2. Le réseau Internet et les protocoles TCP/IP.

Ces 15 dernières années ont vu émerger de nouvelles techniques rendant possible l'interconnexion de réseaux différents (*internetworking*) en les faisant apparaître comme un unique environnement de communication homogène. On désigne ce système d'interconnexion sous le nom d'*internet*, sachant que *réseau Internet* et *Internet* désignent l'ensemble de ces internets dont le point commun est de fonctionner en suivant les protocoles TCP/IP (*Transmission Control Protocol/Internet Protocol*). Le but de ce chapitre est d'étudier comment fonctionne l'ensemble de ces protocoles.

2.1 Historique et organisation d'Internet.

Les travaux de l'ARPA (*Advanced Research Project Agency*) débutèrent au milieu des années 70 et avaient pour but de développer un réseau à commutation de paquets pour relier ses centres de recherches dans le but de partager des équipements informatiques et échanger des données et du courrier. Le but était de concevoir un réseau résistant à des attaques militaires. Il ne fallait donc pas qu'il comporte de points névralgiques dont la destruction aurait entraîné l'arrêt complet du réseau. C'est ainsi, que dès le départ le réseau ARPANET fut conçu sans nœud particulier le dirigeant, et de telle sorte que si une voie de communication venait à être détruite, alors le réseau soit capable d'acheminer les informations par un autre chemin. C'est vers 1980 qu'est apparu le réseau Internet, tel qu'on le connaît maintenant, lorsque l'ARPA commença à faire évoluer les ordinateurs de ses réseaux de recherche vers les nouveaux protocoles TCP/IP et qu'elle se mit à subventionner l'université de Berkeley pour qu'elle intègre TCP/IP à son système d'exploitation Unix (BSD). Ainsi la quasi totalité des départements d'informatique des universités américaines put commencer à se doter de réseaux locaux qui en quelques années seront interconnectés entre eux sous l'impulsion de la NSF (*National Science Foundation*).2 Même si dès son origine Internet comprenait des sociétés privées, celles-ci étaient plus ou moins liées à la recherche et au développement, alors qu'à l'heure actuelle les activités commerciales s'y sont considérablement multipliées, et ceci surtout depuis l'arrivée du *web* en 1993.

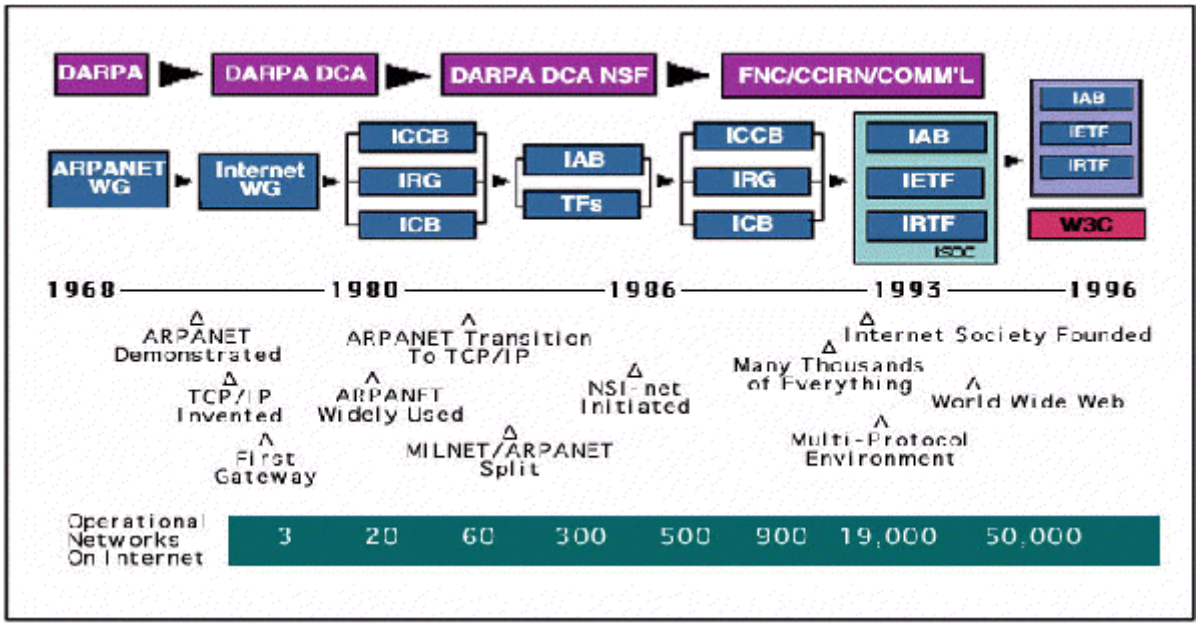

Figure 2.1: Les grandes dates d'Internet.

La figure 2.1 (source ISOC, `www.isoc.org`) donne un résumé des grandes étapes de l'évolution d'Internet au niveau mondial qui comportait en 1996 plus de 100 000 réseaux différents permettant de regrouper presque 10 millions d'ordinateurs dans le monde. Mais les statistiques sont difficiles à établir et sont parfois fantaisistes ou biaisées par des considérations politiques ou commerciales.

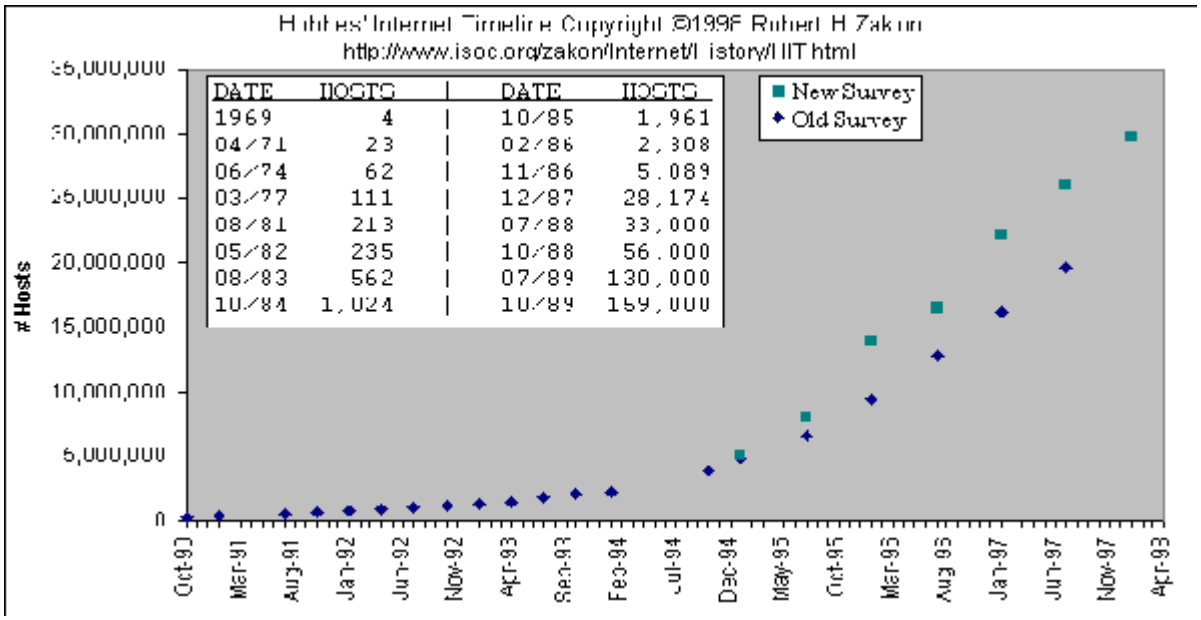

Figure: Évolution du nombre de machines connectées à Internet

Une bonne source d'information est encore l'ISOC dont sont extraites les données de la figure 2.2. Pour ce qui est de la France, après des tentatives avortées de constitution d'un réseau de la recherche, puis l'apparition du réseau EARN (*European Academic and Research Network*) basé sur des protocoles et des ordinateurs IBM, un début de réseau bâti sur des ordinateurs Unix et TCP/IP apparu sous le nom de FNET. C'est à la fin des années 80 que les campus universitaires s'équipèrent massivement de réseaux Ethernet et créèrent des réseaux régionaux basés sur TCP/IP. L'ouverture à l'Internet mondial (à l'époque presque exclusivement nord-américain) eut lieu en 1988 et ensuite la création de Renater (*RÉseau National de Télécommunications pour l'Enseignement et la Recherche*) en 1994 sont les grandes dates de l'évolution d'Internet en France.

Comme l'ensemble des protocoles TCP/IP n'est pas issu d'un constructeur unique, mais émane de la collaboration de milliers de personnes à travers le monde, une structure de fonctionnement originale a été imaginée dès le début. Après des évolutions successives, c'est maintenant l'IAB (*Internet Architecture Board*) qui est le comité chargé de coordonner l'architecture, les orientations, la gestion et le fonctionnement d'Internet. L'IAB comporte deux branches principales :

- l'IETF (*Internet Engineering Task Force*, `www.ietf.org`) s'occupe des problèmes techniques à court et moyen terme et est divisé en 9 zones (applications, sécurité, routage et adressage, etc...) chacune dotée d'un responsable.

- l'IRTF (*Internet Research Task Force*, `www.irtf.org`) coordonne les activités de recherche relatives à TCP/IP.

Par ailleurs, il existe l'ISOC (*The Internet Society*) qui est liée à l'IAB et qui aide ceux qui souhaitent s'intégrer à la communauté d'Internet. De nombreux renseignements sur le fonctionnement et les organismes liés à Internet sont disponibles sur le web de l'ISOC `www.isoc.org`.

Aucun constructeur, ou éditeur de logiciel, ne peut s'approprier la technique TCP/IP, les documentations techniques sont donc mises à disposition de tous par l'INTERNIC (*Internet Network Information Center* à partir de son site web `ds.internic.net/ds/dspg1intdoc.html`). Les documents relatifs aux travaux sur Internet, les nouvelles propositions de définition ainsi que les modifications de protocoles, tous les standards TCP/IP, y sont publiés sous la forme de RFC (*Request For Comments*, appels à commentaires). Tous les RFC sont publiés par un membre de l'IAB et sont numérotés séquentiellement, une proposition de RFC s'appelle un *Internet Draft* qui sera discuté, modifié, et enfin adopté ou rejeté par les membres du domaine concerné par la note technique.

2.2 Architecture des protocoles TCP/IP.

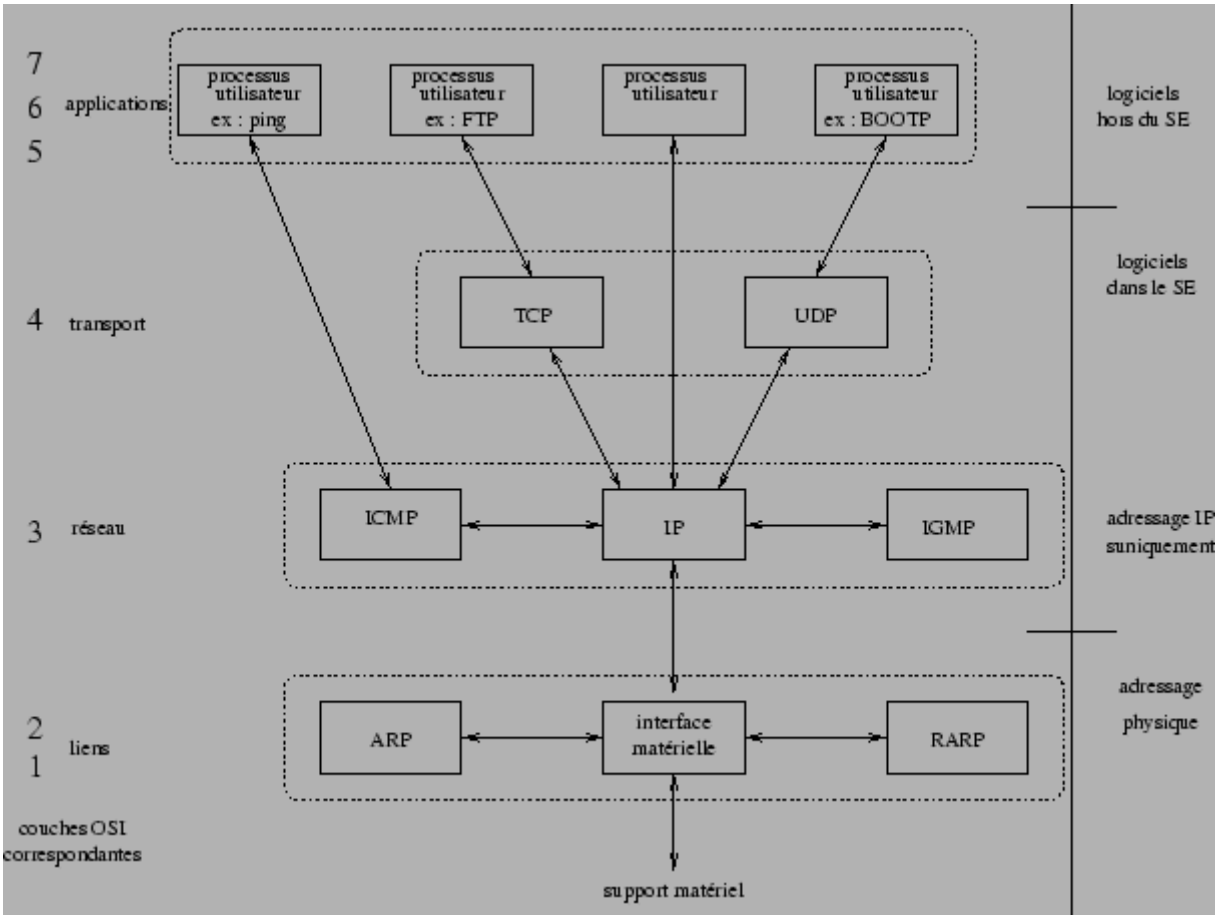

Figure 2.3: Architecture d'une pile TCP/IP

Les logiciels TCP/IP sont structurés en quatre couches de protocoles qui s'appuient sur une couche matérielle comme illustré dans la figure 2.3.

- La couche de *liens* est l'interface avec le réseau et est constituée d'un driver du système d'exploitation et d'une carte d'interface de l'ordinateur avec le réseau.

- La couche *réseau* ou couche IP (*Internet Protocol*) gère la circulation des *paquets* à travers le réseau en assurant leur routage. Elle comprend aussi les protocoles ICMP (*Internet Control Message Protocol*) et IGMP (*Internet Group Management Protocol*)

- La couche *transport* assure tout d'abord une communication de bout en bout en faisant abstraction des machines intermédiaires entre l'émetteur et le destinataire. Elle s'occupe de réguler le flux de données et assure un transport fiable (données transmises sans erreur et reçues dans l'ordre de leur émission) dans le cas de TCP (*Transmission Control Protocol*) ou non fiable dans le cas de UDP (*User Datagram Protocol*). Pour UDP, il n'est pas garanti qu'un paquet (appelé dans ce cas *datagramme*) arrive à bon port, c'est à la couche application de s'en assurer.

- La couche *application* est celle des programmes utilisateurs comme *telnet* (connexion à un ordinateur distant), FTP (*File Transfert Protocol*), SMTP (*Simple Mail Transfert Protocol*), etc...

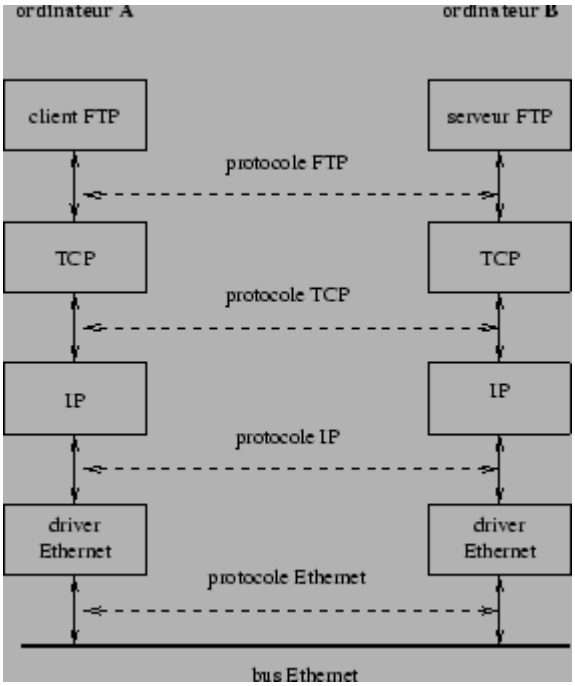

Figure: Communication entre deux machines du même réseau

Cette architecture et ces différents protocoles permettent de faire fonctionner un réseau local, par exemple sur un bus Ethernet reliant un ordinateur client A qui interroge un serveur FTP B, comme illustré dans la figure 2.4 Mais, ceci permet surtout de constituer un internet, c'est-à-dire une interconnexion de réseaux éventuellement hétérogènes comme illustré dans la figure 2.5.

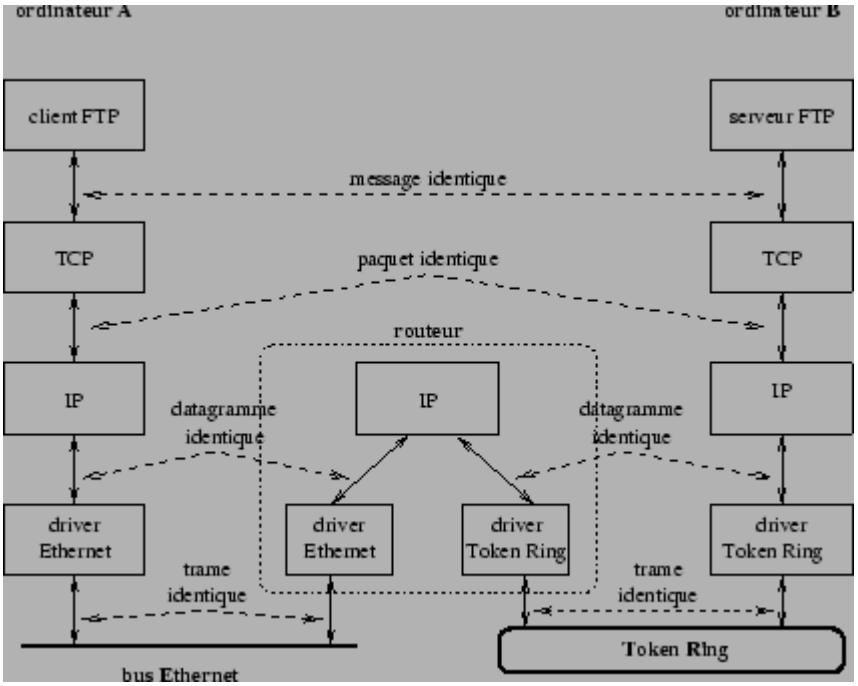

Figure: Interconnexion de deux réseaux

Ici les ordinateurs A et B sont des *systèmes terminaux* et le routeur est un *système intermédiaire*. Comme on peut le voir, la remise du datagramme nécessite l'utilisation de deux *trames* différentes, l'une du réseau Ethernet entre la machine A et le routeur, l'autre du réseau Token-Ring entre le routeur et la machine B. Par opposition, le principe de structuration en couches indique que le paquet reçu par la couche transport de la machine B est identique à celui émis par la couche transport de la machine A.

Lorsqu'une application envoie des données à l'aide de TCP/IP les données traversent de haut en bas chaque couche jusqu'à aboutir au support physique où elles sont alors émises sous forme se suite de bits.

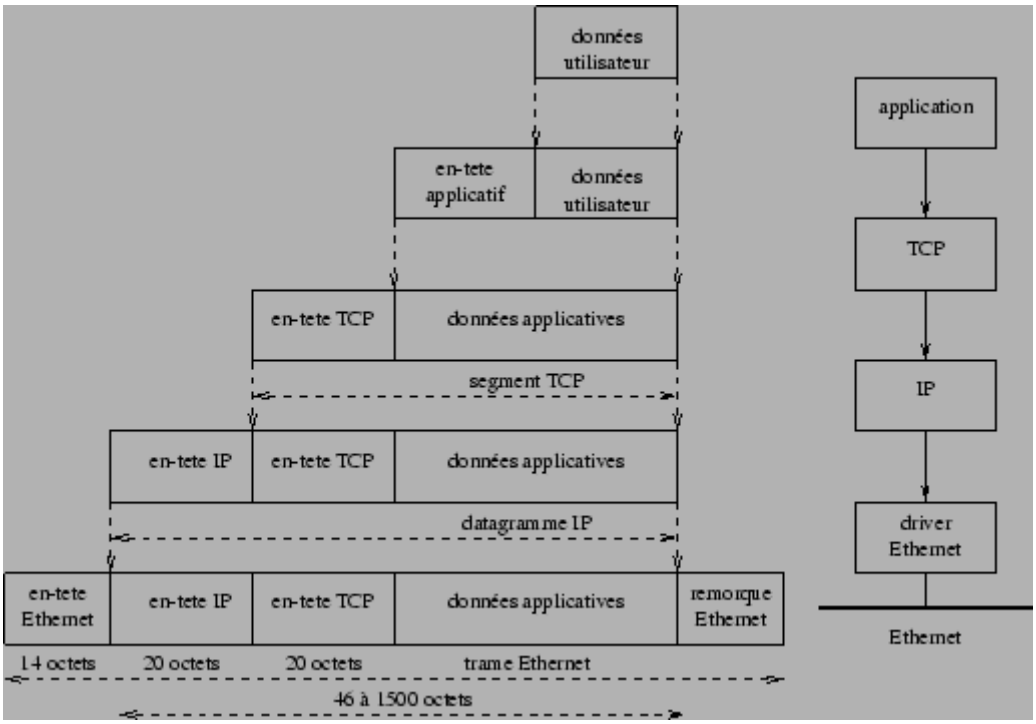

Figure: Encapsulation des données par la pile des protocoles TCP/IP.

L'*encapsulation* illustrée dans la figure 2.6 consiste pour chaque couche à ajouter de l'information aux données en les commençant par des *en-têtes*, voire en ajoutant des informations de remorque. Dans le cas du protocole UDP à la place de TCP, les seuls changements sont que l'unité d'information passé à IP s'appelle un *datagramme UDP* dont l'en-tête a une taille de 8 octets.

2.3 Adressage.

Chaque ordinateur du réseau Internet dispose d'une adresse IP unique codée sur 32 bits. Plus précisément, chaque interface dispose d'une adresse IP particulière. En effet, un même routeur interconnectant 2 réseaux différents possède une adresse IP pour chaque interface de réseau. Une adresse IP est toujours représentée dans une *notation décimale pointée* constituée de 4 nombres (1 par octet) compris chacun entre 0 et 255 et séparés par un point. Ainsi `193.49.144.1` est l'adresse IP d'une des principales machines du réseau de l'université d'Angers.

