permet en outre de grouper plusieurs lignes (accès de base, chacun représentant une bande passante de 64 Kbits/s.) pour former une liaison de capacité plus élevée.
L'*ADSL* (*Asymetric bit rate Digital Subscriber Line* ou ligne numérique d'abonnés à débits asymétriques) permet d'atteindre, toujours sur le réseau téléphonique grand public, des débits jusqu'à 24 Mbits/s. du réseau vers l'abonné et jusqu'à 2 Mbits/s. de l'abonné vers le réseau.
Toutes ces liaisons, basées sur le réseau téléphonique commuté, sont *temporaires*. Les *liaisons louées* (ou *liaisons spécialisées*) permettent de créer des liaisons point à point, entre deux sites, à l'échelle nationale (ou continentale). Ces liaisons peuvent être réalisées par du cable coaxial ou de la fibre optique.
Des liaisons par cable sous-marin et par satellite permettent de relier entre-eux les réseaux situés sur des continents différents.

— Les réseaux locaux : ce sont des réseaux de petite dimension physique (un bâtiment ou un établissement). A l'opposé des liaisons point à point, les réseaux locaux permettent de relier plusieurs machines simultanément. Le protocole (physique) de réseau local le plus

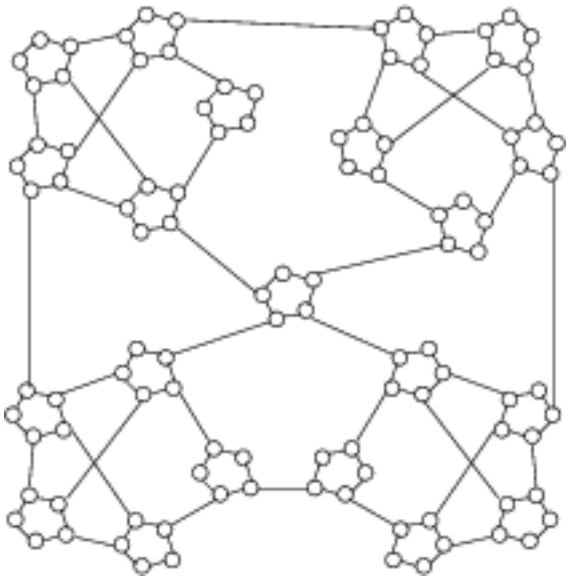


FIGURE 1.2 – Structure décentralisée

connu est *ethernet*. Il existe en quatre technologies correspondant à quatre échelles de bande passante : 10 Mbits/s., 100 Mbits/s., 1 Gbits/s. et 10 Gbits/s. Pour l'instant, les débits de 100 Mbits/s. et 1 Gbits/s. sont les plus répandus.

— Les *backbones* : ce sont des réseaux interconnectant des réseaux plus petits et autonomes. Ils ont une envergure régionale (par exemple le réseau *Gigalis*, nationale (par exemple le réseau *RENATER* (*Réseau National pour l'Enseignement et la Recherche*) ou internationale (par exemple *Geant* ou *EUnet*, pour l'Europe ou *BBNnet* pour les États-Unis).

— Des réseaux IP peuvent également être supportés par d'autres types de réseaux. C'est le cas des réseaux de téléphonie mobile, supportant des accès de type *dialup* (chaque client est connecté à un opérateur agissant comme un fournisseur de service : il participe donc à une liaison point-à-point avec son opérateur). Les débits vont de quelques kbits/s. (GSM) à quelques mbits/s. (3G, 3G+, 4G).

L'architecture d'Internet est décentralisée, les réseaux qui composent Internet fonctionnent de manière autonome. La figure 1.2 représente la structure du réseau et peut être interprété de manière identique à différentes échelles (un rond représentant une machine, un réseau local ou un réseau national).

Les routeurs fonctionnent de manière autonome, sans nécessiter d'information centrale, et les principales applications d'Internet fonctionnent d'une manière distribuée [21] ; les informations sur les noms de domaine sont réparties sur des serveurs de noms de domaine s'interrogeant mutuellement et installés localement au même titre que les machines du domaine.

1.0.3 Notes bibliographiques

La source officielle et la plus complète de documentation sur le réseau *Internet* est l'*ISOC* (*Internet Society*). Un certain nombre de documents sont proposés en ligne sur le site *WWW* de l'*ISOC*, `http://www.isoc.org`. Une partie de ce site est consacrée à l'épopée du réseau, accessible à partir de l'*URL* `http://www.isoc.org/internet/history/`. On trouve notamment une chronologie très complète des évènements ayant marqué l'évolution du réseau [Zakon, 2000], et un historique rédigé par de nombreuses personnes ayant été des acteurs prépondérants dans l'évolution du réseau [Leiner et al., 1998].

Les spécifications officielles des protocoles du réseau *Internet*, les *Request for Comment* (*RFC*), sont accessibles en ligne depuis le site de l'*IETF*, depuis l'*URL* `http://www.ietf.org/rfc`.

21. n'importe quelle machine connectée au réseau d'une manière permanente peut être serveur *WWW* par exemple.

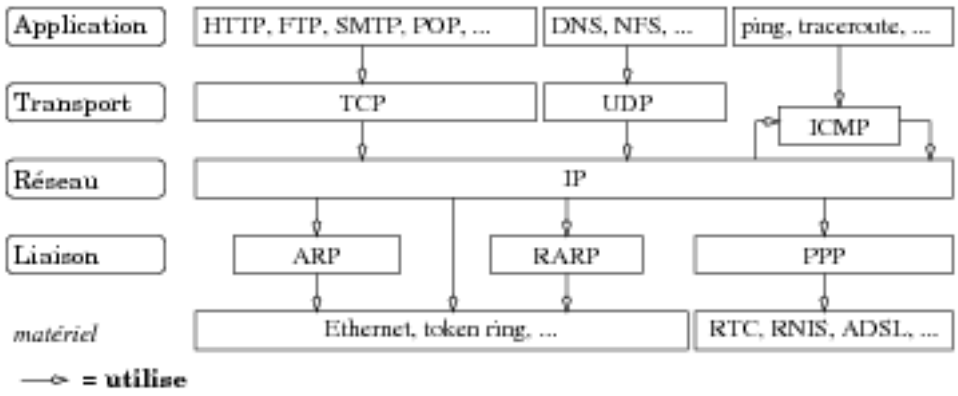


FIGURE 1.3 – Pile de protocoles TCP/IP

Aujourd'hui les RFC représentent un ensemble de plus de 2600 documents pour un total de plus de 40000 pages.

De nombreuses informations sur les standards et les évolutions du *WWW* sont disponibles sur le site du *W3C*, accessible depuis l'*URL* `http://www.w3c.org`. L'histoire du *World Wide Web* y est notamment commentée par Tim Berners-Lee à l'*URL* `http://www.w3.org/People/Berners-Lee/FAQ.html`.

Un certain nombre d'ouvrages plus grand public ont été publiés sur le sujet. Parmi eux, l'ouvrage de *Christian Huitema*, *Et Dieu Créa l'Internet...* [Huitema, 1995], apporte un éclairage précis sur l'entrée de l'*Internet* en France ; en effet, Christian Huitema, travaillant à l'*INRIA* (*Institut National de Recherche en Informatique et en Automatique*) et ayant été élu président de l'*IAB* en Avril 1993 a été l'un des pionniers de l'*Internet* en France.

1.1 Vue globale des protocoles TCP/IP

TCP/IP est organisé en quatre couches de protocoles, illustrées par la figure 1.3. Il existe une correspondance partielle et non normalisée entre ces protocoles et la norme définie par *OSI* (*Open Systems Interconnexion*) et maintenue par l'*ITU-T* (*International Telecommunication Union, Telecommunication Sector* [22]), communément appelée *pile ISO*, présentée à la figure 1.4. Dans TCP/IP, les couches de *session* et de *présentation* de la pile ISO sont prises en charge par les applications et la couche physique est prise en charge par les pilotes des interfaces matérielles (ethernet, par exemple).

Le principe est que les dispositifs mis en place au niveau de chaque couche n'ont à connaître les interfaces que de la couche directement inférieure et ne sont directement utilisés que par la couche directement supérieure. Les dispositifs de chaque couche d'un système donné sont amenés à communiquer avec les dispositifs de la même couche d'un système distant.

L'exemple de la figure 1.5 illustre ce qui se passe pour transmettre des données d'une application (ici pour le WWW), à travers le réseau Internet.

1.1.1 Protocoles de liaison

ARP

ARP (*Address Resolution Protocol*, ou *protocole de résolution d'adresses*) est utilisé par IP pour déterminer les adresses IP sur un réseau local. Les adresses sont établies par une correspondance entre les adresses IP et les adresses matérielles des ordinateurs connectés au réseau local. ARP permet d'obtenir une adresse matérielle à partir d'une adresse IP.

22. `http://www.itu.int`

7	Applications	Fournir aux applications tous les moyens d'accéder à l'environnement réseau : transfert d'informations, allocation de ressources, vérification d'intégrité et de cohérence des données, ...
6	Présentation	Représenter les données de façon à ce qu'elles puissent être utilisées par les applications (tous les ordinateurs ne représentent pas les informations de la même façon).
5	Session	Organiser et synchroniser les dialogues et les échanges de données.
4	Transport	La couche transport assure un transfert de données transparent entre entités se trouvant sur le réseau. Elle a également pour but d'optimiser l'utilisation du réseau afin d'assurer au moindre coût les performances requises pour les applications.
3	Réseau	Assurer le relai des informations entre entités qui ne sont pas physiquement connectées directement entre elles (routage). La couche réseau assure également l'adressage des entités sur le réseau. Elle a également un rôle de contrôle de flux, pour éviter les congestion du réseau notamment.
2	Liaison	Établir, maintenir et libérer des connexions entre deux entités physiquement reliées. La couche liaison détecte et corrige si possible les erreurs dues au transport physique des données. Elle définit également la manière d'enchainer les échanges.
1	Physique	Transmettre des signaux numériques entre deux ou plusieurs entités connectées par un support physique (généralement électrique). La couche physique décrit à la fois le support de transmission en termes physiques (électroniques, ...) mais également en termes fonctionnels.

FIGURE 1.4 – Pile *ISO*

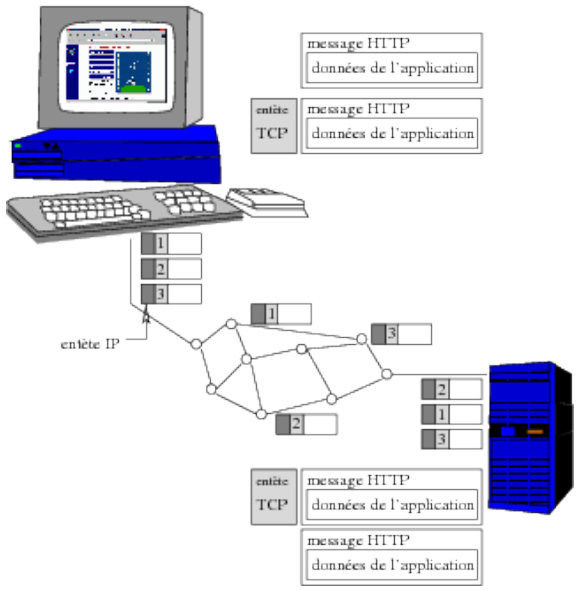


FIGURE 1.5 – Exemple de transmission de données d'application sur Internet

PPP

PPP (*Point-to-Point Protocol*) permet de transmettre des paquets IP[23] sur des liaisons point à point (série, modem, RNIS, ...).

Il assure la gestion du contrôle d'accès au réseau par authentification avec un échange de mots de passe, éventuellement cryptés. Il détecte et corrige les erreurs de transmission et permet de configurer automatiquement un ordinateur établissant une connexion au réseau.

1.1.2 Protocoles de réseau

IP

Le protocole IP (*Internet Protocol*) assure, sans connexion, un service non fiable de délivrance de paquets IP. Le service est non fiable car il n'existe aucune garantie que les paquets IP envoyés arrivent à destination. Les paquets IP peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre.

Le mode de transmission est non connecté car IP traite chaque paquet indépendamment de ceux qui le précèdent ou qui le suivent. Ainsi, deux paquets IP issus d'une même machine et à destination d'une même machine peuvent ne pas suivre le même chemin.

Le routage est une des fonctionnalités les plus importantes d'IP. Il s'agit de choisir un chemin (le meilleur, si possible) pour transmettre un paquet à travers les réseaux interconnectés par Internet.

La version d'IP la plus courante sur Internet est la version 4 (*IPv4*), mais une version 6 (*IPv6* ou *IPng* (*IP new generation*) est en cours de développement [RFC 2460]. Les principales amélioration d'IPv6 par rapport à IPv4 sont les suivantes :

— les adresses réseau passent de 32 bits (IPv4) à 128 bits (IPv6), permettant ainsi d'adresser plus de machines simultanément sur le réseau.
— Certains protocoles sont simplifiés et optimisés.
— Possibilité d'étiqueter des paquets avec une *qualité de service*.
— Ajout de fonctionnalités de sécurité (confidentialité, notamment) réseau.

ICMP

Le protocole ICMP (*Internet Control Message Protocol*) organise un échange d'informations (réalisé sous la forme de paquets IP) permettant la gestion des erreurs sur le réseau (par exemple si un routage circulaire est détecté ou si une adresse IP de destination n'existe pas).

ICMP utilise IP pour son fonctionnement, mais il ne transporte jamais de données d'applications. C'est pourquoi il est placé un peu à cheval entre les couches transport et réseau sur la figure 1.3.

1.1.3 Protocoles de transport

Les protocoles TCP (*Transmission Control Protocol*) et UDP (*User Datagram Protocol*) sont les deux protocoles de transport d'Internet ; ils utilisent tous les deux IP comme couche réseau, mais TCP procure une couche de transport fiable, alors qu'UDP ne fait que transporter de manière non fiable des datagrammes.

UDP

UDP permet d'échanger des datagrammes (messages courts d'applications) entre deux ordinateurs du réseau. UDP ne fournit que les trois seuls services suivants par rapport à IP : la détection d'erreurs de transmission dans le paquet, la vérification de l'adresse de destination et la définition de *ports*. Les *ports* sont des numéros qui permettent de définir des *services* sur une machine. Par exemple, le service de résolution de nom, correspondant au DNS, utilise, par convention le numéro 53.

23. PPP n'est pas limité au seul protocole de réseau IP, il permet également d'utiliser les protocoles *Appletalk*, *IPX*, *X.25*, *Decnet*, ...

Les applications utilisant UDP doivent donc gérer elles-mêmes les pertes de messages, les duplications, déséquencement, ... Par contre UDP permet d'envoyer en une seule fois un message à plusieurs destinataire (*broadcasting*) (ce que ne permet pas TCP).

TCP

TCP procure un service de flux séquentiel de caractères orienté connexion et fiable. Le terme *orienté connexion* signifie que l'une des applications dialoguant à travers TCP est considérée comme un *serveur* et l'autre comme un *client*. Le client doit établir une connexion avec le serveur avant de dialoguer (comme dans le cas du téléphone). La connexion peut se terminer à l'initiative de l'une ou l'autre des deux extrémités, indifféremment. Une connexion TCP ne peut exister qu'entre seulement deux machines à la fois (pas de *broadcast*).

Les connexions TCP sont *bidirectionnelles simultanées* (*full duplex*), ce qui signifient que les informations peuvent circuler dans les deux sens, simultanément, de manière asynchrone (le flux de données dans un sens n'est pas asservi au flux de données dans l'autre).

Pendant une connexion, deux flux séquentiels de caractères circulent entre les deux extrémités, sans qu'il soit possible de séparer par une marque quelconque certaines données.

Enfin, TCP supporte également la notion de *port* de la même façon qu'UDP.

1.1.4 Protocoles d'application

Les protocoles de liaison (ARP, PPP, ...), de réseau (IP, ICMP) et de transport (TCP, UDP) font partie de l'implémentation de TCP/IP et sont généralement intégrés au noyau du système d'exploitation [24].

Les protocoles d'application sont inclus dans les applications (processus utilisateur du système d'exploitation) elles-mêmes. Ainsi, un agent de courrier électronique implémentera les protocoles d'application SMTP et IMAP, par exemple.

Courrier électronique

Le courrier électronique est géré au sein d'Internet par des agents (*Mail Transfer Agent* (*MTA*), relais de messagerie ; le logiciel *MTA* le plus ancien (mais toujours utilisé) est *sendmail*) utilisant le protocole *SMTP* (*Simple Mail Transfer Protocol*).

Le protocole *SMTP* permet d'*expédier* du courrier, c'est à dire qu'une machine, cliente, a un courrier électronique et se connecte à une autre, serveur, et lui envoie le courrier électronique.

D'autres protocoles, *POP* (*Post Office Protocol*) et *IMAP* (*Interactive Mail Access Protocol*) permettent de *relever* une boite aux lettres électronique. Dans ce cas, la machine cliente se connecte à une autre, serveur et va demander le transfert du courrier électronique.

Il existe un certain nombre d'applications utilisant ces protocoles qui permettent aux utilisateurs d'éditer et d'expédier du courrier électronique, ainsi que de consulter leurs boite aux lettres (de telles applications sont appelées *Mail User Agent* (*MUA*)), par exemple *Thunderbird* ou *Outlook*. La plupart de ces applications permettent d'organiser les boites aux lettres selon divers critères, incluant éventuellement des filtres antispam.

Le développement d'interfaces web très dynamiques a également popularisé l'utilisation de *webmails* qui sont des *MUA* réalisés à partir de serveurs WWW.

Transfert de fichiers

Le protocole *FTP* (*File Transfer Protocol*) permet de transférer des fichiers d'une machine à une autre. Il permet d'authentifier un utilisateur avec un nom de connexion et un mot de passe et, après validation de l'utilisateur et de ses droits d'accès, de transférer ou de lire un fichier. Il est également possible d'accéder à des ressources par le protocole *FTP* en s'identifiant de manière anonyme (nom de connexion *anonymous* et adresse de courrier électronique comme mot de passe).

24. Dans le cas de micro-noyaux, comme pour HURD, MacOS X et Windows [25], par exemple, les couches basses de TCP/IP sont implémentés dans des *serveurs*, donc comme des processus utilisateur.

FTP fonctionne en mode client-serveur. D'une manière générale, les serveurs sont des programmes mettant à disposition des ressources et les clients des programmes permettant d'aller chercher ces ressources. Un exemple de serveur *FTP* très répandu est *wu-ftpd* (*Washington University File Transfer Protocol Daemon*). La plupart des clients *WWW* intègrent un client *FTP*.

Partage de fichiers

Le partage de fichiers a également pour vocation de permettre le transfert de fichiers entre machines du réseau Internet, mais, sans nécessiter un fonctionnement en mode client-serveur. Ainsi, plusieurs machines peuvent mettre à disposition un même fichier et plusieurs machines peuvent recevoir le même fichier simultanément.

Ce mode de fonctionnement sert de base à un certain nombre d'applications regroupées sous le terme générique de *partage de fichiers pair-à-pair* (*peer-to-peer*). Les applications de partage de fichier sont nombreuses et les plus répandues sont *gnutella*, *e-Donkey*, *e-Mule* et *BitTorent*.

Système de fichiers réseau

D'une manière conceptuellement assez similaire, un certain nombre de protocoles de partage de fichiers en réseau local ont été développés. Le plus ancien est NFS (*Network File System*), utilisé principalement en environnement Unix. SMB (*Server Message Block*) et CIFS (*Common Internet File System*) sont des protocoles qui ont été développés dans le même but par Microsoft pour ses systèmes Windows.

Ces protocoles permettent l'accès à des fichiers partagés, mais contrairement aux protocoles pair-à-pair ils ne permettent pas une répartition des donnés d'un même fichier sur plusieurs nœuds du réseau.

Ces protocoles sont principalement basés sur un transport en mode non connecté.

Forums de discussion

Le protocole *NNTP* (*Network News Transfer Protocol*) permet l'échange de *news* (messages à un forum de discussion) à travers un réseau de serveurs de news[26]. Le protocole *NNTP* assure l'échange des news entre les serveurs et également entre les serveurs et les postes clients, aussi bien pour l'écriture que pour la lecture des messages.

Les news sont organisées en groupes (forums), eux mêmes structurés d'une manière arborescente (par exemple, `fr.comp.os.linux` représente le forum de discussion en français (`fr`) portant sur le système d'exploitation (`os` pour *Operating System*) informatique (`comp` = *Computing*) linux).

Ainsi, lorsqu'un utilisateur poste un article dans un groupe de news, l'article est dans un premier temps déposé sur le serveur de news auquel le poste client est relié. Puis, ce serveur va réexpédier cet article aux différents serveurs auxquels il est relié, et ainsi de suite.

Lorsqu'un utilisateur consulte des forums de discussion, son programme client lui propose la liste des forums de discussion auxquels il peut s'abonner, puis lui liste les messages des forums auxquels il est abonné. En sélectionnant un message, l'utilisateur déclenche (seulement alors) le rapatriement du message proprement dit.

La plupart des clients *WWW* intègrent un client *NNTP*.

World Wide Web

La plupart des clients *WWW* fournissent également des clients *FTP*, *NNTP*, et des *MUA*. De même, pour transférer des documents hypertexte (en langage *HTML*, *HyperText Markup Language*), le *WWW* utilise parfois le protocole *FTP*.

Le protocole *HTTP* (*HyperText Transfer Protocol*) a été créé spécifiquement pour le *WWW*. Le protocole *HTTP* permet de transférer des documents hypertexte contenant des données textuelles, images, fixes ou animées, vidéo ou sons. D'autres types de contenus peuvent être ajoutés, au moyen d'extensions (*plug-ins*) aux clients *WWW*, par exemple le langage *VRML* (*Virtual Reality*

26. Parfois appelé *Usenet*, car, bien qu'intégré à Internet, il a été construit à partir du réseau *Usenet*.

Modeling Language) permet de décrire des scènes dans un espace tridimensionnel en réalité virtuelle. Il existe également des extensions au protocole *HTTP*, comme par exemple *HTTPS*, utilisant le protocole HTTP conjointement aux protocole de transport SSL (*Secure Socket Layer*) et TLS [27] (*Transport Layer Security*) permettant des transactions chiffrées et authentifiées sur le *WWW*.

HTTP fonctionne en mode client-serveur. De même que pour *FTP*, des serveurs connectés de manière permanente sur Internet mettent à disposition des documents à des clients. Le logiciel serveur *HTTP* le plus répandu sur Internet est *Apache httpd* [28]. Il existe un certain nombre de logiciels clients. Le *W3C* fournit un client, *Amaya*, qui fait office à la fois de client WWW et d'éditeur HTML. *Lynx*, écrit à l'Université du Kansas et au *Worcester Foundation for Biomedical Research* dans le Massachusetts, est un client *WWW* complet, en mode texte et utilisable sur des terminaux textuels. *Firefox*, *Safari*, *Internet Explorer*, *Opera* et *Chrome* sont les clients WWW graphiques les plus répandus.

En plus de ce mode de fonctionnement client-serveur, le *World Wide Web* a donné lieu au développement d'applications spécifiques, en particulier des moteurs de recherche.

