Introduction aux réseaux

Jean-Yves Didier

Table des matières

Table des figures

1 Généralités sur les réseaux

1.1 Définitions

1.1.1 Dans la vie de tous les jours

Un *réseau* est, dans le vocabulaire de tous les jours, un ensemble d'objets ou de personnes connectées ou maintenues en liaison. Par extension, il peut venir à en désigner l'ensemble des connections établies. L'origine étymologique de ce mot est le terme latin *rete* qui signifie *filet*. Ce terme est approprié puisqu'un filet est constitué de noeuds et de liens entre ces derniers. Par similitude, les objets ou personnes reliées sont appelées *noeuds du réseau*.

Ainsi, le terme réseau peut s'appliquer à divers domaines : le réseau social désigne un ensemble de personnes qui se connaissent et restent en contact entre elles, le réseau ferroviaire désigne les lignes de chemin de fer ainsi que les gares. Mais les réseaux ne se limitent pas à ces deux domaines. Ainsi, il existe des réseaux électriques, téléphoniques, ou informatiques.

1.1.2 Les réseaux informatiques

Plus spécifiquement, un réseau informatique désigne *un ensemble de machines inter-connectées qui servent à échanger des flux d'information*. Un réseau informatique répond donc au besoin d'échanger des informations (besoin de communiquer).

Il convient toutefois de faire attention aux situations dans lesquelles le terme réseau est employé en informatique. En effet, il peut désigner :

– L'ensemble des machines,
– Le protocole de communications,
– La manière dont les équipements sont connectés.

Par la suite, lorsque que nous emploierons le terme *réseau(x)*, celui-ci se référera à l'expression *réseau(x) informatique(s)*. Un réseau informatique possède certaines propriétés que nous allons détailler.

1.2 Propriété des réseaux

Les réseaux peuvent s'étendre sur une échelle géographique, avoir une topologie (dont nous reparlerons ultérieurement en section 2.1 page 8) ou un mode d'acheminement des messages.

1.2.1 L'échelle géographique

Les réseaux sont classés en fonction de leur étendue géographique. Ainsi, on parlera de :

- PAN Personnal Area Network pour les réseaux personnels de moins d'une dizaine de machines,

- LAN Local Area Network pour les réseaux locaux à l'échelle d'un bâtiment (par exemple le réseau de l'IUP),

- MAN Metropolitan Area Network pour les réseaux construits à l'échelle d'une ville ou d'un campus (par exemple le réseau REVE reliant les établissements de formation et de recherche de la ville d'Évry),

- WAN Wide Area Network pour les réseaux à l'échelle d'un pays ou mondiale (par exemple RENATER, le réseau reliant les établissements de la recherche française).

Le nombre de machines mises en réseau va définir la différence entre un LAN ou un PAN. En revanche, il est intéressant de noter que les MAN et les WAN peuvent être composés de plusieurs LAN ou PAN. Il y donc non seulement interconnexion de machines mais aussi de réseaux au travers desquels les messages seront acheminés.

1.2.2 Le mode d'acheminement des messages

Comme la mise en réseau de machines correspond au besoin d'échanger des informations, des messages transitent en permanence dans un réseau. Il existe plusieurs stratégies pour acheminer un message au travers du réseau. On parle alors de techniques de *commutation*. Nous détaillerons les 5 méthodes les plus courantes à savoir :

- La commutation de circuits,
- La commutation de messages,
- La commutation de paquets,
- La commutation de trames,
- La commutation de cellules.

La commutation de circuits est une méthode utilisée sur les réseaux téléphoniques classiques. Les communications passent par trois phases distinctes :

- L'établissement de la liaison où l'on va chercher et occuper un itinéraire pour acheminer le message (ce qui est fait lorsque l'on décroche et compose un numéro. La sonnerie indique que la liaison est établie),
- Le maintient de la liaison pendant toute la durée de la connexion,
- La libération des connexions sur ordre et le retour à l'état libre du réseau.

La commutation de messages s'appuie sur les noeuds du réseau. Ainsi, le message transite de noeuds en noeuds jusqu'au destinataire. Un noeud ne peut envoyer le message que s'il a reçu ce dernier complètement. Ce mode de commutation a été abandonné et remplacé au profit de la commutation de paquets.

La commutation de paquets consiste à fragmenter le message en paquets qui seront transmis de noeuds en noeuds jusqu'au destinataire. Il existe plusieurs stratégies pour acheminer les paquets.

- Dans le *mode connecté* les paquets empruntent toujours le même chemin (c'est le cas du réseau TRANSPAC),
- Dans le *mode non-connecté* les paquets peuvent emprunter des itinéraires différents. En réalité, chaque noeud va se charger d'aiguiller l'information. C'est sur ce principe que sont échangées les informations sur internet.

Comme il s'agit de l'un des plus répandus modes de commutation, ce dernier fait l'objet d'une norme internationale qui est l'oeuvre des opérateurs téléphoniques : la norme X25. Pour permettre l'acheminement et le réassemblage des paquets, ce dernier renferme les informations suivantes :

- Un identifiant de source,
- Un identifiant de destination,
- Un numéro de séquence,
- Un bloc contenant les données proprement dites,
- Un code de vérification des erreurs.

La commutation de trames est une extension de la commutation de paquets. La commutation est assurée directement au niveau du matériel employé pour construire le réseau

La commutation de cellules est également une extension de la commutation de paquets à laquelle on ajoute les avantages de la commutation de circuits. Les paquets ont une longueur fixe de 53 octets (avec 5 octets d'en-tête). La connexion est mise en place avant toute émission de cellule. Le réseau ATM (pour Asynchronous Transfer Mode) exploite cette solution. A ce type de commutation est associée la notion de *qualité de service* qui dépend du type d'informations transportées.

Pourquoi la commutation par paquets est un mode de commutation prépondérant ? Il suffit de tracer sur un chronogramme le temps nécessité pour acheminer un message d'un noeud à l'autre du réseau en passant par plusieurs noeuds intermédiaires pour comprendre la raison du choix de la commutation par paquets (voir figures 1 et 2). On constate immédiatement que les temps de transmission au travers de plusieurs noeuds sont inférieurs si la commutation de paquets est employée à la place de la commutation de messages.

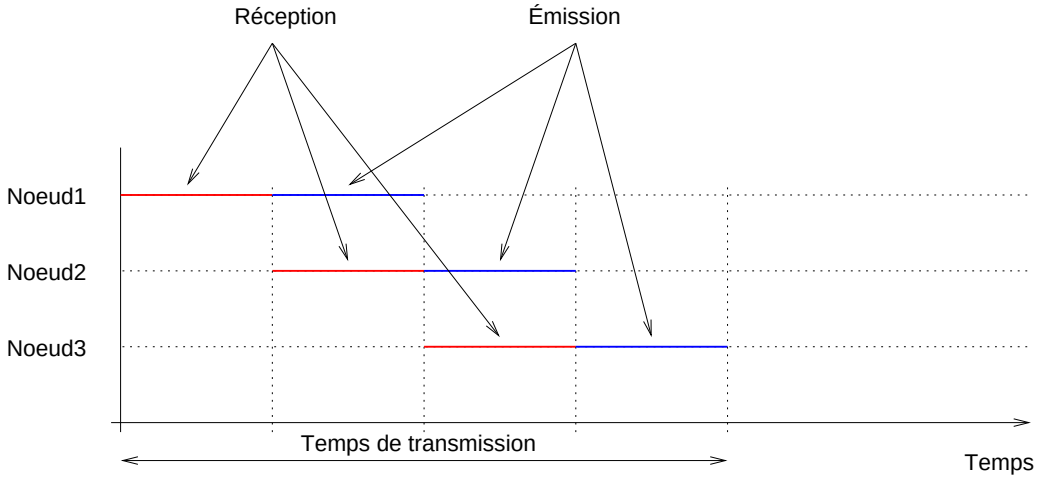

FIG. 1 – Commutation de messages

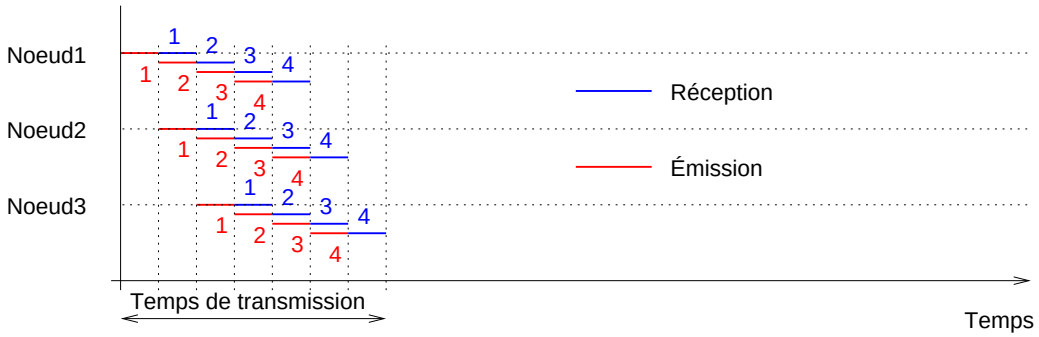

FIG. 2 – Commutation de paquets

En réalité, comme la commutation par paquets est un mode de commutation très répandu (puisqu'utilisé par internet), ce mode fait l'objet d'une norme pour décrire les réseaux. Il s'agit de la norme OSI (pour Open Systems Interconnections).

1.3 La norme OSI

Cette norme ISO a été crée en 1984. Elle s'intéresse aux réseaux à commutation de paquets. Il s'agit d'un modèle à 7 couches (voir figure 3 page suivante) qui décrit les réseaux. Chaque couche possède une problématique qui lui est propre. La mise en place d'un réseau consiste à trouver une solution par couche. Les couches sont en théorie indépendantes. Il est donc possible de modifier la solution adoptée pour une couche sans affecter les couches supérieures ou inférieures. Le modèle OSI décrit ainsi les réseaux des couches basses qui concernent le matériel qui constitue le réseau jusqu'aux couches hautes qui décrivent la communication entre les applications fonctionnant à l'aide du réseau constitué.

Pour chacune des couches, nous aborderons les problèmes à résoudre.

1.3.1 La couche *Matériel*

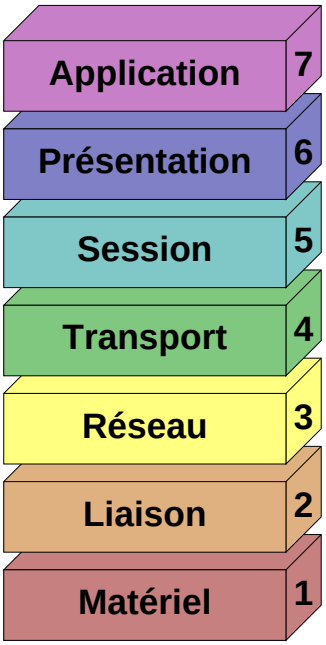

FIG. 3 – Les couches du modèle OSI

Cette couche décrit le support physique employé pour transmettre l'information (fibre optique, câble réseau conventionnel, ondes pour les réseaux sans fils, etc.) ainsi que la topologie du réseau en lui-même c'est à dire comment les machines sont reliées les unes aux autres.

1.3.2 La couche *Liaison*

Dans cette couche, le problème à résoudre est celui de la communication entre plusieurs machines reliées par le même support physique. Ainsi, il faut pouvoir identifier deux machines situées sur le même réseau physique. De même, il faut assurer la transmission des informations sans erreur d'une machine à l'autre ?

1.3.3 La couche *Réseau*

Cette couche s'intéresse à l'interconnection de plusieurs réseaux physiques. Ainsi, les problèmes à résoudre sont l'acheminement d'un paquet d'un point du réseau à un autre (ce que l'on appelle le *routage*), sachant que l'arrivée et le départ ne sont pas sur le même support physique, l'interconnexion de supports physiques et de réseaux hétérogènes ainsi que le contrôle et la régulation du trafic sur le réseau. A ce niveau apparaissent des protocoles de communication réseau tels que le protocole IP (pour Internet Protocol) utilisé par Internet.

1.3.4 La couche *Transport*

Les problèmes abordés dans cette couche concerne le découpage d'un message en paquets, la restitution de ce dernier à partir des paquets. Cette couche doit aussi permettre de s'assurer de la bonne réception du message. Les protocoles couramment utilisés dans cette couche sont les protocoles TCP et UDP.

1.3.5 La couche *Session*

Lorsque deux utilisateurs sont mis en relation, ces derniers effectuent des transactions dans le cadre d'une session. Le problème à résoudre est donc l'établissement d'une session. Il faudra également déterminer les mécanismes de synchronisation à employer.

1.3.6 La couche *Présentation*

La problématique est celle de la présentation de l'information qui transite sur le réseau. Doit-on compresser ou même crypter les données ? Sous quelle forme doit elle circuler ? Doit on coder l'information ?

1.3.7 La couche *Application*

Cette couche s'intéresse aux protocoles à associer aux applications qui nécessitent un support réseau pour échanger des informations. C'est le cas des protocoles pour les échanges d'e-mail, la navigation de pages en pages internet, etc ...

