Cours de réseaux
Maîtrise d'informatique
Université d'Angers

Pascal Nicolas
U.F.R. Sciences de l'Université d'Angers
E-Mail : pascal.nicolas@univ-angers.fr
Web : www.info.univ-angers.fr/pub/pn

Préface

Ceci est le support du cours de réseaux de la maîtrise d'informatique de l'Université d'Angers (année 1999/2000). Le chapitre 1 traite des transmissions de données et des réseaux en général sous l'angle de l'architecture des systèmes ouverts (modèle OSI). Le chapitre 2 traite du réseau Internet et des protocoles qui lui sont associés. Les chapitres 3 et 4 sont plus orientés vers des applications pratiques visant la mise en place de services sur les réseaux. Ils ne sont pas détaillés dans ce document car ils sont plus particulièrement étudiés en TD et TP.

Les principales références bibliographiques en français sont :

- Guy Pujolle - *Les réseaux.* - Eyrolles.
- Patrice Rolin, et al. - *Les réseaux, principes fondamentaux.* - Hermès.
- Douglas Comer - *TCP/IP, architectures, protocoles et applications.* - Interéditions.
- W. Richard Stevens - *TCP/IP illustré - Vol 1,2,3.* - International Thomson Publishing, France.

Tous mes remerciements à mes collègues enseignants ou ingénieurs de l'Université d'Angers : Serge Tahé, Jacques Allo et Stéphane Vincendeau, pour leur aide à la mise en place de ce cours.

Table des matières

Chapitre 1

Principe des réseaux numériques.

Dans ce chapitre nous aborderons les grands principes régissant les équipements matériels et logiciels permettant d'échanger des données mises sous forme numérique et qui forment les réseaux informatiques.

1.1 Introduction.

Les réseaux informatiques qui permettaient à leur origine de relier des terminaux passifs à de gros ordinateurs centraux autorisent à l'heure actuelle l'interconnexion de tous types, d'ordinateurs que ce soit de gros serveurs, des stations de travail, des ordinateurs personnels ou de simples terminaux graphiques. Les services qu'ils offrent font partie de la vie courante des entreprises et administrations (banques, gestion, commerce, bases de données, recherche, etc...) et des particuliers (messagerie, loisirs, services d'informations par minitel et Internet ...).

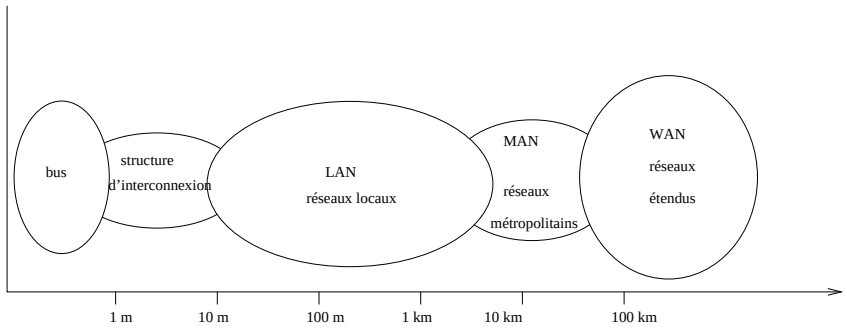

FIG. 1.1 - *Classification des réseaux informatiques selon leur taille.*

On peut faire une première classification des réseaux à l'aide de leur taille comme on peut le voir dans la figure 1.1.

Les bus que l'on trouve dans un ordinateur pour relier ses différents composants (mémoires, périphériques d'entrée-sortie, processeurs, ...) peuvent être considérés comme des réseaux dédiés à des tâches très spécifiques.

Les structures d'interconnexion sont des réseaux de très haut débits, mais de faible étendue, et regroupent les pré et post-processeurs des ordinateurs vectoriels par exemple. En effet l'usage d'un super-calculateur (Cray notamment) nécessite un ordinateur, dit frontal, qui lui prépare les données et recueille les résultats.

Un *réseau local* (Local Area Network) peut s'étendre de quelques mètres à quelques kilomètres et correspond au réseau d'une entreprise. Il peut se développer sur plusieurs bâtiments et permet de satisfaire tous les besoins internes de cette entreprise.

Un *réseau métropolitain* (Metropolitan Area Network) interconnecte plusieurs lieux situés dans une même vile, par exemple les différents sites d'une université ou d'une administration, chacun possédant son propre réseau local.

Un *réseau étendu* (Wide Area Network) permet de communiquer à l'échelle d'un pays, ou de la planète entière, les infrastructures physiques pouvant être terrestres ou spatiales à l'aide de satellites de télécommunications.

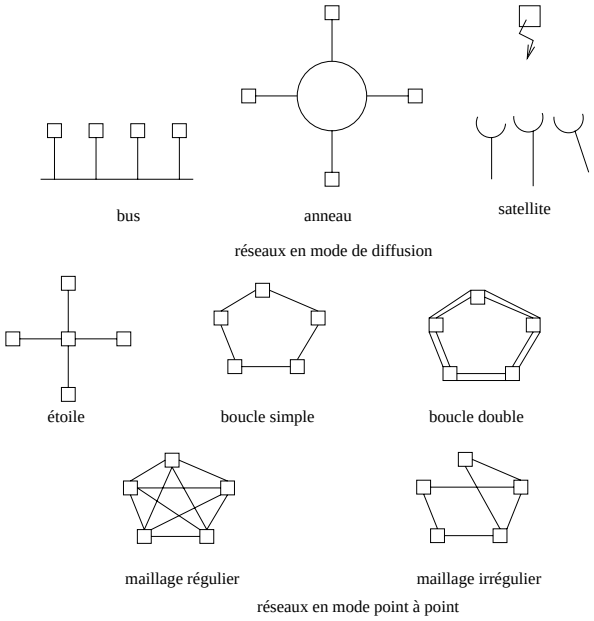

FIG. 1.2 - *Topologie des réseaux informatiques.*

On peut également différencier les réseaux selon leur structure ou plus précisément leur topologie comme illustré dans la figure 1.2. On y distingue ainsi deux classes de réseaux :

- ceux en *mode de diffusion*
- ceux en *mode point à point*

Le premier mode de fonctionnement consiste à partager un seul support de transmission. Chaque message [1] envoyé par un équipement sur le réseau est reçu par tous les autres. C'est l'adresse spécifique placée dans le message qui permettra à chaque équipement de déterminer si le message lui est adressé ou non. À tout moment un seul équipement à le droit d'envoyer un message sur le support, il faut donc qu'il «écoute» au préalable si la voie est libre; si ce n'est pas le cas il attend selon un protocole spécifique à chaque architecture. Les réseaux locaux adoptent pour la plupart le mode diffusion sur une architecture en bus ou en anneau et les réseaux satellitaires ou radio suivent également ce mode de communication. Dans une telle configuration la rupture du support provoque l'arrêt du réseau, par contre la panne d'un des éléments ne provoque pas (en général) la panne globale du réseau.

Dans le mode diffusion point à point le support physique (le câble) relie une paire d'équipements seulement. Quand deux éléments non directement connectés entre eux veulent communiquer ils le font par l'intermédiaire des autres nœuds du réseau.

Dans le cas de l'*étoile* le site central reçoit et envoie tous les messages, le fonctionnement est simple, mais la panne du nœud central paralyse tout le réseau

Dans une *boucle simple*, chaque nœud recevant un message de son voisin en amont le réexpédie à son voisin en aval. Pour que les messages ne tournent pas indéfiniment le nœud émetteur retire le message

1. Terme laissé vague pour l'instant.

lorsqu'il lui revient. Si l'un des éléments du réseau tombe en panne, alors tout s'arrête. Ce problème est partiellement résolu par la *double boucle* dont chacune des boucles fait tourner les messages dans un sens opposé. En cas de panne d'un équipement, on reconstitue une boucle simple avec les éléments actifs des deux boucles, mais dans ce cas tout message passera deux fois par chaque nœud. Il en résulte alors une gestion très complexe.

Dans le *maillage régulier* l'interconnexion est totale ce qui assure une fiabilité optimale du réseau, par contre c'est une solution coûteuse en câblage physique. Si l'on allège le plan de câblage, le maillage devient *irrégulier* et la fiabilité peut rester élevée mais elle nécessite un routage des messages selon des algorithmes parfois complexes. Dans cette architecture il devient presque impossible de prévoir le temps de transfert d'un nœud à un autre.

Quelle que soit l'architecture physique d'un réseau on trouve deux modes de fonctionnement différents :

- *avec connexion*
- *sans connexion*

Dans le mode avec connexion, toute communication entre deux équipements suit le processus suivant:

1. l'émetteur demande l'établissement d'une connexion par l'envoi d'un bloc de données spécial
2. si le récepteur (ou le gestionnaire de service) refuse cette connexion la communication n'a pas lieu
3. si la connexion est acceptée, elle est établie par mise en place d'un circuit virtuel dans le réseau reliant l'émetteur au récepteur
4. les données sont ensuite transférées d'un point à l'autre
5. la connexion est libérée

C'est le fonctionnement bien connu du réseau téléphonique classique. Les avantages du mode avec connexion sont la sécurisation du transport par identification claire de l'émetteur et du récepteur, la possibilité d'établir à l'avance des paramètres de qualité de service qui seront respectés lors de l'échange des données. Les défauts sont la lourdeur de la mise en place de la connexion qui peut se révéler beaucoup trop onéreuse si l'on ne veut échanger que quelques octets ainsi que la difficulté à établir des communications multipoint.

Dans le mode sans connexion les blocs de données, appelés *datagrammes*, sont émis sans vérifier à l'avance si l'équipement à atteindre, ainsi que les nœuds intermédiaires éventuels, sont bien actifs. C'est alors aux équipements gérant le réseau d'acheminer le message étape par étape et en assurant éventuellement sa temporisation jusqu'à ce que le destinataire soit actif. Ce service est celui du courrier postal classique et suit les principes généraux suivants:

- le client poste une lettre dans une boîte aux lettres
- chaque lettre porte le nom et l'adresse du destinataire
- chaque client a une adresse propre et une boîte aux lettres
- le contenu de l'information reste inconnu du prestataire de service
- les supports du transport sont inconnus de l'utilisateur du service

D'autre part il existe plusieurs types de commutation dont les principaux sont :

- la *commutation de circuits* : c'est historiquement la première à avoir été utilisée, par exemple dans le réseau téléphonique à l'aide des auto-commutateurs. Elle consiste à créer dans le réseau

un circuit particulier entre l'émetteur et le récepteur avant que ceux-ci ne commencent à échanger des informations. Ce circuit sera propre aux deux entités communiquant et il sera libéré lorsque l'un des deux coupera sa communication. Par contre, si pendant un certain temps les deux entités ne s'échangent rien le circuit leur reste quand même attribué. C'est pourquoi, un même circuit (ou portion de circuit) pourra être attribué à plusieurs communications en même temps. Cela améliore le fonctionnement global du réseau mais pose des problèmes de gestion (files d'attente, mémorisation,...)

- la *commutation de messages* : elle consiste à envoyer un message [2] de l'émetteur jusqu'au récepteur en passant de nœud de commutation en nœud de commutation. Chaque nœud attend d'avoir reçu complétement le message avant de le réexpédier au nœud suivant. Cette technique nécessite de prévoir de grandes zones tampon dans chaque nœud du réseau, mais comme ces zones ne sont pas illimitées il faut aussi prévoir un contrôle de flux des messages pour éviter la saturation du réseau. Dans cette approche il devient très difficile de transmettre de longs messages. En effet, comme un message doit être reçu entièrement à chaque étape si la ligne a un taux d'erreur de 10^{-5} par bit (1 bit sur 10^5 est erroné) alors un message de 100000 octets n'a qu'une probabilité de 0,0003 d'être transmis sans erreur.

- la *commutation de paquets* : elle est apparue au début des années 70 pour résoudre les problèmes d'erreur de la commutation de messages. Un message émis est découpé en paquets [3] et par la suite chaque paquet est commuté à travers le réseau comme dans le cas des messages. Les paquets sont envoyés indépendamment les uns des autres et sur une même liaison on pourra trouver les uns derrière les autres des paquets appartenant à différents messages. Chaque nœud redirige chaque paquet vers la bonne liaison grâce à une table de routage. La reprise sur erreur est donc ici plus simple que dans la commutation de messages, par contre le récepteur final doit être capable de reconstituer le message émis en réassemblant les paquets. Ceci nécessitera un protocole particulier car les paquets peuvent ne pas arriver dans l'ordre initial, soit parce qu'ils ont emprunté des routes différentes, soit parce que l'un d'eux a du être réémis suite à une erreur de transmission.

- la *commutation de cellules* : une cellule est un paquet particulier dont la taille est toujours fixée à 53 octets (5 octets d'en-tête et 48 octets de données). C'est la technique de base des réseaux hauts débits ATM (Asynchronous Transfert Mode) qui opèrent en mode connecté où avant toute émission de cellules, un chemin virtuel est établi par lequel passeront toutes les cellules. Cette technique mixe donc la commutation de circuits et la commutation de paquets de taille fixe permettant ainsi de simplifier le travail des commutateurs pour atteindre des débits plus élevés.

1.2 Le modèle de référence OSI de l'ISO.

Au début des années 70, chaque constructeur a développé sa propre solution réseau autour d'architecture et de protocoles privés (SNA d'IBM, DECnet de DEC, DSA de Bull, TCP/IP du DoD,...) et il s'est vite avéré qu'il serait impossible d'interconnecter ces différents réseaux «propriétaires» si une norme internationale n'était pas établie. Cette norme établie par l'*International Standard Organization* [4] (ISO) est la norme *Open System Interconnection* (OSI, interconnexion de systèmes ouverts). Un système ouvert est un ordinateur, un terminal, un réseau, n'importe quel équipement respectant cette norme et donc apte à échanger des informations avec d'autres équipements hétérogènes et issus de constructeurs différents.

2. Un message est une suite d'informations formant un tout, par exemple un fichier ou une ligne de commande tapée au clavier d'un ordinateur.

3. Un paquet est une suite d'octets, dont le contenu n'a pas forcément une signification et ne pouvant pas dépasser une taille fixée par avance.

4. L'ITU-T, *International Telecommunication Union, Telecommunication Sector*, anciennement CCITT est également un organisme de normalisation des télécommunications. Il dépend de l'ONU et est basé à Genève. Information à `http://www.itu.int`

7 Application
6 Présentation
5 Session
4 Transport
3 Réseau
2 Liaison
1 Physique

FIG. 1.3 - *Les sept couches du modèle de référence OSI de l'ISO.*

Le premier objectif de la norme OSI a été de définir un modèle de toute architecture de réseau basé sur un découpage en *sept couches* (cf figure 1.3), chacune de ces couches correspondant à une fonctionnalité particulière d'un réseau. Les couches 1, 2, 3 et 4 sont dites *basses* et les couches 5, 6 et 7 sont dites *hautes*.

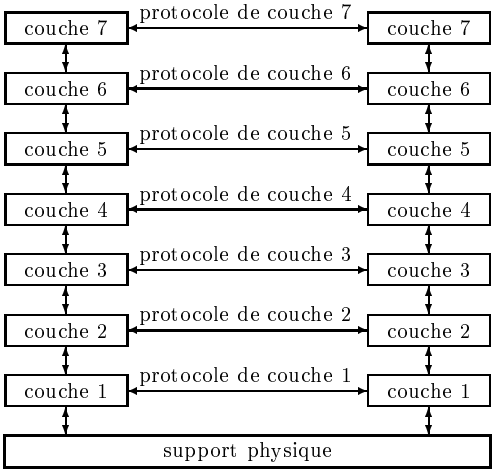

FIG. 1.4 - *Communication entre couches.*

Chaque couche est constituée d'éléments matériels et logiciels et offre un service à la couche située immédiatement au-dessus d'elle en lui épargnant les détails d'implémentation nécessaires. Comme illustré dans la figure 1.4, chaque couche n d'une machine gère la communication avec la couche n d'une autre machine en suivant un *protocole* de niveau n qui est un ensemble de règles de communication pour le service de niveau n.

En fait, aucune donnée n'est transférée directement d'une couche n vers une autre couche n, mais elle l'est par étapes successives. Supposons un message à transmettre de l'émetteur A vers le récepteur B. Ce message, généré par une application de la machine A va franchir les couches successives de A via les interfaces qui existent entre chaque couche pour finalement atteindre le support physique. Là, il va transiter via différents nœuds du réseau, chacun de ces nœuds traitant le message via ses couches basses. Puis, quand il arrive à destination, le message remonte les couches du récepteur B via les différentes interfaces et atteint l'application chargée de traiter le message reçu. Ce processus de communication est illustré dans la figure 1.5.

Nous allons maintenant détailler les caractéristiques de chacune de ces couches en précisant d'abord que les fonctions et services définis dans les couches du modèle OSI peuvent se retrouver dans d'autres couches dans les systèmes opérationnels disponibles sur le marché. Il se peut également qu'une fonctionnalité localisée dans une seule couche dans le modèle OSI se retrouve répartie sur plusieurs couches. Mais cela illustre simplement la distance qui existe entre un modèle théorique et des implantations

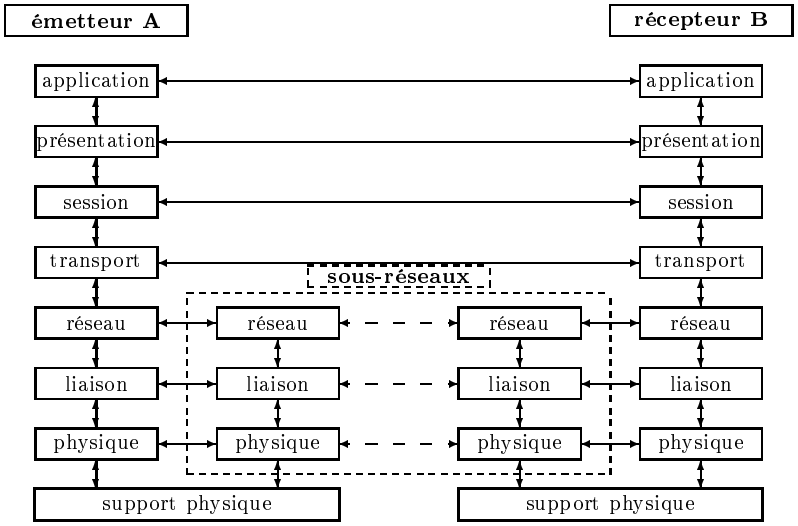

FIG. 1.5 - *Communication entre couches.*

pratiques essayant de suivre ce modèle.

1.3 La couche physique.

Définition 1.3.1 *La couche physique fournit les moyens mécaniques, électriques, fonctionnels et procéduraux nécessaires à l'activation, au maintien et à la désactivation des connexions physiques destinées à la transmission de bits entre deux entités de liaison de données.*

Ici, on s'occupe donc de transmission des bits de façon brute, l'important est que l'on soit sûr que si l'émetteur envoie un bit à 1 alors le récepteur reçoit un bit à 1. Les normes et standards de la couche physique définissent le type de signaux émis (modulation, puissance, portée...), la nature et les caractéristiques des supports (câble, fibre optique...), les sens de transmission...

Tout d'abord une liaison entre 2 équipements A et B peut être *simplex* (unidirectionnelle), dans ce cas A est toujours l'émetteur et B le récepteur. C'est ce que l'on trouve par exemple entre un banc de mesure et un ordinateur recueillant les données mesurées. La communication est *half-duplex* (bidirectionnelle à l'alternat) quand le rôle de A et B peut changer, la communication change de sens à tour de rôle (comme avec des talkies-walkies). Elle est *full-duplex* (bidirectionnelle simultanée) quand A et B peuvent émettre et recevoir en même temps (comme dans le cas du téléphone).

La transmission de plusieurs bits peut s'effectuer en *série* ou en *parallèle*. En série, les bits sont envoyés les uns derrière les autres de manière synchrone ou asynchrone. Dans le mode synchrone l'émetteur et le récepteur se mettent d'accord sur une base de temps (un top d'horloge) qui se répète régulièrement durant tout l'échange. À chaque top d'horloge (ou k tops d'horloge k entier fixé définitivement) un bit est envoyé et le récepteur saura ainsi quand lui arrive les bits. Dans le mode asynchrone, il n'y a pas de négociation préalable mais chaque caractère envoyé est précédé d'un *bit de start* et immédiatement suivi d'un *bit de stop*. Ces deux bits spéciaux servent à caler l'horloge du récepteur pour qu'il échantillonne le signal qu'il reçoit afin d'y décoder les bits qu'il transmet. En parallèle, les bits d'un même caractère sont envoyés en même temps chacun sur un fil distinct, mais cela pose des problèmes de synchronisation et n'est utilisé que sur de courtes distances (bus par exemple).

Quel que soit le mode de transmission retenu, l'émission est toujours cadencée par une horloge dont la vitesse donne le débit de la ligne en *bauds*, c'est-à-dire le nombre de tops d'horloge en une seconde. Ainsi, une ligne d'un débit de 100 bauds autorise 100 émissions par seconde. Si à chaque top d'horloge un signal représentant 0 ou 1 est émis, alors dans ce cas le débit en bit/s est équivalent au débit en *baud*. Cependant, on peut imaginer que le signal émis puisse prendre 4 valeurs distinctes (0, 1, 2, 3)

dans ce cas le signal a une *valence* de 2 et le débit en bit/s est double de celui en *baud*. D'une manière générale, si le signal peut prendre 2^n valeurs distinctes on dit alors que sa valence est de n, ainsi à chaque top n bits peuvent être transmis simultanément et si le débit de la ligne est de **x bauds** il est en fait de **n.x bit/s**.