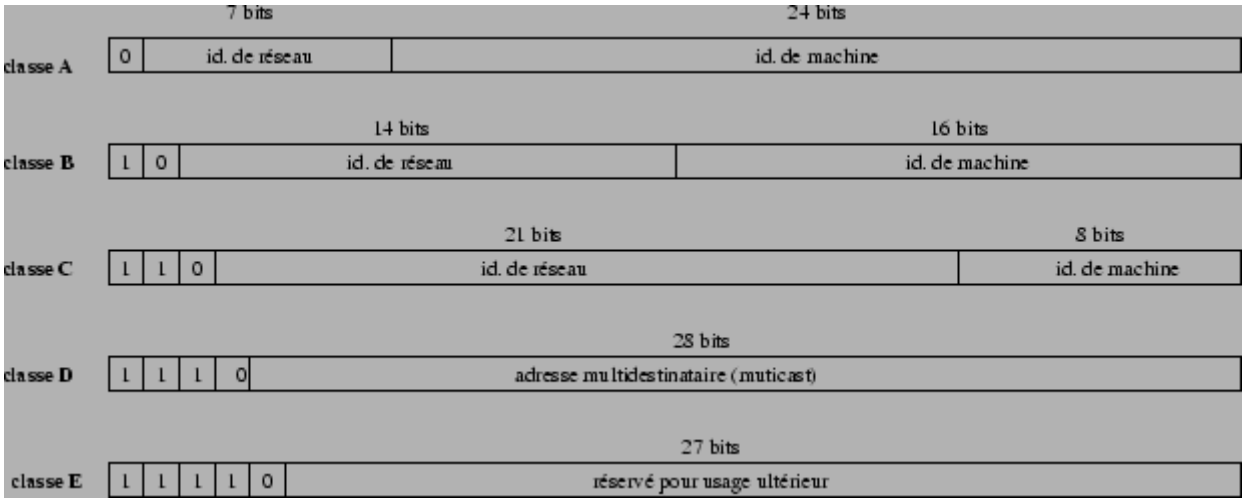

Figure 2.7: Les cinq classes d'adressses IP

Plus précisément, une adresse IP est constituée d'une paire *(id. de réseau, id. de machine)* et appartient à une certaine classe (A, B, C, D ou E) selon la valeur de son premier octet, comme détaillé dans la figure 2.7. Le tableau ci-après donne l'espace d'adresses possibles pour chaque classe.

classe	adresses
A	`0.0.0.0` à `127.255.255.255`
B	`128.0.0.0` à `191.255.255.255`
C	`192.0.0.0` à `223.255.255.255`
D	`224.0.0.0` à `239.255.255.255`
E	`240.0.0.0` à `247.255.255.255`

Ainsi, les adresses de classe A sont utilisées pour les très grands réseaux qui comportent plus de 2^{16}=65 536 ordinateurs. Au niveau mondial, il ne peut exister plus de 127 tels réseaux, par exemple celui de la défense américaine ou du MIT, mais la politique actuelle est de ne plus définir de tels réseaux. Les adresses de classe B sont utilisées pour les réseaux ayant entre 2^8=256 et $2^{16=65\,536}$ ordinateurs, 14 bits définissent l'adresse du réseau et 16 bits celle d'une machine sur le réseau. Seules 256 machines sont possibles sur un réseau de classe C dont le nombre possible dépasse les 2 millions (=2^{21}). L'obtention d'une adresse IP pour créer un nouveau réseau est gérée par l'INTERNIC de manière décentralisée, à savoir qu'un organisme national gère les demandes pour chaque pays. En France c'est l'INRIA (*Institut National de Recherche en Informatique et Automatique*) qui est chargé de cette tâche. Au lieu d'utiliser un adressage plat 1, 2, 3, ... la méthode retenue est plus efficace car elle permet une extraction rapide du numéro de réseau à l'intérieur d'une adresse IP ce qui facilitera le routage.

Toutes les combinaisons mathématiquement possibles pour identifier un réseau ou une machine ne sont pas permises car certaines adresses ont des significations particulières.

- `0.0.0.0` est utilisée par une machine pour connaître sa propre adresse IP lors d'une processus d'amorçage par exemple
- `<id. de réseau nul>.<id. de machine>` est également utilisée pour désigner une machine sur son réseau lors d'un boot également

- `<id. de réseau>.<id. de machine nul>` n'est jamais affectée à une machine car elle permet de désigner le réseau lui-même

- `<id. de réseau>.<id. de machine avec tous ses bits à 1>` est une *adresse de diffusion* ou de *broadcasting*, c'est-à-dire qu'elle désigne toutes les machines du réseau concerné. Un datagramme adressé à cette adresse sera ainsi envoyé à toutes les machines du réseau.

- `255.255.255.255` est une adresse de diffusion locale car elle désigne toutes les machines du réseau auquel appartient l'ordinateur qui utilise cette adresse. L'avantage par rapport à l'adresse précédente est que l'émetteur n'est pas obligé de connaître l'adresse du réseau auquel il appartient.

- `127.X.Y.Z` est une adresse de rebouclage qui est utilisée pour permettre les communications inter-processus sur un même ordinateur ou réaliser des tests de logiciels car tout logiciel de communication recevant des données pour cette adresse les retourne simplement à l'émetteur.

- Les adresses de classe A de `10.0.0.0` à `10.255.255.255`, de classe B de `172.16.0.0` à `172.31.255.255` et de classe C de `192.168.0.0` à `192.168.255.255` sont réservées à la constitution de réseaux privés autrement appelés *intranet*[2.1].

Le système des adresses IP permet également la définition d'adresses de *sous-réseaux* en découpant la partie réservée à l'adresse des machines sur un réseau en deux parties dont la première sera un identificateur de sous-réseau. Ainsi un seul réseau de classe B, sur lequel on pourrait nommer 65 536 machines pourra être décomposé en 254 sous-réseaux de 254 machines, de la manière décrite ci-dessous.

`<id. de réseau sur 16 bits>.<id. de sous-réseau sur 8 bits>.<id. de machine sur 8 bits>`

L'administrateur d'un réseau peut décider de découper où il veut la zone des identificateurs de machines, mais le découpage «autour du .» facilite le travail des routeurs. On peut également adopter le même principe pour un réseau de classe C. Cette technique a pour effet de provoquer un routage hiérarchique.

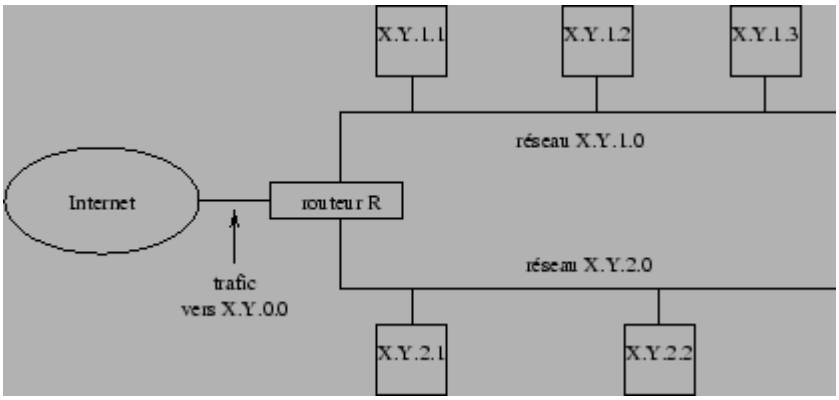

Figure: Adressage de sous-réseau

La figure illustre le cas d'un réseau `X.Y.0.0` découpé en deux sous-réseaux `X.Y.1.0` et `X.Y.2.0`. Pour tout le reste d'Internet, il n'existe qu'un seul réseau `X.Y.0.0` et tous les routeurs traitent les datagrammes à destination de ce réseau de la même façon. Par contre, le routeur R se sert du troisième octet (égal à 1 ou 2) de l'adresse contenue dans les datagrammes qui lui proviennent pour les diriger vers le sous-réseau auquel ils sont destinés assurant ainsi un routage hiérarchique.

Outre l'adresse IP, une machine doit également connaître le nombre de bits attribués à l'identificateur du sous-réseau et à celui de la machine. Cette information est rendue disponible grâce à un *masque de sous-réseau* ou *subnet netmask* qui est un mot de 32 bits contenant des bits à 1 au lieu et place de l'identificateur de réseau et de sous-réseau et des bits à 0 au lieu et place de l'identificateur de machines. Ainsi le masque[2.2] `255.255.255.0` indique que les 24 premiers bits d'une adresse désignent le sous-réseau et les 8 derniers une machine. Le masque `255.255.255.192` ($(192)_{10}=(11000000)_2$) indique que les 26 premiers bits désignent le sous réseau et les 6 derniers une machine. De cette manière à partir de l'adresse d'un datagramme et de son masque de sous-réseau une machine peut déterminer si le datagramme est destiné à une machine sur son propre sous-réseau, à une machine sur un autre sous-réseau de son réseau ou à une machine extérieure à son sous-réseau. Par exemple, dans le cadre du réseau de la figure où le masque de sous-réseau est `255.255.255.0` supposons que notre machine soit celle identifiée par l'adresse IP `X.Y.1.2`.

- Si l'adresse de destination est `X.Y.1.1`, un «et» entre la représentation binaire de cette adresse est de celle du masque de sous-réseau donne `X.Y.1.0` à savoir l'adresse du sous-réseau de notre machine, donc le datagramme est destiné à une machine de ce même sous-réseau.

- Si l'adresse de destination est `X.Y.2.1`, un calcul du même genre donne `X.Y.2.0` c'est-à-dire l'adresse d'un autre sous-réseau du même réseau.

- Si l'adresse de destination est `S.T.U.V` (avec $(S,T) \neq (X,Y)$) le résultat sera l'adresse d'un réseau différent de celui auquel appartient notre machine.

Bien que la numérotation IP à l'aide d'adresses numériques soit suffisante techniquement, il est préférable pour un humain de désigner une machine par un nom. Mais se pose alors le problème de la définition des noms et de leur mise en correspondance avec les numéros IP. Au début des années 80, le réseau ARPANET comportait un peu plus de 200 ordinateurs et chacun possédait un fichier `/etc/hosts` identifiant les noms de ces ordinateurs suivis de leur numéro IP. Lorsqu'une modification intervenait, il suffisait de mettre à jour ce fichier. Pour faire face à l'explosion du nombre d'ordinateurs reliés à Internet, il a été mis en place un système de base de données distribuées : le *système de noms de domaines* (DNS : *Domain Name System*) qui fournit la correspondance entre un nom de machine et son numéro IP.

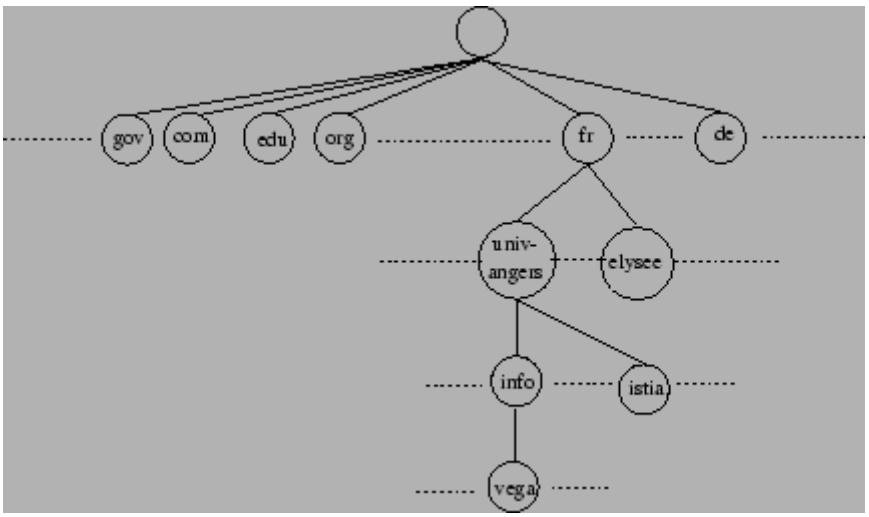

Figure: Système de noms de domaines.

En fait, le DNS est un espace de noms hiérarchisé comme illustré dans la figure 2.9. Chaque nœud a un nom d'au plus 63 caractères et la racine de l'arbre a un nom nul (les minuscules et majuscules sont indifférenciées). Une *zone* est un sous-arbre de cette hiérarchie. Le *nom de domaine* d'un nœud est la concaténation de son nom avec celui de ses ancêtres dans l'arbre. La responsabilité du nommage est subdivisée par niveau, les niveaux supérieurs déléguant leur autorité aux sous-domaines qu'ils créent eux-mêmes. Il faut bien avoir à l'esprit que le découpage n'a dans certains cas aucune base géographique ; on trouve des domaines `.com` partout dans le monde.

Le mécanisme qui permet la *résolution* d'un nom en une adresse IP est géré par des *serveurs de noms* qui représentent une base de données distribuée des noms de domaine. Quand une personne a reçu l'autorité de gérer une zone elle doit maintenir au moins deux serveurs de noms : un *primaire* et un ou plusieurs *secondaires*. Les secondaires ont des serveurs redondants par rapport au primaire de manière à faire face à une défaillance d'un système. Lorsqu'une machine est ajoutée à une zone, l'administrateur de la zone doit ajouter son nom et son numéro IP dans le fichier disque du serveur primaire qui se reconfigure alors en fonction de ces nouvelles données. Quant à eux, les serveurs secondaires interrogent régulièrement (toutes les 3 h) le primaire et fait les mises à jour nécessaires en cas d'évolution de la base de données. Lorsqu'un serveur de noms reçoit une demande, il vérifie si le nom appartient à l'un des sous-domaines qu'il gère. Si c'est le cas il traduit le nom en une adresse en fonction de sa base de données et renvoie la réponse au demandeur. Sinon, il s'adresse à un serveur de nom racine qui connaît le nom et l'adresse IP de chaque serveur de noms pour les domaines de second niveau. Ce serveur de nom racine lui renvoie alors l'adresse d'un serveur de noms à contacter, et ainsi de suite, par interrogations successives de serveurs de noms il sera capable de fournir l'adresse demandée. Pour éviter de faire trop souvent de telles requêtes, tout serveur de noms stocke dans une mémoire cache les correspondances

(numéro IP, nom de machine) de manière à pouvoir fournir la réponse immédiatement si une même demande lui parvient ultérieurement.

2.4 La couche liaison d'Internet.

Le but de la couche de liens de la pile TCP/IP est d'envoyer et recevoir des datagrammes IP pour la couche IP, d'envoyer des requêtes ARP (respt. RARP) et de recevoir des réponses pour le module ARP (respt. RARP). Nous examinons ici les caractéristiques des deux premières couches du modèle OSI (couches physique et de liens) dans le cas d'un réseau local Ethernet et d'une liaison série reliant un ordinateur à Internet via un modem connecté sur le port série de cet ordinateur.

Sous-sections

2.4.1 Le réseau Ethernet

Ethernet est le nom donné à une des technologies les plus utilisées pour les réseaux locaux en bus. Elle a été inventée par Xerox au début des années 70 et normalisée par l'IEEE (*Institute for Electrical and Electronics Engineers)* vers 1980 sous la norme IEEE 802.

Tout d'abord, il existe plusieurs technologies physiques pour établir un réseau Ethernet.

- *10 base 5 ou thick Ethernet* est un réseau à base de câble coaxial de 1,27 cm de diamètre, d'une longueur de 500 m maximum et terminé à chaque extrémité par une résistance. Chaque ordinateur est relié, par un cordon AUI *Attachment Unit Interface*), à un boîtier appelé *transceiver* lui-même connecté au câble par l'intermédiaire d'une prise « vampire ». Le transceiver est capable de détecter si des signaux numériques transitent sur le câble et de les traduire en signaux numériques à destination de l'ordinateur, et inversement.

- *10 base 2 ou thin Ethernet* est un réseau à base d'un câble coaxial plus fin et plus souple, moins résistant aux perturbations électromagnétiques que le 10 base 5, mais d'un coût inférieur. Le transceiver et le câble AUI ne sont plus utiles car l'ordinateur est relié directement au câble par l'intermédiaire d'une prise BNC en T intégrée à la carte Ethernet de l'ordinateur.

- *10 base T ou twisted pair Ethernet* est un réseau dans lequel chaque ordinateur est relié, par un câble de type paire torsadée, à un point central appelé *hub* qui simule l'effet d'un transceiver et de son câble AUI. La connexion des câbles se fait par l'intermédiaire d'une prise RJ45 et les hubs doivent être alimentés électriquement. Ils simulent ainsi le fonctionnement d'un bus alors que la topologie physique du réseau est une étoile.

Majoritairement, les réseaux Ethernet ont un débit de 10Mbit/s[2,3] et les informations sont transmises sur le bus sans garantie de remise. Chaque transceiver capte toutes les trames qui sont émises sur le câble et les redirige vers le contrôleur de l'ordinateur qui rejettera les trames qui ne lui sont pas destinées et enverra au processeur celles qui le concernent, c'est-à-dire celles dont l'adresse de destination est égale à celle de la carte réseau. Comme il n'y a pas d'autorité centrale qui gère l'accès au câble, il est possible que plusieurs stations veuillent émettre simultanément sur le câble. C'est pourquoi chaque transceiver écoute le câble pendant qu'il émet des données afin de détecter des éventuelles perturbations. Si une collision est détectée par le transceiver, celui-ci prévient le coupleur qui arrête d'émettre et attend un laps de temps aléatoire compris entre 0 et une certaine durée δ avant de réémettre ses données. S'il y a encore un problème de collision, alors un nouveau temps d'attente est tiré au sort entre 0 et $2 * \delta$, puis entre 0 et $4 * \delta$, etc... jusqu'à ce que la trame soit émise. Ce principe est justifié par le fait que si une première collision se produit, il y a de fortes chances que les délais d'attente tirés au sort par chacune des 2 stations soient très proches, donc il ne sera pas surprenant d'avoir une nouvelle collision. En doublant à chaque fois l'intervalle des délais d'attente possibles on augmente les chances de voir les retransmissions s'étaler sur des durées relativement longues et donc de diminuer les risques de collision. Cette technologie s'appelle CSMA/CD (*Carrier Sense Multiple Access with Collision Detect)*. Elle est efficace en générale mais a le défaut de ne pas garantir un délai de transmission maximal après lequel on est sûr que la trame a été émise, donc cela ne permet pas de l'envisager pour des applications temps réel.

Les adresses physiques Ethernet sont codées sur 6 octets (48 bits) et sont censées être uniques car les constructeurs et l'IEEE gère cet adressage de manière à ce que deux coupleurs ne portent pas la même adresse[2,4]. Elles sont de trois types

- *unicast* dans le cas d'une adresse monodestinataire désignant un seul coupleur
- *broadcast* dans le cas d'une adresse de diffusion générale (tous les bits à 1) qui permet d'envoyer une trame à toutes les stations du réseau
- *multicast* dans le cas d'une adresse multidestinataire qui permet d'adresser une même trame à un ensemble de stations qui ont convenu de faire partie du groupe que représente cette adresse multipoint.

On voit donc qu'un coupleur doit être capable de reconnaitre sa propre adresse physique, l'adresse de multicast, et toute adresse de groupe dont il fait partie.

Au niveau des trames, la normalisation IEEE 802[2,5]définit un format de trame légèrement différent de celui du véritable Ethernet. Ainsi, le RFC 894 définit les trames Ethernet et le RFC 1042 définit celles des réseaux IEE 802 comme illustré dans la figure 2.10.

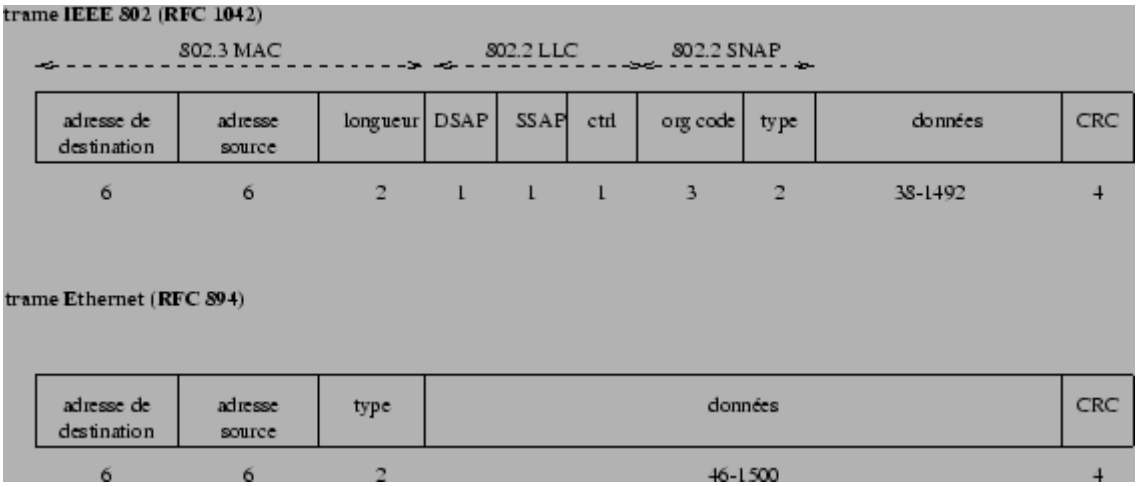

Figure 2.10: Encapsulation Ethernet et IEEE 802.3.

Mais la variante la plus usitée est l'Ethernet.

Les deux trames utilisent des adresses matérielles source et destination de 6 octets (adresse Ethernet) et un CRC de 4 octets mais différent sur les points suivants.

-

Dans le format Ethernet le troisième champ contient le *type* de données transmises selon que c'est un datagramme IP, une requête ou réponse ARP ou RARP. Puis, viennent les données transmises qui peuvent avoir une taille allant de 46 à 1500 octets. Dans le cas de données trop petites, comme pour les requêtes et réponse ARP et RARP (voir la sous-section 2.4.4) on complète avec des *bits de bourrage* ou *padding*.

-

Dans le format IEEE 802, le troisième champ indique le nombre d'octets de la trame sans compter le CRC. Étant donné qu'aucune des valeurs possibles pour le champ type de la trame Ethernet ne peut représenter une longueur de trame, ce champ peut permettre de distinguer les encapsulations. Pour la sous-couche LLC le champ DSAP (*Destination Service Access Point*) désigne le ou les protocoles de niveau supérieur à qui sont destinées les données de la trame et le champ SSAP (*Source Service Access Point*) désigne le protocole qui a émis la trame. Ici leur valeur hexadécimale est `AA`, c'est-à-dire la valeur désignant le protocole SNAP (*Sub-Network Access Protocol*). Le champ de contrôle *ctrl* est mis égal à 3 et les 3 octets du champ *org code* sont mis à 0. Ensuite, on trouve le champ *type* qui a la même signification que celui de la trame Ethernet.

De nombreux équipements matériels interviennent dans la constitution physique d'un réseau Ethernet, ce paragraphe décrit quelques uns de ceux qui interviennent aux niveaux 1 et 2 du modèle OSI.