1.3.8 Les limites du modèle OSI

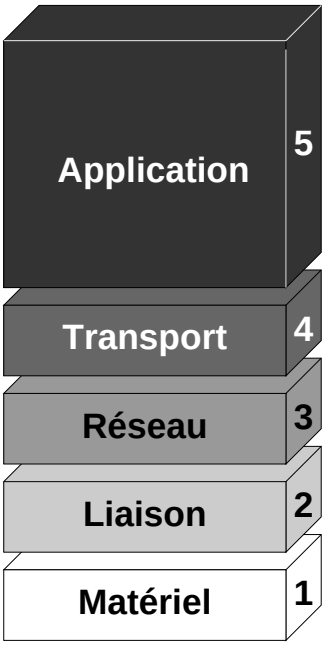

FIG. 4 – Les couches de la famille de protocoles IP

Le modèle OSI est un protocole développé après nombre de protocoles réseaux alors déjà très répandus qui ne respectent pas cette norme. De plus, dans la réalité, l'indépendance des différentes couches n'est pas garantie. Ainsi, la famille de protocoles la plus utilisée de nos jours, la famille de protocoles IP ne respecte pas la norme OSI à la lettre. En effet, les couches Session, Présentation et Application sont confondues dans cette famille de protocoles en une seule couche applicative (voir figure 4).

Par la suite, pour examiner la constitution des réseaux, nous nous appuierons sur le modèle en couches de la famille de protocoles IP. Ainsi, nous verrons comment constituer physiquement un réseau (couche matériel), comment faire communiquer deux machines situées sur le même support physique (couche liaison), comment interconnecter plusieurs réseaux (couche réseau), comment fragmenter et défragmenter les messages tout en contrôlant la manière dont ils sont transportés (couche transport) et enfin comment sont bâties les applications employant le réseau comme support de communication (couche application).

2 La couche *Matériel* : construction physique d'un réseau

Dans cette partie, nous verrons quelles sont les *topologies* que nous pouvons donner à un réseau, les manières de partager le support physique entre plusieurs stations ainsi que les normes internationales qui y sont associées. Nous détaillerons également de manière plus approfondie le matériel employé pour constituer des réseaux de type Pronet-10, Ethernet et WiFi.

2.1 Topologies des réseaux

Lorsqu'il s'agit de connecter 2 stations l'une à l'autre, un seul câble suffit pour les faire communiquer. Toutefois, comment cela se passe si l'on dispose d'une grande quantité de stations. Plusieurs stratégies peuvent être employées pour relier les stations. Ces stratégies de câblages donnent plusieurs configurations physiques de réseaux appelées topologies. Parmi les topologies classiques, nous comptons :

- le câblage en mailles,
- le câblage en bus,
- le câblage en étoile,
- le câblage en anneau,
- le câblage en arbre,
- le câblage dit 'Starlan'.

2.1.1 Le câblage en mailles

Comme nous pouvons le voir à la figure 5(a) page suivante, ce câblage constitue une généralisation du cas à deux stations. En effet, chaque station du réseau est reliée à une autre par un câble. Ce système est inusité de nos jours en raison du nombre de câbles nécessaires pour créer un tel réseau (pour n stations, cela fait $n(n+1)/2$ câbles). De plus, rajouter une station sur ce réseau signifie que l'on doit rajouter autant de câbles que de stations déjà présentes sur le réseau.

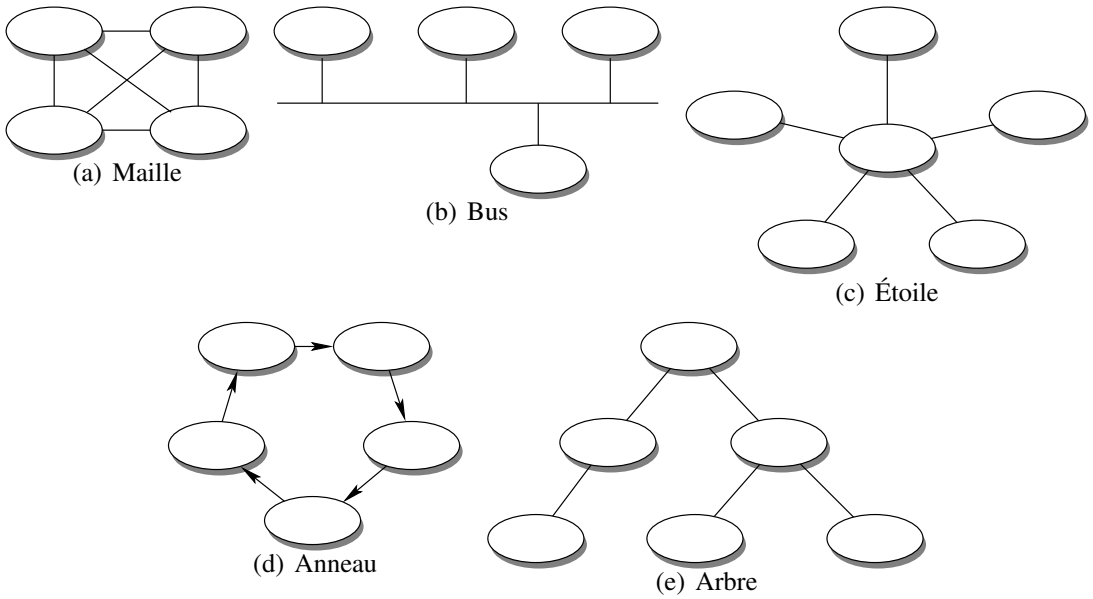

FIG. 5 – Les diverses topologies des réseaux

2.1.2 Le câblage en bus

Dans ce type de câblage (voir figure 5(b)), toutes les stations se partagent le même support physique de communication (le *bus*). Si les stations cherchent à communiquer en même temps, ce partage du même support entraîne la *collision* des différents messages. Il faudra mettre au point des stratégies anti-collision. Les anciens réseaux Ethernet fonctionnent avec ce type de topologie.

2.1.3 Le câblage en anneau

Un anneau est unidirectionnel (le message se propage toujours dans le même sens dans les câbles - voir figure 5(d)). À la différence des topologies précédentes, c chaque station qui reçoit un message va alors le réemettre, donc le transmettre à la station suivante, si elle n'est pas concernée par le message en question. Toutefois, si l'un des câbles est défectueux ou une des stations tombe en panne, l'ensemble du réseau est coupé. Pour éviter ce problème, l'anneau est parfois doublé avec un circuit allant dans le sens contraire. Les réseaux de type Pronet-10 fonctionnent avec une topologie en anneaux.

2.1.4 Le câblage en étoile

Toutes les stations sont reliées à un noeud central (une station du réseau) qui sera le seul à relayer les messages (figure 5(c)). Dans ce type de topologie, le point faible réside dans le noeud central. La fiabilité du réseau est équivalente à celle du noeud central.

2.1.5 Le câblage en arbre

Cette topologie est une généralisation de la topologie en étoile (figure 5(e)). Les stations sont montées en cascade les unes par rapport aux autres. Bien évidemment, la fiabilité du réseau est conditionnée par la fiabilité de la *racine* de l'arbre. De plus, si une station est en panne, la branche qui dépend de cette station est alors isolée du réseau principal.

2.1.6 Le câblage Starlan

Ce type de topologie tient à la fois du bus et de l'étoile. Chaque station est branchée à un appareil central qui se charge de redistribuer les messages. Cette appareil peut être assimilé à un support physique puisqu'il n'est pas une station du réseau. En revanche, la fiabilité du réseau est conditionnée par la fiabilité de cet appareil comme pour les réseaux en étoile. Il est donc nécessaire d'assurer la fiabilité de cet équipement actif au centre du réseau. Cette topologie est couramment utilisée de nos jours car elle permet d'utiliser les câbles du réseau téléphonique. C'est cette structure qui est à présent employée dans les réseaux éthernet.

À la différence du câblage en maille, toutes les autres topologies supposent que les messages se partagent les mêmes supports physique. Nous allons aborder à présent les techniques de partage employées.

2.2 Les techniques de partage du support physique

Lorsque plusieurs machines se partagent le même support physique de transmission, ces dernières peuvent émettre des messages simultanément. Lorsque ceci se produit, les messages se 'mélangent' et il se produit ce que l'on appelle une *collision*.

Il existe plusieurs solutions pour éviter ce type de conflit. Les trois solutions les plus couramment utilisées sont :

- L'allocation statique de *bande passante*,
- Les accès au support par compétition par le biais de l'écoute de la porteuse CSMA/CD,
- Les accès au support par élection en employant les techniques à jeton.

2.2.1 L'allocation statique de *bande passante*

La *bande passante* est une mesure de capacité de transmission de l'information dans un support réseau. Elle est généralement exprimée en bits par seconde. L'idée derrière l'allocation statique de bande passante est que, si N stations se partagent le même support physique de bande passante ou capacité C bits/s alors, on attribuera à chaque station une bande passante de C/N bits/seconde.

Cette technique est utilisée dans les réseaux téléphoniques. Toutefois, elle ne convient pas pour les réseaux informatiques. En effet, si les messages sont transmis de manière sporadiques, cette gestion n'optimise pas la vitesse de transmission des données. c'est pour cela que sont apparues les autres techniques d'accès au support.

2.2.2 Les accès au support par compétition

Ces accès se font par le biais de l'écoute de la *porteuse CSMA/CD*. Elle est employée sur les topologies en bus et les topologies de type Starlan. Les accès multiples au support sont possibles (MA = Multiple Access). Chaque machine écoute et détecte si un signal est présent ou non sur le réseau (CS = Carrier Sense). Le principe de fonctionnement de l'algorithme CSMA est le suivant :

- Si aucun signal n'est détecté, alors émettre,
- Si un signal est détecté, alors différer la transmission.

Toutefois un tel algorithme ne permet pas de détecter les cas de transmission simultanée pendant lesquels les collisions sont possibles. C'est pourquoi le système employé est nommé CSMA/CD (CD = Collision Detection). L'idée, dans le but de minimiser les pertes par détection de collision est la suivante :

- une écoute préalable est effectuée,
- une écoute est effectuée pendant l'émission pour détecter les collisions,

– une écoute est effectuée durant le double du temps de propagation nécessaire pour atteindre le point le plus éloigné du bus,
– si un signal est émis par une autre station pendant ces temps d'écoute, alors il y a collision,
– si il y a collision, alors la transmission est arrêtée. Le message sera réémis après un temps tiré aléatoirement.

C'est ce système qui est employé sur les réseaux de type Ethernet. Toutefois, ce type de système, qui optimise la vitesse de transmission des informations, ne garantit pas la durée effective de la transmission. C'est pourquoi, dans les systèmes temps-réel, une technique d'accès par élection a été développée.

2.2.3 Les accès au support par élection

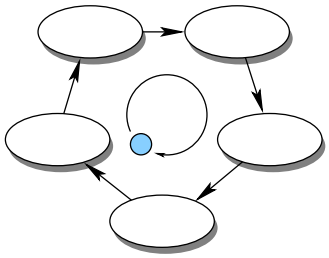

FIG. 6 – Le système token-ring

Cette technique est aussi appelée technique à jeton et est employée sur les topologies en anneau d'où le nom anglais de *token ring* (token = jeton, ring = anneau) comme on le voit sur la figure 6. Dans le principe, une seule trame (message) circule dans le réseau en permanence. De plus, un seule station émet à la fois. Enfin, le jeton contrôle l'accès au support physique.

Pour émettre un message, la station émettrice doit donc se conformer au protocole suivant :

– Capturer le jeton quand il passe à sa portée,
– Émettre une trame,
– Vérifier que le destinataire a reçu le message,
– Libérer le jeton et le passer à la station suivante,

En cas de destruction inopinée du jeton, il existe des algorithmes pour le régénérer.

Les différents réseaux évoqués sont soumis à des normes particulières que nous allons examiner à présent.

2.3 Normes associées aux réseaux physiques

Les normes pour les réseaux locaux datent du mois de février 1980. Elles sont compatibles avec la norme OSI bien qu'elles lui soient antérieures. Il s'agit de la série des normes IEEE 802. Selon ces normes, les réseaux locaux sont subdivisés en 12 catégories. Le tableau 1 donne la liste de ces normes.

Norme	Objet de la norme	Nom anglophone
802.1	Fonctionnement inter-réseaux	Internetworking
802.2	Le contrôle des liaisons logiques	Logical Link Control
802.3	Les réseaux locaux en bus logique	Ethernet LAN
802.4	Les réseaux locaux en bus à jeton	Token Bus LAN
802.5	Les réseaux locaux en anneau logique	Token Ring LAN
802.6	Les réseaux métropolitains MAN	Metropolitan Area Network
802.7	La transmission en large bande	Broadband Technical Advisory Group
802.8	La fibre optique	Fiber-Optic Technical Advisory Group
802.9	Les réseaux intégrant la voix et les données	Integrated Voice / Data Networks
802.10	La sécurité des réseaux	Network security
802.11	Les réseaux sans fil	Wireless network
802.12	La méthode d'accès priorité à la demande	Demand Priority Access on LAN

TAB. 1 – Liste des normes IEEE 802 concernant les réseaux locaux

Nous allons voir à présent des exemples de solutions développées pour la couche matériel du réseau.