Le protocole principal du WWW, le protocole HTTP est également souvent utilisé par des applications autres que l'accès à des documents hypertexte. Il sert par exemple à l'installation et à la mise à jour de composants logiciels (*Windows Update* et la plupart des gestionnaires de paquets de distributions Linux utilisent ce protocole pour leur mise à jour). Il sert également de protocole de transport pour les applications des plus variées, y compris synchrones [29].

Applications synchrones

L'Internet permet également un certain nombre d'applications synchrones, allant de la messagerie instantanée et discussion en ligne (*chat*) à la téléphonie et la visioconférence.

Des applications comme *MSN Messenger* ont défini des protocoles spécifiques pour la messagerie instantanée. Des protocoles standardisés ont été également publiés dans le même but, notamment IRC (*Internet Relay Chat*) et XMPP (utilisé par le réseau *jabber*).

Les protocoles h323 [30] et SIP (*Session Initiation Protocol*) sont les plus répandus pour la téléphonie sur IP et la visioconférence.

Le protocole RTSP (*Real-Time Streaming Protocol*) permet l'accès à des contenus vidéo, en demandant leur diffusion.

Les protocoles utilisés dans ce cadre fonctionnent principalement en mode non connecté, par envoi successif de message élémentaires.

Le DNS

Bien qu'étant au cœur du fonctionnement d'Internet, le *DNS* (*Domain Name System*) est une application. Il est composé de serveurs DNS, effectuant des requêtes entre eux pour résoudre les correspondances *nom de domaine* vers adresse IP et les correspondances adresse IP vers nom de domaine. Cette deuxième correspondance est appelée *résolution inverse*.

C'est le DNS qui permet, par exemple, de résoudre l'association entre le nom de domaine de la machine `ftp.linux.org` et son adresse IP, `199.249.150.4`. La résolution de nom est l'opération permettant d'obtenir l'adresse IP `199.249.150.4` à partir du nom de domaine `ftp.linux.org` et la résolution inverse est l'opération permettant d'obtenir le nom de domaine `ftp.linux.org` à partir de l'adresse IP `199.249.150.4`.

Le client *DNS* est implémenté dans les bibliothèques de fonctions utilisées par les applications (sous Unix ces fonctions sont regroupées dans une bibliothèque spécifique, *libresolv* ou bien directement dans la bibliothèque de base, la *libc*) ; cet ensemble de fonctions permettant la résolution de noms de domaines s'appelle le *resolver*.

27. Ces protocoles ne sont pas uniquement destinés à *HTTP*, et peuvent être associés à n'importe quel protocole fonctionnant en mode connecté comme TCP, mais c'est avec *HTTP* qu'ils sont utilisés le plus souvent (préfixe de protocole `https:` sur les *URL*).
28. `http://httpd.apache.org`
29. l'application de téléphonie sur IP *Skype* par exemple peut utiliser le protocole HTTP comme protocole de transport pour la voix et la vidéo.
30. h323 est progressivement abandonné au profit de SIP, plus simple et plus générique, mais de nombreux matériels et logiciels exploitant h323 existent encore.

Le *DNS* utilise le protocole UDP [31]. Les datagrammes transmis sont soit des requêtes demandant une adresse à partir d'un nom de domaine (ou l'inverse), soit des réponses à une requête, constituée d'un nom de domaine et d'une adresse IP correspondant au nom de domaine.

Le serveur DNS le plus répandu sur Internet est *BIND* (*Berkeley Internet Name Domain*) maintenu par *Internet Software Consortium* (sous Unix, le programme correspondant à *BIND* est *named* (*NAME Daemon*)).

Autres applications

— *telnet* et *ssh* permettent d'ouvrir une session distante en mode ligne de commande et d'y connecter un terminal local.
— *VNC* (*Virtual Network Computing*) et *RDP* (*Remote Desktop Protocol*) permettent l'accès distant en mode graphique.
— *ping* utilise un des types de messages définis par le protocole ICMP : *ICMP echo*. Le protocole ICMP définit le comportement suivant : lorsqu'une machine reçoit un paquet *ICMP echo* dont elle est destinataire (rappel : un routeur peut très bien recevoir un paquet et ne pas en être destinataire), elle renvoie à l'expéditeur de ce paquet un autre paquet *ICMP echo reply*. La commande *ping* permet ainsi de tester le réseau (fonctionnement de la couche IP) entre deux points de ce réseau. La commande *ping* est standard sous Unix et est livrée avec la plupart des autres systèmes d'exploitation implémentant TCP/IP.
— *traceroute* utilise également le protocole ICMP et permet d'afficher les différents routeurs par lesquels peuvent passer un paquet pour aller d'un point à un autre du réseau.

31. dans certains cas, il peut utiliser le protocole TCP, mais l'utilisation d'UDP est la plus courante.

IP

Le protocole IP est aujourd'hui le protocole le plus utilisé pour la construction de réseaux locaux, de réseaux d'entreprises, de réseaux d'opérateurs ainsi que dans sa vocation initiale, c'est-à-dire, l'interconnexion de réseaux pour la constitution d'un réseau global : Internet [1].

Les caractéristiques les plus générales du protocole IP sont la technique de *commutation de paquets*, un fonctionnement asynchrone et sur la base du meilleur effort de chaque élément du réseau, sans garantie de service.

2.1 IPv4

Le protocole IP (*Internet Protocol*) [RFC 791] permet de transmettre des petites quantités de données, appelées *paquets* [2], d'une machine vers une ou plusieurs autre sur une interconnexion de réseaux informatiques à commutation de paquets. Il définit des adresses de longueur fixe (4 octets, dans IPv4 [3] pour représenter les machines sur le réseau.

Le protocole IP est basé sur une *remise au mieux* (*best effort delivery*) des paquets. IP n'implémente aucun mécanisme de contrôle de fiabilité du transport de paquet, ni de contrôle de séquencement des paquets. IP repose sur des protocoles de liaison et de réseaux locaux pour transmettre un paquet d'une machine à une autre ou d'une machine à une passerelle vers un autre réseau [4].

2.1.1 Fonctionnement global

Le protocole IP fournit deux fonctionnalités : l'*adressage* et la *fragmentation*. Les machines d'interconnexion de réseaux (appelées *routeurs*) utilisent des adresses (adresses IP), contenues dans les entêtes des paquets pour transmettre les paquets jusqu'à leur destination. L'opération qui consiste à choisir un chemin pour transmettre un paquet s'appelle le *routage*. Les implémentations d'IP sur les machines (ordinateurs ou routeurs) utilisent les informations contenues dans les entêtes des paquets pour fragmenter des paquets en paquets plus petits ou réassembler des paquets afin d'adapter la taille des paquets à ce que supporter la liaison acheminant les paquets [5].

IP traite chaque paquet comme une entité indépendante, n'ayant aucun lien avec aucun autre paquet. Il n'y a aucune notion de connexion ou de circuit logique.

2.1.2 Format d'un paquet IP

La figure 2.1 est tirée de la RFC 791 [RFC 791] (les noms des champs sont ceux de la RFC 791) et représente l'entête d'un paquet IP. Cette figure se lit de gauche à droite et de bas en haut. Les

1. Les initiales de ce protocole signifient d'ailleurs *Internet Protocol*.
2. Le terme *datagramme* était utilisé à l'origine, à la place de *paquet* [RFC 791], mais le terme de *paquet* lui est presque toujours substitué aujourd'hui [RFC 2460].
3. La version 4 du protocole IP.
4. utilisant éventuellement un protocole de liaison ou de réseau local différent.
5. par exemple, pour mettre un paquet dans une trame Ethernet ou PPP, le paquet doit faire moins de 1500 octets.

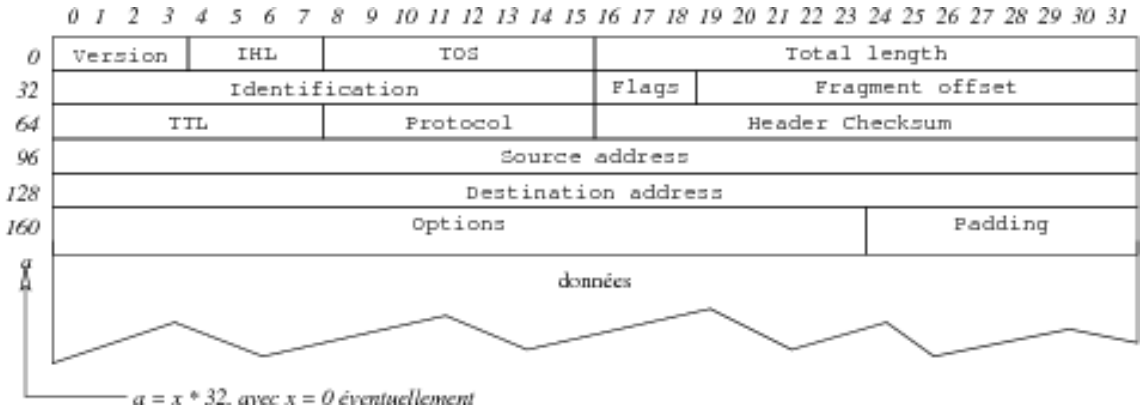


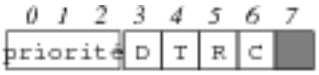
FIGURE 2.1 – Format d'un paquet IP

nombres en italique (étiquettes des lignes et des colonnes) permettent de calculer l'adresse des champs (en bits) dans le paquet. L'adresse d'un champ se calcule en ajoutant le nombre correspondant à la ligne de ce champ + le nombre le plus à gauche de la colonne du champ. Le champ `Flags` a par exemple l'adresse $32 + 16 = 48$ (c'est donc le 6ème octet de l'entête).

La version représente la version du protocole IP. Ce champ vaut 4 pour IPv4 et 6 pour IPv6. Les modules[6] IP vérifient, avant de chercher à traiter un paquet, que la version IP du paquet correspond bien à la version du protocole IP qu'ils reconnaissent. Dans le cas contraire, le paquet est rejeté.

La longueur de l'entête IP (`IHL`, pour `IP Header Length`) représente la longueur (exprimée en nombre de mots de 32 bits) de l'entête IP. Ce champ est nécessaire car la longueur de l'entête IP n'est pas constante (le champ `options`, inclus dans l'entête est facultatif et sa longueur est variable).

Le type de service (`TOS`, *Type Of Service*) permet de définir, de manière générale, la qualité de service désirée. Cette information permet aux routeurs d'adapter le choix d'un chemin pour un paquet en fonction de la nature du paquet. Il se décompose en 6 champs :



- Les trois premiers bits indiquent la priorité du paquet, variant de 0 à 7 du moins prioritaire (0) au moins prioritaire (7). La plupart du temps, ce champ vaut 0 (les autres valeurs sont essentiellement réservées au messages de contrôle).
- Les quatre bits suivants servent à spécifier ce qu'il faut privilégier lors du traitement du paquet (lors d'un routage, notamment) [RFC 791], [RFC 2474] :
  - Si `D` (*Delay*) vaut 1, les modules IP tentent de minimiser le délai d'acheminement du paquet (le délai d'établissement d'une liaison pour commencer à transmettre le paquet).
  - Si `T` (*Throughput* (= débit)) vaut 1, les modules IP tentent de maximiser le débit de la liaison pour transmettre le paquet.
  - Si `R` (*Reliability* (= fiabilité)) vaut 1, les modules IP tentent d'optimiser la fiabilité de la transmission.
  - Si `C` (*Cost* = (= coût)) vaut 1, les modules IP tentent de minimiser le coût de la transmission.

  En général, le choix (lorsqu'il y en a un) d'une liaison plutôt qu'une autre est un compromis (on ne peut pas forcément avoir une liaison optimisant les quatre critères à la fois). Les modules IP ne sont absolument pas tenus d'honorer ces exigences. Ce ne sont que des indices.

6. tout composant, matériel ou logiciel, implémentant le protocole IP.

```
4500 0036 7405 4000 4006 980a c134 5605
c134 5644
```

FIGURE 2.2 – une entête IP

Les applications utilisant le réseau de manière interactive (*telnet*, par exemple) vont chercher à optimiser le temps de réponse, donc à optimiser les délai d'établissement des liaisons (bit `D`). Par contre, les applications transférant des données (*ftp*, par exemple), vont chercher à optimiser le débit.

La longueur totale (`Total length`) représente la taille (en octet) du paquet (entête + données). On en déduit qu'un paquet IP ne peut pas faire plus de 65535 octets.

L'identification, les drapeaux et le déplacement de fragment (`Identification`, `Flags` et `Fragment offset`) sont utilisés pour fragmenter ou réassembler des paquets IP. Leur rôle sera détaillé dans la prochaine section.

La durée de vie (`TTL`, *Time To Live*) indique le nombre maximal de routeurs que peut traverser un paquet avant d'atteindre sa destination. Lorsqu'un paquet traverse un routeur, sa TTL est décrémenté de 1. Lorsqu'elle atteind 0, le paquet est détruit et un message ICMP *Time exceeded* est envoyé à la machine origine du paquet.

Le protocole (`Protocol`) indique le protocole de transport utilisé pour créér le paquet. Les valeurs courantes sont les suivantes :

Champ protocole	Protocole de transport
0x01	ICMP
0x02	IGMP
0x06	TCP
0x11	UDP

La somme de contrôle d'entête (`Header Checksum`) permet de vérifier que l'entête IP est correctement transmise. Elle ne vérifie que l'entête et non les données. La somme de contrôle correspond au *complément à 1* des *sommes en complément à 1* des mots de 16 bits de l'entête IP. L'algorithme suivant permet de calculer la somme de contrôle :

— Le champ de somme de contrôle de l'entête IP est mis à zéro. La somme de contrôle calculée est également mise à zéro.
— Pour chaque mot de 16 bits de l'entête IP, on ajoute la valeur de ce mot à la somme de contrôle.
— Lors de l'ajout de la valeur d'un mot de 16 bits à la somme de contrôle, on vérifie que le résultat est inférieur à `0xffff`. Si le résultat est supérieur à `0xffff` (retenue sur le bit de poids fort), on ajoute 1 à la somme de contrôle (somme en complément à 1). On ne garde que les 16 bits de poids faible de la somme de contrôle (*et* logique bit à bit avec la valeur `0xffff`).
— On complémente à 1 (on inverse tous les bits) de la somme de contrôle.

Dans l'entête IP de l'exemple de la figure 2.2, la somme de contrôle (`0x980a`, au 81ème bit, soit le 6ème mot de 16 bits) peut se recalculer de la façon décrite par la figure 2.3 :

Pour vérifier l'intégrité de l'entête des paquets, les modules IP recalculent cette somme de contrôle de la même façon et comparent les sommes obtenues. A chaque fois qu'un paquet est traité par un routeur, la somme de contrôle doit être recalculée et le champ `header checksum` doit être mis à jour ; en effet, lorsqu'un paquet est traité par un routeur, certains champs de l'entête IP (`TTL` et `Total length`[7], notamment) peuvent être modifiés.

L'adresse IP source (`Source address`) est l'adresse IP de la machine ayant émis le paquet (la machine ayant mis le paquet pour la première fois sur le réseau).

L'adresse IP destination (`Destination address`) est l'adresse IP de la machine destinataire finale du paquet.

7. Si le paquet est fragmenté

```
    0x4500
+   0x0036
-----------
    0x4536
+   0x7405
-----------
    0xb93b
+   0x4000
-----------
    0xf93b
+   0x4006
-----------
    0x13941   > 0xffff ⇒        0x13941
                              &   0xffff
                              -----------
                                  0x3941
                              +        1
                              -----------
                                  0x3942

    0x3942
+   0x0000
-----------
    0x3942
+   0xc134
-----------
    0xfa76
+   0x5605
-----------
    0x1507b   > 0xffff ⇒        0x1507b
                              &   0xffff
                              -----------
                                  0x507b
                              +        1
                              -----------
                                  0x507c

    0x507c
+   0xc134
-----------
    0x111b0   > 0xffff ⇒        0x111b0
                              &   0xffff
                              -----------
                                  0x11b0
                              +        1
                              -----------
                                  0x11b1

    0x11b1
+   0x5644
-----------
    0x67f5    0x67f5 = 0x980a
```

FIGURE 2.3 – Exemple de calcul de somme de contrôle de l'entête IP de la figure 2.2

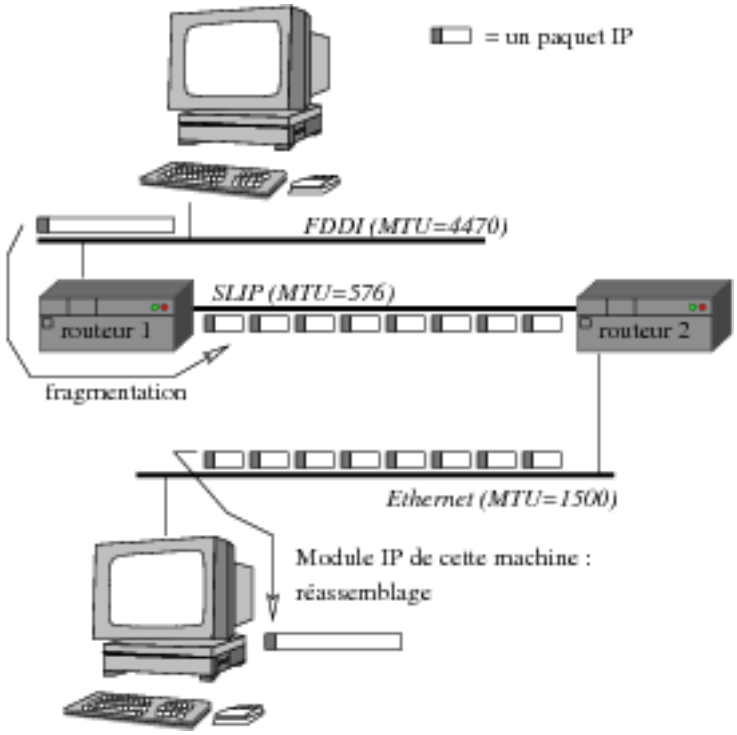


FIGURE 2.4 – MTU et fragmentation/assemblage d'un paquet IP

Le champ d'options (`Options`) est une liste optionnelle (il peut ne pas y avoir d'options du tout) d'options. Ces options servent à spécifier des comportements particuliers (par exemple le routage par la source : le chemin que doit prendre le paquet est spécifié lors de sa création, par l'émetteur du paquet) pour certains paquets et sont rarement utilisées.

Un bourrage (`padding`) est ajouté après les options pour que l'entête ait une taille multiple de 32 bits.

2.1.3 Fragmentation et réassemblage des paquets IP

MTU

Le champ longueur dans l'entête du paquet IP, codé sur 2 octets, limite la taille d'un paquet à 65535 octets (64 Ko). Pour optimiser les performance du réseau, il est préférable qu'un paquet IP puisse être encapsulé dans une seule trame de liaison. Les trames n'ont cependant pas toutes les même longueurs (cette longueur maximale s'appelle la *MTU*, *Maximal Transfer Unit*) (par exemple, *SLIP*[8] (*Serial Line IP*) a un *MTU* de 576 octets et Ethernet a un *MTU* de 1500 octets.

Si la *MTU* d'une liaison est suffisamment grande pour permettre l'encapsulation d'un paquet IP dans une trame, le paquet sera encapsulé tel quel ; dans le cas contraire, il devra être fragmenté. (exemple sur la figure 2.4).

Fragmentation

Dans l'exemple de la figure 2.4, le routeur 1 reçoit un paquet plus grand que la MTU du réseau sur lequel il doit le renvoyer pour l'acheminer vers sa destination. Il va donc devoir fragmenter ce paquet en plusieurs paquets plus petit (de taille compatible avec la MTU de la liaison SLIP), afin de pouvoir encapsuler chaque paquet dans une trame.

La taille de fragmentation (taille des paquets résultat) doit être la plus grande possible (mais inférieure ou égale au MTU de la liaison à utiliser), et multiple de 8 octets.

8. Un protocole de liaison permettant de faire circuler des paquets IP sur une liaison série asynchrone.

Les paquets créés lors d'une fragmentation ont les mêmes entêtes que le paquet d'origine (paquet à fragmenter) à quelques nuances près :

— Le champ de longueur totale (`Total length`) vaut la taille du fragment + la taille de l'entête.
— Tous les fragments correspondant à un même paquet d'origine ont la même valeur pour le champ `Identification`.
— Le champ *drapeaux* (`Flags`) permet de contrôler la fragmentation et le réassemblage, sa signification est la suivante :

0	1	2
0	DF	MF

Bit 0 : vaut toujours 0.

Bit 1 : `DF` (*Don't Fragment*) : si ce bit est à 1, le paquet ne doit pas être fragmenté. Si un routeur rejettera un tel paquet si il est incapable de le délivrer tel quel, sans fragmentation.

Bit 2 : `MF` (*More Fragments*) : ce bit est mis à 0 pour le dernier fragment correspondant au paquet d'origine.

— Le déplacement de fragment (`Fragment offset`) indique à quel endroit du paquet d'origine le fragment correspond. Il est exprimé en unités de 8 octets (64 bits).

Un paquet correspondant à un fragment peut lui être lui-même fragmenté à nouveau. Dans ce cas, le déplacement de fragment (`Fragment offset`) est relatif au paquet d'origine, et l'identification du paquet reste la même.

Le réassemblage des paquets

Un paquet n'est réassemblé que lorsqu'il atteint sa destination finale. Même si le paquet traverse des réseaux avec un plus grand MTU, les routeurs ne réassemblent pas des petits fragments (cf. figure 2.4).

Pour réassembler les fragments et reconstituer le paquet d'origine, le module IP de la machine destinataire du paquet doit attendre tous les fragments. Afin de ne pas garder indéfiniment des fragments dans le cas ou un fragment n'arrive jamais, le module IP de la machine réassemblant les paquets décrémente à intervalles réguliers le champ `TTL` des fragments qu'il a reçus et détruit les fragments si leur `TTL` atteint 0.

Une fois tous les fragments reçus, le module IP peut réassembler les fragments, ayant la même valeur pour le champ `Identification`, grâce aux champs `MF` et `Fragment offset`.

2.1.4 Le routage

Le routage consiste à choisir un chemin pour transmettre un paquet IP à travers le réseau. Un *routeur* est une machine possédant au moins deux interfaces, chacune de ces interfaces étant connectée à des réseaux physiques différents. Son rôle est d'émettre sur une de ses interfaces un paquet qu'il a reçu sur une autre de ses interfaces, dans l'objectif d'acheminer ce paquet vers son destinataire final. Un ordinateur qui n'est pas routeur [9] est soit un expéditeur initial d'un paquet, soit un destinataire final.

Remise directe / remise indirecte

Il y a *remise directe* d'un paquet d'un expéditeur vers un destinataire lorsque les deux machines sont sur le même réseau physique. Par exemple, dans le cas d'un réseau local Ethernet, l'adresse physique du destinataire pourra être déterminée en utilisant le protocole ARP, et l'expéditeur n'aura plus qu'à encapsuler le paquet dans une trame Ethernet avec l'adresse physique du destinataire.