- Un *répéteur* opére de manière physique uniquement, donc au niveau de la couche 1 du modèle OSI. Il se contente de retransmettre et d'amplifier tous les signaux qu'il reçoit, sans aucun autre traitement. Un « hub » est un répéteur 10 base T multiport qui renvoie donc le signal qu'il reçoit par l'un de ses ports vers tous ses autres ports.

- Un *pont* est un équipement qui intervient dans l'architecture d'un réseau en reliant deux segments disjoints de ce réseau. Le pont appartient à la couche 2 du modèle OSI car il va filtrer les trames du réseau en fonction de leur origine et destination, mais il ne se préoccuppe pas du logiciel réseau de niveau supérieur (TCP/IP, DECNet, IPX, ...).

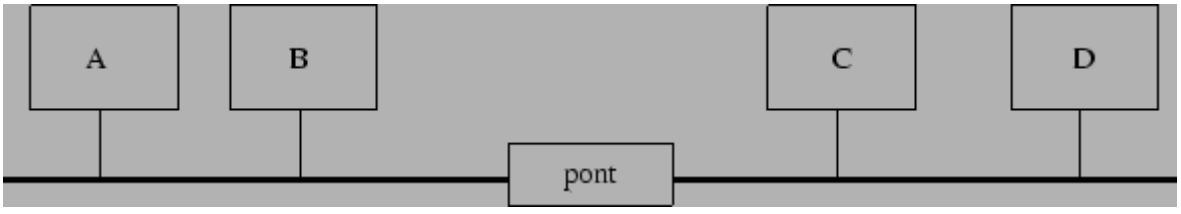

Figure 2.11: Fonctionnement d'un pont..

- Dans la configuration de la figure 2.11 le pont sera capable de déterminer que les ordinateurs A et B sont sur le segment 1 et les ordinateurs C et D sur le segment 2. Il peut obtenir ces informations car il «

voit passer » toutes les trames provenant des ordinateurs appartenant aux deux segments qu'il relie et grâce aux adresses d'origine contenues dans les trames, il peut se construire une table d'adresses mémorisant la cartographie du réseau. Ainsi, si une trame est envoyée de A vers B, ou de C vers D, elle ne franchira pas le pont car celui-ci aura détecté que c'est inutile. Mais si la trame provenant de A est destinée à C ou D, elle le traversera sans aucun autre traitement.

- L'utilisation d'un pont peut ainsi améliorer le débit d'un réseau car toutes les trames ne sont pas transmises sur tout le réseau. D'autre part, cela peut permettre d'augmenter la confidentialité du réseau en isolant certains ordinateurs des autres de manière à ce que certaines trames soient impossibles à capturer par des ordinateurs « espions » collectionnant toutes les trames qui circulent sur le réseau, même celles qui ne lui sont pas destinées.

- Un *commutateur* est en fait un pont multiport qui va aiguiller chacune des trames qu'il reçoit vers le segment sur lequel se trouve l'ordinateur de destination de la trame. Cependant, chacun de ses ports est habituellement relié à un segment contenant un nombre restreint d'ordinateurs, voire à un seul s'il s'agit par exemple d'un serveur très sollicité.

2.4.2 La liaison SLIP

SLIP (*Serial Link Internet Protocol*, RFC 1055) est un protocole permettant d'envoyer des paquets IP entre deux ordinateurs reliés par une liaison série (par exemple, grâce à deux modems branchés sur les ports RS-232 et une ligne téléphonique). Dans ce cas il n'y a pas besoin de prévoir un adressage de niveau 2, puisque la liaison est point à point (une seule machine à chaque extrémité du lien). Par contre, il s'agit de délimiter le début et la fin des paquets IP. L'encapsulation d'un paquet IP avant de l'envoyer sur la ligne consiste simplement à le faire terminer par le caractère spécial `END` (Oxc0) comme illustré dans la figure 2.12.

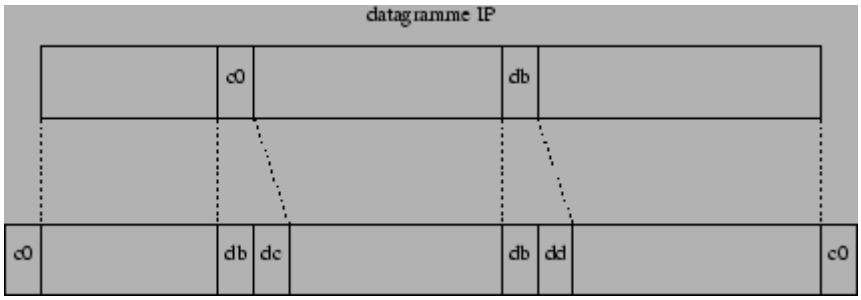

Figure 2.12: Encapsulation SLIP.

Pour éviter des problèmes de bruit, certaines implantations de SLIP font également débuter l'envoi du paquet IP par un caractère `END`. Pour qu'un caractère `END` faisant partie des données du paquet IP ne soit pas interprété comme la fin du paquet, l'émetteur le remplace par la séquence d'échappement `SLIP_ESC ESC_END` (0xdb 0xdc). Si le caractère `SLIP_ESC` fait partie des données à transmettre, alors la séquence `SLIP_ESC ESC_ESC` (0xdb 0xdd) est transmise à sa place.

Un des défauts de ce protocole est qu'il faut que les deux extrémités aient fixé préalablement leurs adresses IP, car la liaison SLIP ne leur permet pas de se les échanger. Si un site offre via un seul modem l'accès à Internet à plusieurs personnes, cela ne posera pas de problème. En effet, chaque personne aura configuré son ordinateur avec le numéro IP fournit par l'administrateur du réseau et comme une seule connexion est possible à la fois la duplication du même numéro IP n'est pas gênante. Seulement, si le site offre un deuxième modem sur le même numéro téléphonique, les utilisateurs ignoreront à quel modem ils sont connectés. À ce moment là, il faudra que le système indique à chaque utilisateur comment configurer son ordinateur en fonction de l'utilisation ou non de l'autre modem de telle manière que la même adresse IP ne soit pas donnée à deux personnes différentes simultanément. Dans ce genre d'utilisation SLIP a le défaut de ne pas offrir d'accès contrôler par mot de passe. De plus, il n'y a pas de champ type donc la ligne ne peut pas être utilisée en même temps pour un autre protocole. Et enfin, il n'y a pas de contrôle de la transmission. Si une trame subit des perturbations, c'est aux couches supérieures de de le détecter. Malgré tout, SLIP est un protocole largement utilisé et existe aussi dans une version améliorée CSLIP (*Compressed SLIP*).

2.4.3 La liaison PPP

PPP (*Point to Point Protocol*) (RFC 1661) est un protocole qui corrige les déficiences de SLIP en offrant les fonctionnalités suivantes.

- utilisation sur des liaisons point à point autres que série, comme X25 ou RNIS
- le transport de protocoles de niveau 3 (IP, Decnet, Appletalk, ...)
- la compression des en-têtes IP et TCP pour augmenter le débit de la liaison
- gestion d'un contrôle d'accès au réseau par authentification selon le protocole PAP qui nécessite la donnée d'un mot de passe au début de la communication ou le protocole CHAP qui permet l'échange de sceaux cryptés tout au long de la communication
- détection et correction d'erreurs de transmission
- ne pas utiliser des codes qui risquent d'être interprétés par les modems
- configuration automatique de la station client selon ses protocoles de couche réseau (IP, IPX, Appletalk).

Le protocole PPP est celui classiquement utilisé par les fournisseurs d'accès à Internet pour connecter leurs abonnés selon le schéma de la figure 2.13.

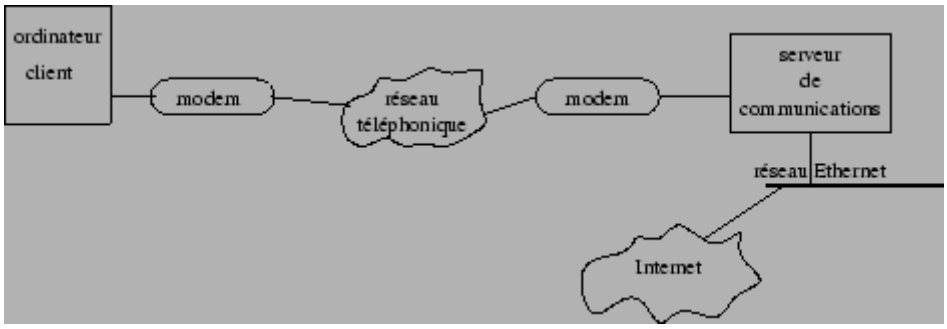

Figure: Connexion à Internet par modem et PPP.

Le processus de connexion d'un client équipé d'un ordinateur sous Windows, MacOS, Linux ou autre est le suivant.

- Le modem du client appelle le numéro de téléphone du fournisseur et la connexion téléphonique s'établit si l'un au moins de ses modems est libre.
- L'identification du client se fait par envoi d'un nom d'utilisateur et d'un mot de passe soit directement par l'utilisateur, soit selon l'un des protocoles PAP ou CHAP. Pour PAP (*Protocol Authentification Protocol*) le serveur de communication envoie à l'ordinateur un paquet pour demander le nom d'utilisateur et le mot de passe et l'ordinateur renvoie ces informations directement. CHAP (*Challenge Handshake Authentification Protocol*) fonctionne de la même manière sauf que le serveur de communication envoie d'abord une clef qui va permettre de crypter l'envoi du nom d'utilisateur et du mot de passe.
- Une fois l'identification du client controlée, le serveur de communication envoie une adresse IP, dite *dynamique* car elle varie selon les connexions, à l'ordinateur du client qui à partir de là se retrouve intégré au réseau Internt avec une adresse IP pour tout le temps que durera sa connexion.

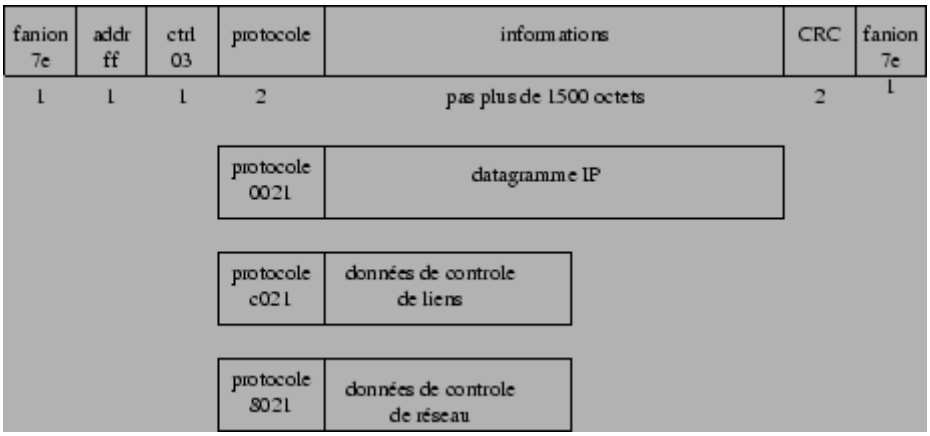

Figure 2.14: Encapsulation PPP.

De manière plus technique l'encapsulation PPP illustrée dans la figure 2.14 est proche du standard HDLC de l'ISO (voir section 1.4.2) et est telle que chaque trame commence et finit par un fanion de valeur `0x7e` soit en binaire `01111110`. La valeur du champ adresse est toujours fixée à `0xff` puisqu'elle est inutile ici dans le cas d'une liaison point à point. Le champ contrôle est fixé à `0x03`. Le champ protocol a le même rôle que la champ type de la trame Ethernet. Le CRC assure la détection des erreurs de transmission.

Le problème de l'apparition du fanion `01111110` au milieu des données à transmettre est réglé des deux manières suivantes.

- Dans le cas d'une liaison synchrone, à l'émission un bit à `0` est systématiquement ajouté après 5 `1` et il est retiré à la réception.
- Dans le cas d'une liaison asynchrone, le fanion `0x7e` est remplacé par la suite `0x7d 0x5e`, et le code `0x7d` est lui-même remplacé par la suite `0x7d 0x5d`. De plus tout octet O de valeur inférieure à `0x20` (32 en décimal), correspondant donc à un code de contrôle ASCII, sera remplacé par la séquence `0x7d O'` où $O' = O \oplus 0x20$. Ainsi, on est sûr que ces caractères ne seront pas interprétés par les modems comme des caractères de commandes. Par défaut, les 32 valeurs sont traitées ainsi mais il est possible d'utiliser le protocole de contrôle de liens pour spécifier pour quels caractères uniquement on fait cette transformation.

2.4.4 Les protocoles ARP et RARP

Étant donné que le protocole IP, et ses adresses, peuvent être utilisés sur des architectures matérielles différentes (réseau Ethernet, Token-Ring, ...) possédant leur propres adresses physiques, il y a nécessité d'établir les correspondances biunivoques entre adresses IP et adresses matérielles des ordinateurs d'un réseau. Ceci est l'objet des protocoles ARP (*Address Resolution Protocol*) et RARP (*reverse Address Resolution Protocol*). ARP fournit une correspondance dynamique entre une adresse IP connue et l'adresse matérielle lui correspondant, RARP faisant l'inverse.

Nous nous plaçons dans le cas d'une correspondance à établir entre IP et Ethernet et la nécessité de la résolution d'adresse fournie par ARP apparaît dans l'exemple ci-dessous décrivant le début d'une connexion FTP.

1. Le client FTP convertit l'adresse du serveur FTP (ex : `vega.univ-angers.fr`) en une adresse IP (`193.49.162.1`) à l'aide du fichiers `/etc/hosts` ou d'un serveur de noms (DNS).

2. Le client FTP demande à la couche TCP d'établir une connexion avec cette adresse.

3. TCP envoie une requête de connexion à ce serveur en émettant un datagramme IP contenant l'adresse IP

4. En supposant que les machines client et serveur sont sur le même réseau local Ethernet, la machine émettrice doit convertir l'adresse IP sur 4 octets en une adresse Ethernet sur 6 octets avant d'émettre la trame Ethernet contenant le paquet IP. C'est ce que va faire ARP.

5. Le module ARP envoie une requête ARP dans une trame Ethernet (donnée dans la figure 2.15) avec une adresse de destination multicast. Ainsi, toutes les machines du réseau local reçoivent cette requête contenant l'adresse IP à résoudre.

6. La couche ARP de la machine visée (ici `vega.univ-angers.fr`) reconnaît que cette requête lui est destinée et répond par une réponse ARP contenant son adresse matérielle `00:20:AF:AB:42:43`. Les autres machines du réseau ignorent la requête.

7. La réponse ARP est reçue par l'émetteur de la requête. Pour ce retour, il n'y a pas de problème de résolution puisque l'adresse physique de l'émetteur étant envoyée dans la requête elle est connue de la machine qui répond.

8. La réponse ARP est reçue par la couche ARP du client FTP, et le driver Ethernet peut alors émettre le paquet IP avec la bonne adresse Ethernet de destination.

Figure: Requête ou réponse ARP sur un réseau Ethernet.

Les deux premiers champs d'une trame Ethernet (voir figure 2.15) émise par ARP sont conformes à l'en-tête d'une trame Ethernet habituelle et l'adresse de destination sera `ff:ff:ff:ff:ff:ff`, l'adresse multicast désignant toutes les machines du réseau à la fois. La valeur du champ type de trame est `0x0806` indiquant le protocole ARP. Le champ type de matériel est égal à 1 pour un réseau Ethernet et et celui type de protocole est égal est `0x800` pour IP. Les tailles en octets spécifiées ensuite sont 6 (6 octets pour une adresse Ethernet) et 4 (4 octets pour une adresse IP). Le champ op vaut `1` pour une requête ARP et `2` pour une réponse ARP. Les quatre champs suivants contiennent des adresse et sont redondants dans le cas de l'adresse Ethernet émetteur d'une requête ARP, et non renseignés dans le cas de l'adresse Ethernet cible d'une requête ARP.

La machine qui reconnaît son numéro IP à l'intérieur d'une requête ARP qu'elle reçoit la renvoie en y intervertissant les adresses IP cible et émetteur, ainsi que les adresses Ethernet cible (après l'avoir substituée à l'adresse de diffusion dans l'en-tête et renseignée dans le corps de la trame) et Ethernet émetteur. Pour éviter la multiplication des requêtes ARP, chaque machine gère un cache dans lequel elle mémorise les correspondances adresses IP/adresses Ethernet déjà résolues préalablement. Ainsi, le module ARP ne lancera une requête que lorsqu'il ne trouvera pas cette correspondance dans le cache, sinon il se contentera d'émettre les données qu'il reçoit d'IP en ayant fixer correctement l'adresse physique de destination. Cependant, les correspondances ne sont pas conservées indéfiniment car cela pourrait provoquer des erreurs lorsque l'on change un ordinateur (ou une

carte réseau) sur le réseau en conservant un même numéro IP pour cet ordinateur mais évidemment pas la même adresse physique.

Quant à lui, le protocole RARP joue le rôle inverse de ARP en permettant de déterminer l'adresse IP d'un équipement dont on connaît l'adresse physique. Ceci est notamment utile pour amorcer une station sans disques, ou un TX, qui n'a pas en mémoire son adresse IP mais seulement son adresse matérielle. Le format d'une trame RARP est celui de la figure 2.15) où le champ type de trame vaut `0x0835` et le champ op vaut `3` pour une requête RARP et `4` pour une réponse. Une requête RARP est diffusée sous forme de broadcast, donc toutes les machines du réseau la reçoivent et la traitent. Mais la plupart des machines ignorent simplement cette demande, seuls, le ou les serveurs RARP du réseau vont traiter la requête grâce à un ou plusieurs fichiers et vont retourner une réponse contenant l'adresse IP demandée.

2.5 Le protocole IP.

Comme on a pu le voir dans la figure 2.3 le protocole IP (*Internet Protocol*, RFC 791) est au cœur du fonctionnement d'un internet. Il assure *sans connexion* un service *non fiable* de délivrance de datagrammes IP. Le service est non fiable car il n'existe aucune garantie pour que les datagrammes IP arrivent à destination. Certains peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre. On parle de remise au mieux (*best effort delivery*) et ni l'émetteur ni le récepteur ne sont informés directement par IP des problèmes rencontrés. Le mode de transmission est non connecté car IP traite chaque datagramme indépendamment de ceux qui le précèdent et le suivent. Ainsi en théorie, au moins, deux datagrammes IP issus de la même machine et ayant la même destination peuvent ne pas suivre obligatoirement le même chemin. Le rôle du protocole IP est centré autour des trois fonctionnalités suivantes chacune étant décrite dans une des sous-sections à venir.

- définir le format du datagramme IP qui est l'unité de base des données circulant sur Internet
- définir le routage dans Internet
- définir la gestion de la remise non fiable des datagrammes

Sous-sections

2.5.1 Le datagramme IP.

Comme cela a déjà été illustré dans la figure 2.6 on rappelle qu'un datagramme IP est constitué d'une en-tête suivie d'un champ de données.

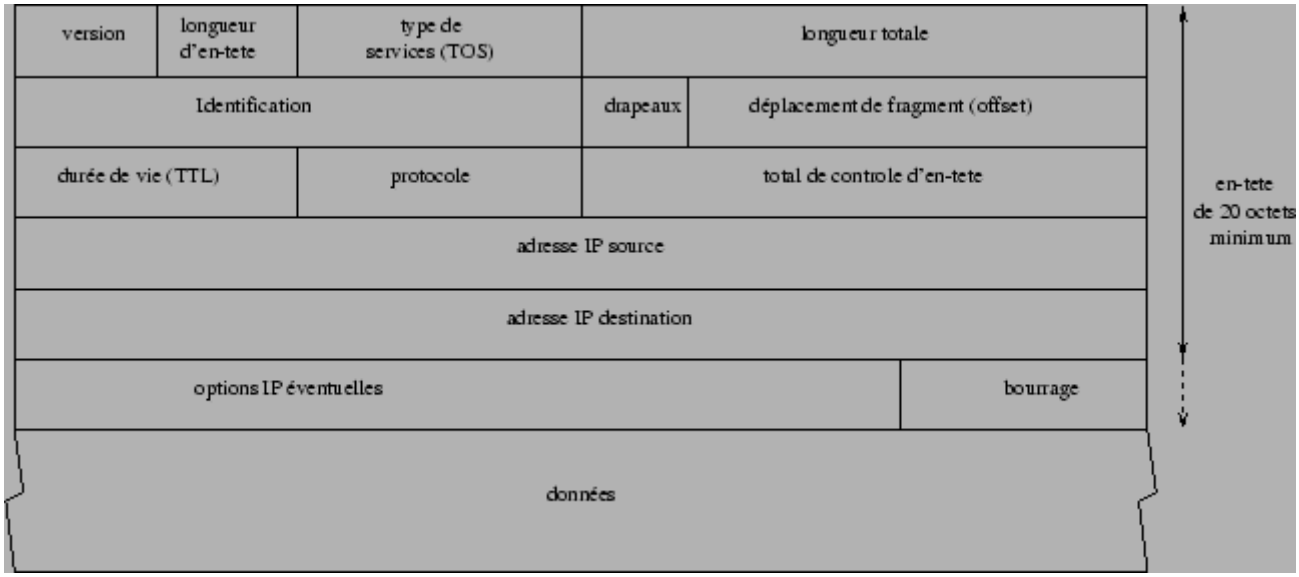

Figure 2.16: Structure d'un datagramme IP.

Sa structure précise est détaillée dans la figure 2.16 et comporte les champs suivants.

- La *version* code sur 4 bits le numéro de version du protocole IP utilisé (la version courante est la 4, d'où son nom d'IPv4). Tout logiciel IP doit d'abord vérifier que le numéro de version du datagramme qu'il reçoit est en accord avec lui-même. si ce n'est pas le cas le datagramme est tout simplement rejeté. Ceci permet de tester des nouveaux protocoles sans interférer avec la bonne marche du réseau.

- La *longueur d'en-tête* représente sur 4 bits la longueur, en nombre de mots de 32 bits, de l'en-tête du datagramme. Ce champ est nécessaire car une en-tête peut avoir une taille supérieure à 20 octets (taille de l'en-tête classique) à cause des options que l'on peut y ajouter.

- Le *type de services (TOS)* est codé sur 8 bits, indique la manière dont doit être géré le datagramme et se décompose en six sous-champs comme suit.