2.4 Exemple de solutions matérielles pour les réseaux

2.4.1 Le réseau Ethernet

Le réseau Ethernet est décrit par la norme IEEE 802.3 qui a été internationalisée depuis en norme ISO 8802.3. Elle concerne des topologies physique en bus, en anneau ou en étoile mais qui seront traitées de manière logique comme des bus. Les accès au support physique sont gérés de manière compétitive avec écoute de la porteuse CSMA/CD. Plusieurs sous-normes décrivent les réseaux éthernet comme nous le montre le tableau 2.

Norme	Débit	Support	Longueur Max	Dénomination
802.3 10B5	10 Mbits/s	Coaxial 50W	500 m	Ethernet standard
802.3 10B2	10 Mbits/s	Coaxial 50W	200 m	Ethernet fin
802.3 10Broad36	10 Mbits/s	Coaxial 75W	3600 m	
802.3 1B5	1 Mbits/s	Paire torsadée	500m, 5 hubs	Starlan
802.3 10BT	10 Mbits/s	Paire torsadée	100m, hubs illimités	Starlan
802.3 10BF	10 Mbits/s	Fibre optique	2 km	Starlan
802.3 100BT	100 Mbits/s	Paire Torsadée		Starlan
802.3 100BF	100 Mbits/s	Fibre optique		Starlan
802.3 1000BT	1 Gbits/s	Paire Torsadée		Starlan
802.3 1000BF	1 Gbits/s	Fibre optique		Starlan

TAB. 2 – Liste des caractéristiques physiques des différents réseaux ethernet

Au niveau du matériel, nous trouvons le plus couramment l'Ethernet fin et les câblage Starlan avec paire torsadée.

L'Ethernet fin est composé de câbles coaxiaux de couleur généralement noire. L'adaptateur réseau (la carte réseau - fig 7(a)) est relié par un 'T' (fig 7(c)) sur lequel partent 2 câbles (fig 7(b)). Si un câble est une des extrémité du bus, celui-ci est terminé par un bouchon terminateur (fig 7(d)).

(a) Carte réseau BNC (b) Câble coaxiaux (BNC) (c) T-BNC (d) Bouchon terminateur

FIG. 7 – Éléments constitutifs d'un réseau Ethernet fin

Le câblage Starlan avec paire torsadée nécessite des appareils spécifiques au centre de l'étoile. Au cartes réseaux (fig 8(a)) employées sont branchés des câbles RJ45 (paires torsadée - fig 8(b)) qui sont ensuite connectés à des *hubs* (qui vont se charger de répliquer le signal reçu sur une des branches à toutes les branches de l'étoile - fig 8(c)) ou à des *switchs* (qui vont analyser le message pour le dupliquer sur la bonne branche menant à la machine avec laquelle on veut communiquer - fig 8(d)).

(a) Carte réseau RJ45 (b) Câble RJ45 (paire torsadée) (c) Hub (d) Switch

FIG. 8 – Éléments constitutifs d'un réseau de type Starlan sur paire torsadée

2.4.2 Le réseau Pronet-10

Ce réseau possède une topologie physique en anneau (ce n'est pas celle décrite dans la norme IEEE 802.5). La méthode pour contrôler l'accès au support physique se fait par élection. Il s'agit donc d'un réseau token-ring. Le câble est une paire torsadée. Ce type de réseau est limté à 254 machines.

2.4.3 Le réseau WiFi

WiFi, pour Wireless Fidelity, caractérise les réseaux sans-fil. Pour ceux-ci plusieurs sous normes de la norme IEEE 802.11 existent. Parmi ces dernières, seulement 3 concernent la constitution physique du réseau qui sont décrites dans le tableau ci-dessous.

Norme	Débit théorique	Fréquence	Canaux	Portée
802.11a	54 Mbits/s	5 GHz	8	10 m
802.11b	11 Mbits/s	2,4 GHz	13	100 m
802.11g	54 Mbits/s	2,4 GHz	13	100 m

Petite particularité, la norme 802.11a ne peut être utilisé en France qu'à l'intérieur puisque la bande passante de 5GHz est réservée à l'armée. De plus, les équipements 802.11a ne sont pas compatibles avec les équipements 802.11b/g.

Les réseaux WiFi peuvent fonctionner suivant deux modes différents : en mode *infrastructure* ou en mode *ad-hoc*.

En mode *infrastructure*, les cartes WiFi communiquent avec une borne d'accès (appelée aussi Access Point) qui gère les différentes communications. Il est possible de relier entre elles plusieurs bornes d'accès pour constituer un *système de distribution étendu*.

En mode *ad-hoc*, les cartes WiFi communiquent directement entre elles. Ceci permet à plusieurs personnes dans la même salle d'échanger rapidement des données. En revanche, la portée maximale de communication est plus faible. Une cellule de communication va temporairement être créée qui sera identifiée par ce que l'on appelle un BSSID.

Enfin, le débit des réseaux WiFi dépend de la distance entre les émetteurs et les récepteurs. Le tableau 3 donne le débit théorique qui peut être atteint suivant les différentes normes de WiFi en fonction de la distance séparant l'émetteur et le récepteur.

Débit (Mbits/s)	**Distance** (m) **intérieur**	**extérieur**
Norme 802.11a		
54	1	
48	17	
36	25	
24	30	
12	50	
6	70	
Norme 802.11b		
11	50	200
5.5	75	300
2	100	400
1	150	500
Norme 802.11g		
55	27	75
48	29	100
36	30	120
24	42	140
18	55	180
12	64	250
9	75	350
6	90	400

TAB. 3 – Portée maximale des réseaux wifi en fonction du débit escompté

Lorsque le réseau physique est constitué, il est possible aux différentes machines de communiquer entre elles. Ce sera la problématique de la couche 2 du modèle OSI : la couche de *liaison*.

3 La couche *Liaison* : communiquer sur le même support physique

Le rôle de cette couche est d'effectuer le partage du support physique qui a été déterminé dans la couche précédente. Chaque message (on parlera plus spécifiquement de trame) est émis sur le support physique et est écouté par toutes les machines.

Toutefois, Comme les machines écoutent tous les messages, ces dernières doivent pouvoir déterminer si ce message leur est adressé et qui est la machine émettrice (pour pouvoir éventuellement répondre).

La solution est alors de donner une adresse physique aux machines et d'incorporer les adresses physiques de source et de destination dans les trames échangées.

Nous allons à présent voir comment la problématique posée par la couche de liaison est résolue dans plusieurs cas particuliers de réseau :

- les réseaux de type Pronet-10,
- les réseaux de type Ethernet,
- les réseaux de type Wifi.

3.1 Le réseau Pronet-10

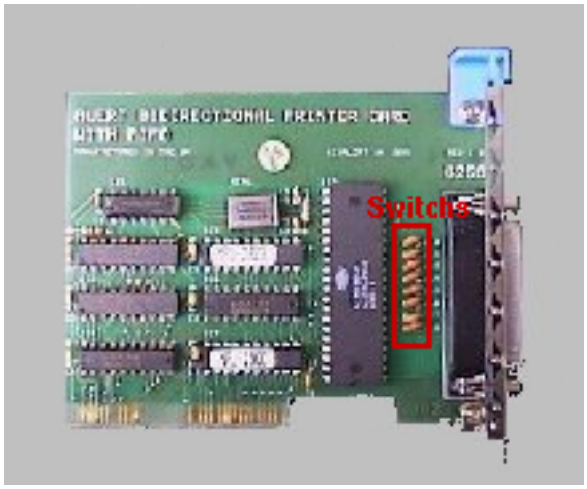

FIG. 9 – Adaptateur Pronet-10 avec l'ensemble de switchs nécessaires à configurer son adresse physique

Sur un réseau de type Pronet-10, l'adresse physique est codée sur un octet (la valeur est comprise entre 1 et 254). Cette adresse est choisie arbitrairement par l'administrateur réseau qui la donne en manipulant des petits interrupteurs (switchs) sur l'adaptateur réseau Pronet-10 comme on peut le voir sur la figure 9.

Les trames Pronet-10 font de 8 à 2052 octets. Elles sont composées de plusieurs éléments qui sont :

- Un **début de message** et une **fin de message** pour repérer les trames,
- Une **adresse de source** et une **adresse de destination**,
- Un bloc de **données**,
- Des structures de contrôle pour vérifier que le message a bien été acheminé : **parité** et **refus**.

Nous obtenons une trame telle que présentée à la figure 10.

Début de message 10 bits	Adresse destination 8 bits	Adresse source 8 bits	Type de trame 24 bits	Données 0 à 16352 bits	Fin de message 9 bits	Parité 1 bit	Refus 1 bit

FIG. 10 – Trame ProNet-10

3.2 Le réseau Ethernet

L'adresse physique des réseaux ethernet est directement codée dans la carte sans intervention de l'administrateur réseau. Cette dernière est aussi parfois appelée *adresse MAC* (MAC pour Media Access Control). Cette adresse physique est codée sur 48 bits ce qui représente 2.8×10^{14} adresses possibles. L'adresse physique d'une carte ethernet est donc unique au monde.

Les trames échangées, quant à elles, font entre 72 et 1526 octets environ. Elles comprennent :
- un **préambule** qui permet de repérer le début de la trame et d'indiquer sa longueur,
- Une **adresse de source** et une **adresse de destination**,
- Un bloc de **données**,
- Une structure de contrôle (**CRC**) qui vérifie que le message a été transmis sans erreur.

Nous obtenons une trame telle que présentée à la figure 11.

Préambule	Adresse destination	Adresse source	Type de trame	Données	CRC
64 bits	48 bits	48 bits	16 bits	368 à 12000 bits	32 bits

FIG. 11 – Trame Ethernet

3.3 Le réseau Wifi

L'adresse physique sur un réseau Wifi est similaire à celle employée sur un réseau de type Ethernet. Il s'agit également d'une adresse MAC codée sur 48 bits.

La trame Wifi se décompose ainsi :
- Un **contrôle de trame** qui est un en-tête indiquant le début du message,
- Un champ **durée/ID** qui indique la durée d'utilisation du canal de transmission,
- 4 champs d'adresse **Adresse 1**, **Adresse 2**, **Adresse 3** et **Adresse 4**,
- Un **contrôle de trame** qui permet d'identifier la trame émise dans une séquence de trames,
- Une structure de contrôle (**CRC**) qui vérifie que le message a été transmis sans erreur.

Nous obtenons une trame telle que présentée à la figure 12.

Contrôle de trame	Durée /ID	Adresse 1	Adresse 2	Adresse 3	Contrôle de séquence	Adresse 4	Corps de la trame	CRC
16 bits	16 bits	48 bits	48 bits	48 bits	16 bits	48 bits	0 à 18496 bits	32 bits

FIG. 12 – Trame Wifi

Si la trame wifi contient 4 adresses, c'est parce que plusieurs adresses peuvent être employées suivant le mode d'utilisation du réseau. En effet, 3 cas peuvent se présenter :
- la communication en mode ad-hoc,
- la communication en mode infrastructure avec le partage d'un seul point d'accès,
- la communication en mode infrastructure dans un système de distribution étendu où les points d'accès employés ne sont pas les mêmes pour les stations émettrices et réceptrices.

3.3.1 Le mode ad-hoc

Dans ce cas 3 adresses sont utilisées :

- Adresse 1 Adresse MAC de l'interface réceptrice,

- Adresse 2 Adresse MAC de l'interface émettrice,

- Adresse 3 BSSID de la cellule de communication temporaire.

3.3.2 Le mode infrastructure

Avec partage de la même borne d'accès

En réalité, deux trames vont transiter : une de l'interface émettrice (I1) à la borne d'accès (BA) puis de la borne d'accès à l'interface réceptrice (I2) chacune de ces trames va également employer 3 adresses. De plus, pour identifier le quelle étape de la transmission a été réalisée, chaque envoi de trame modifie deux champs spécifiques dans le contrôle de trame appelé **To DS** et **From DS**

- de I1 à BA (*To DS*=1) :
  - **- Adresse 1** Adresse MAC de BA,
  - **- Adresse 2** Adresse MAC de I1,
  - **- Adresse 3** Adresse MAC de I2.
- de BA à I2 (*From DS*=1) :
  - **- Adresse 1** Adresse MAC de I2,
  - **- Adresse 2** Adresse MAC de BA,
  - **- Adresse 3** Adresse MAC de I1.

Avec deux bornes d'accès différentes

Elles seront nommées BA1 et BA2. Ceci rajoute une étape supplémentaire au cas précédent : il faut en effet transmettre le message d'une borne d'accès à une autre. Les étapes sont alors les suivantes :

- de I1 à BA1 (*To DS*=1) :
  - **- Adresse 1** Adresse MAC de BA1,
  - **- Adresse 2** Adresse MAC de I1,
  - **- Adresse 3** Adresse MAC de I2.
- de BA1 à BA2 (*From DS*=1 et *To Ds*=1) :
  - **- Adresse 1** Adresse MAC de BA2,
  - **- Adresse 2** Adresse MAC de BA1,
  - **- Adresse 3** Adresse MAC de I2,
  - **- Adresse 4** Adresse MAC de I1,
- de BA2 à I2 (*From DS*=1) :
  - **- Adresse 1** Adresse MAC de I2,
  - **- Adresse 2** Adresse MAC de BA2,
  - **- Adresse 3** Adresse MAC de I1.