1.3.1 Transmission en bande de base.

La transmission en bande de base consiste à envoyer directement les suite de bits sur le support à l'aide de *signaux carrés* constitués par un courant électrique pouvant prendre 2 valeurs (5 Volts ou 0 par exemple). On détaillera ci-après les différents codages des bits possibles, mais dans tous les cas l'émetteur envoie sur la ligne un signal carré du type de celui de la figure 1.6 pour la séquence de bits `1010` par exemple.

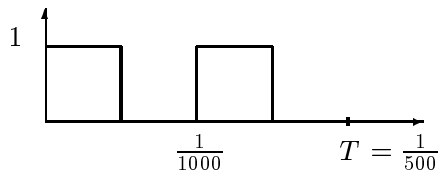

FIG. 1.6 - *Signal carré de la séquence de bits* `1010`.

En considérant ce signal $g(t)$ comme périodique (il suffit de répéter une fois sur $[T..2T]$ le signal donné sur $[0..T]$ pour obtenir un signal périodique sur $[0..2T]$) on peut le décomposer en une *série de Fourier* de la forme

$$g(t) = c/2 + \sum_{n=1}^{\infty} a_n sin(2\pi nft) + \sum_{n=1}^{\infty} b_n cos(2\pi nft)$$

où

$$\begin{aligned} f &= 1/T \text{ la fréquence fondamentale du signal} \\ c &= 2/T \int_0^T g(t)dt \\ a_n &= 2/T \int_0^T g(t)sin(2\pi nft)dt \\ b_n &= 2/T \int_0^T g(t)cos(2\pi nft)dt \end{aligned}$$

On dit que le signal carré est décomposé en une somme infinie d'*harmoniques*, la première étant dénommée *fondamentale*, et cette approximation mathématique permet de savoir quel signal électrique sera réellement reçu au bout du câble.

Cependant, le câble sur lequel est émis le signal possède une *bande passante* qui est l'intervalle des fréquences possibles sur ce support, donc à la réception on ne retrouve pas toute la richesse du signal initial et dans la plupart des cas le signal carré sera très déformé. Par exemple, le câble téléphonique a une bande passante de 300 à 3400 Hz, donc tous les signaux de fréquence inférieure à 300 ou supérieure à 3400 seront éliminées.

Dans notre exemple nous obtenons

$$g(t) = \frac{1}{2} + \frac{2}{\pi}sin(2000\pi t) + \frac{2}{3\pi}sin(6000\pi t) + \frac{2}{5\pi}sin(10000\pi t) + \ldots$$

Dans la figure 1.7 nous trouvons à gauche les 3 premières harmoniques et on remarque que plus la fréquence augmente plus l'amplitude diminue. À droite nous avons le signal réellement perçu par le récepteur si l'on considère que le câble ne laisse passer que ces 3 harmoniques-ci. Dans ce cas le signal reçu reste assez proche du carré émis et le récepteur n'aura pas trop de mal à le décoder.

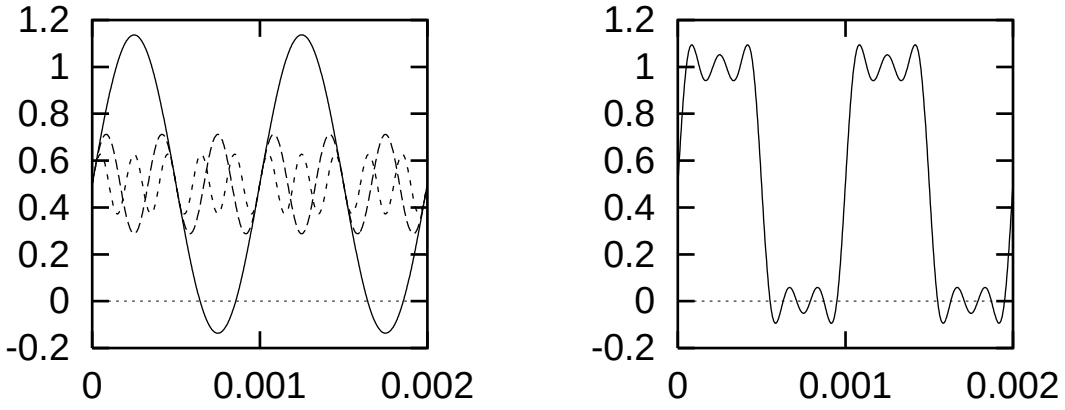

FIG. 1.7 - *Harmoniques et transformée de Fourrier de la séquence de bits 1010.*

Sans entrer dans des détails relevant de la théorie du signal, nous indiquerons simplement que sur une ligne téléphonique dont la bande passante est de 3100Hz et pour un rapport signal/bruit [5] de 10dB on peut atteindre une capacité de 10Kbits/s.

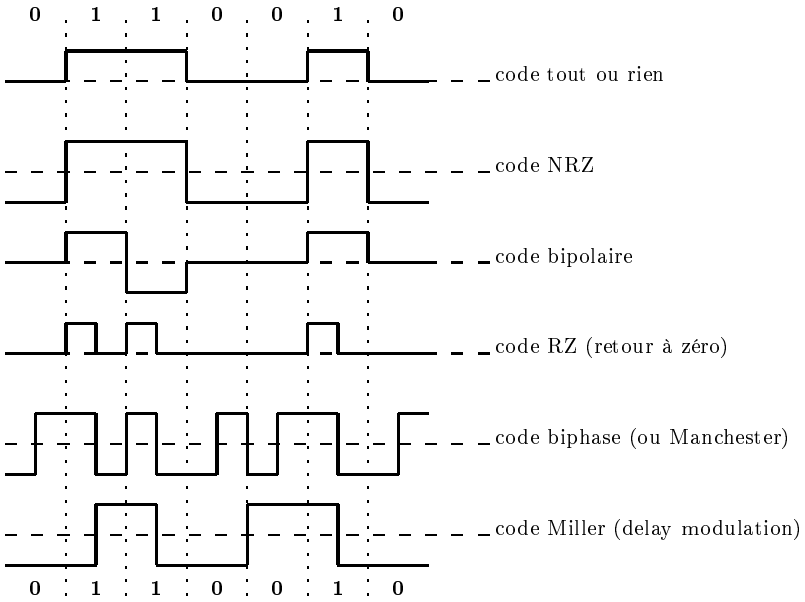

FIG. 1.8 - *Différents codages en bande de base de la séquence* `0110010`.

Dans la figure 1.8 nous trouvons quelques exemple de codage de l'information pour une transmission en bande de base.

- le *code tout ou rien* : c'est le plus simple, un courant nul code le 0 et un courant positif indique le 1
- le *code NRZ* (non retour à zéro): pour éviter la difficulté à obtenir un courant nul, on code le 1 par un courant positif et le 0 par un courant négatif.

5. Rapport de l'énergie du signal émis sur l'énergie du bruit de la ligne. Le bruit étant d'un point de vue mathématique une fonction aléatoire représentant les perturbations (ondes électromagnétiques, défauts dans les composants qui relaient le signal...) que subit la transmission.

- le *code bipolaire* : c'est aussi un code tout ou rien dans lequel le 0 est représenté par un courant nul, mais ici le 1 est représenté par un courant alternativement positif ou négatif pour éviter de maintenir des courants continus.

- le *code RZ* : le 0 est codé par un courant nul et le 1 par un courant positif qui est annulé au milieu de l'intervalle de temps prévu pour la transmission d'un bit.

- le *code Manchester* : ici aussi le signal change au milieu de l'intervalle de temps associé à chaque bit. Pour coder un 0 le courant sera négatif sur la première moitié de l'intervalle et positif sur la deuxième moitié, pour coder un 1, c'est l'inverse. Autrement dit, au milieu de l'intervalle il y a une transition de bas en haut pour un 0 et de haut en bas pour un 1.

- le *code Miller* : on diminue le nombre de transitions en effectuant une transition (de haut en bas ou l'inverse) au milieu de l'intervalle pour coder un 1 et en n'effectuant pas de transition pour un 0 suivi d'un 1. Une transition est effectuée en fin d'intervalle pour un 0 suivi d'un autre 0.

1.3.2 Transmission modulée.

Le principal problème de la transmission en bande de base est la dégradation du signal très rapide en fonction de la distance parcourue, c'est pourquoi elle n'est utilisée qu'en réseau local (<5km). Il serait en effet trop coûteux de prévoir des *répéteurs* pour régénérer régulièrement le signal. C'est pourquoi sur les longues distance on émet un signal sinusoïdal qui, même s'il est affaibli, sera facilement décodable par le récepteur. Ce signal sinusoïdal est obtenu grâce à un *modem* (modulateur-démodulateur)qui est un équipement électronique capable de prendre en entrée un signal en bande de base pour en faire un signal sinusoïdal (modulation) et l'inverse à savoir restituer un signal carré à partir d'un signal sinusoïdal (démodulation). Autrement dit il permet de passer de signaux numériques discrets (0 ou 1) à des signaux analogiques continus.

Il existe trois types de modulation décrits dans la figure 1.9

- la *modulation d'amplitude* envoie un signal d'amplitude différente suivant qu'il faut transmettre un 0 ou un 1. Cette technique est efficace si la bande passante et la fréquence sont bien ajustées. Par contre, il existe des possibilités de perturbation (orage, lignes électriques...), car si un signal de grande amplitude (représentant un 1) est momentanément affaibli le récepteur l'interprétera à tort en un 0.

- la *modulation de fréquence* envoie un signal de fréquence plus élevée pour transmettre un 1. Comme l'amplitude importe peu, c'est un signal très résistant aux perturbations (la radio FM est de meilleure qualité que la radio AM) et c'est assez facile à détecter.

- la *modulation de phase* change la phase du signal (ici de 180°) suivant qu'il s'agit d'un 0 (phase montante) ou d'un 1 (phase descendante).

Dans les exemples donnés ci-dessus on a seulement 2 niveaux possibles à chaque fois, donc on a uniquement la possibilité de coder 2 valeurs différentes à chaque instant, dans ce cas 1 baud = 1bit/s. De manière plus sophistiquée il existe des modems capables de moduler un signal suivant plusieurs niveaux, par exemple 4 fréquences différentes que le modem récepteur saura lui aussi distinguer. Dans ce cas, chaque signal envoyé code 2 bits donc 1 baud = 2bit/s. Il est même possible de transmettre des signaux mêlant les différentes modulations présentées comme dans le cas de la norme V29 qui module à la fois l'amplitude du signal sur 2 niveaux et la phase sur 8 niveaux (0°,45°,...,315°). En combinant les 2 modulations, on obtient ainsi 16 signaux différents possibles à chaque instant, permettant de transmettre simultanément 4 bits à chaque top d'horloge (1 baud = 4 bit/s).

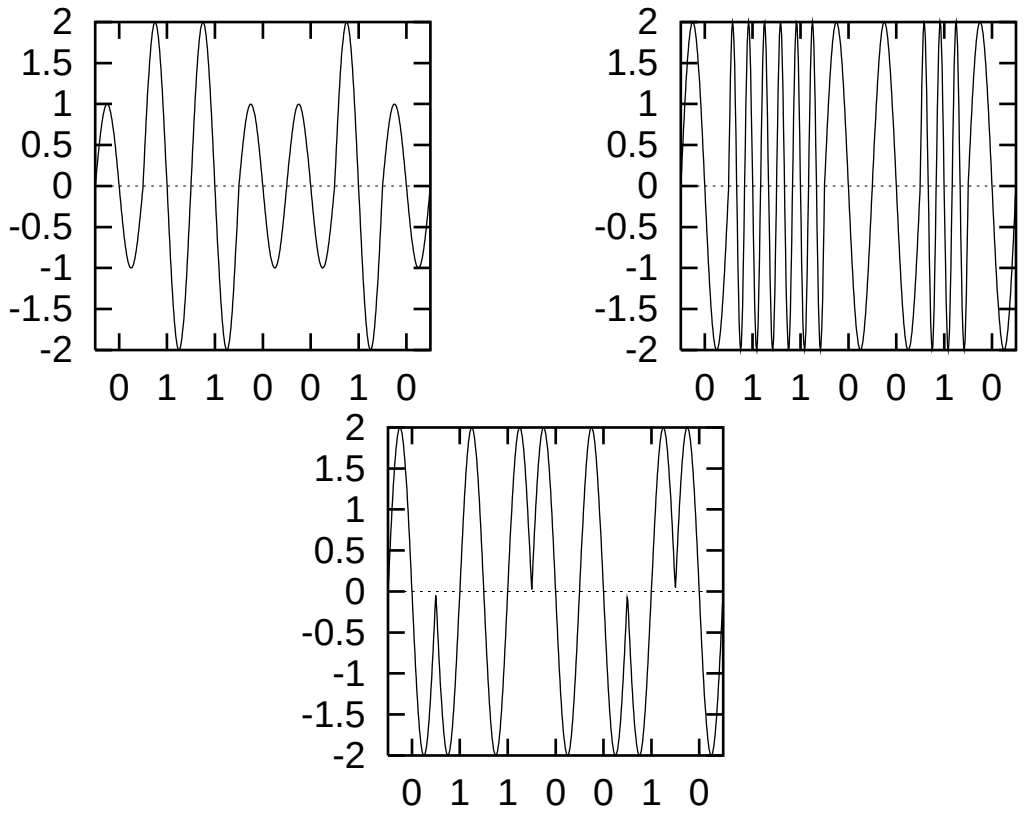

FIG. 1.9 - *Modulations d'amplitude, de fréquence et de phase de la séquence de bits* `0110010`.

1.3.3 Multiplexage.

Le multiplexage consiste à faire transiter sur une seule et même ligne de liaison, dite *voie haute vitesse*, des communications appartenant à plusieurs paires d'équipements émetteurs et récepteurs comme représenté dans la figure 1.10. Chaque émetteur (resp. récepteur) est raccordé à un multiplexeur (resp. démultiplexeur) par une liaison dit *voie basse vitesse*.

Plusieurs techniques sont possibles :

- le *multiplexage fréquentiel* consiste à affecter à chaque voie basse vitesse une bande passante particulière sur la voie haute vitesse en s'assurant qu'aucune bande passante de voie basse vitesse ne se chevauche. Le multiplexeur prend chaque signal de voie basse vitesse et le réemet sur la voie haute vitesse dans la plage de fréquences prévues. Ainsi plusieurs transmissions peuvent être faites simultanément, chacune sur une bande de fréquences particulières, et à l'arrivée le démultiplexeur[6] est capable de discriminer chaque signal de la voie haute vitesse pour l'aiguiller sur la bonne voie basse vitesse.

- le *multiplexage temporel* partage dans le temps l'utilisation de la voie haute vitesse en l'attribuant successivement aux différentes voies basse vitesse même si celles-ci n'ont rien à émettre. Suivant les techniques chaque intervalle de temps attribué à une voie lui permettra de transmettre 1 ou plusieurs bits.

- le *multiplexage statistique* améliore le multiplexage temporel en n'attribuant la voie haute vitesse qu'aux voies basse vitesse qui ont effectivement quelque chose à transmettre. En ne transmettant pas les silences des voies basses cette technique implantée dans des *concentrateurs* améliore

6. Le même équipement joue à la fois le rôle de multiplexeur et démultiplexeur.

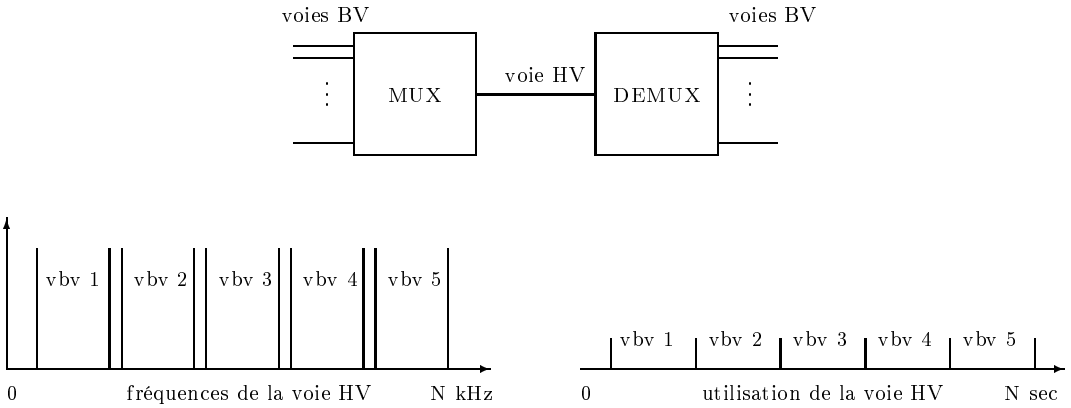

FIG. 1.10 - *Multiplexages d'une ligne.*

grandement le débit global des transmissions mais elle fait appel à des protocoles de plus haut niveau et est basée sur des moyennes statistiques des débits de chaque ligne basse vitesse.

1.3.4 Les supports de transmission.

L'objectif de la couche 1 du modèle OSI est aussi de fixer les caractéristiques des matériels utilisés pour relier physiquement les équipements d'un réseau. Nous décrivons succinctement quelques uns des supports de transmission les plus usités.

- la *paire torsadée* est un câble téléphonique constitué à l'origine de deux fils de cuivre isolés et enroulés l'un sur l'autre (d'où le nom). Actuellement on utilise plutôt des câbles constitués de 2 ou 4 paires torsadées. Elle est très répandue, de connexion facile et d'un faible coût mais elle possède une faible immunité aux bruits. Pour améliorer les performances on utilise la *paire torsadée blindée* plus résistante aux perturbations électromagnétiques et qui autorise un débit pouvant aller jusqu'à 16 Mbits/s. D'une manière générale les performances (et les coûts) de ce support dépendent de la qualité des matériaux employés et des détails de réalisation. On recommande actuellement d'utiliser de la 4 paires non blindées de catégorie 5 avec une impédance de 100 ohm (la catégorie 3 est inférieure en qualité, et l'impédance 120 ohm est une particularité française). Utilisée en ligne de téléphone classique leur débit est au maximum de 56 Kbit/s avec les modems les plus récents, mais les progrès de cette technologie autorisent, sur de courtes distances, des débits de l'ordre de 10 Mbit/s voire 100 Mbit/s. On la rencontre très souvent comme support des réseaux `10 Base T`[7], chaque extrémité d'un tel câble étant muni d'une prise `RJ45`. Son intérêt principal est que cette même paire torsadée peut servir au réseau téléphonique, au réseau informatique et vidéo d'une même entreprise et de plus elle pourra être utilisée ultérieurement pour évoluer vers des réseaux `100 Base T` et même «Gigabits». Dans ce type de réseaux locaux chaque poste est relié à un *hub*, par une liaison point à point, formant physiquement une étoile (dont le centre est un hub) ou un arbre, mais dont le fonctionnement est en mode de diffusion de type bus. Cependant l'orientation actuelle est de remplacer les hubs par des commutateurs qui eux réalisent de la diffusion en mode point à point.

- le *câble coaxial* est un câble utilisé également en téléphonie et en télévision, il est constitué d'un cœur qui est un fil de cuivre. Ce cœur est dans une gaine isolante elle-même entourée par une tresse de cuivre, le tout est recouvert d'une gaine isolante. Certains coaxiaux «large bande» peuvent atteindre un débit maximal de 150 Mhz mais son encombrement est nettement supérieur à celui de la paire torsadée et ses performances n'atteignant pas celle de la fibre optique il a tendance à disparaître des nouveaux plans de câblage.

7. T pour twisted (torsadée), 10 pour 10 Mbits/s et `100 Base T` pour 100 Mbit/s.

On le rencontre dans sa version `10 Base2` (ou Ethernet fin 10 Mbit/s sur 200 m maximum) ou `10 Base5` (ou Ethernet épais 10 Mbit/s sur 500 m maximum) pour la réalisation de réseaux locaux à topologie en bus. Les connexions de chaque poste sur le bus se font à l'aide de connecteur en `T` et la connexion du câble sur le poste se fait à l'aide de connecteur `AUI` pour l'Ethernet épais et `BNC` pour l'Ethernet fin[8]. Il est actuellement beaucoup utilisé pour relier entre eux deux éléments actifs (hub, routeur,...)

– la *fibre optique* est un support d'apparition plus récente mais son utilisation prend de l'ampleur de jour en jour car elle permet(tra) des débits de plusieurs Gbit/s sur de très longues distances. Elle est particulièrement adaptée à l'interconnexion de réseaux par exemple entre plusieurs bâtiment d'un même site. En plus de ses capacités de transmission, ses grands avantages sont son immunité aux interférences électromagnétiques et sa plus grande difficulté d'écoute, contrairement aux supports électriques, ce qui la rend également attrayante dans les contextes où la confidentialité est requise. D'un point de vue technique une fibre optique est constituée d'un cœur et d'une gaine en silice de quelques μm recouvert d'un isolant. À une extrémité une diode électroluminescente (LED) ou une diode laser émet un signal lumineux et à l'autre une photodiode ou un phototransistor est capable de reconnaître ce signal.