0	1	2	3	4	5	6	7
priorité			D	T	R	C	inutilisé

Le champ *priorité* varie de 0 (priorité normale, valeur par défaut) à 7 (priorité maximale pour la supervision du réseau) et permet d'indiquer l'importance de chaque datagramme. Même si ce champ n'est pas pris en compte par tous les routeurs, il permettrait d'envisager des méthodes de contrôle de congestion du réseau qui ne soient pas affectées par le problème qu'elles cherchent à résoudre. Les 4 bits D, T, R et C permettent de spécifier ce que l'on veut privilégier pour la transmission de ce datagramme (nouvel RFC 1455). D est mis à 1 pour essayer de minimiser le délai d'acheminement (par exemple choisir un câble sous-marin plutôt qu'une liaison satellite), T est mis à 1 pour maximiser le débit de transmission, R est mis à 1 pour assurer une plus grande fiabilité et C est mis à 1 pour minimiser les coûts de transmission. Si les quatre bits sont à 1, alors c'est la sécurité de la transmission qui doit être maximisée. Les valeurs recommandées pour ces 4 bits sont données dans la table 2.1.

Tableau 2.1: Type de service pour les applications standard.

application	minimise le délai	maximise le débit	maximise la fiabilité	minimise le coût
telnet/rlogin	1	0	0	0
FTP				
contrôle	1	0	0	0

transfert	0	1	0	0
SMTP				
commandes	1	0	0	0
données	0	1	0	0
NNTP	0	0	0	1
SNMP	0	0	1	0

Ces 4 bits servent à améliorer la qualité du routage et ne sont pas des exigences incontournables. Simplement, si un routeur connaît plusieurs voies de sortie pour une même destination il pourra choisir celle qui correspond le mieux à la demande.

- La *longueur totale* contient la taille totale en octets du datagramme, et comme ce champ est de 2 octets on en déduit que la taille complète d'un datagramme ne peut dépasser 65535 octets. Utilisée avec la longueur de l'en-tête elle permet de déterminer où commencent exactement les données transportées.
- Les champs *identification*, *drapeaux* et *déplacement de fragment* interviennent dans le processus de fragmentation des datagrammes IP et sont décrits dans la sous-section 2.5.2.
- La *durée de vie* (TTL) indique le nombre maximal de routeurs que peut traverser le datagramme. Elle est initialisée à *N* (souvent 32 ou 64) par la station émettrice et décrémenté de 1 par chaque routeur qui le reçoit et le réexpédie. Lorsqu'un routeur reçoit un datagramme dont la durée de vie est nulle, il le détruit et envoie à l'expéditeur un message ICMP. Ainsi, il est impossible qu'un datagramme «tourne» indéfiniment dans un internet. Ce champ sert également dans la réalisation du programme `traceroute` donné dans la section 2.8.2.
- Le *protocole* permet de coder quel protocole de plus haut niveau a servi a créé ce datagramme. Les valeurs codées sur 8 bits sont 1 pour ICMP, 2 pour IGMP, 6 pour TCP et 17 pour UDP. Ainsi, la station destinatrice qui reçoit un datagramme IP pourra diriger les données qu'il contient vers le protocole adéquat.
- Le *total de contrôle d'en-tête* (header checksum) est calculé à partir de l'en-tête du datagramme pour en assurer l'intégrité. L'intégrité des données transportées est elle assurée directement par les protocoles ICMP, IGMP, TCP et UDP qui les émettent. Pour calculer cette somme de contrôle, on commence par la mettre à zéro. Puis, en considérant la totalité de l'en-tête comme une suite d'entiers de 16 bits, on fait la somme de ces entiers en complément à 1. On complémente à 1 cette somme et cela donne le total de contrôle que l'on insère dans le champ prévu. A la réception du datagramme, il suffit d'additionner tous les nombres de l'en-tête et si l'on obtient un nombre avec tous ses bits à 1, c'est que la transmission s'est passée sans problème.

  Exemple : Soit le datagramme IP dont l'en-tête est la suivante `4500 05dc e733 222b ff11 checksum c02c 4d60 c02c 4d01` La somme des mots de 16 bits en compléments à 1 donne `6e08`, son complément à 1 est `91f7`. Le datagramme est donc expédié avec cette valeur de checksum.
- Les *adresses IP source* et *destination* contiennent sur 32 bits les adresses de la machine émettrice et destinataire finale du datagramme.
- Le champ *options* est une liste de longueur variable, mais toujours complétée par des bits de *bourrage* pour atteindre une taille multiple de 32 bits pour être en conformité avec la convention qui définit le champ longueur de l'en-tête. Ces options sont très peu utilisées car peu de machines sont aptes à les gérer. Parmi elles, on trouve des options de sécurité et de gestion (domaine militaire), d'enregistrement de la route, d'estampille horaire, routage strict, etc...

Les champs non encore précisés le sont dans la section suivante car ils concernent la fragmentation des datagrammes.

2.5.2 La fragmentation des datagrammes IP.

En fait, il existe d'autres limites à la taille d'un datagramme que celle fixée par la valeur maximale de 65535 octets. Notamment, pour optimiser le débit il est préférable qu'un datagramme IP soit encapsulé dans une seule trame de niveau 2 (Ethernet par exemple). Mais, comme un datagramme IP peut transiter à travers Internet sur un ensemble de réseaux aux technologies différentes il est impossible de définir, a priori (lors de la définition du RFC), une taille maximale des datagrammes IP qui permette de les encapsuler dans une seule trame quel que soit le réseau (1500 octets pour Ethernet et 4470 pour FDDI par exemple). On appelle la taille maximale d'une trame d'un réseau le *MTU* (*Maximum Transfert Unit*) et elle va servir à fragmenter les datagrammes trop grands pour le réseau qu'ils traversent. Mais, si le MTU d'un réseau traversé est suffisamment grand pour accepter un datagramme, évidemment il sera encapsulé tel quel dans la trame du réseau concerné.

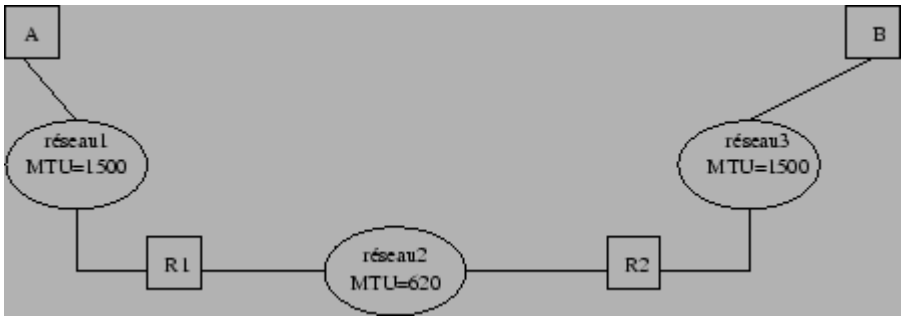

Figure 2.17: Fragmentation d'un datagramme IP.

Comme on peut le voir dans la figure 2.17 la fragmentation se situe au niveau d'un routeur qui reçoit des datagrammes issus d'un réseau à grand MTU et qui doit les réexpédier vers un réseau à plus petit MTU. Dans cet exemple, si la station A, reliée à un réseau Ethernet, envoie un datagramme de 1300 octets à destination de la station B, reliée également à un réseau Ethernet, le routeur R1 va devoir fragmenter ce datagramme de la manière suivante.

datagramme initial	en-tête du datagramme	données1 600 octets	données2 600 octets	données3 100 octets
fragment1	en-tête du fragment1	données1 600 octets	déplacement 0	
fragment2	en-tête du fragment2	données2 600 octets	déplacement 600	
fragment3	en-tête du fragment3	données3 100 octets	déplacement 1200	

La taille d'un fragment est choisie la plus grande possible tout en étant un multiple de 8 octets. Un datagramme fragmenté n'est réassemblé que lorsqu'il arrive à destination finale, même s'ils traversent des réseaux avec un plus grand MTU les routeurs ne réassemblent pas les petits fragments. De plus chaque fragment est routé de manière totalement indépendante des autres fragments du datagramme d'où il provient. Le destinataire final qui reçoit un premier fragment d'un datagramme arme un temporisateur de réassemblage, c'est-à-dire un délai maximal d'attente de tous les fragments. Si, passé ce délai, tous les fragments ne sont pas arrivés il détruit les fragments reçus et ne traite pas le datagramme. Plus précisément, l'ordinateur destinataire décrémente, à intervalles réguliers, de une unité le champ TTL de chaque fragment en attente de réassemblage. Cette technique permet également de ne pas faire coexister au même instant deux datagrammes avec le même identifiant.

Le processus de fragmentation-réassemblage est rendu possible grâce aux différents champs suivants. Le champ *déplacement de fragment* précise la localisation du début du fragment dans le datagramme initial. À part cela, les fragments sont des datagrammes dont l'en-tête est quasiment identique à celle du datagramme original. Par

exemple, le champ *identification* est un entier qui identifie de manière unique chaque datagramme émis et qui est recopié dans le champ identification de chacun des fragments si ce datagramme est fragmenté. Par contre, le champ longueur total est recalculé pour chaque fragment. Le champ *drapeaux* comprend trois bits dont deux qui contrôlent la fragmentation. S'il est positionné à 1 le premier bit indique que l'on ne doit pas fragmenter le datagramme et si un routeur doit fragmenter un tel datagramme alors il le rejette et envoie un message d'erreur à l'expéditeur. Un autre bit appelé *fragments à suivre* est mis systématiquement à 1 pour tous les fragments qui composent un datagramme sauf le dernier. Ainsi, quand le destinataire reçoit le fragment dont le bit fragment à suivre est à 0 il est apte à déterminer s'il a reçu tous les fragments du datagramme initial grâce notamment aux champs offset et longueur totale de ce dernier fragment. Si un fragment doit être à nouveau fragmenté lorsqu'il arrive sur un réseau avec un encore plus petit MTU, ceci est fait comme décrit précédemment sauf que le calcul du champ déplacement de fragment est fait en tenant compte du déplacement inscrit dans le fragment à traiter.

2.5.3 Le routage IP.

Le routage est l'une des fonctionnalités principales de la couche IP et consiste à choisir la manière de transmettre un datagramme IP à travers les divers réseaux d'un internet. On appellera *ordinateur* un équipement relié à un seul réseau et *routeur* un équipement relié à au moins deux réseaux (cet équipement pouvant être un ordinateur, au sens classique du terme, qui assure les fonctionnalités de routage). Ainsi un routeur réémettra des datagrammes venus d'une de ses interfaces vers une autre, alors qu'un ordinateur sera soit l'expéditeur initial, soit le destinataire final d'un datagramme. D'une manière générale on distingue la *remise directe*, qui correspond au transfert d'un datagramme entre deux ordinateurs du même réseau, et la *remise indirecte* qui est mise en œuvre dans tous les autres cas, c'est-à-dire quand au moins un routeur sépare l'expéditeur initial et le destinataire final.

Par exemple, dans le cas d'un réseau Ethernet, la remise directe consiste à encapsuler le datagramme dans une trame Ethernet après avoir utilisé le protocole ARP pour faire la correspondance adresse IP adresse physique (voir les sections 2.4.1 et 2.4.4) et à émettre cette trame sur le réseau. L'expéditeur peut savoir que le destinataire final partage le même réseau que lui-même en utilisant simplement l'adresse IP de destination du datagramme. Il en extrait l'identificateur de réseau et si c'est le même que celui de sa propre adresse IP[2.6]alors la remise directe est suffisante. En fait, ce mécanisme de remise directe se retrouve toujours lors de la remise d'un datagramme entre le dernier routeur et le destinataire final.

Pour sa part, la remise indirecte nécessite de déterminer vers quel routeur envoyer un datagramme IP en fonction de sa destination finale. Ceci est rendu possible par l'utilisation d'une *table de routage* spécifique à chaque routeur qui permet de déterminer vers quelle voie de sortie envoyer un datagramme destiné à un réseau quelconque. Évidemment, à cause de la structure localement arborescente d'Internet la plupart des tables de routage ne sont pas très grandes. Par contre, les tables des routeurs interconnectant les grands réseaux peuvent atteindre des tailles très grandes ralentissant d'autant le trafic sur ces réseaux. D'un point de vue fonctionnel une table de routage contient des paires d'adresses du type (*D, R*) où *D* est l'adresse IP d'une machine ou d'un réseau de destination et *R* l'adresse IP du routeur suivant sur la route menant à cette destination. Tous les routeurs mentionnés dans une table de routage doivent bien sûr être directement accessibles à partir du routeur considéré. Cette technique, dans laquelle un routeur ne connaît pas le chemin complet menant à une destination, mais simplement la première étape de ce chemin, est appelée *routage par sauts successifs* (*next-hop routing*). Une table de routage contient aussi une *route par défaut* qui spécifie un routeur par défaut vers lequel sont envoyés tous les datagrammes pour lesquels il n'existe pas de route dans la table.

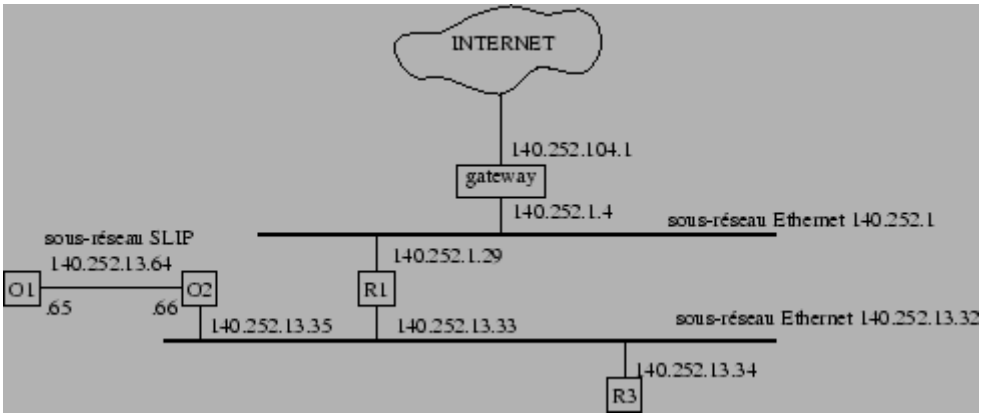

Figure: Exemple d'interconnexion de réseau.

La figure 2.18 donne un exemple d'interconnexion d'un réseau de classe B 140.252.0.0 subdivisé en 3 sous-réseaux de masques respectifs 255.255.255.0 pour l'Ethernet du haut, 255.255.255.224[2.7] pour l'Ethernet du bas et le sous-réseau SLIP. Les masques de sous-réseau sont associés par défaut à chaque interface d'une machine ou sont éventuellement spécifié sur chaque ligne des tables de routage. Dans l'exemple donné ici, la table de routage simple de l'ordinateur *R3* (sous système unix Sun) est la suivante.

destination	gateway	flags	refcnt	use	interface
140.252.13.65	140.252.13.35	UGH	0	0	emd0
127.0.0.1	127.0.0.1	UH	1	0	lo0
default	140.252.13.33	UG	0	0	emd0

140.252.13.32 140.252.13.34 U 4 25043 emdO

Les *flags* ont la signification suivante.

U
: La route est en service.

G
: La route est un routeur (*gateway*). Si ce flag n'est pas positionné la destination est directement connectée au routeur, c'est donc un cas de remise directe vers l'adresse IP de destination.

H
: La route est un ordinateur (*host*), la destination est une adresse d'ordinateur. Dans ce cas, la correspondance entre l'adresse de destination du paquet à «router» et l'entrée destination de la table de routage doit être totale. Si ce flag n'est pas positionné, la route désigne un autre réseau et la destination est une adresse de réseau ou de sous-réseau. Ici, la correspondance des identificateurs de réseaux est suffisante.

D
: La route a été créée par une redirection.

M
: La route a été modifiée par une redirection.

La colonne *compteur de référence* (*refcnt* indique le nombre de fois où la route est utilisée à l'instant de la consultation. Par exemple, TCP conserve la même route tant qu'il l'utilise pour la connexion sous-jacente à une application(telnet, ftp, ...). La colonne *use* affiche le nombre de paquets envoyés à travers l'interface de cette route qui est spécifiée dans la dernière colonne de la même ligne.

L'adresse 127.0.0.1 est celle de *lo0*, l'interface de *loopback*, qui sert à pouvoir faire communiquer une machine avec elle-même.

La destination *default* sert à indiquer la destination de tous les datagrammes qui ne peuvent être «routés» par l'une des autres routes.

L'utilisation d'une table de routage se fait suivant l'algorithme de routage IP donné ci-dessous.

```
Procédure RoutageIP(données Dat : datagramme, Tab : Table de routage)
début
 D := adresse IP de destination de Dat
 N := identificateur du réseau de D
 si N est une adresse de réseau directement accessible
 alors
  envoyer Dat vers l'adresse D sur ce réseau
     { il y a résolution de l'adresse IP, en adresse physique,
       encapsulation de Dat dans une trame physique et émission de la
trame}
 sinon
  pour chaque entrée de Tab faire
   N:=résultat du et logique de D et du masque de sous-réseau
   si N=l'adresse réseau ou sous-réseau de la destination de l'entrée
   alors
    router Dat vers cette destination
    sortir
   finsi
  finpour
  si aucune correspondance n'est trouvée
  alors
   retourner à l'application d'origine du datagramme une erreur de routage
   host unreachable ou unreachable network
  finsi
 finsi
fin
```

L'établissement d'une table de routage est *statique* lorsqu'elle résulte de la configuration par défaut d'une interface, ou de la commande `route` à partir d'un fichier de démarrage, ou grâce à une redirection ICMP (voir la section 2.5.4). Mais dès que le réseau devient non trivial, on utilise le routage *dynamique* qui consiste en un protocole de communication entre routeurs qui informent chacun de leurs voisins des réseaux auxquels ils sont connectés. Grâce à ce protocole, les tables de routage évoluent dans le temps en fonction de l'évolution des routes.

L'un des protocoles de routage les plus populaires est *RIP* (*Routing Information Protocol*) qui est un protocole de type *vecteur de distance*. C'est-à-dire que les messages échangés par des routeurs voisins contiennent un ensemble de distances entre routeur et destinations qui permet de réactualiser les tables de routage. Ce protocole utilise une métrique simple : la distance entre une source et une destination est égale au nombre de sauts qui les séparent. Elle est comprise entre 1 et 15, la valeur 16 représentant l'«infini». Ceci implique que RIP ne peut être utilisé qu'à l'intérieur de réseaux qui ne sont pas trop étendus.

Un message RIP est encapsulé dans un datagramme UDP de la manière décrite dans la figure 2.19.

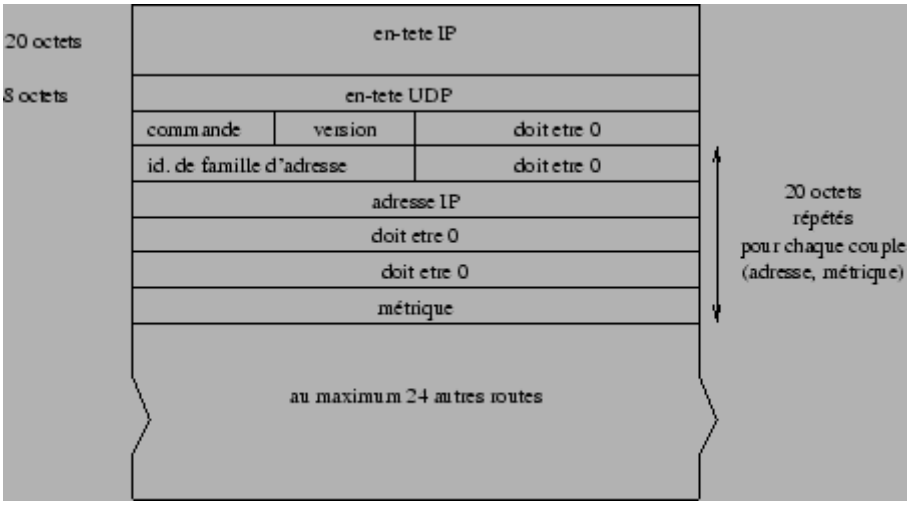

Figure 2.19: Encapsulation d'un message RIP.

Le champ *commande* fixé à 1 indique une requête pour demander tout ou partie d'une table de routage et fixé à 2 pour transmettre une réponse (d'autres valeurs hors normes actuellement existent également). Le champ *version* est positionné à 1 et à 2 dans la version de RIP2. Pour des adresses IP, le champ *identificateur de famille d'adresses* est toujours fixé à 2.

À l'initialisation, le démon de routage envoie une requête RIP à chaque interface pour demander les tables de routage complètes de chacun de ses voisins. Sur une liaison point à point la requête est envoyée à l'autre extrémité, sinon elle est envoyée sous forme de broadcast sur un réseau.

Le fonctionnement normal de RIP consiste à diffuser des réponses soit toutes les 30 secondes, soit pour une mise à jour déclenchée par la modification de la métrique d'une route. Une réponse contient une adresse de destination, accompagnée de sa métrique, de l'adresse du prochain routeur,d'un indicateur de mise à jour récente et de temporisations. Le processus RIP met à jour sa table de routage locale en examinant les entrées retournées dont il vérifie d'abord la validité : adresse de classe A,B ou C, numéro de réseau différent de 127 et 0 (sauf pour l'adresse par défaut 0.0.0.0), numéro d'ordinateur différent de l'adresse de diffusion, métrique différente de l'«infini». RIP effectue ensuite les mises à jour propres à l'algorithme «vecteur de distance» suivant.

- Si l'entrée n'existait pas dans la table et si la métrique reçue n'est pas infinie, alors on ajoute cette nouvelle entrée composée de la destination, de l'adresse du prochain routeur (c'est celui qui envoie la réponse), de la métrique reçue. On initialise la temporisation correspondante.
- Si l'entrée était présente avec une métrique supérieure à celle reçue, on met à jour la métrique et le prochain routeur et on réinitialise la temporisation.
- Si l'entrée était présente et que le routeur suivant correspond à l'émetteur de la réponse, on réinitialise la temporisation et on met à jour la métrique avec celle reçue si elles diffèrent.
- Dans les autres cas on ignore l'entrée.

Cette méthode correspond à l'algorithme de Bellman-Ford de recherche de plus courts chemins dans un graphe.

Un des défauts de RIP est de ne pas gérer les adresses de sous-réseaux. Mais de telles entrées peuvent être annoncées via le une interface appartenant à ce sous-réseau pour pouvoir bénéficier du masque qui y est attaché et être correctement interprétées. Enfin, RIP met un temps assez long (quelques minutes) pour se stabiliser après la défaillance d'une liaison ou d'un routeur ce qui peut occasionner des boucles de routage.

RIP2 est un protocole qui étend RIP en utilisant les 4 champs laissés à 0 par RIP dans ses messages. Le premier sert à fixer un *domaine de routage* identifiant le démon de routage qui a émis le paquet et le quatrième *l'adresse IP d'un routeur de saut suivant*. Ces deux champs ont servi à lancer simultanément plusieurs démons de routage sur un même support leur utilisation a été abandonnée. Le deuxième champ est un *identificateur de route* pour supporter des protocoles de routes externes et le troisième sert à spécifier un *masque de sous-réseau* pour chaque entrée de la réponse.