La couche de liaison nous permet de transmettre des messages au sein d'un même réseau physique. Toutefois, dans le cadre des réseaux de grandes échelle (comme Internet) toutes les machines ne sont pas situées sur le même réseau physique. En réalité, il va s'agir d'interconnexions de plusieurs réseaux physiques qui forment une réseau logique. Nous allons voir comment acheminer les messages d'une machine à une autre au travers de ces ensembles de réseaux. C'est le rôle de la couche 3 du modèle OSI : la couche réseau.

4 La couche *Réseau* : du réseau physique au réseau logique

Comme nous l'avons évoqué, la couche 3 du modèle OSI fait intervenir la notion de réseau logique, par opposition au réseau physique. Ainsi, les principales fonctionnalités qui sont traitées par cette couche sont :

– traduire les adresses physiques en adresse logiques et inversement,
– router les messages (c'est à dire les aiguiller d'un réseau physique à un autre) en fonction de leur priorité et de l'état du réseau,
– gérer le trafic sur le réseau,
– gérer la commutation des paquets sur le réseau,
– découper et réassembler les messages en fonction de la longueur maximale des trames des réseaux physiques par lesquels ils transitent.

4.1 Généralités

4.1.1 De la nécessité des adresses logiques

Lorsque nous avons abordé la couche 2 du modèle OSI, nous avons vu que chaque interface avait une adresse physique. De plus, dans le cas des adaptateurs ethernet couramment répandus, cette adresse est unique au monde. Dans ces conditions, on pourrait se demander quelle est l'utilité d'une adresse supplémentaire.

D'une part, tous les réseaux ne sont pas de type ethernet, d'autre part, une adresse MAC (physique) ne permettrait pas de localiser une machine dans le monde. En effet, les adresses MAC sont attribuées par les fabricants d'adaptateurs réseau qui vendent ensuite leurs cartes. Ainsi, 3 cartes qui ont des adresses MAC successives peuvent être vendues en Europe, aux Etats-Unis et au Japon. Dans ce cas de figure, il est vraisemblable que ces cartes soient sur des réseaux physiques séparés difficiles à localiser.

Ainsi, une adresse logique permet d'identifier un réseau et d'opérer un *routage hiérarchique* entre les réseaux. Une fois les réseaux identifiés, il est alors possible à deux machines situées sur des réseaux physiques différents de communiquer entre elles.

4.1.2 Lorsque le modèle OSI montre ses limites

Dans la couche 3, certaines des limitations du modèle OSI apparaissent. En effet, dans la théorie, la norme OSI garantit l'indépendance des couches mais, dans la réalité bon nombre des protocoles utilisés sont antérieurs à la norme OSI et ne la respectent pas complètement. De plus, un certain nombre de couches se révèlent interdépendantes.

Ainsi, à partir de la couche 3, les protocoles employés ont tendance à s'organiser en familles ou en suites. Parmi les *suites de protocoles* les plus utilisées, nous trouvons :
– La suite IPv4 (plus familièrement appelée IP) utilisée par internet composée des protocoles ARP, RARP, ICMP, ...
– La suite IPv6 qui devrait remplacer la précédente à long terme,
– La suite IPX crée par Novell et qui a été beaucoup utilisée pour les jeux en réseau,
– La suite NetBIOS employée dans les réseaux locaux Microsoft qui ne sera pas traitée dans ce cours.

4.2 Suite de protocoles IPv4

4.2.1 La naissance d'Internet et IPv4

IPv4 (baptisé aussi IP pour faire plus court) est le protocole utilisé par Internet. Aussi la naissance de cette famille de protocole est intimement liée à celle du réseau des réseaux.

Celui-ci a été développé à l'origine par l'armée américaine dans le but d'échanger rapidement des informations entre les bases militaires et ce même si une partie du réseau est détruite.

Ainsi le premier prototype baptisé ARPANET voit le jour en 1969. Celui-ci relie 4 stations situées chacune dans une base différente. Ce réseau continuera de croître et commencera à nécessiter des

protocoles rigoureux pour gérer ce dernier. Ainsi, en 1974 apparaissent les protocoles TCP et IP. Le succès de ce réseau continuera et il s'ouvrira aux université américaine. Une partie de ce réseau devient alors publique. C'est elle qui portera le nom d'Internet à partir du début des années 1980. Enfin, un organisme public est créé pour gérer Internet, lui conférant ses lettres de noblesse en même temps que son importance devient mondiale. C'est ainsi que l'Internet Society est née en 1992.

4.2.2 Brève présentation des protocoles IPv4 de couche 3

L'ensemble des protocoles de la famille IP est régie par des documents appelés *RFC* (Request For Comments). Ils constituent en quelque sorte une norme pour chaque protocole employé. Les RFC sont donc une série de documents techniques et organisationnels au sujet d'Internet. Ils sont considérés comme des standards. Ces RFC peuvent se trouver à plusieurs endroits dont :

- `http ://www.rfc-editor.org` qui est la liste complète en anglais,
- `http ://www.abcdrfc.free.fr` contient une traduction partielle en français des RFC.

Parmi les protocoles détaillés par ces RFC, nous trouverons :

- Le protocole IP qui s'occupe de l'adressage et de la fragmentation des paquets,
- Le protocole ARP qui permet de retrouver l'adresse physique à partir de l'adresse logique (ici appelée adresse IP),
- Le protocole RARP qui est l'inverse du protocole ARP,
- Les protocoles RIP et OSPF qui permettent le routage des paquets,
- Le protocole ICMP qui implémente la gestion d'erreurs et permet de tester la connectivité du réseau.

4.2.3 Adressage logique IPv4

Avant de détailler les divers protocoles de la couche 3, il convient de détailler l'élément de base sur lequel le travail va s'effectuer : l'adresse logique.

Cette dernière est codée sur 4 octets (32 bits) et est notée : $w.x.y.z$. Avec chaque élément compris entre 0 et 255. Le 0 peut être réservé pour les adresses de réseau et le 255 pour les adresses de *broadcast* appelées aussi *adresses de diffusion* (ces dernières permettent de contacter toutes les machines d'un même réseau). Comme nous l'avons vu, une adresse logique permet d'identifier le réseau sur lequel est connecté une machine. Ainsi, une adresse $w.x.y.z$ peut se lire de plusieurs manières :

- La machine d'adresse $w.x.y.z$,
- La machine d'adresse $x.y.z$ sur le réseau d'adresse $w.0.0.0$,
- La machine d'adresse $y.z$ sur le réseau d'adresse $w.x.0.0$,
- La machine d'adresse z sur le réseau d'adresse $w.x.y.0$.

Ceci induit un découpage hiérarchique entre les réseaux également puisque le réseau $w.x.0.0$ sera considéré comme étant un *sous-réseau* du réseau $w.0.0.0$. Ces différents lectures permettent de favoriser le routage des paquets d'un réseau à un autre.

Pour compléter cette présentation des adresses IP, nous devons mentionner les organismes qui les attribuent pour Internet, chaque adresse IP publique devant être unique au monde ! Deux organismes gèrent internet au niveau mondial :

- l'IANA (Internet Assigned Numbers/Naming Authority) : qui va distribuer les adresses IP aux FAI (Fournisseurs d'Accès à Internet),
- InterNIC (Internet Network Information Center) qui possède des antennes nationales (l'AFNIC en France - `http ://www.nic.fr`) et qui attribue des parties d'identifiant réseau pour les dispositifs directement reliées à Internet.

Classes d'adresses IP

L'adresse IP induit également une classification des réseaux (qui tend à disparaître) selon la valeur

du premier octet (w) de l'adresse. Cette classification est donnée dans le tableau 4. Pour chaque type de réseau est donné la longueur de son adresse réseau, ainsi que le nombre de réseaux existants de cette classe. Et enfin le nombre maximal de machine par réseau de chaque classe.

Classe	**Valeur de w**	**Lg adresse réseau**	**Nb de réseaux**	**Nb max de machines**
A	0 - 127	1 octet	127	16777216
B	128 - 191	2 octets	16384	65536
C	192 - 223	3 octets	2097152	256
D (multicast)	224 - 239			
E (expérimental)	240 - 255			

TAB. 4 – Classes de réseaux IP

De plus, certaines adresses sont réservées pour les réseaux locaux et les tests. Celles-ci ne sont pas routées par défaut sur internet. Les plages d'adresse réservées sont ainsi :
- 10.0.0.1 à 10.255.255.254,
- 172.16.0.1 à 172.31.255.254,
- 192.168.0.1 à 192.168.255.254,
- 127.0.0.1 à 127.255.255.254.

La dernière plage est utilisée pour les tests sur la machine (*localhost*) alors que les précédentes sont employées pour les réseaux locaux.

Notion de sous-réseau

Les grands réseaux sont fréquemment subdivisés en *sous-réseaux*. Ceci permet d'utiliser de manière hétérogène plusieurs type de réseaux physiques différents (token-ring et ethernet-STARLAN par exemple). Cela favorise la segmentation du réseau en vue de faciliter le routage des paquets (par effet de bord, ceci réduit aussi les temps de calcul sur les routeurs) et permet de réduire l'encombrement général du réseau. Ceci permet aussi d'isoler certaines parties du réseau peu fiables ou au contraire très sécurisées par rapport à un parc de machines standard. Enfin, ceci permet d'optimiser la plage d'adresse IP à utiliser.

A partir des adresses IP des réseaux, il est possible de définir des *masques de sous-réseau*. Ces derniers permettent de déterminer :
- le sous-réseau auquel appartient une machine,
- l'adresse de diffusion d'un sous-réseau donné,
- le nombre de machines d'un sous-réseau donné.

Un masque de sous-réseau se présente de la manière suivante :

Partie réseau hôte

255.255.255.224 → 11111111.11111111.11111111.11100000

La partie réseau est constituée de tous les bits mis à *1* alors que la partie hôte est constituée de tous les bits mis à *0*. En théorie, rien n'oblige les 1 à être contigus. Toutefois, dans la pratique, on "trie" les bits et on place ceux à 1 au début du masque et ceux à 0 en fin de masque.

Il est alors possible de calculer l'adresse de sous-réseau d'une machine donnée en effectuant un *ET logique* bit à bit entre l'adresse de la machine et le masque de sous-réseau (Ne pas oublier que chaque octet doit nécessairement contenir 8 bits). Ainsi, si on reprend le masque précèdent et on l'applique à l'adresse 200.100.40.33, on obtient :

$$\begin{array}{rcl} 200.100.40.33 & \rightarrow & 11001000.01100100.00101000.00100001 \\ ET & & 11111111.11111111.11111111.11100000 \\ 200.100.40.32 & \leftarrow & 11001000.01100100.00101000.00100000 \end{array}$$

L'adresse de diffusion (broadcast) d'un sous-réseau donné est obtenu en effectuant un *OU logique* entre l'adresse de la machine et l'inverse du masque de sous-réseau. Si l'on reprend l'exemple de tout à l'heure on a :

$$\begin{array}{ccc} 200.100.40.33 & \rightarrow & 11001000.01100100.00101000.00100001 \\ OU & & 00000000.00000000.00000000.00011111 \\ 200.100.40.63 & \leftarrow & 11001000.01100100.00101000.00111111 \end{array}$$

Le nombre de sous-réseaux possibles est compté en fonction du nombre de bits à 1 dans le masque. Si celui-ci est n alors le nombre de sous-réseaux possibles sera 2^n selon la RFC 1878 et $2^n - 2$ selon la RFC 1860. Enfin, le nombre de machines disponibles pour un réseau est donné en fonction du nombre de bits à 0 dans le masque. Si ce dernier est m alors le nombre maximal de machine sur le sous-réseau sera de $2^m - 2$.

Enfin, dans le but de simplifier la notation des masques de sous-réseau, la notation CIDR (pour Classless Inter Domain Routing) est introduite dans les RFC 1518 et 1519. Elle spécifie le nombre de bits à 1 utilisés par le masque. Ainsi l'exemple de tout à l'heure avec l'adresse de réseau 200.100.40.32 et le masque 255.255.255.224 est noté 200.100.40.32/27 puisque 27 bits sont à 1.

4.2.4 Format du datagramme IP

Le datagramme IPv4 est un datagramme aligné sur 32 bits comme on peut le voir à la figure 13.