9. Il existe un terme anglais pour désigner une telle machine : *host*, la traduction littérale en français de ce terme (*hôte*) est rarement employée

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
193.52.86.68    0.0.0.0         255.255.255.255 UH    0      0      264 eth0
193.52.86.0     0.0.0.0         255.255.255.0   U     0      0     2545 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0       47 lo
0.0.0.0         193.52.86.1     0.0.0.0         UG    0      0      110 eth0
```

FIGURE 2.5 – Table de routage d'un ordinateur (non routeur)

Il y a *remise indirecte* d'un paquet lorsqu'au moins un routeur est nécessaire pour l'acheminement du paquet. Le mécanisme de remise directe sera tout de même mis en œuvre pour transmettre le paquet de l'expéditeur au premier routeur traversé par le paquet ainsi que du dernier routeur au destinataire.

Tables de routage

Chaque module IP (que ce soit pour un routeur ou pour un ordinateur normal) maintient une *table de routage* permettant de déterminer, en fonction du destinataire final d'un paquet, si ce paquet doit être acheminé par remise directe ou par remise indirecte et, dans le cas de la remise indirecte, vers quel routeur envoyer le paquet.

Chaque entrée d'une table de routage correspond à une façon de remettre un paquet. Pour envoyer un paquet, un module IP lit la table de routage de haut en bas, de manière séquentielle, et compare l'adresse IP de destination du paquet à chaque entrée de la table de routage. Lorsqu'une entrée est trouvée, le module IP peut déterminer comment l'acheminer.

Dans l'exemple [10] de la figure 2.5, la table de routage indique par exemple qu'un paquet peut être envoyé à destination d'une machine ayant l'adresse IP `193.52.86.`x par remise directe en utilisant l'interface réseau `eth0` [11].

La signification des différents champs est la suivante :

Destination indique une destination pouvant être une machine (1ère ligne de la figure 2.5) ou un ensemble de machines.

Gateway indique l'adresse IP du routeur a utiliser dans le cas d'une remise indirecte (0.0.0.0 dans le cas d'une remise directe). Il doit exister dans la table de routage au moins une entrée permettant de délivrer un paquet à ce routeur.

Genmask indique un masque (appelé également *netmask* ou masque de réseau) appliqué à l'adresse IP de destination d'un paquet avant de la comparer à la destination de la première colonne. Le masque est appliqué à une adresse IP en effectuant un *et* logique entre l'adresse et le masque.

Flags indique la nature de l'entrée et contient un ou plusieurs caractères ayant les significations suivantes :

- `U` l'entrée est en service.
- `H` l'entrée spécifie une destination qui est une machine. Dans ce cas, le masque de réseau doit être `255.255.255.255`.
- `G` l'entrée correspond à une remise indirecte. Si `G` n'est pas présent, c'est obligatoirement une remise directe.
- `D` l'entrée a été créée par une redirection.
- `M` l'entrée a été modifiée par une redirection.

Metric indique la distance qui sépare la machine contenant la table de routage de la destination de l'entrée. Cette distance est généralement exprimée en nombre de traversée de routeurs (*hops*) nécessaires pour acheminer un paquet. Dans l'exemple de la figure 2.5, cette distance vaut 0, car les paquets sont délivrés soit en remise directe, soit à travers un routeur se trouvant sur le même réseau physique.

10. A peu près toutes les implémentations de TCP/IP proposent les commandes `route` et `netstat -r`, permettant (entre autres) d'afficher la table de routage.
11. La première carte Ethernet de la machine.

Ref indique le nombre de fois où l'entrée est utilisée au moment de la consultation de la table de routage.

Use indique le nombre de paquets ayant utilisé cette entrée.

Iface représente l'interface physique utilisée. Dans l'exemple de la figure 2.5, `eth0` correspond à une interface Ethernet et `lo` à une pseudo-interface spéciale, appelée *loopback* (bouclage) présente dans tout module IP. Tout paquet IP envoyé par cette interface est automatiquement renvoyé à cette même interface. Cette interface permet de se connecter à soi-même. Elle est principalement utilisée pour faire fonctionner sur une seule machine les parties serveur et client d'applications client-serveur (exemple : consulter un serveur WWW depuis la machine sur lequel tourne le serveur WWW). Par convention cette interface a le numéro IP `127.0.0.1`.

Exemple

La machine ayant la table de routage de la figure 2.5 doit envoyer un paquet ayant comme adresse IP de destination `193.52.87.33`.

— 1ère entrée : `193.52.87.33` & `255.255.255.255` = `193.52.87.33` $\neq$ `193.52.86.68` $\Rightarrow$ ne correspond pas.
— 2ème entrée : `193.52.87.33` & `255.255.255.0` = `193.52.87.0` $\neq$ `193.52.86.0` $\Rightarrow$ ne correspond pas.
— 3ème entrée : `193.52.87.33` & `255.0.0.0` = `193.0.0.0` $\neq$ `127.0.0.0` $\Rightarrow$ ne correspond pas.
— 4ème entrée : `193.52.87.33` & `0.0.0.0` = `0.0.0.0` = `0.0.0.0` $\Rightarrow$ correspond. Ce sera donc cette entrée qui sera utilisée pour envoyer le paquet.

Routage dynamique

La table de routage peut être configurée manuellement et chargée une fois pour toute lors du démarrage des machines. Dans ce cas, la table de routage est dite *statique*.

Cette méthode convient parfaitement lorsque la structure du réseau est simple et qu'il y a peu de choix de chemins pour acheminer les paquets. Lorsque la structure du réseau est plus complexe, les choses se compliquent.

En effet, dans le cas du routage statique, si une machine ou une liaison tombe en panne, la seule possibilité pour pouvoir délivrer un paquet est de modifier manuellement la (ou les) table(s) de routage (dans le cas où il existe d'autres solutions pour pouvoir acheminer le paquet). Plusieurs techniques ont été développées pour mettre à jour automatiquement les tables de routage pour prendre en compte les pannes de réseau. Ces techniques sont dites de *routage dynamique*.

Il existe plusieurs protocoles de routage dynamique, regroupés en deux grandes familles :

— Les protocoles de gestion de *systèmes autonomes* (*IGP*, *Interior Gateway Protocol*) permettent d'établir les tables des routeurs internes de systèmes autonomes. Un système autonome est un réseau qui assure la connexité totale de toutes ses machines. Les deux protocoles de gestion de systèmes autonomes les plus répandus sont *RIP* (*Routing Information Protocol*), [RFC 2453] et *OSPF* (*Open Shortest Path First Protocol*) [RFC 2328].
— Les protocoles d'interconnexion de systèmes autonomes (*EGP*, *Exterior Gateway Protocol* ou *BGP*, *Border Gateway Protocol*) : Internet est une interconnexion de systèmes autonomes. Les routeurs connectant les systèmes autonomes entre eux (routeurs externes) dialoguent entre eux par un de ces protocoles. Ces protocoles servent essentiellement à la gestion et à l'interconnexion de backbones. Voir [RFC 911] et [RFC 1771] pour plus d'information.

Exemple

Dans l'exemple de la figure 2.6, les trois routeurs *A*, *B* et *C* sont connectés à trois réseaux Ethernet. Les trois routeurs sont connectés entre eux par des liaisons PPP.

Au départ, la table de routage du routeur *A* est :

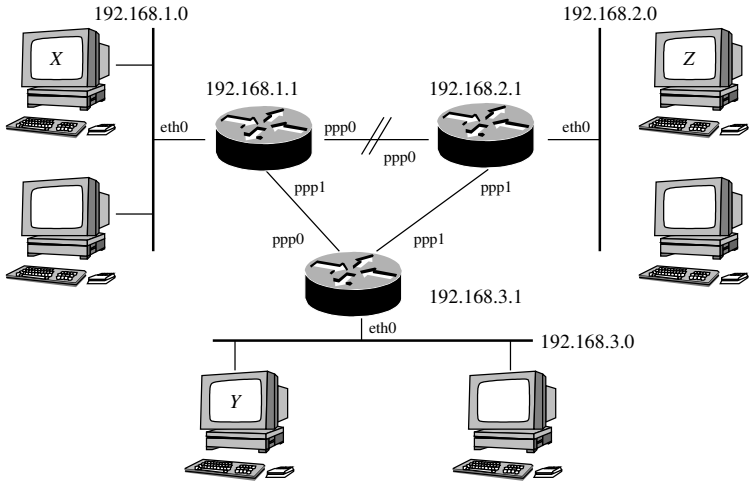


FIGURE 2.6 – Interconnexion de trois réseaux. Exemple tiré de [Dawson & Rubini, 1998]

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.1     0.0.0.0         255.255.255.255 UH    0      0      0   eth0
192.168.1.1     0.0.0.0         255.255.255.255 UH    0      0      0   ppp0
192.168.1.1     0.0.0.0         255.255.255.255 UH    0      0      0   ppp1
192.168.2.1     0.0.0.0         255.255.255.255 UH    0      0      0   ppp0
192.168.3.1     0.0.0.0         255.255.255.255 UH    0      0      0   ppp1
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0      0   eth0
192.168.2.0     192.168.2.1     255.255.255.0   UG    0      0      0   ppp0
192.168.3.0     192.168.3.1     255.255.255.0   UG    0      0      0   ppp1
```

Cette configuration fonctionne normalement ; les machines X, Y et Z peuvent communiquer. Si le lien entre les routeurs A et B sont rompus, la machine X ne peut plus envoyer de paquet vers la machine Z, puisque ces paquets seraient envoyés par le routeur A vers l'interface `ppp0` dont le lien est rompu (avant dernière ligne de la table de routage). Par contre, la machine A peut continuer à envoyer des paquets vers la machine Y car le lien correspondant à l'interface `ppp1` du routeur A fonctionne toujours. De même, la machine Y peut continuer à envoyer des paquets à la machine Z.

Le routeur A peut toujours communiquer avec le routeur C et le routeur C peut communiquer avec le routeur C. Pour que le routeur A envoie ses paquets à destination du routeur B en "passant" par le routeur C, il faudrait ajouter la ligne suivante à la table de routage du routeur A :

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.2.0     192.168.3.1     255.255.255.0   UG    0      0      0   ppp1
```

et supprimer les lignes correspondant à l'interface `ppp0` qui ne fonctionne plus.

Les protocoles de routage dynamique effectuent automatiquement ce type d'opération.

Quelques protocoles de routage dynamique

1. **RIP** (*Routing Information Protocol*) est un protocole de routage dynamique de type IGP. Il fonctionne sur le principe du *vecteur de distance* : chaque routeur annonce à ses voisins (routeurs avec lesquels il partage une liaison) la liste des destinations vers lesquelles il peut acheminer des paquets, et, pour chaque destination, une métrique, exprimée en nombre de sauts pour atteindre cette destination. Il existe plusieurs versions de RIP. La version 1 n'est plus très utilisée. La version 2 diffère de la version 1 en permettant l'annonce des routes en utilisant une diffusion multicast et permet l'authentification des routeurs entre eux. La version RIPng inclue le support d'IPv6.

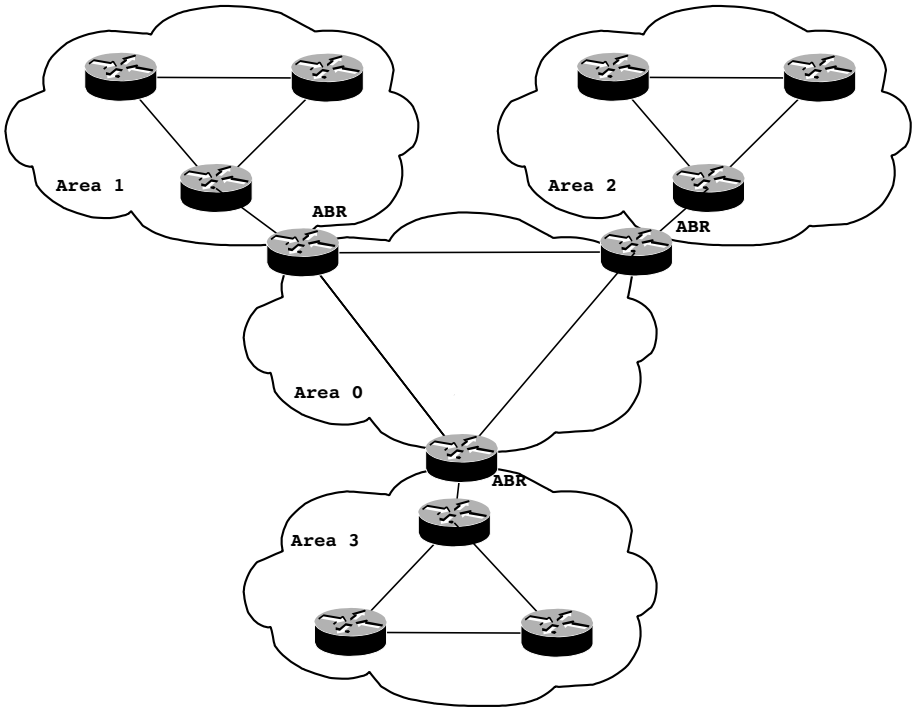


FIGURE 2.7 – routage OSPF sur 4 aires.

2. **OSPF** (*Open Shortest Path First*) est un protocole de routage dynamique IGP à *état de lien*. Chaque routeur OSPF détermine des relations d'adjacence (découverte des voisins) au moyen de messages *hello* envoyés à intervalles réguliers. Chaque routeur communique ensuite à ses voisins la liste des réseaux auxquels il est connecté. De proche en proche, chaque routeur renseigne une base de données (*LSDB*, *Link State DataBase*) qui est la même pour tous les routeurs. L'existence de cette base de données permet de limiter la taille des messages échangés entre les routeurs. Chaque routeur utilise enfin l'algorithme du plus court chemin de Dijkstra afin de déterminer le chemin le plus court vers chacun des réseaux renseignés dans la *LSDB*.

   Le protocole OSPF définit également la notion d'*area* permettant de segmenter un réseau complexe en plusieurs réseaux plus simples (voir figure 2.7). Le protocole OSPF fonctionne comme indiqué ci-dessus à l'intérieur de chaque aire. Les routeurs à l'intersection de deux aires (dits *Area Border Router* ou *ABR*) annoncent des résumés de routes d'une aire à l'autre. L'aire 0 a un rôle particulier d'interconnexion de toutes les autres aires.

   Dans chaque *area*, un routeur, appelé *DR* (*Designated Router*) est désigné pour collecter toutes les informations d'adjacence et les diffuser aux autres routeurs. Ce mécanisme évite de multiplier les échanges d'informations d'adjacence (proportionnel au carré du nombre de routeurs dans le cas d'un réseau dense). Un routeur de secours (*BDR*, *Backup Designated Router*) est déterminé pour servir en cas de défaillance du *DR*.

   Il existe deux versions d'OSPF : OSPFv2 ne fonctionnant qu'en protocole IPv4 et OSPFv3 supportant IPv6.

3. **IS-IS** (*Intermediate System to Intermediate System*) est un protocole de routage dynamique IGP similaire à OSPF, mais contrairement à ce dernier, il est multiprotocole (en particulier, la même version supporte à la fois IPv4 et IPv6). Tout comme OSPF, *IS-IS* est un protocole à état de lien, utilise des messages *hello* en protocole multicast pour déterminer les adjacences ainsi que l'algorithme de Dijkstra pour établir les plus courts chemins. *IS-IS* étend la notion d'aire telle que définie par OSPF : les routeurs *IS-IS* peuvent être à l'intérieur d'une aire (ils sont alors dits *Level 1*), à l'extérieur de toute aire (ils sont alors dits *Level 2*) ou alors à l'interconnexion entre une aire et un routeur *Level 2* (ils sont alors dits *Level 1-2*) (voir figure

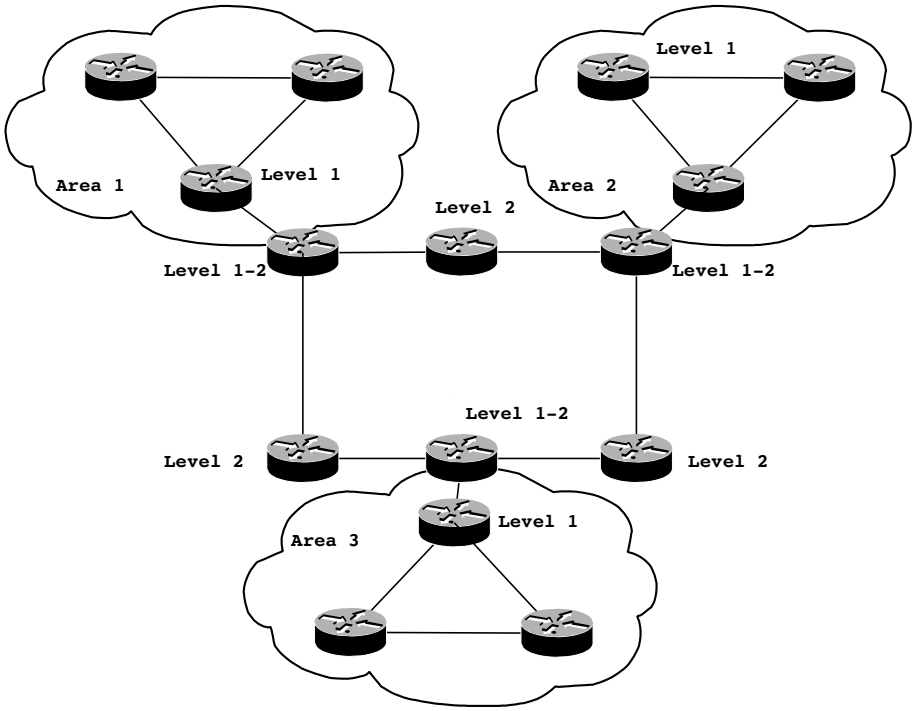


FIGURE 2.8 – Routage IS-IS entre 3 aires.

2.8).

4. **BGP** (*Border Gateway Protocol*) est un protocole de routage dynamique EGP. Contrairement à des protocoles IGP tels que RIP ou OSPF, BGP ne cherche pas à établir l'ensemble des routes possibles, mais établir l'accessibilité de chaque préfixe. BGP permet d'interconnecter des *systèmes autonomes* (*Autonomous System* ou *AS*), un protocole IGP tel qu'OSPF pouvant être utilisé pour assurer l'interconnexion des noeuds à l'intérieur de chaque *AS* (voir figure 2.9). Contrairement aux protocoles IGP qui utilisent généralement des messages UDP multicast, BGP réalise ses échanges au travers d'une connexion TCP (port 179). Il fonctionne donc en mode point à point et les interconnexions entre routeurs BGP doivent être configurées manuellement. Les informations sur l'accessibilité des préfixes sont échangées sous la forme d'une succession d'ajouts ou de suppressions de chemins. Ces chemins contiennent la liste des *AS* permettant d'atteindre le préfixe associé [12]. Toutefois chaque routeur BGP ne conserve qu'un seul chemin pour chaque préfixe.

   une variante de BGP, *iBGP*, peut également être utilisée comme un protocole IGP. Son intérêt est principalement pour réduire le nombre de déclarations de pairs BGP.

2.2 ICMP

Le protocole *ICMP* (*Internet Control Message Protocol*) permet de transporter les messages d'erreur. Ces erreurs servent aux autres couches (TCP ou des applications, par exemple) à détecter certaines anomalies (lors du routage, notamment). Un message ICMP est acheminé dans un paquet IP comme n'importe quel autre donnée. Par contre, le comportement d'IP est modifié lorsqu'il achemine un message ICMP : aucun message ICMP ne doit être généré lors d'une erreur d'acheminement d'un paquet ICMP. Ceci permet d'éviter une avalanche d'erreurs risquant d'engorger le réseau.

12. les *AS* sont numérotés ; les premières spécifications de BGP prévoyaient une numérotation des *AS* sur 16 bits mais les versions récentes permettent une numérotation sur 32 bits. Les *AS* 64512 à 65535 sont des *AS* privés, non utilisés sur Internet.

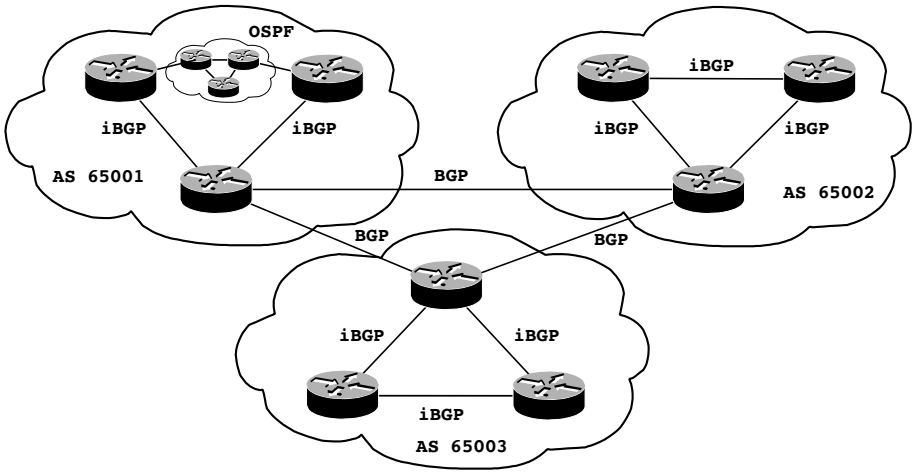


FIGURE 2.9 – Routage BGP entre trois *AS*.

type (1 octet)	code (1 octet)	checksum (2 octets)	arguments longueur (< 64000 octets)

FIGURE 2.10 – Format d'un message ICMP

2.2.1 Format d'un message ICMP

La figure 2.10 décrit le format d'un message ICMP.

Les champs `type` indique la nature du message et le champ `code` sert à préciser le message. Le champ `checksum` est une somme de contrôle portant sur tout le message ICMP (arguments inclus) et se calcule de la même manière que pour la somme de contrôle d'entête des paquets IP (complément à 1 de la somme en complément à 1 du message, découpé par mots de 16 bits.

Le champ `type` peut prendre les valeurs suivantes :

— `0x08` : ICMP *echo* : c'est un type de message ICMP particulier : un module IP, lorsqu'il reçoit un paquet ICMP echo répond en renvoyant à l'adresse IP spécifiée dans le champ `Source address` de l'entête IP, un message ICMP *echo reply*. Ce type de message est essentiellement employé pour tester des réseaux (commande *ping*).

— `0x00` : ICMP *echo reply* : réponse à un message ICMP *echo*.

— `0x03` : ICMP *destination unreachable* Une adresse spécifiée ne peut pas être atteinte. Une machine (ou un routeur) envoie ce message lorsqu'un paquet ne peut pas être acheminé. L'entête et les 8 premiers octets de données du paquet IP qui n'a pas pu être délivré est recopié dans le champ d'arguments du message ICMP. Le champ `code` du message ICMP peut prendre les codes suivants :
  — `0x00` : Réseau inaccessible.
  — `0x01` : Machine inaccessible.
  — `0x02` : Protocole inconnu.
  — `0x03` : Port inaccessible.
  — `0x04` : Une fragmentation est nécessaire (réseau de MTU inférieure à la taille du paquet), mais le bit de non fragmentation (`DF`) de l'entête IP est à 1.
  — `0x05` : Routage à la source impossible.

  D'autre valeurs du champ `code` sont possibles, mais sont utilisées plus rarement.

— `0x04` : Avertissement de congestion. Un élément du réseau peut envoyer ce message pour indiquer qu'il faut lui envoyer moins de paquet.

— `0x05` : Demande de modification de route (entrée dans une table de routage). Ce message permet à un routeur d'avertir une machine qu'une de ses entrée n'est pas correcte et doit être modifiée. Le champ d'arguments du message ICMP contient alors l'entrée à ajouter.

— `0x0b` : La valeur du champ `TTL` de l'entête IP d'un paquet a atteint 0. La valeur du champ

`code` de l'entête ICMP indique si le `TTL` a atteint 0 lors du routage du paquet ou si elle a atteint 0 lors du réassemblage d'un paquet.
— `0x0c` : Entête IP invalide.

2.3 IPv6

L'Internet, sous sa forme actuelle, date du début des années 1980 [RFC 791]. Or, son succès et toutes ses évolutions n'avaient pas été prévues à l'époque. En particulier, trois limites sont aujourd'hui rencontrées :

— la taille de l'espace adressable (2^{32} soit un peu plus de 4 milliards d'adresses), pour l'instant plus grand que le nombre de machines formant l'Internet, mais pour des raisons techniques ou d'organisation, les machines ne sont pas numérotées séquentiellement ; au contraire des plages (préfixes, réseaux, ...) sont alloués.
— La complexité de l'information à distribuer sur les routeurs,
— de nombreuses fonctionnalités qui doivent être mises en œuvre sous la forme d'extensions au protocole IPv4 :
  — les applications synchrones, pair-à-pair, multimédia ou les jeux nécessitent souvent de transmettre les adresses dans les messages, rendant les techniques de translation d'adresses (*NAT*) difficiles à appliquer.
  — Il est quelquefois nécessaire de pouvoir vérifier (authentifier) l'adresse des noeuds du réseau.
  — Il est parfois nécessaire d'assurer la confidentialité des échanges en chiffrant les communications.
  — Certaines applications (la diffusion de vidéos en direct, par exemple) peuvent tirer bénéfice de l'acheminement d'un flux de données vers un ensemble de destinations sans dupliquer ces données lorsque cela n'est pas nécessaire (multicast).
  — Il est parfois nécessaire d'assurer une qualité de service : prioriser certains trafics selon des critères de délais d'acheminement ou de réservation de bande passante, par exemple.
  — Il est de plus en plus nécessaire de tenir compte de la mobilité des machines et des utilisateurs.

  Toutes ces fonctionnalités peuvent être déployées avec IPv4, mais elles dépendent d'extensions parfois difficiles à intégrer et à faire fonctionner au niveau global qu'est celui de l'Internet.

2.3.1 Principales caractéristiques d'IPv6

L'espace d'adressage est, en IPv6, de 2^{128} machines. Le protocole est, malgré tout, simplifié : les entêtes ont une taille fixe, il n'y a pas de somme de contrôle d'entête, il n'y a pas de fragmentation par des routeurs intermédiaires sur le chemin d'un paquet, et le protocole ICMP a un rôle plus étendu.

L'adressage *multicast* est supporté nativement et son utilisation est généralisé à la place de caractéristiques spécifiques de liaisons (broadcast notamment).

IPv6 propose un mécanisme d'autoconfiguration sans état (configuration automatique d'adresses, découverte automatique de voisins et de routeurs).

La mise en place de la qualité de service est simplifiée et l'authentification des noeuds et le chiffrement du trafic sont inclus en intégrant *IPsec*.

2.3.2 Fonctionnement

Les principes généraux de fonctionnement de l'Internet sont les mêmes quelle que soit la version de protocole IP utilisée. IPv6 ne pose pas de nouvelle hypothèse sur l'architecture du réseau. Celle-ci reste identique en IPv6 (fonctionnement non fiable mais robuste). Tout comme IPv4, IPv6 fonctionne sur le principe de la *commutation de paquets*, avec un acheminement individuel des paquets au meilleur effort de chaque routeur du réseau.

IPv6 permet toutefois de repérer tous les paquets appartenant à un même flux (en utilisant le triplet (*adresse source*, *adresse destination*, *flow label*)).

IPv6 est, tout comme IPv4, un protocole de réseau (global) : il ne se préoccupe ni de caractéristiques de liaison, ni de transport en particulier. Les implémentations de protocoles de liaison (ethernet ou PPP par exemple) ou de transport (TCP, UDP ou RTP par exemple), n'ont pas besoin d'être modifiées pour prendre en charge IPv6.

Afin de permettre la migration d'IPv4 à IPv6, un certain nombre de mécanismes ont été prévus, développés et déployés :

— l'encapsulation IPv6 dans IPv4 (*6to4*) consiste à acheminer des paquets IPv6 dans la charge utile de paquets IPv4. Deux réseaux IPv6 distants peuvent ainsi être interconnectés en utilisant un réseau IPv4.
— Des réseaux IPv6 virtuels peuvent ainsi être constitués (*6bone*[13] par exemple). Des fournisseurs de tunnels (*Hurricane Electrics*[14], *gogo6*[15] ou *Sixxs*[16] par exemple) proposent[17] un service d'encapsulation IPv6 dans IPv4 et d'interconnexion de ces tunnels dans l'Internet IPv6.
— Des mécanismes de relai transparent IPv4 vers IPv6 (*NAT64*) permettent à des machines connectées à des réseaux IPv6 uniquement d'accéder à des services fournis par des machines connectées à des réseaux IPv4 uniquement.
— Il est également possible de configurer des machines à *double pile*, c'est-à-dire avec une pile IPv4 et une pile IPv6, pouvant donc bénéficier des deux réseaux.

2.3.3 Niveau de support d'IPv6

Aujourd'hui la plupart des systèmes d'exploitation récents de postes de travail, de serveurs et de routeurs supportent directement IPv6, au moins sous la forme d'une option. Une grande majorité des applications réseau les plus répandues fonctionnent en utilisant IPv6.

Dans la grande majorité des cas, IPv6 peut fonctionner comme seul protocole de réseau ou en double avec IPv4. Dans certains cas, des applications spécifiques peuvent servir de relai.

Un certain nombre d'opérateurs ou de fournisseurs d'accès Internet proposent une interconnexion IPv6 native ou bien en encapsulation.

2.3.4 Entête IPv6

La figure 2.11 présente l'entête IPv6. Les champs ont la signification suivante :

— `Version` : vaut 6 pour IPv6,
— `Priority` : définit la priorité du paquet, selon une définition similaire au champ `DSCP` de l'entête IPv4.
— `Flow Label` : permet d'identifier les paquets correspondant à un même flux, afin de simplifier leur intégration au niveau transport ou pour identifier une classe de priorisation lors de la mise en œuvre de la qualité de service.
— `Payload Length` : taille du paquet.
— `Next Header` : définit le type de l'entête suivant l'entête IPv6. Il peut s'agir d'une entête correspondant à un protocole de transport, une entête définissant une option IPv6 ou une nouvelle entête de réseau (en cas d'encapsulation).
— `Hop Limit` : sa signification est similaire à celle du champ `TTL` d'IPv4.
— `Source Address` : adresse source, sur 128 bits,
— `Destination Address` : adresse destination, sur 128 bits.

Options

Les options peuvent être les suivantes, dans cet ordre :

13. le 6bone est toutefois aujourd'hui terminé.
14. `http://www.tunnelbroker.net`
15. `http://gogo6.com`
16. `http://www.sixxs.net`
17. souvent gratuitement

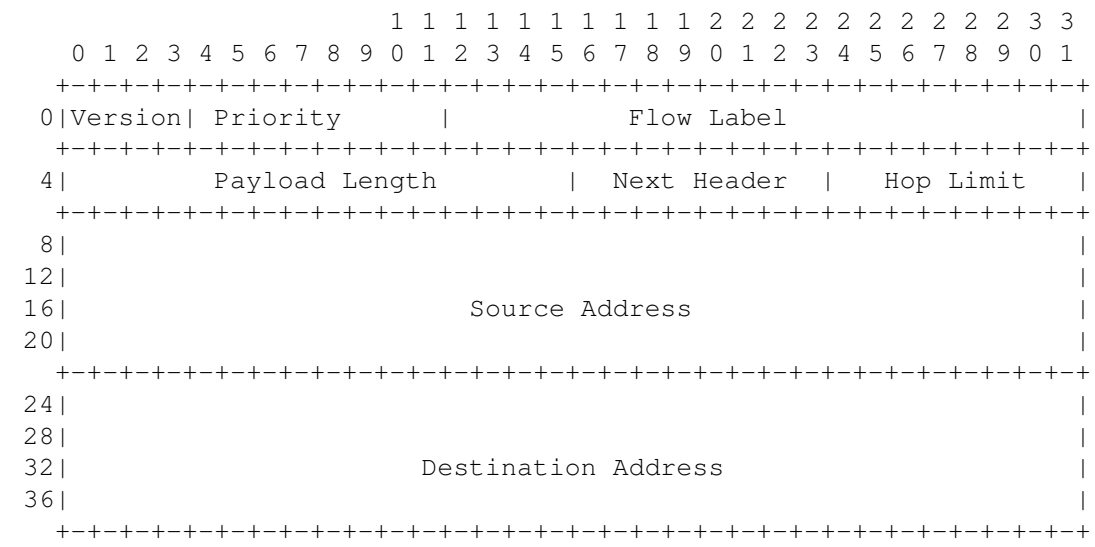


FIGURE 2.11 – Entête IPv6

— options *proche en proche* (*hop-by-hop*) : elles sont examinées par tous les nœuds sur le chemin du paquet. Il s'agit, pour l'essentiel, d'options de bourrage, permettant de garantir une taille des entêtes multiple de 32 bits.
— Les options de *destinataires* ne sont traitées que par le destinataire du paquet. Elles sont utilisées lors de l'établissement de tunnels en particulier.
— Les options de routage sont évaluées par les routeurs et permettent par exemple de spécifier des préférences de routage par l'émetteur du paquet (*source routing*).
— Les options de fragmentation permettent de réaliser des paquets fragmentés.
— Les options d'authentification reprennent les spécifications d'*IPsec AH*. Les options de chiffrement reprennent les spécifications d'*IPsec ESP*.

2.3.5 Adressage IPv6

Les adresses IPv6 ont une taille fixe de 128 bits. Elles sont généralement écrites sous une forme hexadécimale, dans une succession de 8 mots de 16 bits séparés par un caractère `:`, par exemple