OSPF (*Open Shortest Path First*) est un nouveau type de protocole de routage dynamique qui élimine les limitations de RIP. C'est un protocole *d'état de liens*, c'est-à-dire qu'ici un routeur n'envoie pas des distances à ses voisins, mais il teste l'état de la connectivité qui le relie à chacun de ses voisins. Il envoie cette information à tous ses voisins, qui ensuite le propagent dans le réseau. Ainsi, chaque routeur peut posséder une carte de la topologie du réseau qui se met à jour très rapidement lui permettant de calculer des routes aussi précises qu'avec un algorithme centralisé.

En fait, RIP et OSPF, sont des protocoles de type *IGP* (*Interior Gateway Protocol*) permettant d'établir les tables des routeurs internes des *systèmes autonomes*. Un système autonome peut être défini par un ensemble de routeurs et de réseaux sous une administration unique. Cela peut donc aller d'un seul routeur connectant un réseau local à Internet, jusqu'à l'ensemble des réseaux locaux d'une multinationale. La règle de base étant qu'un système autonome assure la connexité totale de tous les points qui le composent en utilisant notamment un protocole de routage unique. À un niveau plus global, Internet apparaît donc comme une interconnexion de systèmes autonomes comme illustré dans la figure 2.20.

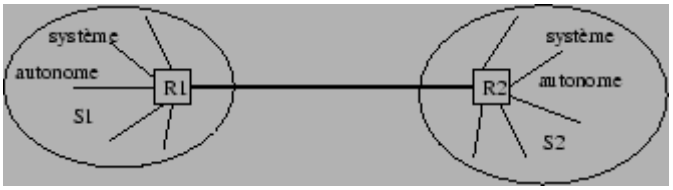

Figure: Interconnexion de systèmes autonomes.

Dans chaque système autonome les tables sont maintenus par par un IGP et sont échangées uniquement entre routeurs du même sous-système. Pour obtenir des informations sur les réseaux externes, ceux de l'autre système autonome, ils doivent dialoguer avec les les routeurs externes R1 et R2. Ceux-ci sont des points d'entrée de chaque système et, via la liaison qui les relie, ils échangent des informations sur la connectivité grâce à *EGP* (*Exterior Gateway Protocol*) ou *BGP* (*Border Gateway Protocol*) qui remplace EGP actuellement.

2.5.4 La gestion des erreurs.

Le protocole ICMP (*Internet Control Message Protocol*) organise un échange d'information permettant aux routeurs d'envoyer des messages d'erreurs à d'autres ordinateurs ou routeurs. Bien qu'ICMP «tourne» au-dessus de IP il est requis dans tous les routeurs c'est pourquoi on le place dans la couche IP. Le but d'ICMP n'est pas de fiabiliser le protocole IP, mais de fournir à une autre couche IP, ou à une couche supérieure de protocole (TCP ou UDP), le compte-rendu d'une erreur détectée dans un routeur. Un message ICMP étant acheminé à l'intérieur d'un datagramme IP[2.8]illustré dans la figure 2.21, il est susceptible, lui aussi, de souffrir d'erreurs de transmission.

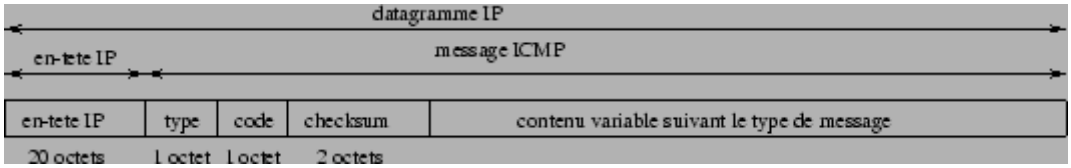

Figure 2.21: Encapsulation d'un message ICMP.

Mais la règle est qu'aucun message ICMP ne doit être délivré pour signaler une erreur relative à un message ICMP. On évite ainsi une avalanche de messages d'erreurs quand le fonctionnement d'un réseau se détériore.

Le champ *type* peut prendre 15 valeurs différentes spécifiant de quelle nature est le message envoyé. Pour certains types, le champ *code* sert à préciser encore plus le contexte d'émission du message. Le *checksum* est une somme de contrôle de tout le message ICMP calculée comme dans le cas de l'en-tête d'un datagramme IP (voir la section 2.5.1). Le détail des différentes catégories de messages est donné dans la liste ci-dessous où chaque alinéa commence par le couple (*type*,*code*) de la catégorie décrite.

(0,0) ou (8,0)
: Demande (type 8) ou réponse (type 0) d'écho dans le cadre de la commande `ping` (voir section 2.8.2).

(3,0-13)
: Compte-rendu de destination inaccessible délivré quand un routeur ne peut délivrer un datagramme. Le routeur génère et envoie ce message ICMP à l'expéditeur de ce datagramme. Il obtient l'adresse de cet expéditeur en l'extrayant de l'en-tête du datagramme, il insère dans les données du message ICMP toute l'en-tête ainsi que les 8 premiers octets du datagramme en cause. Une liste non exhaustive des différents codes d'erreurs possibles est :
  0
  Le réseau est inaccessible.
  1
  La machine est inaccessible.
  2
  Le protocole est inaccessible.
  3
  Le port est inaccessible.
  4
  Fragmentation nécessaire mais bit de non fragmentation positionné à 1.
  5
  Échec de routage de source.
  6
  Réseau de destination inconnu.
  7
  Machine destinataire inconnue.
  8
  Machine source isolée (obsolète)
  9
  Communication avec le réseau de destination administrativement interdite.
  10
  Communication avec la machine de destination administrativement interdite.
  11
  Réseau inaccessible pour ce type de service.
  12
  Machine inaccessible pour ce type de service.
  13
  Communication administrativement interdite par filtrage.

(4,0)

Demande de limitation de production pour éviter la congestion du routeur qui envoie ce message.

(5,0-3)
Demande de modification de route expédiée lorsqu'un routeur détecte qu'un ordinateur utilise une route non optimale, ce qui peut arriver lorsqu'un ordinateur est ajouté au réseau avec une table de routage minimale. Le message ICMP généré contient l'adresse IP du routeur à rajouter dans la table de routage de l'ordinateur. Les différents codes possibles ci-après expliquent le type de redirection à opérer par l'ordinateur.
0
Redirection pour un réseau.
1
Redirection pour une machine.
2
Redirection pour un type de service et réseau.
3
Redirection pour un type de service et machine.

(9,0)
Avertissement de routeur expédié par un routeur.

(10,0)
Sollicitation de routeur diffusé par une machine pour initialiser sa table de routage.

(11,0)
TTL détecté à 0 pendant le transit du datagramme IP, lorsqu'il y a une route circulaire ou lors de l'utilisation de la commande `traceroute` (voir section 2.8.2).

(11,1)
TTL détecté à 0 pendant le réassemblage d'un datagramme.

(12,0)
Mauvaise en-tête IP.

(12,1)
Option requise manquante.

(13-14,0)
Requête (13) ou réponse (14) *timestamp*, d'estampillage horaire.

(15,0) et (16,0)
devenues obsolètes.

(17-18,0)
Requête (17) ou réponse (18) de masque de sous-réseau.

2.6 Les protocoles TCP et UDP.

On présente ici les deux principaux protocoles de la couche transport d'Internet que sont les protocoles *TCP* (*Transmission Control Protocol*) et *UDP* (*User Datagram Protocol*). Tous les deux utilisent IP comme couche réseau, mais TCP procure une couche de transport fiable (alors même que IP ne l'est pas), tandis que UDP ne fait que transporter de manière non fiable des datagrammes.

Sous-sections

2.6.1 Le protocole UDP.

Le protocole UDP (rfc 768)utilise IP pour acheminer, d'un ordinateur à un autre, en mode non fiable des datagrammes qui lui sont transmis par une application (voir la figure 2.3). UDP n'utilise pas d'accusé de réception et ne peut donc pas garantir que les données ont bien été reçues. Il ne réordonne pas les messages si ceux-ci n'arrivent pas dans l'ordre dans lequel ils ont été émis et il n'assure pas non plus de contrôle de flux. Il se peut donc que le récepteur ne soit pas apte à faire face au flux de datagrammes qui lui arrivent. C'est donc à l'application qui utilise UDP de gérer les problèmes de perte de messages, duplications, retards, déséquencement, ...

Cependant, UDP fournit un service supplémentaire par rapport à IP, il permet de distinguer plusieurs applications destinatrices sur la même machine par l'intermédiaire des *ports*. Un port est une destination abstraite sur une machine identifié par un numéro qui sert d'interface à l'application pour recevoir et émettre des données. Par exemple,

```
...
tftp            69/udp
...
snmp            161/udp
...
```

est un court extrait du fichier `/etc/services` de la machine `vega.info.univ-angers.fr` dans lequel sont enregistrés les numéros de port utilisés par chaque application. On y voit que l'application tftp utilise le port 69 et que l'application snmp utilise le port 161[2.9]. Chaque datagramme émis par UDP est encapsulé dans un datagramme IP en y fixant à 17 la valeur du protocole (voir la section 2.5.1). Le format détaillé d'un datagramme UDP est donné dans la figure 2.22.

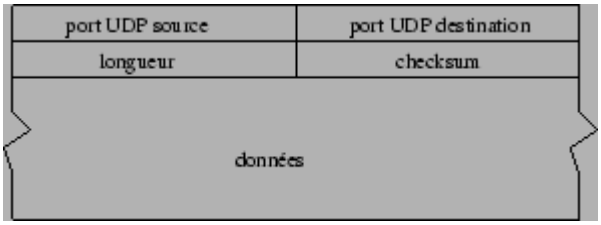

Figure 2.22: Structure d'un datagramme UDP.

Les *numéros de port* (chacun sur 16 bits) identifient les processus émetteur et récepteur. Le champ *longueur* contient sur 2 octets la taille de l'en-tête et des données transmises. Puisqu'un datagramme UDP peut ne transmettre aucune donnée la valeur minimale de la longueur est 8. Le *checksum* est un total de contrôle qui est optionnel car il n'est pas indispensable lorsque UDP est utilisé sur un réseau très fiable. S'il est fixé à 0 c'est qu'en fait il n'a pas été calculé. De manière précise, UDP utilise l'en-tête et les données mais également une *pseudo-entête* pour aboutir à l'ensemble décrit figure 2.23.

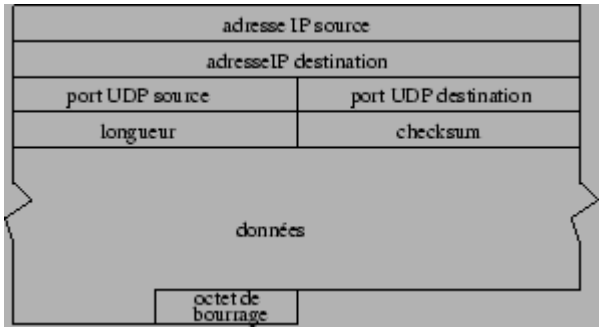

Figure: Champs utilisés pour le calcul du checksum UDP.

Cette pseudo en-tête comprend les adresses IP source[2.10]et destination du datagramme ainsi qu'un éventuel octet de bourrage pour aboutir à un nombre d'octets total pair. À partir de cet ensemble, le total de contrôle est calculé de la même manière que dans le cas du datagramme IP (voir section 2.5.1). Si le résultat donne un checksum nul, son complément à 1, c'est-à-dire 65535 (16 bits à 1), est en fait placé dans la zone de contrôle. Ce détail permet d'éviter la confusion avec le checksum nul qui indique qu'il n'a pas été calculé. Précisons enfin que la pseudo en-tête et l'octet de bourrage ne sont pas transmis et qu'ils n'interviennent pas dans le calcul du champ longueur. À la réception UDP utilise l'adresse IP de destination et l'adresse IP émettrice inscrite dans l'en-tête du datagramme IP pour calculer, de la même manière qu'à l'émission, une somme de contrôle qui permettra d'assurer que le datagramme est délivré sans erreur et à la bonne machine. Si une erreur de transmission est détectée, le

datagramme UDP est détruit «en silence». Sinon, UDP oriente les données du datagramme vers la file d'attente associée au numéro de port destination pour que l'application associée à celui-ci puisse les y lire.

2.6.2 Le protocole TCP.

Contrairement à UDP, TCP est un protocole qui procure un service de flux d'octets orienté connexion et fiable. Les données transmises par TCP sont encapsulées dans des datagrammes IP en y fixant la valeur du protocole à 6.

Le terme *orienté connexion* signifie que les applications dialoguant à travers TCP sont considérées l'une comme un *serveur*, l'autre comme un *client*, et qu'elles doivent établir une connexion avant de pouvoir dialoguer (comme dans le cas de l'utilisation du téléphone). Les ordinateurs vérifient donc préalablement que le transfert est autorisé, que les deux machines sont prêtes en s'échangeant des messages spécifiques. Une fois que tous les détails ont été précisés, les applications sont informées qu'une connexion a été établie et qu'elles peuvent commencer leurs échanges d'informations. Il y a donc exactement deux extrémités communiquant l'une avec l'autre sur une connexion TCP[2.11]. Cette connexion est bidirectionnelle simultanée (*full duplex*) et composée de deux flots de données indépendants et de sens contraire. Il est cependant possible d'inclure dans l'en-tête de segments TCP d'une communication de A vers B des informations relatives à la communication de B vers A. Cette technique de superposition (*piggybacking*) permet de réduire le trafic sur le réseau.

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. Le contenu des octets n'est pas du tout interprété par TCP, c'est donc aux applications d'extrêmité de savoir gérer la structure du flot de données.

Si elles sont trop volumineuses, les données à transmettre pour une application sont fractionnées en fragments dont la taille est jugée optimale par TCP. A l'inverse, TCP peut regrouper des données d'une application pour ne former qu'un seul datagramme de taille convenable de manière à ne pas charger inutilement le réseau. Cette unité d'information émise est appelée *segment* comme déjà présenté dans la figure 2.6. Certaines applications demandent que les données soient émises immédiatement, même si le tampon n'est pas plein. Pour cela, elles utilisent le principe du *push* pour forcer le transfert. Les données sont alors émises avec un bit marquant cela pour que la couche TCP réceptrice du segment remette immédiatement les données à l'application concernée.

La *fiabilité* fournie par TCP consiste à remettre des datagrammes, sans perte, ni duplication, alors même qu'il utilise IP qui lui est un protocole de remise non fiable. Ceci est réalisé à l'aide de la technique générale de l'accusé de réception (*ACK*) présentée de manière simplifiée dans la figure 2.24

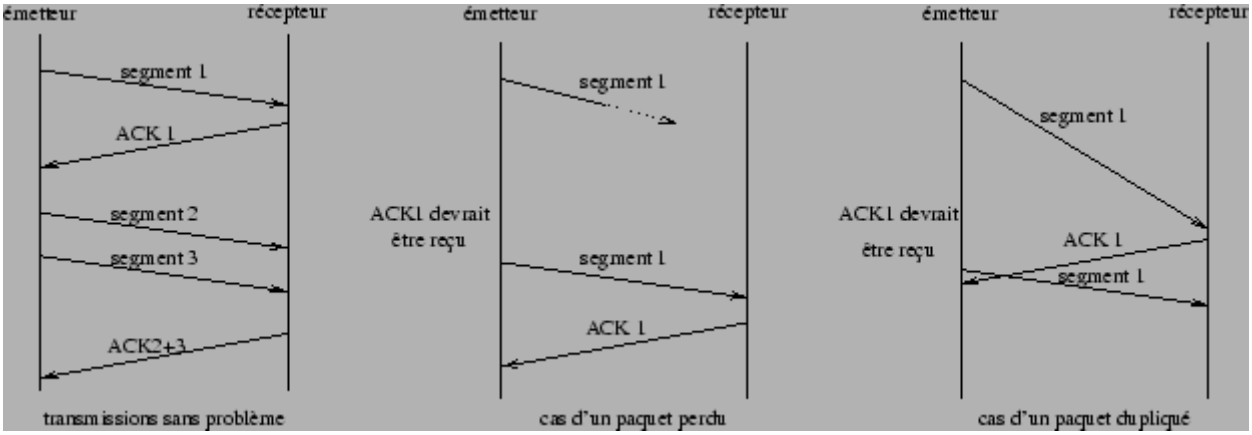

Figure: Échanges de segments TCP.

Chaque segment est émis avec un numéro qui va servir au récepteur pour envoyer un accusé de réception. Ainsi l'émetteur sait si l'information qu'il voulait transmettre est bien parvenue à destination. De plus, à 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 la temporisation expire sans qu'il n'ait reçu de ACK, l'émetteur considère que le segment s'est perdu[2.12] et il le réexpédie. Mais il se peut que la temporisation expire alors que le segment a été transmis sans problème, par exemple suite à un engorgement de réseau ou à une perte de l'accusé de réception correspondant. Dans ce cas, l'émetteur réémet un segment alors que c'est inutile. Mais le récepteur garde trace des numéros de segments reçus, donc il est apte à faire la distinction et peut éliminer les doublons.

La figure 2.25 donne le format d'un segment TCP qui sert aux trois fonctionnalités de TCP : établir une connexion, transférer des données et libérer une connexion.

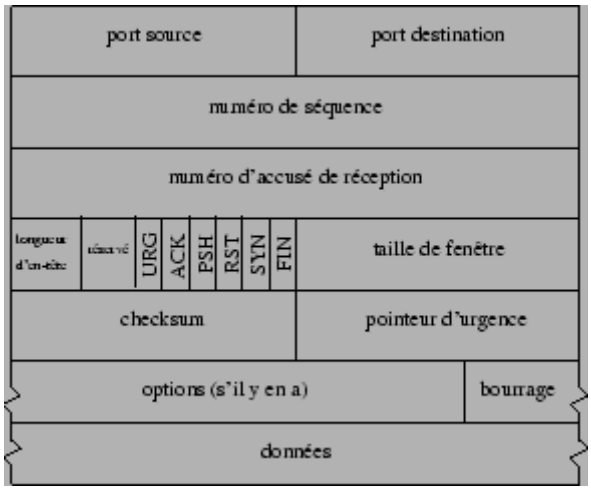

Figure 2.25: Format du segment TCP.

L'en-tête, sans option, d'un segment TCP a une taille totale de 20 octets et se compose des champs suivants.

-
  Le *port source* et le *port destination* identifient les applications émettrice et réceptrice. En les associant avec les numéros IP source et destination du datagramme IP qui transporte un segment TCP on identifie de manière unique chaque connexion[2.13].

-
  Le *numéro de séquence*[2.14] donne la position du segment dans le flux de données envoyées par l'émetteur; c'est-à-dire la place dans ce flux du premier octet de données transmis dans ce segment.

-
  Le *numéro d'accusé de réception* contient en fait le numéro de séquence suivant que le récepteur s'attend à recevoir; c'est-à-dire le numéro de séquence du dernier octet reçu avec succès plus 1. De manière précise, TCP n'acquitte pas un à un chaque segment qu'il reçoit, mais acquitte l'ensemble du flot de données jusqu'à l'octet *k*-1 en envoyant un acquittement de valeur *k*. Par exemple, dans une transmission de 3 segments de A vers B, si les octets de 1 à 1024 sont reçus correctement, alors B envoie un ACK avec la valeur 1025. Puis, si le segment suivant contenant les octets de 1025 à 2048 se perd et que B reçoit d'abord correctement le segment des octets de 2049 à 3072, B n'enverra pas d'accusé de réception positif pour ce troisième segment. Ce n'est que lorsque B recevra le deuxième segment, qu'il pourra envoyer un ACK avec la valeur 3073, que A interprétera comme l'acquittement des deux derniers segments qu'il a envoyés. On appelle cela un acquittement cumulatif.

-
  La *longueur d'en-tête* contient sur 4 bits la taille de l'en-tête, y compris les options présentes, codée en multiple de 4 octets. Ainsi une en-tête peut avoir une taille variant de 20 octets (aucune option) à 60 octets (maximum d'options).

-
  Le champ *réservé* comporte 6 bits réservés à un usage ultérieur.

-
  Les 6 champs *bits de code* qui suivent permettent de spécifier le rôle et le contenu du segment TCP pour pouvoir interpréter correctement certains champs de l'en-tête. La signification de chaque bit, quand il est fixé à 1 est la suivante.

  .
  URG, le *pointeur de données urgentes* est valide.

  .
  ACK, le champ *d'accusé de réception* est valide.

  .
  PSH, ce segment requiert un *push*.

  .
  RST, réinitialiser la connexion.

  .
  SYN, synchroniser les numéros de séquence pour initialiser une connexion.

  .
  FIN, l'émetteur a atteint la fin de son flot de données.

-

La *taille de fenêtre* est un champ de 16 bits qui sert au contrôle de flux selon la méthode de la fenêtre glissante. Il indique le nombre d'octets (moins de 65535) que le récepteur est prêt à accepter. Ainsi l'émetteur augmente ou diminue son flux de données en fonction de la valeur de cette fenêtre qu'il reçoit.

-

Le *checksum* est un total de contrôle sur 16 bits utilisé pour vérifier la validité de l'en-tête et des données transmises. Il est obligatoirement calculé par l'émetteur et vérifié par le récepteur. Le calcul utilise une pseudo-entête analogue à celle d'UDP (voir la section 2.6.1).

-

Le *pointeur d'urgence* est un offset positif qui, ajouté au numéro de séquence du segment, indique le numéro du dernier octet de donnée urgente. Il faut également que le bit *URG* soit positionné à 1 pour indiquer des données urgentes que le récepteur TCP doit passer le plus rapidement possible à l'application associée à la connexion.

-

L'*option* la plus couramment utilisée est celle de la taille maximale du segment TCP qu'une extrémité de la connexion souhaite recevoir. Ainsi, lors de l'établissement de la connexion il est possible d'optimiser le transfert de deux manières. Sur un réseau à haut débit, il s'agit de remplir au mieux les paquets, par exemple en fixant une taille qui soit telle que le datagramme IP ait la taille du MTU du réseau. Sinon, sur un réseau à petit MTU, il faut éviter d'envoyer des grands datagrammes IP qui seront fragmentés, car la fragmentation augmente la probabilité de pertes de messages.

L'*établissement* et la *terminaison* d'une connexion suit le diagramme d'échanges de la figure 2.26.