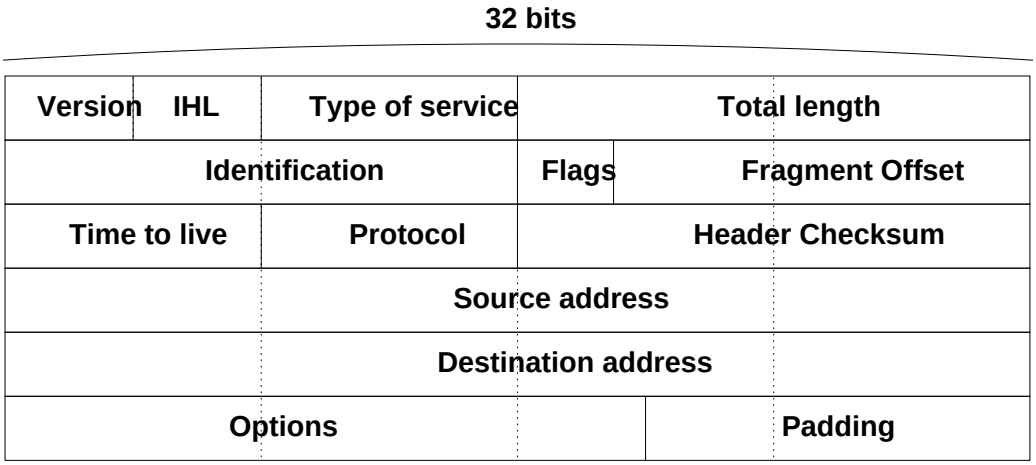

FIG. 13 – Format de l'en-tête d'un datagramme IPv4

Nous allons détailler le rôle des différents champs contenus dans le datagramme IP :

Version traduit la version du protocole IP utilisé. Celle-ci est codée sur 4 bits. Parmi ceux ci, les codes suivants sont attribués :
- 00 réservé,
- 04 IPv4,
- 05 ST datagram mode (inusité),
- 06 IPv6,
- 15 réservé.

IHL pour Internet Header Length est un champ codé sur 4 bits qui donne le nombre de mots de 32 bits constituant l'en-tête. Celui-ci fait entre 5 (le nombre par défaut) et 15 mots suivant les options employées,

Type of service est un champ codé sur 8 bits qui attribue une qualité de service liée au paquet. C'est à dire sa priorité, le délai, le débit et la fiabilité du paquet,

Total Length est un champ codée sur 16 bits qui indique la longueur du paquet en octets, en-tête inclus. Si dans la théorie le paquet peut faire au maximum 65535 octets, dans la pratique, la taille est d'environ 576 octets au plus,

Identification est une valeur donnée par la station émettrice en vue de reconstruire les fragments d'un message. Le détail du rôle de ce champ est expliqué dans la RFC 815,

Flags est un champ codé sur 3 bits qui gère également la fragmentation des messages, suivant la valeur de chaque bits du champ :
- Bit 0, valeur 0,
- Bit 1, 0 = may fragment, 1 = do not fragment. Celui-ci indique si le paquet peut-être fragmenté ou non,
- Bit 2, 0 = last fragment, 1 = more fragments. Celui-ci indique si des fragments doivent encore venir ou non.

Fragment offset (sur 13 bits) indique la place du fragment dans le paquet. Le premier fragment d'un paquet a ainsi un offset de '0',

Time to Live est aussi appelé en abrégé TTL. Codé sur 8 bits, ce champ indique la durée de vie du paquet. Cette dernière est exprimée en nombre de sauts de réseaux à réseaux. Chaque fois que le paquets change de réseau, le champ est décrémenté. Quand ce dernier tombe à 0, le paquet est alors détruit. Ceci permet de détruire des paquets qui ne peuvent pas être acheminé à leur destination.

Protocol est un identifiant codé sur 8 bits permettant de déterminer le protocole qui est encapsulé dans le datagramme IP. Parmi les valeurs les plus courantes nous avons :
- 01 : ICMP (que nous verrons plus loin),
- 06 : UDP (qui est un protocole de niveau 4),
- 17 : TCP (qui est un autre protocole de niveau 4).

Header Checksum est un code (sur 16 bits) permettant de contrôler la validité de l'en-tête du paquet. L'algorithme permettant d'obtenir ce code est détaillé dans la RFC 1071,

Source address est l'adresse IP de la machine qui a émis le datagramme,

Destination address est l'adresse IP de la machine à laquelle est adressé le datagramme IP,

Options décrit les options s'appliquant au datagramme IP que nous ne détaillerons pas ici,

Padding est un champ de *bourrage*. Celui-ci est constitué de '0' pour compléter l'en-tête jusqu'à ce que sa longueur soit un multiple de 4 octets.

Comme on peut le constater, le protocole IPv4 permet de fragmenter les paquets. Ceci est principalement dû au fait que changer de réseau implique aussi parfois changer de type de réseau physique. Dans ce cas, il se peut que la taille des trames échangées soient différentes. Il faut alors fragmenter le paquet pour que les fragments puissent tenir dans une trame. Lors de la réception du paquet, il faudra bien sûr reconstituer le paquet à partir des fragments.

Toutefois, dans le cas où les deux machines mises en communication sont sur le même réseau physique, il va falloir, dans la trame faire figurer l'adresse physique des machines de source et de destination. Si le problème est simple pour la machine source, il faut en revanche un mécanisme permettant de retrouver celle de la machine de destination à partir de son adresse IP. C'est le rôle du protocole ARP.

4.2.5 Les protocoles ARP et RARP

ARP, pour Adress Resolution Protocol, est un protocole qui permet de donner l'adresse physique d'une machine à partir de son adresse logique. Ce dernier est décrit par la RFC 826 qui décrit aussi les échanges qui s'effectuent alors sur le réseau. Ces derniers sont les suivants :

– la machine source émet une trame ARP sur le réseau physique qui est donc reçue par toutes les machines,
– la machine cible se reconnaît et répond à la machine source par une autre trame ARP,
– la machine source, ayant reçu la réponse, connaît alors l'adresse physique associée à l'adresse logique de la machine de destination ou machine cible.

Trame et requête ARP

Cette trame sera encapsulée dans une trame du réseau physique. Dans le cas d'une trame ethernet (voir figure 11 page 15), le champ type de trame sera positionné à la valeur `0x0806` pour indiquer que les données de la trame contiennent une trame ARP. De plus, la trame ARP sera structurée comme à la figure 14.

Type phys. 16 bits	Type log. 16 bits	Taille phys. 8 bits	Taille log. 8 bits	Op. 16 bits	Adresse physique émetteur	Adresse logique émetteur	Adresse physique récepteur	Adresse logique récepteur

FIG. 14 – Trame ARP

Cette trame indique les types de protocoles physiques et logiques ainsi que les tailles des adresses physiques et logique. Le champ **Op.** indique le type de la trame (requête ARP, réponse ARP, etc ...).

Dans un requête ARP sur un réseau éthernet, nous aurons ainsi :
– l'adresse de destination de la trame ethernet sera positionnée à `ff :ff :ff :ff :ff :ff` indiquant qu'elle concerne toutes les machines du réseau,
– le réseau éthernet sera signalé par un champ type physique positionné à `0x01`,
– le type logique sera IPv4. La valeur sera `0x0800`,
– la taille de l'adresse ethernet vaut 6,
– la taille de l'adresse IPv4 vaut 4,
– l'opération est un requête, le champ sera positionné à `0x01`,
– la seule adresse indéterminée est celle de l'adresse physique du récepteur. Le champ sera alors positionné à `00 :00 :00 :00 :00 :00`,

Pour récapituler, la trame ARP de requête aura alors la forme donnée à la figure 15 dans laquelle, nous avons substitué pour les premiers champs leur valeur.

01	0800	06	04	01	adresses...

FIG. 15 – Début d'une trame ARP de requête

Réponse ARP

La trame de réponse sera substantiellement la même. Les différences seront :
– la trame ethernet de transport contiendra l'adresse de source et de destination. Comme il s'agit de la réponse, l'ancienne adresse de source figure dans le champ de l'adresse de destination,
– le champ **Op.** a pour valeur `0x02`,
– les 4 adresses des champs adresses sont remplis.

Le rôle du cache ARP

Chaque échange de donnée présuppose que la machine source émette une trame ARP pour retrouver l'adresse physique de la machine de destination lorsque ces dernières sont sur le même réseau physique. Toutefois, dans le but de décongestionner le réseau, un mécanisme de cache est mis en

place afin que les machines se souviennent des correspondances entre adresses physiques et logiques. Dans ce cache sont stockées ces correspondances.

Toutefois, pour éviter les problèmes liés au changements de configuration des machines du réseau (changement de carte réseau lorsque celle-ci devient défectueuse par exemple), les données mises dans le cache ont une durée d'expiration de l'ordre de la minute. Ainsi, à l'issue de cette durée, une requête ARP doit à nouveau être émise.

Le protocole RARP

RARP signifie Rerverse-ARP. Il permet de déterminer à partir d'une adresse physique l'adresse logique qui lui correspond. Les mêmes trames que ARP sont employées. Le cache ARP sera également utilisé par RARP Les différences avec le protocole ARP sont principalement :

- dans le cas d'un réseau ethernet, la trame ethernet contiendra comme type de trame la valeur `0x0835`,
- la requête RARP mettra l'adresse logique de source à `0.0.0.0`, les adresses de destination physique et logique à `00 :00 :00 :00 :00 :00` et `0.0.0.0`,
- le champ **Op.** sera positionné à `0x03` pour les requêtes et `0x04` pour les réponses.

Si les protocoles ARP et RARP permettent à deux machines de s'échanger des informations sur le même réseau physique, le protocole Ip doit en revanche permettre à deux machines sur des réseaux physiques différents de communiquer entre elles. Ceci est assuré par la fonction que l'on appelle le *routage*.

4.2.6 Mécanismes de routage IPv4

Il s'agit de l'un des rôles essentiels de la couche 3 du modèle OSI : acheminer les paquets d'un réseau à un autre. Pour ce faire, les divers réseaux physiques seront reliés entre eux par des machines qui seront sur plusieurs réseaux à la fois : les *routeurs*. De plus, comme il est impossible de relier tous les réseaux physiques directement, les paquets doivent parfois passer par des réseaux physiques intermédiaires comme on peut le voir sur la figure 16 où les paquets, pour passer de la machine source sur le réseau (physique) 1 à la machine destination sur le réseau 3 doivent transiter par le réseau 2 ainsi que par les routeurs A et B.

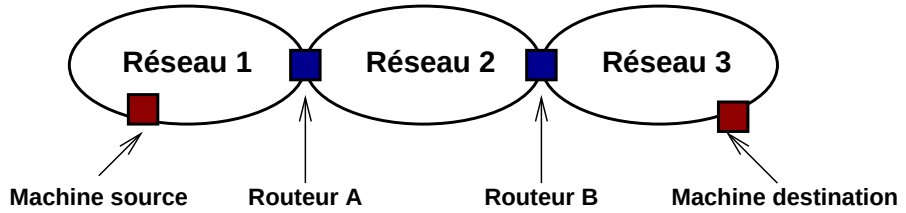

FIG. 16 – Exemple de routage au travers de plusieurs réseaux

Pour parvenir à effectuer ces fonctions de routage, il existe, sur chaque machine du réseau ce que l'on appelle une *table de routage*.

Les tables de routage

Une table de routage va contenir plusieurs *routes*. Ces routes sont des règles qui permettent de déterminer, à partir de la destination que l'on cherche à atteindre, le prochain routeur à contacter.

Une table de routage ressemblera à la figure 17 page suivante. Celle qui est représentée ici est un table de routage sur une machine de type unix.

```
Table de routage IP du noyau
Destination   Passerelle      Genmask       Indic Metric Ref Use Iface
192.168.42.0  195.221.158.121 255.255.255.0 U     0      0   0   eth0
195.221.158.0 *               255.255.255.0 U     0      0   0   eth0
127.0.0.0     *               255.0.0.0     U     0      0   0   lo
default       192.168.42.1    0.0.0.0       UG    0      0   0   eth0
```

FIG. 17 – Exemple de table de routage

La table de routage comprend, pour chaque règle, un ensemble de paramètres qui sont les suivants :

Destination indique une adresse de destination. Ce peut être un réseau ou une machine. La destination peut également prendre la valeur `default` qui correspond à une destination par défaut,

Passerelle (parfois aussi appelée *gateway*) indique l'adresse du routeur à joindre pour parvenir à la destination. Si ce paramètre est positionné à '$\star$' alors il n'y a pas besoin de transiter par un autre réseau,

Genmask donne le masque de réseau à appliquer pour déterminer si la destination est un réseau ou une machine,

Indic indique le statut de la route (U = up, la route est opérationnelle - G = gateway, pour la passerelle par défaut),

Metric donne le nombre de sauts pour joindre la destination,

Ref donne le nombre de références à la route (non utilisé sous Linux),

Use donne le nombre d'utilisation de la route,

Iface donne l'interface physique utilisée par la route. `ethX` désigne les interfaces ethernet, `lo` définit l'interface locale utilisée pour les tests (loopback).