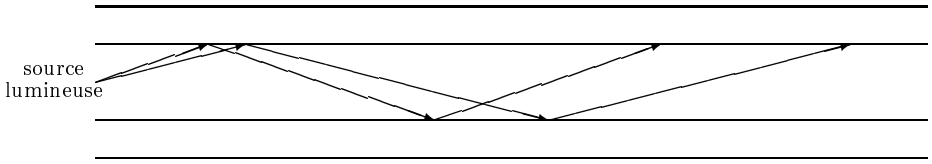

FIG. 1.11 - *Réflexion interne.*

Les différents rayons lumineux issus de la source sont guidés par le fil de verre en suivant un principe de réflexion interne qui se produit au niveau de la frontière entre le cœur et la gaine comme illustré dans la figure 1.11. Si la réflexion ne laisse subsister qu'un seul rayon, car le diamètre du fil est très réduit, alors on parle de fibre *monomode* sinon, lorsqu'il existe plusieurs rayons simultanément on parle de fibre *multimode*. Enfin, la bande passante d'une fibre optique étant très large (plusieurs MHz) il est aisé de faire du multiplexage fréquentiel pour faire transiter simultanément plusieurs communications.

– les *liaisons sans fil* sont possibles grâce à des liaisons infrarouges ou laser sur de courtes distances et grâce aux faisceaux hertziens pour les liaisons satellitaires. Les débits sont très élevés mais les transmissions sont sensibles aux perturbations et les possibilités d'écoute sont nombreuses.

1.3.5 Exemple de l'ADSL.

La technique de l'`ADSL` (Asymetric bit rate Digital Subscriber Line ou ligne numérique d'abonnés à débits asymétriques) est une technique récente qui permet d'utiliser, sur de courtes distances, les lignes téléphoniques classiques mais avec un débit très supérieur à celui des normes plus classiques (V34 ou V90). Par exemple, dans sa version Lite, elle permet de connecter à Internet un particulier en utilisant simplement sa ligne téléphonique habituelle comme illustré dans la figure 1.12.

De manière théorique, cette technologie offre un débit maximal descendant (d'Internet vers l'abonné) de 8,2 M bit/sec et un débit maximal montant (de l'abonné vers Internet) de 640 K bit/sec. Cependant, ces performances ne sont pas possibles sur une grande distance (plus de 5 km) et les solutions commerciales grand public proposées en France actuellement (fin 1999) fixent par exemple le débit entrant à 512 Kbit/sec et le débit sortant à 128 Kbit/sec.

8. Le fonctionnement d'un réseau Ethernet sera approfondi dans la section 2.4.1.

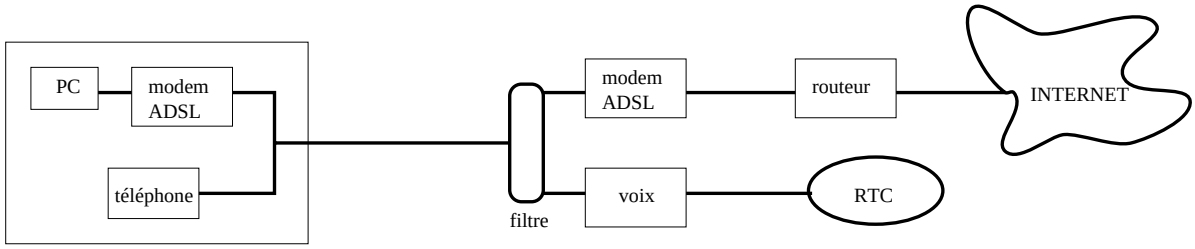

FIG. 1.12 - *Connexion à Internet via ADSL Lite.*

D'un point de vue technique ADSL fonctionne en full duplex grâce à un multiplexage fréquentiel, permettant de faire transiter simultanément les signaux montant et descendant accompagnés également des signaux portant la voix téléphonique. La figure 1.13 illustre ce multiplexage dans le cas où les

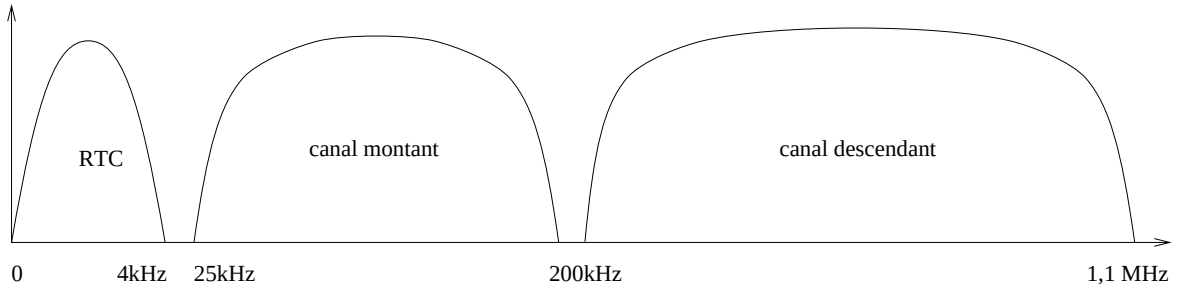

FIG. 1.13 - *Multiplexage fréquentiel utilisé par ADSL (échelle des fréquences non réelles).*

fréquences pour les voies montantes et descendantes ont été clairement séparées. Pour gagner encore en largeur de bande, et donc en débit, on peut envisager de rapprocher les deux espaces de fréquences mais il faut alors annuler les perturbations (phénomène d'écho) que subissent les signaux montant et descendant émis simultanément.

Les différents signaux sont transmis selon la technologie DMT (*Discrete MultiTone*) qui divise la totalité de la bande passante en 256 sous-canaux d'une largeur de 4,3 kHz. Ainsi, le 1er canal est réservé à la téléphonie. Les canaux 2 à 6 servent à séparer la voix des données numériques. Le flux montant occupe les 32 canaux suivants et le flux descendant tous les canaux restant, dans le cas où aucune zone de fréquence ne sépare les deux sens de communication et que l'annulation d'écho est en place. Le fait que la largeur de bande montante soit plus faible que la descendante explique le terme *asymétrique* dans la dénomination ADSL. De plus, certains sous-canaux sont utilisés pour la gestion de la transmission

Chacun des sous-canaux est modulé indépendamment en utilisant la technique du QAM (Quadrature amplitude modulation), qui est une méthode de modulation d'amplitude de deux porteuses en quadrature (4 niveaux d'amplitude). Avant tout transfert de données, une procédure de négociation (handshake) est mise en place pour mesurer la qualité de la transmission et l'adapter en fonction de la ligne. On appelle cette technique *rate adaptative*, car elle est capable de diminuer le débit si la qualité de la transmission se dégrade.

1.4 La couche liaison.

Définition 1.4.1 *La couche liaison de données fournit les moyens fonctionnels et procéduraux nécessaires à l'établissement, au maintien et à la libération des connexions de liaison de données entre entités du réseau. Elle détecte et corrige, si possible, les erreurs dûes au support physique et signale à la*

couche réseau les erreurs irrécupérables. Elle supervise le fonctionnement de la transmission et définit la structure syntaxique des messages, la manière d'enchainer les échanges selon un protocole normalisé ou non.

Une connexion de liaison de données est réalisée à l'aide d'une ou plusieurs liaisons physiques entre deux machines adjacentes dans le réseau donc sans nœuds intermédiaires entre elles.

Nous commençons par examiner les différentes techniques de détection et correction d'erreur (changement de `1` par `0` ou vice-versa), puis nous étudierons deux familles de protocoles de liaison de données.

1.4.1 Détection et correction d'erreurs.

Le taux d'erreurs de transmission est de l'ordre de 10^{-5} sur une ligne téléphonique, de 10^{-7} à 10^{-8} sur un coaxial et de 10^{-10} à 10^{-12} sur une fibre optique. À ce niveau-là il ne s'agit pas d'assurer la correction globale d'un échange, mais de détecter et d'éventuellement corriger des erreurs de transmissions dans un bloc de bits acheminé par le support physique. En effet, puisque la couche 2 ne connaît pas le propriétaire des paquets qu'elle manipule, elle ne peut pas se substituer aux couches de niveau supérieur.

Les techniques employées ici reposent sur l'utilisation de *codes correcteurs* ou *codes détecteurs* d'erreurs qui chacun transforme la suite de bits à envoyer en lui ajoutant de l'information à base de *bits de redondance* ou *bits de contrôle*. Le récepteur se sert de cette information ajoutée pour déterminer si une erreur s'est produite et pour la corriger si la technique employée le permet.

- la *parité* ajoute à chaque bloc de i bits (i=7 ou 8) émis un *bit de parité* de telle sorte que parmi les $i+1$ bits émis le nombre de bits à 1 soit toujours pair (ou impair). Par exemple, pour une parité paire si le bloc initial est de 7 bits et est égal à `1000001` le bloc de 8 bits émis est `10000010`, pour envoyer `0110100` le bloc `01101001` est émis. À la réception, le décodeur calcule le nombre de bits à 1 et dans le cas d'une parité paire si ce nombre de bits est pair on suppose qu'il n'y a pas eu d'erreur. Sinon, on sait alors qu'il y a eu une erreur de transmission mais on ne sait pas la localiser et il faut alors demander la réémission du bloc. La technique de parité est simple à mettre en œuvre cependant elle ne permet pas de détecter $2n$ erreurs dans le même bloc de bits transmis, car dans ce cas la parité ne sera pas changée.

- les *codes à redondance cyclique* (CRC) ajoutent des bits qui sont des combinaisons linéaires des bits de l'information à transmettre. La suite de bits à transmettre $u_1, u_2, \ldots, u_k$ est considérée comme un polynôme $M(x) = u_1 x^{k-1} + u_2 x^{k-2} + \ldots + u_k$. Par exemple `1101011011` est représenté par $x^9 + x^8 + x^6 + x^4 + x^3 + x + 1$. À l'émission, on calcule la division du polynôme M multiplié par x^r par le *polynôme générateur* G de degré r. On appelle Q le polynôme quotient et R le polynôme reste de cette division, on a donc : $x^r M(x) = Q(x).G(x) + R(x)$.

  La suite de bits correspondant au polynôme R constitue le CRC qui est ajouté à l'information à transmettre, le polynôme total émis est donc $E(x) = x^r M(x) + R(x)$ Par exemple, à l'aide du polynôme générateur $G(x) = x^4 + x + 1$, la suite `1101011011` sera transmise accompagnée du CRC `1110` car

$$\begin{aligned} x^4 M(x) &= x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 \\ &= (x^9 + x^8 + x^3 + x)(x^4 + x + 1) + \mathbf{x^3} + \mathbf{x^2} + \mathbf{x} \end{aligned}$$

  À la réception, on divise le polynôme M' correspondant à la suite totale de bits reçus (information+CRC) par le polynôme générateur. Si le reste calculé est non nul, c'est qu'une erreur s'est produite dans la transmission. Si le reste est nul, on est à peu près sûr (99,975% avec le polynôme générateur $x^{16} + x^{12} + x^5 + 1$ de la norme V41 du ITU-T) que la transmission s'est faite sans erreur.

Pourquoi cela fonctionne-t-il ? Il est évident que $x^r M(x) - R(x)$ est divisible par $G(x)$, mais en arithmétique modulo 2 addition et soustraction sont équivalentes (ce sont des OU exclusifs en fait) donc on a également $E(x) = x^r M(x) + R(x) = G(x)Q(x)$ montrant que E est un polynôme multiple de G. Si lors de la transmission des erreurs se sont produites, cela se traduit par le fait que le polynôme reçu $M'(x) = E(x) + T(x)$, T étant le polynôme correspondant aux erreurs ($T(x) = x^i$ si le i$^{\text{e}}$ bit a été inversé). À la réception le décodeur calcule le reste de $\frac{E(x)+T(x)}{G(x)}$ qui est en fait le reste de $\frac{T(x)}{G(x)}$ puisque E est un multiple de G. Si ce résultat est non nul, c'est que T est non nul et que des erreurs se sont produites. Évidemment, le résultat est également nul si T est un multiple de G ce qui masque des erreurs, mais le choix judicieux de G permet de minimiser ces erreurs non détectées. Enfin, il faut aussi remarquer un inconvénient de cette méthode qui signale des erreurs de transmission même si celles-ci ont eu lieu dans le CRC et non dans l'information à transmettre initialement. Dans ce cas il ne devrait pas être nécessaire de retransmettre l'information, or c'est ce qui est fait puisque globalement le transfert (info+CRC) a subi des perturbations.

– le *code de Hamming* est un code correcteur d'erreurs basé sur la notion de *distance de Hamming*. Soit un alphabet composé de 4 caractères (00,01,10,11). Si une erreur se produit alors le caractère émis est transformé en un autre caractère et il n'y a pas moyen de retrouver le caractère original. Par contre, en ajoutant de l'information de telle sorte que les caractères soient très différents les uns des autres cela devient possible. Par exemple, on peut coder les 4 caractères de la manière

caractère initial	00	01	10	11
caractère émis	00000	01111	10110	11001
caractères erronés possibles	00001	01110	10111	11000
	00010	01101	10100	11011
	00100	01011	10010	11101
	01000	00111	11110	10001
	10000	11111	00110	01001

TAB. 1.1 - *Codage de Hamming.*

illustrée dans la table 1.1. Ainsi si un bit (parmi les 5 émis) est erroné on sait quand même déterminer quel caractère a été émis, car comme on peut le voir dans la table 1.1 la modification d'un bit ne peut pas faire passer d'un caractère initial à l'autre. On a des ensembles «d'erreurs possibles» totalement disjoints. Par contre la modification de 2 bits dans cet exemple peut amener à des confusions et à l'impossibilité de corriger les erreurs.

Soit x et y deux caractères d'un alphabet $\mathcal{A}$ et soit N la longueur du codage des mots de cet alphabet, x_i et y_i désignent respectivement le i$^{\text{e}}$ bit de x et y. On peut alors définir la distance $d(x,y) = \sum_{i=1}^{N} (x_i - y_i) \bmod 2$ qui permet de compter le nombre de bits qui diffèrent entre x et y. On définit alors la *distance de Hamming* par $d_H = \inf_{(x,y)\in\mathcal{A}^2, x\neq y} d(x,y)$. Dans l'exemple choisi ci-dessus, $d_H = 1$ dans le premier codage sur 2 bits et $d_H = 3$ dans le codage sur 5 bits. Chaque erreur sur un bit d'un caractère x donne un caractère x' tel que $d(x,x') = 1$, donc pour pouvoir détecter et corriger une seule erreur il faut que $d_h \geq 3$ et pour corriger 2 erreurs il faut que $d_h \geq 5$. D'une manière générale on détecte et corrige n erreurs quand la distance de Hamming est $2n+1$.

On peut voir ceci dans la figure 1.14 où sont représentés x et y (2 caractères parmi les plus proches de l'alphabet), x_i et y_i des «déformations» de x et y après une erreur et x'_1 et y'_1 des «déformations» après deux erreurs. Ainsi, quand on reçoit un caractère x (erroné ou non on ne peut pas le savoir à l'avance) il suffit de chercher le caractère $c \in \mathcal{A}$ le plus proche de x selon la distance d pour obtenir le caractère émis.

Un exemple de *code de Hamming* est donné par la technique suivante où l'on veut envoyer des caractères codés sur 4 bits de données $ABCD$. Pour cela on va émettre la suite $ABCP_2DP_1P_0$

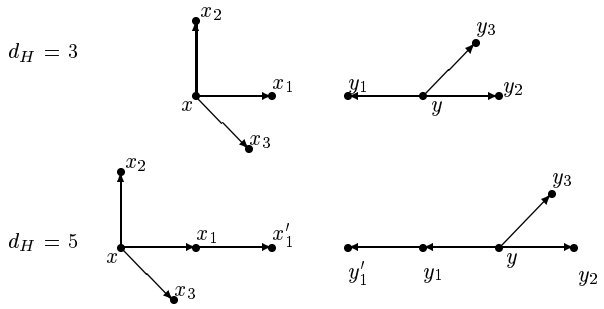

FIG. 1.14 - *Distance de Hamming.*

dans laquelle les bits de contrôle P_i sont placés sur les bits de rang 2^i et sont définis par

$$\begin{cases} P_0 &= A \oplus C \oplus D \\ P_1 &= A \oplus B \oplus D \\ P_2 &= A \oplus B \oplus C \end{cases}$$

Les P_i sont des bits de parité définis à l'aide des bits de données de rang k tels que la décomposition de k en somme de puissances de 2 contienne 2^i. Par exemple, A est un bit de donnée de rang $7 = 2^0 + 2^1 + 2^2$ donc A sert au calcul de P_0, P_1 et P_2. De même, le rang de D est $3 = 2^0 + 2^1$ donc il sert au calcul de P_0 et P_1.

À la réception on calcule

$$\begin{cases} P'_0 &= P_0 \oplus A \oplus C \oplus D \\ P'_1 &= P_1 \oplus A \oplus B \oplus D \\ P'_2 &= P_2 \oplus A \oplus B \oplus C \end{cases}$$

si on obtient $P'_2 = P'_1 = P'_0 = 0$ alors c'est que la transmission s'est passée sans problème. Sinon, la valeur binaire de $P'_2P'_1P'_0$ donne la place de l'erreur dans les bits reçus (en commençant par la droite). On corrige alors l'erreur et on recalcule les P'_i, s'ils sont devenus tous nuls l'erreur a été corrigée, sinon il y avait eu au moins deux erreurs et on peut rejeter la suite de bits mais pas la corriger.

Par exemple, si l'on veut envoyer les 4 bits `0010`, on va finalement émettre `0011001`. Si l'on reçoit `0010001`, on trouve $P'_2P'_1P'_0 = 100$ c'est-à-dire 4, donc l'erreur était en 4^{e} place. On corrige, pour obtenir `0011001` et un nouveau calcul donne $P'_2 = P'_1 = P'_0 = 0$ assurant que l'on a corrigé l'erreur. On peut remarquer qu'en fait on a corrigé une erreur qui n'était pas sur les données initiales mais sur les bits de parité rajoutés.

1.4.2 Protocoles de liaison de données.

Le rôle d'un protocole de liaison de données est évidemment de fixer comment doivent être réalisées les différentes tâches qui incombent à la couche 2 du modèle OSI. Deux grandes familles de telles procédures sont employées. Les procédures *orientées caractères* (BSC de chez IBM) sont assez anciennes et sont utilisées pour des communications à l'alternat sur le principe *send and wait*. Les procédures *orientées bits* (HDLC) sont prévues pour des transmissions full-duplex et à haut débits.

– le *protocole BSC* (Binary Synchronous Communications) est basé sur la transmission de blocs de caractères représentés principalement en ASCII (7 bits) ou EBCDIC (8 bits de chez IBM) avec acquittement à l'alternat. Il utilise à la fois des messages d'information pour transporter les données et des messages de service pour superviser ces échanges.

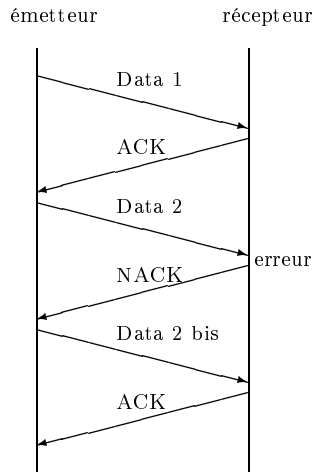

FIG. 1.15 - *Dialogue de type send and wait.*

Les erreurs sont donc détectées et corrigées par demande de répétition comme illustré dans la figure 1.15. La gestion des échanges se fait grâce à l'ensemble de caractères de commandes de la table 1.2.

SYN	synchronous idle	utilisé pour la synchronisation caractère et émis en début de séquence de caractères
ENQ	enquiry	invite une station à émettre ou recevoir
SOH	start of heading	début d'en-tête
STX	start of text	fin d'en-tête et début de texte
ETB	end of transmission block	fin de bloc de données
ETX	end of text	fin du texte et début des caractères de contrôle nécessaires à la détection des erreurs
ACK	acknowledgement	accusé de réception positif
NACK	negative acknowledgement	accusé de réception négatif
DLE	data link escape	caractère d'échappement de transmission
EOT	end of transmission	fin d'un transfert de données

TAB. 1.2 - *Caractères de commande BSC.*

D'autres caractères spécifiques ont été ajoutés par des constructeurs mais ils ne sont pas détaillés ici.

Le rôle du caractère DLE est primordial car il permet d'obtenir une transmission transparente au code. En effet, si les données peuvent contenir des caractères de commande des confusions deviennent possibles. Pour éviter cela, tous les caractères de commande sont précédés de DLE lorsque l'alphabet est tel que les codes de commande sont susceptibles d'apparaître dans les données transmises. De plus, si l'on doit transmettre dans les données le caractère DLE lui-même, il sera alors lui-même précédé de DLE de manière à ce que le caractère qui le suit ne soit pas pris à tort pour une commande.

Les messages (données de taille quelconques) émis selon le protocole BSC sont émis sous forme de blocs de taille appropriée aux possibilités de la ligne. Ainsi un message constitué d'une en-tête

et d'un texte de données, le tout ne constituant qu'un seul bloc sera émis de la manière suivante [9]

`SYN SYN SYN SYN SOH` ...en-tête... `STX` ...texte... `ETX BCC EOT`

`BCC` (Block Check Character) est en fait un ensemble de caractères de contrôle pour détecter les erreurs de transmission. Pour l'EBCDIC et l'ASCII en mode transparent le `BCC` est le polynôme de la norme V41 présenté dans la section 1.4.1

- le *protocole HDLC* (High level Data Link control) est un protocole orienté bit et définit un ensemble de procédures normalisées par l'ISO pour des communications, aussi bien point à point que multipoint, half ou full-duplex, mais toujours entre une machine primaire et une (ou plusieurs) machine(s) secondaires. Les différents modes sont les suivants :
  - le mode *ABM* (Asynchronous Balanced Mode) est un mode de réponse asynchrone équilibré utilisé sur une liaison full-duplex entre 2 machines uniquement (liaison point à point) qui ont chacune le statut de primaire et de secondaire. Le secondaire peut émettre sans avoir reçu de permission du primaire.
  - le mode *NRM* (Normal Response Mode) est utilisé sur une liaison half duplex et ici le secondaire ne peut transmettre que sur invitation du primaire.
  - le mode *ARM* (Asynchronous Response Mode, connu également sous le nom LAP) est utilisé sur une liaison half duplex également, mais le secondaire peut émettre sans que le primaire l'ait sollicité. Ceci peut alors provoquer des problèmes si primaire et secondaire veulent simultanément émettre des données.