```
2001:0db8:0000:0000:0000:0000:0000:cafe
```

Les valeurs 0 en tête de chaque mot peuvent être omises :

```
2001:db8:0:0:0:0:0:cafe
```

Une (et une seule pour toute l'adresse) succession de mots tous à 0 peut être remplacée par le symbole `::` :

```
2001:db8::cafe
```

L'adresse `::` représente une adresse constituée uniquement de 0. Le préfixe correspondant à la route par défaut s'écrit ainsi `::/0`.

Adresses standardisées

La figure 2.12 résume les préfixes IPv6 standardisés.

0000 : :/8	Réservé pour la transition et loopback
0100 : :/8	Réservé
0200 : :/7	Réservé (ex NSAP)
0400 : :/6	Réservé (ex IPX)
0800 : :/5	Réservé
1000 : :/4	Réservé
2000 : :/3	Unicast Global
2001 :db8 : :/32	Préfixe de documentation
2002 : :/16	Migration 6to4
4000 : :/3	Réservé
6000 : :/3	Réservé
8000 : :/3	Réservé
A000 : :/3	Réservé
C000 : :/3	Réservé
E000 : :/4	Réservé
F000 : :/5	Réservé
F800 : :/6	Réservé
FC00 : :/7	Unique Local Unicast
FE00 : :/9	Réservé
FE80 : :/10	Lien-local
FEC0 : :/10	Réservé
FF00 : :/8	Multicast

FIGURE 2.12 – Préfixes IPv6 standardisés

Les protocoles de matériel et de liaison

Bien que certains de ces protocoles soient indépendants de TCP/IP (Ethernet et PPP, par exemple), ce chapitre décrira le fonctionnement des couches matérielles les plus courantes sur Internet : le réseau local *Ethernet* très utilisé dans les entreprises, les administrations et les universités pour leurs réseaux locaux, ainsi que les liaisons point à point *PPP*, utilisées par une grande majorité de fournisseurs d'accès Internet pour les connexions des particuliers au réseau Internet en utilisant des lignes téléphoniques classiques. Ce chapitre terminera par la description du protocole *ARP* permettant à la couche IP de résoudre les adresses IP des machines connectées à un réseau local.

3.1 Ethernet

Ethernet est normalisé par l'*IEEE* (*Institute for Electrical and Electronics Engineers*) sous la norme *IEEE* 802. Il permet de connecter ensemble sur un *bus*, un petit nombre de machines (de 2 à quelque centaines) se trouvant dans le même bâtiment ou le même site industriel. Charles Spurgeon propose un site WWW très complet dédié à Ethernet [Spurgeon, 1999].

3.1.1 Supports physiques

Historiquement, *ethernet* a été conçu de manière indépendante à tout support physique [1].
Ethernet utilise cependant aujourd'hui principalement trois types de support :

1. Les signaux électriques sur des fils de cuivre, généralement cablés sous la forme de paires torsadées. Il existe plusieurs normes :
   — *10 base 5* (*Ethernet épais*), et *10 base 2* (*Ethernet fin*) correspondent à l'utilisation d'un cable coaxial. La bande passante sur ce type de support est de 10 Mbits/s. Ces supports sont aujourd'hui obsolètes.
   — *10 base T*, ou *Ethernet sur paire torsadée* (*twisted pair Ethernet*), figure 3.2 : chaque ordinateur est relié à un point central au moyen d'un cable constitué de 4 conducteurs enroulés les uns autour des autres deux par deux (paire torsadée). Ce câble est identique au câble utilisé par le téléphone ; ainsi, d'anciennes installations téléphoniques peuvent être réutilisées pour le réseau local. Les extrémités des paires torsadées sont généralement des prises au format RJ45. Le point central s'appelle un *hub* [2]. Le *hub* simule une topologie physique en étoile, alors que la topologie logique du réseau est un

1. littéralement, le préfixe *ether* se rapportant à sa définition donnée par Aristote et signifiant tout ce qui peut permettre le passage de l'information.
2. La traduction normale de *hub* en français est *moyeu*. Cette traduction n'est jamais employée, même en français ; le terme anglais est conservé.

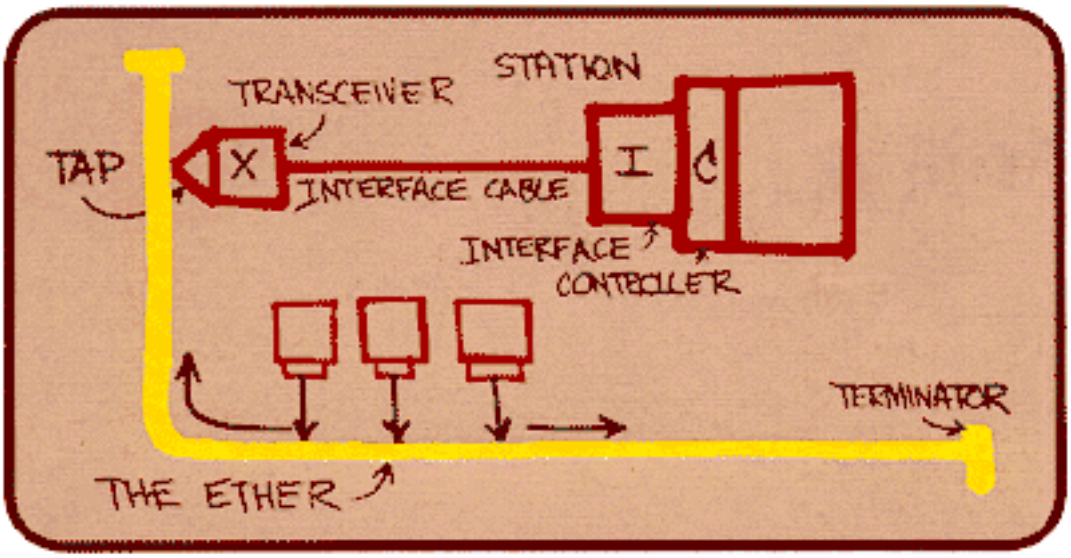


FIGURE 3.1 – Ce schéma a été dessiné par Robert M. Metcalfe en 1976 pour présenter Ethernet à la *National Computer Conference* de Juin 1976. Les termes présents sur cette figure sont ceux de l'époque. (source : [Spurgeon, 1999]).

bus. Le *hub* a également un rôle de *répéteur*, c'est à dire qu'il amplifie et corrige le signal provenant de chaque ordinateur. Pour cette raison, il doit être alimenté. Plusieurs *hub*s peuvent être reliés entre eux.

— *100 base TX* : il utilise les mêmes principes que pour le *10 base T*, mais la bande passante est portée à 100 Mbits/s. tout en utilisant les mêmes types de câbles (paire torsadée). Les hubs utilisés doivent tenir compte de la bande passante à 100 Mbits/s. et sont généralement des commutateurs (ayant une fonction de pont multiport), c'est-à-dire qu'ils sont capables d'interpréter les signaux ethernet de manière à n'acheminer les signaux que vers les ordinateurs auxquels ils sont destinés.

— *1000 base T* : il utilise les mêmes principes que 10 base T et 100 base TX, mais permet une bande passante jusqu'à 1 Gbits/s. Il utilise pour cela 4 paires (donc 8 conducteurs).

La longueur maximum d'un câble ethernet cuivré est de 100 mètres en paire torsadée, quelle que soit la bande passante utilisée. Il existe plusieurs catégories de câblage en paire torsadée :

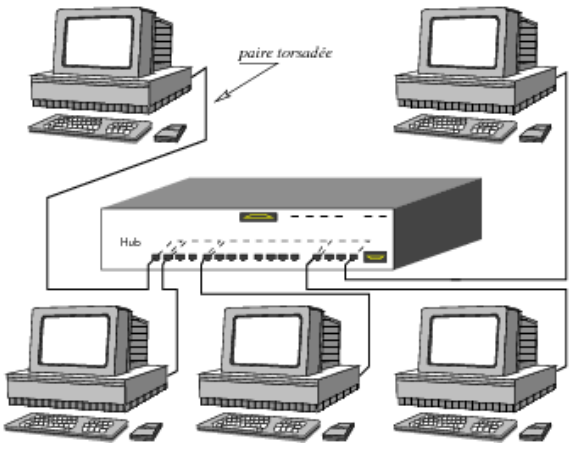


FIGURE 3.2 – Ethernet 10 base T

Adresse de destination	Adresse source	Type	Données	CRC
6 octets	6 octets	2 octets	46 à 1500 octets	4 octets

FIGURE 3.3 – Trame Ethernet (RFC 894)

— Catégorie 3 : 2 paires téléphoniques ; pas de blindage.
— Catégorie 5 UTP (*Unshielded Twisted Pair*) : 4 paires, non blindé.
— Catégorie 5 FTP (*Foiled Twisted Pair*) : 4 paires, blindage autour des 4 paires permettant d'éviter le parasitage.
— Catégorie 6 : 4 paires, blindage autour de chaque paire + blindage autour des 4 paires.

2. Les signaux optiques sur fibres : il existe plusieurs normes (100 base FX, 1000 base LX par exemple, correspondant à une bande passante de respectivement 100 Mbits/s. et 1 Gbits/s.). Les distances possibles vont de quelques centaines de mètres (1000 base SX) à quelques kilomètres (100 base FX, 1000 base LX) voire quelques centaines de kilomètres (1000 base ZX).
3. Les signaux radio : il existe une série de normes [3] (IEEE 802.11) reprenant les définitions d'Ethernet, mais en utilisant des micro-ondes comme support. Elles permettent des débits jusqu'à 54 Mbits/s. sur des distances de l'ordre de la centaine de mètres.

3.1.2 Trames Ethernet

Les données circulent sur le réseau Ethernet sous la forme de *trames*. Ce sont des suites d'octets, dont la longueur varie entre 46 et 1500 octets. Ces trames sont munies d'entêtes décrivant le type d'information qu'elles font circuler et identifiant la machine source et la machine destination de la trame. La figure 3.3 donne le format d'une trame Ethernet [RFC 894] [4].

— Les adresses physiques Ethernet (quelquefois appelées adresses *MAC* (*Medium Access Control*)) sont codées sur 48 bits (6 octets) et sont uniques (il n'existe *n*ormalement pas deux cartes Ethernet avec la même adresse physique). Une adresse particulière, de *broadcast* (adresse dont tous les bits sont à 1 : `ff:ff:ff:ff:ff:ff`), permet d'envoyer une trame à tous les ordinateurs connectés au réseau.
— Le champ *type* spécifie le type des données transmises. Si les données correspondent à un paquet IP, le champ type vaut 0x0800 [5]. Si elles correspondent à une requête ARP, le champ type vaut 0x0806, et enfin, il vaut 0x8035 pour une requête RARP.
— Le champ *CRC* (*Cyclic Redundant Check* ou *vérification à redondance cyclique*) contient une somme de contrôle, calculée sur les données transmises et permettant de vérifier que les données ont été transmises sans erreur.

3.1.3 Echange des trames

Chaque interface Ethernet (transciever ou carte Ethernet) capte toutes les trames circulant sur le cable mais n'envoie à l'ordinateur que celles qui lui sont destinées [6], c'est à dire dont l'adresse de destination est égale à l'adresse physique de la carte ou est égale à l'adresse de broadcast.

3.1.4 Collisions

Il n'y a pas d'autorité centrale qui gère l'accès au câble. Il est donc possible que plusieurs interfaces Ethernet veuillent émettre une trame simultanément. Or, le câble, partagé par toutes

3. Le nom commercial plus connu pour ces normes est *WiFi* (*Wireless Fidelity*).
4. Il existe quelques petites différences entre la normalisation IEEE 802 et la définition d'Ethernet de la RFC 894 ; la norme 802.3 ajoute notamment la longueur de la trame dans l'entête.
5. 800 en hexadécimal
6. Il est possible de paramétrer une interface Ethernet dans un mode spécial (*promiscuous* ou de *promiscuité*), indiquant à l'interface Ethernet d'envoyer *tous* les paquets à l'ordinateur. Ce mode est essentiellement utilisé par des logiciels d'analyse de trafic sur un réseau local, *tcpdump*, par exemple.

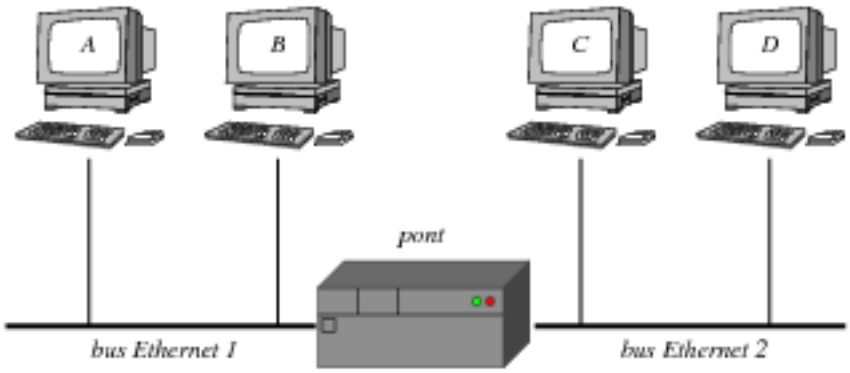


FIGURE 3.4 – Pont entre deux bus

les machines du réseau local, ne peut transmettre qu'une seule trame à la fois. Lorsque deux interfaces émettent une trame en même temps, une *collision* se produit. Le résultat est qu'aucune des deux trames ne pourra être reçue correctement.

Pour éviter ce problème, chaque interface écoute le câble pendant qu'elle émet une trame, afin de détecter une éventuelle collision. Si une collision est détectée par une interface, elle prévient l'ordinateur pour qu'il arrête d'émettre des données et attend un temps aléatoire compris entre 0 et une certaine durée δ avant de ré-émettre la trame. Si il y a de nouveau une collision, un nouveau temps d'attente est tiré au sort entre 0 et $2 \times \delta$, puis entre 0 et $4 \times \delta$, et ainsi de suite en doublant à chaque fois le délai maximal de ré-émission. Ce principe s'appelle *CSMA/CD* (*Carrier Sense Multiple Access with Collision Detect*) et permet de diminuer les risques de collisions : si une première collision se produit, il y a de fortes chances pour que les délais d'attente tirés au sort par les interfaces ayant émis les trames entrées en collision soient très proches, donc il sera probable qu'il y ait une nouvelle collision. En doublant à chaque fois l'intervalle des délais d'attente possibles, on augmente les chances de voir les ré-émissions s'étaler sur des durées plus longues et donc de diminuer les risques de collision.

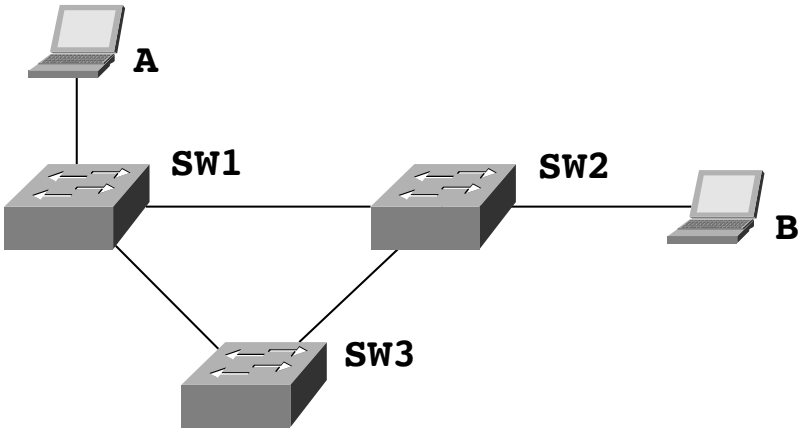
En pratique, un taux de collision de 5% (pour 100 trames émises sur le réseau, 5 ont donné lieu à une collision) est tout à fait normal pour un réseau local. *CSMA/CD* n'assure pas par contre un délai maximal de transmission pour une trame.

3.1.5 Equipements matériels

— Un *répéteur* retransmet et amplifie tous les signaux qu'il reçoit sans effectuer aucun autre traitement sur ces signaux. Il permet par exemple de constituer un bus de 300 mètres de long en interconnectant deux câbles Ethernet fin. Un *hub* est un répéteur multi-port, il envoie les signaux qu'il reçoit sur l'un de ses ports à tous ses autres ports.

— Un *pont* permet de relier des segments de réseau local. Il a des interfaces sur plusieurs réseaux locaux disjoints. Il filtre et oriente les trames en fonction de leurs adresses de destination et d'une table d'adresses représentant la cartographie du réseau. Sur la figure 3.4, une trame circulant de l'ordinateur A vers l'ordinateur B ne traversera pas le pont. Par contre, une trame de l'ordinateur A vers l'ordinateur C traversera le pont. L'intérêt du pont est d'améliorer la qualité du réseau ; sur l'exemple de la figure 3.4, la bande passante de 10 Mbits/s. du bus 1 est partagée par les machines A et B la bande passante de 10 Mbits/s. du bus 2 est partagée par les machines C et D. Sans pont, les quatres machines auraient partagé la bande passante du même bus.
Le pont permet également d'interconnecter des supports physiques de nature différente (par exemple Ethernet et RNIS).

— Un *commutateur* est un pont multi-ports. Il va aiguiller chacune des trames qu'il reçoit vers le segment (le bus) sur lequel se trouve l'ordinateur à destination de la trame. Ils sont aujourd'hui majoritairement utilisés en remplacement des hubs.



La machine **A** envoie une trame à la machine **B**. **SW1** ne connaissant pas le port associé à **B** envoie la trame vers **SW2** et **SW3**. **SW2** va renvoyer la trame vers **SW3** et vers la machine **B**. **SW3** renverra alors une trame vers **SW1**, entrainant alors une surcharge des liaisons. De plus, **SW2** ayant reçu une trame à la fois de **SW1** et de **SW3** ne pourra pas associer l'adresse de **A** ni au port sur lequel est branché **SW1** ni à celui sur lequel est branché **SW3**.

FIGURE 3.5 – Boucle de commutation sans *Spanning Tree* : cas 1

3.1.6 Spanning Tree

Lorsque plusieurs commutateurs sont interconnectés, il est possible que des *boucles de commutation* se forment. De telles boucles sont illustrées par les figures 3.5 et 3.6.

Le protocole *STP* (*Spanning Tree Protocol*), faisant partie de la norme IEEE 802.1d, permet de définir une topologie sans boucle tout en permettant une redondance de liaisons en cas de défaillance d'une liaison entre commutateurs. Le principe de ce protocole est de déterminer dynamiquement un arbre minimum couvrant l'ensemble des ports ethernet interconnectés.

Le fonctionnement de ce protocole est le suivant (voir figure 3.7) :

1. **élection d'un commutateur racine :** chaque commutateur reçoit un identifiant unique (*BID* ou *Bridge Identifier*), composé de l'adresse MAC du commutateur ainsi que d'un niveau de priorité paramétrable. La racine de l'arbre est déterminée comme étant le commutateur de priorité la plus basse, ou, en cas d'égalité, d'adresse MAC la plus basse.
2. **Élection d'un port racine sur chaque commutateur :** chaque port de chaque commutateur se voit attribuer un *coût*, correspondant à la capacité de la liaison branchée sur le port. Le port le plus proche du commutateur racine et ayant la priorité la plus faible est élu port racine.
3. **Détermination du chemin vers la racine :** pour chaque commutateur, le chemin le plus court vers la racine (somme des coûts de chaque liaison empruntée) est déterminé. Les autres chemins, mettant en œuvre des ports autres que les ports racines sont bloqués.

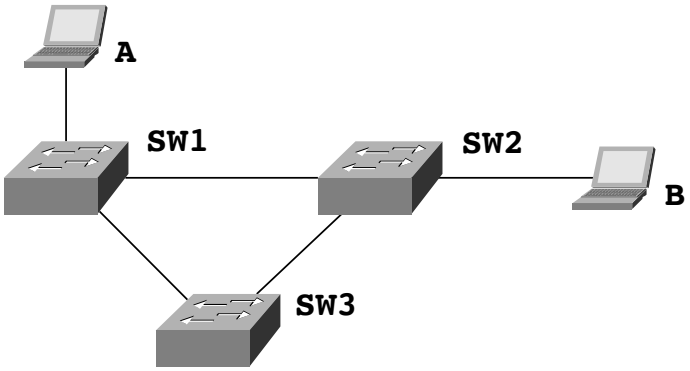
Les messages spécifiques permettant le fonctionnement de *STP* sont envoyées dans des trames spéciales *BPDU* (*Bridge Protocol Data Units*) envoyées régulièrement[7] par chaque commutateur sur l'adresse multicast `01:80:C2:00:00:00`.

Le protocole *RSTP* (*Rapid Spanning Tree Protocol*) est une évolution du protocole *STP*, permettant une convergence plus rapide.

3.1.7 VLANs

Les *VLANs* (*Virtual Local Area Network*) permettent de créer des réseaux locaux virtuels restreints à certains ports d'un réseau ethernet commuté. Ils sont définis par la norme IEEE 802.1q.

7. toutes les 2 secondes par défaut.



A envoie une trame à **B**. Chaque commutateur transmet la trame sur le segment sur lequel se trouve la machine **B**. Les deux commutateurs voient chacun une trame provenant de la machine **A** émise sur le segment de la machine **B** et conclue donc que la machine **A** se trouve sur ce segment.

FIGURE 3.6 – Boucle de commutation sans *Spanning Tree* : cas 2

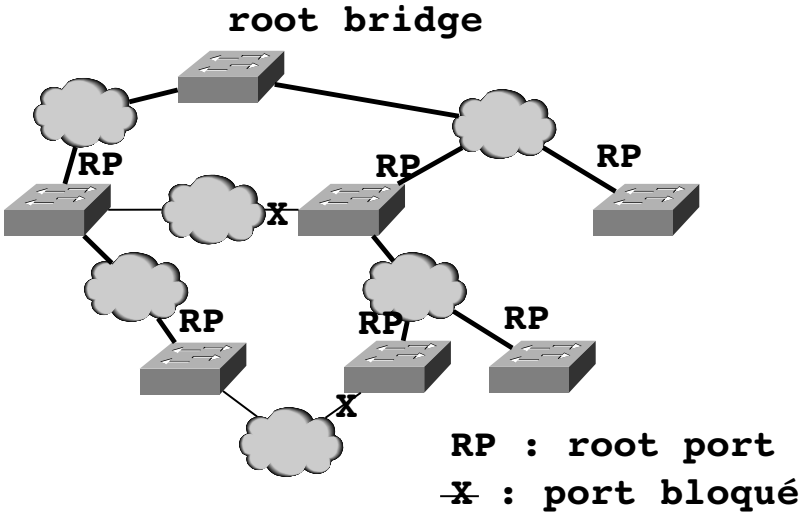


FIGURE 3.7 – Topologie de réseau commuté après convergence

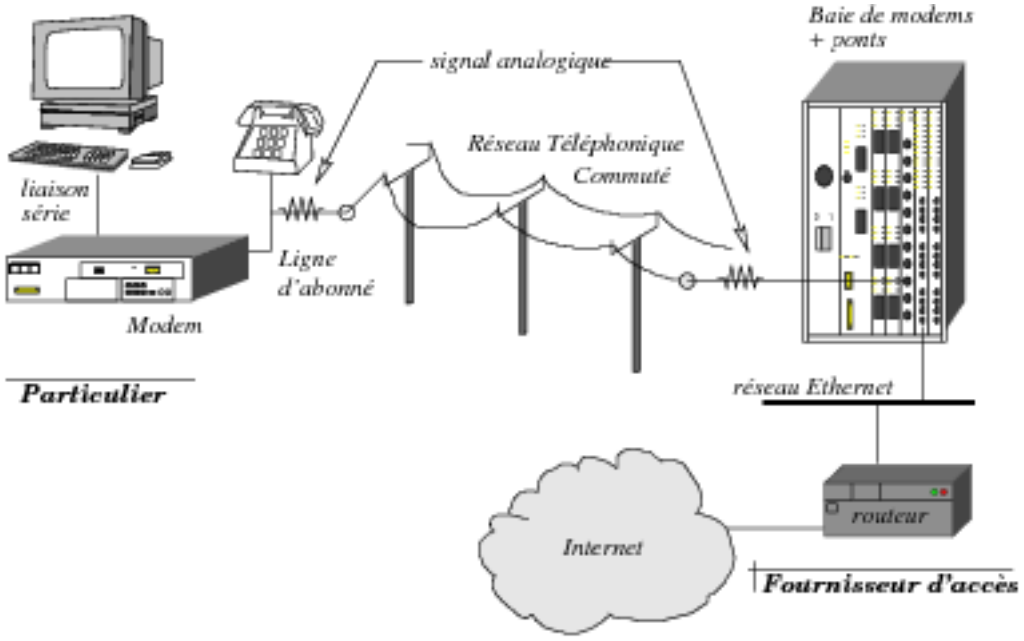


FIGURE 3.8 – Connexion à Internet avec un fournisseur d'accès

Leur objectif est double :
- réduire les diffusions (*broadcast*) à un ensemble de stations,
- isoler logiquement les stations selon un certain nombre de critères.

Les critères permettant de définir les VLANs au niveau des commutateurs sont les suivants :
- les ports de raccordement sur le commutateur,
- selon les adresses MAC des stations,
- par protocole de réseau ou par adresse de réseau (adresse IP typiquement).

Les informations d'appartenance à un VLAN sont associées à chaque trame ethernet sous la forme d'une étiquette (*tag*) comportant un numéro codé sur 12 bits [8]. Ces étiquettes ne sont généralement pas associées aux trames échangées entre les stations (machines non commutateurs) et les commutateurs [9]. En revanche, elles sont généralement associées aux trames échangées entre commutateurs [10], permettant ainsi d'interconnecter des commutateurs partageant plusieurs VLANs.

3.2 PPP

PPP (*Point-to-Point Protocol*) [RFC 1661] permet d'échanger des trames entre *deux* ordinateurs reliés par une liaison point à point, par exemple un ordinateur personnel équipé d'un modem relié à un modem d'un serveur d'un fournisseur d'accès par une liaison téléphonique illustré par la figure 3.8. PPP ne décrit pas la transformation des données en signal analogique, c'est le rôle du modem (les normes V.34 et V.90 de l'ITU-T décrivent cette opération), pas plus que le transfert physique entre l'ordinateur et le modem (décrit notamment par la norme V.24 de l'ITU-T, pour les liaisons séries asynchrones).

3.2.1 Établissement de la liaison

Le déroulement d'une connexion est le suivant :

8. 4096 VLANs peuvent donc coexister sur un même réseau commuté.
9. les trames sont dites *untagged*.
10. les trames sont alors dites *tagged* ; de telles interconnexions entre commutateurs sont désignées par *trunks* (faisceaux).

Balise (0x7e)	addr (0xff)	ctrl (0x03)	Protocole	Données	CRC
1 octet	1 octet	1 octet	2 octets	$\leq$ 1500 octets	2 octets

(les champs dont les noms sont soulignés sont définis par la norme *HDLC*))

FIGURE 3.9 – Trame PPP

1. Le modem de l'utilisateur (le client) appelle le numéro de téléphone du fournisseur d'accès Internet.
2. Une fois que les deux modems (celui du serveur (fournisseur d'accès) et celui du client (utilisateur)) sont connectés, le client s'identifie au serveur suivant une des trois manières suivantes :
   — *login* : Le serveur envoie un invité de connexion : la chaine de caractères `Login:` à laquelle le client répond en envoyant le nom de connexion de l'utilisateur au serveur. Ensuite, le serveur envoie un invité de saisie de mot de passe : la chaine `password :` à laquelle le client répond en envoyant le mot de passe de l'utilisateur. Le mot de passe est envoyé en clair (non crypté) sur la liaison téléphonique. Cette méthode, héritée de *UUCP*, ne fait pas partie du protocole PPP. Une fois le client identifié par cette méthode, la liaison PPP est établie sans autre identification de l'utilisateur.
   — *PAP* (*Password Authentication Protocol*) : l'identification se fait lors de l'établissement du lien par un sous protocole de PPP : *LCP* (*Link Control Protocol*). Le fonctionnement est toute fois similaire à celui de la méthode précédente : le serveur envoie au client un paquet spécial pour demander un nom d'utilisateur et un mot de passe. Puis, l'ordinateur envoie ces informations directement. Avec cette méthode, le mot de passe est également envoyé en clair.
   — *CHAP* (*Challenge Handshake Authentication Protocol*) fait également partie de *LCP* et fonctionne d'une façon similaire à *PAP*, mais le serveur envoie d'abord une clef qui va permettre de crypter l'envoi du nom de l'utilisateur et du mot de passe.
3. Une fois que l'utilisateur est identifié, LCP permet de négocier un certain nombre de paramètres entre le client et le serveur, notamment :
   — les adresses réseau (IP, par exemple) des deux machines.
   — Les protocoles de compression des entètes.

   Le terme *négocier* indique chacune des deux machines peut fixer certains paramètres ou bien accepter ceux fixés par l'autre machine. Dans le cas où les deux machines n'arrivent pas à se mettre d'accord (par exemple si les deux machines veulent fixer des adresses réseau différentes à l'une d'entre elles), le lien est rompu.

   Dans le cas de la connexion d'un utilisateur à un fournisseur d'accès Internet, la machine de l'utilisateur ne fournit généralement pas d'adresse réseau et accepte les adresses réseaux fournies par le fournisseur d'accès.
4. Une fois la liaison établie, les machines s'attribuent leurs adresses réseau qui seront valides le temps de la connexion.
5. Le serveur fait en général office de pont, de routeur ou de proxy et renvoie le trafic (les trames) provenant du client vers un autre réseau (Internet, par exemple).

3.2.2 Trame PPP

La figure 3.9 donne le format d'une trame PPP. Ce format de trame est celui défini par la norme *HDLC* (*High level Data Link Control*) de l'*ISO*. La norme *HDLC* dont la portée est plus générale que PPP (c'est par exemple la norme qui définit la couche liaison de X25 du CCITT, utilisé par le réseau *Transpac* de France Telecom, notamment) définit des liaisons point à point ou multipoint entre une machine primaire et n machines secondaires.

Balise de début et de fin de trame : Une trame PPP commence et se termine par un octet de valeur `0x7e`. La valeur `0x7e` est une constante, fixée par la norme HDLC.

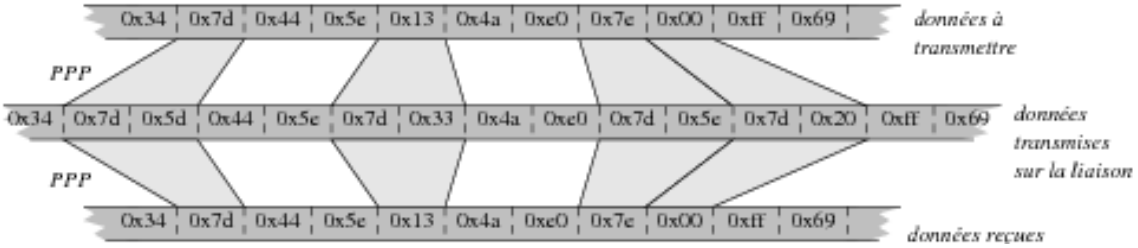


FIGURE 3.10 – Substitution des caractères de contrôle

Le champ d'adresse (`addr`) détermine l'adresse physique de la machine secondaire à laquelle est destinée la trame (définition de la norme HDLC). Cependant, la notion d'adresse physique, telle qu'elle existe pour Ethernet, par exemple, n'a pas de raison d'exister pour PPP, puisqu'il n'y a qu'une seule machine à chaque extrémité du lien. Pour cette raison, et spécifiquement à PPP (par rapport à la norme HDLC), la valeur d'adresse est fixée à une valeur constante : `0xff`.

Le champ de contrôle (`ctrl`) détermine le type de la trame (données, contrôle de la liaison, ...) pour HDLC. Dans le cas particulier (par rapport à HDLC) de PPP, le type de la trame est déterminé par le champ `protocole`. Le champ `ctrl` est lui fixé à la valeur constante `0x03`.

Le champ de protocole est spécifique à PPP (pour HDLC, ce sont simplement les deux premiers octets de données). Il permet de déterminer si la trame est une trame de contrôle de liaison (pour le protocole LCP) ou une trame de données (contenant un paquet IP, par exemple). Il a la valeur `0x0021` dans le cas d'une trame de données et `0xc021` dans le cas d'une trame de contrôle de liaison.

Le champ CRC contient une somme de contrôle permettant de détecter les erreurs de transmission. Les trames erronées sont retransmises.

Les trames sont envoyées les unes à la suite des autres sur le lien physique. La longueur de la trame étant variable (le champ de données ayant lui même une longueur variable) et non codée dans la trame, la balise `0x7e` permet de découper les trames. Pour résoudre le problème de l'apparition d'un octet valant `0x7e` au milieu des données (qui pourrait être interprété comme une fin de trame), les substitutions suivantes (dans cet ordre sur les octets de la trame) ont lieu avant d'émettre une trame et les substitutions inverses ont lieu lors de la réception de la trame :