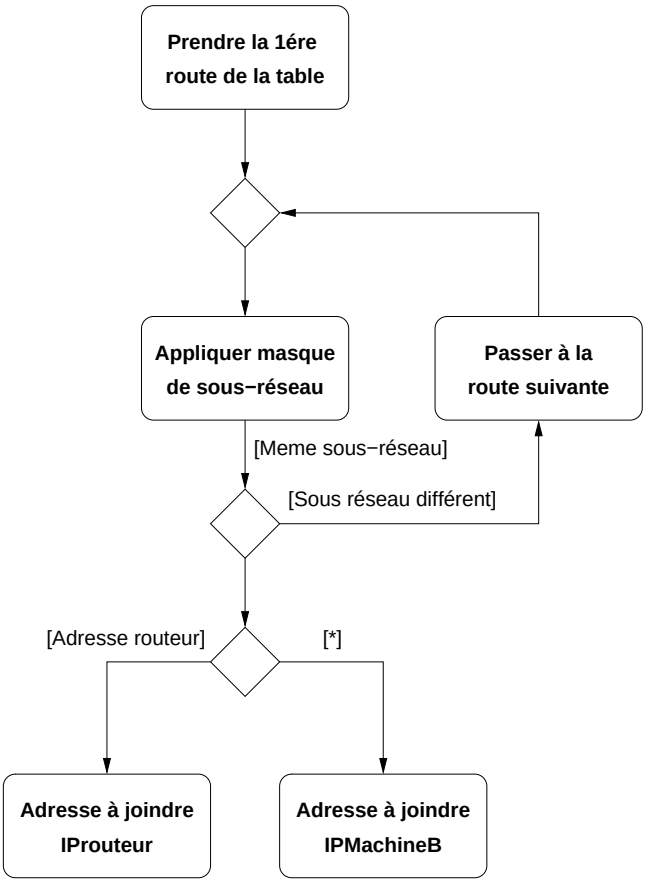

FIG. 18 – Algorithme exploitant la table de routage

A partir des règles définies dans la table de routage, il est alors possible de déterminer quelle machine est à contacter suivant la destination que l'on cherche à atteindre. Cet algorithme est décrit de manière succincte à la figure 18.

Si ce mécanisme est suffisant pour des réseaux locaux, ce dernier n'est pas adapté pour Internet. En effet, ce réseau des réseaux doit pouvoir acheminer des paquets d'un réseau à un autre même si certaines portions du réseau sont subitement coupées. Pour ce faire, les routeurs s'échangent des informations sur leurs tables de routage respectives via des *protocoles de routage*.

Les protocoles de routage

La problématique est de permettre trouver, dans cet enchevêtrement de réseaux, le chemin permettant de relier deux machines et, accessoirement, le plus court chemin possible entre les deux. De nos

RTA		
Dest	**Coût**	**Src**
a	0	*

RTB		
Dest	**Coût**	**Src**
b	0	*

RTC		
Dest	**Coût**	**Src**

(a) Étape 0

RTA		
Dest	**Coût**	**Src**
a	0	*
b	1	rtb

RTB		
Dest	**Coût**	**Src**
b	0	*
a	1	rta

RTC		
Dest	**Coût**	**Src**
b	1	rtb
a	1	rta

(b) Étape 1 : 1er échange

RTA		
Dest	**Coût**	**Src**
a	0	*
b	1	rtb
b	1	rtb
a	2	rtb
b	2	rtc
a	2	rtc

RTB		
Dest	**Coût**	**Src**
b	0	*
a	1	rta
a	1	rta
b	2	rta
b	2	rtc
a	2	rtc

RTC		
Dest	**Coût**	**Src**
b	1	rtb
a	1	rta
a	1	rta
b	2	rta
b	1	rtb
a	2	rtb

(c) Étape 2 : 2ème échange

RTA		
Dest	**Coût**	**Src**
a	0	*
b	1	rtb

RTB		
Dest	**Coût**	**Src**
b	0	*
a	1	rta

RTC		
Dest	**Coût**	**Src**
b	1	rtb
a	1	rta

(d) Étape 2 : suppression des routes redondantes

FIG. 19 – Vecteurs distance : exemple de mise à jour.

jours, deux protocoles sont principalement utilisés. Il s'agit de RIP (Routing Information Protocol) et de OSPF (Open Shortest Path First). Si le deuxième est plus performant que le premier, il est en revanche d'une complexité plus importante. Aussi, nous ne nous intéresserons qu'au premier protocole qui est encore très répandu bien que remplacé de plus en plus par le deuxième.

4.2.7 Le protocole RIP

RIP est défini par la RFC 1058 et fait l'objet d'une extension depuis 1994 définie dans la RFC 1723 : RIP2. Par le biais de ce protocole, chaque routeur échange les identificateurs des réseaux qu'il peut atteindre ainsi que la distance (en nombre de sauts de réseau à réseau) qui le sépare de ces réseaux.

Les mises à jour sont effectuées à intervalles de temps réguliers mais également lorsque la topologie du réseau a changé. Chaque mise à jour va être constituée d'échanges de *vecteurs-distance*.

Les vecteurs-distance et mise à jour des routes

Un vecteur-distance est composé de plusieurs informations :

- La **destination** : le réseau de destination,
- Le **coût** : le nombre de sauts pour parvenir à la destination (métrique),
- La **source** : l'identifiant du routeur source qui fait parvenir ce vecteur distance.

La mise à jour s'effectue de la manière suivante qui est également illustrée par la figure 19 :

- Au départ, chaque routeur ne connaît que son réseau direct (*),

- Le coût est, bien sûr, de 0 pour chaque réseau direct (figure 19(a)),
- Puis, les routeurs s'échangent les routes (des vecteurs-distance). Chaque route envoyée à un nouveau routeur voit son coût augmenté de 1 (étapes 1 et 2 des figures 19(b) et 19(c)),
- Enfin, les vecteurs-distance arrivés sont comparés à la table courante :
  - Si la destination est nouvelle, le vecteur-distance sera conservé,
  - Si la destination est existante, les vecteurs-distance seront remplacés si la source est la même o u si la source est différente mais le coût est meilleur,
  - Dans tous les autres cas, le vecteur-distance sera supprimé (voir les différences entre les figures 19(c) et 19(d)).

Mise en place du protocole RIP

Le protocole RIP améliore l'algorithme des vecteurs-distance en introduisant le *concept d'infinité*, les stratégies de type *Split horizon* et la gestion temporelle.

Le *concept d'infinité* traduit la notion de réseau inatteignable : on considère que ce dernier est à l'infini, sachant que l'infini est considéré comme étant à 16 sauts dans la pratique (coût de 16).

La stratégie *Split Horizon* consiste, pour un routeur, à ne pas renvoyer les routes reçues d'un autre routeur à ce dernier. Son implémentation est **obligatoire** dans le protocole RIP.

La stratégie *Split Horizon with Poisoned Reverse* consiste, quant à elle, à renvoyer les routes reçues d'un autre routeur à ce dernier en leur donnant un coût infini. Son implémentation est **recommandée** dans le protocole RIP.

La *gestion temporelle* s'effectue de deux manières différentes :

- La façon préconisée par les RFC :
  - le temps entre les mises à jour est de 30 secondes plus un petit délai aléatoire,
  - un délai d'expiration de 180 secondes est appliqué sur une route. Si pendant ce temps, elle n'est pas mise à jour, elle est marquée pour effaçage,
  - lorsqu'une route est marqué pour effaçage, elle est effacée 120 secondes après s'il n'y a toujours pas eu de mise à jour (étape de destruction).
- La façon CISCO (grand fabricant de routeurs) :
  - le temps entre les mises à jour est de 30 secondes plus un petit délai aléatoire,
  - une route est marquée invalide au bout de 180 secondes sans mise à jour,
  - une phase de rétention suit où les mises à jour sont refusées pendant 180 secondes pour une route invalide,
  - la destruction intervient 240 secondes après le marquage d'une route en invalidité.

Pour ce qui est des détails du protocole RIP, celui-ci envoie ses paquets par le biais du protocole UDP (un protocole de la couche 4 !) sur le port 520. Le protocole RIPv1 utilise le broadcast alors que le protocole RIPv2 utilise le multicast. La deuxième version du protocole implémente également la notion de masques de sous-réseaux ainsi qu'un mécanisme simple d'authentification.

Si ce protocole est simple à implémenter et consomme peu de bande passante sur les petits réseaux, il présente également comme inconvénients d'utiliser UDP, de posséder une infinité limitée à 16 et d'être gourmand en bande passante sur les grands réseaux, en particulier lorsque la topologie de ces derniers change.

Datagrammes RIP

Ces datagrammes sont alignés sur 32 bits tout comme les datagrammes IP. Les 4 premiers octets constituent l'en-tête des datagrammes. Le premier octet *commande* prend pour valeur 1 en cas de requête et 2 en cas de réponse. Dans le cas de RIPv2 le dernier champ (*Domaine de routage*) de l'en-tête est utilisé dans le cas où il y a plusieurs processus sur le routeur qui gèrent le protocole RIP.

Cet en-tête est suivi de paquets de 20 octets qui décrivent chaque vecteur distance reçu. Les figures 20 et 21 donnent la disposition des différents champs des datagrammes détaillant ainsi l'en-tête et les

champs associés à la première route échangée.

Commande	Version = 1	0
Famille d'adresses = 2		0
Adresse IP destination		
0		
0		
Métrique		

⋮

FIG. 20 – Datagramme RIPv1

Commande	Version = 1	Domaine de routage
Famille d'adresses = 2		Route tag
Adresse IP destination		
Masque de sous-réseau		
Adresse IP du saut suivant		
Métrique		

⋮

FIG. 21 – Datagramme RIPv2

4.2.8 Le protocole ICMP

ICMP (pour Internet Control Message Protocol) est l'un des derniers protocoles de la famille IPv4 à se situer en couche 3. Ce dernier gère les erreurs relatives au protocole IP et permet de tester la connectivité de certaines parties du réseau. Ce protocole peut être employé par la machine ou le routeur qui est à la source du problème pour signaler les incidents rencontrés. La RFC 792 préconise l'encapsulation des trames ICMP dans un datagramme IP. Ce dernier voit alors ses champs *Type of service* et *Protocol* positionnés respectivement à 0 et 1 (cf Figure 13 page 20).

Le datagramme ICMP est composé de 4 champs (comme on peut le voir à la figure 22) :
- Un champ *type* qui indique le type de message à faire transiter,
- Un champ *code* permettant de spécifier des informations supplémentaires,
- Un champ *CRC* qui permet de contrôler la validité du datagramme envoyé,
- Un champ *Message* qui sera plus ou moins long suivant la valeur du champ *type*.

Type (8 bits)	Code (8 bits)	CRC (16 bits)	Message (longueur variable)

FIG. 22 – Datagramme ICMP

Les valeurs du champ *type* et les messages correspondants sont consignés dans la table 5 page suivante.

L'outil *ping* :
Ping est un utilitaire installé sur toutes les machines utilisant la famille de protocoles IPv4. Ce dernier permet de contrôler la connectivité du réseau, en particulier si une machine donnée est connectée

Type	Contenu du message	Type	Contenu du message
0	Réponse echo (ping)	3	Destination non accessible
4	Contrôle de flux	5	Redirection
8	Echo (ping)	9	Avertissement routeur
10	Sollicitation routeur	11	Durée de vie écoulée
12	Erreur de paramètres	13	Demande d'horodatage
14	Réponse d'horodatage	15	Demande d'informations
16	Réponse à 15	17	Demande de masque d'adresse
18	Réponse à 17		

TAB. 5 – Messages ICMP envoyés en fonction de la valeur du champ *Type*

au réseau ou non. Ce programme s'appuie sur le protocole ICMP. Le trames envoyées correspondent à la représentation donnée à la figure 23. Les valeurs des deux premiers champs sont notés *A/B*, *A* étant la valeur du champ en requête, *B* étant la valeur du champ lors de la réponse.

8/0 (8 bits)	0/3 (8 bits)	CRC (16 bits)	Application ID (16 bits)	Numéro séquence (16 bits)

FIG. 23 – Datagramme ICMP associé aux requêtes et réponse *ping*

Nous avons ainsi évoqué l'ensemble des protocoles de la famille IPv4 qui composent la couche 3. Il existe toutefois d'autres familles de protocoles existantes dans cette couche qui correspondent à d'autres besoins. Nous allons à présent détailler le successeur de IPv4 qui est IPv6.

4.3 La famille IPv6

La famille IPv6 est apparue pour succéder à IPv4 qui date du début des années 80. La dernière RFC concernant IPv6 porte le numéro 2460. Le changement majeur concerne l'en-tête du datagramme IPv6 ainsi que la nouvelle forme prise par l'adressage logique.

4.3.1 Adressage IPv6

Les adresses IPv6 sont codées non plus sur 32 bits comme IPv4 mais sur 128 bits. Toutefois, sur ce large nombre, seulement 64 sont réservés pour l'adressage sur le réseau. Ceci permet de repousser les limites de la pénurie des adresses IP qui commence à arriver, les adresses IP publiques devant être uniques sur internet.

La notation employée pour les adresses IPv6 est la notation hexadécimale, chaque paquets de 2 octets étant séparés par des doubles points *' :'*. Ainsi, voici une adresse IPv6 valide :

```
3ffe:ffff:0100:f101:0210:a4ff:fee3:9566
```

Il faut ajouter que des blocs de 0 contigus peuvent être simplifiés en notant `::`. Cette notation n'est bien sûr utilisable qu'une seule fois dans une adresse IPv6 pour garantir l'unicité de la représentation.