  Les trames échangées ont l'allure suivante

  `fanion adresse commande` ...données... `contrôle`

  Le fanion est égal à `01111110` et pour que la transparence au code soit possible, c'est-à-dire pour que la présence d'une suite de 6 bits à `1` dans les données ne soit pas interprétée comme un fanion, l'émetteur insère un `0` après chaque suite de 5 `1`. Le récepteur supprime ce `0` supplémentaire après 5 `1` consécutifs de manière à restaurer le caractère réellement émis. Il existe trois types de trame distingués par les 2 premiers bits du champ de commande. Les trames d'information contiennent des données en provenance, ou à destination, des couches supérieures. Les trames de supervision assurent le contrôle d'erreur et de flux. Les trames non numérotées servent à l'initialisation de la liaison et aux problèmes de reprise sur erreur non réglés à la couche 2. Le contrôle est assuré par la technique du polynôme générateur de la norme V41 (cf section 1.4.1).

1.5 La couche réseau.

Définition 1.5.1 *La couche réseau assure toutes les fonctionnalités de relai et d'amélioration de services entre entité de réseau, à savoir : l'adressage, le routage, le contrôle de flux et la détection et correction d'erreurs non réglées par la couche 2.*

À ce niveau là de l'architecture OSI il s'agit de faire transiter une information complète (un fichier par exemple) d'une machine à une autre à travers un réseau de plusieurs ordinateurs. Il existe deux grandes possibilités pour établir un protocole de niveau réseau : le mode avec connexion et le mode sans connexion déjà présentés dans 1.1. Le premier cas est celui adopté dans la norme X25.3 (composante de la norme X25 du CCITT, également norme ISO 8208 et quasi standard international des années 80, utilisé dans le réseau français TRANSPAC) et décrit partiellement ci-après et le second est celui du protocole IP du réseau Internet décrit dans la section 2.5.

9. On se place dans le cas où le caractère `DLE` n'est pas nécessaire et de plus quelques détails sur la synchronisation sont omis

1.5.1 Le contrôle de flux.

Le *contrôle de flux* consiste à gérer les paquets pour qu'ils transitent le plus rapidement possible entre l'émetteur et le récepteur. Il cherche à éviter les problèmes de congestion du réseau qui surviennent lorsque trop de messages y circulent. On peut citer les quelques méthodes suivantes :

- Dans le *contrôle par crédits*, seuls N paquets sont autorisés à circuler simultanément sur le réseau, donc un paquet ne peut entrer dans le réseau qu'après avoir acquis un *jeton* qu'il relache lorsqu'il arrive à destination. Dans la méthode isarythmique tous les jetons sont banalisés et la difficulté réside dans leur distribution correcte aux bonnes portes du réseau pour assurer un fonctionnement optimal.

- Cette technique est améliorée en fixant des *jetons dédiés* par nœud d'entrée dans le réseau. Chaque nœud gère avec ses jetons une file d'attente des paquets qu'il émet. Quand un paquet arrive à destination, le récepteur renvoie à l'émetteur le jeton correspondant au paquet reçu.

- Dans le cadre d'un circuit virtuel établi pour le mode avec connexion des réseaux X25 on utilise un mécanisme de *fenêtre*. Les paquets de données sont numérotés modulo 8 et contiennent deux compteurs : $P(S)$ un compteur de paquets émis et $P(R)$ un compteur de paquets reçus. L'émetteur n'est autorisé à émettre que les paquets inclus dans la fenêtre, c'est-à-dire les paquets dont le compteur de paquet émis est tel que

$$dernierP(R)reçu \leq P(S)courant \leq dernierP(R)reçu + W$$

où W est la taille de la fenêtre d'émission. De son côté le récepteur renvoie à l'émetteur le compteur de paquets reçus $P(R)$ en l'incrémentant du nombre de paquets reçus correctement et en séquence. Le gestionnaire du réseau peut très bien ne pas renvoyer immédiatement les acquittements s'il désire décharger momentanément le réseau. Par exemple, si le dernier paquet $P(R)$ reçu est égal à 1 et que la fenêtre $W = 4$ cela signifie que le paquet 0 a bien été reçu et que l'émetteur peut envoyer les paquets 1, 2, 3 et 4.

FIG. 1.16 - *Contrôle de flux par fenêtre.*

Si l'émetteur a déjà expédié les paquets 1, 2 et 3 et qu'il reçoit un compteur de paquets reçu $P(R) = 2$ il déplace sa fenêtre d'émission de deux positions comme illustré dans la figure 1.16 et peut expédier les paquets suivants.

1.5.2 Le problème de la congestion.

Malgré tous les efforts pour contrôler le flux d'information dans un réseau celui-ci peut se retrouver face à un problème de congestion. Il s'agit alors de résoudre le problème sans l'aggraver. En effet, les

problèmes de congestion arrivent lorsque les nœuds d'un réseau saturent leurs files d'attente et donc perdent des paquets. Si ces paquets sont réexpédiés ou si des messages de gestion de réseau se mettent à circuler en grand nombre les performances du réseau vont s'écrouler très vite. On essaye d'éviter le problème de la congestion en autorisant un paquet à ne rester dans le réseau qu'un temps limité par un temps maximal fixé par le gestionnaire du réseau. Tout paquet est donc émis avec une date fixée par une horloge commune au réseau, si un nœud s'aperçoit que le temps de présence dans le réseau d'un paquet est dépassé il le détruit. Cela permet ainsi de détruire les paquets perdus par erreur d'adressage ou de routage, ainsi que ceux bloqués dans un nœud. Mais cette méthode basée sur une horloge est assez difficile à mettre en œuvre et on utilise souvent une méthode plus simple consistant à mémoriser simplement dans la zone de temps un nombre décrémenté à chaque traversée de nœud. Lorsque ce nombre atteint la valeur 0 il est détruit.

1.5.3 Le routage.

Le *routage* des paquets dans un réseau maillé consiste à fixer par quelle ligne de sortie chaque commutateur réexpédie les paquets qu'il reçoit. Ceci se fait en fonction de la destination finale du paquet et selon une table de routage qui indique pour chaque destination finale quelles sont les voies de sortie possible.

Pour l'exemple de la figure 1.17 on pourrait avoir la table de routage suivante :

destination finale	voie de sortie
D1	A1, A2
D2	A2
D3	A2, A3
D4	A3

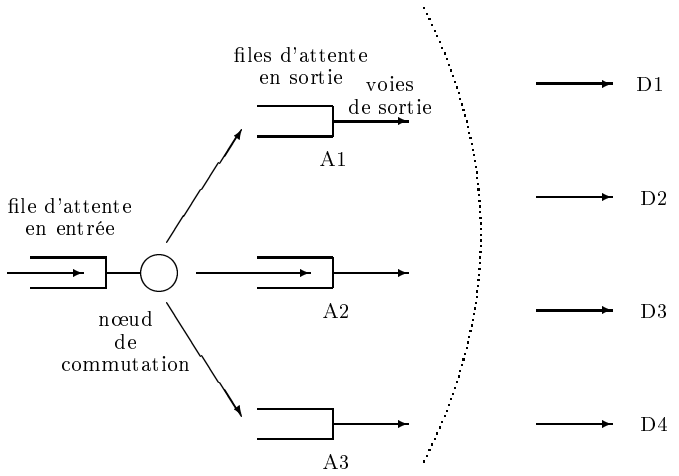

FIG. 1.17 - *Routage de paquets.*

D'une manière générale le routage est un ensemble de processus algorithmiques devant prendre des décisions dispersés dans le temps et dans l'espace. Les différents algorithmes sont répartis sur chaque nœud du réseau et l'ensemble peut fonctionner de manière centralisée ou répartie.

- Le *routage centralisé* est géré par un nœud particulier du réseau qui reçoit des informations de chacun des nœuds du réseau et leur envoie leur table de routage. Pour fixer ces tables on prend en compte notamment le coût des liaisons, le coût de passage dans un nœud, le débit demandé, le nombre de nœuds à traverser, la sécurité de transport de certains paquets, l'occupation des mémoires des nœuds de commutation,... Le plus souvent un algorithme de plus court chemin donne de bons résultats en fixant à 1 le coût de franchissement d'un nœud (on peut également

pondérer plus fortement les nœuds qui sont les plus occupés). La mise à jour des tables de routage peut se faire de manière

  - *fixe* : en fait il n'y a pas de mise à jour, la table de routage est fixée une fois pour toute en fonction de la topologie du réseau.
  - *synchrone* : toutes les tables sont mises à jour au même moment par le centre de contrôle qui reçoit des informations de la part de tous les nœuds à intervalles réguliers (toutes les 10 sec par exemple).
  - *asynchrone* : les tables sont mises à jour indépendamment les unes des autres dans certaines parties du réseau, chaque nœud envoyant un compte-rendu de son état au centre de contrôle lorsqu'il observe des changements significatifs.

  Mais le routage centralisé dans un réseau à grande échelle est peu performant, car un routage est d'autant meilleur qu'il réagit rapidement aux informations qui lui parviennent. De plus, si une panne survient dans l'ordinateur qui assure ce contrôle, c'est tout le réseau qui tombe en panne.

- Le *routage décentralisé* ne possède pas de centre de contrôle et les règles de passage d'un paquet (paquet d'appel pour établissement d'un circuit virtuel) sont :

  - l'*inondation* : à la réception d'un paquet celui-ci est renvoyé sur toutes les lignes de sortie. Cette technique simpliste et rapide est efficace dans les réseaux à trafic faible et où le temps réél est nécessaire mais elle est pénalisante en flux de données, inadaptée aux réseaux complexe et au circuit virtuel.
  - la technique «*hot patatoes*» : un paquet reçu est renvoyé le plus tôt possible par la première ligne de sortie vide. On améliore ce principe en affectant des coefficients à chaque ligne de sortie en fonction de la destination voulue. On tient compte de l'état des nœuds voisins, sans utiliser de paquet de contrôle, mais simplement en comptabilisant le nombre de paquets reçus de chacun d'eux. Ils peuvent également envoyer de manière synchrone ou asynchrone un compte-rendu de leur état, permettant ainsi de choisir la «meilleure» ligne à un instant donné. Mais ceci reste local et une panne du réseau localisée au-delà du premier nœud voisin ne pourra pas être prise en compte.
  - le *routage adaptatif à la fois dans l'espace et dans le temps* demande, de la part de chaque nœud, une connaissance complète du réseau. Les différents nœuds s'échangent donc des messages, mais si chacun envoie des messages à tous les autres le trafic va augmenter de manière baucoup trop grande. C'est pourquoi un nœud ne transmet un compte-rendu qu'à ses voisins immédiats qui doivent en tenir compte dans leur propre compte-rendu. (voir exemple)

D'une manière générale, le routage doit éviter l'usage d'algorithmes adaptatifs trop complexes et limiter les dialogues de services entre les nœuds sinon l'effet obtenu sera à l'opposé de celui recherché.

1.5.4 La norme X25, niveau réseau.

Cette norme a été établie en 1976 par le CCITT pour les réseaux à commutation de paquets sur proposition de 4 pays qui l'utilisent pour leurs réseaux publics de communication : Transpac pour la France, EPSS pour la Grande-Bretagne, Datapac pour le Canada et Telenet pour les USA.

Comme illustré dans la figure 1.18 le protocole X25 contient les trois premières couches du modèle OSI et définit l'interface entre un ETTD (Équipement Terminal de Traitement de Données) et un ETCD (Équipement de Terminaison de Circuit de Données) pour la transmission de paquets. Elle fixe donc les règles de fonctionnement entre un équipement informatique connecté à un réseau et le réseau lui-même.

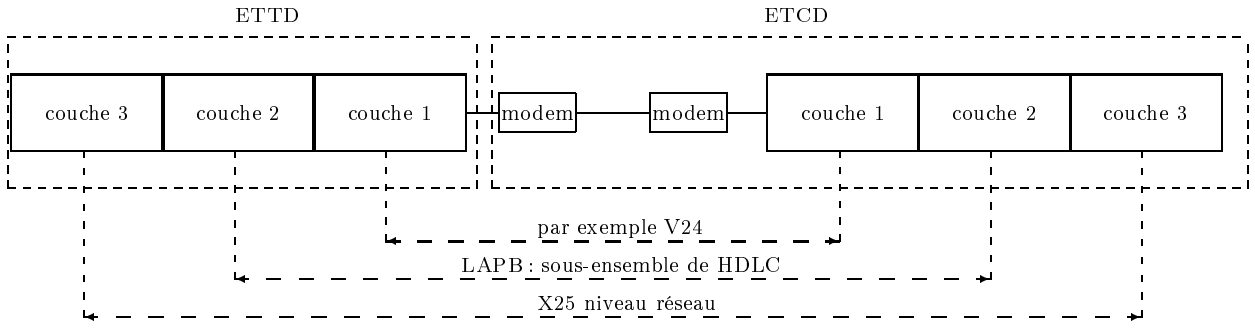

FIG. 1.18 - *Niveaux du protocole X25.*

X25 définit les types de paquets et leur format est le suivant (chaque ligne correspond à un octet[10])

8	7	6	5	4	3	2	1
identificateur général de format				numéro de groupe de voie logique			
numéro de voie logique							
identificateur général de type de paquet							
données							

X25 utilise le mode avec connexion, sachant que la connexion est réalisée à l'aide d'un circuit virtuel dont le fonctionnement se déroule comme illustré dans la figure 1.19.

L'établissement du circuit virtuel se fait de bout en bout par l'envoi d'un paquet d'appel (`call request`). Il existe des circuits virtuels permanents (CVP) ou commutés (CVC). À un instant donné, plusieurs CVC et CVP peuvent coexister. Chaque circuit virtuel utilise une voie logique repérée par un numéro de groupe logique (entre 0 et 16) et un numéro de voie (entre 0 et 255). Ainsi, 4095 (la voie 0 est réservée) voies sont utilisables par une entrée. Pour un CVC les deux numéros sont attribués pendant la phase d'établissement de la communication, pour un CVP ils le sont lors de l'abonnement. Lorsque le paquet d'appel arrive à l'ETCD destinataire, celui-ci peut refuser la connexion en envoyant une demande de libération (`clear request`) ou l'accepter en envoyant un paquet de communication acceptée (`call accepted`). À ce moment-là le circuit virtuel est établi et les données peuvent être échangées, celles-ci emprunteront alors toutes le même chemin marqué à travers le réseau par le paquet d'appel. Pour leur part la demande et la confirmation de libération sont traitées localement et la demande peut être traitée par n'importe lequel des deux équipements.

1.6 La couche transport.

Définition 1.6.1 *La couche transport assure un transfert de données transparents entre entités de session et en les déchargeant des détails d'exécution. Elle a pour rôle d'optimiser l'utilisation des services de réseau disponibles afin d'assurer au moindre coût les performances requises par la couche session.*

C'est la première couche à résider sur les systèmes d'extrêmité. Elle permet aux deux applications de chaque extrémité de dialoguer directement indépendamment de la nature des sous-réseaux traversés et comme si le réseau n'existait pas. Au niveau inférieur de la couche réseau seule la phase d'établissement de la liaison logique s'effectue de bout en bout, alors que les transferts d'information se font de proche en proche.

10. Le troisième octet contient les compteurs P(S) et P(R) décrits dans la section 1.5.1.

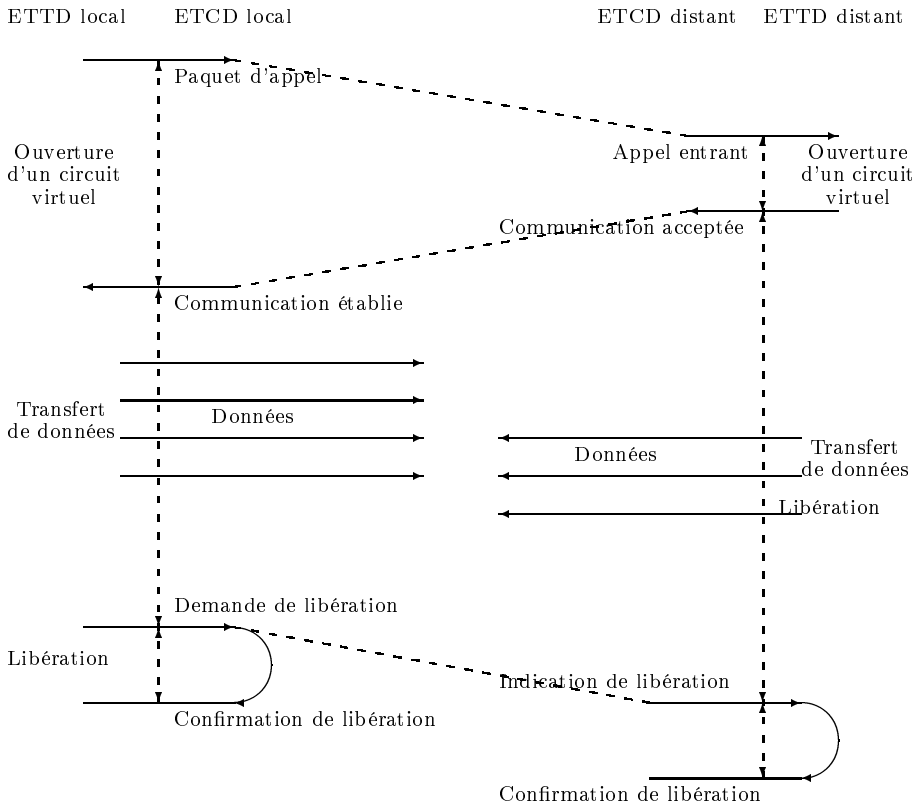

FIG. 1.19 - *Vie d'un circuit virtuel.*

La couche transport doit assurer, en mode connecté ou non connecté, un transfert transparent de données entre utilisateurs de service réseau en leur rendant invisible la façon dont les ressources de communication sont mises en œuvre. Cette notion de transparence implique par exemple de pouvoir acheminer de bout en bout des TPDU (Transport Protocol Data Unit) dont la taille peut varier de 128 à 8192 octets. Cette taille est cependant fixe une fois que la valeur a été négociée.

1.6.1 Qualité de service.

Une autre façon de définir la couche transport est de la considérer garante de la *qualité de service* (QOS) rendue par la couche réseau. Si cette dernière est sans faille, alors le travail de la couche transport est minime. Dans le cas contraire elle assure la jonction entre ce que désire l'utilisateur et ce que la couche réseau met à disposition en terme de qualité. La qualité est évaluée sur certains paramètres avec trois types de valeurs possibles : *préféré*, *acceptable* et *inacceptable* qui sont choisis lors de l'établissement d'une connexion (certains paramètres sont cependant disponibles pour un mode sans connexion). La couche transport surveille alors ces paramètres pour déterminer si la couche réseau sous-jacente assure la qualité de service demandée.

Les différents paramètres de la qualité de service sont :

- le *temps d'établissement de la connexion transport*: c'est la durée qui s'écoule entre le moment où une demande de connexion est émise et le moment où la confirmation de cet établissement est reçu et plus ce délai est court meilleure est la qualité de service.

- la *probabilité d'échec d'établissement* mesure la chance (ou plutôt malchance) qu'une connexion ne puisse s'établir dans un délai maximum défini. On ne tient pas compte ici du refus de l'entité

distante d'établir cette connexion, mais on considère plutôt les problèmes d'engorgement de réseau.

- le *débit de la liaison* mesure le nombre d'octets utiles qui peuvent être transférés en une seconde, ce débit est évalué séparément dans les deux sens.
- le *temps de transit* mesure le temps écoulé entre le moment où l'utilisateur du service de transport envoie un message et celui où l'entité de transport réceptrice le reçoit, ce temps est évalué séparément dans les deux sens.
- le *taux d'erreur résiduel* est le rapport entre le nombre de messages perdus ou mal transmis et le nombre total de messages émis au cours d'une période considérée. Ce nombre, en théorie nul, a une valeur faible.
- la *probabilité d'incident de transfert* mesure le bon fonctionnement du service transport. Lorsqu'une connexion est établie, un débit, un temps de transit, un taux résiduel d'erreurs sont négociés. La probabilité d'incident mesure la fraction de temps durant laquelle les valeurs fixées précédemment n'ont pas été respectées.
- le *temps de déconnexion* mesure la durée s'écoulant entre une demande de déconnexion émise et la déconnexion effective du système distant.
- la *probabilité d'erreur de déconnexion* est le taux de demandes de déconnexion non exécutées pendant le temps maximum défini.
- la *protection* est définie comme la possibilité de se prémunir contre les intrusions passives (interférences sur une même ligne) et actives (écoute et modification des données transmises).
- La *priorité* permet à l'utilisateur de privilégier certaines transmissions par rapport à d'autres.
- La *résiliation* est la liberté laissée à la couche transport de décider elle-même de la déconnexion suite à un problème.