1. un octet de valeur `0x7d` est remplacé par la séquence `0x7d 0x5d`,

   puis

2. un octet de valeur `0x7e` dans les données est remplacé par la séquence `0x7d 0x5e`

   puis

3. un octet x de valeur $x <$`0x20` est remplacé par la séquence `0x7d` ($x +$ `0x20`).

Ainsi, quelque soient les données transmises, l'octet `0x7e` n'apparaitra jamais après substitution. La deuxième substitution permet d'interpréter la présence d'un octet `0x7d` dans les données comme étant soit l'octet `0x7d` lui-même (dans ce cas, l'octet `0x7d` est suivi de l'octet `0x5d`), soit l'octet `0x7e` (dans ce cas, l'octet `0x7d` est suivi de l'octet `0x5e`), soit un octet inférieur à `0x20`. La figure 3.10 illustre ces substitutions.

Les octets de valeur $<$ `0x20` peuvent être interprétés par des périphériques de transmission de données (modems, par exemple) comme des caractères de contrôle. Par exemple, les octets de valeur `0x13` (`XOFF`) et `0x11` (`XON`) sont utilisés par certains de ces périphériques pour contrôler le flux de données (contrôle de flux *xon/xoff*). La troisième substitution (portant sur des octets $<$ `0x20`) permet de ne pas générer sur la liaison des octets de ce type.

Le protocole PPP est souvent utilisé pour réaliser des liaisons point-à-point exploitant un autre réseau pour l'acheminement des données. Ainsi *PPPoE* (*PPP over Ethernet*) permet de construire

des liaisons point-à-point à partir d'une liaison ethernet et *PPPoA* (*PPP over ATM*) permet de construire une liaison PPP en se basant sur un réseau ATM. Ces deux protocoles dits d'*encapsulation* sont principalement utilisés dans le cadre de connexions ADSL ou SDSL.

3.3 ARP

Les protocoles *ARP* (*Address Resolution Protocol*) et *RARP* (*Reverse Address Resolution Protocol*) permettent d'établir une correspondance entre une adresse IP et une adresse physique d'une interface d'un réseau local (adresse physique d'une interface Ethernet, par exemple). *ARP* permet d'obtenir une adresse physique à partir d'une adresse IP et *RARP* permet d'obtenir une adresse IP à partir d'une adresse physique.

3.3.1 Déroulement d'une requête ARP

Lorsque le protocole IP est utilisé sur un réseau local, ARP est le seul moyen d'adresser les machines de ce réseau. Le déroulement d'une requête ARP est le suivant :

1. La machine *A* a un paquet IP a envoyer à la machine *B*, avec une adresse IP destination sur le réseau local (les autres cas, impliquant le routage du paquet, seront examinés dans le chapitre suivant). L'adresse IP de la machine *B* est connue, mais pas son adresse physique, nécessaire pour délivrer le paquet à la machine *B*. C'est cette adresse IP qui doit être résolue par ARP en l'adresse physique de la machine *B*.
2. La machine *A* envoie une requête ARP sur le réseau local avec comme adresse physique de destination, l'adresse de *broadcast* (trame à destination de toutes les machines du réseau local ; dans le cas d'Ethernet, l'adresse de broadcast est `ff:ff:ff:ff:ff:ff`), l'adresse physique de destination (l'adresse physique de la machine *B*), n'étant pas encore connue. La requête ARP contient l'adresse IP à résoudre et l'adresse physique de la machine émettrice.
3. Toutes les machines du réseau local reçoivent la requête ARP. La machine ayant comme adresse IP l'adresse IP contenue dans la requête ARP (c'est à dire la machine *B*) renvoie une trame contenant la réponse à la requête à destination de l'adresse physique de l'émetteur (donc de la machine *A*), contenue dans la requête ARP. La réponse contient, dans le champ de données de la trame, l'adresse physique de la machine ayant répondu à la requête (la machine *B*).
4. La machine *A* reçoit la réponse à la requête. Elle peut alors faire l'association entre l'adresse IP et l'adresse physique de la machine destinataire du paquet IP.
5. La machine *A* peut alors envoyer son paquet IP sur le réseau local en l'encapsulant dans une trame ; l'adresse physique de destination de la trame est connue.

Sur le schéma de la figure 3.11, quatre machines sont connectées sur un réseau local Ethernet. La machine d'adresse IP `193.52.86.3` effectue une requête contenant son adresse IP, sa propre adresse Ethernet et l'adresse IP à résoudre (trame `REQ`). L'adresse Ethernet n'est pas connue. La requête est envoyée à toutes les machines du réseau local (*broadcast*). La machine `193.52.86.6`, ayant comme adresse IP l'adresse de destination de la requête ARP, renvoie une réponse à la requête contenant sa propre adresse IP vers la machine ayant pour adresse Ethernet `00:50:04:B5:FE:CA` (adresse source dans la requête ARP).

3.3.2 Format d'une requête ARP

La figure 3.12 décrit le format d'une requête ou d'une réponse ARP (les deux types de message sont identiques), décomposé selon les champs suivants :

Le type de matériel désigne le type physique de support employé pour le réseau local. Ce champ prend la valeur `0x0001` pour Ethernet (il vaut `0x0004` pour token-ring ou `0x0008` pour *AppleTalk*).

Le protocole réseau vaut `0x0800` pour indiquer le protocole réseau IP.

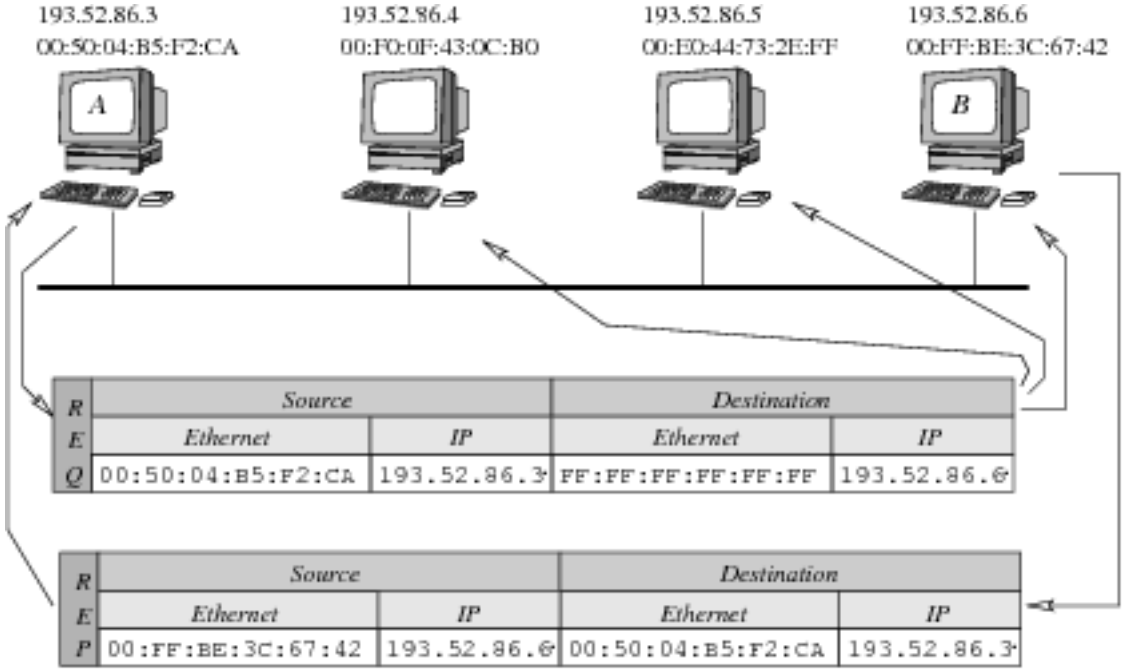


FIGURE 3.11 – Déroulement d'une requête ARP

Type matériel	protocole réseau	taille des adresses physiques	taille des adresses réseau	commande	arguments
(2 octets)	(2 octets)	(1 octet)	(1 octet)	(2 octets)	(taille variable)

FIGURE 3.12 – Requête ou réponse ARP

La taille des adresses physiques représente le nombre d'octets d'une adresse physique correspondant au type de matériel décrit par le champ type de matériel. Dans le cas d'un réseau Ethernet, les adresses physiques sont codées sur 6 octets, ce champ vaudra donc `0x06`.

La taille des adresses réseau représente le nombre d'octets d'une adresse réseau correspondant au protocole réseau décrit par le champ protocole réseau. Les adresses IP sont codées sur 4 bits, ce champ vaudra donc `0x04`.

La commande correspond au type de message ARP :

Type de message ARP	Valeur du champ
Requète ARP	0x0001
Réponse ARP	0x0002
Requète RARP	0x0003
Réponse RARP	0x0004

Le champ arguments correspond aux arguments ou à la réponse d'une requête ARP. Ce champ a une longueur variable dépendant de la taille des adresses physique et réseau. Dans le cas du réseau IP sur Ethernet, cette longueur est de 20 octets.

Dans le cas du réseau IP sur Ethernet, le format de l'argument est donné par la figure 3.13. Ce format de requête ou de réponse peut être interprété comme dans l'exemple de la figure 3.11. Le type de trame Ethernet (champ type de la figure 3.3) vaut `0x0806` pour une requête ARP.

Pour limiter la multiplication de requêtes ARP sur le réseau local, ARP tient à jour une table (la *table ARP*) qui mémorise les correspondances résolue pendant une certaine durée. ARP ne lance une requête sur le réseau local que lorsqu'il ne trouve pas l'adresse physique dans la table. Les entrées dans la table ARP ont une durée de vie limitée pour permettre un changement de

Source		**Cible**	
adresse Ethernet (6 octets)	adresse IP (4 octets)	adresse Ethernet (6 octets)	adresse IP (4 octets)

FIGURE 3.13 – Argument d'une requête ou d'une réponse ARP ou RARP sur un réseau Ethernet

configuration du réseau local (changement d'adresse IP ou d'adresse Ethernet sur une machine, par exemple).

Les protocoles de transport

Les protocoles de transport permettent de transférer les données pour les applications. Le transfert de données est transparent pour les applications ; par exemple, deux machines pourront échanger des données pour une application de la même façon, que ces deux machines soient sur le même réseau local ou sur deux continents différents.

Les protocoles de transport assurent notamment le découpage des données des applications en paquets IP pour leur acheminement sur le réseau, ainsi que la reconstitution des données transférées, impliquant éventuellement des demandes de réexpédition de paquets perdus.

Les deux principaux protocoles de transport de la famille TCP/IP sont TCP (*Transfer Control Protocol*) et UDP (*User Datagram Protocol*). Ils utilisent tous les deux IP comme couche réseau, mais TCP procure un service de flux séquentiel de caractères orienté connexion et fiable alors qu'UDP ne fait que transporter de manière non fiable des messages (courts) d'applications.

4.1 UDP

UDP (*User Datagram Protocol*, [RFC 768]) permet aux applications d'utiliser presque directement les services fournis par IP, c'est à dire l'acheminement, sans connexion, de petites quantités de données.

4.1.1 Ports

UDP permet de distinguer plusieurs applications sur une même machine (le *DNS* et *X-Window*, par exemple) en représentant ces applications par des numéros de *port* ; le numéro de port donne ainsi l'abstraction d'avoir plusieurs interfaces de transport de données simultanément sur la même machine.

Le numéro de port peut varier entre 0 et 65535 (codage sur 2 octets) et certains numéros sont réservés pour certaines applications [1] (SMTP a le numéro 25, par exemple).

Certains numéros de port sont vacants (ne sont pas affectés aux applications) et sont utilisés pour différentier les messages provenant d'une même machine et à destination d'une même machine, mais représentant des instances différentes d'une même application (par exemple deux requêtes DNS, émises simultanément d'une même machine vers une autre machine).

4.1.2 Format d'un paquet UDP

La figure 4.1 représente le format d'un datagramme IP (hors entête IP).

Les ports source et destination indiquent l'instance (processus) d'application récepteur du message (port destination) et l'instance d'application émettrice du message (port source).

1. Sous *Unix*, le fichier `/etc/services` définit ces affectations.

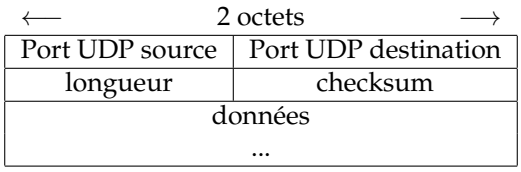


FIGURE 4.1 – Format d'un datagramme UDP

Le champ longueur représente la longueur, sur 2 octets, du datagramme (entête UDP + données), exprimée en octets. La longueur minimale est 8 octets (un paquet UDP peut ne pas contenir de données). La longueur maximale est 655515 octets (la longueur maximale d'un paquet IP est de 65535 octets et l'entête IP a une longueur minimale de 20 octets).

Le checksum est une somme de contrôle optionnelle. Si la somme de contrôle n'est pas calculée, le champ checksum est initialisé à la valeur `0x0000`. Le checksum est calculé de la même façon que pour le champ checksum de l'entête IP. Dans le cas ou le champ checksum est calculé et vaut `0x0000`, le champ est mis à `0xffff` ; ceci afin d'éviter avec la valeur `0x0000` indiquant que le checksum n'a pas été calculé. Le checksum est recalculé lors de la réception du datagramme UDP et comparé avec la valeur contenue dans le champ checksum (dans le cas où il est différent de `0x0000`). Si les checksums ne correspondent pas, le paquet est détruit.

4.2 TCP

Le protocole TCP [RFC 793] permet de simuler des connexions sur le réseau. Les applications dialoguant à travers TCP sont considérée l'une comme un *serveur*, l'autre comme un *client* [2]. Ces deux machines doivent établir une connexion avant de dialoguer. Les deux machines, lors de l'établissement de la connexion vérifient qu'elles sont autorisées à dialoguer et qu'elles sont prêtes pour le transfert de données en s'échangeant des messages spécifiques (négociation de la connexion).

Une fois que ces négociations sont terminées, la connexion est établie et les applications sont informées qu'elles peuvent commencer à envoyer ou commencer à attendre des données sur la connexion. Il y a donc deux et exactement deux machines communiquant l'une avec l'autre à un instant donné. La connexion est bidirectionnelle simultanée (*full duplex*) et composée de deux flux de données indépendants, séquentiels et asynchrones, en sens contraire.

Tout au long de la connexion, TCP échange un flux d'octets sans qu'il soit possible de séparer par une marque quelconque certaines données. Si elles sont trop nombreuses pour être transmises sur le réseau (pour être encapsulées dans des paquets IP), les données sont fragmentés en *segments*. A l'inverse, les *segments* peuvent être réassemblés en des *segments* plus gros.

4.2.1 Accusés de réception

TCP est fiable, alors qu'il repose sur IP qui ne l'est pas. Cette fiabilité est rendue possible en incluant des accusés de réception aux segments de données transmises.

Sur l'exemple de la figure 4.2, un segment est émis avec un numéro qui va servir au récepteur pour envoyer un accusé de réception. Ainsi, l'émetteur sait si le segment qu'il a transmis a été reçu correctement. A chaque envoi de segment, l'émetteur arme une temporisation qui lui sert de délai d'attente de l'accusé de réception correspondant à ce segment. Lorsque le délai d'attente expire sans qu'il ait reçu d'acquittement (ACK, pour *acknowledgement*) pour le segment, l'émetteur considère que le segment a été perdu et il le réexpédie. Il se peut que la temporisation expire alors que le segment a été reçu par le destinataire (accusé de réception perdu, par exemple). Dans ce cas, l'émetteur réexpédiera un segment alors que c'est inutile. Pour éviter d'avoir des segments

2. Modèle client-serveur.

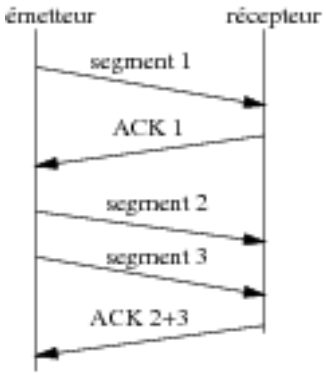


FIGURE 4.2 – Transmission d'un segment

en double, le récepteur garde la trace des numéros de segments reçus et peut donc éliminer ainsi éliminer les segments en trop.

4.2.2 Format d'un segment TCP

La figure 4.3 donne le format d'un segment TCP. Le même format est utilisé pour établir une connexion, transférer des données ou libérer une connexion.

Les différents champs du segment présenté figure 4.3 ont la signification suivante :

- *port source* et *port destination* ont le même rôle que pour le protocole UDP.
- Le *numéro de séquence* est un numéro attribué au segment par l'émetteur du paquet,
- le *numéro d'accusé de réception* est renvoyé par le destinataire d'un segment à son expéditeur, et correspond au numéro de séquence d'un segment reçu.
- La *taille de l'entête* (*doff*) représente la taille de l'entête TCP, exprimée en multiples de 32 bits.
- Les drapeaux sont des indicateurs booléens ayant la signification suivante :
  - *URG* : indique des données urgentes ou dites transmises hors-bande. Ces données sont alors identifiées dans le segment TCP au moyen du champ *pointeur d'urgence*.
  - *ACK* : indique que le paquet contient un accusé de réception. Ce drapeau est normalement mis à 1 pour tout paquet TCP échangé lors d'une même connexion TCP, excepté pour le premier paquet échangé, ne correspondant pas à un acquittement de données précédemment reçues.
  - *PSH* : indique que les données doivent être envoyées tout de suite.
  - *RST* : indique une rupture de connexion anormale, à l'initiative de l'un des deux nœuds participant à la connexion.
  - *SYN* : indique une demande de connexion ou une demande de synchronisation d'une connexion. Ce drapeau est normalement mis à 1 sur le premier paquet envoyé par le nœud à l'initiative d'une connexion TCP vers le second nœud, ainsi que sur le paquet renvoyé par ce second nœud pour indiquer qu'il accepte la connexion.
  - *FIN* : indique une demande de fermeture de connexion ; à l'initiative d'un des deux nœuds participants.
- La *taille de fenêtre* est définie par un récepteur de segment pour indiquer la taille maximale (en octets) de données qu'il peut accepter avant de renvoyer un acquittement. Ce paramètre est notamment utilisé par TCP pour permettre le contrôle de flux.
- La *somme de contrôle* (*checksum*) est une somme de contrôle, calculée par l'émetteur sur le segment émis. Cette somme de contrôle est le complément à 1 de la somme complémentée à 1 des entêtes ainsi que des données.
- Le *pointeur d'urgence* représente la position relative des données urgentes dans le segment, lorsque le drapeau *URG* est mis à 1.

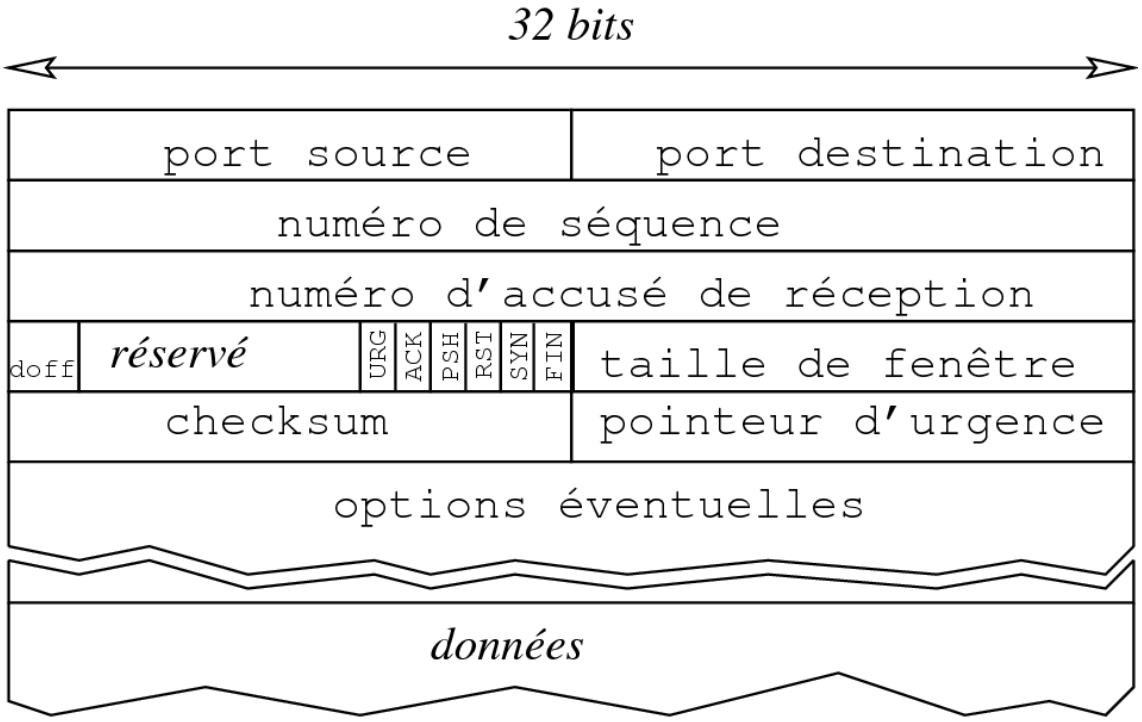


FIGURE 4.3 – Format d'un segment TCP

4.2.3 Négociation (*3 ways handshake*)

La négociation d'une connexion TCP se fait en trois temps :

1. Le nœud à l'origine de la demande de connexion (*client*) émet un message TCP avec un drapeau *SYN* levé et un drapeau *ACK* baissé.
2. Le nœud recevant la demande de connexion (*serveur*) répond alors avec un message TCP dans lequel le drapeau *SYN* est toujours levé, ainsi que le drapeau *ACK*.
3. Les paquets suivants sont émis avec les drapeaux *SYN* baissé et *ACK* levé.

4.2.4 Contrôle de flux

Chaque segment TCP contient la taille disponible dans le tampon de réception de l'hôte qui l'a envoyé (*taille de fenêtre*). Le nœud émetteur d'un flux va pouvoir exploiter ce paramètre afin d'envoyer des quantités de données correspondant à cette taille de tampon.

TCP utilise un mécanisme de retransmission temporisée de segments perdus afin d'assurer un transport fiable. Après l'envoi d'un segment, TCP va attendre un certain temps la réception de l'acquittement correspondant. Ce délai est ajusté car un délai trop court entraînerait un grand nombre de retransmissions inutiles et un délai trop long ralentirait la réaction en cas de perte d'un segment.

Le délai avant retransmission doit être supérieur au RTT [3] moyen d'un segment. Cette valeur pouvant varier dans le temps, TCP réalise des mesures (échantillons) de RTT à intervalle régulier et en calcule une moyenne pondérée :

*RTT moyen = (1-a) * RTT moyen + a * RTT échantillon*

3. *Return Trip Time*, le délai d'aller et retour d'un paquet entre les 2 nœuds participant à la connexion TCP.

Une valeur typique pour a est 0.125 [4]. L'influence des échantillons diminue de manière exponentielle dans le temps.

Le délai à utiliser est obtenu à partir de cette estimation du RTT moyen et en y ajoutant une marge de sécurité. Plus la différence entre un échantillon et la moyenne est grande, plus la marge de sécurité à prévoir est importante. Le calcul se fait à partir de la variance pondérée entre l'échantillon et la moyenne :

$$Variance\ RTT = (1\text{-}b) * Variance\ RTT + b * |RTT\ échantillon - RTT\ moyen|$$

Une valeur typique pour b est 0.25. Le délai à utiliser est finalement donné par la formule suivante :

$$Délai = RTT\ moyen + 4 * Variance\ RTT$$

Parfois, quand le délai est trop long, il est avantageux de ne pas attendre avant de retransmettre un segment. Si un hôte reçoit 3 acquittements du même segment, alors il considère que tous les segments transmis après le segment acquitté ont été perdus et les retransmet donc immédiatement (*Fast retransmit*).

4. Une valeur proche de 1 va permettre de prendre en compte plus rapidement des changements de capacités de l'infrastructure réseau utilisée, au prix d'un risque de surestimer ceux-ci ; une valeur proche de 0 va permettre un comportement plus constant de TCP lorsque l'infrastructure réseau est perturbée, au risque de minimiser des perturbations temporaires.

Autoconfiguration et nommage

5.1 DHCP

5.1.1 Principe

Le protocole *DHCP* (*Dynamic Host Configuration Protocol*) permet l'autoconfiguration d'une machine interconnectée à un réseau. Ce protocole propose de déterminer automatiquement les paramètres suivants :

- Son adresse IP, qui doit correspondre au même préfixe de réseau logique que les adresses des autres machines. Cette adresse doit de plus être unique dans le préfixe.
- Le masque associé au préfixe. Ce masque permet à la machine de différencier les adresses appartenant au préfixe pour lesquelles les paquets seront remis directement des adresses extérieures au préfixe pour lesquelles il faudra passer par un routeur.
- L'adresse du routeur par défaut, utilisé justement pour les adresses hors du préfixe de réseau logique. Sans cette information, on ne pourra communiquer qu'avec les machines du réseau logique correspondant au préfixe.
- L'adresse d'un serveur DNS qui permettra de traduire un nom de machine en adresse IP. Sans cette information, il faut connaître l'adresse des machines plutôt que leur nom, plus facile à retenir.

Ces informations peuvent être fournies par l'administrateur réseau qui délivre des adresses à la demande pour les machines répertoriées, elles ne sont valides que dans le réseau d'origine. Il faut réactualiser ces paramètres à chaque connexion sur un nouveau réseau. Le rôle du protocole DHCP est de distribuer ces informations automatiquement aux machines qui se connectent sur le réseau, cela évite d'avoir à les configurer manuellement (ce qui peut être fastidieux surtout lors d'un changement de plan d'adressage du réseau) et permet éventuellement à des machines inconnues d'intégrer le réseau.

Une machine qui n'est pas configurée connaît néanmoins sa propre adresse de liaison et l'adresse de diffusion (*broadcast*) au niveau de sa liaison.

L'autoconfiguration des nœuds d'un réseau par DHCP est faite *avec état*, c'est-à-dire, que la configuration d'un nœud dépend de la configuration de nœuds antérieurement configurés ou de configurations antérieures du même nœud.

5.1.2 Messages DHCP

L'autoconfiguration d'un nœud en protocole DHCP correspond aux échanges de messages suivants :

1. `DHCPDISCOVER` : le client DHCP émet une requête en diffusion (adresse ethernet MAC `FF:FF:FF:FF:FF:FF` par exemple) en divulgant sa propre adresse de liaison. Il utilise temporairement l'adresse IP 0.0.0.0.

2. `DHCPOFFER` : chaque serveur DHCP sur le réseau local répond par une proposition de bail. Le message `DHCPOFFER` envoyé (en diffusion lorsque l'adresse IP du destinataire n'est pas connue) contient l'adresse IP du serveur.
3. `DHCPREQUEST` : le client répond en diffusion et indique quelle offre il accepte (normalement la première qu'il a reçue, mais il se peut aussi qu'un client cherche à renouveller un bail qu'il a déjà obtenu lors d'une configuration précédente).
4. `DHCPACK` : le serveur concerné envoie une confirmation de bail par ce message.

En plus de l'adresse IP pour le client associée à une durée de validité, le bail peut contenir :
— l'adresse d'un (ou plusieurs) serveur de nom,
— l'adresse du routeur par défaut,
— l'adresse du serveur DHCP.

5.1.3 Renouvellement du bail

— à 50% du temps de validité du bail, le client essaie de renouveler le bail,
— à 87.5% du temps de validité du bail, le client essaie d'étendre son bail avec un nouveau serveur DHCP,
— à l'expiration du bail, le client cesse d'utiliser l'adresse allouée dans le bail et relance le processus complet d'autoconfiguration.

5.1.4 Exemple de configuration