Ainsi l'adresse *localhost* pour IPv6 sera notée

```
0000:0000:0000:0000:0000:0000:0000:0001
```

ou, plus simplement : `::1`.

Les adresses de type link-local

Il s'agit des adresses sur les réseaux locaux. Ces dernières sont préfixées `fe8x`, `fe9x`, `feax` et `febx`.

A Commandes réseau Windows XP

Cette annexe donne une liste non exhaustive de commandes réseau sous Windows XP. A chaque fois, les principaux paramètres des commandes sont présentés, mais ça ne veut pas dire que ce sont les seuls disponibles pour les commandes en question. Si, dans la syntaxe de la commande, un terme est en italique, cela veut dire qu'il faut le remplacer par une valeur.

A.1 La commande ipconfig

Cette commande affiche tous les paramètres de configuration liés au protocole IP. Elle permet aussi de manipuler le cache des adresses DNS et bien d'autres choses encore. La commande ipconfig employée sans paramètre affiche l'adresse IP, le masque de sous-réseau et la passerelle par défaut pour chaque interface réseau.

Syntaxe	ipconfig [/all]	
Paramètres	/all	affiche la configuration TCP/IP complète
	/ ?	affiche l'aide associée à la commande

A.2 La commande tracert

Cette commande permet de déterminer le chemin pris par les paquets entre deux machines reliées entre elles par divers embranchements réseaux. Pour déterminer ce chemin, cette commande emploie des paquets ICMP.

Syntaxe	tracert [-d] [-h *MaximumHops*] [-w *Timeout*] [*TargetName*]	
Paramètres	-d	empêche de résoudre le nom associé à une adresse IP
	-h *MaximumHops*	nombre maximum de sauts (hops) inter-réseaux
	-w *Timeout*	temps maximal en ms pendant lequel une réponse est attendue (défaut 4000 ms)
	TargetName	le nom ou l'adresse IP de la machine de destination
	- ?	affiche l'aide associée à la commande

A.3 La commande ping

Cette commande, basée sur le protocole ICMP permet d'effectuer des contrôle de connectivité du réseau. Utilisée sans paramètres, cette commande affiche un message d'aide

Syntaxe	ping *TargetName*	
Paramètres	*TargetName*	Le nom de la machine ou l'adresse IP à atteindre.
	/ ?	Affiche un message d'aide

A.4 La commande arp

Cette commande manipule le cache ARP de la machine locale. Utilisée sans paramètre, la commande arp affiche un message d'aide.

Syntaxe	arp [-a [*InetAddr*] [-N *IfaceAddr*]] [-d *InetAddr* [*IfaceAddr*]] [-s *InetAddr* *EtherAddr* [*IfaceAddr*]]	
Paramètres	-a [*InetAddr*] [-N *IfaceAddr*]]	Affiche le cache arp pour toutes les interfaces. Pour afficher l'entrée dans le cache arp d'une adresse spécifique, utiliser *arp -a* avec le paramètre *InetAddr* qui est une adresse IP. Pour afficher le cache arp d'une interface spécifique, utiliser *arp -a -N* avec le paramètre *IfaceAddr* qui est l'adresse IP associée à l'interface.
	-d *InetAddr* [*IfaceAddr*]	Efface une entrée du cache arp où *InetAddr* est l'adresse IP à retirer. Pour la retirer seulement pour une interface spécifique, ajouter le paramètre *IfaceAddr* qui est l'adresse IP associée à l'interface. *arp -d ** effacera toutes les entrées du cache.
	-s *InetAddr* *EtherAddr* [*IfaceAddr*]	Ajoute de manière statique une entrée dans le cache ARP.
	/ ?	Affiche un message d'aide.

A.5 La commande route

Cette commande manipule la table de routage de la machine locale. Utilisée sans paramètre, la commande route affiche un message d'aide.

Syntaxe	route [-f] [*Command* [*Destination*] [mask *Netmask*] [*Gateway*] [metric *Metric*]] [if *Interface*]]	
Paramètres	-f	Fait du nettoyage dans la table de routage.
	Command	Spécifie la commande à employer. Au choix : **add** Pour ajouter une route, **del** Pour effacer une route, **change** Pour modifier une route, **print** Pour afficher une route. La commande *route print* sans autres paramètres affiche la table de routage complète.
	Destination mask *Netmask* *Gateway* metric *Metric* if *Interface*	Adresse IP de destination. Donne le masque de sous-réseau Adresse IP de la passerelle (ou routeur) Donne le coût de la route Spécifie l'interface

A.6 La commande nslookup

Cette commande affiche des informations pour diagnostiquer les dysfonctionnements de l'infrastructure DNS. Utilisée sans paramètre, cette application se lance en mode interactif.

Syntaxe	nslookup *ComputerTofind*	
Paramètres	*ComputerToFind*	Nom ou adresse IP de la machine dont on veut les informations DNS.

B Commandes réseau Unix

Cette annexe donne une liste non exhaustive de commandes réseau sous Linux. A chaque fois, les principaux paramètres des commandes sont présentés, mais ça ne veut pas dire que ce sont les seuls disponibles pour les commandes en question. Si, dans la syntaxe de la commande, un terme est en italique, cela veut dire qu'il faut le remplacer par une valeur. Les commandes citées ici sont extraites des pages de *man*. Dans tous les cas, sous un système de type unix, un "*man*" de la commande ne fait pas de mal.

B.1 La commande route

route manipule les tables de routage IP du noyau. Son utilisation première consiste à configurer des routes statiques vers des hôtes ou des réseaux via une interface, après sa configuration par le programme *ifconfig*.

Avec les options *add* ou *del*, le programme route modifie les tables de routage. Sans ces options, il affiche simplement le contenu actuel des tables.

Syntaxe	route [-vn] route [-v] add [-net|-host] *cible* [netmask *Nm*] [gw *Gw*] [metric *N*] [reject] [[dev] *If*] route [-v] del [-net|-host] *cible* [gw *Gw*] [netmask *Nm*] [metric *N*] [[dev] *If*]	
Paramètres	-v	Active le mode verbeux.
	-n	Affiche les adresses sous forme numérique, au lieu d'essayer de déterminer les noms d'hôtes symboliques. C'est utile si vous désirez comprendre pourquoi la route vers votre serveur de noms a disparu.
	del	Supprime une route.
	add	Ajoute une route.
	cible	Le réseau ou l'hôte de destination, sous forme d'adresse IP en notation décimale pointée, ou sous forme de noms d'hôte ou de réseau.
	-net	La cible est un réseau.
	-host	La cible est un hôte.
	netmask *Nm*	Spécifie le masque de réseau de la route à ajouter.
	gw *Gw*	Route les paquets via une passerelle. La passerelle concernée doit pouvoir être atteinte. Ceci signifie qu'une route statique vers cette passerelle doit préalablement exister.
	metric *M*	Affecte la valeur M au champ métrique de la table de routage.
	reject	Installe une route bloquante, qui force l'échec d'une recherche de route. On l'utilise par exemple pour dissimuler des réseaux avant d'utiliser la route par défaut. Ce n'est pas une fonction de pare-feu.
	dev *If*	Force la route à être associée au périphérique spécifié, sinon le système tente de le déterminer par lui-même. Dans la plupart des réseaux normaux, vous n'en aurez pas besoin.

B.2 La commande ifconfig

ifconfig permet de configurer les interfaces de réseau présentes dans le noyau. On peut les configurer lors du démarrage quand c'est nécessaire. Ensuite, on l'utilise généralement pour le débogage ou pour d'éventuels réglages.

Si aucun argument n'est donné, *ifconfig* affiche l'état des interfaces actives. Si seul le paramètre interface est donné, il affiche seulement l'état de l'interface correspondante ; si seul le paramètre *-a* est fourni, il affiche l'état de toutes les interfaces, même celles qui sont inactives. Autrement, il permet de configurer une interface.

Syntaxe	ifconfig [-v] [-a] [-s] [*interface*] ifconfig [-v] *interface* [*aftype*] *options* | *adresse* ...	
Paramètres	-a	Affiche toutes les interfaces actuellement disponibles, même celles qui sont inactives.
	-v	Mode verbeux pour certains types d'erreurs.
	interface	Correspond au nom de l'interface de réseau. C'est généralement un nom de pilote suivi d'un chiffre, comme eth0 pour la première interface Ethernet.
	up	Active l'interface donnée. Cette option est implicite si une adresse est affectée à l'interface.
	down	Désactive le pilote pour l'interface donnée.
	metric *N*	Définit la métrique de l'interface.
	netmask *adresse*	Définit le masque de réseau IP pour cette interface.
	adresse	Correspond à l'adresse IP affectée à cette interface.

B.3 La commande arp

arp manipule la table ARP du noyau de différentes façons. Les options principales permettent d'effacer une correspondance d'adresses et d'en définir une manuellement. Pour les besoins de débogage, le programme *arp* permet aussi d'effectuer un dump complet de la table ARP.

Syntaxe	arp [-vn] [-i *if*] -a [*nomhôte*] arp [-v] [-i *if*] -d *nomhôte* [pub] arp [-v] [-i *if*] -s *nomhôte hwaddr* [temp|pub]] arp [-v] [-i *if*] -Ds *nomhôte ifa* pub arp [-vnD] [-i *if*] -f *nomfichier*	
Paramètres	-v	Active le mode verbeux.
	-n	Affiche les adresses sous forme numérique.
	-a [*nomhôte*]	Affiche les entrées concernant l'hôte spécifié. Si le paramètre *nomhôte* n'est pas utilisé, toutes les entrées seront affichées.
	-d *nomhôte*	Enlève une entrée pour l'hôte spécifié.
	-D	Utilise l'adresse matérielle de l'interface *ifa* .
	-i *If*	Sélectionne une interface. Lors du dump du cache ARP, seules les entrées correspondant à l'interface spécifiée seront affichées.
	-s *nomhôte hwaddr*	Crée manuellement une correspondance d'adresses ARP pour l'hôte *nomhôte* avec l'adresse matérielle positionnée à *hwaddr*.
	-f *nomfichier*	Similaire à l'option -s , mais cette fois les informations d'adresses sont prises dans le fichier *nomfichier*.

B.4 La commande ping

ping utilise le datagramme obligatoire ECHO_REQUEST du protocole ICMP pour requérir une réponse ICMP ECHO_RESPONSE d'un hôte ou d'une passerelle. Les datagrammes ECHO_REQUEST (« pings ») comportent les en-têtes IP et ICMP, suivis d'une « struct timeval » et d'un nombre arbitraire d'octets de bourrage utilisés pour remplir le paquet.

Syntaxe	ping [-bnqv] [-c *nombre*] [-i *intervalle*] [-t *ttl*] [-w *heure-limite*]	
Paramètres	-b	Permettre de “pinger” une adresse de diffusion (broadcast). (Pour les versions récentes de ping).
	-c *nombre*	S’arrêter après l’envoi de *nombre* paquets ECHO_REQUEST. Avec l’option *heure-limite*, ping attend jusqu’à *nombre* paquets ECHO_REPLY avant que la temporisation n’expire.
	-i *intervalle*	Attendre *intervalle* secondes entre chaque envoi de paquet. Le délai par défaut est normalement d’une seconde.
	-n	Affiche les adresses sous forme numérique.
	-q	Rien n’est affiché à part les lignes de résumé au démarrage et à la fin de l’exécution.
	-t *ttl*	Spécifier le champ IP Time to Live.
	-v	Active le mode verbeux.
	-w *heure-limite*	Spécifier un délai, en secondes, avant que ping ne se termine quel que soit le nombre de paquets envoyés ou reçus.

B.5 La commande traceroute

traceroute donne la route prise par les paquets pour joindre une machine ou un réseau distant à partir d’une machine hôte.

Syntaxe	traceroute [-Inv] [-i *iface*] [-m *max_ttl*] [-p *port*] [-w *waittime*] *host*	
Paramètres	-I	Utilise ICMP ECHO à la place des datagrammes UDP.
	-n	Affiche les adresses sous forme numérique.
	-v	Active le mode verbeux.
	-i *iface*	Spécifie une interface réseau pour obtenir l’adresse IP des paquets de sondage envoyé. Utile sur les systèmes à plusieurs interfaces.
	-m *max_ttl*	Donne le Time-to-live maximum pour les paquets envoyés (30 par défaut).
	-p *port*	Donne le port UDP de base (33434 par défaut).
	-w *waittime*	Donne le temps en secondes pendant lequel une réponse est attendue (5 secondes par défaut).
	host	Le nom où l’adresse de la machine à joindre.

B.6 La commande tcpdump

tcpdump affiche les informations relatives aux paquets qui transitent sur le réseau. L'affichage peut être conditionné pour n'afficher qu'un certain type de paquets via les options de la commande *tcpdump*.