En mode non connecté, ces paramètres indiquent uniquement les souhaits de l'utilisateur en ce qui concerne le débit, le délai de transit, le taux d'erreur résiduel et la priorité d'une transmission. Ces paramètres sont utilisées apr la couche transport pour fixer des options de protocoles avant d'être soumis à la couche réseau.

Lorsqu'une connexion est demandée, tous ces paramètres sont transmis par l'utilisateur à la couche transport, les valeurs désirées et minimales sont spécifiées. Les différents cas de figure et étapes peuvent alors se présenter.

- Si la demande semble irréaliste pour certains paramètres alors la demande de connexion n'est même pas exécutée et un message d'erreur est renvoyé pour expliquer le problème.
- Si la demande ne peut pas être satisfaite complètement mais partiellement (par exemple avec un débit moindre mais acceptable), alors c'est cette demande avec des objectifs moindres qui est soumise.
- Si l'ordinateur distant ne peut satisfaire complètement la demande, mais reste au-dessus du minimum requis, alors il modifie aussi le paramètre.
- S'il ne peut pas rester au-dessus de ce minimum, il rejette la demande de connexion.
- Enfin, la couche transport avertit son utilisateur de la bonne fin (ou non) de la procédure de connexion et lui transmet les paramètres acceptés.

Cette procédure s'appelle la *négociation des options* qui, une fois fixées, restent inchangées pendant toute la connexion. Pour que tous les utilisateurs ne demandent pas une qualité de service optimale, les prix pratiqués par les fournisseurs de réseaux croissent avec cette qualité de service.

1.6.2 Primitives du service transport.

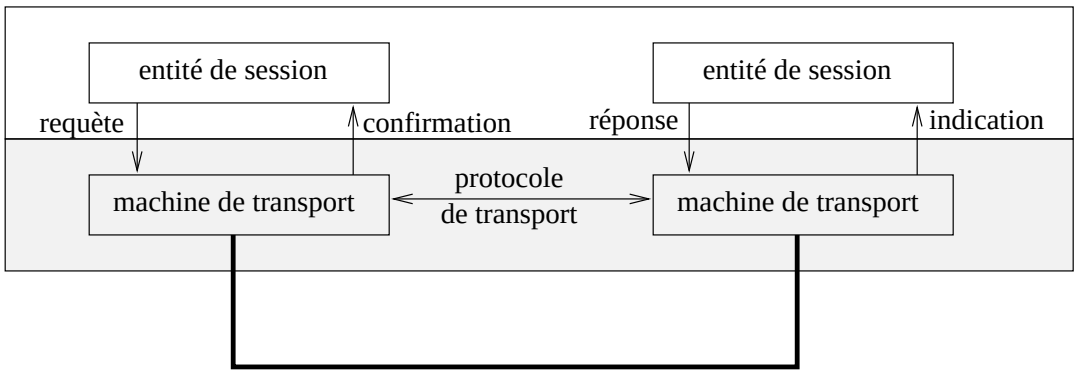

FIG. 1.20 - *Dialogue de niveau transport.*

Les services qu'offrent la couche transport en mode connecté sont rendues par les primitives données ci-dessous. Celles-ci se décomposent comme dans tout dialogue entre couches en quatre catégories comme illustré dans la figure 1.20.

- phase d'établissement de la connexion
  - `T_CONNECT.request(adresse source, adresse distante, données_exprès, qos, données_utilisateur)` pour demander une connexion
  - `T_CONNECT.indication(adresse source, adresse distante, données_exprès, qos, données_utilisateur)` pour indiquer une connexion de transport
  - `T_CONNECT.response(adresse source, adresse distante, données_exprès, données_utilisateur)` pour répondre à une demande de connexion de transport
  - `T_CONNECT.confirm(adresse source, adresse distante, données_exprès, qos, données_utilisateur)` pour confirmer l'établissement d'une connexion de transport
- phase de transfert de données
  - `T_DATA.request(données_utilisateur)` pour demander le transfert de données
  - `T_DATA.indication(données_utilisateur)` pour indiquer un transfert de données
  - `T_EXPEDITED_DATA.request(données_utilisateur)` pour demander le transfert de données exprès
  - `T_EXPEDITED_DATA.indication(données_utilisateur)` pour indiquer un transfert de données exprès
- phase de libération de la connexion
  - `T_DISCONNECT.request(données utilisateur)` pour demander une déconnexion de transport
  - `T_DISCONNECT.indication(raison, données_utilisateur)` pour indiquer une déconnexion de transport

La figure 1.21 détaille les différents enchaînements de primitives possibles. Pour chaque cas, un utilisateur est placé à gauche des lignes doubles et son homologue est à droite, la couche transport est entre les lignes et le temps s'écoule de haut en bas.

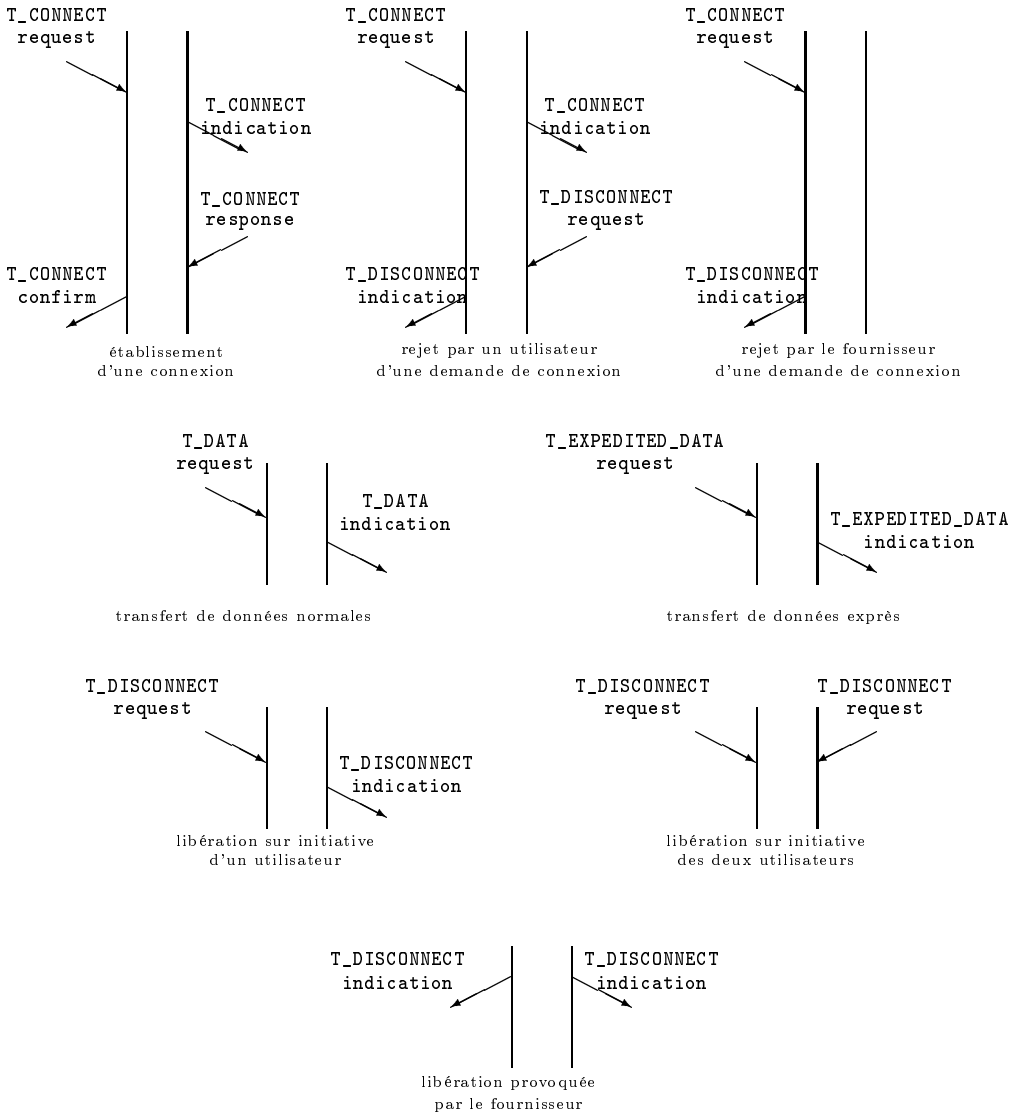

FIG. 1.21 - *Enchaînements de primitives en mode connecté.*

Seules les successions de primitives décrites dans l'automate de la figure 1.22 sont autorisés assurant ainsi qu'une machine réalisant le service de transport ne peut se trouver que dans l'un des quatre états représentés à savoir

- *veille* : aucune connexion n'est établie, une demande de connexion peut être émise ou reçue
- *connexion sortante en attente* : la machine a demandé une connexion et la réponse de l'autre extrêmité n'est pas encore arrivée
- *connexion entrante en attente* : la machine a reçu une demande de connexion qu'elle n'a pas encore acceptée ou rejetée.
- *transfert de données prêt* : une connexion a été établie, les transferts de données peuvent commencer

Pour ce qui est du mode non connecté seules les primitives suivantes sont disponibles.

- `T_UNIDATA.request(appelé, appelant, qos, données utilisateur)`

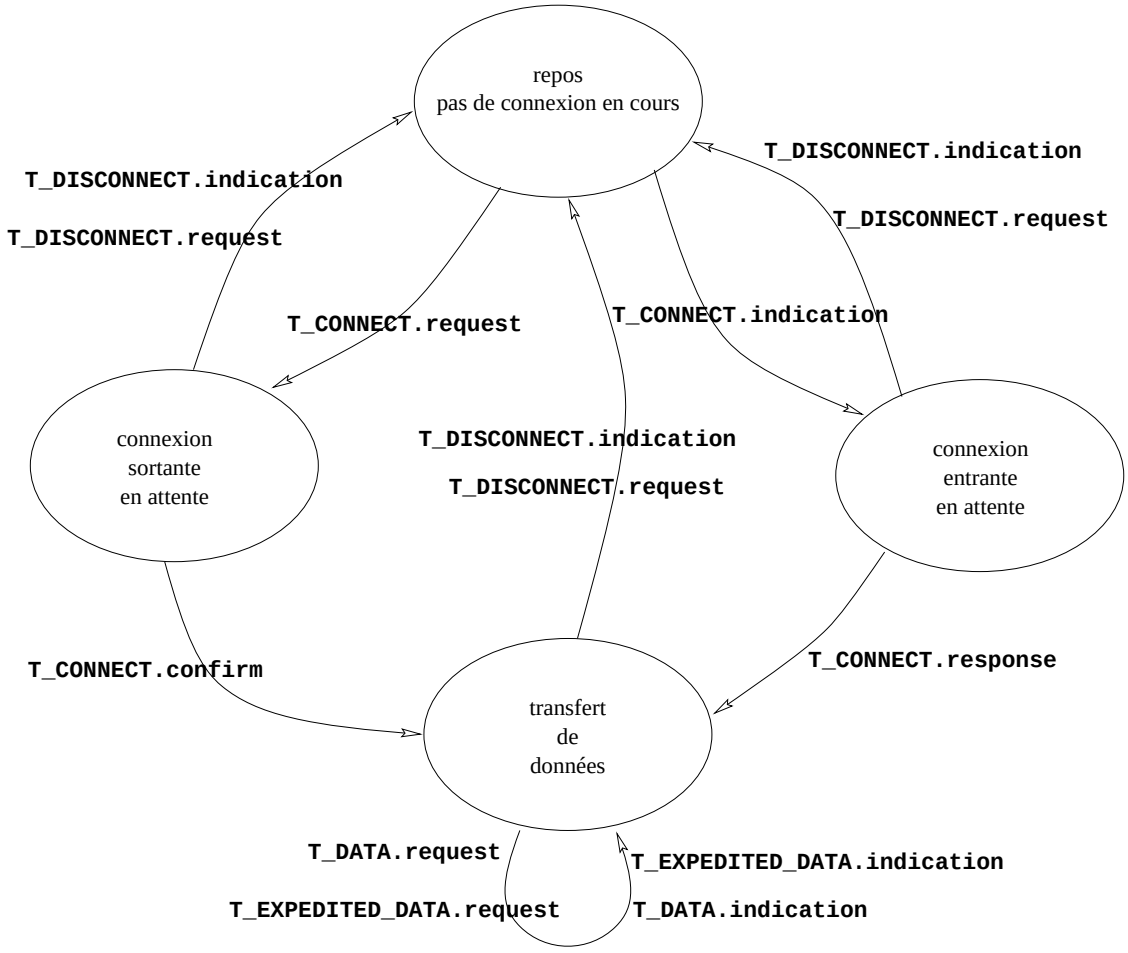

FIG. 1.22 - *Diagramme d'états d'une machine de service de transport.*

– `T_UNIDATA.indication(appelé, appelant, qos, données utilisateur)`

1.6.3 Le protocole de transport ISO en mode connecté (ISO 8073 ou X.224)

Les fonctions de la couche transport ISO sont établies dans le but de combler l'écart existant entre les services offerts par les couches basses et les services à offrir.

Les principales fonctions réalisées pendant l'établissement de la connexion sont les suivantes.

– La sélection d'une connexion réseau appropriée.

– La décision de mettre en place un multiplexage ou un éclatement. Le multiplexage a pour but de regrouper plusieurs connexions de transport sur une même connexion de réseau, ce qui va permettre de minimiser les ressources utilisées au niveau réseau, donc de diminuer les coûts de communication. Ceci est surtout utile si l'on a de nombreuses connexions de faible débit à établir. À l'inverse, l'éclatement consiste à répartir une connexion de transport simultanément sur plusieurs connexions réseau dans le but d'accroître le débit et la fiabilité (voir la figure 1.23).

– la détermination de la taille optimale des TPDU

– la mise en relation des adresses transport et réseau

– l'identification de la connexion

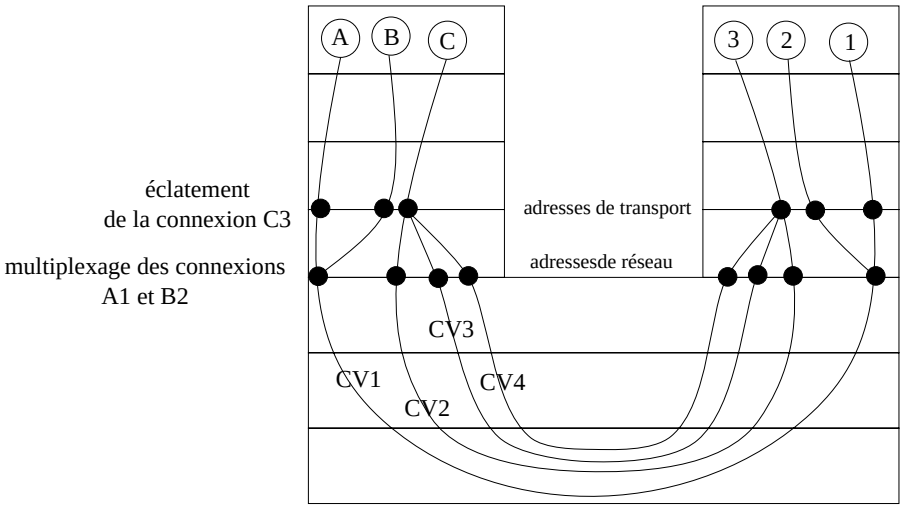

FIG. 1.23 - *Adressage, multiplexage et éclatement de connexions transport.*

Lors de la phase de transfert les données exprès (interruptions, alarmes de petite taille) ne sont pas soumises au contrôle de flux et sont assurées d'être délivrées avant toutes données normales émises après elle. La *fragmentation-réassemblage* permet, si nécessaire, de découper des TSDU (Transport Service Data Unit) afin de tenir dans un seul TPDU. Ainsi les «lettres» trop grandes sont découpées en fragments de taille fixe (sauf éventuellement le dernier), numérotés et expédiés chacun dans un paquet pour être réassemblés à l'arrivée. La numérotation des paquets permet également de détecter les paquets perdus ou dupliqués. Les erreurs de transmission étant normalement détectées par les couches inférieures, il n'est pas toujours nécessaire de protéger chaque TPDU. Cependant, si la qualité de service réseau est médiocre, on ajoute des mécanismes de détection d'erreur par total de contrôle associé à un mécanisme de retransmission. Le contrôle de flux est fondé sur une fenêtre coulissante de taille variable, similaire à une notion de crédit. Le destinataire impose la cadence de transfert en indiquant à l'expéditeur des crédits d'émission (numéro dans la séquence à ne pas dépasser) qu'il lui transmet dans ses acquittements.

Enfin, la libération de la connexion de transport peut être décidée et déclenchée de manière inconditionnelle par l'un quelconque des correspondants, ce qui peut provoquer des pertes de données. Cette libération est explicite quand il y a échanges d'unités de données spéciales ou implicite quand en fait c'est la connexion réseau sous-jacente qui est libérée.

Pour simplifier le choix des fonctions à mettre en œuvre, l'ISO a défini 5 classes de protocoles chacune étant adaptée à un type de réseau particulier [11] et choisi lors de l'établissement de la connexion. L'appelant déclare sa classe préférée et des classes de repli, l'appelé choisi parmi ces propositions ; si aucune classe ne lui convient il refuse la connexion. Succinctement, on trouve

- la *classe 0* : elle assure une connexion standard pour les réseaux de *type A* (établissement négociée de la connexion, libération implicite, transfert de données normales, segmentation et réassemblage des données.

- la *classe 1* : pour les réseaux de *type B* elle ajoute à la classe 0 le transport de données exprès, la reprise sur erreurs signalées, la déconnexion et la reprise sur réinitialisation du réseau.

11. Les réseaux sont réparties en 3 classes :

- type A : peu d'erreurs signalées et non signalées
- type B : peu d'erreurs non signalées, mais trop d'erreurs signalées
- type C : trop d'erreurs des deux types.

- la *classe 2* : pour les réseaux de *type A* elle ajoute à la classe 0 le multiplexage et un contrôle de flux amélioré.
- la *classe 3* : pour les réseaux de *type B* elle est constituée des fonctionnalités des classes 1 et 2.
- la *classe 4* : pour les réseaux de *type C* elle correspond à la classe 3 à laquelle on ajoute la détection et la reprise d'erreurs non signalées et l'éclatement de connexions de transport.

1.7 Les couches hautes : session, présentation et application.

Les couches session, présentation et application constituent les couches hautes du modèle OSI et offrent des services orientés vers les utilisateurs alors que les couches basses sont concernées pas la communication fiable de bout en bout. Elles considèrent que la couche transport fournit un canal fiable de communication et ajoutent des caractéristiques supplémentaires pour les applications.

1.7.1 La couche session.

Définition 1.7.1 *La couche session fournit aux entités de la couche présentation les moyens d'organiser et synchroniser les dialogues et les échanges de données.*

Une session peut par exemple être utilisée pour la connexion à distance d'un terminal à un ordinateur ou pour le transfert d'un fichier et ceci en mode connecté. Bien que très similaires la session et la

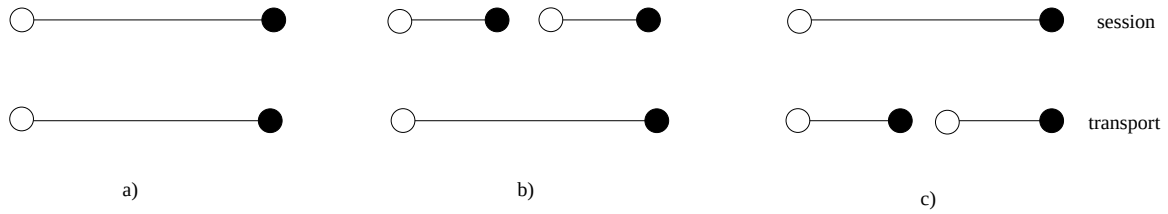

FIG. 1.24 - *Sessions et connexions de transport.*

connexion de transport ne sont pas identiques, les trois cas de la figure 1.24 peuvent se présenter.

a) Il y a correspondance exacte entre une session et une connexion de transport.

b) Plusieurs sessions successives sont établies sur une seule et même connexion de transport. Par exemple, ceci peut être utilisé dans le contexte d'une agence de voyage dans laquelle chaque employé utilise un terminal relié à un ordinateur local. Celui-ci est relié à une base de données centrale de la compagnie pour enregistrer les réservations. Chaque fois qu'un employé veut faire une réservation, une session est ouverte avec l'ordinateur central et elle est fermée lorsque la réservation est terminée. Mais il est inutile de libérer la connexion de transport sous-jacente car celle-ci sera utilisée quelques instants plus tard par un autre employé.

c) Plusieurs connexions de transport successives sont nécessaires pour une seule et même session. Ceci peut arriver lorsqu'une connexion de transport tombe en panne, la couche session établit alors une nouvelle connexion de transport de manière à poursuivre la connexion commencée.

Par contre, il n'est pas possible de multiplexer plusieurs sessions sur une même connexion de transport. Celle-ci ne transporte à tout instant qu'au plus une session.