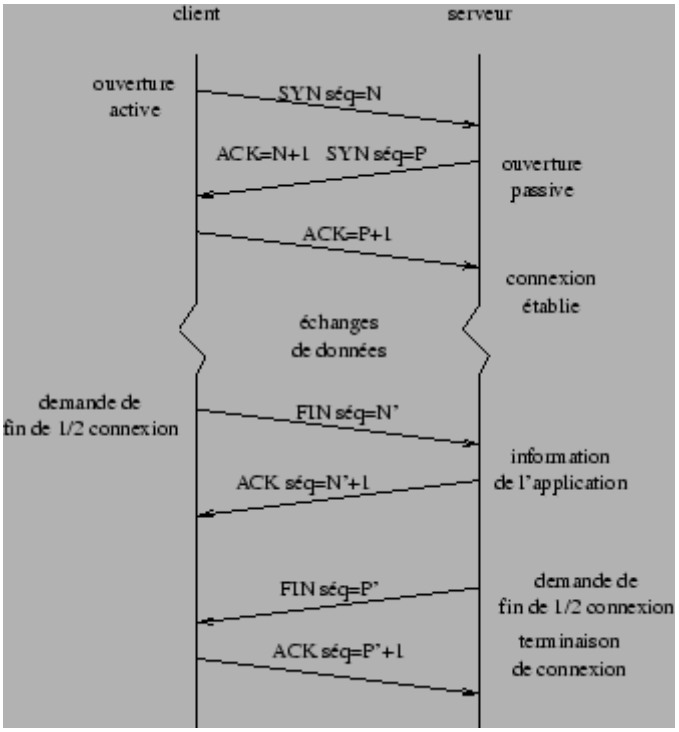

Figure: Établissemnet et terminaison d'une connexion TCP.

L'extrémité demandant l'ouverture de la connexion, est le *client*. Il émet un segment SYN (où le bit SYN est fixé à 1) spécifiant le numéro de port du *serveur* avec lequel il veut se connecter. Il expédie également un numéro de séquence initial *N*. Cette phase est appelée ouverture active et «consomme» un numéro de séquence. Le serveur répond en envoyant un segment dont les bits ACK et SYN sont fixés à 1. Ainsi, dans un même segment il acquitte le premier segment reçu avec une valeur de ACK=*N*+1 et il indique un numéro de séquence initial. Cette phase est appelée ouverture passive. Le client TCP doit évidemment acquitter ce deuxième segment en renvoyant un segment avec ACK=*P*+1[2.15].

La terminaison d'une connexion peut être demandée par n'importe quelle extrémité et se compose de deux *demi-fermetures* puisque des flots de données peuvent s'écouler simultanément dans les deux sens. L'extrémité qui demande la fermeture (le client dans l'exemple de la figure 2.26 émet un segment où le bit FIN est fixé à 1 et où le numéro de séquence vaut *N*'. Le récepteur du segment l'acquitte en retournant un ACK=*N*'+1 et informe l'application de la demi-fermeture de la connexion. À partir de là, les données ne peuvent plus transiter que dans un sens (de l'extrémité ayant accepté la fermeture vers l'extrémité l'ayant demandée), et dans l'autre seuls des

accusés de réception sont transmis. Quand l'autre extrémité veut fermer sa demi-connexion, elle agit de même que précédemment ce qui entraîne la terminaison complète de la connexion.

Le *transfert de données* de TCP est de deux types : le transfert *interactif* dans lequel chaque segment transporte très peu d'octets, voire un seul, et le transfert *en masse* où chaque segment transporte un maximum d'octets. Cette distinction est confortée par une étude de 1991 qui indique que la moitié des paquets TCP contient des données en masse (ftp, mail, news, ...) et l'autre moitié des données interactives[2.16] (telnet, rlogin, ...). Mais, 90% des octets transmis proviennent de données en masse et 10% seulement de données interactives car il apparaît que 90% des paquets de telnet et rlogin comportent moins de 10 octets.

Un exemple de transfert interactif est celui généré par la commande `rlogin` lancée depuis un client vers un serveur. Dans ce cas de figure, tous les caractères tapés par l'utilisateur sur le client sont envoyés vers le serveur en utilisant un caractère par segment, et ils sont ensuite renvoyés en sens inverse par le serveur pour un écho sur l'écran du client. Tous les segments échangés dans ce cas là ont leur bit *PSH* fixé à 1. Or, tout segment doit être acquitté dans un sens comme dans l'autre; ce qui devrait amener au diagramme d'échanges illustré dans le a) de la figure 2.27.

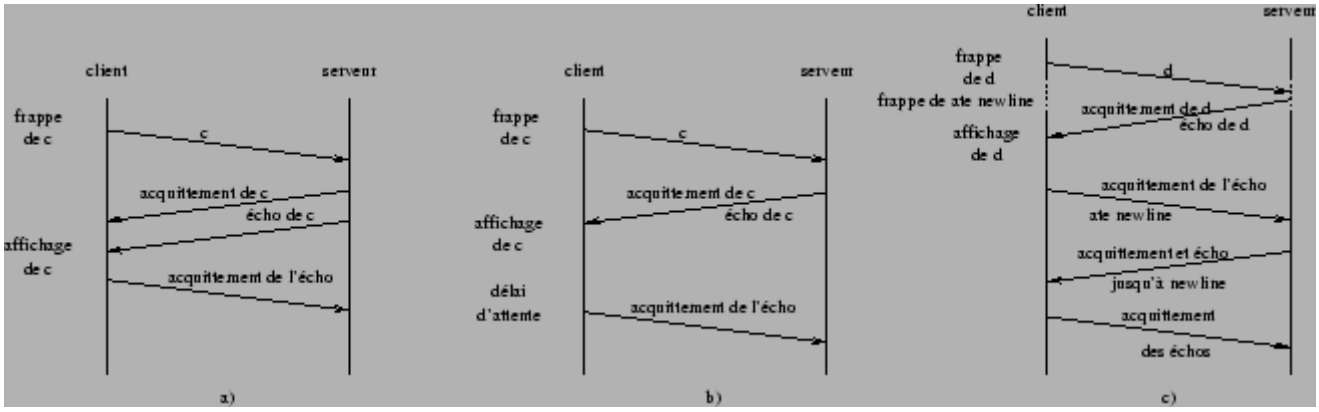

Figure: Échange de données interactif.

En fait, TCP gère ce type d'échange avec la procédure d'*acquittement retardé* qui consiste à envoyer l'acquittement en l'incluant dans un segment qui transporte également des données comme illustré dans le b) de la figure 2.27. Pour cela l'acquittement est généralement retardé de 200ms dans le but d'attendre d'éventuelles données à transmettre. Cependant, dans ce contexte la frappe de la commande `date` sur le client provoquerait quand même l'échange de 15 datagrammes IP de 41 octets[2.17], pour transporter à chaque fois seulement 1 octet utile. Ce type de situation est peu gênant sur un LAN mais devient très vite préjudiciable au bon fonctionnement d'un WAN. Une solution à ce problème est l'*algorithme de Nagle* (RFC 896) qui spécifie qu'une connexion TCP ne peut avoir seulement un petit segment non encore acquitté. Ainsi, si un réseau a un fort débit et n'est pas du tout chargé la procédure sera celle du b) de la figure 2.27. Par contre, dès que le temps de transfert est non négligeable, les acquittements vont arriver plus lentement que la frappe des caractères par l'utilisateur du poste client. Dans ce cas, la couche TCP du client va accumuler de petits volumes de données (quelques caractères) et les envoyer dans le même segment TCP diminuant ainsi considérablement le nombre d'échanges comme illustré dans le c) de la figure 2.27. Dans les cas où de petits messages doivent être remis sans délai (mouvement de souris dans le cas d'un serveur X) l'algorithme de Nagle doit être invalidé pour donner une impression de temps réel.

Dans le cas d'un transfert de données en masse, TCP utilise la technique de la *fenêtre glissante* pour contrôler le flux des échanges. Ceci est primordial quand un micro ordinateur communique avec un gros ordinateur, sinon le tampon d'entrée du micro sera très vite saturé. Ceci consiste en un contrôle de flux de *bout en bout*. Mais il s'agit aussi de réguler le trafic en fonction de la charge des routeurs et du débit des réseaux traversés. On rappelle que l'ensemble d'un flux de données unidirectionnel d'une machine A vers une machine B est constitué d'une séquence d'octets tous numérotés individuellement. La fenêtre glissante va consister à fixer quels sont les octets appartenant à ce flux que A peut émettre comme c'est illustré dans la figure 2.28.

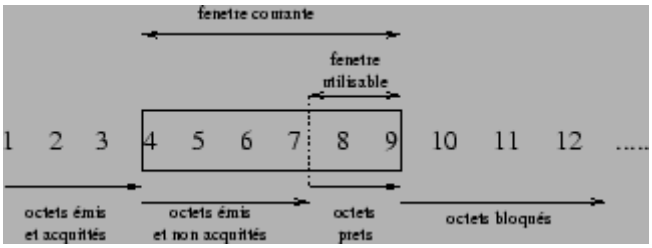

Figure: Fenêtre glissante de TCP.

Dans cet exemple la fenêtre couvre les octets de 4 à 9, car la taille de la fenêtre courante est 6 et que tous les octets jusqu'au troisième inclus ont été émis et acquittés.

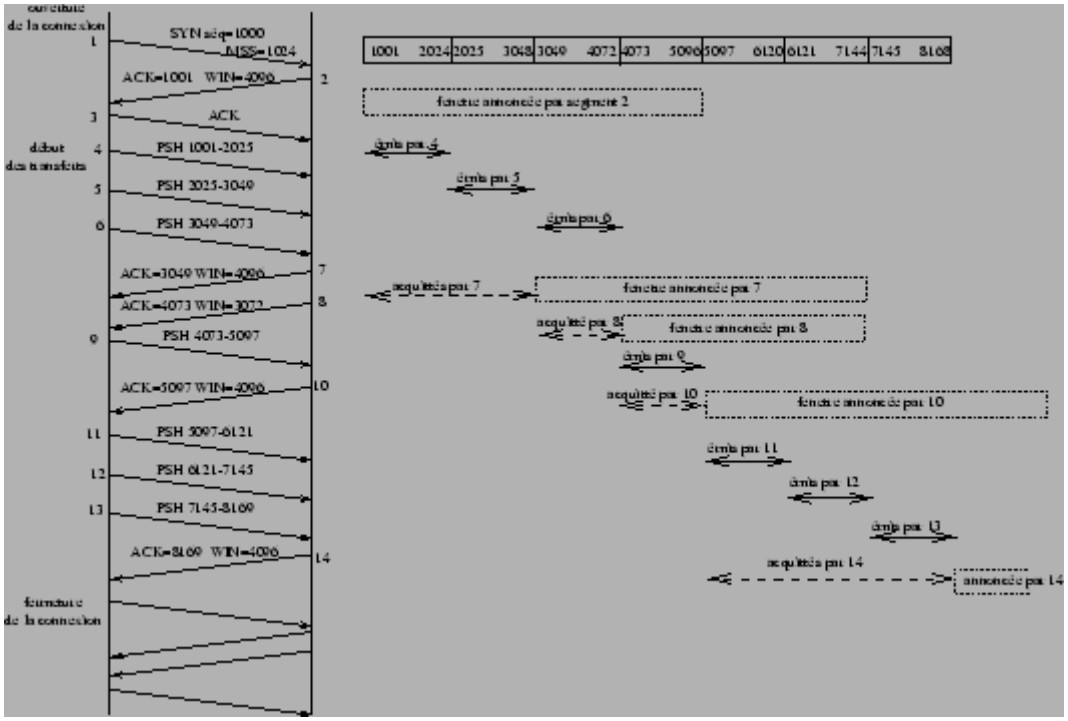

Figure: Exemple d'évolution de fenêtre glissante.

A tout instant TCP calcule sa fenêtre utilisable qui est constituée des octets présents dans la fenêtre et non encore envoyés. Ces octets sont généralement immédiatement transmis. Pour le flot de A vers B, la taille de la fenêtre est contrôlée par B qui envoie dans chacun de ses accusés de réception la taille de la fenêtre qu'il désire voir utiliser. Si la demande exprime une augmentation, A déplace le bord droit de sa fenêtre courante et émet immédiatement les octets qui viennent d'y entrer. Si la demande exprime une diminution, il est déconseillé de déplacer réellement le bord droit de la fenêtre vers la gauche. Ce rétrécissement est opéré lors des glissements de la fenêtre vers la droite avec l'arrivée des accusés de réception.

La figure 2.29 illustre[2.18] les échanges de segments entre un émetteur A et un récepteur B et l'évolution de la fenêtre glissante de A suivant les indications de B.

2.7 Les applications.

On décrit ici de manière succincte quelques applications majeures que l'on trouve sur Internet. Toutes ces applications sont bâties sur le modèle «client-serveur» à savoir qu'une des deux extrémités de la connexion (TCP UDP)/IP rend des services à l'autre extrémité.

Sous-sections

2.7.1 Protocole de démarrage : BOOTP.

BOOTP (*Bootsrap Protocol*) est un protocole de démarrage de terminaux X ou stations sans disque qui utilise UDP comme couche de transport et est généralement associé à TFTP (voir 2.7.4) ou NFS (voir 2.7.3). Comme RARP (voir section 2.4.4), il sert principalement à fournir son adresse IP à une machine que l'on démarre sur un réseau. Cependant il est plus intéressant que RARP, car il se situe à un niveau supérieur, il est donc moins lié au type de matériel du réseau. De plus, il transmet plus d'information que RARP qui, lui, ne renvoie qu'une adresse IP.

Un autre protocole, *DHCP* (*Dynamic Host Configuration Protocol*) permet, lui, d'attribuer cette adresse IP dynamiquement, c'est-à-dire que l'adresse IP affectée à la machine qui démarre peut changer d'un démarrage à l'autre. BOOTP fait cela de manière statique en utilisant un serveur (ou plusieurs) qui contient dans un fichier l'adresse IP à distribuer à chaque machine. Le fichier est maintenu à jour par l'administrateur du réseau et contient pour chaque machine plusieurs informations comme illustré ci-après pour le terminal X de l'auteur.

```
# .........
#
#       ba -- broadcast bootp reply for testing with bootpquery
#       bf -- bootfile (for tftp download)
#       ds -- domain name server IP address
#       gw -- gateway IP address
#       ha -- hardware address (link level address) (hex)
#       hd -- home directory for bootfile (chrooted to tftp home directory)
#       hn -- send nodename (boolean flag, no "=value" needed)
#       ht -- hardware type (ether) (must precede the ha tag)
#       ip -- X terminal IP address
#       sm -- network subnet mask
#       tc -- template for common defaults (should be the first option
listed)
#       vm -- vendor magic cookie selector (should be rfc1048)
#       T144  remote config file name (file name must be enclosed in "")
#

# H104 (Pascal Nicolas) prise I141 :
tx-pn:\
        ht=Ethernet:\
        ha=0x08001103ec2c:\
        bf=/usr/tekxp/boot/os.350:\
        ip=193.49.162.63:\
        sm=255.255.255.0:\
        gw=193.49.162.220:\
        vm=rfc1048:\
        ds=193.49.162.9:
```

Le format du message BOOTP est donné dans la figure 2.30.

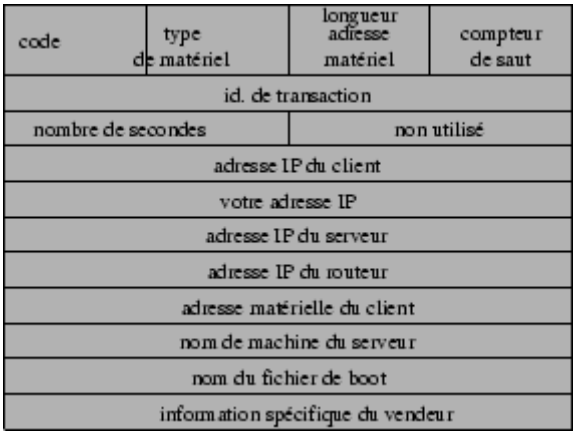

Figure: Format de requête ou réponse BOOTP.

Le *code* vaut 1 pour une requête et 2 pour une réponse, le *type de matériel* vaut par exemple 1 pour un réseau Ethernet et dans ce cas le champ *longueur de l'adresse physique* vaut 6 (octets). Le champ *compteur de saut* vaut 0 en standard, mais si la demande transite par un routeur celui-ci l'incrémente de 1. L'*identificateur de transaction* est un entier de 32 bits fixé aléatoirement et qui sert à faire correspondre les réponses avec les requêtes. Le *nombre de secondes* est fixé par le client et sert à un serveur secondaire de délai d'attente avant qu'il ne réponde au cas où le serveur primaire serait en panne. Parmi les 4 adresses IP d'une requête, le client remplit celles qu'il connaît et met les autres à 0 (généralement il n'en connait aucune). Dans sa réponse, le serveur indique l'adresse IP de la machine client dans le champ *votre adresse IP* et sa propre adresse dans le champ *adresse IP du serveur*. Il peut aussi donner son nom terminé par le caractère `nul`. Si un proxy est utilisé, celui-ci indique son adresse dans le champ prévu. Le champ *adresse matérielle du client* sert à celui-ci pour y indiquer son adresse physique. Ainsi cette adresse sera plus facilement disponible pour le processus BOOTP serveur que celle placée dans la trame Ethernet. Le *nom du fichier de boot* est le nom du fichier transmis par le serveur au client pour que celui-ci puisse ensuite continuer son démarrage. La dernière partie du message est réservée à des caractéristiques particulières données par chaque constructeur de matériel et permet d'ajouter des fonctionnalités supplémentaires.

Quand il émet une requête BOOTP, le client l'encapsule dans un datagramme UDP en fixant le port source à 68 et le port destination (celui du serveur) à 67. Dans la majorité des cas, il ne sait pas préciser la valeur de l'adresse IP de destination et la fixe donc églae à `255.255.255.255` l'adresse de diffusion. Ainsi, la trame Ethernet correspondante sera diffusée à toutes les machines du réseau et grâce au numéro de port seuls le(s) serveur(s) BOOTP reçoit le message et le traite. Pour cela, il consulte la table de correspondance et retourne sa réponse en y fixant l'adresse IP du client, le nom du fichier à télécharger et l'adresse IP du serveur.

Il reste un problème à régler. Lors de l'émission du datagramme IP contenant la réponse, la couche de liens doit établir la correspondance adresse IP/adresse physique du client pour construire la trame physique à émettre. Or, cette correspondance ne peut être connue dans la table ARP à cet instant puisque la machine client démarre. Et si le logiciel de liens émet une requête ARP (voir section 2.4.4) la machine client ne sait pas répondre puisqu'elle ne peut pas reconnaître son adresse IP qu'elle ne connaît pas encore. Il existe deux solutions à ce problème. Soit, le module BOOTP, qui dispose de cette correspondance, enrichit la table ARP avant d'émettre. Soit, il n'en a pas les droits et alors il diffuse la réponse à toutes les machines du réseau, mais cette solution est à éviter.

2.7.2 Connexion à distance : Telnet et Rlogin.

Telnet et *Rlogin* sont deux applications qui permettent à un utilisateur de se connecter à distance sur un ordinateur, pourvu que cet utilisateur y dispose d'un accès autorisé. Ces deux applications permettent toutes les deux de prendre le contrôle (du moins partiellement) d'un ordianteur distant, mais Rlogin ne permet de le faire qu'entre deux machines Unix, tandis qu'il existe des clients Telnet pour de nombreuses plateformes (Unix, Windows, MacOs, ...). Telnet et Rlogin sont tous les deux bâtis sur TCP.

Le schéma de fonctionnement de Telnet est donné dans la figure 2.31.

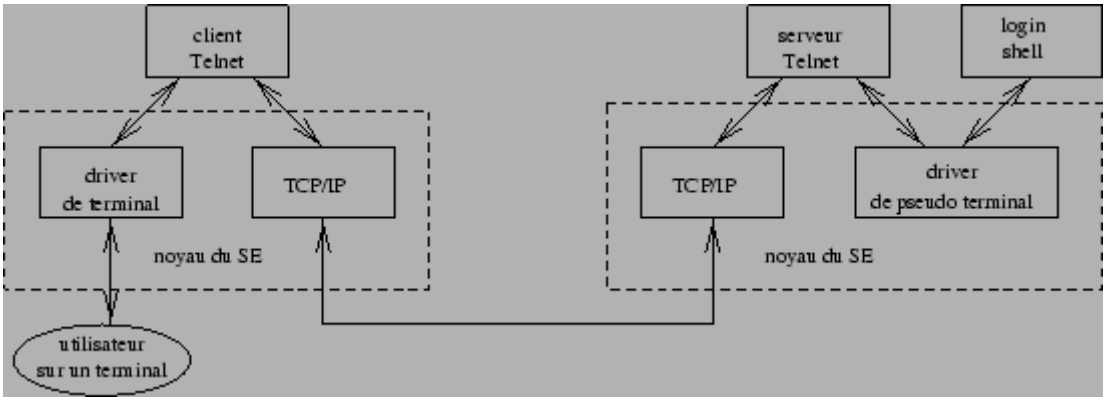

Figure: Schéma de focntionnement de l'application Telnet.

Telnet définit une interface de communication, le terminal virtuel de réseau, pour que clients et serveurs n'aient pas à connaître les détails d'implantation de chaque système d'exploitation. De cette façon, les échanges se font dans un langage commun compris à la fois par le client et le serveur qui n'ont qu'à assurer une traduction de (ou vers) leur propre langage vers (depuis) ce langage cible.

2.7.3 Système de fichiers en réseau : NFS.

NFS (*Network File System*) est un système qui permet de rendre transparente l'utilisation de fichiers répartis sur différentes machines. Son architecture générale est donnée dans la figure 2.32.

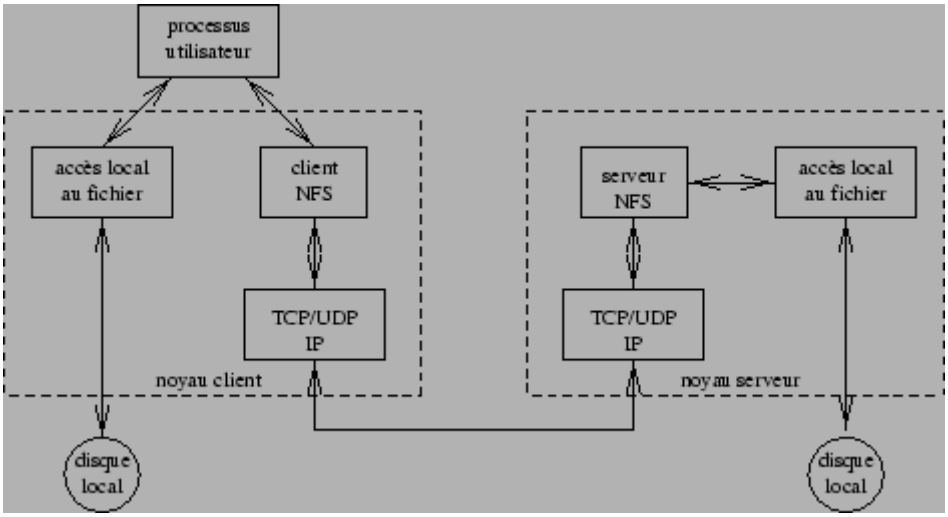

Figure 2.32: Configuration client serveur de NFS.