```
ddns-update-style ad-hoc;
shared-network cpl {
    subnet 192.168.0.0 netmask 255.255.255.128 {
        option subnet-mask 255.255.255.128;
        option domain-name "";
        option routers 192.168.0.127;
        option domain-name-servers 192.168.0.254;
        range dynamic-bootp 192.168.0.1;
        default-lease-time 3600;
        max-lease-time 7200;
    }
    subnet 192.168.0.128 netmask 255.255.255.128 {
        option subnet-mask 255.255.255.128;
        option domain-name "";
        option routers 192.168.0.254;
        option domain-name-servers 192.168.0.254;
        range dynamic-bootp 192.168.0.129;
        default-lease-time 3600;
        max-lease-time 7200;
    }
}
```

5.2 DNS

Le réseau IP fonctionne entièrement et de manière autonome en désignant les noeuds par des adresses IP. Pour des raisons de confort d'utilisation, les applications et les utilisateurs préfèrent toutefois utiliser des noms de machines. Ces noms sont organisés hiérarchiquement dans un annuaire distribué sur l'Internet : le DNS (*Domain Name System*). Le DNS a deux fonctions principales :
— déterminer une adresse IP numérique à partir d'un nom de machine,
— inversément, déterminer le nom correspondant à une adresse IP.

5.2.1 Principes

Le DNS fonctionne selon les principes suivants :
— le système de nommage est cohérent et hiérarchique pour désigner les ressources qui peuvent être de natures différentes (adresses, serveurs de courrier électronique...) ;
— un système distribué au niveau global de l'Internet, où chaque serveur ne connaît qu'une portion de l'espace sur lequel il a autorité par délégation ;
— une redondance de serveurs permet d'assurer la disponibilité et la continuité du service ;
— le temps de lecture (une résolution d'une adresse par exemple) est plus critique que le temps de modification et de synchronisation de la base distribuée ;
— l'utilisation de mémoire cache permet d'améliorer les performances.

5.2.2 Composants

Le service de noms de domaines comprend trois principaux types de composants :
— les *zones*, correspondant à des espaces de nommages ou *domaines*, ceux-ci contenant une liste d'enregistrements (*RR*, *Resource Records*),
— le service distant de nommage ou serveur de noms [1], permettant l'accès à la base de données des noms à travers l'accès aux zones,
— le service local de résolution de noms (*resolver*), intégré aux bibliothèques de fonctions (la *GNU libc* par exemple), permettant d'interroger des serveurs de noms distants [2].

5.2.3 Organisation hierarchique des espaces de nommage

L'espace de nom du DNS est hiérarchique avec une racine unique, non nommée. Chaque nœud est désigné par un label, les labels sont séparés par le caractère '.'. Le nom de domaine est hiérarchique de droite à gauche. Par exemple, dans le nom `www.w3.org`, le label `org` désigne le domaine de niveau supérieur (*TLD*, *Top Level Domain*), `w3` en est un domaine et `www` est un sous-domaine de `w3.org`.

Domaines de niveau supérieur (TLD)

Les domaines de premier niveau (TLD) sont soit :
— des domaines génériques ou organisationnels (*gTLD*, *Generic Top Level Domain*), tels que `com`, `edu`, `gov`, `int`, `mil`, `net`, `org`,
— des domaines géographiques (*ccTLD*, *country code Top Level Domain*), tels que `fr`, `eu`, `au`, `us`,
— des domaines d'infrastructure : le seul dans cette catégorie est le domaine `arpa`.

Chacun de ces domaines est divisé en sous-domaines, par exemple `gouv.fr` ou `univ-nantes.fr` (voir figure 5.1).

Nom de domaine absolu (*FQDN, Fully Qualified Domain Name*)

Un nom de domaine est dit absolu s'il se termine par un point, par exemple `gouv.fr.` , sinon, c'est un nom relatif. Lorsqu'une requête est faite sur un nom relatif, le résolveur essaie dans un premier temps d'y ajouter le domaine par défaut, en cas d'échec, il essaie un domaine nul, par exemple :