Le transfert des données est régi par les trois phases habituelles : établissement de la session, transfert des données et libération de la session. Les primitives fournies par la couche session sont semblables à celles fournies par la couche transport à la couche session. L'ouverture d'une session

nécessite la négociation de plusieurs paramètres entre les utilisateurs des extrémités, certains (qos et existence ou non de données exprès) sont identiques à ceux de la couche transport à qui ils sont passés tels quels. Un autre paramètre peut permettre de décider quelle extrémité aura l'initiative du dialogue dans le cas d'une session bidirectionnelle. Les différences entre transport et session vont se situer au niveau de la libération de la connexion. Dans le cas du transport, la primitive `T_DISCONNECT.request` provoque une libération brutale de la connexion avec perte des données en cours de transfert. Une session, elle, sera terminée par la primitive `S_RELEASE.request` qui opère une libération ordonnée de la connexion, sans perte de données, appelée terminaison négociée. Les différences entre ces deux modes

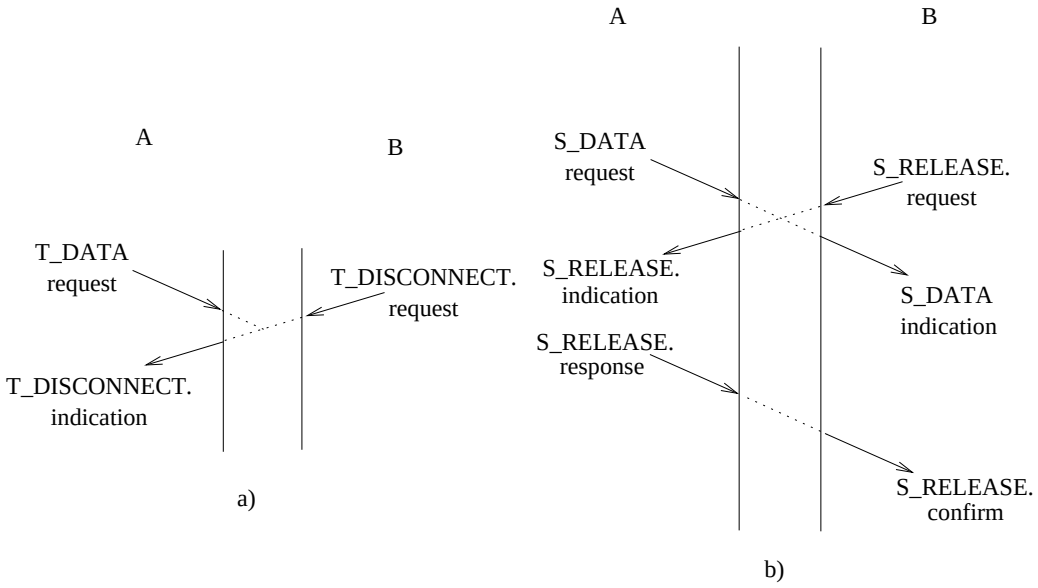

FIG. 1.25 - *Libération brutale a) et ordonnée b).*

de libération sont illustrées dans la figure 1.25. Dans le premier cas, B ne peut plus recevoir le message qui est en transit à partir du moment où il a émis une demande de déconnexion (voir l'automate de la figure 1.22). Dans le deuxième cas, B continue d'accepter des données après avoir émis sa demande de libération jusqu'à ce que A confirme positivement cette déconnexion. Si A veut refuser la déconnexion, il le signale par l'un des paramètres de `S_RELEASE.response` et il peut ainsi continuer d'envoyer des données, la session se prolonge comme si de rien n'était.

Les sessions peuvent autoriser le dialogue bidirectionnel ou unidirectionnel ou à l'alternat. Ce dernier cas n'est pas forcément dû à des limitations matérielles mais est souvent inhérent au fonctionnement des applications de plus haut niveau. Par exemple, l'accès à une base de données distante est établi sur ce mode. Un utilisateur envoie depuis son terminal une requête à un ordinateur central, puis il attend la réponse de celui-ci. Lorsque l'ordinateur central a envoyé sa réponse, l'utilisateur peut envoyer une autre requête. Ainsi, le mode de fonctionnement à l'alternat est naturel. Dans ce cas il s'agit alors de gérer à qui c'est le tour d'envoyer des données. Ce service est mis en œuvre grâce à un *jeton de données* (data token). Seule l'extrémité qui possède le jeton peut envoyer des données, l'autre doit rester silencieuse. La négociation initiale à l'établissement de la connexion de session fixe quelle extrémité reçoit en premier le jeton. Lorsque le possesseur du jeton a fini d'expédier ses données il peut passer le jeton à son correspondant à l'aide de la primitive `S_TOKEN_GIVE.request`. Si l'extrémité qui ne possède pas le jeton veut expédier des données il peut le demander, *poliment*, à l'aide de la primitive `S_TOKEN_PLEASE.request`. Le possesseur du jeton peut alors le lui envoyer ou lui refuser.

La synchronisation est également l'un des services de la couche session et consiste à placer les entités de session dans un état connu des deux interlocuteurs en cas d'erreur. Il ne s'agit pas ici de régler des erreurs qui seraient dûes à un problème de transmission de données puisque la couche transport doit les régler. Mais celle-ci ne peut effectuer des reprises sur erreurs dues aux couches supérieures puisqu'elle

n'en a pas connaissance. Par exemple, si lors d'un transfert de fichiers très long une erreur se produit du côté du récepteur mais en dehors de la transmission proprement dite (problème d'accès à un disque par exemple), il faut que la couche session soit capable de s'en apercevoir et qu'elle évite de relancer simplement tout le transfert de fichiers. C'est pourquoi, grâce à des points de synchronisation dans le flot d'échanges de données, elle reprendra les transferts au niveau du dernier point de reprise correct. Ces points consistent à découper le flot de données à transmettre en pages et évidemment, le mécanisme nécessite que l'utilisateur de la session émetteur soit capable de mémoriser suffisamment longtemps les pages envoyées pour pouvoir en réexpédier certaines si nécessaires.

1.7.2 La couche présentation.

Définition 1.7.2 *La couche présentation s'occupe de la syntaxe et de la sémantique des informations transportées en se chargeant notamment de la représentation des données.*

Par exemple, sur un ordinateur à base d'un processeur de la famille des 68 000 les entiers sont représentés avec les bits de poids fort à gauche et ceux de poids faible à droite. Or, c'est l'inverse sur un ordinateur basé sur un processeur de la famille du 80x86. Cette difficulté sera prise en compte par la couche présentation qui effectuera les opérations nécessaires à la communication correcte entre ces deux familles de machines. Pour ce faire l'ISO a défini une norme appelée *syntaxe abstraite numéro 1* (Abstract Syntax Notation 1) permettant de définir une sorte de langage commun (une syntaxe de transfert) dans lequel toutes les applications représentent leurs données avant de les transmettre.

C'est aussi à ce niveau de la couche présentation que peuvent être implantées des techniques de compression (code de Huffman par exemple) et de chiffrement de données (RSA, DSE, etc...) non détaillées ici (voir cours correspondants de licence).

1.7.3 La couche application.

Définition 1.7.3 *La couche application donne au processus d'application le moyen d'accéder à l'environnement OSI et fournit tous les services directement utilisables par l'application, à savoir:*

- *le transfert d'informations*
- *l'allocation de ressources*
- *l'intégrité et la cohérence des données accédées*
- *la synchronisation des applications coopérantes*

En fait, la couche application gère les programmes de l'utilisateur et définit des standards pour que les différents logiciels commercialisés adoptent les mêmes principes, comme par exemple :

- Notion de fichier virtuel représenté sous forme d'arbre pour les applications de transfert de fichiers, opérations permises sur un fichier, accès concurrentiels, ...
- Découpage des fonctions d'une application de courrier électronique qui se compose d'un le contenu (en-tête et corps) et d'une enveloppe. Une fonctionnalité de l'application gère le contenu et une autre le transfert en interprétant l'enveloppe.

Chapitre 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. 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. 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

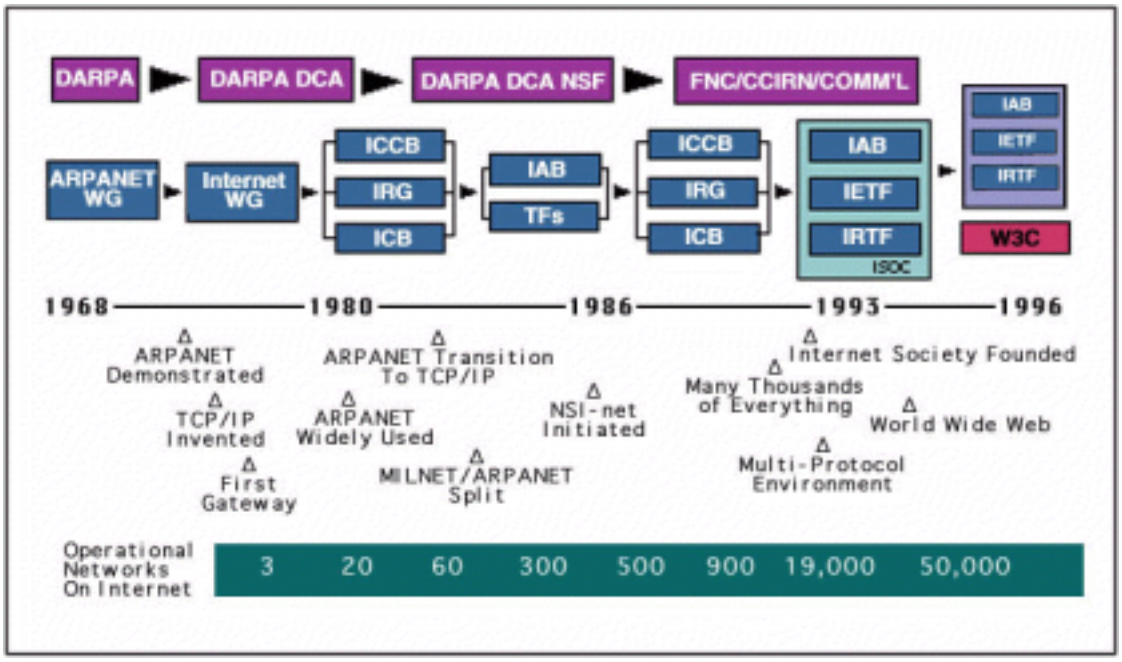

FIG. 2.1 - *Les grandes dates d'Internet.*

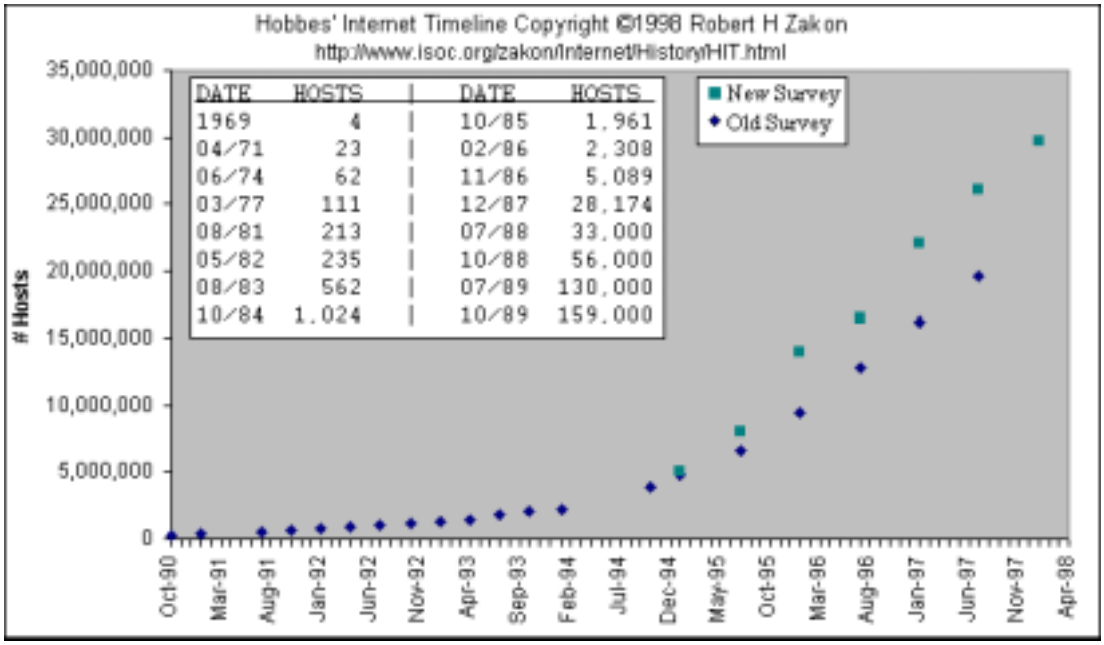

FIG. 2.2 - *Évolution du nombre de machines connectées à Internet*

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.

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...

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. Ici les

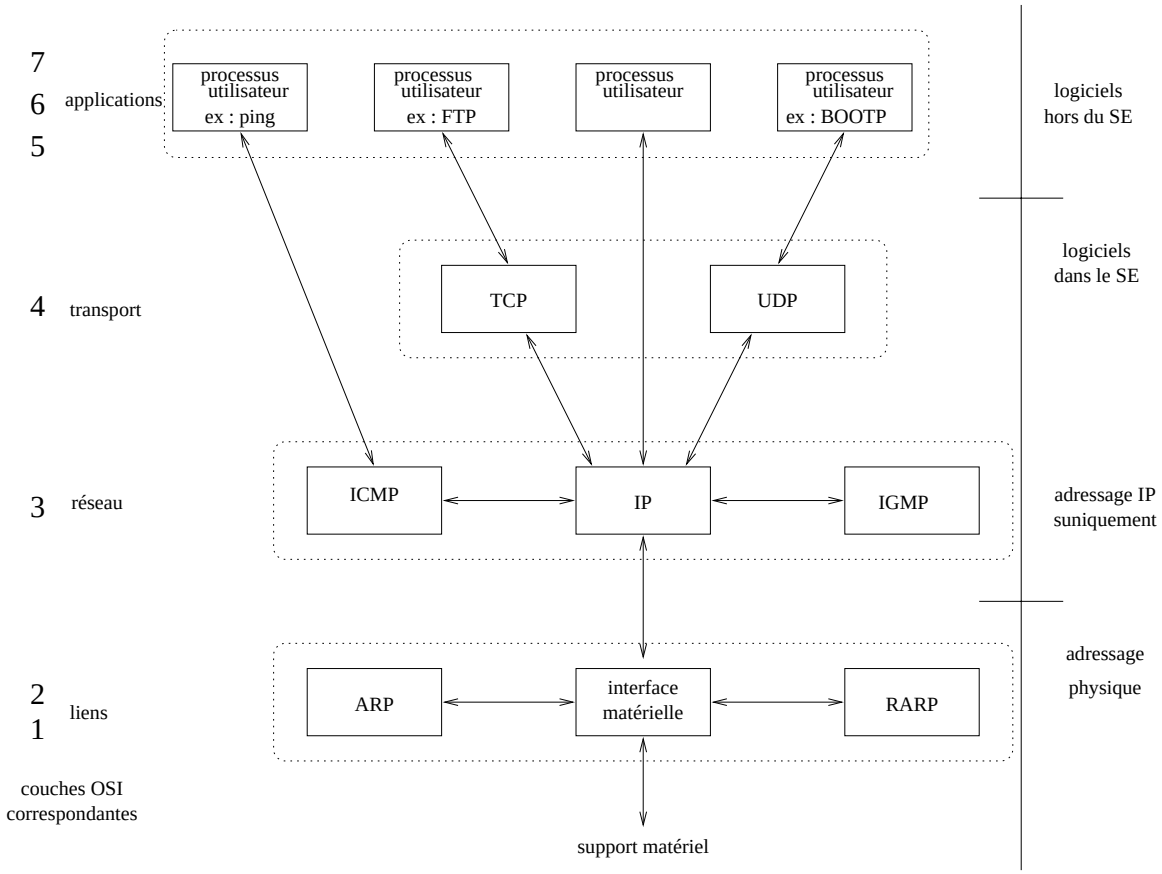

FIG. 2.3 - *Architecture d'une pile TCP/IP*

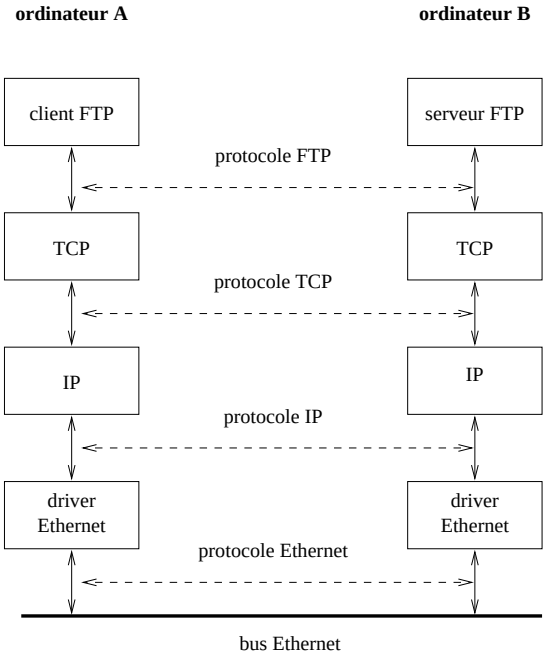

FIG. 2.4 - *Communication entre deux machines du même réseau*

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. 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. 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 $2^{16} = 65536$ 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 $2^{16=65536}$ 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

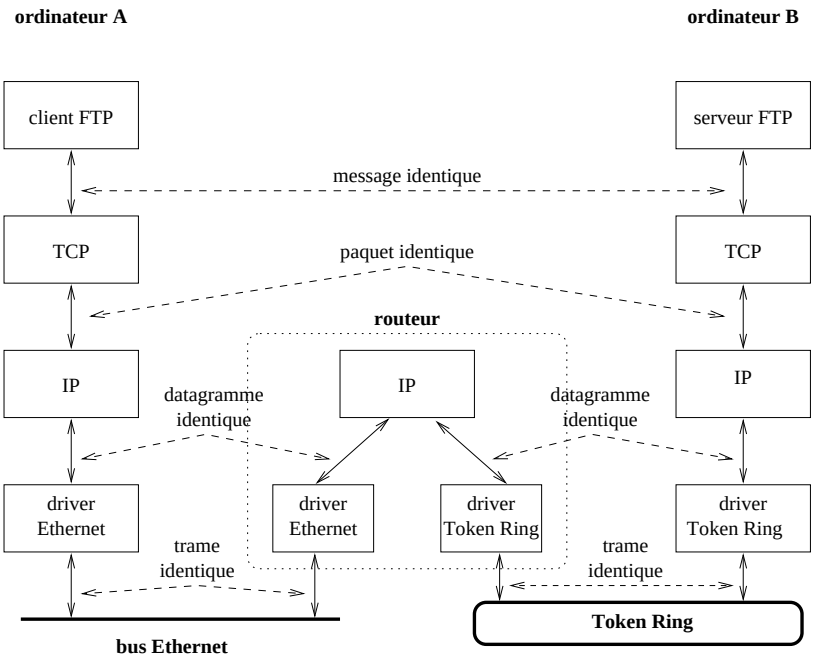

FIG. 2.5 - *Interconnexion de deux réseaux*

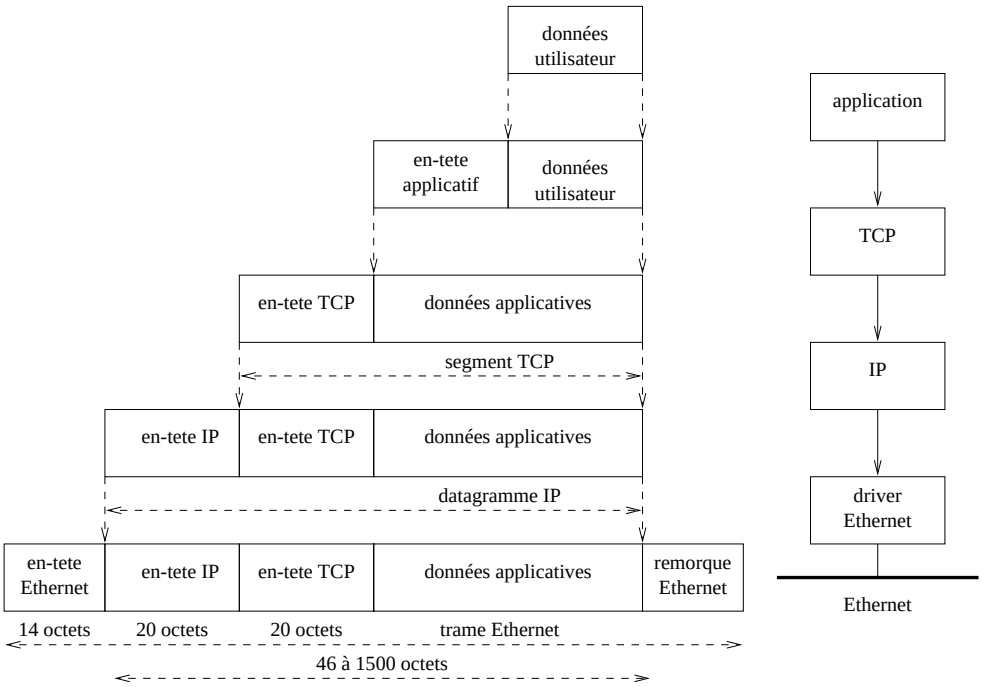

FIG. 2.6 - *Encapsulation des données par la pile des protocoles TCP/IP.*

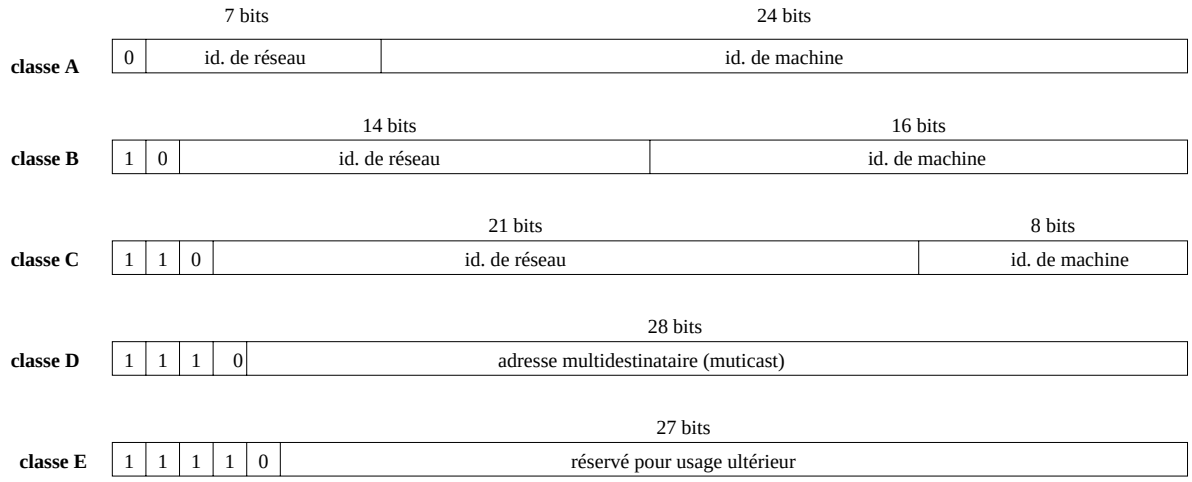

FIG. 2.7 - *Les cinq classes d'adressses IP*

- `<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*[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. La figure 2.8 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

1. Un intranet est un réseau d'étendue géographique très limitée, par exemple pour une entreprise, basé sur la technologie TCP/IP mais non relié à Internet. Un *extranet* est également un réseau privé bâti sur TCP/IP, non connecté à Internet, mais réparti sur des sites géographiques distants

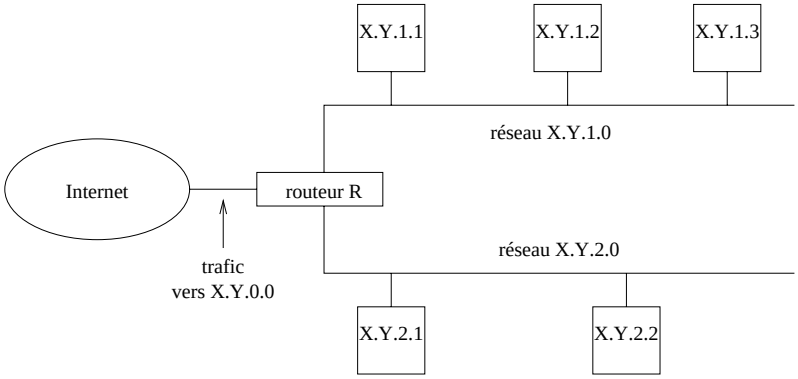

FIG. 2.8 - *Adressage de sous-réseau*

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] `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 2.8 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. 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.