NFS utilise principalement UDP, mais ses nouvelles implantations utilisent également TCP. Lorsqu'un processus utilisateur a besoin de lire, écrire ou accéder à un fichier le système d'exploitation transmet la demande soit au système de fichier local, soit au client NFS. Dans ce dernier cas, le client NFS envoie des requêtes au serveur NFS de la machine distante. Ce serveur s'adresse à la routine locale d'accès au fichiers qui lui retourne le résulat retransmis vers le client par la connexion UDP (ou TCP) IP. Il ne s'agit pas ici de transférer un fichier d'une machine à l'autre mais simplement de le rendre disponible de manière totalement transparente.

2.7.4 Transfert de fichier : TFTP et FTP.

TFTP (*Trivial File Transfert Protocol*) et *FTP* (*File Transfert Protocol*) permettent tous les deux de transférer des fichiers d'une machine à une autre. Cependant TFTP, bâti sur UDP, est beaucoup plus sommaire que FTP qui utilise TCP. L'utilisation de FTP depuis un poste client pour aller chercher ou déposer un fichier sur un serveur nécessite de la part de l'utilisateur de se connecter avec un nom et un mot de passe. Donc, si l'utilisateur n'est pas reconnu la connexion FTP ne sera pas établie. Dans le cas particulier d'un serveur ftp public, la connexion se fait avec le nom `anonymous` et il est conseillé de donner son adresse électronique comme mot de passe. Dans le cas de TFTP, aucune authentification préalable n'est nécessaire. C'est pourquoi, lorsqu'un serveur TFTP est installé sur une machine il n'offre des possibilités d'accès qu'à un nombre restreint de fichiers bien spécifiques. Ces fichiers sont généralement des fichiers de démarrage de terminaux X ou stations sans disque qui les récupérent après en avoir été informés par le protocole BOOTP (voir section 2.7.1).

Les différents messages TFTP sont donnés dans la figure 2.33 et se distingeunt selon leur *code d'opération*.

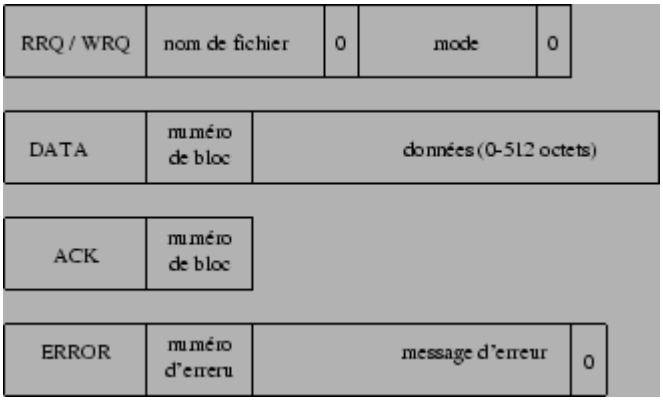

Figure 2.33: Format des messages TFTP.

- `RRQ` indique une requête de lecture de fichier (transmis au client) et `WRQ` une requête d'écriture de fichier (transmis au serveur). Ensuite, vient le nom du fichier terminé par un caractère nul. Le champ mode, terminé par un caractère nul également, est égal à `netascii` pour indiquer que le fichier est un fichier texte où chaque ligne est terminée par `CR LF`. Ces deux caractères doivent peut-être être convertis dans la syntaxe utilisée par la machine locale pour marquer les fins de ligne des fichiers textes. Il est égal à `octet` dans le cas d'un fichier binaire à transférer tel quel.
- `DATA` débute les paquets de données à transmettre. Un fichier de *N* octets sera découpé en *N* div 512 tels paquets contenant chacun 512 octets de données et un paquet contenant *N* mod 512 octets qui sera reconnu comme le paquet final puisqu'il contient moins de 512 octets. Le champ *numéro de bloc* sert à numéroter chaque paquet de données et est utilisé pour l'accusé de réception.
- *ACK* indique que le message acquitte le bloc de numéro spécifié dans le message. TFTP est obligé de s'assurer lui même de la bonne transmission des données puisqu'il utilise UDP qui est un protocole non fiable. Le protocole d'acquittement est de type *stop-and-wait* car après avoir envoyé un paquet l'émetteur attend l'accusé de réception du récepteur avant d'envoyer le paquet suivant. Si l'émetteur ne reçoit pas d'acquittement avant l'expiration de son délai d'attente il réexpédie le paquet perdu. De même, le récepteur qui ne reçoit plus de paquets après son délai d'attente renvoie à nouveau son acquittement. Seulement, si le ACK *k* est retardé mais non perdu, l'émetteur va retransmettre le paquet *k* que le récepteur va à nouveau acquitter. Donc, deux ACK *k* vont finalement parvenir à l'émetteur ce qui va déclencher de sa part l'envoi de deux paquets *k*+1, qui provoqueront deux ACK *k*+1 et donc l'envoi de deux paquest *k*+2, etc...
- Les messages débutant par *ERROR* indiquent une erreur de transmission et transportent un code et un message d'erreur. Lorsque cela arrive, le transfert est immédiatement interrompu.

Lorsqu'il demande une connexion le client s'attribue un port éphémère UDP et envoie sa requête au serveur sur le port 69 prévu pour FTP. À ce moment-là, le serveur va s'attribuer un nouveau port éphémère qui devra être détecté par le client et qui servira tout le temps de la connexion. Il ne conserve pas le port 69 tout au long de l'échange car cela l'obligerait soit à refuser d'autres connexions pendant ce temps, soit à les multiplexer ensemble ce qui alourdirait le protocole.

Quant à lui, FTP est défini au-dessus de TCP et utilisent deux connexions TCP IP pour fonctionner comme illustré dans la figure 2.34.

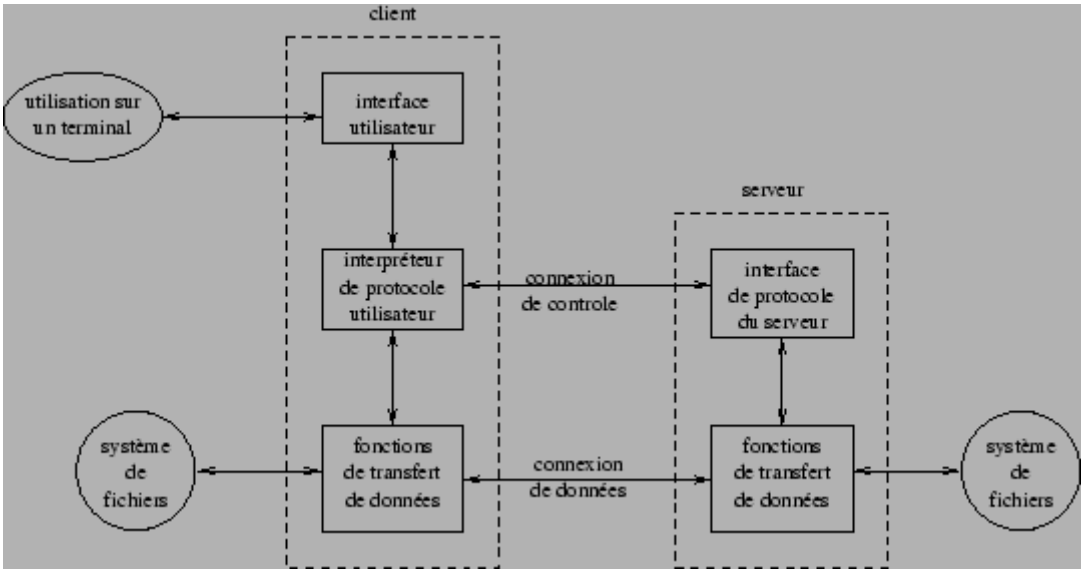

Figure 2.34: Tansfert de fichier par FTP.

Tout d'abord on y voit que le client utilise FTP à travers une interface qui peut être graphique (logiciels XFTP, WS-FTP, Fetch, ...) ou texte (mode commandes d'unix par exemple). La *connexion de contrôle* est établie de façon normale en mode client serveur sur le port 21 du serveur et sur un port aléatoire du client pour tout ce qui est de type transfert interactif. Elle sert donc tout le temps de la session à transférer les commandes du client et presque toutes les réponses du serveur. La *connexion de données* sert à transférer les fichiers et les contenus de répertoires du serveur, c'est-à-dire tous les transferts de masse. En effet, lorsque le client demande le contenu d'un répertoire la réponse peut être très longue et il est préférable de l'envoyer sur cette connexion plutôt que sur celle du transfert interactif. À chaque fois qu'un fichier doit être transféré, dans un sens ou dans l'autre, le client initie une connexion de données en s'attribuant un port et envoie au serveur une demande de connexion sur la connexion de contrôle. Le serveur se sert du numéro de port reçu pour établir la connexion de données entre son port 20 et ce port indiqué par le client.

2.7.5 Courrier électronique : smtp.

Le courrier électronique au sein d'Internet est géré par le protocole SMTP (*Simple Mail Transfer Protocol*) bâti sur TCP (port 25). Il permet d'échanger des messages entre un expéditeur et un (ou plusieurs) destinataire pourvu que leurs adresses soient connues. Une adresse de courrier électronique se présente sous la forme `nom@domaine` est doit être composée de lettres (minuscules ou majuscules sont indifférenciées), de chiffres, de _ (souligné) et de . (point). Il est à noter qu'un mécanisme d'alias permet de définir des équivalences entre adresses, notamment de préciser quelle machine parmi toutes celles d'un même domaine gère réellement le courrier de chaque utilisateur.

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

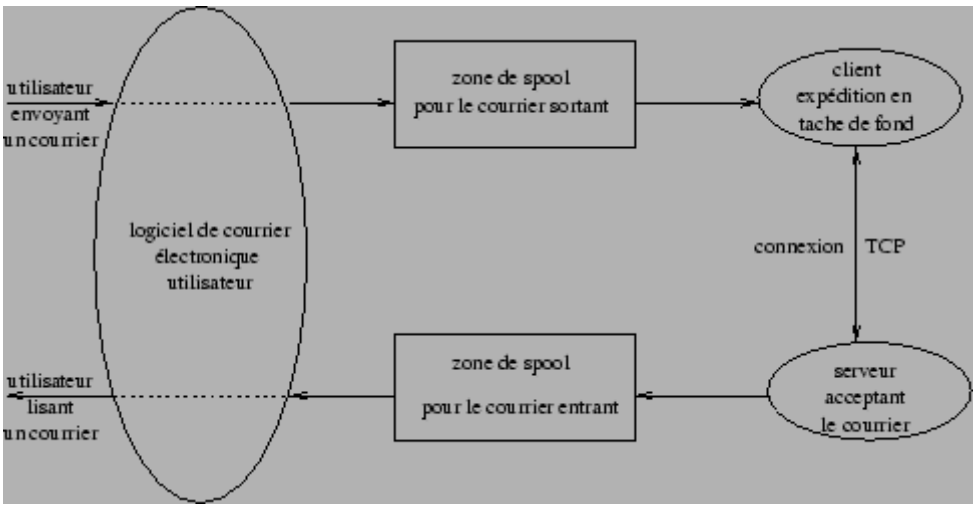

Figure: Schéma d'une messagerie SMTP.

Pour cela le système de messagerie fonctionne de la manière décrite en figure 2.35. Un courrier expédié par un utilisateur est d'abord copié dans une mémoire de *spool* accompagné des noms de l'expéditeur, du récepteur, de l'ordinateur destinataire et de l'heure de dépôt. Puis le système de messagerie active en tâche de fond le processus de transfert de courrier qui devient un client. Il associe le nom de l'ordinateur destinataire à une adresse IP et tente d'établir une connexion TCP avec le serveur SMTP de celui-ci. Si cela réussit, le processus de transfert envoie une copie du message au destinataire qui l'enregistre dans une zone de spool spécifique. Lorsque le client et le serveur se sont confirmés l'envoi et l'enregistrement complet du message le client supprime sa copie locale. Si le client n'arrive pas à établir une connexion TCP, ou si elle est rompue lors du transfert d'un message, il enregistre l'heure de cette tentative et réessaye quelque temps plus tard d'expédier le message. D'une manière générale un système de messagerie examine régulièrement sa zone de spool en envoi et tente d'expédier les messages (nouveau ou en attente à cause d'échec) qui s'y trouvent. Il finira par retourner à son expéditeurun message impossible à expédier après un délai important. Ce mode de fonctionnement (établir une connexion de bout en bout) assure qu'aucun message ne peut se perdre, soit il est délivré, soit son expéditeur est prévenu de l'échec.

Le tableau ci-dessous donne le détail d'une connexion TCP réussie qui envoie un message de l'utilisateur `toto@expediteur.fr` dont le courrier est géré par l'ordinateur `exp.expediteur.fr` vers l'utilisateur `titi@destinataire.fr` dont le courrier est géré par l'ordinateur `dest.destinataire.fr`. La première colonne décrit les étapes, la deuxième (respectivement troisième) colonne indique les commandes envoyées par l'expéditeur (respectivement destinataire) du courrier.

	client SMTP expéditeur sur `exp.expediteur.fr`	serveur SMTP destinataire sur `exp.destinataire.fr`
`exp.expediteur.fr` demande une connexion TCP sur le port 25 à `exp.destinataire.fr`		
`dest` accepte la demande		`220`

de connexion		`dest.destinataire.fr ...`
`exp` s'identifie	`HELO exp.expediteur.fr`	
`dest` accepte l'identification		`250 dest.destinataire.fr Hello exp.expediteur.fr pleased to meet you`
`exp` indique l'expéditeur	`MAIL From:<toto@expediteur.fr>`	
`dest` accepte l'expéditeur		`250 <toto@expediteur.fr> Sender Ok`
`exp` donne le destinataire	`RCPT To:<titi@destinataire.fr`	
`dest` a vérifié et accepté le destinataire		`250 <titi@destinataire.fr> Recipient Ok`
`exp` va envoyer les données	DATA	
`dest` est prêt à accepter le message		`354 Enter mail, end with ...`
`exp` envoie le message terminé par une ligne ne contenant qu'un point.	bla, blabla	
	.	
`dest` accepte le message		`250 OK`
`exp` demande à terminer la connexion	`QUIT`	
`dest` accepte de terminer la connexion		`221 dest.destinataire.fr closing connection`

2.7.6 News : nntp

NNTP (*Network News Transfert Protocol*) est le protocole d'échange des news[2.19] ou forums de discussions à travers *Usenet* (nom donné au réseau logique constitué des serveurs de news disséminés sur la panète). Comme illustré dans la figure 2.36, il assure l'échange des news entre les serveurs et également la communication entre serveur et postes clients aussi bien pour la lecture que pour l'écriture de messages.

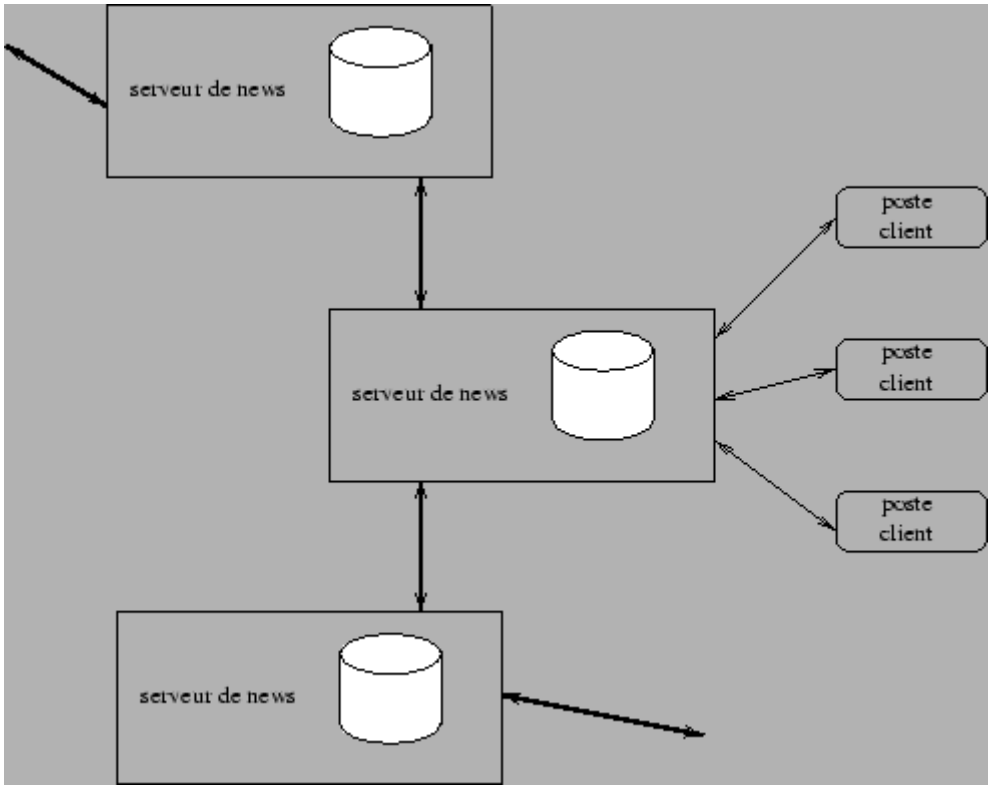

Figure 2.36: Communication au sein de Usenet.

Ainsi, lorqu'un utilisateur poste un article dans un groupe de news, il 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, qui eux-mêmes procéderont de la sorte. Ainsi, en quelques heures un message posté à Angers peut se retrouver sur un serveur de news en Australie. Mais, ce processus de diffusion systématique, n'est pas assuré pour tous les groupes de news existant au niveau mondial, car chaque serveur de news n'assure le relai que de certains groupes. En effet, il n'est peut-être pas très utile de diffuser sur les serveurs de news japonais le groupe `fr.petites-annonces.automobiles` :-). De plus, tout serveur de news fixe pour chaque groupe la durée de conservation des messages sur ses disques durs.

De manière plus technique, NNTP utilise TCP via le port 119, le client envoyant une commande ASCII à laquelle le serveur répond par un code numérique éventuellement suivi par des données. Ces données sont disposées sur plusieurs lignes terminées chacune par `CR/LF` et terminées par une ligne réduite à un point.

Tout d'abord il faut savoir qu'un serveur de news ne répond pas systématiquement à toutes les requêtes des postes clients, mais uniquement à celles provenant de machines qu'il autorise, par exemple celles de son domaine. Ceci est illustré ci-après où l'on voit que le serveur de news `news.univ-angers.fr` accepte la connexion depuis une machine située en Allemagne uniquement pour la lecture des news et accepte la lecture et l'écriture de messadepuis une machine située sur son réseau.

```
[brehat.haiti.cs.uni-potsdam.de]telnet news.univ-angers.fr 119
Trying 193.49.144.4...
Connected to news.univ-angers.fr.
Escape character is '^]'.
201 univ-angers.fr InterNetNews NNRP server INN 1.7.2 08-Dec-1997 ready (no
posting).

[helios]telnet news.univ-angers.fr 119
```

```
Trying 193.49.144.4...
Connected to news.univ-angers.fr.
Escape character is '^]'.
200 univ-angers.fr InterNetNews NNRP server INN 1.7.2 08-Dec-1997 ready
(posting ok).
```

La liste des commandes connues d'un serveur de news peut-être obtenue en l'interrogeant au moyen de la commande `help`, comme décrit ci après.

```
help
100 Legal commands
  authinfo user Name|pass Password|generic <prog> <args>
  article [MessageID|Number]
  body [MessageID|Number]
  date
  group newsgroup
  head [MessageID|Number]
  help
  ihave
  last
  list
[active|active.times|newsgroups|distributions|distrib.pats|overview.fmt|sub
scriptions]
  listgroup newsgroup
  mode reader
  newgroups yymmdd hhmmss ["GMT"] [<distributions>]
  newnews newsgroups yymmdd hhmmss ["GMT"] [<distributions>]
  next
  post
  slave
  stat [MessageID|Number]
  xgtitle [group_pattern]
  xhdr header [range|MessageID]
  xover [range]
  xpat header range|MessageID pat [morepat...]
  xpath MessageID
Report problems to <newsmaster@univ-angers.fr@news.univ-angers.fr>
.
```

Quant aux codes de réponses du serveur de news, ils sont décrits dans la table 2.2.

Tableau: Signification des 2 premiers chifres des codes de réponses de NNTP

code	description
1yz	information
2yz	requête acceptée
3yz	début de requête correcte, la suite eut être envoyée
4yz	requête correcte, mais non traitée
5yz	requête incorrecte, non implantée ou erreur fatale
x0z	connexion, mise en place et divers
x1z	choix des groupes de news
x2z	choix des articles
x3z	fonctions de distributions
x4z	postage

x8z	extension non standard
x9z	sortie de debug

Le rôle de quelques commandes de NNT est illustré ci-après en interrogeant le serveur `news.univ-angers.fr`.

- **`list`** retourne la liste des groupes de news, en indiquant leur nom, le numéro de l'article le plus récent, le numéro de l'article le plus ancien encore conservé, et la lettre `y` si le postage est libre ou `m` s'il est controlé par un modérateur.
- `list`
- `215 Newsgroups in form "group high low flags".`
- `control 0001251116 0001053999 y`
- `junk 0000000486 0000000480 y`
- `bionet.agroforestry 0000002281 0000002229 y`
- `bionet.announce 0000000165 0000000116 m`
- `....`
- **`group`** fixe un groupe courant et renvoie une estimation du nombre d'articles dans le forum, le numéro de l'article le plus ancien, celui de l'article le plus récent et le nom du groupe.
- `group fr.comp.os.linux`
- `211 6543 27618 34211 fr.comp.os.linux`
  La différence 34211-27618=6593 est plus grande que 6543 car tous les messages ne sont pas forcément conservés aussi longtemps les uns que les autres.

  Si le nom du groupe est erroné on obtient ceci :

  ```
  group fr.comp.linux
  411 No such group fr.comp.linux
  ```
- **`head`** envoie les en-têtes d'un article dont le numéro est spécifié.
- `head 34205`
- `221 34205 <3634E2EC.9B76A7E8@cern.ch> head`
- `Path: univ-angers.fr!enst!isdnet!newsgate.cistron.nl!het.net!news.belnet.be!`
- `news-zh.switch.ch!news-ge.switch.ch!cern.ch!news`
- `From: Nuno DOS SANTOS <Nuno.Dos.Santos@ces20s15ss15s12s9s3s4s5s`
- `Newsgroups: fr.comp.os.linux`
- `Subject: Instal carte de son`
- `Date: Mon, 26 Oct 1998 22:00:28 +0100`
- `Organization: CERN`
- `Lines: 10`
- `Message-ID: <3634E2EC.9B76A7E8@cern.ch>`
- `NNTP-Posting-Host: pcst101.cern.ch`
- `Mime-Version: 1.0`
- `Content-Type: text/plain; charset=us-ascii`
- `Content-Transfer-Encoding: 7bit`
- `X-Trace: sunnews.cern.ch 909435628 12357 (None) 128.141.182.63`
- `X-Complaints-To: news@sunnews.cern.ch`
- `X-Mailer: Mozilla 4.05 [en] (Win95; I)`
- `Xref: univ-angers.fr fr.comp.os.linux:34205`
- `.`
- **`body`** retourne le coprs de l'article dont le numéro est spécifié.
- `222 34205 <3634E2EC.9B76A7E8@cern.ch> body`

- Salut,
-
- J'arrive pas a instaler ma carte de son SB PCI awe64. Dans le HOW-TO ils
- parlent toujours de make config, xconfig, etc., mais la commande make ne
- passe pas. Il me donne l'erreur "make:*** No rule to make target
- 'xconfig'. Stop.". Le fichier sndstat est vide.
-
- Vous pouvez m'aider? Une sugestion?
-
- Merci
- .