```
$ host -v www
Trying "www.polytech.univ-nantes.fr"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23154
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
```

1. le logiciel serveur de noms le plus répandu est *BIND* (*Berkeley Internet Name Domaine server*).
2. la plupart des *resolvers* permettent une résolution vers des bases multiples, locales (fichiers `/etc/hosts`, pour un site (annuaires LDAP par exemple) ou distantes (DNS).

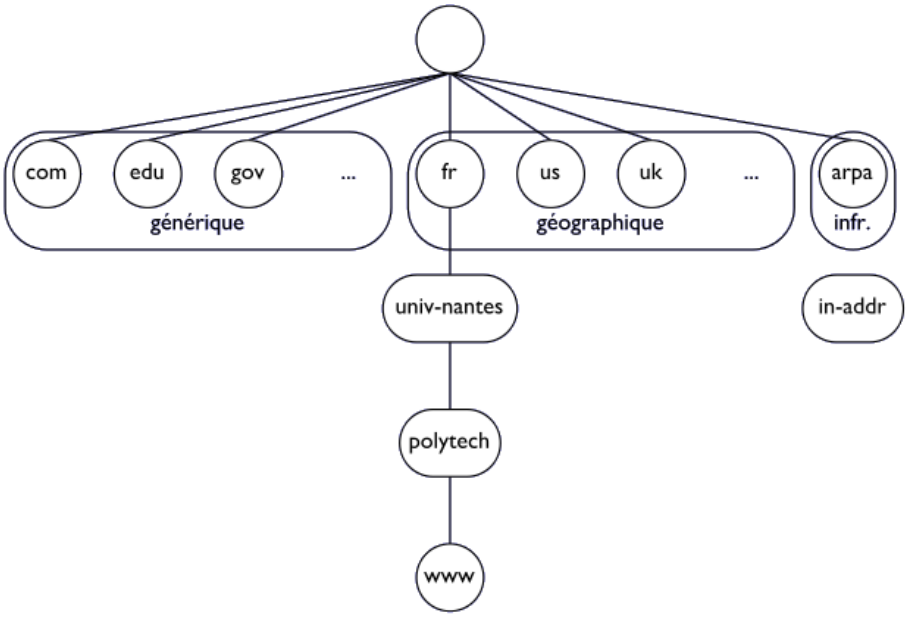


FIGURE 5.1 – Domaines de premier niveau

```
;www.polytech.univ-nantes.fr. IN A

;; ANSWER SECTION:
www.polytech.univ-nantes.fr. 120 IN CNAME revaccess3.univ-nantes.fr.
revaccess3.univ-nantes.fr. 345600 IN CNAME revaccess3.d101.univ-nantes.fr.
revaccess3.d101.univ-nantes.fr. 345600 IN A 193.52.101.171

Received 116 bytes from 10.44.34.1#53 in 75 ms
Trying "revaccess3.d101.univ-nantes.fr"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57141
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
;revaccess3.d101.univ-nantes.fr. IN AAAA

;; AUTHORITY SECTION:
univ-nantes.fr. 928 IN SOA dns1.univ-nantes.fr. dnsmaster.univ-nantes.fr. 2009101501 1

Received 99 bytes from 10.44.34.1#53 in 59 ms
Trying "revaccess3.d101.univ-nantes.fr"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38935
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;revaccess3.d101.univ-nantes.fr. IN MX

;; AUTHORITY SECTION:
univ-nantes.fr. 86400 IN SOA dns1.univ-nantes.fr. dnsmaster.univ-nantes.fr. 200910150

Received 99 bytes from 10.44.34.1#53 in 68 ms

$ host -v inexistant
Trying "inexistant.polytech.univ-nantes.fr"
```

Rémi Lehn

```
Trying "inexistant"
Host inexistant not found: 3(NXDOMAIN)
Received 28 bytes from 10.44.34.1#53 in 1 ms
```

5.2.4 Découpage en zones

Zone administrative

L'espace de nommage est divisé en zones administrées séparément qui peuvent à leur tour être divisées. L'autorité de ces zones est déléguée.

Serveur primaire

Pour chaque zone, le serveur de nom primaire enregistre les informations des machines sur lesquelles il a autorité. Un serveur primaire a la responsabilité du stockage permanent (dans des fichiers ou des bases de données) des enregistrements concernant la zone pour laquelle il est serveur primaire.

Serveur secondaires

Les serveurs secondaires sont indépendants (machines différentes) et redondants. Contrairement aux serveur primaires, ils ne stockent pas les enregistrements concernant les zones sur lesquelles ils ont autorité, mais obtiennent les informations périodiquement (dans un intervalle de quelques heures en général) à partir du serveur primaire par transfert de zone.

Les serveurs secondaires, tout comme les serveurs primaires sont considérés comme ayant *autorité* sur les zones qu'ils renseignent, c'est-à-dire qu'ils fournissent une information considérée comme valide et unique.

Serveurs racines

Lorsqu'un serveur DNS ne sait pas répondre à une requête, il doit interroger un autre serveur. Pour déterminer quel est ce serveur, il va préalablement interroger un serveur racine (il en existe actuellement 13) qui connaît tous les serveurs primaires des zones de second niveau.

Mémoire cache

Lorsqu'un serveur reçoit une information, il la conserve en mémoire cache pour d'éventuelles requêtes identiques. Certains serveurs de noms n'ont qu'une fonction de relai et de cache (*forward only*) et n'ont autorité sur aucune zone : ces serveurs peuvent répondre à des requêtes de résolution de noms, mais pour résoudre ces noms, ils doivent interroger d'autres serveurs. Ils peuvent mettre en cache les informations obtenues.

5.2.5 Enregistrements de ressource (*RR*)

Il existe différents types d'enregistrements renseignant différents types de noms. Chaque zone est renseignée par un nom, suivi par une durée de validité (exprimée en secondes) de l'enregistrement à partir de son obtention, puis du type de réseau sur lequel porte le nommage (classe : `IN` désigne Internet), puis le type d'enregistrement, puis les données de l'enregistrement. Par exemple, l'enregistrement

```
debian.univ-nantes.fr. 500 IN AAAA 2001:660:7220:381::deb1
```

associe :

1. le nom `debian.univ-nantes.fr.`, qui est un FQDN,
2. un enregistrement qui a une durée de vie de 500 secondes,

3. dans l'espace de nommage de l'Internet (`IN`),
4. associé à une adresse IPv6 (enregistrement de type `AAAA`),
5. qui est `2001:660:7220:381::deb1`.

Enregistrements A

Les enregistrements `A` renseignent les associations d'un nom vers une adresse IP (IPv4). Par exemple :

```
mx1.univ-nantes.fr. 344841 IN A 193.52.101.135
```

définit que le nom `mx1.univ-nantes.fr.` correspond à l'adresse IP `193.52.101.135`.

Enregistrements AAAA

Les enregistrements `AAAA` renseignent les associations d'un nom vers une adresse IPv6. Par exemple :

```
www.renater.fr. 1200 IN AAAA 2001:660:3001:4002::10
```

Enregistrements CNAME

Les enregistrements `CNAME` renseignent les associations d'un nom vers un autre nom (alias). Par exemple :

```
www.kernel.org. 600 IN CNAME www.geo.kernel.org.
```

définit que `www.kernel.org.` est un alias de `www.geo.kernel.org.`. La résolution de nom complète du nom `www.kernel.org.` vers une adresse IP demandera donc une nouvelle requête DNS pour `www.geo.kernel.org.`.

Enregistrements PTR

Les enregistrements `PTR` renseignent les associations d'une adresse IP (IPv4 ou IPv6) vers un nom.

Par exemple :

```
135.101.52.193.in-addr.arpa. 604792 IN PTR MX1.univ-nantes.fr.
```

associe l'adresse IPv4 193.52.101.135 vers le nom `MX1.univ-nantes.fr.`. Afin de préserver l'organisation hierarchique du DNS, les adresses IP sont lues de droite à gauche, chaque octet de l'adresse IP correspondant à un sous-domaine. Le domaine `in-addr.arpa` est la racine correspondant aux résolutions d'adresses IPv4 vers des noms de domaines [3].

Le même principe s'applique aux adresses IPv6. Par exemple :

```
1.b.e.d.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.8.3.0.0.2.2.7.0.6.6.0.1.0.0.2.ip6.arpa. 604800 IN P
debian.univ-nantes.fr.
```

associe l'adresse IPv6 `2001:660:7220:381::deb1` au nom `debian.univ-nantes.fr.`. Chaque groupe de 4 bits de l'adresse IPv6 correspond à un niveau de hierarchie dans le DNS. Le domaine `ip6.arpa.` est la racine correspondant aux résolutions d'adresses IPv6 vers des noms.

3. ce type de requête demandant la résolution d'une adresse vers un nom est parfois désigné sous le terme de *résolution inverse*.

Enregistrements MX

Les enregistrement `MX` désignent les serveurs de courrier électroniques associés aux domaines. Par exemple :

```
univ-nantes.fr. 331832 IN MX 10 mx1.univ-nantes.fr.
univ-nantes.fr. 331832 IN MX 10 mx3.univ-nantes.fr.
univ-nantes.fr. 331832 IN MX 20 mx2.univ-nantes.fr.
```

associent au domaine `univ-nantes.fr.` les serveurs de courrier électronique `mx1.univ-nantes.fr.`, `mx2.univ-nantes.fr.` et `mx3.univ-nantes.fr.`. La valeur suivant la chaine `MX` dans l'enregistrement indique une priorité : le serveur ayant la priorité la plus faible sera sollicité.

Enregistrements NS

Les enregistrement `NS` spécifient les serveurs de noms (primaires ou secondaires) qui ont autorité pour la zone. Par exemple :

```
orange.fr. 339 IN NS ns.wanadoo.fr.
orange.fr. 339 IN NS ns2.wanadoo.fr.
```

indique que les enregistrements de la zone `orange.fr.` sont renseignés par les serveurs `ns.wanadoo.fr.` et `ns2.wanadoo.fr.`.

Enregistrements SOA

Les enregistrements `SOA` (*Start Of Authority*) renseignent les données de gestion d'une zone (domaine) ainsi que le serveur primaire ayant autorité sur cette zone. Sa forme est la suivante :

```
zone TTL IN SOA NS1 MAIL serial refresh retry expiry min
```

Les champs `TTL` (*Time To Live* ou durée de vie) et `IN` sont des champs génériques de tout enregistrement. Le champ `zone` désigne le domaine sur lequel porte l'enregistrement SOA.

- Le champ `NS1` désigne un serveur de noms valide (primaire ou secondaire) pour la zone, en général le serveur de noms principal.
- Le champ `MAIL` renseigne une adresse de courrier électronique de contact pour la gestion de la zone. Le caractère `@` de l'adresse de courrier électronique est remplacé par un point.
- Le champ `serial` contient un numéro de série de la zone. Ce numéro doit être incrémenté à chaque modification de zone car il est utilisé par les serveurs de noms secondaires pour déterminer la version de la zone.
- Le champ `refresh` désigne le délai, exprimé en secondes, au bout duquel les serveurs de noms secondaires vont réintéroger les serveurs de noms primaires pour prendre en compte d'éventuelles modifications d'enregistrements sur la zone.
- Le champ `retry` désigne l'intervalle dans lequel les serveurs de noms secondaires vont réessayer d'interroger les serveurs primaires en cas d'absence de réponse.
- Le champ `expiry` désigne le délai d'expiration de la zone. Les serveurs secondaires ne vont plus accepter de requêtes sur la zone si ce délai est dépassé et que les serveurs primaires ne sont plus joignables.
- Le champ `minimum` (*negative caching time*) désigne le délai minimum pendant lequel un sous-domaine de la zone est considéré comme inexistant dans les caches. Par exemple si un *resolver* demande une information sur un sous-domaine inexistant dans une zone, il pourra garder l'information sur la non-existance du sous-domaine pendant ce délai. Selon les implémentations de serveurs de noms, l'interprétation de cette valeur peut être différente (d'anciennes versions de BIND considéraient cette valeur comme étant une valeur de TTL par défaut pour tous les enregistrements de la zone ne renseignant pas explicitement de TTL ; la version 9 de BIND rend obligatoire la définition de TTL pour tout enregistrement et donc confère à cette valeur sa définition de délai négatif de cache).

Exemple :

```
univ-nantes.fr. 86400 IN SOA dns1.univ-nantes.fr.
                                        dnsmaster.univ-nantes.fr.
                                        2009101501 ; serial
                                        10800      ; refresh
                                        3600       ; retry
                                        604800     ; expiry
                                        86400      ; minimum
```

5.2.6 Résolution itérative / récursive

Pour obtenir une information concernant un nom de domaine, le *resolver* cherche le premier serveur qui possède l'information. La recherche est soit *itérative* lorsque le *resolver* interroge lui-même les serveurs de noms correspondant aux domaines renseignés dans le nom à résoudre ou bien *récursive* lorsque le *resolver* interroge un serveur de noms qui va lui-même potentiellement interroger un autre serveur de nom et ainsi de suite récursivement.

Par exemple : pour obtenir l'adresse associée au nom `www.polytech.univ-nantes.fr.`, le *resolver* détermine d'abord s'il a cette information dans son cache ; sinon, il a deux options :

1. la recherche itérative :
   (a) le *resolver* interroge un serveur racine pour déterminer l'adresse d'un serveur de noms du domaine `fr`,
   (b) il interroge ensuite un serveur de noms du domaine `fr` pour déterminer l'adresse d'un serveur de noms du domaine `univ-nantes`,
   (c) il interroge ensuite un serveur de noms du domaine `univ-nantes` pour déterminer l'adresse d'un serveur de noms du domaine `polytech`,
   (d) enfin, il interroge ce serveur afin de déterminer l'adresse associée au nom `www`.
2. la recherche récursive : le *resolver* interroge le serveur de noms qui lui est associé. Ce serveur peut alors soit résoudre le nom de manière itérative ou bien à son tour faire une requête récursive vers un autre serveur de noms.

5.3 LDAP

LDAP (*Lightweight Directory Access Protocol*) est un protocole permettant l'accès à des annuaires. Ces annuaires sont standardisés et extensibles [RFC 2251]. LDAP était à l'origine une passerelle d'accès à des annuaires X500 (normalisé ISO mais très complexe), mais dès 1995, l'Université du Michigan, puis à partir de 1998, l'initiative *OpenLDAP* [OpenLDAP] ont proposé des annuaires autonomes, accessibles à partir du protocole LDAP.

5.3.1 DIT

Les données LDAP sont structurées dans une arborescence hiérarchisée comparable aux systèmes de fichiers. Chaque nœud de l'arbre (ou *Directory Information Tree (DIT)*), correspond à une entrée de l'annuaire ou *Directory Service Entry (DSE)*. La racine de l'arbre correspond généralement à l'entité organisationnelle de l'annuaire [4]. La figure 5.2 propose un exemple d'un tel arbre. Contrairement aux bases de données, les accès en lecture dans les données d'annuaires sont supposés plus fréquents que les modifications.

4. l'espace de nommage d'un annuaire LDAP est donc local, contrairement au DNS, qui lui est global.

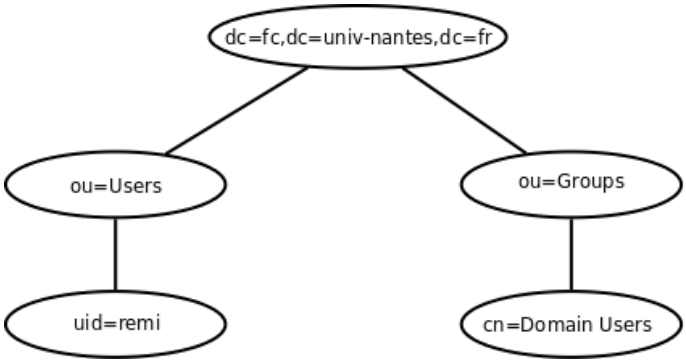


FIGURE 5.2 – Exemple de *DIT*

Attributs

Chaque entrée de l'annuaire contient une suite de couples *attribut : valeur*. Chaque attribut est caractérisé par :

— un nom qui l'identifie,
— un Object Identifier (OID) qui l'identifie également,
— s'il est mono ou multi-valué,
— une syntaxe et des règles de comparaison,
— un indicateur d'usage,
— un format ou une limite de taille de valeur qui lui est associée.

5.3.2 Les schémas

Les entrées d'un annuaire LDAP sont décrites en suivant un *schéma*. Ce schéma décrit des classes d'objet, les types d'attribut que renseignent ces classes ainsi que leur syntaxe.

Classes d'objets

Les classes d'objets modélisent des objets réels ou abstraits en les caractérisant par une liste d'attributs optionnels ou obligatoires. Une classe d'objet est définie par :

— un nom qui l'identifie,
— un *OID* (*Object IDentifier*) qui l'identifie également,
— des attributs obligatoires,
— des attributs optionnels,
— un type (structurel, auxiliaire ou abstrait).

Les classes d'objets forment une hiérarchie, au sommet de laquelle se trouve l'objet `top`. Chaque objet hérite des propriétés (attributs) de l'objet dont il est le fils. On peut donc enrichir un objet en créant un objet fils qui lui rajoute des attributs supplémentaires. On précise la classe d'objet d'une entrée à l'aide de l'attribut `objectClass`. Il faut obligatoirement indiquer la parenté de la classe d'objet en partant de la classe `top` et en passant par chaque ancêtre de l'objet. Par exemple, la classe `inetOrgPerson` a la filiation suivante :

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

Dans cet exemple :

— la classe `person` a comme attributs : `commonName`, `surname`, `description`, `seeAlso`, `telephoneNumber`, `userPassword`,
— la sous-classe `organizationalPerson` ajoute des attributs comme : `organizationUnitName`, `title`, `postalAddress`...

— La sous-classe `inetOrgPerson` lui rajoute des attributs comme : `mail`, `labeledURI`, `uid` *(userID)*, `photo` ...

Les OIDs

Les objets, les classes et les attributs LDAP sont référencés par un numéro unique ou (*OID* pour *Object IDentifier*), standardisé par l'IANA (*Internet Assigned Numbers Authority*).

Un OID est une séquence de nombres entiers séparés par des points. Les OID sont alloués de manière hiérarchique de telle manière que seule l'autorité qui a délégation sur la hiérachie "1.2.3" peut définir la signification de l'objet "1.2.3.4". Par exemple :

`2.5`	fait référence au service X500
`2.5.4`	est la définition des types d'attributs
`2.5.6`	est la définition des classes d'objets
`1.3.6.1`	OIDs pour l'Internet
`1.3.6.1.4.1`	OIDs qui peuvent être délégués
`1.3.6.1.4.1.4203`	OpenLDAP

5.3.3 Représentation de données d'annuaires

Le *Distinguish Name* (*dn*)

Chaque entrée est référencée de manière unique dans le DIT par son *distinguished name (DN)*. Le DN représente le nom de l'entrée sous la forme du chemin d'accès à celle-ci depuis le sommet de l'arbre. On peut comparer le DN à un chemin d'accès dans un système de fichiers. Par exemple, par rapport à la figure 5.2, le DN suivant :