2. En notation décimale pointée.

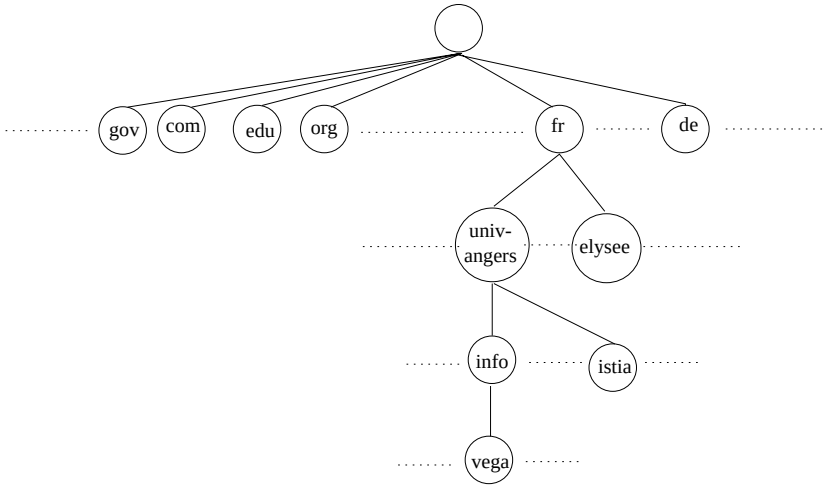

FIG. 2.9 - *Système de noms de domaines.*

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.

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[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[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.

3. Les technologies 100 Mbit/s et Gigabit sont également opérationnelles.

4. En pratique, certains constructeurs ne respectent pas cette règle et plusieurs cartes peuvent avoir la même adresse, voire une adresse nulle.

Au niveau des trames, la normalisation IEEE 802[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. Mais la variante la plus usitée est l'Ethernet.

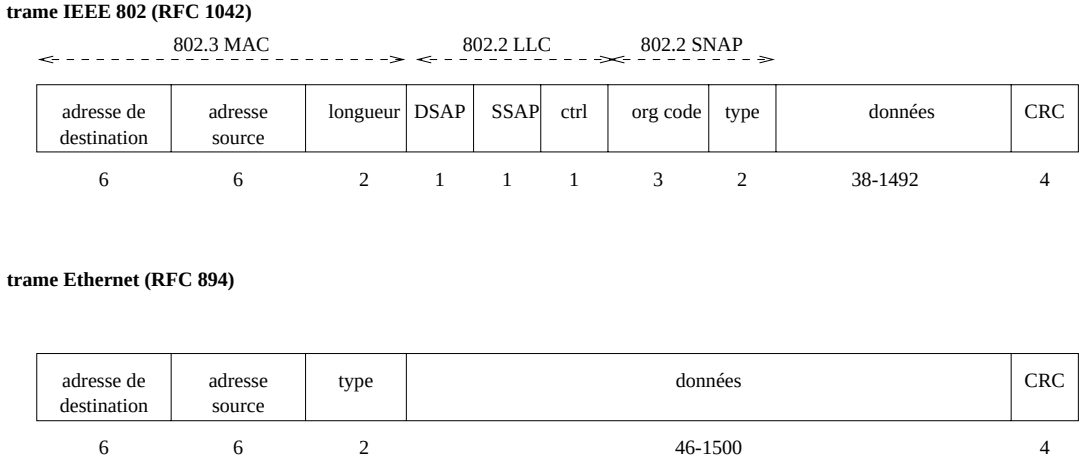

FIG. 2.10 - *Encapsulation Ethernet et IEEE 802.3.*

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, ...).

5. Elle réunit IEEE 802.2 qui définit la sous-couche de contrôle de liens LLC (*Logical Link control*) qui repose sur la sous-couche physique MAC (*Medium Access Control*) IEEE 802.3.

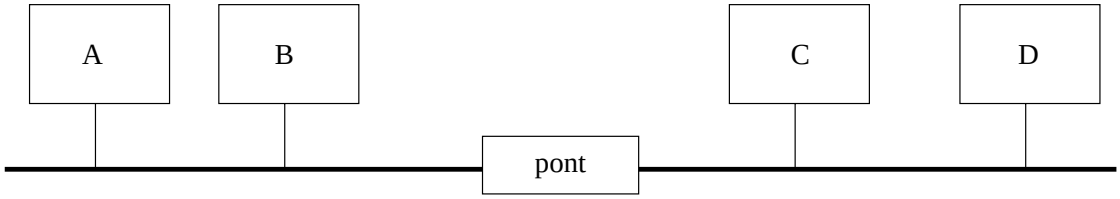

FIG. 2.11 - *Fonctionnement d'un pont..*

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. Pour éviter des problèmes de bruit, certaines implantations de SLIP font

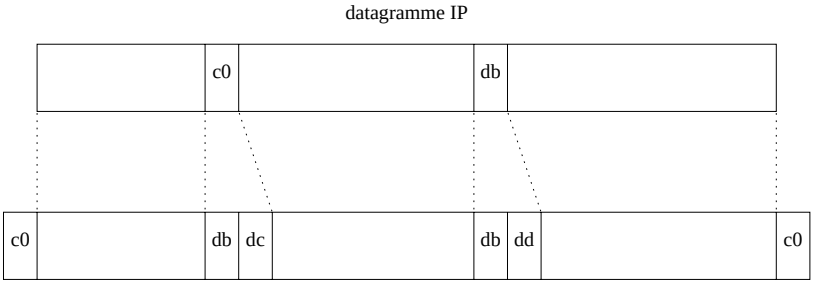

FIG. 2.12 - *Encapsulation SLIP.*

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. Le processus de connexion d'un client équipé d'un

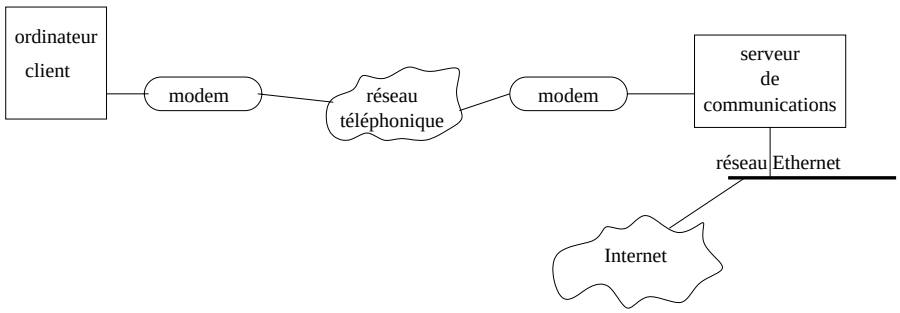

FIG. 2.13 - *Connexion à Internet par modem et PPP.*

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.

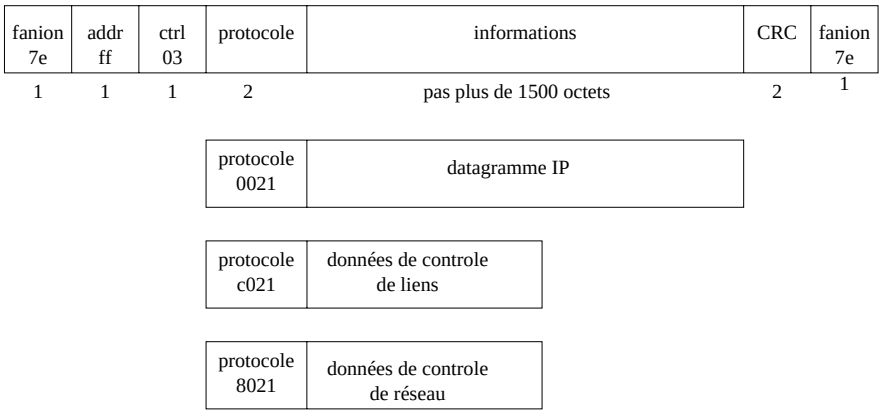

FIG. 2.14 - *Encapsulation PPP.*

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.

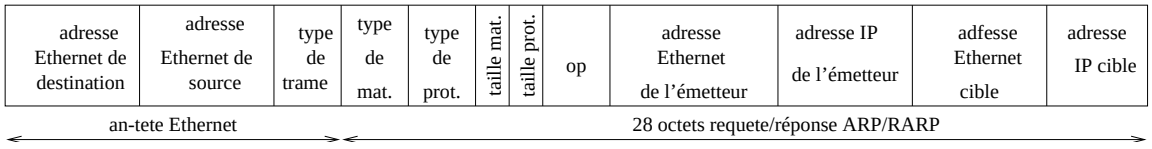

FIG. 2.15 - *Requête ou réponse ARP sur un réseau Ethernet.*

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

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. 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.

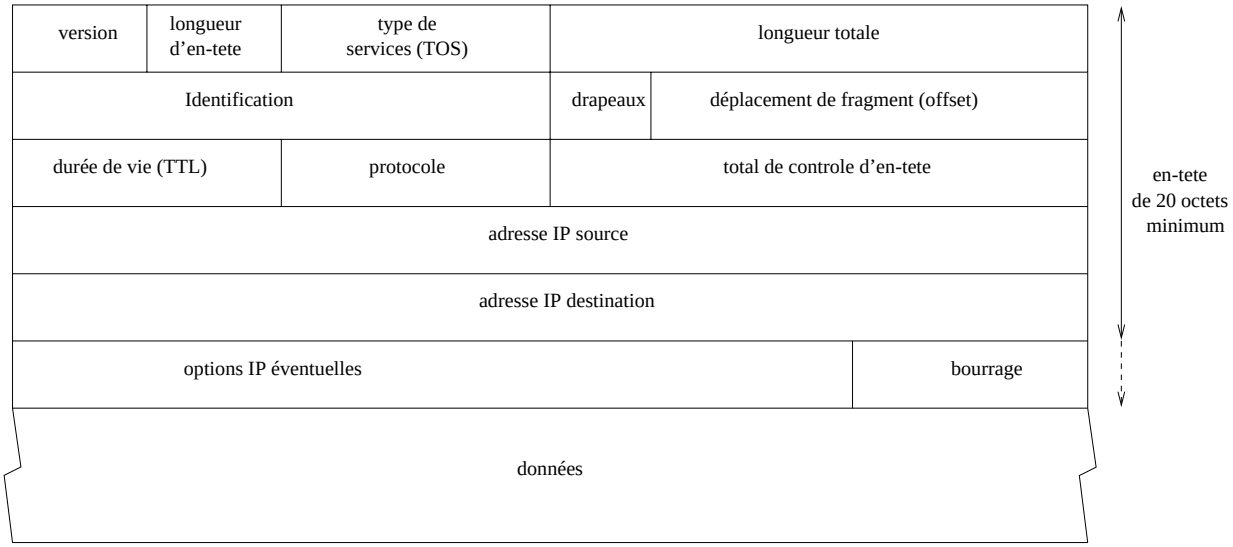

FIG. 2.16 - *Structure d'un datagramme IP.*

- 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. Ces 4 bits servent à améliorer la qualité du routage et

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

TAB. 2.1 - *Type de service pour les applications standard.*

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. Comme

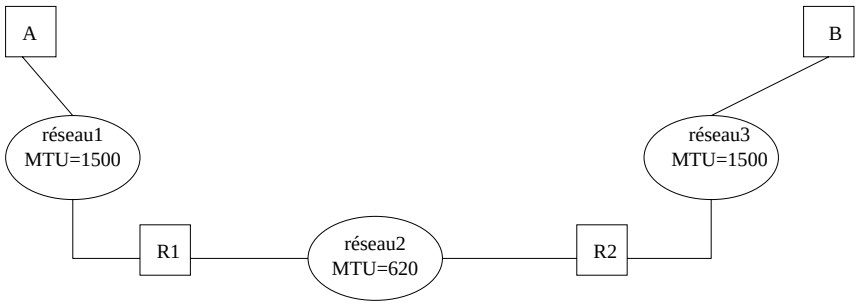

FIG. 2.17 - *Fragmentation d'un datagramme IP.*

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[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. 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[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.

6. Dans le cas d'un routeur, relié à k réseaux différents par k interfaces, donc possédant k adresses IP distinctes, la comparaison sera faite pour chacun de ces réseaux.

7. En binaire, ce masque est `11111111.11111111.11111111.11100000`.

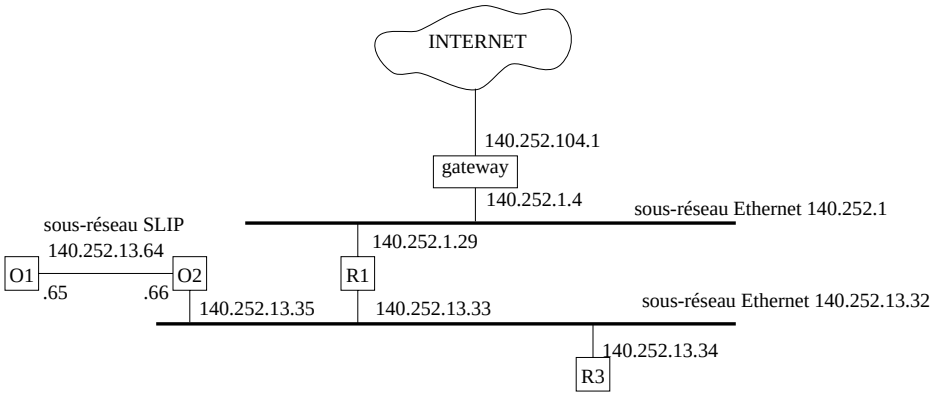

FIG. 2.18 - *Exemple d'interconnexion de réseau.*

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. Le champ *commande* fixé à 1 indique une requête pour demander tout ou partie d'une table de routage

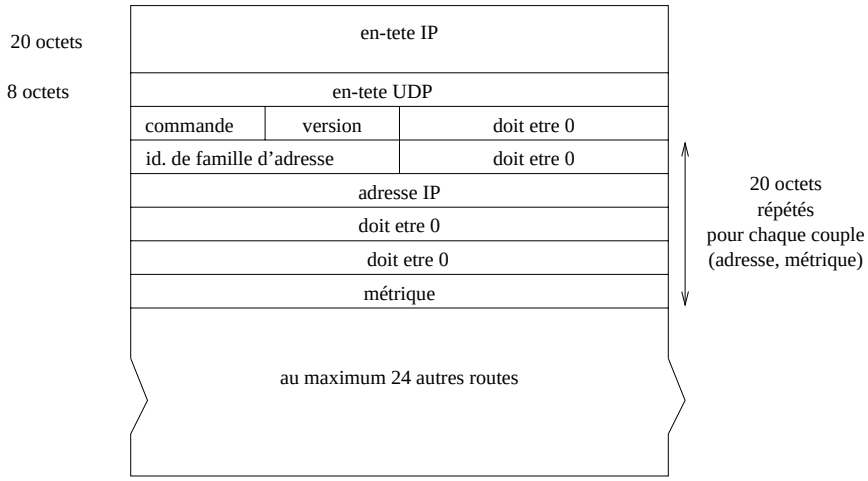

FIG. 2.19 - *Encapsulation d'un message RIP.*

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. 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.

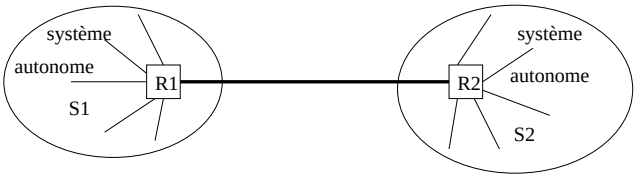

FIG. 2.20 - *Interconnexion de systèmes autonomes.*

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 [8] illustré dans la figure 2.21, il est susceptible, lui aussi, de souffrir d'erreurs de transmission. Mais la règle est qu'aucun message ICMP

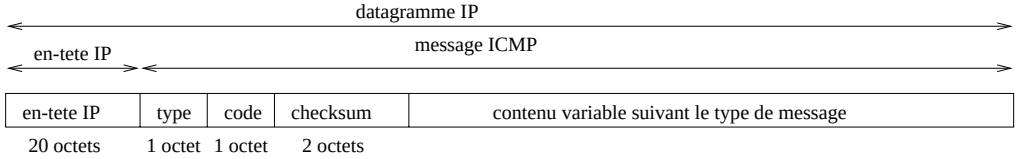

FIG. 2.21 - *Encapsulation d'un message ICMP.*

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.

8. L'en-tête de ce datagramme contient un champ protocole égal à 1.

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.

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[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. Les *numéros de port* (chacun

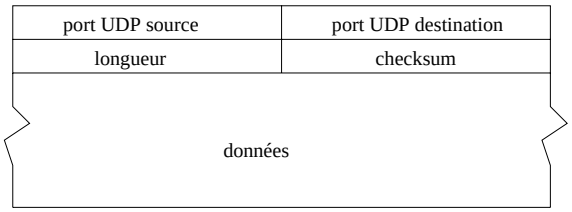

FIG. 2.22 - *Structure d'un datagramme UDP.*

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. Cette pseudo en-tête comprend les adresses IP source[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

9. Certains numéros de port (comme 69) sont statiques c'est-à-dire que tous les systèmes les associent de la même manière aux applications et tous les logiciels applicatifs se conforment à cette règle. D'autres sont dynamiques et un numéro de port est attribué par les logiciels de communication à l'application au moment où celle-ci le demande.

10. L'obtention de cette adresse IP source oblige UDP à entrer en contact avec sa couche IP. En effet, sur un ordinateur relié à plusieurs réseaux l'adresse IP source est celle de l'interface de sortie et celle-ci est fonction de la destination finale et de la table de routage gérée par IP. Ce dialogue UDP/IP préliminaire à l'encapsulation du datagramme UDP est une légère entorse à la notion séparation en couches rendue nécessaire, voire indispensable, pour pouvoir s'assurer pleinement de la remise à bonne destination des datagrammes UDP.

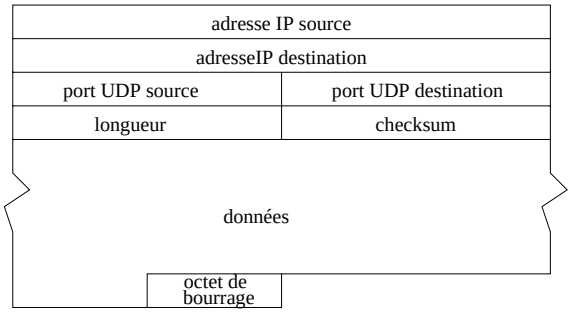

FIG. 2.23 - *Champs utilisés pour le calcul du checksum UDP.*

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 [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.

11. UDP permet lui de mettre en place du broadcasting et du multicasting (via le protocole IGMP) lorsqu'une application veut envoyer un unique message simultanément vers plusieurs destinataires.