Le point final ne fait pas partie du message mais est envoyé par NNTP pour terminer sa réponse

2.7.7 World Wide Web : http..

HTTP (*HyperText Transfer Protocol*) est le protocole de communication du web[2.20] permettant d'échanger des documents hypertextes contenant des données sous la forme de texte, d'images fixes ou animées et de sons.

Tout client web communique avec le port 80 d'un serveur HTTP par l'intermédiaire d'une, ou plusieurs, connexions TCP simultanées, chacune des connexions TCP ouvertes servant à récupérer l'un des composants de la page web.

Trois types de requêtes sont disponibles

- `GET url` renvoie l'information spécifiée par l'url.
- `HEAD url` renvoie l'en-tête de l'information demandée et non pas le contenu du document.
- `POST` pour envoyer du courrier électronique, des messages de news, ou des formulaires interactifs remplis par l'utilisateur.

La requête du client se compose de lignes de texte ASCII terminées par les caractères `CR/LF` et organisées comme ci-après :

```
requête url-demandé HTTP-version
en-têtes (0 ou plus)
<ligne blanche>
corps de la requête (seulement pour une requête POST)
```

Une réponse du serveur web se présente comme suit :

```
HTTP-version code-réponse phrase-réponse
en-têtes (0 ou plus)
<ligne blanche>
corps de la réponse
```

Les en-têtes de requêtes ou de réponses ont la forme :

```
nom-de-champ: valeur
```

et se classent ainsi :

- en-têtes de requête : `authorization`, `date`, `from`, `if-modified-since`, `location`, `mime-version`, `pragma`, `referer`,`user-agent`
- en-têtes de réponse : `date`, `location`, `mime-version`, `pragma`, `server`, `www.authenticate`
- en-têtes de corps dans les réponses HTTP ou les requêts POST : `allow`, `content-encoding`, `content-length`, `content-type`, `expires`, `last-modified`

Les codes de réponses sont des nombres de 3 chiffres rangés en 5 catégories comme décrits dans la table 2.3.

Tableau: Codes de réponses de HTTP

code	description
1yz	non utilisé
	succès
200	OK, requête réussie
201	OK, nouvelle ressource créée (commande POST)
202	requête acceptée mais traitement incomplet
204	OK, mais pas de contenu à envoyer
	redirection (à gérer par le client)
301	le document demandé a été définitivement déplacé vers une autre url
302	le document demandé a été temporairement déplacé vers une autre url

304	le document n'a pas changé (dans le cas d'un GET conditionnel)
	erreur du client
400	requête mal formulée
401	interdit, la requête nécessite une certification
403	interdit sasn raison sépcifique
404	document non trouvé
	erreur du serveur
500	erreur interne du serveur
501	non implanté
502	mauvaise passerelle, réponse invalide d'une passerelle
503	service temorairement indisponible

Ci -dessous est décrit un exemple de requête et réponse HTTP, après s'être connecté à un serveur web, par exemple avec un client telnet.

```
helios|~>telnet www.yahoo.fr 80
Trying 195.67.49.47...
Connected to www.yahoo.fr.
Escape character is '^]'.
get / http/1.0

HTTP/1.0 200 OK
Last-Modified: Mon, 26 Oct 1998 19:13:02 GMT
Content-Type: text/html
Content-Length: 13163

<head>
<title>Yahoo! France</title>
<base href="http://www.yahoo.fr/">
</head>
<body>
....
</body>
</html>
```

2.8 Outils communs d'utilisation d'un réseau sous Unix.

Le but de cette section est de décrire les principaux fichiers de configuration et les principales commandes relatives à l'utilisation du réseau Internet depuis une machine unix. Elle sert de support à des manipulations devant ordinateur. On ne décrit pas ici les commandes liées aux services Internet les plus classiques : mail, ftp, web, news... qui sont présentées par ailleurs. Les informations données ici sont relatives à la machine `vega.info.univ-angers.fr` *(un Linux BiPentium Pro, 200Mhz, 256 Mo) fin novembre 1997.*

Sous-sections

2.8.1 Fichiers de configuration.

- `/etc/hosts`

  Il est structuré sous la forme

  adresse IP nom de machine liste d'alias commentaire

```
127.0.0.1               localhost
193.49.162.1            vega.info.univ-angers.fr vega
193.49.162.2            sirius.info.univ-angers.fr sirius
193.49.162.10           moinefou.info.univ-angers.fr moinefou
193.49.162.4            helios.info.univ-angers.fr helios
```

  Il assure la correspondance (nom, adresse IP). S'il existe un serveur de noms sur cette machine, il contient très peu de lignes et ne sert qu'au démarrage de la machine avant que le serveur de noms ne soit lancé. Si le serveur de noms est sur une machine distante il ne contient que les machines du réseau local.

  Celui de la machine moinefou se présente de la manière suivante

```
127.0.0.1       localhost       loopback
193.49.144.1    lagaffe
193.49.144.220  gw-info
193.49.162.1    vega.info.univ-angers.fr vega
193.49.162.2    sirius.info.univ-angers.fr sirius
193.49.162.10   moinefou
193.49.162.4    helios
```

  dans lequel est spécifiée l'adresse d'une passerelle (*gateway* d'adresse IP qui se termine par .220) à laquelle est renvoyé tout paquet qui n'appartient pas au réseau local `193.49.162.0`.

- `/etc/resolv.conf`
- `domain info.univ-angers.fr`
- `search info.univ-angers.fr univ-angers.fr etud.univ-angers.fr`
- `nameserver 193.49.162.9`
- `nameserver 193.49.144.1`

  Il précise le domaine d'appartenance de la machine, les noms de domaines avec lesquels compléter le nom d'une machine pour connaître son nom complet, ainsi que les adresses de serveurs de noms à interroger.

- `/etc/protocols`
- `ip      0       IP      # internet protocol, pseudo protocol number`
- `icmp    1       ICMP    # internet control message protocol`
- `igmp    2       IGMP    # internet group multicast protocol`
- `ggp     3       GGP     # gateway-gateway protocol`
- `tcp     6       TCP     # transmission control protocol`
- `pup     12      PUP     # PARC universal packet protocol`
- `udp     17      UDP     # user datagram protocol`
- `idp     22      IDP     # WhatsThis?`
- `raw     255     RAW     # RAW IP interface`

  Il contient la liste des protocoles connus et utilisés dans Internet sous la forme

  nom du protocole numéro du protocole liste d'alias commentaire

- `/etc/services`
- `                .....`
- `netstat         15/tcp`
- `qotd            17/tcp          quote`
- `chargen         19/tcp          ttytst source`
- `chargen         19/udp          ttytst source`
- `ftp-data        20/tcp`

```
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
                .....
```

Il contient la liste des services Internet connus sous la forme

nom du service numéro de port/protocole liste d'alias commentaire

- `/etc/inetd.conf`

```
                .....
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
gopher  stream  tcp     nowait  root    /usr/sbin/tcpd  gn
                .....
```

Il contient les liens entre nom de services et «exécutables» réalisant ce service.

2.8.2 Quelques commandes utiles

Les commandes sont illustrées avec leur résultats après les avoir lancées sur `vega.info.univ-angers.fr`. Il est conseillé d'utiliser la commande `man` du système pour obtenir de plus amples informations sur celles-ci.

- `hostname` retourne le nom de la machine.

- `rusers` (`rwho` est une commande similaire) renvoie la liste des utilisateurs connectés sur les machines du réseau local.

- `finger` renvoie des informations sur un utilisateur, en utilisant s'ils existent les fichiers .plan et .project.

- `ping` permet de tester l'accessibilité d'une machine. Elle envoie une requête ICMP (echo) à destination d'une machine cible, spécifiée par son nom ou son adresse IP, qui lui retourne une réponse ICMP (echo). Si une machine ne répond pas au ping, elle est inutilisable pour toute autre application.

```
|vega|~>ping babinet
PING babinet.univ-angers.fr (193.49.163.20): 56 data bytes
64 bytes from 193.49.163.20: icmp_seq=0 ttl=254 time=1.7 ms
64 bytes from 193.49.163.20: icmp_seq=1 ttl=254 time=1.7 ms
64 bytes from 193.49.163.20: icmp_seq=2 ttl=254 time=1.6 ms

--- babinet.univ-angers.fr ping statistics ---
3 packets transmitted, 3 packets received, 0\% packet loss
round-trip min/avg/max = 1.6/1.6/1.7 ms
```

  Ping renvoie également le numéro de la séquence ICMP exécutée, la valeur du champ TTL et le temps d'aller-retour entre les deux machines. Elle peut le faire car dans le paquet expédié pour la requête elle place son heure d'émission et lorsqu'elle reçoit la réponse elle la soustrait de l'heure système. Puis, en fin d'exécution (provoquée par l'utilisateur ou suite à l'envoi du nombre de paquets spécifiés), quelques données statistiques sont affichées permettant d'évaluer la qualité de la liaison.

- `traceroute` renvoie la route prise par des paquets pour atteindre une destination. Elle utilise le champ TTL (Time to Live) des paquets IP transmis selon le protocole UDP. Ce champ est décrémenté d'une unité à chaque traversée de routeur, et normalement devrait être décrémenté également quand il stationne trop longtemps dans un routeur. Quand un routeur reçoit un paquet IP avec un *TTL*=1 (ou 0), il le détruit et renvoie le message ICMP *time exceeded* à l'expéditeur. Ce message ICMP contient l'adresse IP du routeur qui le produit. Pour connaître le chemin reliant une machine A à une machine B, le programme traceroute de A envoie donc à destination de B un premier paquet IP avec un champ TTL égal à 1. Ainsi le premier routeur lui renvoie un paquet ICMP avec son adresse. Traceroute renvoie alors un 2^{e}paquet à destination de B avec un *TTTL*=2, celui-ci sera refusé par le 2^{e}routeur, et ainsi de suite traceroute augmente le champ TTL jusqu'à avoir traversé tous les routeurs séparant A de B. Comme les messages ICMP contiennent les adresses des routeurs il suffit de les visualiser à chaque retour de message. Bien qu'en théorie 2 paquets se suivant ne prennent pas forcément la même route, ils le font la plupart du temps, c'est pourquoi traceroute donne des informations fiables, en tous les cas au moment où on fait appel à lui.

```
|vega|~>traceroute www.sciences.univ-nantes.fr
traceroute to www.sciences.univ-nantes.fr (193.52.109.12), 30 hops
max, 40 byte packets
 1  gw-maths.net.univ-angers.fr (193.49.162.220)  2.882 ms  3.246 ms
3.034 ms
 2  193.52.254.254 (193.52.254.254)  4.638 ms  2.239 ms  2.298 ms
 3  gw-ft.net.univ-angers.fr (193.49.161.1)  2.353 ms  2.419 ms
2.627 ms
 4  angers.or-pl.ft.net (193.55.153.41)  4.027 ms  4.117 ms  13.75 ms
 5  nantes.or-pl.ft.net (193.55.153.9)  8.747 ms  10.96 ms  7.499 ms
```

- 6 u-sciences-nantes.or-pl.ft.net (193.54.136.138) 14.468 ms 9.77 ms 16.273 ms
- 7 193.52.96.2 (193.52.96.2) 15.29 ms 13.83 ms 12.026 ms
- 8 www.sciences.univ-nantes.fr (193.52.109.12) 12.149 ms 9.715 ms 20.812 ms

30 hops signifie que l'on ne fera pas plus de 30 sauts. 40 octets pour le datagramme IP correspondent à 20 octets pour l'en-tête IP, 8 octets pour l'en-tête UDP, 12 octets de données utilisateurs dont une copie du TTL et l'heure d'émission.

On obtient en résultat le TTL, le nom et l'adresse IP des routeurs intermédaires (et de la machine destinatrice). Puis, comme pour chaque valeur de TTL, traceroute envoie 3 datagrammes IP, on obtient les temps d'aller (du paquet IP) et retour (du paquet ICMP). On peut donc connâtre le temps de transmission entre le routeur N et N+1 en faisnat la différence des temps retournés sur les lignes N+1 et N. Si on a une * c'est qu'il n'y a pas eu de réponses dans les 5 secondes pour le paquet concerné. Si on a toute une ligne d'*, c'est que les paquets ICMP de retour ne sont pas parvenus à la machine A, par exemple car il ont été émis par B avec un TTL trop faible, ou qu'il ont tous mis plus de 5 sec à revenir.

- `arp` permet de visualiser et modifier (si on a le droit) la table de translation adresses Ethernet/adresses Internet (pour plus de détails consulter la section 2.4.4). Cette table est en fait un cache qui évolue au fur et à mesure des sollicitations des machines du réseau.

- `|vega|~>arp -a`
- `Address HWtype HWaddress Flags Mask Iface`
- `tx-fb.info.univ-angers. ether 08:00:11:06:98:77 C * eth0`
- `tektro1.info.univ-anger ether 08:00:11:03:31:2D C * eth0`
- `tx-bd.info.univ-angers. ether 08:00:11:06:97:21 C * eth0`
- `sirius.info.univ-angers ether 00:20:AF:BB:BB:6A C * eth0`
- `helios.info.univ-angers ether 08:00:20:88:0F:4E C * eth0`
- `kitsch.info.univ-angers ether 08:00:09:6D:AE:6C C * eth0`
- `moinefou.info.univ-ange ether 08:00:09:70:14:A3 C * eth0`
- `tektro10.info.univ-ange ether 08:00:11:03:31:35 C * eth0`
- `gw-maths.net.univ-anger ether 00:20:DA:78:F9:39 C * eth0`
- `tx-pn.info.univ-angers. ether 08:00:11:03:EC:2C C * eth0`

- `nslookup` permet d'interroger les serveurs de noms d'Internet. Elle fonctionne en mode interactif quand on l'appelle sans argument ou avec les arguments – `nom ou adresse IP de machine`, à ce moment-là on obtient

- `|vega|~>nslookup`
- `Default Server: kitsch.info.univ-angers.fr`
- `Address: 193.49.162.9`
-
- `>`

où l'on a le nom du serveur de noms par défaut. Dans le mode interactif on obtient de l'aide sur les différentes commandes à l'aide de la commande `?`.

```
> ?
$Id: nslookup.help,v 8.3 1996/08/05 08:31:39 vixie Exp $
```

```
Commands:       (identifiers are shown in uppercase, [] means
optional)
NAME            - print info about the host/domain NAME using default
server
NAME1 NAME2     - as above, but use NAME2 as server
help or ?       - print info on common commands; see nslookup(1) for
details
set OPTION      - set an option
    all         - print options, current server and host
    [no]debug   - print debugging information
    [no]d2      - print exhaustive debugging information
    [no]defname - append domain name to each query
    [no]recurse - ask for recursive answer to query
    [no]vc      - always use a virtual circuit
    domain=NAME - set default domain name to NAME
    srchlist=N1[/N2/.../N6] - set domain to N1 and search list to
N1,N2, etc.
    root=NAME   - set root server to NAME
    retry=X     - set number of retries to X
    timeout=X   - set initial time-out interval to X seconds
    querytype=X - set query type, e.g.,
A,ANY,CNAME,HINFO,MX,PX,NS,PTR,SOA,TXT,WKS
    port=X      - set port number to send query on
    type=X      - synonym for querytype
    class=X     - set query class to one of IN (Internet), CHAOS,
HESIOD or ANY
server NAME     - set default server to NAME, using current default
server
lserver NAME    - set default server to NAME, using initial server
finger [USER]   - finger the optional USER at the current default
host
root            - set current default server to the root
ls [opt] DOMAIN [> FILE] - list addresses in DOMAIN (optional: output
to FILE)
    -a          -  list canonical names and aliases
    -h          -  list HINFO (CPU type and operating system)
    -s          -  list well-known services
    -d          -  list all records
    -t TYPE     -  list records of the given type (e.g., A,CNAME,MX,
etc.)
view FILE       - sort an 'ls' output file and view it with more
exit            - exit the program, ^D also exits
```

- whois permet d'interroger une base de données sur les réseaux et leurs administrateurs. Par défaut le serveur `rs.internic.net` est interrogé, mais on peut spécifier quel serveur whois on interroge, comme par exemple dans la commande suivante.

- `|vega|~>whois univ-angers.fr@whois.ripe.net`
- `[bsdbase.ripe.net]`
-
- `\% Rights restricted by copyright. See http://www.ripe.net/db/dbcopyright.html`
-
- `domain:      univ-angers.fr`
- `descr:       Universite d'Angers`
- `descr:       30, rue des Arenes`
- `descr:       49035 Angers CEDEX 01, France`
- `admin-c:     Jacques Allo`
- `tech-c:      Olivier Girard`
- `tech-c:      Jean-Marie Chretien`

- zone-c: AR41
- nserver: lagaffe.univ-angers.fr 193.49.144.1
- nserver: biotheo.ibt.univ-angers.fr 193.49.145.60
- nserver: resone.univ-rennes1.fr
- dom-net: 193.49.144.0 193.49.145.0 193.49.146.0
- mnt-by: FR-NIC-MNT
- changed: Vincent.Gillet@inria.fr 970219
- source: RIPE
- ..... etc ....

  On peut aussi utiliser `whois nom_de_personne@nom_de_serveur`.

- `netstat` permet d'obtenir des statistiques sur le nombre de paquets, les erreurs, les collisions etc... sur une interface en donnant la liste de toutes les sockets ouvertes.

- |vega|~>netstat -e|more
- Active Internet connections (w/o servers)
- Proto Recv-Q Send-Q Local Address Foreign Address State User
- tcp 0 0 vega.info.univ-ang:1023 helios.info.univ-an:993 ESTABLISHED root
- tcp 0 0 vega.info.univ-ang:1022 helios.info.univ-a:1016 ESTABLISHED root
- tcp 0 0 vega.info.univ-ang:1021 moinefou.info.uni:login ESTABLISHED frantz
- tcp 57 0 vega.info.univ-ang:1632 istia.istia.univ-an:ftp CLOSE_WAIT frantz
- ..... etc ....

  On obtient la table de routage par

  ```
  |vega|~>netstat -rn
  Kernel IP routing table
  Destination     Gateway         Genmask         Flags   MSS Window
  irtt Iface
  193.49.162.0    0.0.0.0         255.255.255.0   U      1500 0
  0 eth0
  127.0.0.0       0.0.0.0         255.0.0.0       U      3584 0
  0 lo
  0.0.0.0         193.49.162.220  0.0.0.0         UG     1500 0
  0 eth0
  ```

  U route utilisable, G route indirecte

- `tcpdump` visualise différentes informations (selon les options de la commande) sur les paquets qui passent par l'interface réseau de la machine. C'est une commande réservée à `root`.

  Ci-dessous on a les 20 derniers paquets qui ont transité.

  ```
  tcpdump -c 20
  tcpdump: listening on eth0
  12:41:15.069542 vega.info.univ-angers.fr.syslog > sirius.info.univ-
  angers.fr.syslog: udp 98
  12:41:15.079542 vega.info.univ-angers.fr.1164 > tx-pn.info.univ-
  angers.fr.6000: . ack 965632022 win 31744
  12:41:15.079542 vega.info.univ-angers.fr.6000 > helios.info.univ-
  angers.fr.47589: . ack 504866793 win 31744
  12:41:15.069542 theo.maisel.int-evry.fr.6665 > vega.info.univ-
  angers.fr.1951: . 59767366:59768826(1460)
  ack 579761084 win 8760 (DF)
  12:41:15.079542 vega.info.univ-angers.fr.1951 > theo.maisel.int-
  evry.fr.6665: . ack 4294963200 win 31744 [tos 0x8]
  12:41:15.079542 vega.info.univ-angers.fr.1164 > tx-pn.info.univ-
  angers.fr.6000: P 0:128(128) ack 1 win 31744 (DF)
  ```

```
12:41:15.079542 theo.maisel.int-evry.fr.6665 > vega.info.univ-
angers.fr.1951: P 1460:2636(1176) ack 1 win 8760 (DF)
12:41:15.079542 vega.info.univ-angers.fr.1951 > theo.maisel.int-
evry.fr.6665: . ack 4294963200 win 31744 [tos 0x8]
12:41:15.079542 vega.info.univ-angers.fr.1953 > kitsch.info.univ-
angers.fr.domain: 53323+ (43)
12:41:15.079542 vega.info.univ-angers.fr.nfs > moinefou.info.univ-
angers.fr.131f35e: reply ok 128
12:41:15.089542 vega.info.univ-angers.fr.1909 > tx-is.info.univ-
angers.fr.6000: . ack 720241486 win 31744
12:41:15.089542 kitsch.info.univ-angers.fr.domain > vega.info.univ-
angers.fr.1953: 53323* 1/2/2 (181)
12:41:15.089542 vega.info.univ-angers.fr.1954 > kitsch.info.univ-
angers.fr.domain: 53324+ (43)
12:41:15.089542 moinefou.info.univ-angers.fr.131f35f >
vega.info.univ-angers.fr.nfs: 144 getattr [|nfs]
12:41:15.089542 vega.info.univ-angers.fr.nfs > moinefou.info.univ-
angers.fr.131f35f: reply ok 96 getattr [|nfs]
12:41:15.089542 moinefou.info.univ-angers.fr.131f360 >
vega.info.univ-angers.fr.nfs: 152 readdir [|nfs]
12:41:15.089542 kitsch.info.univ-angers.fr.domain > vega.info.univ-
angers.fr.1954: 53324* 1/2/2 (179)
12:41:15.089542 vega.info.univ-angers.fr.nfs > moinefou.info.univ-
angers.fr.131f360: reply ok 376 readdir [|nfs]
12:41:15.089542 vega.info.univ-angers.fr.1164 > tx-pn.info.univ-
angers.fr.6000: P 128:256(128) ack 1 win 31744 (DF)
12:41:15.089542 vega.info.univ-angers.fr.1956 > kitsch.info.univ-
angers.fr.domain: 53325+ (44)
```

`tcpdump tcp port 21` ne renverra que les paquets liés au protocole tcp sur le port 21, à savoir ftp.

`tcpdump arp` ne renverra que les paquets liés au protocole arp.

3. Langages pour le web.

Ce chapitre est traité lors de travaux dirigés et travaux pratiques et explore différentes techniques utiles à la conception de sites webs.

- html
- javascript
- applets java
- cgi perl et php pour accéder aux bases de données