```
uid=remi,ou=Users,dc=fc,dc=univ-nantes,fc=fr
```

représente un chemin absolu permettant d'identifier une entrée dans l'annuaire. Il est possible d'exprimer des chemins d'accès relatifs, par exemple par rapport à la figure 5.2 et relativement à la racine de l'arbre (`dc=fc,dc=univ-nantes,dc=fr`) :

```
ou=Users
ou=Groups
uid=remi,ou=Users
cn=Domain Users,ou=Groups
```

LDIF

Le format *LDIF* (*LDAP Data Interchange Format*) permet de représenter des données d'annuaire sous forme de documents textuels. Il sert notamment à réaliser des imports ou des exports de données de l'annuaire ou à effectuer des modifications sur ces données.

La syntaxe de ce fichier correspond à une liste d'entrées (1 paragraphe par entrée d'annuaire). Pour chaque entrée de l'annuaire, les attributs et leurs valeurs sont listées (1 ligne par attribut) sous la forme `attribut: valeur`. Chaque entrée doit commencer par son DN. Par exemple :

```
dn: uid=remi,ou=Users,dc=fc,dc=univ-nantes,dc=fr
objectClass: top
objectClass: person
```

```
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: sambaSamAccount
cn: remi
sn: remi
givenName: remi
uid: remi
uidNumber: 1136
gidNumber: 1005
homeDirectory: /home/remi
loginShell: /bin/bash
gecos: Remi Lehn
structuralObjectClass: inetOrgPerson
entryUUID: 72d9e204-3ff3-102b-8112-b7dc9dc9ae4e
creatorsName: cn=admin,dc=fc,dc=univ-nantes,dc=fr
createTimestamp: 20070124123751Z
sambaLogonTime: 0
sambaLogoffTime: 2147483647
sambaKickoffTime: 2147483647
sambaPwdCanChange: 0
displayName: Remi Lehn
sambaSID: S-1-5-21-2357754865-1191189541-1474276241-3272
sambaPrimaryGroupSID: S-1-5-21-2357754865-1191189541-1474276241-1005
sambaLMPassword: ********************************
sambaAcctFlags: [U]
sambaNTPassword: ********************************
sambaPwdLastSet: 1169642285
sambaPwdMustChange: 1173530285
userPassword:: ************************************************=
mail: remi@fc.univ-nantes.fr
entryCSN: 20070124123805Z#000002#00#000000
modifiersName: cn=admin,dc=fc,dc=univ-nantes,dc=fr
modifyTimestamp: 20070124123805Z
```

5.3.4 Accès aux annuaires

LDAP standardise un certain nombre d'accès de base, notamment :
— **Search** : recherche dans l'annuaire d'objets à partir de critères,
— **Compare** : comparaison de 2 entrées de l'annuaire,
— **Add** : ajout d'une entrée,
— **Modify** : modification du contenu d'une entrée,
— **Delete** : suppression d'une entrée,
— **Rename** : modification du DN d'une entrée,
— **Bind** : connexion au serveur d'annuaire,
— **Unbind** : deconnexion du serveur d'annuaire.

Les commandes *Search* et *Compare* se font au moyen d'une requête composée de 8 paramètres :
— **base object** : l'endroit de l'arbre où doit commencer la recherche,
— **scope** : la profondeur de la recherche,
— **derefAliases** : si on suit les liens ou pas,
— **size limit** : nombre de réponses limite,
— **time limit** : temps maxi alloué pour la recherche,
— **attrOnly** : renvoie ou pas la valeur des attributs en plus de leur type,
— **search filter** : le filtre de recherche,

— **list of attributes** : la liste des attributs à renvoyer.

Le *scope* définit la profondeur de recherche dans le DIT :

— si `scope=base`, seule l'entrée correspondant au paramètre `base object` est évaluée,
— si `scope=one` (*one level*), permet d'interroger un niveau de profondeur,
— si `scope=sub` (*subtree*), permet d'interroger le sous-arbre dont la racine correspond au paramètre `base object`.

Le *filtre* permet de spécifier les critères de recherche. Il peut être construit au moyen des opérateurs suivants :

— égalité (par exemple `(uidNumber=1136)`),
— approximation (par exemple `(displayName =Remi Lehn)`) : avec une orthographe voisine,
— comparaison (par exemple `(gidNumber>1000)`), <= , >= , < : la comparaison est alphabétique ou numérique selon le type de l'attribut,
— présence (par exemple `(sambaSID:=*)`) : toutes les entrées ayant un attribut donné,
— sous-chaîne (par exemple `(displayName=*Lehn*)`) : permet des expressions avec une syntaxe similaire à celle de la commande `grep`,
— et logique (par exemple `(&(displayName=Remi Lehn) (ou=fc))`),
— ou logique (par exemple `(|(ou=fc) (ou=polytech))`),
— négation logique (par exemple `(!(mail=*))` : toutes les entrées sans attribut `mail`).

La commande `ldapsearch` permet d'interroger un annuaire LDAP. Par exemple :

```
$ ldapsearch -H ldap://ldap.univ-nantes.prive -b o=univ-nantes.fr -x \
      '(telephoneNumber=0251250747)'
```

Interroge l'annuaire de l'Université de Nantes pour déterminer l'entrée correspondant au numéro de téléphone 0251250747. L'option `-b` indique la base de la recherche, le *scope* est par défaut *subtree* et le dernier argument de la commande spécifie le filtre.

Les URL LDAP

LDAP permet d'exprimer des critères de recherche au moyen d'URLs de la forme :

```
ldap[s]://<hostname>:<port>/<base_dn>?<attributes>?<scope>?<filter>
```

Protocoles SMB et CIFS

6.1 Introduction

Les protocoles *SMB* (*Server Message Block*) et *CIFS* (*Common Internet File System*) sont des protocoles d'application permettant le partage de fichiers, d'imprimantes, voire d'autres périphériques. Il est notamment mis en œuvre sur les systèmes Microsoft Windows [1].

Ces protocoles s'appuient aujourd'hui majoritairement sur TCP/IP pour le transport, mais historiquement, un transport spécifique, *NetBEUI* avait été défini par Microsoft, spécialement pour le transport de messages *SMB*. Un protocole spécifique, *NetBios* a également été introduit pour le nommage et la localisation des ressources.

Il existe différentes versions de ces protocoles :

— *SMB*, proposé sur les systèmes Windows depuis Windows 3.11,
— *CIFS*, introduit par Microsoft dans Windows NT 4,
— *SMB 2*, introduit par Microsoft dans Windows Vista, Windows 7 et Windows 2008 Server,
— *SMB 2.1*, introduit par Microsoft dans Windows Server 2008 R2.

SMB est un protocole fonctionnant principalement en mode non connecté (*UDP*). Les accès sont transcrits en une suite de messages d'une taille de 64 Ko. maximum. *CIFS* ajoute un certain nombre de fonctionnalités : la possibilité de liens et de liens symboliques et le support des fichiers de taille supérieure à 2 Go. Il propose également un fonctionnement en mode connecté en utilisant un transport TCP. Enfin il s'affranchit du protocole *NetBIOS* pour la résolution de noms, au profit de l'utilisation du DNS.

SMB 2 correspond principalement à une amélioration en termes de performances des protocoles SMB/CIFS, ainsi que le support d'un fonctionnement hors-ligne, en constituant un cache local.

6.2 Principes de fonctionnement

6.2.1 Client-serveur

Les protocoles SMB/CIFS fonctionnent selon un principe client-serveur. Il n'est cependant fréquent que des noeuds jouent à la fois le rôle de serveur (mettent des ressources (fichiers, imprimantes, ...) à disposition) et de clients (accèdent à des ressources partagées par des serveurs).

6.2.2 Authentification

Les protocoles SMB/CIFS proposent quatre modes d'identification :

1. par ressource partagée : seule l'identification de la ressource partagée (ensemble de fichiers, imprimante, ...) est utilisée conjointement à un mot de passe,

1. le logiciel libre *Samba* met toutfois en œuvre ce protocole.

2. par utilisateur : des utilisateurs et des droits d'accès des utilisateurs sur les ressources partagées sont définis, au niveau de chaque partage,
3. par domaine : la description des utilisateurs et des droits d'accès est définie de manière centralisée, au niveau d'un *contrôleur de domaine*,
4. par annuaire : la description des utilisateurs et des droits d'accès est définie de manière centralisée[2] par un annuaire ; Microsoft propose un service spécifique, *Active Directory*, basé sur LDAP, pour l'implémentation de cet annuaire. Il est également possible d'utiliser d'autres serveurs d'annaires (*OpenLDAP* par exemple), conjointement au logiciel *Samba* pour réaliser le même rôle.

2. voire décentralisée.

Le protocole HTTP

7.1 Applications

7.1.1 Le World Wide Web

HTTP (*HyperText Transfer Protocol*) [RFC 2068] est le protocole le plus utilisé pour faire communiquer des clients WWW avec des serveurs WWW.

Le protocole *HTTP* est un protocole fonctionnant sur le modèle *requête/réponse*. Un client envoie une requête au serveur qui lui renvoie une réponse correspondant au document demandé.

Requête HTTP

La requête provenant du client contient :
— Une méthode de requête (`HEAD`, `GET`, `POST`, `OPTIONS`, `PUT`, `DELETE`, `TRACE`).
— Un `URI` (*Uniform Resource Identifier*) représentant le document demandé.
— Le protocole utilisé.
— Une zone d'informations sur le client, contenant le type de documents que le client accepte, notamment.

La figure 7.1 donne un exemple d'une requête HTTP.
— La première ligne de la requête se décompose en :
  — Méthode : `GET`
  — URI : `/index.html`
  — Protocole : `HTTP/1.0`
— Les autres lignes correspondent aux informations sur le client.
— La requête est terminée par une ligne vide.

Méthodes de requête

La méthode `GET` permet de récupérer un document à partir de son URI.

```
GET /index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.06 [en] (X11; I; Linux 2.0.35 i586)
Host: gin:80
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

FIGURE 7.1 – Exemple de requête HTTP

La méthode `HEAD` est similaire à la méthode `GET`, mais le serveur ne renvoie pas le corps du document dans ce cas.

La méthode `POST` permet d'envoyer des données (des réponses à un formulaire, par exemple) à un serveur WWW.

URI

Les URI permettent d'identifier un document sur le réseau. Un URI peut être soit relatif à un serveur (il commence par le caractère `/`), soit correspondre à un document se trouvant sur un autre serveur WWW.

Dans le premier cas, le document peut être directement retrouvé sur le serveur WWW, l'URI étant concaténée au chemin d'accès correspondant au répertoire racine contenant les documents servis par le serveur. Par exemple, l'URI :

`/repertoire/page.html`

correspondra au document stocké dans le fichier ayant pour chemin d'accès :

base`/repertoire/page.html`

sur le serveur WWW. Cet URI correspondant à l'*URL* (≡ *Universal Resource Locator*) :

`http://`*serveur*`[:`*port*`]/repertoire/page.html`

URL

Les champs URI des requètes HTTP peuvent être des *URL* (*Universal Resource Locator*). Dans ce cas, une nouvelle requète HTTP est émise par le serveur [1] ayant reçu la requète pour aller chercher le document sur un autre serveur WWW.

Réponse à une requète HTTP

Une réponse d'un serveur WWW à une requète HTTP contient :
- Une ligne d'état, contenant le protocole utilisé par le serveur, ainsi qu'un code indiquant si la transaction demandée a pu être effectuée ou si il y a une erreur.
- Une zone d'information sur le serveur.
- Eventuellement, une zone de meta-information sur le document demandé.
- Eventuellement, le corps du document demandé.

La figure 7.2 correspond à la réponse donnée par un serveur WWW à la requète 7.1.
- La première ligne de cette réponse correspond à :
  - Protocole : `HTTP/1.1`
  - Code d'erreur : `200 OK`
- Le reste de l'entète contient des informations sur le serveur et des méta informations sur le document (ex. la taille du document, 1938 octets).
- Après une ligne vide, on a le corps du document, ici au format *HTML* (≡ *HyperText Markup Language*).

Zone d'information du client

La zone d'information du client peut contenir (entre autre) les champs suivants :
- `Connection` : ce champ peut avoir comme valeurs `Keep-Alive` et `Close`. Si il a la valeur `Keep-Alive`, celà signifie que le client souhaite que la même connexion soit utilisée pour échanger plusieurs documents (c'est une recommandation du protocole `HTTP/1.1`). Si il a la valeur `Close`, celà signifie que la connexion sera refermée après l'échange du document.
- `User-Agent` : ce champ identifie le logiciel client.
- `Host` : ce champ identifie le nom réseau du client ainsi que le port utilisé sur le serveur.

1. Dans le cas où le serveur a une fonctionnalité de *proxy*.

```
HTTP/1.1 200 OK
Date: Wed, 06 Jan 1999 14:40:38 GMT
Server: Apache/1.2.6 Red Hat
Last-Modified: Mon, 10 Aug 1998 15:11:30 GMT
ETag: "1b650-792-35cf0da2"
Content-Length: 1938
Accept-Ranges: bytes
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Test Page for Red Hat Linux's Apache Installation</TITLE>
 </HEAD>
<!-- Background white, links blue (unvisited), navy (visited), red (active) -->
 <BODY
  BGCOLOR="#FFFFFF"
  TEXT="#000000"
  LINK="#0000FF"
  VLINK="#000080"
  ALINK="#FF0000"
 >
  <H1 ALIGN="CENTER">It Worked!</H1>
  <P>
  If you can see this, it means that the installation of the
  <A
   HREF="http://www.apache.org/"
  >Apache</A>
  software on this <a href="http://www.redhat.com/">Red Hat Linux</a>
  system was successful. You may now add content to
  this directory and replace this page.
  </P>

(...)

 </BODY>
</HTML>
```

FIGURE 7.2 – Réponse à la requète de la figure 7.1

— `Accept` : ce champ indique les types de documents que peut accepter le client. Ces types sont indiqués sous la forme de types *MIME* (≡ *Multipurpose Internet Mail Extensions*), avec éventuellement des caractères *jockers*. Par exemple, le type *MIME* `text/html` désigne du texte au format *HTML* et le type `text/*` désigne du texte, sous n'importe quel format.

Zone d'information sur le serveur

La zone d'information sur le serveur peut contenir entre autres, les champs suivants :

— `Date` : correspond à la date de la requète sur le serveur. Elle est exprimée de la façon suivante :

`Jjj, NN Mmm AAAA HH:MM:SS GMT`

avec :

— `Jjj`, le jour de la semaine, selon la correspondance suivante :

Jour	`Jjj`
Lundi	`Mon`
Mardi	`Tue`
Mercredi	`Wed`
Jeudi	`Thu`
Vendredi	`Fri`
Samedi	`Sat`
Dimanche	`Sun`

— `NN`, le jour du mois.

— `Mmm`, le mois, avec la correspondance suivante :

Mois	`Mmm`
Janvier	`Jan`
Février	`Feb`
Mars	`Mar`
Avril	`Apr`
Mai	`May`
Juin	`Jun`
Juillet	`Jul`
Août	`Aug`
Septembre	`Sep`
Octobre	`Oct`
Novembre	`Nov`
Décembre	`Dec`

— `AAAA`, l'année.

— `HH:MM:SS`, heures, minutes, secondes.

— `GMT`, les trois caractères G.M.T, indiquant que la date doit être donnée dans la zone GMT.

Exemple :

Wed, 06 Jan 1999 14 :40 :38 GMT

correspond au "Mercredi 6 Janvier 1999, 15 heures 40 minutes et 38 secondes", heure de Paris.

— `Server` : Identification du logiciel serveur.

Méta-information sur le document

La zone de méta-information sur le document renvoyé par le serveur peut comporter, entre autres, les champs suivants :

— `Last-Modified` : la date de dernière modification du document ; elle est au même format que pour le champ `Date`.

— `Content-Length` : la taille (en octets) du document renvoyé.

— `Content-type` : le type MIME du document renvoyé.

— `Connection` : ce champ peut correspondre aux valeurs `Keep-Alive` ou `Close`, indiquant si le serveur a refermé (`Close`) ou non (`Keep-Alive`) la connexion.

Codes d'erreur HTTP

Les codes de retour sont définis sur 3 chiffres en 5 classes selon la valeur du premier chiffre du code de retour.

— *1xx* : codes d'information.
— *2xx* : succès. La requète a été traitée normalement.
— *3xx* : redirection. Un de ces codes est renvoyé lorsque le client doit effectuer une nouvelle requète pour obtenir le document demandé. Ceci peut se produire lorsque un document a été déplacé, par exemple.
— *4xx* : erreur dans la requète du client. Ce code d'erreur est renvoyé par le serveur lorsqu'il reçoit une requète qu'il ne peut pas traiter. Parmi ces erreurs, les codes suivants sont les plus fréquents :
  — *400 Bad Request* : Requète mal formulée par le client ; le serveur ne peut pas interpréter la requète.
  — *404 Not Found* : Le document demandé n'est pas disponible sur le serveur.
— *5xx* : erreurs du serveur.

Bibliographie

[Leiner et al., 1998] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff. *A Brief History of the Internet*, Internet Society, `http://www.isoc.org/internet/history/brief.html`, 1998.

[Cerf & Kahn, 1974] Vinton G. Cerf and Robert E. Kahn. "A protocol for packet network interconnection", *IEEE Transactions on Communication Technologies*, volume COM-22 V 5, pages 627–641, May 1974.

[Zakon, 2000] Robert H. Zakon. *Hobbes' Internet Timeline*, Internet Society, `http://www.isoc.org/guest/zakon/Internet/History/HIT.html`, 2000.

[Huitema, 1995] Christian Huitema. *Et Dieu Créa l'Internet*, Edition Eyrolles, 1995.

[RFC 2460] S. Deering, R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, IETF, 1998.

[Spurgeon, 1999] Charles Spurgeon. *Charles Spurgeon's Ethernet Web Site*, `http://wwwhost.ots.utexas.edu/ethernet`, 1999.

[RFC 894] Charles Hornig. *A Standard for the Transmission of IP Datagrams over Ethernet Networks*, IETF, 1984.

[RFC 1661] W. Simpson. *The Point-to-Point Protocol (PPP)*, IETF, 1994.

[Leiner et al., 1998] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff. *A Brief History of the Internet*, Internet Society, `http://www.isoc.org/internet/history/brief.html`, 1998.

[Zakon, 2000] Robert H. Zakon. *Hobbes' Internet Timeline*, Internet Society, `http://www.isoc.org/guest/zakon/Internet/History/HIT.html`, 2000.

[Ikonicoff, 1999] Roman Ikonicoff. "La communication globale", *Science & vie*, numéro 987, pages 89–101, Décembre 1999.

[Huitema, 1995] Christian Huitema. *Et Dieu Créa l'Internet*, Edition Eyrolles, 1995.

[RFC 2460] S. Deering, R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, IETF, 1998.

[RFC 894] Charles Hornig. *A Standard for the Transmission of IP Datagrams over Ethernet Networks*, IETF, 1984.

[RFC 1661] W. Simpson. *The Point-to-Point Protocol (PPP)*, IETF, 1994.

[RFC 791] Jon Postel. *Internet Protocol*, Defense Advanced Research Project Agency (DARPA), 1981.

[RFC 2474] K. Nichols, S. Blake, F. Baker, D. Black. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, IETF, 1998.

[RFC 2453] G. Malkin. *RIP Version 2*, IETF, 1998.

[RFC 2328] J. Moy. *OSPF Version 2*, IETF, 1998.

[RFC 911] P. Kirton. *EGP Gateway under Berkeley UNIX 4.2*, IETF, 1984.

[RFC 1771] Y. Rekhter and T. Li. *A Border Gateway Protocol 4 (BGP-4)*, IETF, 1995.

[Dawson & Rubini, 1998] Terry Dawson and Alessandro Rubini. *Linux NET-3-HOWTO, Linux Networking*, Linux Documentation Project, 1998.

[RFC 768] J. Postel. *User Datagram Protocol*, IETF, 1980.

[RFC 793] J. Postel. *Transmission Control Protocol*, IETF, 1981.

[RFC 2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, IETF & w3C, 1997.

[Leiner et al., 1998] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff. *A Brief History of the Internet*, Internet Society, `http://www.isoc.org/internet/history/brief.html`, 1998.

[RFC 894] Charles Hornig. *A Standard for the Transmission of IP Datagrams over Ethernet Networks*, IETF, 1984.

[RFC 1661] W. Simpson. *The Point-to-Point Protocol (PPP)*, IETF, 1994.

[RFC 791] Jon Postel. *Internet Protocol*, Defense Advanced Research Project Agency (DARPA), 1981.

[RFC 2474] K. Nichols, S. Blake, F. Baker, D. Black. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, IETF, 1998.

[RFC 2453] G. Malkin. *RIP Version 2*, IETF, 1998.

[RFC 2328] J. Moy. *OSPF Version 2*, IETF, 1998.

[RFC 911] P. Kirton. *EGP Gateway under Berkeley UNIX 4.2*, IETF, 1984.

[RFC 1771] Y. Rekhter and T. Li. *A Border Gateway Protocol 4 (BGP-4)*, IETF, 1995.

[Dawson & Rubini, 1998] Terry Dawson and Alessandro Rubini. *Linux NET-3-HOWTO, Linux Networking*, Linux Documentation Project, 1998.

[RFC 768] J. Postel. *User Datagram Protocol*, IETF, 1980.

[RFC 793] J. Postel. *Transmission Control Protocol*, IETF, 1981.

[RFC 2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, IETF & w3C, 1997.

[Wikipedia TCP] Wikipedia, *Transmission Control Protocol*. `http://fr.wikipedia.org/wiki/Transmission_Control_Protocol`

[RFC 2251] M. Wahl, T. Howes, S. Kille *Lightweight Directory Access Protocol (v3)* IETF, 1997

[OpenLDAP] `http://www.openldap.org/`

[Tuto LDAP] Laurent Mirtain *LDAP* `http://www-sop.inria.fr/members/Laurent.Mirtain/ldap-livre.html` Inria, 1999

[RFC 2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, IETF & w3C, 1997.