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 Chaque

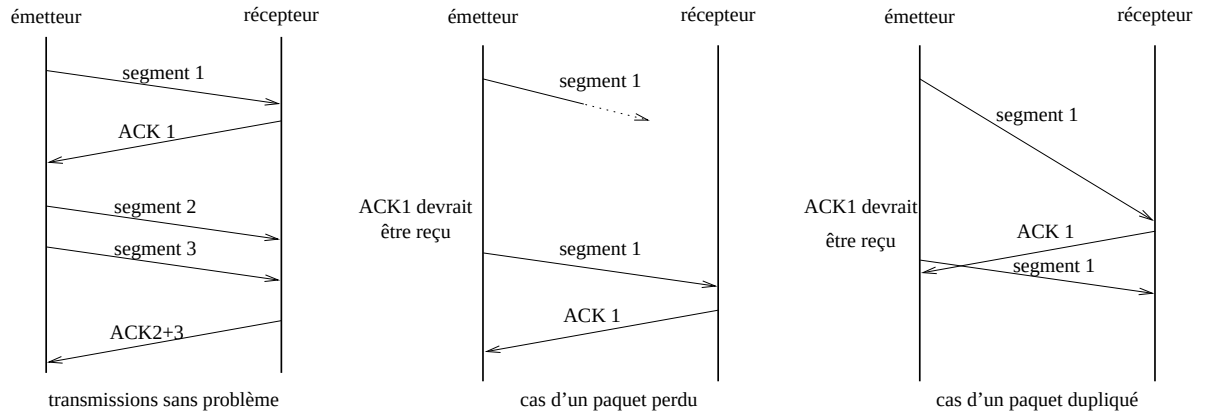

FIG. 2.24 - *Échanges de segments TCP.*

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 [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. L'en-tête, sans option, d'un segment

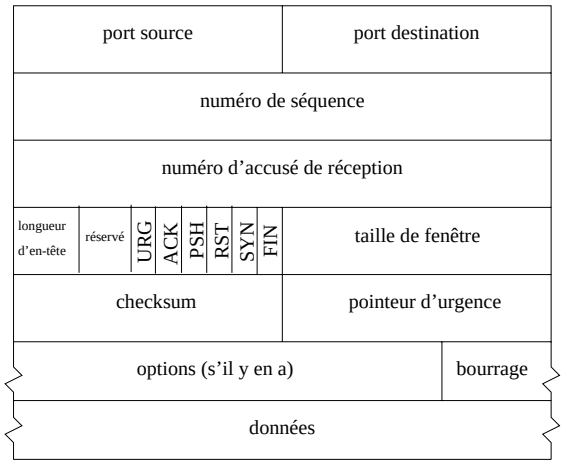

FIG. 2.25 - *Format du segment TCP.*

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

12. Cela peut arriver puisque la couche IP sous-jacente n'est pas fiable.

associant avec les numéros IP source et destination du datagramme IP qui transporte un segment TCP on identifie de manière unique chaque connexion[13].

- Le *numéro de séquence*[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.

13. Cette association (numéro IP, port) est appelée *socket*.
14. C'est un entier non signé codé sur 32 bits qui retourne à 0 après avoir atteint la valeur $2^{31}-1$.

- 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. L'extrémité demandant l'ouverture de la connexion, est le *client*. Il émet un segment SYN (où le bit

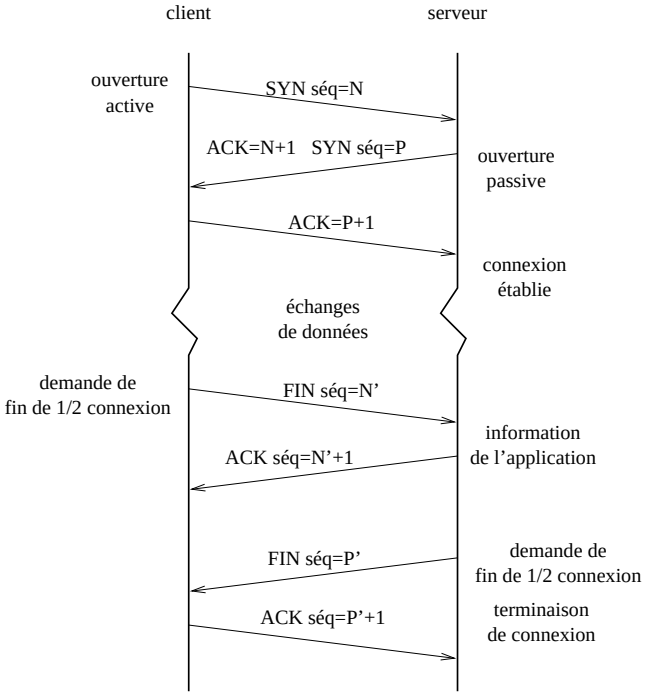

FIG. 2.26 - *Établissemnet et terminaison d'une connexion TCP.*

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=$P+1$[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.

15. Un mécanisme comportant 2 couples de segments (SYN, ACK SYN) gère le cas où deux demandes de connexion ont lieu en même temps chacune dans un sens.

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[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. En fait, TCP gère ce type

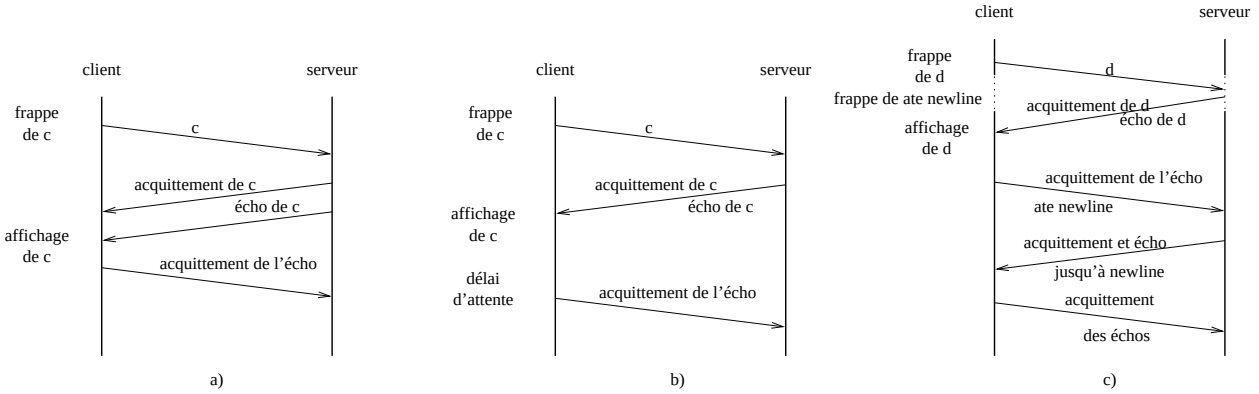

FIG. 2.27 - *Échange de données interactif.*

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[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

16. Aujourd'hui cette statistique a évolué étant donné l'importance du trafic de données de masses dû au protocole http du web qui représente plus de 25% des paquets transmis en avril 95.

17. Pour chacun des 5 caractères `d a t e newline`, il faut 1 octet pour le caractère+20 octets d'en-tête TCP+ 20 octets d'en-tête IP.

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. Dans cet exemple la fenêtre couvre les octets de 4 à 9, car

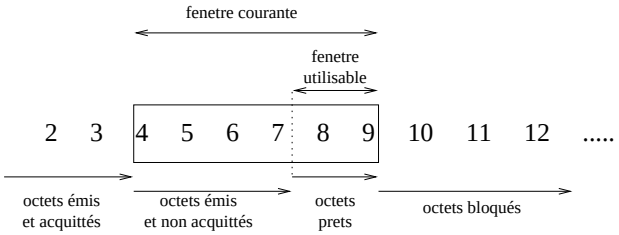

FIG. 2.28 - *Fenêtre glissante de TCP.*

la taille de la fenêtre courante est 6 et que tous les octets jusqu'au troisième inclus ont été émis et acquittés. A tout instant TCP calcule sa fenêtre utilisable qui est constituée des octets présents dans

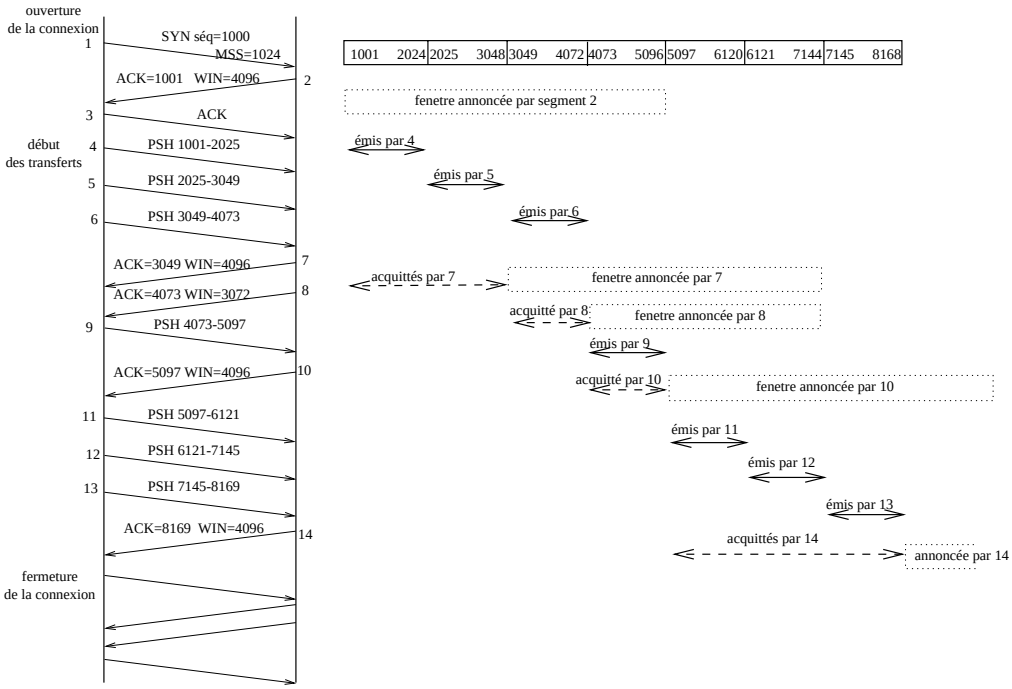

FIG. 2.29 - *Exemple d'évolution de fenêtre glissante.*

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 [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.

18. Seules les informations nécessaires à la compréhension du processus de fenêtre glissante sont présents, d'autres détails sont omis.

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.

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. 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

code	type de matériel	longueur adresse matériel	compteur de saut
id. de transaction			
nombre de secondes		non utilisé	
adresse IP du client			
votre adresse IP			
adresse IP du serveur			
adresse IP du routeur			
adresse matérielle du client			
nom de machine du serveur			
nom du fichier de boot			
information spécifique du vendeur			

FIG. 2.30 - *Format de requête ou réponse BOOTP.*

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. 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.

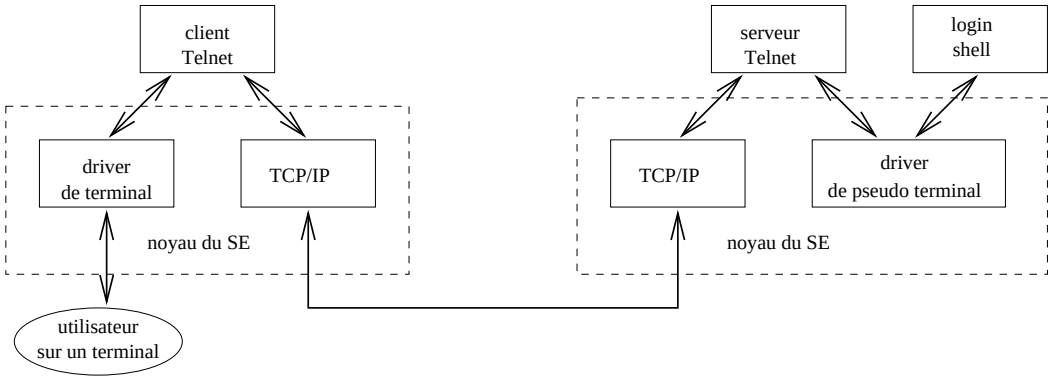

FIG. 2.31 - *Schéma de focntionnement de l'application Telnet.*

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. NFS utilise principalement UDP, mais ses nouvelles implantations utilisent également TCP. Lorsqu'un

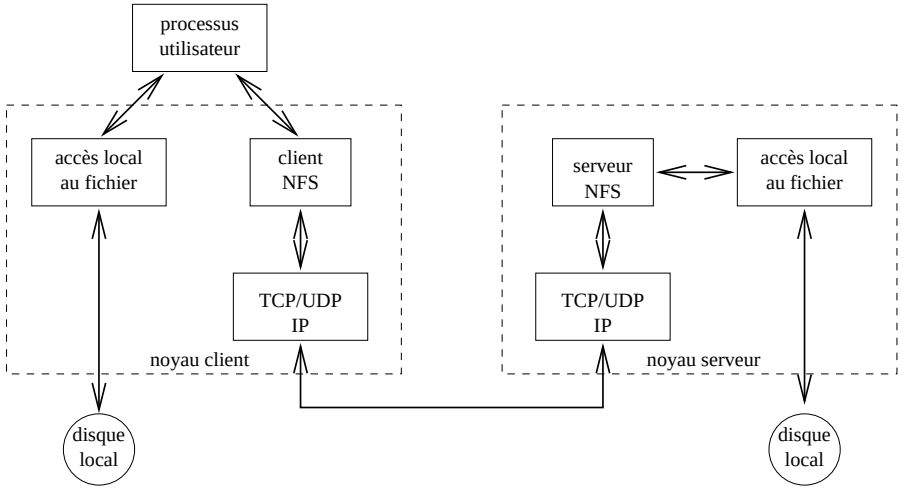

FIG. 2.32 - *Configuration client serveur de NFS.*

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*.

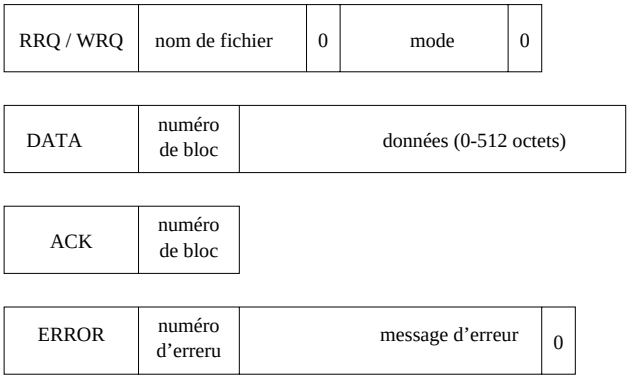

FIG. 2.33 - *Format des messages TFTP.*

- `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. Tout d'abord on y voit que le client utilise FTP à travers une interface

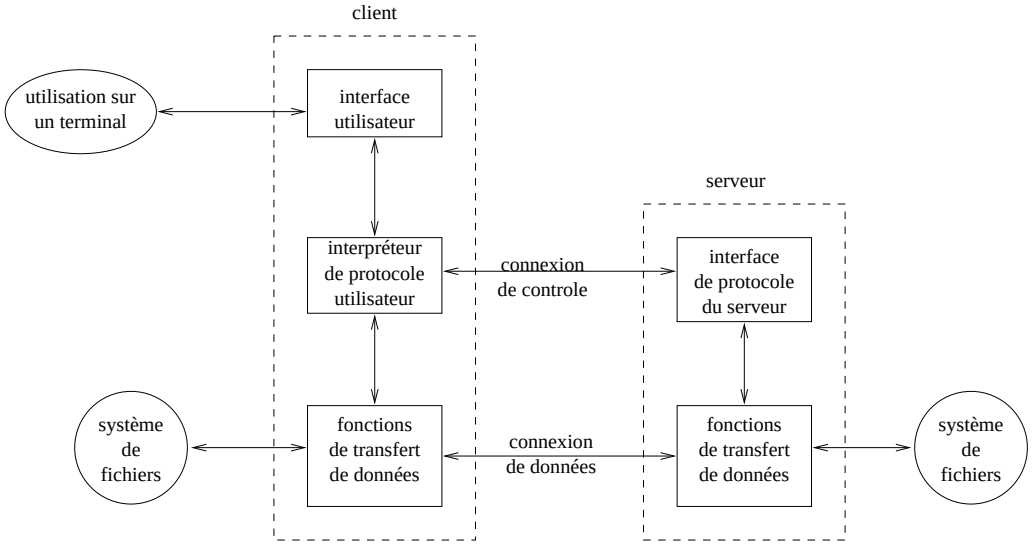

FIG. 2.34 - *Tansfert de fichier par FTP.*

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. 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

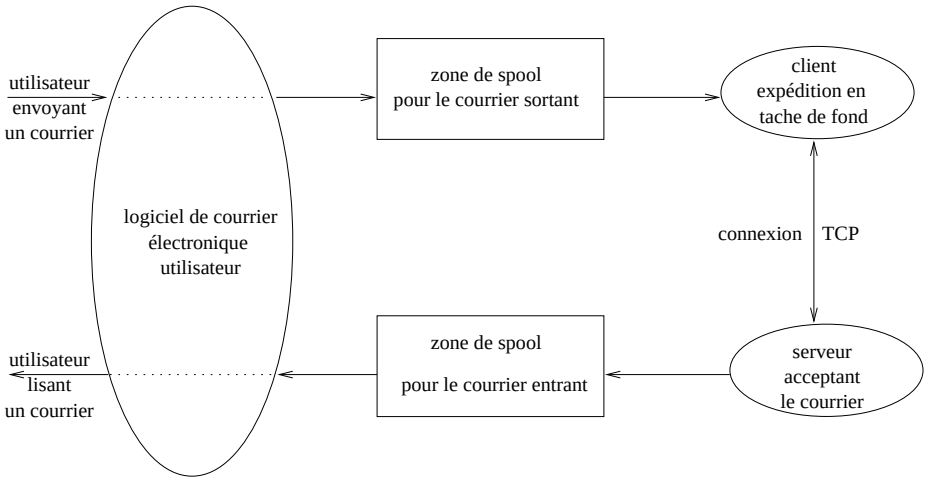

FIG. 2.35 - *Schéma d'une messagerie SMTP.*

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 de connexion		`220 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[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.

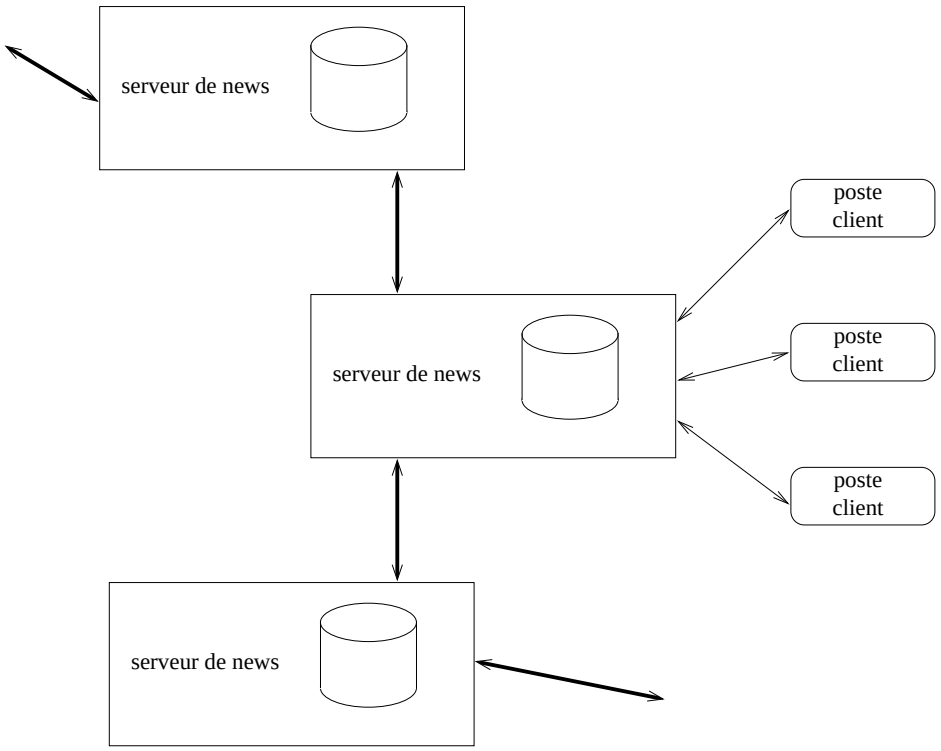

FIG. 2.36 - *Communication au sein de Usenet.*

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

19. Le contenu des news, la description des groupes et l'utilisation pratique d'un logiciel lecteur de news est supposée connue ici.

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|subscriptions]
  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.
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
  ```

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

TAB. 2.2 - *Signification des 2 premiers chifres des codes de réponses de NNTP*

```
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[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
```

20. L'utilisation du web est supposée connue ici.

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

TAB. 2.3 - *Codes de réponses de HTTP*

```
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.

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.*

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.

Chapitre 3

Applets Java.

Ce chapitre est traité lors de travaux dirigés et travaux pratiques et aborde les généralités nécessaires à l'écriture d'applets.

Chapitre 4

Installation d'un intranet

Ce chapitre est traité lors d'une séance de travaux pratiques qui consiste en l'intallation d'un système Linux RedHat (à partir du serveur `ftp.univ-angers.fr`) et la configuration rapide des services réseaux (web, news, mail, ftp, ...).