Syntaxe	tcpdump [-dnpqStvxX] [-c *count*] [-i *interface*] [-w *file*] [*expression*]	
Paramètres	-c *count*	S'arrêter au bout de *count* paquets.
	-d	Affiche le paquet sous une forme lisible par un humain.
	-i *interface*	Ecouter sur une interface spécifique.
	-n	Affiche les adresses sous forme numérique.
	-p	Désactive le mode promiscuous. Ce mode permet d'afficher tous les paquets passant sur le réseau. Le désactiver revient à n'afficher que les paquets explicitement adressés à la machine locale.
	-q	Affiche les informations de manière concise.
	-S	Affiche les numéros de séquences TCP en mode absolu et non pas relatif.
	-t	N'affiche pas la date à laquelle le paquet a été réceptionné.
	-v	Active le mode verbeux.
	-w *file*	Ecrit les paquets dans un fichier.
	expression	Sélectionne les paquets à afficher en fonction de l'expression donnée. Se reporter à la page de man pour plus de détails.

B.7 La commande hostname

hostname est le programme utilisé a la fois pour fixer le nom d'hôte, et pour afficher les noms d'hôte ou de domaine du système. Ces noms sont utilisés par de nombreux programmes fonctionnant en réseau, afin d'identifier la machine. Quand il est invoqué sans arguments, ce programme affiche le nom actuel, précédemment fixe avec la commande hostname. Seul l'administrateur peut modifier le nom de la machine.

Syntaxe	hostname [-d] [-F *nom_de_fichier*] [-f] [-h] [-s] [-v] [*nom*]	
Paramètres	-d	Affiche le nom de domaine DNS si celui-ci existe.
	-F *nom_de_fichier*	Utilise le nom donné dans le fichier *nom_de_fichier*.
	-f	Affiche le nom complet de la machine avec le domaine DNS.
	-h	Affiche un message d'aide.
	-v	Active le mode verbeux.
	nom	Donne le nouveau nom d'hôte de la machine locale.

C Primitives de programmation réseau

C.1 Headers à inclure

Pour avoir accès aux primitives de programmation réseau, vos programmes incluront ces lignes :

```
#include <sys/types.h>
#include <sys/socket.h>
```

Pour la primitive `gethostbyname()`, il faudra rajouter en plus :

```
#include <netdb.h>
```

C.2 Les familles d'adresse

Il existe plusieurs familles d'adresses, chacune correspondant à un protocole particulier. Les familles les plus répandues sont :

- `AF_UNIX` : Protocoles internes de UNIX,
- `AF_INET` : Protocoles Internet,
- `AF_NS` : Protocoles de Xerox NS,
- `AF_IMPLINK` : Famille spéciale pour des applications particulières auxquelles nous ne nous intéresserons pas.

C.3 Les structures d'adresses

La structure d'une adresse de socket est la suivante :

```
struct in_addr {
u_long s_addr;
};

struct sockaddr_in {
u_short sin_family;/* famille d'adresses */
u_short sin_port; /* numéro de port */
struct in_addr sin_addr; /* adresse IP */
char sin_zero[8]; /* inutilisé */
};
```

Les champs de la structure ont pour valeur, plus précisément :

sin_family : prend la valeur `AF_INET`,

sin_port : numéro de port codé sur 16 bits,

sin_addr : adresse IP codée sur 32 bits,

sin_zero[8] : inutilisé.

C.4 L'appel système socket

Appel système		`int socket (int family, int type, int protocole);`
Paramètres	family	Peut prendre 4 valeurs : `AF_UNIX`, `AF_INET`, `AF_NS` et `AF_IMPLINK`
	type	Prend essentiellement 2 valeurs : `SOCK_STREAM` utilisé en mode connecté TCP `SOCK_DGRAM` utilisé en mode non-connecté UDP
	protocole	généralement mis à 0 sauf pour certaines applications spécifiques
Valeur renvoyée		L'appel système socket retourne un entier dont la fonction est similaire à celle d'un descripteur de fichier. Nous appellerons cette entier un descripteur de sockets (sockfd).

C.5 L'appel système bind

Il y a trois utilisations possibles de bind :

1. Le serveur enregistre sa propre adresse auprès du système. Il indique au système que tout message reçu pour cette adresse doit lui être fourni. Que la liaison soit avec ou sans connexion, l'appel de bind est nécessaire avant l'acceptation d'une requête d'un client.
2. Un client peut enregistrer une adresse spécifique pour lui même.
3. Un client sans connexion doit s'assurer que le système lui a affecté une unique adresse que ses correspondants utiliseront afin de lui envoyer des messages.

Appel système		`int bind (int sockfd, struct sockaddr *myaddr , int addrlen);`
Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	myaddr	Pointeur sur une structure d'adresse telle que décrite en C.3.
	addrlen	Taille de la structure d'adresse.
Valeur renvoyée		bind renvoie 0 s'il réussit, ou -1 s'il échoue.

bind fournit à la socket *sockfd*, l'adresse locale *my_addr*. C'est donc l'appel système **bind** qui remplit la structure *my_addr* passée en paramètre.

C.6 L'appel système connect

Un socket est initialement créé dans l'état non connecté, ce qui signifie qu'il n'est associé à aucune destination éloignée. L'appel système connect associe de façon permanente un socket à une destination éloignée et le place dans l'état connecté.

Un programme d'application doit invoquer connect pour établir une connexion avant de pouvoir transférer les données via un socket de transfert fiable en mode connecté.

Les sockets utilisées avec les services de transfert en mode datagramme n'ont pas besoin d'établir une connexion avant d'être utilisés, mais procéder de la sorte interdit de transférer des données sans mentionner à chaque fois, l'adresse de destination.

Appel système		`int connect (int sockfd, struct sockaddr *servaddr , int addrlen);`
Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	servaddr	Pointeur sur une structure d'adresse telle que décrite en C.3. Celle-ci indique l'adresse de destination avec laquelle la socket doit se connecter.
	addrlen	Taille de la structure d'adresse.
Valeur renvoyée		connect renvoie 0 s'il réussit, ou -1 s'il échoue.

C.7 L'appel système listen

Cet appel est généralement utilisé après les appels **socket** et **bind** et juste avant l'appel **accept**.

Appel système		`int listen ( int sockfd, int backlog);`
Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	backlog	spécifie le nombre de connections à établir dans une file d'attente par le système lorsque le serveur exécute l'appel accept. Cet argument est généralement mis à 5 qui est la valeur maximale utilisée.
Valeur renvoyée		listen renvoie 0 s'il réussit, ou -1 s'il échoue.

C.8 L'appel système accept

Appel système		`int accept (int sockfd, struct sockaddr *peer, int *addrlen);`
Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	peer	Pointeur vers une structure d'adresse de socket .
	addrlen	Longueur de la structure d'adresse *peer*.
Valeur renvoyée		accept renvoie -1 en cas d'erreur. S'il réussit il renvoie un entier non-négatif, constituant un descripteur pour la nouvelle socket.

Lorsqu'une requête arrive, le système enregistre l'adresse du client dans la structure **peer* et la longueur de l'adresse dans **addrlen*. Il crée alors un nouveau socket connecté avec la destination spécifiée par le client, et renvoie à l'appelant un descripteur de socket. Les échanges futurs avec ce client se feront donc par l'intermédiaire de ce socket. Le socket initiale *sockfd* n'a donc pas de destination et reste ouverte pour accepter de futures demandes.

Tant qu'il n'y a pas de connexions le serveur se bloque sur cette appel. Lorsqu'une demande de connexion arrive, l'appel système **accept** se termine. Le serveur peut gérer les demandes itérativement ou simultanément.

Dans l'approche itérative, le serveur traite lui même la requête, ferme le nouveau socket puis invoque de nouveau **accept** pour obtenir la demande suivante.

Dans l'approche parallèle, lorsque l'appel système accept se termine le serveur crée un serveur fils chargé de traiter la demande (appel de **fork** et **exec**). Lorsque le fils a terminer il ferme le socket et meurt. Le serveur maître ferme quand à lui la copie du nouveau socket après avoir exécuté le **fork**. Il appel ensuite de nouveau accept pour obtenir la demande suivante.

C.9 L'appel système gethostbyname

La fonction gethostbyname (rechercher une machine par son nom) accepte un nom de domaine et renvoie un pointeur vers une structure qui contient l'information relative à cette machine. Cet appel système renvoie en réalité les informations DNS associée à la machine.

Appel système		`struct hostent * gethostbyname (char *chaîne_nom);`
Paramètres	chaîne_nom	nom de domaine à traiter.
Valeur renvoyée		renvoie un pointeur vers une structure qui contient l'information relative à cette machine ou un pointeur `NULL` si une erreur se produit.

La structure `struct hostent` est de la forme :

```
struct hostent {
char *h_name ; /*nom officiel de la machine*/
char **h_aliases ; /*liste des surnoms de cette machine*/
int h_addrtype ; /*type d'adresse (exemple adresse IP) */
int h_length ; /*longueur de l'adresse */
char **h_addr_list ; /*liste des adresses de la machine
                       les routeurs peuvent en avoir plusieurs */
};
```

C.10 L'émission d'informations

Une fois que le programme d'application dispose d'un socket, il peut l'utiliser afin de transférer des données. Les trois appels système courant sont : **send**, **sendto** et **write**. **send** et **write** ne sont utilisables qu'avec des sockets en mode connecté car ils ne permettent pas d'indiquer d'adresse de destination.

Appel système		`int write(int sockfd, void *buf, int count);`
Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	buf	tampon mémoire dans lequel sont entreposées les données à écrire.
	count	nombre d'octets à lire dans *buf* pour les écrire dans *sockfd*.
Valeur renvoyée		write renvoie le nombre d'octets écrits (0 signifiant aucune écriture), ou -1 s'il échoue.

Appels système		`int send(int sockfd, void *msg, int len, int flags);` `int sendto(int sockfd, void *msg, int len, int flags, struct sockaddr *to, int tolen);`
Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	msg	tampon mémoire dans lequel est stocké le message.
	len	taille des informations à transférer.
	flags	combinaison de diverses valeurs à mettre à 0 par défaut.
	to	adresse de destination.
	tolen	taille de la structure stockant l'adresse de destination.
Valeur renvoyée		renvoient le nombre de caractères émis, ou -1 s'ils échouent.

C.11 La réception d'informations

Ici encore, trois appels systèmes sont courants : **read**, **recv** et **recvfrom**, les deux premiers étant utilisés pour le mode connecté et le dernier pour le mode non connecté.

Appels système		`int read ( int sockfd, char *buff, int nbytes );`
		`int recv (int sockfd, char *buff, int nbytes, int flags );`
		`int recvfrom (int sockfd, char *buff, int nbytes, int flags, struct sockaddr *from, int *addrlen);`

Paramètres	sockfd	Descripteur de socket retourné par l'appel socket effectué précédemment.
	buff	tampon mémoire dans lequel est stocké le message après lecture sur la socket.
	nbytes	taille maximale des informations à lire.
	flags	combinaison de diverses valeurs à mettre à 0 par défaut.
	from	adresse de la socket émettrice.
	addrlen	taille de la structure de l'adresse **from**.
Valeur renvoyée		renvoient le nombre de caractères lus, ou -1 s'ils échouent.

D Conversion binaire/décimal

Ce genre de conversion est essentiellement employé pour les masques de sous réseau. Pour rappel, 1 octet est composé de 8 bits. Chaque bit, suivant sa position dans l'octet, a une valeur décimale qui lui est affectée suivant la règle donnée dans le tableau ci-dessous :

Position	7	6	5	4	3	2	1	0
Valeur	128	64	32	16	8	4	2	1

En réalité, la valeur val peut s'obtenir par une formule générale en fonction de la position pos : $val = 2^{pos}$. Pour obtenir la valeur décimale de l'octet, il suffit alors d'additionner les valeurs des bits qui ont été positionnés à 1.

Dans le cadre des masques de sous-réseau, les possibilité sont réduites du fait que les bits positionnés à 1 doivent être contigus en partant de la gauche. Ceci nous laisse qu'une poignée de valeurs à connaître :

Octet	**Valeur**	
`00000000`	0	
`10000000`	128	128
`11000000`	192	128 + 64
`11100000`	224	128 + 64 + 32
`11110000`	240	128 + 64 + 32 + 16
`11111000`	248	128 + 64 + 32 + 16 + 8
`11111100`	252	128 + 64 + 32 + 16 + 8 + 4
`11111110`	253	128 + 64 + 32 + 16 + 8 + 4 + 2
`11111111`	255	128 + 64 + 32 + 16 + 8 + 4 + 2 + 1

La conversion inverse se fait en appliquant des divisions entière successives par 2 en notant les restes au fur et à mesure. Les restes, repris dans l'ordre inverse donnent la valeur en binaire du nombre décimal. Ainsi, pour 169, nous obtenons, en posant les division successives :

169	2							
1	84	2						
	0	42	2					
		0	21	2				
			1	10	2			
				0	5	2		
					1	2	2	
						0	1	2
							1	0

En reprenant les restes dans l'ordre inverse, nous pouvons écrire : $169_{10} = 10101001_2$. La conversion inverse, pour vérification, permet d'obtenir : $169 = 128 + 32 + 8 + 1$.